USER GUIDE

applied biosystems by life technologies™

LifeScope™ Genomic Analysis Software 2.5

Command Shell

DATA ANALYSIS METHODS AND INTERPRETATION

Publication Part Number 4471875 Rev. A
Revision Date November 2011

life technologies™

For Research Use Only. Not intended for any animal or human therapeutic or diagnostic use.

Information in this document is subject to change without notice.

LIFE TECHNOLOGIES AND/OR ITS AFFILIATE(S) DISCLAIM ALL WARRANTIES WITH RESPECT TO THIS DOCUMENT, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THOSE OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL LIFE TECHNOLOGIES AND/OR ITS AFFILIATE(S) BE LIABLE, WHETHER IN CONTRACT, TORT, WARRANTY, OR UNDER ANY STATUTE OR ON ANY OTHER BASIS FOR SPECIAL, INCIDENTAL, INDIRECT, PUNITIVE, MULTIPLE OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING FROM THIS DOCUMENT, INCLUDING BUT NOT LIMITED TO THE USE THEREOF.

LIMITED USE LABEL LICENSE

No right to resell this product or any of its components is conveyed expressly, by implication, or by estoppel. For information on obtaining additional rights, please contact outlicensing@lifetech.com or Out Licensing, Life Technologies, 5791 Van Allen Way, Carlsbad, California 92008.

NOTICE TO PURCHASER: DISCLAIMER OF LICENSE

Purchase of this software product alone does not imply any license under any process, instrument or other apparatus, system, composition, reagent or kit rights under patent claims owned or otherwise controlled by Applied Biosystems, either expressly, or by estoppel.

TRADEMARKS

The trademarks mentioned herein are the property of Life Technologies Corporation or their respective owners. Windows and Internet Explorer are registered trademarks of Microsoft Corporation in the United States and other countries. Mozilla is a trademark of Mozilla Foundation Corporation. Sage is a trademark of Genzyme Corporation.

© 2011 Life Technologies Corporation. All rights reserved.

Part Number 4471875 Rev. A
Document Revision Date November 2011

Contents

About This Guide 21

Purpose 21
Prerequisites 21
Contents of this guide 21
GUI guide 24

PART I Overview 25

CHAPTER 1 **Introduction to LifeScope™ Genomic Analysis Software 27**

LifeScope™ Software Overview 27
- Features 27
- Data analyses 27
- LifeScope™ Software Components 28

Workflows 28
- Standard workflow 28
- Project workflow 29
- Analysis workflows 29

Primary and secondary file types 34
Input and output file formats 34

PART II Installation 35

CHAPTER 2 **LifeScope™ Genomic Analysis Software Installation 37**

Overview 37
Prerequisites 37
- Workstation requirements 38
- Cluster hardware requirements 38
- Server requirements 38
- Check firewall restrictions 39
- Check DNS hostname 39
- Client machines 39
- Browser Requirements 39

LifeScope™ Software Administration 39
Installation workflow overview 41

Copy data drive content 41
Download LifeScope™ Software 42
Install LifeScope™ Software 42
Hardware configuration 42
Authentication realm 43
Installation instructions 44
Configure the license server 46
Continue installing LifeScope™ Software 46
Set up a workstation with remote submission 47
Configure the workstation 47
Activate the LifeScope™ Software License Key 48
Prerequisites 48
Internet connectivity 48
Check the computer date 48
Obtain the MAC address 49
Obtain the MAC Address with the LifeScope™ Installer 50
Activate License Key and obtain license file 51
Apply the License File to LifeScope™ Software 51
Set the LifeScope™ Software environment 52
Start LifeScope™ Software servers and check servers' status 53
Start the licensing server 53
Check the licensing server 53
Start the LifeScope™ Software server 54
Check the LifeScope™ Software server status 55
Download documentation and additional resources 55
Integrative Genomics Viewer (IGV) 55

CHAPTER 3 **Verify the LifeScope™ Genomic Analysis Software Installation . 57**

Overview 57
Workflow 58
Verify the installation 58
(*Optional*) Download and install demos files 59
(*Optional*) Run demo analyses 59
Check results 60
(Optional) Download and install performance verification scripts 60
(*Optional*) Run the performance verification test 60
Monitor progress 61
Check results 61
Create user accounts 61
Enable named users 62
Distribute the URL and user credentials 62
Maintenance 63
Stop the LifeScope™ Software server 63

Stop the licensing server 63
Upgrade 64
Uninstall 64

PART III Getting Started 65

CHAPTER 4 **Before You Begin 67**

Overview 67
Verify the browser version 68
Set the LifeScope™ Software environment 68
BioScope™ Software users' PATH variable 68
BioScope™ profile scripts 68
Clusters running both versions 68
Check LifeScope™ Software servers 69
Check the licensing server 69
Check the LifeScope™ Software server status 69

CHAPTER 5 **Understand The LifeScope™ Software Shell 71**

Overview 71
Terminology and concepts 72
Projects and analyses 74
Naming restrictions 76
Reads Data 76
Repositories 76
The reference repository 77
Analysis space 77
Find reference files for an analysis 78
Find data for an analysis 79
View analysis information 79
Common scenarios 80
Basic 80
A sample sequenced on multiple lanes 80
Data from multiple samples 80
Naming restrictions 81

CHAPTER 6 **Run a Command Shell Analysis 83**

Overview 83
Run a standard workflow 84
Example steps to run a standard workflow 84
Explanation of example steps 85
Use the command shell 85
Run mode 86

Status mode 87
Scripted mode 88
Shell mode 89
List shell commands 97
Options 97
Define input data 98
Example input data scenarios 98
Index IDs 100
Run a grouped analysis 101
Example of a groupsfile 101
Run a multi-group analysis 101
Run a tertiary-only workflow 103
With add bam commands 103
With a groupsfile 103
Run a multi-group tertiary analysis 104
Run an individual analysis 105
Resume, re-run, and re-use 106
Re-run an analysis in the command shell 106
Review job status 107
Review results 108
Review logs 108
Error behavior and error messages 108
Case sensitivity 108
Unrecognized commands 109
Illegal usage 109

CHAPTER 7 **Run a Standard Workflow Analysis 111**

Overview 111
Standard workflows 111
List standard workflows 115
List a workflow's INI files and parameters 116
Workflow directory structure 116
Workflow modules and parameters 117
Run a standard workflow 118
Prepare to run a standard workflow 118
Order commands correctly 118
Specify the regions of interest file 119
Run a workflow in the command shell 119
Run a workflow in lscope.sh run mode 121
Create a new workflow 121
With the Linux `cp` command 122
With the shell `get workflow` command 122
Workflow control files 123
PLN files 123

INI files 124
.run parameters 124
Run the tertiary portion of a workflow 125

PART IV Analysis Modules 127

CHAPTER 8 **Run the SOLiD® Accuracy Enhancement Tool 129**

Overview 129
SAET usage guidelines 129
Examples of how to run SAET 130
Workflows using SAET 130
SAET example in a workflow 130
SAET usage in standard workflows 130
SAET input files 131
SAET parameters 131
Explanation of parameters 133
SAET internal parameters 134
SAET output files 134
Algorithm description 135
SAET implementation 135
Phases 135
SAET run times 135

CHAPTER 9 **Run a Resequencing Mapping Analysis 137**

Overview 137
Examples of how to run a mapping analysis 138
In a standard workflow 138
As a demo analysis 139
Input files 139
Plan your input read-sets 139
Legacy data 139
Stages of mapping 140
Scatter 141
Fragment mapping 142
Mate-pair mapping 142
Pairing 142
Mark duplicates 144
BAM file generation 145
Quality alignments and unique alignments 148
Fragment mapping parameters 149
Fragment mapping parameters table 150
Mapping performance 152

Mapping algorithm 153
Internal mapping parameters 153
Mapping schemes 159
Pairing algorithm 163
Pairing parameters 163
Internal pairing parameters 163
Indel sizes 166
Mapping output files 167
Mapping statistics 167
Mapping statistics parameters 168
Summary of mapping statistics output 169
Mapping statistics output files 171
Mapping statistics example output 175
Run BAMStats standalone 178
FAQ – Mapping 179
FAQ – Pairing 182
FAQ – BAMStats 184

CHAPTER 10 **Run a SNPs Analysis 187**

Overview 187
SNPs input files 188
Examples of running a SNPs analysis 190
Examples in a standard workflow 190
As a demo analysis 190
SNPs runtime parameters 191
The call stringency parameter 196
The skip high coverage filter 197
The reads mapping QV parameter 197
The minimum base QV filters 197
Other filtering parameters 197
SNPs internal parameters 198
SNPs output files 198
GFF file format 198
GFF file example 200
Combined GFF for all contigs 201
Consensus_Calls file format 201
Consensus calls file example 204
Contig base space FASTA file 205
Combined FASTA for all contigs 205
Quartiles file 205
SNPs algorithm description 206
Frequentist algorithm 206
Bayesian algorithm 206
Data flow 207

Internal module flow 207
FAQ – SNPs 209

CHAPTER 11 **Run a Human CNVs Analysis 213**

Overview 213
Examples of running a CNV analysis 214
In a standard workflow 214
As a demo analysis 214
CNV module parameter descriptions 215
Human CNVs results file format description 218
OUT files 218
Wiggle file 219
GFF file 219
Human CNVs results file examples 220
Algorithm for the Human CNV module 221
Preprocessing 221
Coverage calculation 221
Sampling into windows 221
Normalization 222
Segmentation 223
Post processing 224
FAQ – Human CNVs 224

CHAPTER 12 **Run an Inversions Analysis 227**

Overview 227
Examples of running an inversion analysis 228
In a standard workflow 228
As a demo analysis 228
Inversion module parameters 228
Input files 230
Inversion output files 231
Inversion algorithm 234
Overview 234
Input data 236
Workflow 236
Scoring 237
Pairing 237
Ranking 238
Inversion length thresholds 238
Tiny inversions 238
Normal pair coverage 238

CHAPTER 13 **Run a Small Indels Analysis 239**

Overview 239
Examples of running a small indels analysis 240
In a standard workflow 240
As a demo analysis 240
Small indel parameter description 241
Small indel module output file formats 245
Small indel GFF format 246
Example output file 249
Small indel TXT and SQL formats 252
Small indel ALIGN format 254
UNGAPPED and PAS.SUM formats 255
Gap alignment mapping 256
Paired tag approach (paired libraries only) 257
Single tag approach (for both paired and fragment libraries) 259
Small indel calling from gap alignments 260
Pileup handling 261
Mapping quality adjustment (paired-end libraries only) 263
Color space considerations 263
Allele calling and sequence context determination 264
Indel size determination 267
Reference allele calling 268
Zygosity Calling 268
Sampling of gap alignments 269
Examples 269
FAQ – Small indels 272

CHAPTER 14 **Run a Large Indels Analysis 275**

Overview 275
Examples of running a large indel analysis 276
In a standard workflow 276
As a demo analysis 276
Large indel module parameters descriptions 276
Large indel internal parameters 278
Large indel output files 278
Large indel analysis algorithm 280
Identify candidate indels 281
Assign statistical significance to candidate indels 283
Determine zygosity 284
Filter alignments and parameter optimization 285
Input files for large indel analysis 286
Interpreting results from the large indel module 286
FAQ – Large indels 289

CHAPTER 15 **Add Genomic Annotations to Analysis Results 291**

Overview 291
Input file handling 292
Filters 292
Statistics 293
Workflows 295
Example shell commands 295
Annotation sources 296
Use custom reference and custom dbSNP files 297
The UCSC GTF file 297
The Ensembl GTF file 297
The dbSNPs files 298
Annotation input files 299
GFFv3 variant file 299
GTF 299
dbSNP 299
Prepare to run annotation processing 300
Select the required input files 300
Set your memory requests 301
Complete the prerequisites 301
Annotation parameters 301
Output annotation 303
Annotation output files 307
Annotated Variants and Filtered Variants output files 308
Variant Statistics output file for SNPs 309
Variant Statistics output file for small indels 311
Variant Statistics output file for large indels 314
Variant Statistics output file for CNVs 316
Mutated Genes output file 317
diBayes tab-delimited output file 319
diBayes annotated tab-delimited output file 321
diBayes filtered annotated tab-delimited output file 321
About annotations and LifeScope™ Software modules 322
FAQ - Annotations 323

PART V Targeted Resequencing and Low Frequency Variants 325

CHAPTER 16 **Run Targeted Resequencing Analyses 327**

Overview 327
Example of running targeted resequencing analyses 328
Targeted resequencing read selection algorithm 329

Run targeted resequencing modules 330

CHAPTER 17 **Run Low Frequency Variant Analyses 333**

Overview 333
Examples of running low frequency variant analyses 334
Low frequency variant read selection algorithm 335
SNPs optimizations 336
Run low frequency variant modules 338

CHAPTER 18 **Run Enrichment Analyses 339**

Overview 339
Enrichment 339
Enrichment statistics 340
Examples of running the enrichment module 340
In a standard workflow 340
As a demo analysis 341
Enrichment parameters 341
Enrichment statistics input files 343
The input target regions file 343
The input aligned reads file 344
Enrichment statistics output files 344
The output target file 345
The output alignment file 345
The target coverage BEDGRAPH file 345
The genome coverage frequency file 346
The target coverage frequency file 347
The target statistics report file 347
The enrichment statistics report 348

PART VI Whole Transcriptome Analyses 351

CHAPTER 19 **Run a Whole Transcriptome Mapping Analysis 353**

Overview 353
Examples of running the WT mapping module 355
In a standard workflow 355
As a demo analysis 356
Input files 356
Reads input files 356
Reference input files 357
Annotations input files 357
Stages of mapping 359

Scatter . . . 361
Single-read mapping . . . 362
Filter mapping . . . 362
Junction mapping . . . 363
Exon mapping . . . 363
Rescue method . . . 363
Pairing reads . . . 365
BAM file generation . . . 366
Quality alignments and unique alignments . . . 368
Single-read mapping parameters . . . 369
Mapping performance . . . 371
WT mapping internal parameters . . . 373
Mapping schemes . . . 374
Paired-end parameters . . . 375
Mapping output files . . . 380
Overview . . . 380
BAM file differences . . . 381
WT filtering stats . . . 382
Mapping statistics . . . 383
Mapping statistics parameters . . . 383
Summary of mapping statistics output . . . 384
Mapping statistics output files . . . 386
Mapping statistics example output . . . 390
Run BAMStats standalone . . . 392
FAQ – Whole transcriptome . . . 393
FAQ – BAMStats . . . 396

CHAPTER 20 **Run a WT Coverage Analysis . . . 399**

Overview . . . 399
WT coverage input files . . . 399
WT coverage parameters . . . 400
Coverage output files . . . 401

CHAPTER 21 **Run a WT Count Known Genes and Exons Analysis . . . 403**

Overview . . . 403
Input files . . . 404
WT counts parameter description . . . 404
WT counts algorithm description . . . 405
Filters . . . 405
Counts . . . 406
RPKM . . . 407
Output files . . . 407
GTF output . . . 407

Tab output 407
FAQ – WT counts 408

CHAPTER 22 **Run a WT Splice Finder Analysis 409**

Overview 409
Splice finder parameters 410
SASR splice finder 413
SASR splice finder module summary 413
SASR splice finder module evidence evaluator 414
Call junctions and fusions with single read only 415
Output files 416
Tabular junction files 416
Junction examples 418
Browser Extensible Display (BED) output 420
SEQ 421
Circos 421
Output format values 421

PART VII Small RNA Analyses 423

CHAPTER 23 **Run a Small RNA Mapping Analysis 425**

Overview 425
Example of running the small RNA mapping module 426
Small RNA mapping input files 427
Stages of mapping 428
Scatter 429
Filter map 429
miRBase map 430
Genome map 430
miRBase2Genome 430
BAM file generation 430
Quality alignments and unique alignments 432
Small RNA mapping parameters 433
Mapping performance 436
Internal parameters 437
Mapping schemes 439
Small RNA mapping output files 440
Mapping statistics 441
Mapping statistics parameters 441
Summary of mapping statistics output 443
Mapping statistics output files 444
Mapping statistics example output 447

Run BAMStats standalone 449
FAQ - Small RNA mapping 450
FAQ – BAMStats 454

CHAPTER 24 **Run a Small RNA Coverage Analysis 457**

Overview 457
Example of running the small RNA coverage module 457
Small RNA coverage input files 458
Small RNA coverage parameters 458
Small RNA coverage output files 459

CHAPTER 25 **Run a Small RNA Counts Analysis 461**

Overview 461
Example of running the small RNA count module 461
Small RNA counts 462
Small RNA counts input files 462
Alignments files 463
Precursor sequences 463
Mature form sequences 463
Small RNA counts parameters 464
Mapped file parameters 464
Filtered BAM file parameters 465
Small RNA counts output files 465
Mapped output 465
Filtered counts output 467

PART VIII Additional Mapping Analyses 469

CHAPTER 26 **Run a ChIP-Seq Mapping Analysis 471**

Overview 471
Run ChIP-Seq mapping 471
Run in the lscope command shell 471
Do not rename 473
Turn off SAET 473
Run ChIP-Seq as an individual analysis 474
Mapping algorithm 474
Use results files 474

CHAPTER 27 **Run a MethylMiner™ Mapping Analysis 477**

Overview 477

Run MethylMiner™ mapping 478
Run MethylMiner™ workflows 479
Run in the lscope command shell 479
Do not rename 480
Visualize MethylMiner™ results 481
MethylMiner™ mapping output files 481
Mapping algorithm 481
Further analysis of MethylMiner™ mapping results 481

PART IX Appendices 483

APPENDIX A **FAQ 485**

FAQ lists for LifeScope™ Genomic Analysis Software modules 485
General FAQ for LifeScope™ Software 485

APPENDIX B **Troubleshooting 489**

Overview 489
Logs 489
Troubleshooting 490
Login issues 490
Analysis failures 490
The server responds with an error 490
Message: Read-only db connection 490
Sequence name not found in reference 490
Memory errors 491
Special characters in BAM header fields 491

APPENDIX C **File Format Descriptions 493**

Introduction 493
XSQ file format 494
XSQ file content overview 494
Relation to sequencing instrument 495
XSQ file format description 495
BAM header usage 498
Sequence dictionary (@SQ) 498
Read group (@RG) 498
Header (@HD) sort order 499
BAM file validation 499
XSQ metadata in BAM headers 500
Color-space attributes 504
Pairing information in a BAM file 505

Calculation of tag names . . . 505
Proper pairs . . . 505
Single read mapping quality . . . 506
Hard clipping of incomplete extensions . . . 506
Visualize BAM output . . . 507
Integrative Genomics View (IGV) . . . 507
UC Santa Cruz (UCSC) genome browser . . . 507
Indel alignments . . . 508
BED file format . . . 510
BEDGRAPH file format . . . 511
Reference file data overview . . . 512
Contig multi-fasta file . . . 512
Single contig FASTA file . . . 512
CMAP file . . . 512
GTF file . . . 512
VCF file . . . 513
Reference sequence data validation . . . 513
Concatenation . . . 513
Select a reference file . . . 513
Read-set file format . . . 513
Legacy format translation . . . 515
Legacy CMAP file format description . . . 516

APPENDIX D **XSQ Tools . . . 519**

Overview . . . 519
The XSQ Tools package . . . 520
Package components . . . 520
Download instructions . . . 521
Conversion to the XSQ format . . . 521
Conversion from the XSQ format . . . 525
Conversion syntax . . . 525
Filter option . . . 525
Options table . . . 526
XSQ file splitting . . . 526
XSQ index reassignment . . . 526
Purpose . . . 526
Usage . . . 527
Index reassignment options table . . . 527
HDF5 tools . . . 527
Internal conversion parameters . . . 528
Parameters table . . . 528
Internal INI file . . . 530
Links to resources . . . 533

FAQ - XSQ Tools 533

APPENDIX E **The Reference Repository 539**

Overview 539
Repository structure 539
Repository location 540
Reference file conflicts 540
Assembly names and reference parameters 540
Initial repository contents 541
Add new reference files 547
Add by assembly name 547
Example steps to add a new assembly 547
Prepare reference files 550
Validate new reference files 550
Convert a GTF file 550

APPENDIX F **Command Shell Control Files 553**

Overview 553
Configuration files 554
PLN files 554
INI files 555
Properties files 557
Property files vs. INI files 558
Analysis module XML files 558
Set up your own analysis run 558
LifeScope™ Software parameters 559
General parameters 559
Reference parameters 559
Execution control parameters 560
Read-only parameters 560
(*Optional*) The Linux `mail` command 561

APPENDIX G **Administration 563**

Overview 563
Admin tasks 563
User management 563
Scratch space cleanup 564
Reference file conflicts 564
Reset the admin password 564
Read-set repository path: notify users 565
License management 565
Admin interfaces 565

GUI Admin Portal 565
Shell admin commands 565
Other information 566

APPENDIX H **Demo Analyses 567**

Overview 567
List of demo modules 567
Demos location 568
How a demo is setup 568
Workflow 569
Input files 569
Run all demos 569
Run a single demo analysis 570
Output files and logs 570

APPENDIX I **LIFESCOPE™ GENOMIC ANALYSIS SOFTWARE v2 . . . . . 571**

END USER LICENSE AGREEMENT 571

Glossary 583

Documentation 597

Related documentation 597
Obtaining support 597

Index 599

About This Guide

Purpose

This guide is designed to help you quickly perform next-generation sequencing analyses using LifeScope™ Software. These analyses support fragment, paired-end (PE) and long mate pair (LMP) library types of analyses. Specifically, the following workflows are supported:

- Whole genome resequencing
- Targeted resequencing
- Low frequency variant resequencing
- Whole transcriptome resequencing
- Small RNA resequencing
- ChIP-Seq mapping
- MethylMiner™ mapping

Prerequisites

It is assumed that you have working knowledge of the:

- Linux® operating system
- Internet Protocol (IP) address of the LifeScope™ Software cluster.
- Linux environment and know how to:
  - Navigate to directories.
  - Edit and save files in a text editor.
  - Run Linux shell scripts.
  - Run basic Linux commands such as `chmod`, `ps`, `pwd`, `cd`, `echo`, `grep`, and other commands.

Contents of this guide

The chapters and appendices in this guide are described in the following table.

Chapter	Title	Description
Part I Overview		
1	"Introduction to LifeScope™ Genomic Analysis Software"	A high-level description of LifeScope™ Software, including features, workflows, and file formats.
Part II Installation		

Chapter	Title	Description
2	"LifeScope™ Genomic Analysis Software Installation"	For the LifeScope™ Software administrator, describes the LifeScope™ Software installation procedure, and describes system software and hardware requirements.
3	"Verify the LifeScope™ Genomic Analysis Software Installation"	For the LifeScope™ Software administrator, describes the LifeScope™ Software installation procedure, and describes system software and hardware requirements.
Part III Getting Started		
4	"Before You Begin"	Describes general prerequisites and provides procedures to complete after installation and before you run LifeScope™ Software.
5	"Understand The LifeScope™ Software Shell"	Describes the new concepts and terminology required when working with LifeScope™ Software command shell.
6	"Run a Command Shell Analysis"	Lists and explains the LifeScope™ Software shell commands, and describes how to run analyses and workflows in the shell.
7	"Run a Standard Workflow Analysis"	Lists and describes the standard workflows available with LifeScope™ Software.
Part IV Analysis Modules		
8	"Run the SOLiD® Accuracy Enhancement Tool"	Describes using SAET to improve color call accuracy prior to mapping.
9	"Run a Resequencing Mapping Analysis"	Describes secondary processing, including mapping, pairing, BAM generation, mapping statistics, parameters, and algorithms.
10	"Run a SNPs Analysis"	Describes the Single Nucleotide Polymorphisms analysis module, including command shell examples, parameters, and algorithms.
11	"Run a Human CNVs Analysis"	Describes the Human Copy Number Variation analysis module, including command shell examples, parameters, and algorithms.
12	"Run an Inversions Analysis"	Describes the inversions analysis module, including command shell examples, parameters, and algorithms.
13	"Run a Small Indels Analysis"	Describes the small indel analysis module, including command shell examples, parameters, and algorithms.
14	"Run a Large Indels Analysis"	Describes the large indel analysis module, including command shell examples, parameters, and algorithms.
15	"Add Genomic Annotations to Analysis Results"	Describes the genomic annotations analysis module, including command shell examples, parameters, annotation sources, and reports.
Part V Targeted Resequencing and Low Frequency Variants		
16	"Run Targeted Resequencing Analyses"	Describes targeted resequencing analyses within LifeScope™ Software.
17	"Run Low Frequency Variant Analyses"	Describes low frequency variant detection analyses within LifeScope™ Software.
18	"Run Enrichment Analyses"	Describes the enrichment module used with targeted resequencing and low frequency variant workflows, including command shell examples, parameters, and reports.
Part VI Whole Transcriptome Analyses		

Chapter	Title	Description
19	"Run a Whole Transcriptome Mapping Analysis"	Describes WTA secondary processing, including mapping, pairing, BAM generation, mapping statistics, mapping and pairing parameters, and mapping and pairing algorithms.
20	"Run a WT Coverage Analysis"	Describes the coverage module, which calculates read coverage per position.
21	"Run a WT Count Known Genes and Exons Analysis"	Describes the counts module, including command shell examples, parameters, and algorithms. This module counts the number of reads that align within genomic features, and counts and annotates the number of exons that are expressed in a gene region.
22	"Run a WT Splice Finder Analysis"	Describes the splice finder module, including parameters and algorithms. This module discovers splice junctions (introns from pre-messenger-RNAs to generate mature messenger-RNAs) and annotates the type and evidence for each discovered junction.
Part VII Small RNA Analyses		
23	"Run a Small RNA Mapping Analysis"	Describes small RNA secondary processing, including mapping, pairing, BAM generation, mapping statistics, parameters, and algorithms.
24	"Run a Small RNA Coverage Analysis"	Describes the coverage module, which calculates read coverage per position.
25	"Run a Small RNA Counts Analysis"	Describes the counts module, including command shell examples and parameters.
Part VIII Additional Analyses		
26	"Run a ChIP-Seq Mapping Analysis"	Describes using LifeScope™ Software to map data and create an output file type compatible with a variety of third-party Chromatin Immunoprecipitation Sequencing (ChIP-Seq) data analysis tools.
27	"Run a MethylMiner™ Mapping Analysis"	Describes using LifeScope™ Software for focused evaluation of methylation patterns in genome-wide studies (with the use of 5500 or SOLiD® System sequencing data).
Part IX Appendices		
A	"FAQ"	Lists frequently-asked questions, and answers, for LifeScope™ Software. as a whole. FAQs for individual modules are located in the module chapters.
B	"Troubleshooting"	Describes LifeScope™ Software log files and tips for dealing with certain errors.
C	"File Format Descriptions"	Describes LifeScope™ Software's use of BAM and XSQ files and their fields; XSQ, BED, and BEDGRAPH file formats; an overview of reference files; clipping issues; and how to visualize BAM output.
D	"XSQ Tools"	Describes XSQ conversion and manipulation tools available in the XSQ Tools package (a separate download from LifeScope™ Software).
E	"The Reference Repository"	Describes the LifeScope™ Software reference repository, including the repository structure, its initial contents (from the separate data drive), the importance and use of assembly names, adding a new species assembly, converting reference files, and reference file conflicts.

Chapter	Title	Description
F	"Command Shell Control Files"	Describes the control files used with the command shell (INI, PLN, properties, and XML files), and LifeScope™ Software framework parameters.
G	"Administration"	Describes the admin functionality available in the command shell. The GUI Admin Portal, not the command shell, is recommended for admin tasks. Refer to the *LifeScope™ Genomic Analysis Software User Guide* (Part no. 4471877).
H	"Demo Analyses"	Describes the example demos, which are used for installation verification.
I	"LIFESCOPE™ GENOMIC ANALYSIS SOFTWARE v2" EULA	The LifeScope™ Software End User License agreement.
—	Glossary	Explains select terminology used with the 5500 Series Genetic Analysis Systems, SOLiD® Systems, LifeScope™ Software, genomic sequencing, and bioinformatics.
—	Documentation	Lists other documentation related to LifeScope™ Software.
—	Index	Index to this guide.

GUI guide

See the companion document *LifeScope™ Genomic Analysis Software User Guide* (Part. No. 4471877) for instructions to use the LifeScope™ Software graphical user interface (GUI).

The GUI guide also covers these topics that are not available in the command shell or are available with limited functionality:

- Appendix D, "Administration"
- Appendix E, "Convert 1.3 BAM Files"

PART I
Overview

1 Introduction to LifeScope™ Genomic Analysis Software

This introduction covers:

- LifeScope™ Software Overview 27
- Workflows 28
- Primary and secondary file types 34
- Input and output file formats 34

LifeScope™ Software Overview

After years of development on analysis tools for SOLiD® data, in response to customer feedback, Applied Biosystems LifeScope™ Genomic Analysis Software enables fast translation of next-generation data for biologically meaningful results. LifeScope™ Software matches the accuracy of the next generation 5500 Series Genetic Analysis Systems with Exact Call Chemistry (ECC) and streamlines your data analysis.

Features

LifeScope™ Software is part of the LifeScope™ Genomic Analysis Solution. The complete informatics solution is comprised of genomics analysis software combined with a specified hardware platform. LifeScope™ Software features:

- Seamless integration with the 5500 Series Genetic Analysis Systems
- Performance-tuned algorithms for the 5500 Series and ECC module
- Push-button workflows, intuitive user interface, and secure project management
- Optimized mapping and smaller file formats
- Annotated variant reports, numerous charts, and select visualization tools for simple data interpretation
- Graphically driven configuration of multistep analysis workflows
- Ability to save and reuse workflows
- Ability to resume a workflow without repeating completed analyses
- Secure project-based data management specific to your data analysis
- Projects can be stored and data reanalyzed

Data analyses

Complementing LifeScope™ Software features are the following types of data analyses:

- Whole genome sequencing
- Targeted resequencing
- Low frequency variant
- Whole exome sequencing
- Whole transcriptome RNA sequencing

- Small RNA sequencing
- MethylMiner™ mapping
- CHiP-Seq mapping
- Detection of wide range of genomic and RNA variation, including:
  - SNP detection
  - Large and small indel detection
  - Copy number variation detection
  - Inversion detection
  - Fusion transcript detection
  - Exon counting
  - Splice finding

LifeScope™ Software Components

LifeScope™ Software components include:

- LifeScope™ Server Software, the main software architecture that maintains the interaction of the graphical user interface (GUI) and compute engine. The server works with high performance cluster schedulers to maximize the computation demands of the finely tuned data analysis algorithms.
- The research GUI, which enables scientists to perform mapping and the detection of genomic variation from the convenience of a desktop computer, is comprised of a compute engine capable of mapping billions of reads of next-generation sequencing (NGS) data. The GUI provides auto-generated charts and plots for each genomic analysis run within the software.
- The command-line user interface, which gives bioinformaticians the power to customize workflows, and to manipulate every parameter available in each analysis module using the specific LifeScope™ command-line syntax.
- The Admin GUI Portal, which is an administration tool for managing user accounts and licensing permissions. Refer to *LifeScope™ Genomic Analysis Software User Guide* (Part no. 4471877) for information on the Admin Portal.

Workflows

This section describes a standard workflow, a project workflow, and several analysis workflows.

Standard workflow

A standard workflow is a built-in series of commonly used analyses, which correspond to a common biological application. Using one of these workflows enables you to run an analysis with a minimum of setup required.

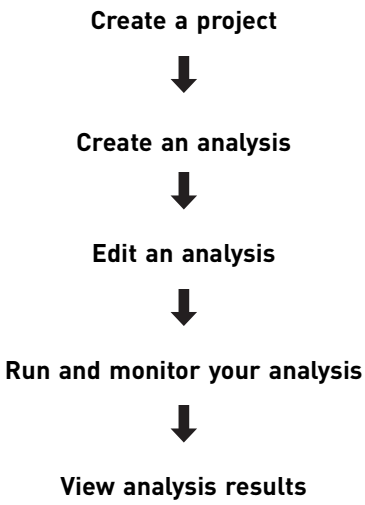
Standard workflows are provided for each of the analysis types listed above, on page 27.

In LifeScope™ Software, running a standard workflow requires the following steps:

1. Create a project and an analysis.
2. Identify your input data (read set from your 5500 Series sequencing instrument).
3. Identify the reference genome.

4. Choose the workflow to be executed on your data (from the above list).
5. Start and monitor your analysis.
6. View the results of your analysis

Project workflow



Analysis workflows

In LifeScope™ Software, a workflow is a coordinated series of data analysis steps, with each step dependent only upon the steps that precede it. This section describes the flow of secondary and tertiary analyses in extensive detail, while primary analysis is performed on your 5500 Series sequencing instrument.

Secondary analysis

Secondary analysis begins with mapping. The input files required by mapping modules are eXtensible SeQuence files (XSQ) or converted color-space fasta (CSFASTA) files, and quality value (QV) files. Secondary analysis modules include:

Table 1 Secondary analysis modules

SOLiD® Accuracy Enhancement Tool (SAET)
Mapping
Pairing
Binary Sequence Alignment Map statistics (BAMStats)
Mark duplicates

Tertiary analysis

The results of mapping and pairing in secondary analysis are used as input for tertiary analysis modules, which include:

Table 2 Tertiary analysis modules

SNP	Enrichment
CNV	Genomic annotations
Small indel	Gene and exon fusion

Table 2 Tertiary analysis modules

Large indel	Gene and exon count
Inversion	

The analysis workflows and the modules they execute are described in the following table.

Table 3 LifeScope™ Software analysis workflows and modules

Workflow	Library type	LifeScope™ Software modules involved
ChIP-Seq	Fragment	Secondary: • Fragment mapping • Mapping statistics
Genomic resequencing	Fragment	Secondary: • Mapping • Mapping statistics
		Tertiary: • SNP Finding • CNV • Small indels • Annotations
	Mate-pair	Secondary: • Paired mapping • Mapping statistics • Mark duplicates
		Tertiary: • SNP Finding • CNV • Inversions • Large indels • Small indels • Annotations
	Paired-end	Secondary: • Paired mapping • Mapping statistics • Mark duplicates
		Tertiary: • SNP Finding • Inversions • Large indels • CNV • Small indels • Annotations

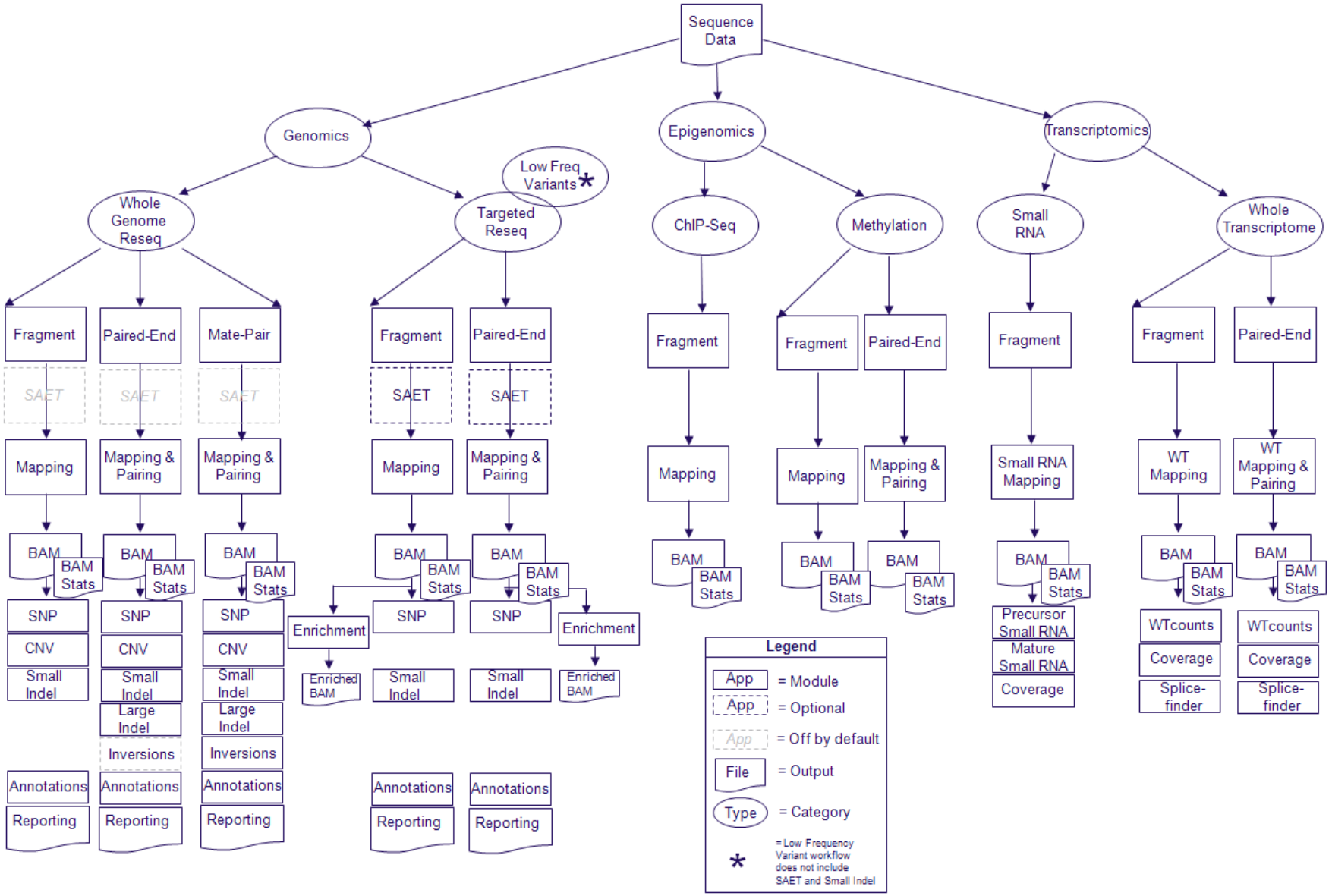
Table 3 LifeScope™ Software analysis workflows and modules *(continued)*

Workflow	Library type	LifeScope™ Software modules involved
MethylMiner™	Fragment	Secondary: • Paired mapping • Mapping statistics
	Paired-end	Secondary: • Fragment mapping • Mapping statistics • Mark duplicates
Small RNA	Small RNA	Secondary: • Small RNA mapping
		Tertiary: • Small RNA count • Small RNA coverage
Targeted resequencing	Fragment	Secondary: • SAET • Fragment mapping • Mapping statistics
		Tertiary: • Enrichment • SNP Finding • Small indels • Annotations
	Paired-end	Secondary: • SAET • Paired mapping • Mapping statistics • Mark duplicates
		Tertiary: • Enrichment • SNP Finding • Small indels • Annotations

Table 3 LifeScope™ Software analysis workflows and modules *(continued)*

Workflow	Library type	LifeScope™ Software modules involved
Low frequency variant	Fragment	Secondary: • Fragment mapping • Mapping statistics
		Tertiary: • Enrichment • SNP Finding • Annotations
	Paired-end	Secondary: • Paired mapping • Mapping statistics • Mark duplicates
		Tertiary: • Enrichment • SNP Finding • Annotations
Whole transcriptome	Fragment	Secondary: • WT splice junction extractor • WT fragment mapping • Mapping statistics
		Tertiary: • WT counts • WT coverage • Splice finder
	Paired-end	Secondary: • WT exon sequence extractor • WT splice junction extractor • WT paired-end mapping • Mapping statistics
		Tertiary: • Splice finder • WT counts • WT coverage

The illustration below shows the data analysis modules used in predetermined workflows.



Primary and secondary file types

The following table lists the files used in secondary and tertiary analysis.

Table 4 File types used in LifeScope™ Software analyses

Analysis type	File type	File name extension	File content	Used in . . .
Primary	Raw reads file	*.xsq	Extensible sequence file (XSQ)	Secondary analysis
Secondary	BAM file	*.bam	Binary Alignment sequence Map (BAM), a generic file format used to store large numbers of nucleotide sequence alignments	Tertiary analysis

Input and output file formats

All analysis modules use specific input files and produce specific output files. The input and output file requirements vary, depending on the type of analysis that you want to perform. The following table provides the names of the input and output file types.

Table 5 Input and output file formats

Software or bioinformatics tool	Input file types	Output file type
Mapping tool	XSQ	BAM
MaToBAM tool	MA	Converts a MA file to a BAM file.
Mapping statistics (BAMStats)	BAM	Position error, probe error, multiple chart and statistics files
Small indel	BAM	GFFv3
diBayes	BAM	GFFv3, CSFASTA, consensus_calls
CNV	BAM	GFFv3, *.out
Large indel	BAM	GFFv3
Inversion	BAM	GFFv3, text
WT mapping	XSQ, FASTA, filter reference FASTA, WT GTF reference	BAM
WT counts	BAM	GTF
WT coverage	BAM	WIG
WT splice junction	BAM	WIG
Small RNA mapping	XSQ, GTF reference	BAM
Small RNA counts	BAM	GTF
Small RNA coverage	BAM	WIG

PART II
Installation

2 LifeScope™ Genomic Analysis Software Installation

This chapter covers:

- Overview 37
- Prerequisites 37
- LifeScope™ Software Administration 39
- Installation workflow overview 41
- Copy data drive content 41
- Download LifeScope™ Software 42
- Install LifeScope™ Software 42
- Set up a workstation with remote submission 47
- Activate the LifeScope™ Software License Key 48
- Apply the License File to LifeScope™ Software 51
- Set the LifeScope™ Software environment 52
- Start LifeScope™ Software servers and check servers' status 53
- Download documentation and additional resources 55

Overview

This chapter is intended for use by the LifeScope™ Software administrator. The chapter describes the LifeScope™ Genomic Analysis Software installation procedure, and system software and hardware requirements.

Prerequisites

Some procedures in this chapter require that you:

- Know the Linux® operating system of the cluster on which LifeScope™ Software is to be installed
- Know the Internet Protocol (IP) address of the cluster
- Have a login (ID) on the cluster

  Portions of the install require root access or your system administrator's assistance
- Know how to:
  - Navigate to directories in a Linux environment
  - Edit and save files in a text editor

- Run Linux shell scripts
- Run basic Linux commands such as `chmod`, `ps`, `pwd`, `cd`, `echo`, `grep`, and other commands

Workstation requirements

Workstations are shipped with CentOS 5.7 and are ready to run LifeScope™ Software. Customer upgrade of the CentOS software is not supported.

WARNING! LifeScope™ Software does not operate properly on workstations running operating systems other than CentOS 5.7. CentOS 6.0 is not supported.

Cluster hardware requirements

To successfully install and run the LifeScope™ Software application, we recommend that your cluster meets the hardware requirements listed in Table 6.

Table 6 Cluster requirements

CPU Speed	2 GHz Minimum
Cores	8 Minimum
Memory	24 GB Minimum per compute node
Disk Space	500 GB of local or shared storage per node
	200 MB for LifeScope™ Software installation
	225 GB for reference files installation‡
	2 GB for examples installation
	25 GB for performance verification data

‡ You can use any type of shared storage to meet this requirement.

Note: Distributed parallel file systems such as Lustre, GlusterFS, GPFS require additional tuning to be compatible with LifeScope™ Software. For example, we have encountered instances of job failures caused by subpar small file performance.

Server requirements

Before you install LifeScope™ Genomic Analysis Software, be sure that your cluster meets the following requirements:

- LifeScope™ Genomic Analysis Software supports only the following distributions, and only on 64-bit platforms:
  - Red Hat Enterprise Linux 4.7 through 5.7
  - CentOS 4.7 through 5.7

    Note: LifeScope™ Software does not support CentOS 6.0 or RHEL 6.0.
- PBS/TORQUE v2.3+, SGE v6.2+, or Platform LSF 7 Update 6.

  Note: LifeScope™ Software requires a job submission queue on the cluster. The LifeScope™ Software installer does not create this queue on the cluster.

  Note: If you use SGE, you must create a symmetric multiprocessing (SMP) parallel environment.

  Note: If you use LSF, it is highly recommended that LifeScope™ Software jobs are non-preemptable by LSF.

Before you install LifeScope™ Software on the head node, pre-install compatible versions of these software packages on *all* compute nodes:

- Python 2.3 (or later version)
- Perl v.5.8.5 (or later version)

  Note: Inversion module works with Perl versions up through 5.8.8. The most recent version of Perl has a backwards compatibility issue which causes the inversion module to fail.

GCC Compiler

Ensure that the GNU Compiler Collection (GCC) compiler v3.4.6 or v4.1.2 installed and configured. While LifeScope™ Software does not use the compiler itself, having the compiler installed and configured ensures that the necessary dynamically linked libraries are available.

Check firewall restrictions

Ensure that there is no firewall restriction on the host running the LifeScope™ Software server. The LifeScope™ Software accesses the LifeScope™ Software license server on TCP/IP port 27000, by default. The LifeScope™ Software UI clients and command shell clients access the LifeScope™ Software server on TCP/IP port 9998 by default, although this port can be changed during the installation. Consult your system administrator to check the firewall access for LifeScope™ Software users.

Check DNS hostname

The DNS fully-qualified domain name and the local Linux hostname must match. Also, the forward and reverse lookup on the DNS against the hostname and its public-facing IP address must be consistent. The LifeScope™ Software licensing server may not work correctly if there are DNS discrepancies. Consult the system administrator to check this requirement.

Client machines

Java v1.6 (or higher) is required on client machines.

For Mac users who do have Java v1.6, follow these steps to update:

1. Use the Mac Software Update (**apple ▸ Software Update...**) to install.
2. Once you have installed, go **Applications ▸ Utilities ▸ Java Preferences**.
3. Select only Java SE 6.0.

Browser Requirements

LifeScope™ Software supports these browsers:

- Internet Explorer® versions 6 and 7
- Mozilla® Firefox 3.0.1

LifeScope™ Software Administration

This section uses the acronyms in this list to describe installation roles and locations:

- **SA –** The Linux system administrator for the cluster. The administrator's root access is required for user management and copying the data drive.

- **LSA –** The LifeScope™ Software administrator. The installation instructions recommend creating a new user account for the LSA. The LSA must run the installation script.
- **LSU –** LifeScope™ Software users. These users do not have admin permissions.
- **LSBF –** The LifeScope™ Software Binaries Folder. The LSBF is the installation directory.
- **LSDF –** The LifeScope™ Software Data Folder. The LSDF is a location for LifeScope™ Software repositories, resources, and other files.

To install or run LifeScope™ Software, one person must be designated as the master LifeScope™ Software user. The master user also becomes the default LifeScope™ Software Administrator (LSA). The LSA and regular LifeScope™ Software Users (LSUs) must share a common UNIX® user group.

We recommend that the LSA not share the same user account as the UNIX System Administrator (SA), because of a requirement that all LSUs be members of the LSA's primary UNIX user group. The SA's primary UNIX user group is usually `root`. Non-administrator users typically are not allowed to be members of the `root` user group.

We recommend creating a new Linux user account, `lifescope`, for the LSA. The primary UNIX user group for the `lifescope` user should be the same group as most LifeScope™ Software Users (for example, the group `users`). The Linux account for the LSA should have this shared group as its primary group when the LSA starts the LifeScope™ Software server.

After installation, LifeScope™ Software files and folders are owned by the LSA. All application jobs and processes are submitted by the LSA.

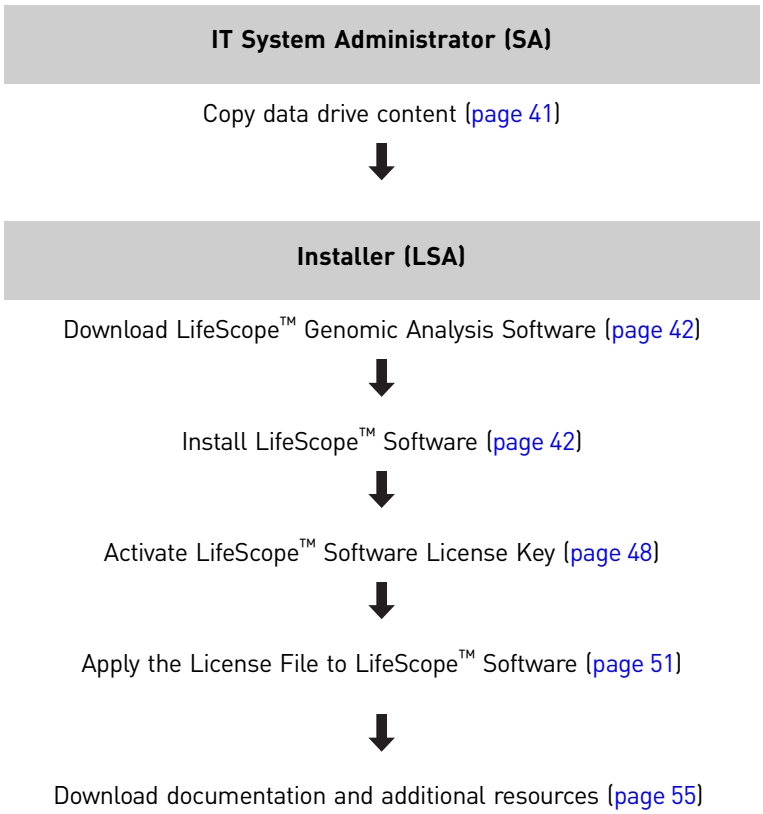
If you are not the LSA or if an LSA does not exist, please contact your System Administrator to have the master user and user group created. You must be logged in as the LSA to install LifeScope™ Software.

Identify two shared folders for holding the software and data in a suitably common location, accessible to all LifeScope™ Software users. Both locations should be accessible (mounted) from the compute nodes.

- Identify the LifeScope™ Software Binaries Folder (LSBF) to install LifeScope™ Software. This folder should be writable to LSA and readable to LSUs. Typically you may choose `/opt/lifescope`. If the LSBF does not exist, create it with appropriate permissions.
- Identify the location of the LifeScope™ Software Data Folder (LSDF) to hold all LifeScope™ Software resources, repositories, pre-installation tar files, installer, examples, test data, and other files. This location should have read and write permissions for the LSA and all LSUs. This location also should be sufficiently large to hold the initial data and the results of future analysis. Typically you may choose `/share/lifescope`. If the LSDF does not exist, create it with appropriate permissions.

Make sure that you have the appropriate read and write permissions for the LSBF and LSDF folders, before proceeding with the installation.

Installation workflow overview



Copy data drive content

(System Administrator) Copy the data drive content to the LifeScope™ Software Data Folder (LSDF) on the LifeScope™ Software server, and change `owner:group` to those of the LifeScope™ Software Administrator (LSA).

The data drive's file system is ext3, which typically is supported only on Linux.

The drive content includes the following:

- Files for use with hg18 and hg19 genomes:
  - Hash tables for the main mapping schemes
  - Mappability files for use with the CNV analysis module
- LifeScope™ Software reference repository files:
  - Hg18 and hg19 reference files and filter reference files
  - Genomic annotation files (dbSNP)
  - Annotation files from the miRNA database
- Data files for installation verification

The reference files downloaded in this step become the reference repository for your LifeScope™ Software. The location of the reference files (the `referenceData` directory) is required during installation.

IMPORTANT! If you add your own reference files to the reference repository, you must follow the instructions in "Add new reference files" and "Prepare reference files" on page 550.

Download LifeScope™ Software

1. (LSA) Visit the LifeScope™ Software project page:
   http://solidsoftwaretools.com/gf/project/lifescope
2. On the project page, you must:
   a. Read and accept the LifeScope™ Software End User License Agreement.
   b. Create an email with the following information:
      - Your acceptance of the LifeScope™ Software End User License Agreement.
      - Your LifeScope™ Software license key.
      - Your solidsoftwaretools account name.

      Send the email to:
      lifescope@lifetech.com
      Instructions to download LifeScope™ Software are emailed to you.
3. Go to the download page, as directed by the emailed instructions.
4. Download LifeScope™ Software `.tar.gz` file (for example, *LifeScope-2.0r8380-2011040333223344.tar.gz*) to the LifeScope™ Software Data Folder (LSDF).
5. Download the associated installer (`install.sh`) to the LifeScope™ Software Data Folder (LSDF).

Install LifeScope™ Software

This procedure provides typical, general instructions for installing LifeScope™ Software. The actual procedures for your site might vary, depending on the configuration of the LifeScope™ Software cluster and the installation options you select.

WARNING! LifeScope™ Software must ***NOT*** be installed as `root`.

Hardware configuration

This section explains the three types of installation based on hardware configuration. You must select your configuration when prompted by the installer in "Select the type of installation:", step 6 on page 44.

- **Standalone Workstation –** LifeScope™ Software is installed completely on the local workstation. There is no access to a cluster. Resource management software is still required on the workstation.

- **Cluster –** LifeScope™ Software is installed on a cluster with a head node, a set of compute nodes, shared storage, node local storage, and resource management software, as described in "Cluster hardware requirements" on page 38.
- **Workstation with Remote Submission –** The local workstation, instead of the cluster head node, is used for running the LifeScope™ Software server. The requirements for the workstation are:
  - A regular Linux 64-bit workstation with the following:
    - 250 GB harddrive
    - 4 GB RAM
    - Dual core
  - The operating system requirements are the same as those listed for the cluster in "Server requirements" on page 38.

  LifeScope™ Software jobs are submitted from the workstation onto the cluster. This requires that you *first* setup the workstation as described in "Set up a workstation with remote submission" on page 47.

  You must be logged in as the LifeScope™ Software administrator (LSA) on the local workstation, and not on the cluster. The installer (`install.sh`) also must be run on the local workstation and not on the remote cluster. You must provide the IP address of the cluster head node, and the LSA username and password, when asked by the installer. This information is required for the workstation to automatically login to the cluster for submitting jobs.

  By default the password appears as plain text in the file `<LSBF>/etc/analysis/system.properties` file. If the LSA is setup for password-less login as described in "Configure the workstation" on page 47, then you can give a blank password when asked by the installer, and prevent the password from appearing in the properties file.

Authentication realm

This section explains the authentication realms supported by LifeScope™ Software. You must supply your realm type when prompted by the installer in "Select an authentication realm to be used with LifeScope™ Software:", step 3 on page 46.

- **LifeScope –** This authentication realm refers to the LifeScope™ Software application user database. If this realm is chosen, user passwords are maintained in the software database. Passwords are stored as one-way MD5 digests for security reasons. Users must be manually created in this realm. There is no relationship between the application users and Linux® host system users. User account management can be done by the LSA using the LifeScope™ Software Admin module (included in this installation).
- **LDAP –** This realm refers to an LDAP-compliant authentication server (OpenLDAP, Active Directory, etc.). If this realm is chosen, user passwords are *not* stored in LifeScope™ Software. User credentials are authenticated against the configured LDAP server. Users need not be manually created in LifeScope™ Software with this realm (though they may be). A valid LDAP server address and LDAP Bind DN must be provided in the configuration. User account management can be done by the LSA using the LifeScope™ Software Admin module (included in this installation).

- **Host –** This realm refers to the Linux host machine on which the LifeScope™ Software server is running. If this realm is chosen, user passwords are *not* stored in LifeScope™ Software. User credentials are authenticated against the local machine using SSH (therefore uses whatever authentication scheme SSH uses, for example, NIS, `/etc/passwd`). There is a one-to-one relationship between the application users and Linux host system users. Users need not be manually created in the LifeScope™ Software Admin Portal with this realm.

Note: If you change the authentication realm after installation, you must stop and restart the LifeScope™ Software server.

Note: Changing a realm does not remove users from the previous realm. However, users might not be able to login in the new realm, unless they both exist in the new realm and are enabled in the new realm.

Installation instructions

1. (LSA) Go to the downloaded folder LSDF:

   ```
   cd /path/to/<LSDF>
   ```

2. Run this UNIX® command:

   ```
   chmod +x install.sh
   ```

   to make the downloaded `install.sh` file executable.

3. Run the installer:

   ```
   ./install.sh
   ```

   The welcome banner appears:

   ```
   ****************************************************************
   This is the LifeScope™ Genomic Analysis Software installation and
   licensing program
   ****************************************************************
   ```

4. Installation options include:

   ```
       1) Install the LifeScope™ Genomic Analysis Software.
       2) Upgrade your current installation of LifeScope™ Genomic
   Analysis Software.

   The following options require successful installation of
   LifeScope:
   ----------------------------------------------------------------
       3) Re-configure the LifeScope™ Genomic Analysis Software.
       4) Request a LifeScope™ Software license.
       5) Register a LifeScope™ Software license.

       6) Exit this application.

   Please select the operation you would like to perform (Q/q to
   quit) [1]:
   ```

   To install LifeScope™ Software, type **1**.

5. At the prompt, type **y** to begin installation.

6. Select the type of installation:

```
1) Standalone Workstation Server (default)
2) Cluster
3) Workstation w/Remote Submission
```

Review "Hardware configuration" on page 42 to determine your installation type.

7. Enter the location where LifeScope™ Software is to be installed. The default directory is `/opt/lifescope`.

   Accept the default or change the directory. Provide the location you have chosen for the LifeScope™ Software Binaries Folder (LSBF):

   `/path/to/<LSBF>`

   If the directory you enter already exists, you are prompted "Do you still want to use this location?"

8. Enter the location for reference files. Provide the destination location of the data drive contents (from the "Copy data drive content" step on page 41):

   `/path/to/<LSDF>/referenceData`

   Note: LifeScope reference files are required for full functionality of LifeScope™ Software. The default directory is `/data/results/referenceData`.

9. Enter the location for reads. This entry sets the location of the reads repository. You may choose

   `/path/to/<LSDF>/reads`

   The default directory is `/data/results/reads`.

10. Enter the location for projects. This entry sets the location of the projects repository. You may typically choose

    `/path/to/<LSDF>/projects`

    The default directory is `/data/results/projects`.

11. Enter the location for Binary Alignment Map (BAM) file repository. You may choose

    `/path/to/<LSDF>/bams`

    The default directory is `/data/results/bams`.

12. Enter the location for the scratch location for LifeScope. The default directory is `/scratch`.

    Note: The scratch location is used as a temporary work space for submitted jobs. For best performance, cluster and remote server installations require a scratch location on the local file system of each *compute node* of the system, not on the file system of the *head node*.

13. Enter the IP address or Fully Qualified Domain Name (FQDN) for the LifeScope™ Software server, for example: `192.168.1.27`.

    The address of the system where LifeScope™ Software is installed is required to successfully access and run the software. The system address is the IP address or Fully Qualified Domain Name (FQDN) of this system or head node.

    Note: Do not use the name `localhost` or IP address `127.0.0.1`. The address should be the public-facing name or IP address of the LifeScope™ Software server. The LSUs access the software using this name or IP address. Typically, the address is the domain name system (DNS) name of the server.

Test the connection to the IP address or FQDN to verify a connection to the system.

Enter the Web URL port address to be used to access the LifeScope Server. Default is 9998.

Configure the license server

1. Enter the IP address or Fully Qualified Domain Name (FQDN) for the LifeScope License Server.

   The installer tests the connection to the given IP address or FQDN to verify a connection to the system.

2. Enter the port address for the license server to be used with LifeScope™ Software.

3. Select an authentication realm to be used with LifeScope™ Software:

   1) LifeScope™ Software (default)

   2) Lightweight Directory Access Protocol (LDAP)

   3) Host

   Review **"Authentication realm" on page 43** to determine your realm type.

   Note: If you change the authentication realm after installation, you must stop and restart the LifeScope™ Software server.

   Note: Changing a realm does not remove users from the previous realm. However, users might not be able to login in the new realm, unless they both exist in the new realm and are enabled in the new realm.

4. Select the cluster resource manager to be used by LifeScope™ Software:

   1) PBS/Torque (default)

   2) SGE

   3) LSF

5. Enter the cluster-resource-manager-job-submission-queue-name to be used by LifeScope, for example, *lifescope*.

6. Enter the number of nodes available on the cluster. Default is 4.

7. Enter the number of cores for each compute node. Default is 8.

8. Enter the memory size (in GB) for each compute node. Default is 22.

IMPORTANT! Allow 2 GB for the compute nodes' OS. For example, for compute nodes with 24 GB, enter 22; for compute nodes with 48 GB, enter 46.

Continue installing LifeScope™ Software

1. At the prompt "Do you want to change any of these entries?" type **n** to accept the entries.

   The selected LifeScope™ Software configurations are saved.

2. At the prompt, "Do you want to continue installing the LifeScope package?" type **y** to continue the installation.

Set up a workstation with remote submission

This section describes prerequisites for the install type *Workstation w/Remote Submission.* Skip this section if your installation type is either of these:

1) Standalone Workstation Server (default)

2) Cluster

In the *Workstation w/Remote Submission* scenario, LifeScope™ Software is installed a workstation but submits analysis jobs to a remote cluster. The LifeScope™ Software server and licensing server run on the local workstation. The analysis jobs are submitted to the remote cluster using SSH. LifeScope™ Software services do *not* run on the cluster head node.

The workstation and cluster nodes share common file system locations through NFS mounting or other means. The installation location `<LSBF>/lifescope` and the LifeScope™ Software data folder `<LSDF>` must be on the common shared file system. Both the `<LSBF>/lifescope` and the `<LSDF>` folders must first be created on the cluster. (See **"LifeScope™ Software Administration" on page 39** for a description of `<LSBF>` and `<LSDF>`.)

The cluster head node must be accessible (via SSH) from the workstation while the analysis is executing.

As with the normal cluster installation, the LifeScope™ Software administrator account must exist on the cluster, as described in **"LifeScope™ Software Administration" on page 39**.

Configure the workstation

These instructions are only for Workstation w/Remote Submission, option 3) in "Select the type of installation:", step 6 on page 44. The IT system administrator performs these configuration steps. The commands given in this section are not applicable to all cluster and workstation combinations. The given commands are expected to serve as general guidelines to the system administrator to give an example of the procedures required. The actual commands to be used may vary depending on the system configuration.

1. Note down the UID and GID (user and group ids) of the LifeScope™ Software administrator (LSA) on the cluster.
2. Create a user and group on the workstation with the same UID and GID as that of the LSA on the cluster. If the LSA user account already exists on the workstation, the IT system administrate must modify the UID and GID of that user account on the workstation. For example:

   ```
   usermod -u <UID> <LSA>
   ```

3. Set up password-less login for the LSA user account from the workstation to the cluster, using `ssh-keygen`, `.ssh/authorized_keys` file on the destination, etc.
4. On the workstation, mount the location of the `<LSBF>`and `<LSDF>` directories from the cluster, using the same path. The path used on the workstation must be the same as the path used on the cluster.

   Example commands are:

   ```
   mkdir -p /path/to/<LSBF>
   mkdir -p /path/to/<LSDF>
   mount -t nfs cluster_ip:/path/to/<LSBF> /path/to/<LSBF>
   ```

```
mount -t nfs cluster_ip:/path/to/<LSDF> /path/to/<LSDF>
```

These commands may require root permissions. These are example commands. The actual commands may vary according to the system administrator's choice.

You are now ready to proceed with the installation procedures described in these sections:

- "Hardware configuration" on page 42
- "Authentication realm" on page 43
- "Installation instructions" on page 44

Activate the LifeScope™ Software License Key

(LSA) These instructions describe how to apply the License Key. The process includes (each of these steps is described below):

- Obtaining the MAC address of the server that has LifeScope™ Software
- Activating your License Key and obtain a license file specific to your server
- Applying the license file to activate your copy of LifeScope™ Software

When applied to LifeScope™ Software, the core license file grants up to five users access to the software for one year.

Prerequisites

Before activating the LifeScope™ Software License Key, please ensure that:

- You have downloaded and installed LifeScope™ Software on computer hardware that meets the LifeScope™ Software minimum hardware requirements (please see **www.lifetechnologies.com/LifeScope** for more details).
- You have performed the DNS hostname check described in "Check DNS hostname" on page 39.
- You are familiar with using command-line Linux®, and you have the necessary permissions to log on to the server that has LifeScope™ Software.
- You have a LifeScope™ Software License Key.

Internet connectivity

Direct internet connectivity for the host computer is highly recommended for the license activation process.

Check the computer date

Check that the computer date matches the current date and time. Each license has a start date. If the computer date is earlier than the license start date, access to the software is disabled until the license start date. If you need to reset the computer date, only reset it *forward*. ***Do not*** reset the computer date backward; doing so prevents access to LifeScope™ Software. Contact your system administrator (SA) if you need to change the date on the system.

IMPORTANT! Resetting your computer date backwards disables your license key and prevents the operation of LifeScope™ Software.

Obtain the MAC address

IMPORTANT! Run the Linux commands used in this procedure on the computer hardware that has LifeScope™ Software installed on it. If you want to run the LifeScope™ license server on a different machine, then you must obtain the MAC address for that machine.

IMPORTANT! The MAC address used for licensing LifeScope™ Software should be associated with a permanent Ethernet card on the host computer system. If there is a possibility that `eth0` is not a permanent Ethernet card, consult your system administrator to determine which Ethernet card should be used instead. If the Ethernet card associated with the LifeScope™ Software license is replaced on the host system, a new license file must be generated for your account.

To obtain the MAC address of the server that has LifeScope™ Software installed on it:

1. Run the Linux command

   `/sbin/ifconfig`

   Output similar to the one shown below appears:

```
[cora@foshtv02 ~]$ /sbin/ifconfig
eth0  Link encap:Ethernet  HWaddr 00:1C:23:BB:E1:F3
      inet addr:10.1.1.1  Bcast:10.1.1.255  Mask:255.255.255.0
      inet6 addr: fe80::21c:23ff:febb:e1f3/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:259842678 errors:0 dropped:0 overruns:0 frame:0
      TX packets:296616310 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:214032251167 (199.3 GiB)  TX bytes:33928429 (315.9 GiB)
      Interrupt:169 Memory:f4000000-f4012800

eth1  Link encap:Ethernet  HWaddr 00:1C:23:BB:E1:F1
      inet addr:167.116.6.72  Bcast:167.116.6.255  Mask:255.255.255.0
      inet6 addr: fe80::21c:23ff:febb:e1f1/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:25669852 errors:0 dropped:0 overruns:0 frame:0
      TX packets:82980132 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:10648346642 (9.9 GiB)  TX bytes:117855364 (109.7 GiB)
      Interrupt:169 Memory:f8000000-f8012800

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:11049203 errors:0 dropped:0 overruns:0 frame:0
      TX packets:11049203 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:6069766283 (5.6 GiB)  TX bytes:6069766283 (5.6 GiB)
```

2. In the paragraph labeled `eth0` , the first line should contain text that starts with `HWaddr`. The MAC address follows this text, and should look similar to

   `00:30:48:9F:72:76`

Obtain the MAC Address with the LifeScope™ Installer

Note: In this method, the MAC address is referred to as the "Computer ID."

If you have difficulty finding the MAC address, the LifeScope™ Installer can find the the address for you. To use the LifeScope™ Installer:

1. From the software download directory, run
   `install.sh`
   The LifeScope™ Installer information shown below appears:

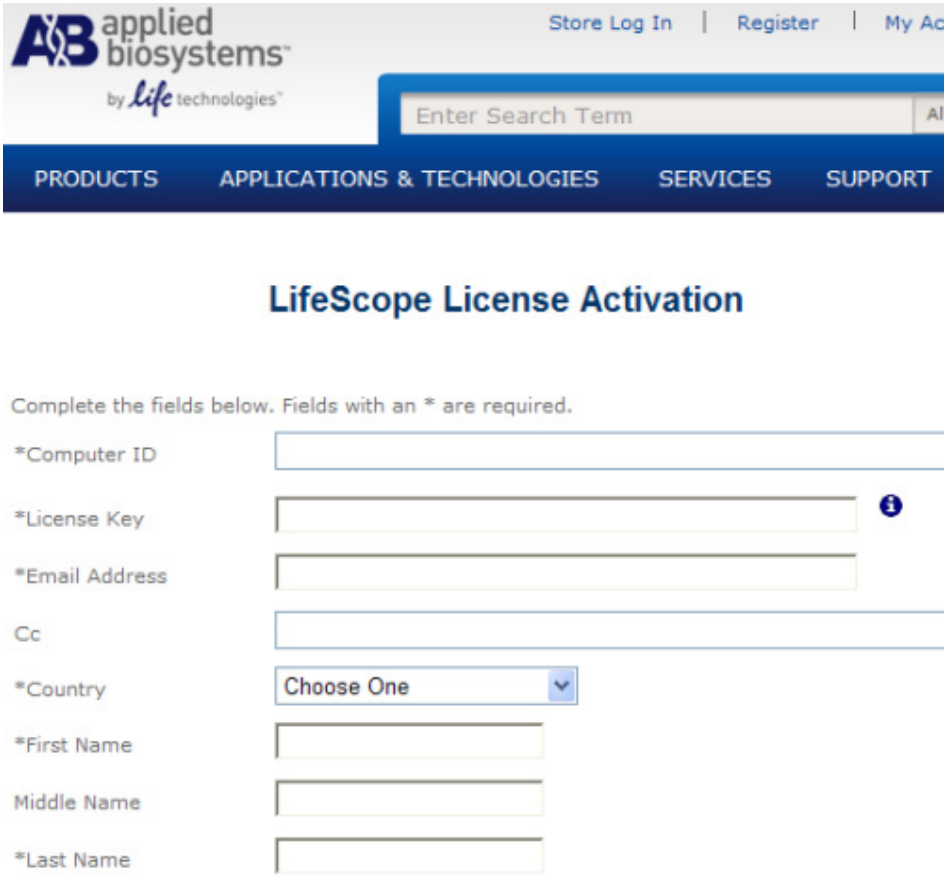
```
*******************************************************************************
This is the LifeScope Genomic Analysis Software installation and licensing program
*******************************************************************************

    1) Install the LifeScope Genomic Analysis Software.

The following options require successful installation of LifeScope:
-------------------------------------------------------------------
    2) Re-configure the LifeScope Software.
    3) Request a LifeScope Software license.
    4) Register a LifeScope Software license.

    5) Exit this application.

Please select the operation you would like to perform (Q/q to quit) [1]: 3

To request a LifeScope product license for operating the application:
      Enter the following URL into a Web browser. This will present a form to
      request a LifeScope product application license.
      ***************************************************************
      *    https://licensing.appliedbiosystems.com/activation/lifescope    *
      ***************************************************************

      Enter the Computer ID listed below on the form. If this utility is being run on a
      system other than the system running LifeScope, this Computer ID is invalid.
      Stop and re-run this utility on the system where LifeScope is installed.
      ************************************
      *    ComputerID: 00:1C:23:BB:E1:F3    *
      ************************************

      The License Key you must enter on the form is the license key
      you were given with the LifeScope system.

      When the form is completed and submitted, a LifeScope product license will be sent
      to the email address specified on the form. If multiple people are to receive this key,
      their email addresses should be entered in the "CC'ed" field of the form.

      Once the LifeScope product license key is received, execute "Step 2" of this utility
      to register the license with the License Server.
```

2. Choose option **3** ("Request a LifeScope product license.") to display the information about product licensing, including the MAC address of the first Ethernet card.

Activate License Key and obtain license file

To activate the license and obtain the license file associated with your MAC address:

1. Visit the LifeScope™ License activation page, shown below, at **https://licensing.appliedbiosystems.com/activation/lifescope**



2. Enter the MAC address in the **Computer ID** field.
3. Enter the License Key included in the LifeScope™ 2 Genomic Analysis Software license instructions.
4. Enter the email address where your license file (.lic) should be sent.
5. Fill in the remaining information, then click **Submit** to activate your license.

   Note: Your license begins the moment you activate your License Key.

The License File "Lifescope.lic" is sent to you as an attachment in an email from "RightNow AB Technical Support <RightNow.ABTechnicalSupport@lifetech.com>".

Apply the License File to LifeScope™ Software

(LSA) To apply the license to your copy of LifeScope™ Software:

1. Download the license file (.lic) to the system that has LifeScope™ Software has installed on it.

If you are using another computer for email, you can use tools such as `ftp` or `winscp` to transfer the.lic file to your LifeScope™ Software system. For more information on transferring files, contact your system administrator.

2. Move the .lic file to
   <*lifescope-installed-dir*>/server/licenses/
   Your LifeScope™ Software is now licensed.

IMPORTANT! Keep a backup copy of your license file in a safe place.

Later in "Start the licensing server" on page 53, you start the license server and confirm that the license has been correctly applied.

You have completed the LifeScope™ licensing process.

After the license has been activated, refer to the *LifeScope™ Genomic Analysis Software User Guide* (Part no. 4471877) administration appendix for instructions to make LifeScope™ Software available to users. These topics include the following:

- Setting a named license per user.
- Enabling and disabling LifeScope™ Software users.

LifeScope™ Software installation is completed. However, you are recommended to do at least one installation verification step. These steps are optional but confirm that the software is functioning properly and that the LifeScope™ Software Data Folder (LSDF) is properly configured. The optional installation verification steps are:

- The demo analyses
- The performance verification tests

See Chapter 3, "Verify the LifeScope™ Genomic Analysis Software Installation" on page 57 for instructions on optionally running example analysis workflows and optionally performing verification tests.

Set the LifeScope™ Software environment

Follow the instructions in "Set the LifeScope™ Software environment" on page 68 to set your `PATH` variable.

Start LifeScope™ Software servers and check servers' status

This section describes starting the LifeScope™ Software servers and checking their status. Commands described in this section are run in the Linux shell (not in the LifeScope™ Software command shell).

The LifeScope™ Software licensing server must be started first, before the LifeScope™ Software web server. Only the LifeScope™ Software administrator (LSA) may start the licensing server.

Start the licensing server

To start the licensing server, follow these steps:

1. Make sure LifeScope™ Software binaries are in the `PATH` environment variable. If the `PATH` variable is not set properly, set `PATH` using this Linux® command:

   ```
   export PATH=<lifescope-installed-dir>/bin:$PATH
   ```

2. Start the license server, using this Linux command:

   ```
   lscope-lmgrd.sh start
   ```

   If the license server is already running, a second instance is not started.

Check the licensing server

1. Use the following command to check if the LifeScope™ Software licensing server is running:

   ```
   ps -ef | grep lmgrd
   ```

   If the LifeScope™ Software licensing server is not running, the Linux command prompt returns (with no output). If the LifeScope™ Software licensing server is running, this command displays two processes, one for the license manager daemon (`lmgrd`) and one for the vendor daemon (`lifetech`). Example output when the licensing server is running is:

   ```
   lifescope 20447     1  0 Mar04 00:00:00 ./lmgrd -c /opt/
   lifescope/lifescope/server/licenses/LifeScope.lic
   lifescope 20448 20447  0 Mar04 00:00:00 lifetech -T
   foshtdvv12.na.ab.applera.net 11.9 4 -c /opt/lifescope/
   lifescope/server/licenses/LifeScope.lic -lmgrd_port 6979
   --lmgrd_start 4d59fb1f
   ```

2. Verify the license status and the available user licenses, shown below, using the Linux command:

   ```
   lscope-lmgrd.sh status
   ```

```
[panak1@fospanak2 licenses]$ cd /share/apps/lifescope/server/licenses/
[panak1@fospanak2 licenses]$ ls -l
total 8
 -rwxrw-r-- 1 panak1 panak1 1144 May  5 17:50 LifeScope.lic
[panak1@fospanak2 licenses]$ echo $PATH
/share/apps/lifescope/bin:/usr/lib64/qt-3.3/bin:/usr/kerberos/bin:/usr/local/bin:/usr/bin:
/bin:/usr/X11R6/bin:/home/panak1/bin
[panak1@fospanak2 licenses]$ lscope-lmgrd.sh start
Started License Server

[panak1@fospanak2 licenses]$ lscope-lmgrd.sh status
lmutil - Copyright (c) 1989-2010 Flexera Software, Inc. All Rights Reserved.
Flexible License Manager status on Thu 5/5/2011 17:53

License server status: 27001@fospanak2.ads.invitrogen.net
     License file(s) on fospanak2.ads.invitrogen.net: /share/apps/LifeScope-2.0.r0-86630_2011
0412120300/server/licenses//LifeScope.lic:

fospanak2.ads.invitrogen.net: license server UP (MASTER) v11.9

Vendor daemon status (on fospanak2.ads.invitrogen.net):

  lifetech: UP v11.9
Feature usage info:

Users of LTC.BIOSCP.USERS:  (Total of 5 licenses issued;  Total of 0 licenses in use)

Users of LTC.BIOSCP.LAUNCH:  (Total of 1 license issued;  Total of 0 licenses in use)

Users of LTC.BIOSCP.CONCURRENT:  (Total of 1 license issued;  Total of 0 licenses in use)
```

3. Check that you have activated the same number of licenses that you ordered by cross-verifying the number with the result of the `status` command.

You must start the license server every time the system undergoes a reboot.

Start the LifeScope™ Software server

Only the LifeScope™ Software administrator (LSA) must start the LifeScope™ Software server. The command to start the LifeScope™ Software server is:

```
lscope-server.sh start
```

The server by default uses port 9998, but the default port number might be changed during installation. To run on a different port, use the following version:

```
lscope-server.sh start -p port
```

If you start the server on a port other than the default 9998, you must check that the firewall is also enabled for the new port number. See "Check firewall restrictions" on page 39.

If the port is busy, the server start fails.

If the server is already running, a second instance is not started. The server issues the following message:

```
Web server already running (pid=nnnnn) on port nnnn
```

You must start the LifeScope™ Software server every time the system undergoes a reboot.

Check the LifeScope™ Software server status

The following command checks if the LifeScope™ Software server is running:

```
ps -ef | grep WebServer
```

If the LifeScope™ Software server is not running, the Linux command prompt returns (with no output). Example output when the web server is running is:

```
lifescope 23521     1  0 13:26 pts/1    00:00:05 java -server
-Xmx3072m -XX:PermSize=512m -XX:MaxPermSize=512m -
XX:+UseParallelGC com.lifetechnologies.bioscope.webservice.
WebServer -b /opt/lifescope/lifescope
```

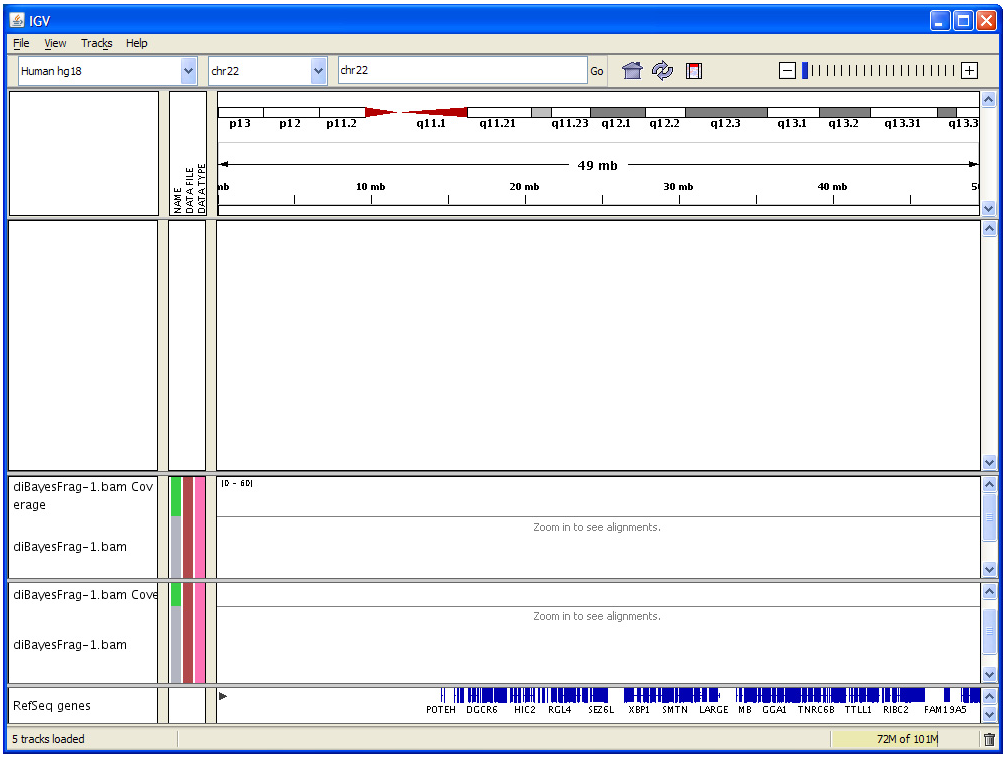
Download documentation and additional resources

To ensure you have the latest version of LifeScope™ Software, please visit **http://solidsoftwaretools.com/gf/project/lscope_release** and look for the Download option. You also find user documentation and the latest information and technical resources. The full resources package includes pre-installation scripts, smaller data files, the installation wizard, and performance verification scripts. Additional links and online support resources are also available at this site.

Integrative Genomics Viewer (IGV)

You can optionally install the Integrative Genomics Viewer (IGV), shown on page 55, available from the Broad Institute. The IGV is a visualization tool for interactive exploration of large, integrated datasets. It directly reads BAM files, which enables you to easily view and inspect alignments against the genome.

For more information about the IGV, go to **www.broadinstitute.org/igv/**.



3 Verify the LifeScope™ Genomic Analysis Software Installation

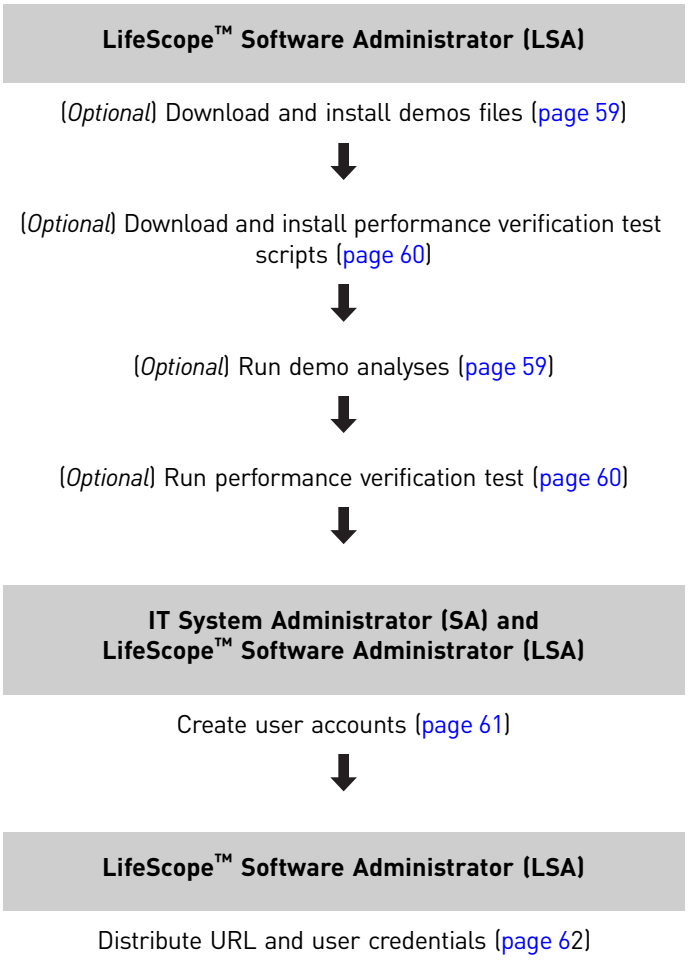
This chapter covers:

- Overview 57
- Workflow 58
- Verify the installation 58
- (Optional) Download and install demos files 59
- (Optional) Run demo analyses 59
- (Optional) Download and install performance verification scripts 60
- (Optional) Run the performance verification test 60
- Create user accounts 61
- Distribute the URL and user credentials 62
- Maintenance 63

Overview

This chapter describes procedures and tests that the LifeScope™ Software administrators should perform after the software has been installed. LifeScope™ Software users may skip this chapter.

Workflow



Verify the installation

(LSA) Two mechanisms are provided to verify the installation:

- **Demos –** A set of individual analysis modules with small data sets. The demos take approximately 30 minutes or less to complete.
- **Performance verification –** A realistic computation scenario that takes approximately 24 hours to complete.

IMPORTANT! You are strongly recommended to run one or both of the verification procedures. The procedures are listed as optional because you are not required to run both of them. Run at least one of the optional verification procedures.

You may run the demo analyses and performance verification scripts as the LifeScope™ Software administrator or as a LifeScope™ Software user. See "Create user accounts" on page 61 create a LifeScope™ Software user account.

(*Optional*) Download and install demos files

(LSA) This step is required in order to run the optional sample analyses, "(Optional) Run demo analyses" on page 59.

Example scripts and data for demo LifeScope™ Software analyses are available from the following site:

http://solidsoftwaretools.com/gf/project/lscope_release

Included in this download are the required PLN and INI files for running individual LifeScope™ Software analysis modules, and scripts to run the analyses in the LifeScope™ Software command shell.

Follow these steps to install the demo files:

1. Download the `.tar.gz` file to the LifeScope™ Software Data Folder (LSDF).
2. Go to the LSDF directory and untar the `.tar.gz` file:

   ```
   cd <LSDF>
   tar -xzf exampleWorkflows.tar.gz
   ```

Note: The demo analyses are not recommended as the first experience for users new to LifeScope™ Software. The standard workflows (Chapter 7, Run a Standard Workflow Analysis) are recommended for new users to learn LifeScope™ Software command-line analyses.

(*Optional*) Run demo analyses

(LSA) Run demo analyses, which takes approximately 30 minutes.

These instructions require the following (in addition to the software installation):

- The default reference repository, which includes hg18 and hg19 assemblies. The reference repository is created during the "Copy data drive content" on page 41 and "Install LifeScope™ Software" on page 42 procedures.
- The demos files and sample data installed during the "(Optional) Download and install demos files" on page 59.

Follow these steps to execute the demos in one run:

1. Follow the instructions in "Set the LifeScope™ Software environment" on page 68 to set the LifeScope™ Software environment.
2. Go to the demos directory:

   ```
   cd <LSDF>/examples/demos
   ```

3. Execute this command in the Linux shell:

   ```
   ./runall.sh -u username
   ```

where *username* must be for the LifeScope™ Software administrator or for a valid LifeScope™ Software user account. Provide the password when prompted.

This script runs each demo one at a time.

Check results

The results files are generated in the projects repository, at a location which is set during installation. Its default location is at `<LSDF>/projects`. Within the projects repository directory, the demos results are generated at the following directories:

username`/examples/`*analysis_name*`/outputs/`*sample_name*

Replace *username* with the user name used to run `runAll.sh`. The default is `lifescope`, the LSA account. The string `examples` is the project name used by all demos. *analysis_name* is the name of the analysis module used by a demo, and is for example, `cnv`, `dibayes`, `enrichment`, etc. Each *analysis_name* folder has an `outputs` subfolder. The final folder, *sample_name*, is determined by the sample in the input data.

Each of the demo analyses creates a log file named `summary.log` in the `lifescope/examples/`*analysis_name* directory. Check for a success message in the last line of each log file.

The logs and output files are overwritten on subsequent runs executed by the same user.

(Optional) Download and install performance verification scripts

(LSA) This step is required in order to run the optional performance verification tests, "(Optional) Run the performance verification test" on page 60.

The performance verification test data file are in the LSDF directory, from the "Copy data drive content" step on page 41.

The performance verification test scripts are available from the following site:

http://solidsoftwaretools.com/gf/project/lscope_release/performanceVerificationTestScripts.tar.gz

Follow these steps to install the performance verification test scripts:

1. Download the file `performanceVerificationTestScripts.tar.gz` to the LifeScope™ Software Data Folder (LSDF).
2. Go to the LSDF directory and untar the `.tar.gz` file:

```
cd <LSDF>
tar -xzf performanceVerificationTestScripts.tar.gz
```

(*Optional*) Run the performance verification test

(LSA) Run a performance verification test, which takes approximately 24 hours.

The performance verification data files are installed during the "Copy data drive content" step on page 41. The performance verification test scripts are installed during the "(Optional) Download and install performance verification scripts" step on page 60.

Follow these steps to execute the performance verification test:

1. Follow the instructions in "Set the LifeScope™ Software environment" on page 68 to set the LifeScope™ Software environment.
2. Confirm that the performance verification test data is installed at this directory:

   `<LSDF>/performanceVerificationData`
3. Go to the performance verification test scripts directory:

   `cd <LSDF>/performanceVerificationTestScripts`
4. Execute this command in the Linux shell:

   `./runAll.sh -u` *username*

   where username must be for the LifeScope™ Software administrator or for a valid LifeScope™ Software user account. Provide the password when prompted.

Monitor progress

Monitor the progress of your run by checking the output log files and also through job status commands such as `qstat`.

Check results

The results files are generated in the projects repository, at a location which is set during installation. Its default location is at `<LSDF>/projects`. Within the projects repository directory, the performance verification results are generated at the following results directories:

username`/performanceVerification/`*analysis_name*`/outputs`

Replace *username* with the user name used to run `runAll.sh`. The default is `lifescope`, the LSA account. The string `performanceVerification` is the project name used by the performance verification test. The analysis names used by `runAll.sh` are `genomicResequencing.20x`, `genomicResequencing.2x`, and `wholeTranscriptomePE`. The `outputs` subfolder name is fixed.

The performance verification tests creates a log file in the *analysis_name* directory. Check for a success message in the last line of each log file.

The results files are overwritten on subsequent runs executed by the same user.

Create user accounts

(LSA or SA) Create user accounts.

For instructions in this section that require using the LifeScope™ Software Admin Portal, refer to the administration appendix in the *LifeScope™ Genomic Analysis Software User Guide* (Part no. 4471877).

The method for user account creation depends on the authentication realm for your installation:

- **LifeScope realm –** (LSA) The LifeScope™ Software administrator creates LifeScope™ Software user accounts using LifeScope™ Software. The LifeScope™ Software Admin Portal is recommended for user administration.

  LifeScope™ Software shell commands for user administration are described in Table 13 on page 89, in Chapter 6, Run a Command Shell Analysis. Although shell commands include some user administration commands, the Admin Portal is recommended for general user administration.

- **LDAP realm –** (SA and LSA) The Linux system administrator creates user accounts on the LDAP server. The LifeScope™ Software administrator must use the LifeScope™ Software Admin Portal to enable each LifeScope™ Software user.
- **Host realm** – (SA) The Linux system administrator creates Linux user accounts on the local operating system.

Enable named users

(LSA) This step is not required if your installation uses the concurrent licensing scheme. (With concurrent licensing, all users in the authentication realm are automatically enabled for LifeScope™ Software.)

If your installation uses a named licensing scheme, the LSA must use the Admin Portal to identify the users for the named license.

Distribute the URL and user credentials

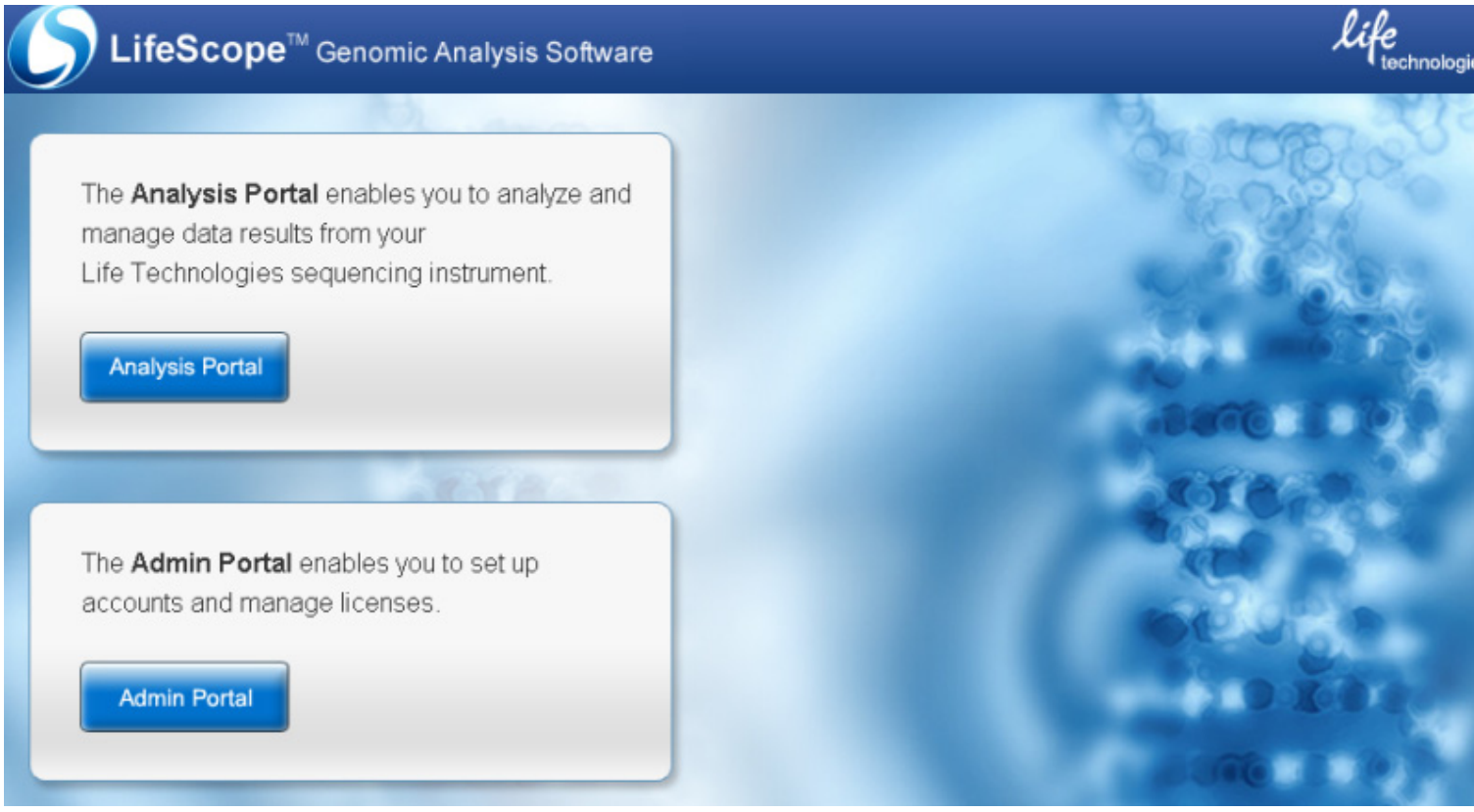
(LSA) The LifeScope™ Software URL follows this pattern:

http://*<fqdn of host>*:*<port number>*/LifeScope.html

For example, for an installation on host `orange.intranet.company.com`, using the default port `9998` for the LifeScope™ Software server, the URL is:

http://orange.intranet.company.com:9998/LifeScope.html

With the LifeScope™ Software servers running, verify the LifeScope™ Software URL on a supported browser. The following page appears:



Notify LifeScope™ Software users (LSUs) of the LifeScope™ Software URL.

If new user accounts are created for LifeScope™ Software users, notify them of the username and password credentials required to access the LifeScope™ Software.

Maintenance

Stop the LifeScope™ Software server

The following cautions apply to stopping the LifeScope™ Software server:

- The LifeScope™ Software server should only be stopped for required maintenance such as license upgrades.
- Only the LSA account can stop the LifeScope™ Software server.
- In order to stop both the LifeScope™ Software server and the licensing server, you must stop the LifeScope™ Software server first.
- All running analyses are terminated when you stop the LifeScope™ Software server.

Follow these steps to stop the LifeScope™ Software server:

1. If you have not already done so, set the LifeScope™ Software environment. See "Set the LifeScope™ Software environment" on page 68.
2. Check for LifeScope™ Software jobs running on the cluster with a command such as `qstat`. Notify LifeScope™ Software users that their jobs are affected.
3. Execute this Linux command:

   ```
   lscope-server.sh stop
   ```

Stop the licensing server

The following cautions apply to stopping the licensing server:

- The LifeScope™ Software licensing server must not be stopped except for required maintenance such as license upgrades.
- Only the LSA account can stop the licensing server.
- You must stop the LifeScope™ Software server first before stopping the licensing server.
- Running LifeScope™ Software analyses are affected if the licensing server is stopped.

IMPORTANT! If the LifeScope™ Software licensing server stops accidentally while the LifeScope™ Software server is running, the licensing server must be restarted as soon as possible. The licensing checks fail if the licensing server is not operating, and running analysis jobs may stop.

Follow these steps to stop the LifeScope™ Software licensing server:

1. If you have not already done so, set the LifeScope™ Software environment. See "Set the LifeScope™ Software environment" on page 68.
2. Execute this Linux command:

   ```
   lscope-lmgrd.sh stop
   ```

3. Verify that the LifeScope™ Software licensing server has stopped, with this command:

```
lscope-lmgrd.sh status
```

Upgrade

Upgrading to a new version of LifeScope™ Software does not require you to uninstall your existing version of LifeScope™ Software.

Follow the installation instructions for the new version of LifeScope™ Software. This procedure installs the new version in the <*LSBF*> alongside your existing version, and modifies the soft link <*LSBF*>/lifescope to point to the new version. The older version of LifeScope™ Software is not deleted during installation.

After the upgrade, ensure that your license files are moved or copied from the older installation to the newer one.

Once the new installation has been verified, you may remove the older installation at your discretion.

Uninstall

This section describes the steps for the LifeScope™ Software administrator (LSA) to uninstall LifeScope™ Software. If you use an authentication realm that requires the IT system administrator (SA) to create user accounts, the SA deletes those user accounts as part of the uninstall process.

Follow these steps to uninstall LifeScope™ Software:

1. (LSA or SA) Backup your LifeScope™ Software license file or files, from the directory <*LSBF*>/lifescope/server/licenses.
2. (LSA or SA) Backup any required files, from the <*LSDF*> directories (these directories hold the LifeScope™ Software repositories):
   - Reads files (XSQ and BAM)
   - Reference files
   - Output files
   - Data
   - Projects and analyses

   Notify LifeScope™ Software users to backup any files they might need from the <*LSDF*> directories.
3. (LSA or SA) Delete the LifeScope™ Software installation directory, with this Linux command:

   ```
   /bin/rm -rf <LSBF>/lifescope
   ```
4. (*Optional*) (LSA or SA) Delete the LifeScope™ Software data folder, with this Linux command:

   ```
   /bin/rm -rf <LSDF>
   ```
5. (*Optional*) (SA) If you use host or LDAP authentication, delete the user accounts you created for LifeScope™ Software users.

(*Optional*) (LSA or SA) If you modified any bash profile scripts in /etc or in the local home directory to change the PATH variable, remove the lifescope location entries in the PATH variable.

PART III
Getting Started

4 Before You Begin

This chapter covers:

- Overview 67
- Verify the browser version 68
- Set the LifeScope™ Software environment 68
- Check LifeScope™ Software servers 69

Overview

This chapter is for LifeScope™ Genomic Analysis Software users. Some procedures described in this section require that you:

- Know the IP address of the LifeScope™ Software cluster
- Have a login ID on the LifeScope™ Software cluster
- Know how to:
  - Navigate to directories in a Linux environment
  - Edit and save files in a text editor
  - Run Linux shell scripts
  - Run Linux commands such as `ls`, `pwd`, `cd`, `echo`, `grep`, and other common utilities

This chapter describes the following procedures:

- Verify your browser version
- Set your shell environment to use LifeScope™ Software

  Note: In some installations, the LifeScope™ Software system administrator sets your environment for you.

- Check whether the LifeScope™ Software servers are running

Verify the browser version

LifeScope™ Genomic Analysis Software supports:

- Internet Explorer® versions 6 and 7
- Mozilla® Firefox 3.0.1

Set the LifeScope™ Software environment

To set your environment to run LifeScope™ Software, add the LifeScope™ Software `bin` directory to your PATH and export the new PATH variable. Enter this command every time you open a Linux shell window to run LifeScope™ Software:

```
export PATH=<installdir>/bin:$PATH
```

where *installdir* is the LifeScope™ Software installation directory.

The *installdir* is the `<LSBF>/lifescope` directory defined in the administrator instructions ("LifeScope™ Software Administration" on page 39).

If BioScope™ Software is not used on the same machine, you can make your `PATH` setting permanent by modifying your local `.bashrc` file or the global `/etc/bashrc` files. Contact your system administrator for more information.

BioScope™ Software users' PATH variable

If you install LifeScope™ Software on a machine that also has BioScope™ Software installed and the BioScope™ Software is actively being used, you must follow these instructions:

1. LifeScope™ Software users must *not* use their shell profile script to update the `PATH` variable with the LifeScope™ Software `bin` directory. If the LifeScope™ Software `bin` directory is added to the `PATH`, BioScope™ Software cannot run correctly.
2. Run BioScope™ Software and LifeScope™ Software in different Linux® shell windows.
3. Set and export your `PATH` variable *every* time you open a Linux shell window to run LifeScope™ Software server or client software.

BioScope™ profile scripts

Some customers with early versions of BioScope™ Software use a profile script to set the `BIOSCOPEROOT` environment variable for all users. If you have the file `/etc/profile.d/bioscope_profile.sh` on your system, you must delete this file, or remove it from the `/etc/profile.d` directory, for LifeScope™ Software to run properly.

Clusters running both versions

If you currently run BioScope™ Software on your cluster, and need to support both `BioScope™ and LifeScope™` users on the same cluster, follow these steps:

- Remove the file `bioscope_profile.sh` from the `/etc/profile.d` directory (the script cannot be in any `/etc` init directory). Having this script in a system init directory prevents LifeScope™ Software from running properly. (Save the script in your own private location in order to configure the environment for BioScope™ Software users, as described below.)

- Users who run LifeScope™ Software must not have the environment variable BIOSCOPEROOT set and must not have the BioScope™ Software `/bin` directory in their PATH.
- Users who still run BioScope™ Software should configure their environment by copying the content of the file `/etc/profile.d/bioscope_profile.sh` into their own shell setup file (such as `.bash_profile`).

Check LifeScope™ Software servers

This section describes checking the status of LifeScope™ Software servers. Commands described in this section are run in the Linux shell (not in the LifeScope™ Software command shell).

The LifeScope™ Software licensing server must be started first, before the LifeScope™ Software web server. Only the LifeScope™ Software administrator (LSA) may start the servers.

Check the licensing server

Use the following command to check if the LifeScope™ Software licensing server is running:

```
ps -ef | grep lmgrd
```

If the LifeScope™ Software licensing server is not running, the Linux command prompt returns (with no output). If the LifeScope™ Software licensing server is running, this command displays two processes, one for the license manager daemon (`lmgrd`) and one for the vendor daemon (`lifetech`). Example output when the licensing server is running is:

```
lifescope 20447     1  0 Mar04 00:00:00 ./lmgrd -c /opt/lifescope/
lifescope/server/licenses/LifeScope.lic
lifescope 20448 20447  0 Mar04 00:00:00 lifetech -T
foshtdvv12.na.ab.applera.net 11.9 4 -c /opt/lifescope/lifescope/
server/licenses/LifeScope.lic -lmgrd_port 6979 --lmgrd_start
4d59fb1f
```

Check the LifeScope™ Software server status

The following command checks if the LifeScope™ Software server is running:

```
ps -ef | grep WebServer
```

If the LifeScope™ Software server is not running, the Linux command prompt returns (with no output). Example output when the web server is running is:

```
lifescope 23521     1  0 13:26 pts/1    00:00:05 java -server
-Xmx3072m -XX:PermSize=512m -XX:MaxPermSize=512m -
XX:+UseParallelGC com.lifetechnologies.bioscope.webservice.
WebServer -b /opt/lifescope/lifescope
```

5 Understand The LifeScope™ Software Shell

This chapter covers:

- Overview 71
- Terminology and concepts 72
- Common scenarios 80
- Naming restrictions 81

Overview

This chapter describes the concepts and terminology required when working with LifeScope™ Software command shell. New concepts used with the shell include the following list. These terms are explained in more detail in this chapter.

- **Project –** A collection of analyses based on scientific experiments or computational methodologies.
- **Module –** A module is a single step in an analysis workflow. For example, mapping, SAET, and CNV are modules.
- **Analysis –** A set of modules within an analysis workflow. An analysis can contain one or more modules.
- **Analysis workflow –** A pre-defined set of modules in a workflow based on common applications areas, such as targeted resequencing or small RNA analysis.
- **Read-set –** Sequencing data associated with one XSQ file or with an indexed (barcoded) sample from one XSQ file.
- **Read-set repository –** An indexed database of XSQ files based on individual read-sets.
- **Group –** A collection of data that can be analyzed together. Grouping allows for aggregrated data (for instance, data from different lanes or different runs) to be analyzed as one set of data.

The command shell presents a virtual directory hierarchy to the user. The shell operations are performed on the LifeScope™ Software server.

Terminology and concepts

Table 7 defines terminology and concepts used to describe the LifeScope™ Software command shell. Review these terms before proceeding to the shell syntax in Chapter 6.

Table 7 Concepts related to the LifeScope™ Software command shell

Term	Description
category	Category refers to the various types of information containers at the root level of the LifeScope™ Software command shell. An `ls` command run at the shell root level lists the following categories: `bams`, `projects`, `reads`, `references`, `users`, and `workflow`.
level, virtual directory	Levels refer to the virtual directory structure within the LifeScope™ Software command shell. For example: The root level: `/` The workflow level: `/workflows` The project level: `/projects` The analysis level: `/projects/proj1/run1` These levels are a representation of your projects and other shell elements on the LifeScope™ Software server.
prompt	The shell prompt displays your current level within the LifeScope™ Software command shell environment. The following are examples of the prompt strings at various shell levels: At the root level: `server:/>` At the workflow level: `server:/workflows>` At the project level: `server:/projects>` At the analysis level: `server:/projects/proj1/run1>`
projects	A project is a collection of analyses based on scientific experiments or computational methodologies. In the LifeScope™ Software command shell, projects appear at `/projects` virtual directory. You create your projects at the `/projects` level. Within each project, you create one or more analyses. You may use projects as you wish, to group your analyses in a way that is convenient and meaningful for you. Each project is private to an individual LifeScope™ Software command shell user.
analysis	An analysis is a set of modules within an analysis workflow. An analysis can contain one or more LifeScope™ Software modules. An analysis is defined by the following configuration: • The analysis methodology (through the `set workflow` command, specifying which LifeScope™ Software modules are executed). • The source (input) reads (through the `import`, `add xsq`, and `set groups`). • The genome reference to be used (with the `set reference` command). In addition, for tertiary-only workflows you can specify source BAM files (through the `add bam` command).
module	A module is a single step in an analysis workflow. For example, mapping, SAET, and CNV are modules.

Table 7 Concepts related to the LifeScope™ Software command shell *(continued)*

Term	Description
analysis workflow	A pre-defined set of modules in a workflow based on common applications areas such as small RNA analysis or targeted resequencing. Also known as a standard workflow. A workflow combines a set of analysis modules into one run of LifeScope™ Software. See Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information.
analysis space	Your analyses can be performed in either base space or color space (if both base and color are available in your input data). You are recommended to perform secondary and tertiary analyses in the same space. See "Analysis space" on page 77.
read-set	Sequencing data associated with either: • For indexed (barcoded) data: an indexed sample from one XSQ file. • For non-indexed data: the data in one XSQ file.
group	A collection of data that can be analyzed together. Grouping allows for aggregrated data (for instance, data from different lanes or different runs) to be analyzed as one set of data. The implications of using groups with your read-sets include the following: • Reads that are grouped together are treated as one specimen, even if the reads are in different XSQ input files. • LifeScope™ Software supports processing multiple groups in one analysis run, while treating each group's data as an independent run. Groups are created through the following methods: • The group option of the `add xsq` command. • The `set groups` command. • The `-g` option of the LifeScope™ Software shell run command. A read-set can belong to a maximum of one group.
groupsfile	A text file optionally specified with an analysis. The groups file is a tab-separated text file with multiple lines in the following format: groupname <*path*>/xsqorbamfile:*indexid* <*path*>/reference_file If no barcode *indexid* is specified, it is assumed to be "1" (corresponding to the non-barcoded data). If a reference column entry does not exist, the reference from the previous line in the file is used. All members of a group must use the same reference. A groupsfile supports only a single reference, and that reference must be specified on the *first* line of the groups file.
repository	A LifeScope™ Software shell repositories are virtual containers that persist your reference files, reads files, and projects. The projects repository, at `/projects`, contains your projects. The read repository, at `/reads`, is an indexed database of XSQ files based on individual read-sets. The references repository, at `/references`, contains the reference genomes used in your analyses.

Table 7 Concepts related to the LifeScope™ Software command shell *(continued)*

Term	Description
directory	The LifeScope™ Software command shell's repositories and analyses are represented in a structure resembling directories, but these are in a virtual system that lets you browse a representation of your projects and analyses on the LifeScope™ Software server.
shell	This document discusses two different shells and uses the following terms to distinguish between them: • LifeScope™ Software command shell and LifeScope™ Software shell—Refer to the LifeScope™ Software command-line environment that is invoked with the `lscope.sh` command. • Linux shell—Refers to the shell of the underlying operating system.
local file system	This document uses the phrase "on your local file system" to describe a file that is specified by a Linux absolute path, such as `/data/results/example.sh`. Such a file could be on a different physical disk instead of the local file system, but is available to Linux through an absolute path. The file is navigable from or available to the local file system.
INI and PLN files	(*Optional*) INI and PLN (pronounced "plan" or "PLN") files define your LifeScope™ Software experiment. The LifeScope™ Software command shell creates these file for you, based on the projects and analyses you create and on how you have configure your analyses. Each INI file usually corresponds to one LifeScope™ Software module (such as mapping, SNPs, or CNV), and defines the behavior of that module through a set of key-value parameters (for example, the parameters for allowing IUB codes in references or setting the call stringency level). Each PLN file defines which LifeScope™ Software modules execute to perform your analysis. A PLN file lists the INI files corresponding to the LifeScope™ Software modules to execute. The PLN fie list defines the order of execution and execution dependencies. You can define and run your analyses in the LifeScope™ Software command shell, without directly editing INI and PLN files. It is also possible to define analyses outside of the command shell, using only INI and PLN files. See Appendix F, "Command Shell Control Files" on page 553 for information on INI and PLN files.
run in the projects repository, or run on the local file system	When "run in the projects repository", a LifeScope™ Software analysis creates its output files in the projects repository and the analysis (with its configuration, status, and results files) can be viewed through the LifeScope™ Software UI. When "run on the local file system" (also called "run in the current directory"), the output files are created in the current directory on the local file system, and the analysis cannot be viewed through the UI. See "Run mode" and Table 12 on page 87 for more information.

Projects and analyses

Projects are containers for your analysis runs. You may use projects as you wish, to organize your analyses as is convenient and meaningful for you.

You create your projects at the `/projects` level. A project itself does not have a configuration or definition. A project is a way of organizing your LifeScope™ Software analysis runs.

Within each project, you create one or more analyses. Through the configuration of an analysis, you define the executable run.

Table 8 lists example steps to create a run in the LifeScope™ Software shell. These steps assume you have logged into the shell. (See Chapter 6, "Run a Command Shell Analysis" on page 83 for complete instructions.)

Table 8 Setting up an analysis run in the LifeScope™ Software shell

Command	Description
`cd projects`	Enter your projects area.
`mk proj1`	Create a project named `proj1`.
`cd proj1`	Enter the `proj1` level.
`mk run1`	In your `proj1` project, create an analysis named `run1`.
`cd run1`	Open the `run1` analysis, to configure the reference and reads files, and the workflow (the analysis modules).

The configuration of analysis includes these areas:

- The reference genome.
- The reads files.
- The type of analysis to be performed (which LifeScope™ Software modules perform the analysis). The analysis type can be any of the following:
  - A pre-defined workflow, which is a built-in sequence of commonly-used analyses, corresponding to a common biological application.
  - An analysis sequence you define, by either creating your own or customizing one of the LifeScope™ Software pre-defined workflows.

An example of a pre-defined workflow is the targeted resequencing fragment workflow. This workflow includes the following LifeScope™ Software modules:

1. SAET
2. Mapping
3. Mapping statistics (BAMStats)
4. Enrichment
5. SNPs
6. Small indels
7. Annotations

One LifeScope™ Software shell command defines this workflow for your analysis (the `set workflow targeted.resequencing.frag` command).

Once an analysis is configured, it is launched with a single shell command (the `run` command).

Your projects are private and cannot be seen by other users or shared with other users.

Naming restrictions

Project and analysis names must conform to Linux filename conventions:

- Contain only alphanumeric and underscore characters.
- Do not contain spaces.
- Must be unique (among siblings).
- Do not begin with an underscore character. (This requirement is a LifeScope™ Software limitation, not a Linux rule.)

In sample group names and reference file names, hyphens and spaces are not permitted.

Reads Data

The data from each lane of a 5500 Series SOLiD™ Sequencers instrument results in a single XSQ output file. Different models of the 5500 support up to 6 or 12 lanes, so typically multiple XSQ files are generated per sequencing run. You can direct your LifeScope™ runs to analyze data from specific barcodes in multiple lanes, or group the output of multiple lanes or multiple sequencing runs to be processed as a single specimen.

The type of data used in LifeScope™ Software is called a read-set, which is a collection of reads belonging to one index (barcode) from one XSQ file. The smallest set of data you can specify in the shell is one index from one XSQ file.

You can optionally use groups to combine more than one index into an analysis. Reads that have been combined into a group are processed as one specimen, even if the read-sets come from different input XSQ files. Each group generates a combined set of output files.

In the LifeScope™ Software command shell, input data is added at the analysis level. Input data is associated with individual analyses, not with entire projects.

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. Index IDs and group names give you flexibility in handling your input data. For example:

- Combine specific read-sets from one XSQ file or multiple XSQ files into one group to be analyzed together. All read-sets within a group must be of the same library type.
- Assign read-sets to different groups, to be analyzed separately within one LifeScope™ Software run.

For more information and examples of LifeScope™ Software shell commands to use with your input data, see "Define input data" on page 98.

For information about XSQ data and the XSQ file format, see the following sections:

- "XSQ file format" on page 494, in Appendix C, "File Format Descriptions" on page 493
- Appendix D, "XSQ Tools" on page 519

For information about special character restrictions in XSQ file headers, see "Special character restrictions" on page 504.

Repositories

LifeScope™ Software repositories organize and persist your projects, your input data, and your reference data files.

The reads repository is a storage place in LifeScope™ Software for instrument data intended to be input data for LifeScope™ Software analyses. Data imported into the reads repository are in unmapped XSQ-format files.

Importing reads and references file into LifeScope™ Software repositories is optional, but preferred. The following are two advantages to using the shell's repositories:

- The shell is aware of barcoded read-sets in the read repository. For reads files not in the read repository, you must manually specify barcodes.
- In your shell commands you can use repository names for files in the read and reference repositories. You do not have to use Linux paths to these files.

Files imported into the read and reference repositories are visible to all LifeScope™ Software users. However, each user has their own projects repository. The contents of your projects repository are private and cannot be seen by other users within the command shell.

Note: In the current release, one repository cannot be split across multiple partitions or file systems. (The various repositories can each be on a different partition or file system.)

The reference repository

The reference repository is initially populated during the installation process with reference files for hg18 and hg19.

For more information (not required for most use), see Appendix E, "The Reference Repository" on page 539.

Note: In this release, hyphen ('-') and space characters are not permitted in reference file names.

Analysis space

Mapping and select tertiary modules support the option to have your analysis done in base space or color space (if the space is present in your input data). The analysis space for your runs is controlled by these factors and parameters:

- **Input data –** Whether base space and color space are present in your input data.
- **Secondary processing –** Your setting for the `mapping.in.base` parameter. This parameter controls the analysis space for secondary processing. The run fails if the input data does not have the space specified by the `mapping.in.base` parameter.
- **Tertiary processing –** Your setting for the `analysis.space` parameter.

Use the global workflow parameter `analysis.space` to enter your preference for base space or color space tertiary analysis. Three tertiary modules – SNPs, small indel, and splice finder – use your preference, in conjunction with information about whether color space is present in the BAM file and whether mapping is done in base space or color space, to determine their analysis space.

The recommended usage scenarios for the `analysis.space` parameter are:

- For color space analysis, map in color space and set `analysis.space` to color (or use the default value of auto). The following mapping parameter settings are required (these settings are the defaults):
  - `mapping.in.base=0`
  - `bamgen.refcor.addcs=1`

- For base space analysis, map in base space and set `analysis.space` to base (or use the default value of auto). The following mapping parameter setting is required:
  - `mapping.in.base=1`

An example command to set the `analysis.space` parameter is:

```
set analysis.space color tertiary/global.ini
```

If you create your own INI files, set the `analysis.space` parameter in your `tertiary/global.ini file`. An example setting is:

```
set analysis.space=base
```

The possible conditions determining the analysis space are given in the following table. **The resulting analysis space is …** column refers to the SNPs, small indel, and splice finder tertiary modules. (This table is provided for reference only. Recommended scenarios are given above.)

Table 9 Effect of the analysis.space parameter

With these conditions …			The resulting analysis space is …
Mapping space	**Color present in BAM**	**analysis.space value**	
Base	No	Base	Base
Base	No	Color	Base
Base	No	Auto	Base
Base	Yes	Base	Base
Base	Yes	Color	Color
Base	Yes	Auto	Base
Color	No	Base	Base
Color	No	Color	Base
Color	No	Auto	Base
Color	Yes	Base	Base
Color	Yes	Color	Color
Color	Yes	Auto	Color

Find reference files for an analysis

To find reference files in the references repository, follow these steps:

Log into the shell.

```
cd /references
ls
```

The above example lists reference files that have previously been imported (registered) into the references repository. Reference files can alternatively be specified with an absolute path to the file on the local file system (or navigable from the local file system).

See also the shell command `set reference` *`assembly_name`* in Table 13 on page 89. When used with a known assembly such as hg18 or hg19, this command automates using the correct reference in your analysis.

One reference is supported per analysis run.

Find data for an analysis

To find data in the read-set repository, follow these steps:

Log into the shell.

```
cd /reads
ls
```

The above example lists reads files that have previously been imported (registered) into the reads repository. Reads files can alternatively be specified with an absolute path to the file on the local file system (or navigable from the local file system).

View analysis information

To show the status of your jobs, run a shell status command in the Linux shell. The format of the command depends on whether the analysis is in the project repository or in the current directory.

For an analysis in the project repository, run this command:

```
lscope.sh status -u user -p projectname -a analysisname
```

For an analysis in the current directory, run this command:

```
lscope.sh status
```

An alternate method is to run an `ls` command while in an analysis virtual directory, in the LifeScope™ Software command shell. For example:

```
cd /projects
cd proj1          (This command opens your project named proj1.)
cd run1           (This command opens your analysis named run1.)
ls
--ini--
  secondary/bamstats.ini
  secondary/global.ini
  secondary/fragment.mapping.ini

--readsets--
  lung:DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq:1:DH10B.fasta

--jobs--
  run1 [70%]
```

The `ls` output lists the defined parameters and input read-sets. For running analyses, the output also lists the estimated percentage of completion.

View analysis read-sets

To show the read-sets in an analysis, cd to that analysis in the shell and run an `ls` command. Part of the `ls` output displays the read-sets which have been added to this analysis:

```
--readsets--(group:file:index:reference)
  Group_1:helen_20100402_2_10103F_F3_1.3.1_2.0_converted.bam::
  Group_1:helen_20100402_2_10103F_F3_1.3.1_2.0_converted.bam::
  Group_1:WT_Chr17_Frag.xsq:1:
```

The output includes the group name, file name, index number, and reference file, separated by colons. Except for the file name field, some fields may be empty.

Common scenarios

This section describes how LifeScope™ Software command shell handles common biologic analysis scenarios. New terms and concepts, such as project and analysis, are explained later in this chapter. Specific instructions and commands are given in Chapter 6, Run a Command Shell Analysis.

Basic

A basic scenario is to run a bioinformatics analysis, such as targeted resequencing, on sequence data. The steps to do this in LifeScope™ Software shell are:

1. Create a project.
2. In that project, create an analysis.
3. In that analysis:
   a. Specify the type of analysis to be performed on the data (set the workflow).
   b. Add the sequenced data file.
   c. Specify the reference genome for this analysis.
   d. Run the analysis.

A sample sequenced on multiple lanes

In this scenario one sample is sequenced in multiple lanes on the sequencing instrument. The output data from the sequencing instrument is contained in multiple XSQ files.

In the LifeScope™ Software command shell, the multiple XSQ files are analyzed in one run and as one sample. LifeScope™ Software treats the multiple data files as one set of data.

The steps to do this in LifeScope™ Software shell are:

1. Create a project.
2. In that project, create an analysis.
3. In that analysis:
   a. Specify the type of analysis (such as a standard workflow) to be performed on the data.
   b. Add the XSQ data files. When you add the XSQ files, specify the same group name for each file.
   c. Specify the reference genome for this analysis.
   d. Run the analysis.

Data from multiple samples

This scenario performs, in one run, the same analysis on sequence data from multiple specimens. The data can be in one or multiple XSQ files.

LifeScope™ Software keeps the data from each specimen separate, and each specimen is analyzed separately. The steps to do this in LifeScope™ Software shell are:

1. Create a project.
2. In that project, create an analysis.
3. In that analysis:

a. Specify the type of analysis (such as a standard workflow) to be performed on the data.

b. Add the sequenced data file, specifying different grouping for the data.

c. Specify the reference genome for this analysis.

d. Run the analysis.

Naming restrictions

LifeScope™ Software places certain character restrictions on file names and other names (in addition to the Linux filename conventions), as shown in the following table:

Table 10 Naming restrictions

Use	Character	
	Hyphen ('-')	**Space**
Reference file names	Not allowed	Not allowed
Assembly names	Not allowed	Not allowed
Sample group names	Not allowed	Not allowed
Names of INI and PLN files	Not allowed	Not allowed
Names of input CSFASTA and QUAL files	Not allowed	—
Names of input XSQ files	Not allowed	—
Project names	—	Not allowed
Analysis names	—	Not allowed

For files listed in the table, the restrictions apply to the base file name only, not to paths up to the filename.

In addition, project names and analysis names within LifeScope™ Software must contain only alphanumeric and underscore characters. Project names must not begin with an underscore.

6 Run a Command Shell Analysis

This chapter covers:

- Overview 83
- Run a standard workflow 84
- Use the command shell 85
- Run a grouped analysis 101
- Run a tertiary-only workflow 103
- Run an individual analysis 105
- Resume, re-run, and re-use 106
- Review job status 107
- Review results 108
- Review logs 108
- Error behavior and error messages 108

Overview

The LifeScope™ Software command shell provides a way to quickly run bioinformatics experiments. The shell has its own working environment, which is organized around the concepts of projects and of analyses within projects. The LifeScope™ Software command shell has elements similar to the Linux shell but with a different syntax and interpretation. If you are new to LifeScope™ Software, please review Chapter 5, "Understand The LifeScope™ Software Shell" on page 71 before reading this chapter.

New users are recommended to use the standard workflows built into LifeScope™ Software. Standard workflows are developed with defaults optimized for common analyses. The applications supported by standard workflows are:

- Genomic resequencing
- Targeted resequencing
- Low frequency variant detection
- Whole transcriptome
- Small RNA
- ChIP-Seq
- MethylMiner™

Run a standard workflow

A standard workflow is a built-in series of commonly-used analyses, which correspond to a common biological application. The pre-built standard workflows include genomic resequencing, targeted resequencing, small RNA, ChIP-Seq, MethylMiner™, and whole transcriptome. Using one of the workflows allows you to run an analysis with a minimum of setup.

In the LifeScope™ Software command shell, running a standard workflow requires the following steps:

1. Create a project and analysis.
2. Identify the workflow to be executed on your data.
3. Identify your input data (your reads).
4. Identify the reference genome.
5. For targeted resequencing and low frequency variant workflows, identify the regions of interest file.
6. Issue the `run` command to start your analysis.

Example steps to run a standard workflow

These steps are an example of how to run a workflow in the LifeScope™ Software command shell:

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow targeted.resequencing.frag
# define the input
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq lung
# specify the reference to be used
set reference /data/results/references/DH10B.fasta
# define the any analysis-specific parameters
set analysis.regions.file /data/results/readsdata/dh10b.bed
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
```

Explanation of example steps

Table 11 explains the purpose of the steps shown in "Example steps to run a standard workflow" on page 84.

Table 11 Sample shell commands to run a workflow

Command	Purpose
`lscope.sh shell -u username`	Log into the LifeScope™ Software command shell.
`cd /reads` `import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq`	Go into the read repository and import an XSQ file (the input data file). This command in effect brings the reads file into the shell read repository. Once in the read repository, the reads file is available as input to your analysis runs.

`cd /projects`
`mk ecoli`
`cd ecoli`
	In the projects repository, create a project named `ecoli`, and open that project.

`mk run1` `cd run1`	Create an analysis named `run1`. Open the `run1` analysis. **Note:** The open analysis (`cd run1`, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.
`set workflow targeted.resequencing.frag`	Define the workflow used with this analysis. The workflow in this example includes SAET, mapping, mapping statistics, SNPs, inversion, large indel, small indel, and annotations.
`add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq`	Associate the XSQ file with this analysis. This command directs LifeScope™ Software to use this XSQ file as the input reads data for this analysis. This example uses the repository name for the XSQ file. The absolute path to the XSQ file on the local file system is also supported.
`set reference /data/results/results/references/DH10B.fasta`	Define the reference file for this analysis.
`set analysis.regions.file /data/results/readsdata/dh10b.bed`	Define the regions of interest file, for targeted resequencing (also low frequency variant) workflows.
`ls`	Display the configuration of your analysis, including input reads, reference files, workflow, and select parameter settings.
`run`	Start the analysis run.
`wait`	Wait for the analysis run to complete.
`ls`	Display configuration and status information about the analysis run.

Use the command shell

The LifeScope™ Software shell is a command-line tool for running LifeScope™ Software data analyses. The shell also allows you to create and manage your LifeScope™ Software data, projects, analyses, and users.

If you instead want to run the LifeScope™ Software user interface (UI), refer to the *LifeScope™ Genomic Analysis Software User Guide* (Part no. 4471877).

There are four modes of operation for the LifeScope™ Software command shell:

- **Run mode –** Runs a LifeScope™ Software analysis or workflow from the Linux shell, without entering the LifeScope™ Software command shell. The LifeScope™ Software control file (*plnfile*) must be created and configured before executing a run mode command.

  Example commands:

  ```
  lscope.sh run plnfile -u marie -p proj1 -a run1 -g group1
  lscope.sh resume plnfile -u joe
  ```

  See "Run mode" on page 86.
- **Status mode –** Opens the LifeScope™ Software shell status viewer, which shows the dynamic status of running and completed jobs for specified analysis.

  Example commands:

  ```
  lscope.sh status -p proj1 -a analysis1
  lscope.sh status
  ```

  See "Status mode" on page 87.
- **Scripted mode –** You invoke the LifeScope™ Software shell from commands listed in a file on your local system. See "Scripted mode" on page 88 for information on running in scripted mode.
- **Shell mode –** Enters the LifeScope™ Software command shell. In this shell, only LifeScope™ Software shell commands are accepted. The LifeScope™ Software shell commands are described in Table 13 on page 89. The LifeScope™ Software shell commands allow you to create, manage, and run your LifeScope™ Software analyses.

  Example commands:

  - To enter the shell: `lscope.sh shell -u marie`

  Within the shell:

  - `set workflow whole.transcriptome.frag`
  - `set reference DH10B.fasta`

  See "Shell mode" on page 89.

Note: Do not confuse the LifeScope™ Software shell commands with Linux or UNIX commands. Some commands may have similar names, but the behavior of the LifeScope™ Software shell commands is entirely different from the Linux or UNIX commands. LifeScope™ Software shell commands also depend on the shell level (analysis, project, reads, references, or root) at which they are executed.

Run mode

With the run mode, you enter the LifeScope™ Software command in the Linux shell. You remain in the Linux shell, and the LifeScope™ Software command shell does not open. Run mode also includes the `resume` command.

The following are example run mode commands:

```
lscope.sh run ./splice.finder.pln  -p wt_spliceFinder \
    -a wt_spliceFinder_PE_TC12 -u rjones
lscope.sh resume ./splice.finder.pln  -p wt_spliceFinder \
    -a wt_spliceFinder_PE_TC12 -u rjones
```

The `-p` option specifies the project name. The `-a` option specifies the analysis name. The `-u` option specifies the username.

A `-g groupsfile` parameter is optional. When a groupsfile is provided, LifeScope™ Software performs multi-group analysis, which involves analyzing the data for each group independently. See "Run a grouped analysis" on page 101 for more information about multi-group analyses.

When to use run mode

An `lscope.sh` run mode command is appropriate when you have already configured you analysis. Prerequisites to running an `lscope.sh` run mode command include:

- The LifeScope™ Software web server and license server must be running.
- The LifeScope™ Software user account must exist.
- The reads files and reference files required by the analysis must have been transferred to or available to the LifeScope™ Software cluster or workstation.
- Reference files, reads files, and read-sets must either have been specified in a previous LifeScope™ Software command shell session, or specified by absolute paths in the analysis PLN and INI files.
- The analysis PLN and INI files are configured correctly.

Run "in the projects repository" or "in the local file system"

A `run` command that includes the project and analysis options is considered to "run in the projects repository". The shell projects repository determines the output file location, and the analysis run is visible and trackable from the LifeScope™ Software UI.

Without the project and analysis options, a `run` command is considered to "run in the local file system". Output files are created in the current directory on the file system, and the run is tied to the current directory for checking status.

Table 12 compares the two methods.

Table 12 Comparison of running in the projects repository and in the local file system

Category	Run in the projects repository	Run in the local file system
Command format	lscope.sh run plnfile -u user -p proj -a analysis -g groupsfile	lscope.sh run plnfile -u user Must be executed in the local file system in the same directory as plnfile.
Status command	lscope.sh status -u user -p proj -a analysis	lscope.sh status Must be executed in the same directory of the local file system.
Output file location	In the /projects repository	In the current directory of the local file system.
Visible in the UI?	Yes. You can view the analysis configuration, reads files, parameters, status, and results files in the UI.	No.

Status mode

The `lscope.sh status` command displays information about your running LifeScope™ Software analyses.

To check the status of the example run described in "Run mode", enter this command in the Linux shell:

```
lscope.sh status -u rjones -p wt_spliceFinder \
```

```
-a wt_spliceFinder_WT_SF_PE_TC0012
```

If your `run` command is running in the local file system, use one of the following commands to check status. The format depends on whether or not your analysis is in the projects repository:

- For an analysis in the project repository, run this command:
  `lscope.sh status -u user -p projectname -a analysisname`
- For an analysis in the current directory, run this command:
  `lscope.sh status`
  The following is an example of the `lscope.sh status` screen:

```
Status (t=toggle subjobs, arrows=scroll, q=quit view):     [running]

TASK                                    STATUS       PROGRESS INFO
human_hg18-pair.mapping                 started      0%
  human_hg18-pair.mapping.1.splittin    completed    2%       Pair End Mapping.doSpl
  human_hg18-pair.mapping.2.run         started      7%       mapreads
  human_hg18-pair.mapping.3.run         started      8%       mapreads
  human_hg18-pair.mapping.4.run         started      8%       mapreads
  human_hg18-pair.mapping.5.run         started      7%       mapreads
human_hg18-PairEndMapping.MarkDupl      not started  %
human_hg18-PairEndMapping.BAMStats      not started  %
solid0064_20101210_MP_2X60_T3_1_F3      not started  %
solid0064_20101210_MP_2X60_T3_1_F3      not started  %
solid0064_20101210_MP_2X60_T3_1_F3      not started  %
solid0064_20101210_MP_2X60_T3_1_F3      not started  %
solid0064_20101210_MP_2X60_T3_1_F3      not started  %
solid0064_20101210_MP_2X60_T3_1_F3      not started  %
solid0064_20101210_MP_2X60_T3_1_F3      not started  %
solid0064_20101210_MP_2X60_T3_1_F3      not started  %
solid0064_20101210_MP_2X60_T3_1_F3      not started  %
```

You must be in the correct directory on your local file system (the directory from which you executed the `run` command).

(For another status command, see Table 13. The shell `ls` command, when executed with the shell in an analysis level, gives the status of an analysis run.)

Scripted mode

The LifeScope™ Software command shell accepts as input a text file containing shell commands.

1. Create a text file with one command per line. For example:

```
cd proj1
cd run2
set workflow analysis.pln
add xsq xsqfile
set reference hg19
run
ls
exit
```

2. Save the file, for example as `proj1script`.

3. Then run with this command:

```
lscope.sh -u user -w password < proj1script
```

Note: This example does not include the commands necessary to step up a new analysis.

Shell mode

In shell mode, you enter the LifeScope™ Software command shell. In this shell, only LifeScope™ Software shell commands are accepted. The LifeScope™ Software shell commands allow you to create, manage, and run your LifeScope™ Software analyses.

Terminology

Table 7 on page 72, in Chapter 5, Understand The LifeScope™ Software Shell, defines concepts and terminology used with the LifeScope™ Software command shell. Review these terms before proceeding to the shell commands and their syntax.

lscope.sh commands

Table 13 describes the commands available within the LifeScope™ Software command shell. The behavior of some shell commands depends on the level at which they are executed, as noted in Table 13.

Table 13 LifeScope™ Software shell commands

Command	Description
Project and analysis commands:	
mk *name*	Creates a new project or a new analysis within the current project. Must be executed at the `/projects` or `/projects/`*`projname`* levels. When executed at the `/projects` level, creates a new project named *`name`*. When executed at the `/projects/`*`projname`* level, creates a new analysis named *`name`*, within your project *`projname`*. Command examples: • At the `/projects` level: `mk testproj` • At the `/projects/`*`projname`* level: `mk run1` See "Project and analysis naming rules" on page 97 for restrictions on *name*.

Table 13 LifeScope™ Software shell commands *(continued)*

Command	Description

ls	Lists information about your project or analysis.
	When executed at the `/projects` level, lists the names of all your projects.
	At the `/projects/`*`projname`* level, lists the names of all analyses contained within your project named *`projname`*.
	At the `/projects/`*`projname/analysisname`* level, provides configuration and progress reporting for the analysis:
	• Lists INI files involved in the analysis, if `set workflow` has been given.
	• Lists read-sets involved in the analysis, if read-sets have been added. The read-set listing includes the group, file name, index, and reference.
	• Lists the status of jobs for the analysis.
	Example:
	`ls`
	`--ini--`
	`fragmentMapping.ini`
	`global.ini`
	`--readsets--(group:file:index:reference)`
	`Group_1:case0030_Fragment.xsq:1:chr16.validated.fasta`
	`--jobs--`
	`run1 [100%] [JOB_SUCCESS]`
	`fragmentMapping [100%] [JOB_SUCCESS]`
	Example two:
	`ls`
	`--ini--`
	`secondary/global.ini`
	`secondary/smallrnamapping.ini`
	`tertiary/global.ini`
	`tertiary/srCount.ini`
	`tertiary/srCoverage.ini`
	`--readsets--(group:file:index:reference)`
	`HumanFrag_smRNA:reads.xsq:1:hg1_1b.fasta`
	`--jobs--`
	`smallRNAworkflow [100%] [JOB_SUCCESS]`
	`secondary-hg1_1b-smallrnamapping [100%] [JOB_SUCCESS]`
	`tertiary-HumanFrag_smRNA-srCount [100%] [JOB_SUCCESS]`
	`tertiary-HumanFrag_smRNA-srCoverage [100%] [JOB_SUCCESS]`

Table 13 LifeScope™ Software shell commands *(continued)*

Command	Description
rm *name*	Permanently deletes a project or an analysis within the current project. Must be executed at the `/projects` or `/projects/`*`projname`* levels. When executed at the `/projects` level, deletes the project named *`name`*. When executed at the `/projects/`*`projname`* level, creates a new analysis named *`name`*, within your project *`projname`*. Command examples: • At the `/projects` level: `rm testproj` • At the `/projects/`*`projname`* level: `rm analysis2` **WARNING!** This command permanently deletes the underlying project and analysis directories on the server. Recovery is *not* supported.

set workflow *path*	Sets the workflow or PLN file for the current analysis. This command is a prerequisite to running an analysis. The `set workflow` command must also precede any `set` *`param value`* commands.
	Examples:
	`set workflow targeted.resequencing.frag`
	`set workflow /share/lifescope/human/test1/my.pln`
	The `set workflow` command replaces the manual configuration of PLN and INI files used in earlier versions of BioScope™ Software. With this command the LifeScope™ Software shell automatically creates the required PLN and INI files.

get workflow	Copies the current workflow's INI files, PLN files, and their directory structure to the local file system. These files are copied to the directory you are in when you give the `lscope.sh` command. You must have a shell analysis open (be in the analysis virtual directory) and that analysis must have a workflow already set (with the `set workflow` command).

ls	When executed at the level of a specific standard workflow, such as `/workflows/small.rna`, lists the INI files for the workflow. For example:
	`server:/workflows> cd small.rna`
	`server:/workflows/small.rna> ls`
	`secondary/bamstats.ini`
	`secondary/global.ini`
	`secondary/smallrnamapping.ini`
	`tertiary/filtermapping.bamstats.ini`
	`tertiary/global.ini`
	`tertiary/srCount.ini`
	`tertiary/srCoverage.ini`

Reads file (input data) commands:	

Table 13 LifeScope™ Software shell commands *(continued)*

Command	Description
import *path* [*reponame*]	When executed at the `/reads` level, imports a source XSQ file from the Linux file system into the reads repository (and updates the read repository index). The import command imports to the reads repository top folder, not to a subfolder. When executed at the `/bams` level, imports a BAM input data file into the BAM repository (for tertiary analysis). The `/bams` virtual directory is reserved for BAM files that have been converted from earlier versions of the software or imported from another installation of the same version of LifeScope™ Software. The *path* argument is one of the following: • The absolute path to the file on the Linux file system (recommended), *or* • A relative path to the file, using as the current directory the Linux directory in which the `lscope.sh shell` command was given. The *reponame* argument is optional, and allows you to assign a repository name different from the filename. The file appears as *reponame* when the repository contents are displayed, and is accessed as *reponame* with commands such as `add xsq` *reponame*. The `import` command may be one of these forms: • **import-ln** – Links the file (with `ln -s`) given by *path* to the repository directory on the LifeScope™ Software server. Default. • **import-cp** – Copies the file given by *path* to the repository directory on the LifeScope™ Software server. • **import-mv** – Moves the file given by *path* to the repository directory on the LifeScope™ Software server.
add xsq *name* [*:id*] [*group*]	The `add xsq` command specifies input for secondary analysis. It adds the read-sets with index (barcode id) *id* in the XSQ file specified by *name* into the current analysis. `add xsq` must be run at the analysis level. *name* must match either a reads file that has been imported into the reads repository or the full path to an XSQ file on the local file system. The `ls` command run at the `/reads` level lists available read repository files. *id* is optional. If no *id* is provided, all indices (barcodes) from the XSQ file are added. Each *id* refers to the data for one index. *group* is optional. If *group* is provided, all read-sets are grouped together, and analyzed together as one specimen. Read-sets from multiple XSQ files may be added the same *group* name. If *group* is not provided with indexed data, the following occurs: • Read-sets are grouped by index name from the XSQ file. • The read-sets from each index (barcode) are processed independently. All read-sets in a group are analyzed together as one specimen. The group name is not required to match the specimen name or other information from the instrument run. Do not use spaces or special characters in a group name. Multiple `add xsq` commands are supported per analysis.
del xsq *filename*	Removes the XSQ file specified by *filename* from the current analysis. *filename* must appear in the listing of the read repository (as listed by the LifeScope™ Software shell `ls` command at the `/reads` level). This command does not physically delete the XSQ file from your file system.

Table 13 LifeScope™ Software shell commands *(continued)*

Command	Description
rm xsq *name*	Removes the XSQ file *name* from the /reads repository. *name* is the file's repository name. If the XSQ file *name* was originally imported with *import-ln*, *rm* deletes the link in the repository directory on the local file system. If the XSQ file *name* was originally imported with *import-cp* or *import-mv*, *rm* deletes the file from the repository directory on the local file system.
set groups *path*	Creates read-set groups from the file given by *path*. Read-sets belonging to one group are analyzed together as one specimen. Each group generates an independent set of results files (containing data combined for the whole group). LifeScope™ Software can process multiple groups in one analysis run, while treating each group's data as an independent run. The groups file must be a tab-separated text file with multiple lines in the following format: `groupname <path>/XSQorBAMfile:indexid <path>/reference_file` *path* must be an absolute path to the groups file on (or available to) the local file system. If no index id is specified, it is assumed to be "1". All read-sets from the file are included. If the reference column entry does not exist, the reference from the previous line in the file is used. The XSQ, BAM, and reference files may be specified by their repository names, if the files have been imported into a LifeScope™ Software shell repository. If these files are specified by a path on the local file system, an absolute path is recommended. Relative paths are interpreted relative to the directory where you open the LifeScope™ Software command shell with the `lscope.sh` shell command.
add bam *path* [*group*]	(*Optional*) Adds BAM file *path* into the current analysis. Use this command to specify input for tertiary analysis. *path* must match either a BAM file that has been imported into the bams repository or an XSQ file on the local file system. The `ls` command run at the `/bams` level lists available bams repository files. *group* is optional. If no *group* is provided, *group* is assumed to be "Default". All BAM files in a group are analyzed together as one specimen.
del bam *name*	Removes the BAM file *name* from read-sets to be analyzed. This command does not physically delete the BAM file from your file system.
rm bam *name*	Removes the BAM file *name* from the /bams repository. *name* is the file's repository name. If the BAM file *name* was originally imported with *import-ln*, *rm* deletes the link in the repository directory on the local file system. If the BAM file *name* was originally imported with *import-cp* or *import-mv*, *rm* deletes the file from the repository directory on the local file system.
cat bam *path*	Displays header information for the BAM file given by *path*. *path* must be the absolute path to a BAM file on the file system.

ls	When executed at the `/reads` level, lists the XSQ files in the read repository. When executed at the `/bams` level, lists the files in the BAM repository. The library type and data type (base space or color space) are included in the listing. Example output at the `/reads` level is:
	`smallindel_simulation_2INS_22_25.xsq [Fragment] [color]`
	`AllMapBarcodes.xsq [MatePair] [mixed] [Barcoded]`
	`DH10B_ColorSpaceOnly_50_Frag.xsq [Fragment] [color] [Non-barcoded]`
	`DefaultLibrary_PE.xsq [PairedEnd] [color] [Non-barcoded]`
	`Fragment3panels.xsq [MatePair] [mixed] [Non-barcoded]`

Table 13 LifeScope™ Software shell commands *(continued)*

Command	Description

cat *xsqfile*	In `/reads`, the `ls` command with an XSQ file argument displays the libraries and barcodes contained in the XSQ file. For example:
	`server:/reads> ls Indexing3panels.xsq`
	`[lib:Library1] [idx:1] [MatePair] [50,50] [mixed]`
	`[lib:Library1] [idx:13] [MatePair] [50,50] [mixed]`
	`[lib:Library1] [idx:5] [MatePair] [50,50] [mixed]`
	`[lib:Library1] [idx:9] [MatePair] [50,50] [mixed]`
	`[lib:Library2] [idx:10] [MatePair] [50,50] [mixed]`
	`[lib:Library2] [idx:14] [MatePair] [50,50] [mixed]`
	`[lib:Library2] [idx:2] [MatePair] [50,50] [mixed]`
	`[lib:Library2] [idx:6] [MatePair] [50,50] [mixed]`
	`[lib:Library3] [idx:11] [MatePair] [50,50] [mixed]`

Reference file commands:	
set reference *assembly* | *path*	*assembly* names an assembly in the LifeScope™ Software shell reference repository. The LifeScope™ Software shell sets reference-related parameters to appropriate default values. An example assembly name command is: `set reference hg18` *path* sets the genome reference to the reference file *path* in current analysis. *path* can either be an absolute filepath (on the Linux file system), or a reference name that exists in the shell reference repository (as listed by the LifeScope™ Software shell `ls` command at the `/references` level). If *assembly* names a recognized assembly in the reference repository, then the following reference-related parameters are automatically set to default values for *assembly*: • `annotation.dbsnp.file` • `annotation.gtf.file` • `analysis.mirbase.precursor.file` • `analysis.mirbase.mature.file` • `analysis.filter.reference` • `analysis.assembly.name` • `analysis.species.name` The set values can be changed with the `set` *param.key param.value* command, if necessary. A `set reference` command (either *assembly* or *path*) is mandatory before running an analysis. The use of *assembly* is recommended. A maximum of one reference is allowed per analysis. Run the LifeScope™ Software shell `ls` command to review the parameter settings. Before starting your analysis, confirm that these parameters are set correctly.
Parameter commands:	

Table 13 LifeScope™ Software shell commands *(continued)*

Command	Description

set *param.key param.value* [*inifile*]	Sets a parameter for the analysis. The analysis workflow must be defined (with the `set workflow` *arg* command) before you set a parameter for that analysis. For example:
	`set analysis.regions.file` *`absolute_path/regions_bed_file`*
	`set annotation.dbsnp.file` *`absolute_path/dbsnp_file`*
	`set annotation.gtf.file` *`absolute_path/refgene_gtf_file`*
	`set analysis.filter.reference` *`absolute_path/filter_reference_file`*
	`set analysis.mirbase.mature.file` *`absolute_path/mature_forms_file`*
	`set analysis.mirbase.precursor.file` *`absolute_path/precursor_forms_file`*
	`set analysis.assembly.name name_of_reference_assembly`
	`set analysis.species.name name_of_reference_species`
	Some parameters are automatically pre-populated based on the reference chosen. See "List shell commands" on page 97.
	System parameters may also be set. For example:
	`set scratch.dir /scratch/lifescope`
	The *ini* argument is optional. When *ini* is specified, the parameter is set for an individual INI file within the analysis. For example, the following command sets the value of the parameter `bamstats.wig.primary.only` to `0`, for only the BAMStats module:
	`set bamstats.wig.primary.only 0 secondary/bamstats.ini`
	The *ini* value is case sensitive. The secondary or tertiary location of the INI file is required.
	If the *ini* argument is not provided, the parameter is added to both the secondary and tertiary `global.ini` files.
	The *inifile* version of the command supports turning off certain analysis modules. This method is not recommended for a module that has dependencies in the PLN file. The following example turns off the bamstats module:
	`set bamstats.run 0 secondary/bamstats.ini`
	Note: The `set workflow` command must precede any `set` *`param value`* commands.
	Note: Do not use an equals sign '=' with the `set` command.

cat *inifle*	Lists parameters for the current analysis, including ones defined with the `set` *param value* [*inifile*] command. Example:
	`cat pair.mapping.ini`
	`default.maximum.insert.size=20000`
	`default.minimum.insert.size=150`
	`pair.mapping.run=1`

del *param inifile*	Undefines a parameter from an INI file. The *inifile* value is case sensitive. If no INI file is specified, deletes the parameter from all global.ini files. If an INI file is specified, the secondary or tertiary location of the INI file is required. The following example deletes the definition of the parameter `analysis.input.readset.file` from a mapping INI file. The parameter `analysis.input.readset.file` must not be set if the SAET module is not run. `del analysis.input.readset.file secondary/fragment.mapping.ini`
Execution commands:	
run	Begins execution of the current analysis.
wait	Waits for the currently running analysis to complete. If the current analysis not running, returns immediately. Must be run in an analysis directory.
stop	Stops execution of the current analysis.

Table 13 LifeScope™ Software shell commands *(continued)*

Command	Description
resume	Resumes execution of the current analysis, if that analysis or a portion of it is in a failed or stopped state. Modules that successfully completed execution are not re-run. Job resumption is at the module level. A module that partially completed but then failed is restarted from the beginning by the `resume` command. The shell resume command can also be issued in the Linux shell, as a run mode command. The syntax is: `lscope.sh resume plnfile -u user -p project -a analysis`
Navigation commands:	
cd /	Navigates to the root level of the LifeScope™ Software command shell. May be executed at any shell level. **Note:** '/' is the only argument allowed. You may not specify multiple levels separated by one or more '/' characters.
cd ..	Navigates up one level (one virtual directory) in the shell. **Note:** The `cd` command navigates only one level at a time.
cd *name*	Navigates down one level (one virtual directory) in the shell. When executed at the shell root level (`/`), opens the category named *name*. Allowed categories names are `bams`, `projects`, `reads`, `references`, `users`, and `workflow`. **Note:** Only admin users have access to the `users` category. When executed at the `/projects` level, opens your project named *name*. At the `/projects/`*projname* level, opens your analysis named *name*. At the `/workflows` level, opens the standard workflow named *name*. **Note:** The `cd` command navigates only one level at a time.
Administrative user commands: **Note:** Only admin users have access to the `users` category. Must be executed at the `/users` level. These commands require admin role privilege.	
ls	Lists all users.
mk *name*	Creates a new user named *name*, with the password *name* and the role `user`.
rm *name*	Deletes the LifeScope™ Software shell user named *name*.
loggedin	Shows logged in users.
set username *role* [password]	Sets a user's role and optionally set the user's password. Valid roles are: admin, user
makepasswdfile [passwordfile]	Creates an encrypted password file for non-interactive login. By default, the file is named `.lscope.passwd`, and is created in your home directory. Use the optional `passwordfile` parameter to create the password file with a non-default name or location. When the `lscope.sh shell` command is given without a password, LifeScope™ Software looks for a `.lscope.passwd` file in the user's directory. If the file exists, `lscope.sh` uses its password. This allows the command `lscope.sh -u` *username* to log the user in without asking for password. To use a password file of a different name or location, use the `-e` option: `lscope.sh -e passwordfile`

Table 13 LifeScope™ Software shell commands *(continued)*

Command	Description
Other administrative commands: These commands require admin role privilege.	
rebuild	In `/reads`, rebuilds the read repository index, which is used to display the list of read-sets in the repository. Recent changes (additions or deletions) to the reads repository are not reflected in the `ls` output until the index is rebuilt.
Miscellaneous commands:	
config	Displays configuration information for the LifeScope™ Software server.
help	Briefly lists the LifeScope™ Software shell commands and syntax. The man page is more verbose. (The man page is access with the `lscope.sh man` command, issued in the Linux shell.)
exit	Logs out of the LifeScope™ Software web server, and exits the shell. Running analyses are not affected.
!*cmd*	Runs a Linux shell command. Example: `!pwd` *Do not* use !*cmd* to run a Linux `lscope.sh` command from within the command shell.

Project and analysis naming rules

Project and analysis names must conform to Linux filename conventions:

- Contain only alphanumeric and underscore characters.
- Do not contain spaces.
- Must be unique (among siblings).
- Do not begin with an underscore character. (This requirement is a LifeScope™ Software limitation, not a Linux rule.)

Command completion and command history

The LifeScope™ Software command shell supports command completion, with the keyboard Tab key. The command shell supports command history, with the keyboard up arrow.

List shell commands

There are two methods to list shell commands and command syntax.

- On the Linux command line, enter this command: `lscope.sh man`
  This command lists the `lscope.sh` man page. This version is more verbose than the help output within the command shell.
- In the LifeScope™ Software command shell, enter this command: `help`
  Help lists the shell commands, with their syntax and with a brief description.

Options

The options `lscope.sh` command supports are described in Table 14.

Table 14 lscope.sh options

Option	Description
-a	Analysis name

Table 14 lscope.sh options

Option	Description
-e *passwordfile*	The command shell uses the password in *passwordfile* on startup. This option is not required when you use the default password file name and location. See the makepasswdfile [passwordfile] command.
-g	Group name.
-p	Project name.
-u	Username.

For information on analysis name and project name see Table 7, "Concepts related to the LifeScope™ Software command shell" on page 72.

For information on groups, see the following:

- "Define input data"
- "Run a grouped analysis" on page 101

Define input data

This section describes adding reads input data to one analysis. The reads input can either be in the read repository or specified by absolute path to a file on the local file system.

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. Index IDs and group names give you flexibility in handling your input data.

Index IDs

The shell supports two methods of specifying index IDs with your reads data. Both the `add xsq` command and groupsfiles support index IDs. Using an index ID restricts the input to only the data for a specific index. For example:

```
add xsq xsqfile:62
```

Groups processing

Combining reads data into groups causes the data to be treated as one specimen, with one set of results files that represents the combined data for the group.

Groups are created with the optional group field of the `add xsq` command, or with a groupsfile and either the `set groups` command or the `-g groupsfile` option of the `lscope.sh run` command. See "Run a grouped analysis" on page 101 for an example of a groupsfile.

The following is an example of an `add xsq` command with a group field:

```
add xsq xsqfile:62 Human
```

Example input data scenarios

The following sections describe scenarios for defining your input data.

Define the input reads for a sequencing library instance

Suppose your specimen is sequenced on the instrument with three indices (barcodes). You require that LifeScope™ Software treat the data from these three indices as one specimen and generate one set of results files for the specimen. You combine your reads data into a group to achieve this effect. The following are two methods to combine these indices into one group.

These examples imply the data is in one XSQ file, but that is not required (`XSQfile` does not have to be the same XSQ file for all the indices).

The add xsq command

If the indices for your specimen are 3, 4, and 9, and you choose the group name Human for your specimen's data, the following commands cause your data to be analyzed as one specimen:

```
add xsq XSQfile:3    Human
add xsq XSQfile:4    Human
add xsq XSQfile:9    Human
```

A groupsfile

A groupsfile also associates the specimen's indices with a group, and is more convenient for large numbers of indices. Here are example contents of the groupsfile:

```
Human <path>/XSQfile:3 <path>/reference_file
Human <path>/XSQfile:4
Human <path>/XSQfile:9
```

Use this groupsfile with either the `set groups` shell command or the `-g` option of the `lscope.sh run` command:

```
set groups <path>/groupsfilename
lscope.sh run ... -g <path>/groupsfilename
```

Define reads in separate groups

Contrast the following treatment of reads data with the example in "Define the input reads for a sequencing library instance" on page 99:

```
add xsq XSQfile:3    Human
add xsq XSQfile:4    Rat
add xsq XSQfile:9     Ecoli
```

In this example the analysis is run in three groups, with each group containing the data from one index. All groups receive the same analysis modules processing (for example, mapping, SNPs, and CNV). The data from each index is processed separately, and each index generates an independent set of results files.

Non-indexed data

When you use non-indexed XSQ files, all read-sets in those files become your input. (If you use a group name that includes other read-sets, the new read-sets are combined with the other read-sets already assigned to that group.)

Both the `add xsq` command and a groupsfile can add non-indexed data to your analysis:

```
add xsq xsqfile1 Human
add xsq xsqfile2 Human
```

```
Human <path>/XSQfile1 <path>/reference_file
Human <path>/XSQfile2
```

The effect of not using the `id` index argument with indexed data

Consider what happens if you add multiple indexed XSQ files to an analysis and do not use the `id` argument to specify individual indices. The effect is very different when you assign a group and you do not assign a group.

Example with a group

```
add xsq xsq1 groupX
add xsq xsq2 groupX
add xsq xsq3 groupX
```

For example, the above `add xsq` commands, with a group specified, define as input all the indices in the three XSQ files. All read-sets (in all three XSQ files) are treated as one specimen, and in the output files the data from all the read-sets is combined.

Example without groups

If you do not specify the group name, the shell assigns the data to groups based on index name. For example, consider the following commands, assuming each XSQ file contains indexed data:

```
add xsq xsq1
add xsq xsq2
add xsq xsq3
```

The shell assigns the read-sets to groups based on index name, using group names of `Group_`*id*. Each index is treated as a different specimen, the data for different indices are kept separate, and a separate output directory is used for each index. If the same index name occurs in more than one XSQ file, the read-sets for those indices are combined into one group. This combination might not be what you intend. For example, if files xsq1, xsq2, and xsq3 each contains indices 1–96, the data is processed as 96 separate groups. Group 1 contains index 1 from all 3 XSQ files, group 2 contains index 2 from all three XSQ files, and so on.

Granularity

The most specific set of data that can be defined as input to an analysis run is the data from one index in one XSQ file. For example:

```
add xsq xsqfile:62
```

Index IDs

Using the correct index IDs is important in your data input commands. Use the LifeScope™ Software shell `cat` command to find the index ID for your data. For example:

```
cat CS_BC16_PE50X25.xsq
```

Example output (truncated) is shown below. The index id appears in the "`idx:`" field.

```
[lib:Library1] [idx:1] [PairedEnd] [50,25] [color]
[lib:Library1] [idx:10] [PairedEnd] [50,25] [color]
[lib:Library1] [idx:11] [PairedEnd] [50,25] [color]
[lib:Library1] [idx:12] [PairedEnd] [50,25] [color]
```

WARNING! XSQ files created with ICS versions earlier than 1.1.1 allow the index ID and index name to be different. Mistakenly using the index name in place of the index ID gives erroneous results.

Run a grouped analysis

This section describes the behavior of a grouped analysis. If an analysis has read-set groups associated with it, the following jobs are run:

- One secondary sub-analyses: one job for all read-sets.
- A number of tertiary sub-analyses: one job per read-set group.

The following mechanisms add read-set groups to an analysis:

- The `group` option of the `add xsq` command.
- The `set groups` command.
- The `-g groupsfile` option of the `lscope.sh run` command.

Example of a groupsfile

The following gives an example of a basic groupsfile:

```
Human /data/reads/xsqfile1.xsq:3 <path>/reference_file1
Human /data/reads/xsqfile1.xsq:21
Human /data/reads/xsqfile2.xsq:51
```

The pattern is: *groupname XSQfile:id referencefile*. These field are separated by tabs.

The *referencefile* field is optional, except for the first entry of a group. If the *referencefile* field is empty, the last reference specified is used. All read-sets within one group must use the same reference (and the groupsfile must contain only one reference).

Run a multi-group analysis

This section describes how to do a run that analyzes reads from more than one sample, using a groupsfile that specifies multiple groups. All read-sets within the analysis must use the same reference.

Multi-group groupsfile

With a multi-group analysis, a single run of LifeScope™ Software processes the input data from more than one group. For each group, LifeScope™ Software keeps the data, processing, and results separate.

The following is an example groups file for a multi-group analysis:

```
lung /data/reads/xsqfile1.xsq:1 /refs/reference_file
lung /data/reads/xsqfile1.xsq:5
lung /data/reads/xsqfile1.xsq:13
```

```
brain /data/reads/xsqfile1.xsq:2
brain /data/reads/xsqfile1.xsq:6
brain /data/reads/xsqfile1.xsq:9
brain /data/reads/xsqfile1.xsq:10
brain /data/reads/xsqfile1.xsq:14
kidney /data/reads/xsqfile1.xsq:3
kidney /data/reads/xsqfile1.xsq:11
kidney /data/reads/xsqfile1.xsq:15
```

This read-set file defines three sample groups:

- lung – barcodes 1, 5, and 13
- brain – barcodes 2, 6, 9, 10, and 14
- kidney – barcodes 3, 11, and 15

With this read-set file, LifeScope™ Software in one run performs the processing of three separate runs, one for each group. All groups in the groupsfile must use the same reference, and the reference must be specified on the first line of the file.

Example commands for a multi-group analysis

This section describes the steps to run a multi-group analysis. The groupsfile used in "Multi-group groupsfile" is suitable for this example.

Follow these steps to run the multi-group analysis:

1. Login to lscope shell:
   `lscope.sh shell -u user`
2. Create a project and analysis, and open the analysis:
   ```
   cd /projects
   mk proj1
   cd proj1
   mk run1
   cd run1
   ```
3. Set the groups:
   `set groups path_to_groupsfile.`
4. (*Optional*) Set these parameters:
   `set analysis.species.name species`
   `set analysis.assembly.name name`
   An example of a species name is `ecoli`. An example of an assembly name is `DH10B`.
   If the assembly name is not provided, an assembly name of `unknown` is used.
   If the species name is not provided, the name of the reference file is used.
5. Ensure that the INI files in your analysis *do not* set the `analysis.sample.name` parameter. If this parameter is set in your INI files, the LifeScope™ Software shell overrides each sample group it processes with the value from the INI file.
6. Launch the multi-sample analysis:
   `run`

7. Optionally check the status of the analysis:

   `ls`

   This alternate method to check an analysis status is entered on the Linux shell:

   `lscope.sh status -u user`

Run a tertiary-only workflow

When you run a standard workflow and specify only BAM input files (not XSQ files), the framework automatically runs only the tertiary portion of the workflow. The secondary (mapping) portion of the workflow is not attempted.

Two examples are given, one with `add bam` commands and one with the `set group` command and a groupsfile.

With add BAM commands

The following is an outline of the shell commands to run only the tertiary portion of a workflow:

```
cd projects
mk tertiaryProj
cd tertiaryProj
mk tertiaryRun
cd tertiaryRun
set workflow genomic.resequencing.frag
add bam /path/to/bamfile1 group1
add bam /path/to/bamfile2 group1
add bam /path/to/bamfile3 group1
set reference /path/to/reference
run
```

The `genomic.resequencing.frag` workflow includes both secondary and tertiary analyses. The secondary analyses take as input XSQ files, and the tertiary analyses take BAM files as input. Because the above example has only BAM files as input, LifeScope™ Software automatically skips the secondary analyses. the workflow runs only the tertiary analyses.

With a groupsfile

A tertiary-only workflow can also be run using a groupsfile and the `set groups` command. For example, outside of the LifeScope™ Software shell, create a groupsfile named `tertiarygroups` with this content:

```
group1 /path/to/bamfile1 /path/to/reference
group1 /path/to/bamfile2
group1 /path/to/bamfile3
```

The following is an outline of the LifeScope™ Software shell commands to run only the tertiary portion of a workflow, using the groupsfile:

```
cd projects
mk tertiaryProj
cd tertiaryProj
mk tertiaryRun2
cd tertiaryRun2
set workflow genomic.resequencing.frag
set groups ./tertiarygroups
```

```
set reference /path/to/reference
run
```

With these commands, the groupsfile `tertiarygroups` must be in the current directory, which is the directory you are in when you give the `lscope.sh` command.

Run a multi-group tertiary analysis

This section describes how to run a tertiary-only workflow using BAM file input and using groups. With a multi-group analysis, a single run of LifeScope™ Software processes the input data from more than one group. For each group, LifeScope™ Software keeps the data, processing, and results separate.

BAM files with different library types can be combined as long as the workflow and tertiary modules can process them.

Two examples are given, one with `add bam` commands and one with the `set group` command and a groupsfile.

With add bam commands

This example uses `add bam` commands to create 2 groups:

```
set workflow yourworkflow
add bam /path/to/bamfile1 group1
add bam /path/to/bamfile2 group1
add bam /path/to/bamfile3 group1
add bam /path/to/bamfile4 group2
add bam /path/to/bamfile5 group2
set reference /path/to/reference
run
```

With a groupsfile

This example uses a groupsfile to create the same 2 groups. Outside of the LifeScope™ Software shell, create a groupsfile with this content:

```
group1 /path/to/bamfile1 /path/to/reference
group1 /path/to/bamfile2
group1 /path/to/bamfile3
group2 /path/to/bamfile4
group2 /path/to/bamfile5
```

These example LifeScope™ Software shell commands use the groupsfile:

```
set workflow yourworkflow
set groups /path/to/groupsfile
set reference /path/to/reference
run
```

Run an individual analysis

The optional examples download includes an examples of how to run selected LifeScope™ Software analyses individually, without being part of a standard workflow or other sequence of analyses. Users new to LifeScope™ Software are recommended to use either a standard workflow or one of the optional examples, instead of the procedures described in the rest of this section.

This section describes how to run a single LifeScope™ Software analysis module, using the command shell.

Some options required you to create or edit PLN and INI files. See Appendix F, "Command Shell Control Files" on page 553 for information about these files.

- If you have already run the analysis, and the command shell has created the underlying PLN and INI files. You can use these PLN and INI files to run the analysis.
- You can optionally configure the analysis independently outside of the shell, using INI and PLN files that you create yourself.

This section describes these methods of running a single module:

- **In shell mode –** Open the LifeScope™ Software command shell, make a project and analysis, configure the analysis, and run the analysis in the command shell.
- **In run mode –** Use the `lscope.sh run` command to start the analysis. To use this command the PLN and INI file must already be created and customized. Two possibilities for creating the PLN and INI files are:
  - If you have already run the analysis in the command shell, then LifeScope™ Software has created these files for you.
  - You can optionally create the PLN and INI manually, for instance with a text editor.

The run mode command you use to run the analysis depends on whether you are running in the project repository or in the local file system.

- **Run in the project repository –** The run command is in this format:

  ```
  lscope.sh run plnfile -u user -p project -a analysis \
                                -g groupsfile
  ```

  When you run in the projects repository with the `lscope.sh run` command, the results files are handled by the shell project repository. Where the results files appear on your local file system is configured during installation, and is given by the `lifescope.projects` parameter in the `<installdir>/server/server.properties` file.
- **Run in the local file system –** The run command is in this format:

  ```
  lscope.sh run plnfile -u user
  ```

  When you run in the local file system with the `lscope.sh run` command, the results files appear in a subdirectory to your current directory on the file system. This command is executed in the same directory as *plnfile*.

Resume, re-run, and re-use

This section explains analysis reuse concepts and supported functionality, for both the command shell and GUI interfaces.

- **Resume –** A stopped job is restarted, without repeating modules that have successfully completed. The command shell also supports resuming a failed job (the GUI does not).

  Note: With the `resume` command, every module that is in progress at the time the job is stopped (or fails) is processed from the beginning of that module.

- **Re-use –** An existing analysis is used as the basis for a new analysis (with a new analysis name). You can optionally change the configuration of the new analysis.
- **Re-run –** An analysis is repeated (with the same configuration). All processing is redone.

Table 15 Comparison of resume, re-use, and re-run functionality

Purpose	Interface		Comments
	GUI	Command shell	
Resume	Use the **Analysis > Resume...** command. Only stopped jobs are supported.	Use the shell `resume` command. Supports stopped jobs and failed jobs.	—
Re-use	Use the **Analysis > Reuse...** command. **Note: Reuse**... is available only for analyses with status of Running, Completed, or Failed.	Supported indirectly: Use Linux commands to make a copy of the analysis in the project directory, under a new analysis name.	You may optionally change the configuration of the new analysis.
Re-run	Supported indirectly: use the **Analysis > Reuse...** command, and run the same analysis content under the new analysis name.	Not recommended, because the new run overwrites output from the previous run. See "Re-run an analysis in the command shell". Supported, but *not recommended.* Restart the analysis (same as the initial run).	You may optionally change the analysis configuration before restarting.

Note: If all secondary processing completed successfully and the resulting BAM files are correct, use the BAM input option (not the XSQ input option) when re-using or re-running a job. If you use the XSQ option, then all secondary processing is repeated. With the BAM option, only tertiary processing is done.

Re-run an analysis in the command shell

When you re-run an analysis in the command shell, log files are appended and generated files are overwritten:

- Log files are appended. Log entries for the different runs can be distinguished by their time stamps.
- All output files are overwritten, and the output of the previous run is lost. Overwritten files include GFF output, generated BAMs, statistics files, and others.

The mechanism (*not recommended*) to re-run an analysis in the command shell is the following (these steps assume that the project and analysis are created and configured, and that the analysis previously ran to conclusion):

1. Open the project, and `cd` to the analysis.
2. Enter the `run` command, and press **Enter** or **Return**.

Review job status

The ways to check on your run vary according to how you started your run, as shown in the following table.

Table 16 Methods to review job status

How you started your run	How to check the status

In LifeScope™ Software command shell, with the shell `run` command.	Open the LifeScope™ Software shell, cd to your project and then to your analysis, and execute the shell `ls` command:
	Log in to the LifeScope™ Software command shell.
	`cd /projects`
	`cd proj1`
	`cd run1`
	`ls`
	Output similar to the following is displayed:
	`--ini--`
	`secondary/bamstats.ini`
	`secondary/fragment.mapping.ini`
	`secondary/global.ini`
	`tertiary/dibayes.genome.ini`
	`--readsets--`
	`lung:DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq:1:DH10B.fasta`
	`--jobs--`
	`run1 [70%]`
	Output for completed jobs also has a success or failure notice:
	`run1 [100%] [JOB_FAILURE]`, or
	`run1 [100%] [JOB_SUCCESS]`

In the Linux shell, with either the `lscope.sh run` or `lscope.sh resume` command, *and* the analysis is running in the projects repository.	At the Linux prompt, enter a `lscope.sh status` command, with the `-p project` and `-a analysis` options:
	`lscope.sh status -u rjones -p project \`
	`-a analysis`

In the Linux shell, with either the `lscope.sh run` or `lscope.sh resume` command, *and* the analysis is running in the current directory.	At the Linux prompt, enter a `lscope.sh status` command. Do not specify the -p project and -a analysis options: `lscope.sh status -u rjones`

Review results

Reviewing the output files generated by your the LifeScope™ Software analyses is performed outside of the LifeScope™ Software command shell. You can either inspect your results files on the Linux cluster or transfer the results files to your local machine to review.

If your analysis is run in the projects repository, you can view results and download results files using the LifeScope™ Software UI. Refer to the *LifeScope™ Genomic Analysis Software User Guide* (Part no. 4471877).

If your analysis is not run in the projects repository, you must find your results files on the local file system. The results files are under the following directory:

<projects_repo_dir>/*<user_name>*/*<project_name>*/*<analysis_name>*/outputs

where *projects_repo_dir* is the location of the shell projects repository on the local file system, and *user_name*, *project_name*, and *analysis_name* are your shell user id, project name, and analysis name. Contact your LifeScope™ Software administrator for the location of the shell projects repository. The projects repository directory is also given by the `lifescope.projects` parameter in the file *<installdir>*/`server/server.properties`.

Review logs

You can typically use the LifeScope™ Software shell status commands (see "Review job status" on page 107) to determine your jobs success or failure status and to examine any error messages.

In the event you also want to look at your job's run logs, by default they are located in the LifeScope™ Software shell projects repository. The location of this repository is set at installtime and is specific to your installation.

If you run your analysis in the current directory, your logs are in that directory.

For successful runs, the last line of `summary.log` ends in:

```
Analysis finished successfully.
```

When `lscope.sh` finishes, it prints the status of each job. If an error occurs, then the job's summary log is also printed.

Error behavior and error messages

Case sensitivity

LifeScope™ Software shell levels, names, and commands are case sensitive. For example, the following sequence shows that a project named `D157` exists, and the command `cd d157` (with incorrect case) is not successful.

```
~/projects> ls
D152
D157
~/projects> cd d157
Server returned error status: 404, message: Project d157 does
not exist
```

```
~/projects> cd D157
~/projects/D157>
```

Unrecognized commands

The LifeScope™ Software shell display an error message "Invalid command" for unrecognized commands. In the following examples, the `ls` command is first mistyped and then given in the wrong case. The string `mkdir proj1`, while a correct Linux command, is not recognized by the LifeScope™ Software command shell. (The correct LifeScope™ Software shell command is `mk proj1`.)

```
~/projects> lss
Invalid command
~/projects> LS
Invalid command
~/projects> mkdir proj1
Invalid command
~/projects>
```

Illegal usage

```
Error: String index out of range: -1
```

The message is thrown when the shell command is recognized but an illegal or unrecognized argument is used. Remember that shell commands do not behave like Linux commands. For example, the LifeScope™ Software shell `ls` command has a specific function, to list the contents of the current level. When mistakenly used to list the contents of a child or parent level, or used with wildcards, the error message appears:

```
~/projects> ls
D152
D157
~/projects> ls D157
Error: String index out of range: -1
~projects/D157/> ls ..
Error: String index out of range: -1
~projects/D157/> ls D1*
Error: String index out of range: -1
```

7 Run a Standard Workflow Analysis

This chapter covers:

- Overview 111
- Standard workflows 111
- Run a standard workflow 118
- Create a new workflow 121
- Workflow control files 123
- Run the tertiary portion of a workflow 125

Overview

A standard workflow is a built-in sequence of experiments that combine multiple LifeScope™ Software modules into a single command shell run. A workflow's analyses correspond to a common biological application, such as resequencing.

Standard workflows

The standard workflows are provided for the following analysis types:

- Targeted resequencing
- Genomic resequencing
- Low frequency variants
- Whole transcriptome
- Small RNA
- ChIP-Seq
- MethylMiner™

Note: The standard workflow directory also includes two file conversion utilities for XSQ and BAM conversion. The conversion utilities are for internal use and do not work on the command line or in the command shell.

The workflows and the modules they execute are described in Table 17.

Table 17 Standard workflow libraries, locations, and modules

Analysis type	Library type	Workflow name	LifeScope™ Software modules involved
ChIP-Seq	Fragment	chip.seq	Secondary: • SAET • Fragment mapping • Mapping statistics
Genomic resequencing	Fragment	genomic.resequencing.frag	Secondary: • (*Optional*) SAET • Mapping • Mapping statistics
			Tertiary: • SNPs • Human CNV • Small indels • Annotations
Genomic resequencing	Mate-pair	genomic.resequencing.lmp	Secondary: • (*Optional*) SAET • Paired mapping • Mark duplicates • Mapping statistics
			Tertiary: • SNPs • Human CNV • Inversions • Large indels • Small indels • Annotations

Table 17 Standard workflow libraries, locations, and modules *(continued)*

Analysis type	Library type	Workflow name	LifeScope™ Software modules involved
Genomic resequencing	Paired-end	genomic.resequencing.pe	Secondary: • (*Optional*) SAET • Paired mapping • Mark duplicates • Mapping statistics
			Tertiary: • SNPs • Inversions • Large indels • Human CNV • Small indels • Annotations
Low frequency variant	Fragment	low.frequency.variant.frag	Secondary: • Fragment mapping • Mapping statistics
			Tertiary: • Enrichment • SNPs • Annotations
Low frequency variants	Paired-end	low.frequency.variant.pe	Secondary: • Paired mapping • Mark duplicates • Mapping statistics
			Tertiary: • Enrichment • SNPs • Annotations
Methyl Miner	Fragment	methyl.miner.frag	Secondary: • (*Optional*) SAET • Fragment mapping • Mapping statistics
Methyl Miner	Paired-end	methyl.miner.pe	Secondary: • (*Optional*) SAET • Paired mapping • Mark duplicates • Mapping statistics

Table 17 Standard workflow libraries, locations, and modules *(continued)*

Analysis type	Library type	Workflow name	LifeScope™ Software modules involved
Small RNA		small.rna	Secondary: • Small RNA mapping • Mapping statistics
			Tertiary: • Small RNA count • Small RNA coverage • Filter mapping counts
Targeted resequencing	Fragment	targeted.resequencing.frag	Secondary: • SAET • Fragment mapping • Mapping statistics
			Tertiary: • Enrichment • SNPs • Small indels • Annotations
Targeted resequencing	Paired-end	targeted.resequencing.pe	Secondary: • SAET • Paired mapping • Mark duplicates • Mapping statistics
			Tertiary: • Enrichment • SNPs • Small indels • Annotations
Whole transcriptome	Fragment	whole.transcriptome.frag	Secondary: • WT exon sequence extractor • WT splice junction extractor • WT fragment mapping • Mapping statistics
			Tertiary: • WT counts • WT coverage • Splice finder

Table 17 Standard workflow libraries, locations, and modules *(continued)*

Analysis type	Library type	Workflow name	LifeScope™ Software modules involved
Whole transcriptome	Paired-end	whole.transcriptome.pe	Secondary: • WT exon sequence extractor • WT splice junction extractor • WT paired-end mapping • Mapping statistics
			Tertiary: • WT counts • WT coverage • Splice finder
—	—	bam.convert	For internal use only.
—	—	xsq.convert	For internal use only.

Note: For improved accuracy in the SNPs module, turn on the BAMStats parameter `bamstats.enable.probe.position`, to generate position error files and probe error files:

```
set bamstats.enable.probe.position 1 secondary/bamstats.ini
```

Turn off SAET

If you turn off SAET in a workflow, make sure your mapping INI file does not contain the line:

```
analysis.input.readset.file = ${analysis.output.dir}/saet/*.rrs
```

Comment out this line if it appears in your mapping INI file.

List standard workflows

List workflows using the LifeScope™ Software command shell

This section describes two methods of listing the standard workflows available on your installation of LifeScope™ Software.

Workflows are listed in the LifeScope™ Software command shell by doing an `ls` command in the `/workflows` virtual directory. To display a list of the standard workflows,

1. Open the LifeScope™ Software command shell:
   `lscope.sh -u` *`username`* `[-w` *`password`*`]`
2. Enter the workflow level:
   `cd workflows`
3. List the available workflows:
   `ls`
   The workflows are listed.

List workflows using Linux commands

The LifeScope™ Software standard workflows are installed under the directory `<installdir>/etc/workflows`. To list the standard workflows, run this command in the Linux shell:

```
ls <installdir>/etc/workflows
```

List a workflow's INI files and parameters

The command shell provides commands to list a workflow's control files and parameters. The `ls` command, when run in at the level of a standard workflow, lists its INI files. The `cat` command is used to list the parameter settings of one of the workflow's INI files. The shell commands are shown in bold in the following example:

```
server:/workflows> cd small.rna
server:/workflows/small.rna> ls
secondary/bamstats.ini
secondary/global.ini
secondary/smallrnamapping.ini
tertiary/filtermapping.bamstats.ini
tertiary/global.ini
tertiary/srCount.ini
tertiary/srCoverage.ini
server:/workflows/small.rna> cat secondary/bamstats.ini
  bamstats.input.dir=${analysis.output.dir}/smallrnamapping
  bamstats.run=1
```

Workflow directory structure

All standard workflows follow a consistent directory structure, shown in Table 18.

Table 18 Workflow directory and file layout

Contents of top-level workflow directory	Contents of sub-directories
analysis.pln	—
secondary	secondary.pln
	global.ini
	module1.ini
	module2.ini ...
	moduleN.ini
tertiary	tertiary.pln
	global.ini
	module1.ini
	module2.ini ...
	moduleN.ini

Do not delete or rename any of the following directories or files for your workflow:

- The `secondary` directory
- The `tertiary` directory
- The `secondary.pln` file
- The `tertiary.pln` file
- The `analysis.pln` file

Workflow modules and parameters

The analysis parameters used in the workflow modules control the behavior of the modules' algorithms. Table 19 lists where the module parameters are described in this guide.

Table 19 Location of parameter descriptions

Workflow	Module	Chapter	Section	Page
Genomic resequencing	Mapping	Chapter 9	"Fragment mapping parameters"	149
	Mapping statistics	Chapter 9	"Mapping statistics"	167
	SNPs	Chapter 10	"SNPs runtime parameters"	191
	CNV	Chapter 11	"CNV module parameter descriptions"	215
	Small indel	Chapter 13	"Small indel parameter description"	241
	Annotations	Chapter 15	"Annotation parameters" See also the SNPs, CNV, and small Indel chapters.	301
Targeted resequencing	SAET	Chapter 8	"SAET parameters"	131
	Mapping	Chapter 9	"Fragment mapping parameters"	149
	Mapping Statistics	Chapter 9	"Mapping statistics"	167
	Enrichment	Chapter 16	"Targeted resequencing read selection algorithm"	329
	SNPs	Chapter 10	"SNPs runtime parameters"	191
	Small indel	Chapter 13	"Small indel parameter description"	241
	Annotations	Chapter 15	"Annotation parameters" See also the SNPs, and small Indel chapters.	301
Low frequency variants	Mapping	Chapter 9	"Fragment mapping parameters"	149
	Mapping Statistics	Chapter 9	"Mapping statistics"	167
	Enrichment	Chapter 16	"Targeted resequencing read selection algorithm"	329
	SNPs	Chapter 10	"SNPs runtime parameters"	191
	Annotations	Chapter 15	"Annotation parameters" See also the SNPs, and small Indel chapters.	301
Small RNA	Small RNA mapping	Chapter 23	"Small RNA mapping parameters"	433
	Mapping statistics	Chapter 23	"Mapping statistics"	425
	Small RNA coverage	Chapter 24	"Small RNA coverage parameters"	458
	Small RNA counts	Chapter 25	"Small RNA counts parameters"	464

Table 19 Location of parameter descriptions *(continued)*

Workflow	Module	Chapter	Section	Page
Whole transcriptome	WT mapping	Chapter 19	"Single-read mapping parameters", "Paired-end parameters"	362, 375
	Mapping statistics	Chapter 19	"Mapping statistics"	383
	WT coverage	Chapter 20	"WT coverage parameters"	400
	WT counts	Chapter 21	"WT counts parameter description"	404
	WT splice finder	Chapter 22	"Splice finder parameters"	410
ChIP-Seq	Mapping	Chapter 9	"Fragment mapping parameters"	149
	Mapping statistics	Chapter 9	"Mapping statistics"	167
MethylMiner™	Mapping	Chapter 9	"Fragment mapping parameters"	149
	Mapping statistics	Chapter 9	"Mapping statistics"	167

Run a standard workflow

Prepare to run a standard workflow

This section explains the preparation required to run a standard workflow.

Complete the prerequisites

Complete the applicable prerequisites described in Chapter 4, "Before You Begin" on page 67.

Determine your input and reference files

Before you can run the standard workflows you must know:

- The read and references files required by your analyses
- The regions of interest file, for targeted resequencing and low frequency variant analyses
- Any non-default module parameter values

Order commands correctly

Shell commands must be ordered in the following sequence:

1. Create the project.
2. Create the analysis.
3. Set the workflow.
4. Add the read-sets.
5. Set the reference.
6. Set the assembly name, species name, and other parameters, if required.

These rules apply to the analysis configuration commands:

- The `set workflow` command must precede all other analysis configuration commands.
- Read-sets must be added before the reference is set. (These commands add read-sets to your analysis: `set groups` *path*, `add xsq` *path*, and `add bam` *path*.)
- The `set reference` command must precede these two commands:
  - `set analysis.assembly.name`
  - `set analysis.species.name`

Specify the regions of interest file

Only for targeted resequencing and low frequency variant workflows, you must provide the regions of interest file:

```
set analysis.regions.file path
```

An absolute path is recommended for *path*. A relative path is interpreted relative to the directory you are in, in the Linux shell, when you execute the `lscope.sh` command.

Note: Targeted resequencing and low frequency variant analyses requires the regions of interest file, but the command shell does not enforce this requirement. Be sure you specify a regions of interest file when using a targeted resequencing or low frequency variant workflow.

Run a workflow in the command shell

This section describes how to run the fragment targeted resequencing workflow in the LifeScope™ Software command shell.

In the LifeScope™ Software command shell, running a standard workflow requires the following steps:

1. Create a LifeScope™ Software command shell project and analysis.
2. Identify the workflow to be executed on your data.
3. Identify your input data (your reads).
4. Identify the reference genome.
5. For targeted resequencing workflows, identify the regions of interest file.
6. Issue the `run` command to start your analysis.

Example steps to run a standard workflow

This section shows the steps to run a workflow in the LifeScope™ Software command shell. These steps use the targeted resequencing fragment workflow as an example. A description of each step is given in Table 20.

```
# log into the shell
lscope.sh shell -u username -w password
# import your data into the reads repository
cd /reads
import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
```

```
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow targeted.resequencing.frag
# define your input data
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
# specify the reference to be used
set reference hg19
# optionally change parameter defaults after this line or
# set analysis-specific parameters
set analysis.regions.file /data/results/readsdata/dh10b.bed
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
```

Paths (`/data/results`) and filenames are examples only. In your actual workflow runs you use paths to your own data and reference files.

Table 20 explains the purpose of the steps shown above. **Information about your**

Table 20 Description of sample shell commands to run a workflow

Command	Purpose
`lscope.sh shell -u username`	Log into the command shell. The shell prompts for your password if you do not provide it with the `-w` option.
`cd /reads` `import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq`	Go into the reads repository and import an XSQ file (the input data file). This command brings the XSQ reads file into the LifeScope™ Software reads repository.

`cd /projects`
`mk ecoli`
`cd ecoli`
`mk run1`
`cd run1`
	In the projects repository, create a project named `ecoli`, and in that project create an analysis named `run1`. Open the `run1` analysis.
	Note: The open analysis (`cd run1`, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.

`set workflow targeted.resequencing.frag`	Define the workflow used with this analysis. The workflow in this example includes SAET, mapping, mapping statistics, enrichment, SNPs, small indel, and annotations.
`add xsq DH10B_Unbarcoded_50_Frag.xsq`	Define the XSQ file containing the input reads data for this analysis. Multiple input files are allowed.
`set reference /data/results/results/references/DH10B.fasta`	Define the reference file for this analysis.
`set analysis.regions.file /data/results/readsdata/dh10b.bed`	Define the regions of interest file, for targeted resequencing.
`ls`	Display the configuration of your analysis, including input reads, reference files, workflow, and select parameter settings.
`run`	Start the analysis run.

Table 20 Description of sample shell commands to run a workflow *(continued)*

Command	Purpose
`ls`	Display status information about the analysis run.

Information about your workflow job

The output of the `ls` command displays the analysis' INI files, read-sets, and run completion percentage or completion status. Example output is shown below:

```
--ini--
   fragmentMapping.ini
   global.ini

--readsets--(group:file:index:reference)
   Group_1:case0030_Fragment.xsq:1:chr16.validated.fasta

--jobs--
   run1 [100%] [JOB_SUCCESS]
      fragmentMapping [100%] [JOB_SUCCESS]]
```

Run a workflow in lscope.sh run mode

The `lscope.sh` run mode command executes the workflow without opening the LifeScope™ Software command shell. This command is used at the Linux command prompt.

The following command runs the small RNA workflow in run mode:

```
lscope.sh run <path>/small.rna/analysis.pln  -p rna \
-a run1 -u sjones
```

where *path* is the path to the workflow's location in the LifeScope™ Software installation directory. The project and analysis are created if they do not already exist.

As an alternative, you can copy an existing workflow to a directory of your choice (use a Linux command such as `cp -p` to preserve the directory structure, or use the command shell `get workflow` command). Then use the following version of the command:

```
lscope.sh run ./analysis.pln  -p rna -a run1 -u sjones
```

See "Use the command shell" on page 85 for more information about run mode.

Create a new workflow

This section describes how to create a new workflow. The new workflow appears in the shell `ls` command (but not in the LifeScope™ Software UI list of workflows). You run the new workflow following the directions for a standard workflow. The new workflow must have a unique name that both follows Linux file naming conventions and also does not start with an underscore character.

Note: When you create your own workflow, you can view the analysis and results files in the LifeScope™ Software UI. But the LifeScope™ Software UI does not support other analysis activities, such as reuse, with a custom workflow created on the command line.

This section describes two methods to create a new workflow, one with a Linux `cp` command, and the other with the shell `get workflow` command.

With the Linux `cp` command

Using the Linux `cp` command method requires that you know the location of the LifeScope™ Software installation directory on the local file system. You do not open the LifeScope™ Software command shell.

The create a new workflow, follow these steps:

1. Identify an existing workflow similar to the one you want to create.
2. Change directory to the `<installdir>/etc/workflows` directory.
3. Create a directory with the name of your new workflow:

   ```
   mkdir newworkflow
   ```

   You use this name to define your analysis type with the `get workflow` command, and the workflow appears as this name when you list the contents of the `/workflows` virtual directory.
4. Copy an existing workflow into the new directory. Use a command such as `cp -p` in order to copy the directory structure as well as the files.

   ```
   cp -p small.rna/* newworkflow
   ```
5. Edit the PLN and INI files in `newworkflow` to customize it according to the requirements of your analysis.

   Do not rename the `secondary` or `tertiary` directory, and do not rename any of the PLN files (`analysis.pln`, `secondary.pln`, or `tertiary.pln`).

With the shell `get workflow` command

The shell `get workflow` command copies the current workflow's INI files, PLN files, and directory structure to your current directory on the local file system. This method requires you to have (or create) a LifeScope™ Software command shell analysis configured with the shell `set workflow` command.

The create a new workflow, follow these steps:

1. Identify an existing workflow similar to the one you want to create.
2. Log into the LifeScope™ Software command shell. Cd to your project and open the analysis. Or, if the project and analysis do not exist, create them with commands such as the following:

   ```
   cd /projects
   mk myproj
   cd myproj
   mk run1
   cd run1
   ```
3. If the analysis does not already have a workflow defined, use the shell `set workflow` command and name the workflow whose files you want to copy. For example:

   ```
   set workflow genomic.resequencing.pe
   ```
4. Use the `get workflow` command. For example:

   ```
   server:/projects/demo/getwf> get workflow
   Analysis workflow saved to /home/dir/demo/getwf
   ```

This commands pulls the workflow files from the server and copies them to the your local file system. The destination directory is determined by the Linux current working directory when the command shell was opened (where the `lscope.sh shell` command was given).

The following files and directories are copied:

`analysis.pln`

`secondary` (including its PLN and INI files)

`tertiary` (including its PLN and INI files)

5. Edit the copied PLN and INI files to customize them according to the requirements of your analysis.

Do not rename the `secondary` or `tertiary` directory, and do not rename any of the PLN files (`analysis.pln`, `secondary.pln`, or `tertiary.pln`).

Workflow control files

This section describes the files that define the behavior of a standard workflow. The fragment genomic resequencing workflow, `genomic.resequencing.frag`, is used as an example. This section describes the workflow's directory structure and the purpose of its PLN and INI files.

Do not edit INI or PLN files for the standard workflows. Such changes affect all workflow users, including both command shell and GUI users.

PLN files

Each standard workflows contains PLN files of the same name and at the same relative location. These PLN files are described in Table 21.

Table 21 PLN files in a standard workflow

File name and relative location	File contents and explanation
<*workflow_name*>/analysis.pln	`secondary.pln` `tertiary.pln < secondary.pln`
	First, runs the modules listed in the file `secondary.pln`. If the `secondary.pln` runs complete successfully, then the modules listed in the file `tertiary.pln` are run.
<*workflow_name*>/secondary/secondary.pln	Lists the INI files involved in secondary analysis for this workflow.
	Runs the LifeScope™ Software modules specified in these INI files.
<*workflow_name*>/tertiary/tertiary.pln	Lists the INI files involved in tertiary analysis for this workflow.
	Runs the LifeScope™ Software modules specified in these INI files.

INI files

Table 22 shows the INI files involved in the genomic resequencing standard workflow, along with the PLN files that direct the execution of the workflow. The `global.ini` files define parameters that are imported into one or more module INI files in the same directory. (These parameters are considered global for the single directory, not for the entire workflow.)

Table 22 PLN and INI files in a standard workflow

File name and relative location	File contents and explanation
<*workflow_name*>/analysis.pln	`secondary.pln` `tertiary.pln < secondary.pln`
	First, runs the modules listed in the file `secondary.pln`. If the `secondary.pln` runs complete successfully, then the modules listed in the file `tertiary.pln` are run.

<*workflow_name*>/secondary/secondary.pln	`saet.ini`
	`fragment.mapping.ini < saet.ini`
	`bamstats.ini < fragment.mapping.ini`

	Runs SAET using the parameters in the file `saet.ini`. If SAET completes successfully, runs fragment mapping using the parameters in the file `fragment.mapping.ini`. If the fragment mapping module completes successfully, runs bamstats mapping statistics.

<*workflow_name*>/tertiary/tertiary.pln	`dibayes.genome.ini`
	`cnv.ini`
	`small.indel.ini`
	`annotation.dibayes.ini < dibayes.genome.ini`
	`annotation.cnv.ini < cnv.ini`
	`annotation.small.indel.ini < small.indel.ini`

	Runs, in order, the (1) SNPs, (2) CNV, and (3) small indel modules, using the parameters in the respective INI files. If the SNPs module completes successfully, runs SNPs annotations. If the CNV module completes successfully, runs CNV annotations. If the small indel module completes successfully, runs small indel annotations.

.run parameters

The `*.run` parameters control which LifeScope™ Software modules execute. The following are examples of run parameters used in standard workflows:

```
dibayes.run=1
small.indel.run=1
cnv.run=1
```

The allowed values for run parameters are 0 and 1:

- 0: Do not run the module.
- 1: Run the module analysis (default).

Because mapping is a prerequisite for all other runs, `fragment.mapping.run` may not be set to 0 unless it has already been successfully run to completion. You may set the .run parameters other modules to 0, under these conditions:

- You do not want to run that module's analysis.
- That module's analysis has already been completed (for instance, if you are restarting the workflow run and repeating a particular module is not necessary).

The run parameters are set automatically by the shell during standard workflows. Accept the default for most use.

Run the tertiary portion of a workflow

This section describes running only the tertiary part of a standard workflow. With this method, you can optionally use input data with mixed library types.

First, set a standard workflow, such as `targeted.resequencing.pe`. You use one or more `add bam` commands to define the input data for the tertiary analysis. When run, LifeScope™ Software automatically only runs the tertiary portion of the workflow, because there are no XSQ read-sets in the analysis.

The following are partial example steps:

```
set workflow targeted.resequencing.pe
add bam file1.bam
add bam file2.bam
add bam file3.bam
set reference referencefile1
run
```

In this scenario, you can optionally combine BAM files of different library types. The type of data you have determines which type of standard workflow you must use. Follow these guidelines for the workflow to specify:

- If your input includes mate-pair data, use a `.lmp` workflow. Only the `genomic.resequencing.lmp` workflow supports mate-pair libraries.
- If your input does not include mate-pair data, and does include paired-end data, use a `.pe` workflow.
- If your input does not include any paired libraries, use a `.frag` workflow.

Note: The large indel and inversion modules do not support input BAM files containing mixed types.

PART IV
Analysis Modules

8 Run the SOLiD® Accuracy Enhancement Tool

This chapter covers:

- Overview 129
- SAET usage guidelines 129
- Examples of how to run SAET 130
- SAET input files 131
- SAET parameters 131
- SAET output files 134
- Algorithm description 135

Overview

The SOLiD® Accuracy Enhancement Tool (SAET) improves color call accuracy prior to mapping. This method implements a proprietary spectral alignment error correction algorithm that uses quality values and properties of color space. Significantly improving its performance over conventional error correction methods, the use of SAET produces more accurate mapping with both an increased number of reads with zero miscalls and an increased number of mapped reads. The SNP calling on error-corrected data results in an increase in true positive calls and a decrease in the number of false positive calls. Using the *de novo* assembly on error-corrected reads results in an increase that is as much as three to five times higher in average contig length.

SAET performance was tested on various datasets, including a large spectrum of genome sizes and complexities, coverages, read lengths, and experiment types. SAET shows similar performance on datasets sequenced from enriched regions or genomes of 1 kbp to 3 Gbp, with coverage from 10 to 4000x and a read length of 25 to 75 bp.

On average SAET requires up to 5 GB RAM for each 1 Gbp of sequenced region. SAET can be applied for diploid organisms. SAET should not be used for rare variant detection datasets where the frequency of rare variant may be less that twice that of the color calling error rate.

SAET usage guidelines

SAET is recommended for use with targeted resequencing and is on by default in targeted resequencing standard workflows. SAET has not been validated with other experiment types and may have insufficient memory to run. If you do choose to run SAET with other experiment types, you can do the following:

- Run SAET before you run the resequencing or whole transcriptome mapping and pairing tools.
- Use the SAET output with resequencing and whole transcriptome mapping and pairing.
- Use updated quality values file for SNP calling.
- Avoid the generation of the updated quality values, if the downstream analysis modules in your workflow do not use quality values. Your SAET runtime is reduced approximately by half if you do not update quality values.

Examples of how to run SAET

Workflows using SAET

The following standard workflows provide examples of running SAET:

- targeted.resequencing.frag
- targeted.resequencing.pe

The optional examples download includes an example of how to run SAET by itself, as an individual analysis that is not part of a standard workflow.

See the following sections for information on these methods:

- As part of a workflow: Chapter 7, "Run a Standard Workflow Analysis" on page 111
- As an individual analysis: Appendix H, "Demo Analyses" on page 567

SAET example in a workflow

The targeted resequencing Fragment standard workflow gives an example of running SAET on with targeted resequencing analyses. The analyses are:

- SAET
- Mapping
- Enrichment
- SNPs
- Small indels
- Annotations

SAET error corrects reads prior to mapping. This correction significantly improves mapping quality and the quality of variant calls.

Note: If you turn off SAET, make sure your mapping INI file does not contain the line:

```
analysis.input.readset.file = ${analysis.output.dir}/saet/*.rrs
```

Comment out this line if it appears in your mapping INI file.

SAET usage in standard workflows

SAET is on by default in the targeted resequencing workflows.

In the genomic resequencing, CHiP-Seq, and MethylMiner™ workflows, SAET is off by default. To enable SAET in these workflows, change the parameter `saet.run` from `0` to `1` with either one of the following methods:

- If you edit the INI file, change `saet.run=0` to `saet.run=1`.
- If you configure the workflow in the command shell, use the following shell command (*after* setting the workflow):

```
set saet.run 1 secondary/saet.ini
```

SAET input files

By default, SAET takes an input file of SOLiD® system reads in XSQ format, and an expected length of the sequenced region, such as the:

- Genome size for whole genome sequencing
- Size of enriched region in targeting resequencing
- Size of transcribed region in whole transcriptome sequencing

The required input file is in XSQ format, which is typically generated by 5500 Series Genetic Analysis Systems . The file must contain reads in color-space that require correction. SAET constructs a spectrum of all *k*-mers from the set of all reads. Then each read is corrected individually if necessary. (The value *k* is the seed length. The default can be overridden with the parameter `saet.seed`.)

In targeted resequencing, the SAET module accepts as an input the BED, GFF, or TXT file with segments of target regions. In this case the `refLength` parameter is computed from the targeted regions information.

SAET parameters

This section describes the parameters that control the runtime behavior of the SAET module. In order to change a parameter value, use the `set param` shell command. For example, to turn off `qvupdate`, change the value of the parameter `saet.qvupdate` with this shell command:

```
set saet.qvupdate 0 secondary/saet.ini
```

Table 23 SAET parameter descriptions

Parameter name	Default value	Description
Required parameters		
saet.refLength	0	Expected length of sequenced (or enriched) DNA region. For example, 4,600,000 for the E.Coli 4.6 MB genome or 30,000,000 for the entire Human Transcriptome. Allowed values: Integers >= 0. The default value for saet.refLength varies by workflow. For example: • Default in genomic resequencing workflows: 2800000000 • Default in targeted resequencing workflows: 30000000
Parameters required for targeted resequencing		
saet.on.target.ratio	0.5	The expected ratio of reads that come from the targeted region. Allowed values: Floats 0.0–1.0.
Optional parameters		

Table 23 SAET parameter descriptions

Parameter name	Default value	Description
saet.qvupdate	1	Whether or not to update the quality value of modified calls. Allowed values: • **0**: Do not update the quality value of modified calls. This option is approximately twice as fast. • **1**: Update the quality value of modified calls. Required for SNPs.
saet.qvhigh	25	Correction is applied to calls with a quality value below the value of this parameter. Allowed values: Integers >= 1.
saet.seed	0	Size of *k*-mer (>=5) used in spectrum construction and error correction. (If set to 0, then the value is computed internally.) Allowed values: Integers 0, 5-28.
saet.suppvotes	3 for targeted reseq; 2 for other analyses	The minimum number of *k*-mer votes required to make a correction. Allowed values: Integers >= 1. For datasets forming coverage less than 50x, the recommended value is 1 or 2. For high coverage data, the value can be increased up to k-1. Increasing this value results in a lower number of over-corrections in repetitive regions and a lower number of corrections over all. The maximum recommended value is 7.
saet.trustfreq	0	The lowest multiplicity of the seed to be included in the spectrum. (If set to 0, then the value is computed internally.) Allowed values: Integers >= 0.
saet.localrounds	0	Maximum number of allowed corrections per read. (If set to 0, then the value is set to [readLength/8]). Reduce if over-corrections are observed, or increase if under-corrections are observed. Allowed values: Integers 0–9.
saet.globalrounds	1 for targeted reseq and reseq of large genomes; 2 for general reseq of small genome	The error correction step is repeated the provided number of times. Reduce if over-corrections are observed, or increase if under-corrections are observed. Allowed values: 1, 2, or 3.
saet.trustprefix	0	Position in the read at which the error rate inflates, for instance, 35-40 for 50bp long reads. If set to 0, then the value used is 0.8*readLength. Allowed values: Integers >=k, where *k* is the size of the *k*-mer (either set by saet.seed or computed internally).
saet.nosampling	0	Disables random sampling in spectrum building. If set to 0, then for large datasets (coverage > 300x), a subset of reads is used in spectrum building. Allowed values: • **0**: Do not disable random sampling in spectrum building. • **1**: Disables random sampling in spectrum building.
Resource parameters		
processors.per.node	8	Number of parallel threads used to run SAET on a node.
saet.minreads.per.node	1000000	If SAET is run on multiple nodes, then each node receives at least this amount of reads to process.

Table 23 SAET parameter descriptions

Parameter name	Default value	Description
memory.request	15GB	Maximum amount of memory available on the node. **IMPORTANT!** SAET Accuracy Enhancement Tool requires 20GB to run on large genomes, such as experiments sequencing whole human genomes. With the minimum memory (15MB), SAET may not run on whole large genomes, and in this case the original reads are used for mapping. For applications such as targeted resequencing, SAET runs with the minimum memory. If SAET does not have enough memory, it issues a warning and allows mapping to continue with the original uncorrected reads.

Explanation of parameters

The processing time and the level of error correction aggressiveness depend mainly on the values defined for `saet.localrounds` and `saet.globalrounds`. The `saet.localrounds` parameter allows you to correct up to `saet.localrounds` errors in a read by using a precomputed spectrum. The `saet.globalrounds` parameter recomputes the spectrum after each global round, and allows you to correct up to `saet.localrounds` errors in a read based on the precomputed spectrum.

SAET is designed to reduce the error rate in the reads generated by the SOLiD® System. Reducing the error rate increases the number of mapped reads for resequencing analyses, which can for example lead to an increase in true positive SNP-calls and a decrease in false positive SNP-calls. (If the SAET is used for SNP calls, the `saet.qvupdate` parameter must be set to 1.) Use the `saet.qvhigh` parameter to restrict corrections to positions that have a quality value below the `saet.qvhigh` threshold.

The optional parameters available in SAET allow you to modify the algorithm's behavior. For example, you might find that the globally computed cutoff for frequency of trusted seeds does not meet your purpose. For example, the cutoff might be too low, causing too many junk seeds to be considered correct. However, if the cutoff is too high, then many correct but low-frequency seeds are inappropriately filtered out. You can change the `saet.trustfreq` parameter to overwrite the estimated frequency cutoff.

If you trust the quality of your reads more or less than the value specified in the `saet.trustprefix` parameter, then make the corresponding changes to the value. If SAET makes many corruptions in the regions of reads with highly packed errors, you can increase `saet.suppvotes` to improve the tendency to correct only isolated errors. SAET provides options for reading and writing spectrum files. The options enable you to build spectrum from better-quality reads and use the spectrum to correct lower-quality reads. You can also build a spectrum from a reference, or build a spectrum from multiple files, such as data from multi-run experiments.

SAET internal parameters

Table 24 lists SAET parameters that we do not recommend changing.

Table 24 SAET internal parameters

saet.run	1	Whether or not to run the SAET module. Allowed values: • **0**: Do not run SAET. • **1**: Run SAET during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
saet.outspecbin	0	Whether or not to output a spectrum in binary format in the file ${task.out.dir}/analysis.input.readset.file.spect.bin Allowed values: • **0**: Do not output the spectrum. • **1**: Output the spectrum.
saet.inspecbin	—	Uses pre-generated file(s) with spectrum (in binary format) for error correction. Use a comma to separate multiple files.
saet.maxtrim	0	Trims erroneous tails of reads up to first trusted seed or up to maxtrim. If the remaining part of a read is shorter than seed size + 2, then the read is discarded. Do not use this option together with the saet.qvupdate parameter. If this parameter is set to 0, then no trimming is done. Allowed values: Integers >= 0.
saet.trimqv	—	Trims erroneous tails of reads up to first trusted seed or up to a position with quality value higher than trimqv. If the remaining part of a read is shorter than seed size + 2, then the read is discarded. Do not use this option together with the saet.qvupdate parameter. If this parameter is set to 0, then no trimming is done. Allowed values: Integers >= 0.
saet.log	saet.log	Log file name.
saet.progress.file	.progress	Progress file name.
saet.progress.start	0	Progress start point. Allowed values: Integers 0–100.
saet.progress.end	100	Progress end point. Allowed values: Integers 0–100.
maximum.workers	4	Maximum number of nodes that can be used by SAET.

SAET output files

SAET writes the corrected reads and quality values into new XSQ files.

The SAET output files are used as input to mapping analyses.

Algorithm description

SAET implementation

SAET implements a modified version of the spectral alignment error correction algorithm proposed by Pevzner et al., 2001. SAET extends the original algorithm by taking quality values and properties of color-space into account, significantly improving the performance of the original method. The method corrects reads given a set of *k*-mers (seeds) spectrum. Given a set of reads (*R*), and a frequency threshold (*m*), the spectrum is defined as the set of all *k*-mers that appear at least *m* times in reads from *R*. The set of such trusted *k*-mers approximates the set of all *k*-mers in the genome. A read is defined as error-free if all of its *k*-mers are trusted. If not all *k*-mers are trusted, SAET attempts to make a read error-free by mutating a few colors in the read. SAET only considers substitution mutations since there no insertions or deletions in SOLiD® system reads. When making error corrections in the read, SAET gives priority to positions with missing calls, followed by positions that have lower-quality values. In addition, SAET avoids correcting two adjacent colors with color-call quality values above a certain threshold, in order to preserve any SNP evidence. SAET computes the *k*-mer size and the multiplicity of trusted *k*-mers, based on the estimated reference length and number of reads in the dataset. Using default parameters, the color call changes introduced by SAET are expected to be 99.99% accurate.

SAET supports multi-core runs. SAET provides a mechanism for multi-node runs by creating sub-spectra from subsets of reads. SAET merges the sub-spectra into the final spectrum that can be used for correction of any subset of reads.

Phases

The SAET algorithm has three steps as described below.

Spectrum build

In this step, a temporary file with a spectrum constructed from all of the reads is generated in the output directory. This process is very fast and does not require large RAM resources However, SAET requires a large amount of disk space (~1 GB for each 1Mbp of genome). SAET creates temporary swap files when memory usage approaches 2 GB.

Error correction

The error correction step takes each read in the input file and attempts to correct it if any of its *k*-mers are not present in the spectrum. This step is the most time-consuming. LifeScope™ Software runs SAET on multiple cores to improve the error correction speed.

Update of quality values for corrected calls

The quality value of each corrected position is replaced with a quality value derived from spectral probability. This step is time-consuming and can be disabled if the update of quality values is not required. (To disable, set `saet.qvupdate` to 0.)

SAET run times

The SAET processing time depends on the input size and number of global and local rounds. With a large number of global and local rounds, an expected throughput is 1 Gbp per hour on a single core. The amount of used RAM is approximately 5 GB for 1 Gbp of genome size.

9 Run a Resequencing Mapping Analysis

This chapter covers:

- Overview 137
- Examples of how to run a mapping analysis 138
- Input files 139
- Stages of mapping 140
- Quality alignments and unique alignments 148
- Fragment mapping parameters 149
- Mapping algorithm 153
- Pairing algorithm 163
- Indel sizes 166
- Mapping output files 167
- Mapping statistics 167
- Mapping statistics output files 171
- FAQ – Mapping 179
- FAQ – Pairing 182
- FAQ – BAMStats 184

Overview

This chapter describes how to perform resequencing mapping. The mapping analysis results files are required as input for tertiary analysis modules.

During secondary analysis LifeScope™ Software performs the following steps:

- Maps or aligns reads to a reference genome.
- Accepts one or more input XSQ (eXtensible SeQuence) files to generate one or more sorted BAM files containing aligned reads.
- For mate-pair or paired-end libraries, pairs alignments that have the same bead ID.
- (*Optional*) Generates mapping statistics.

The 5500 Series Genetic Analysis Systems generates XSQ files, optionally with ECC (Exact Call Chemistry) reads. If an ECC primer round has been performed, the XSQ output also includes the sequence information in base space, in addition to color space. The ECC primer round is recommended during sequencing, for significant increases in sequencing accuracy.

XSQ is an extensible binary file format for storing sequence data, and supports multiple independent reads at the same position in a fragment. For information on the XSQ file format refer to this site:

http://solidsoftwaretools.com/gf/project/xsq

Examples of how to run a mapping analysis

In a standard workflow

The following standard workflows provide examples of running the mapping module:

- genomic.resequencing.frag
- genomic.resequencing.lmp
- genomic.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository. This example turns off tertiary modules in the workflow.

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.pe
# define the input
add xsq /data/xsq/run0209a_50_PE.xsq
add xsq /data/xsq/run0209b_50_PE.xsq
# specify the reference to be used
set reference hg19
# these commands turn off the tertiary modules
set cnv.run 0 tertiary/cnv.ini
set dibayes.run 0 tertiary/dibayes.ini
set inversion.run 0 tertiary/inversion.ini
set large.indel.run 0 tertiary/large.indel.ini
set small.indel.run 0 tertiary/small.indel.ini
# optionally change mapping parameter defaults after this line
# list the analysis configuration
ls
# run the analysis
run
# list progress information for the analysis
ls
```

Dummy XSQ file names are used in this example.

In order to change a mapping parameter value in your analysis, use the `set param` shell command after the line `# optionally change mapping parameter defaults after this line`. For instance, to turn on the `mapping.in.base` parameter, use this shell command:

```
set mapping.in.base 1 secondary/pair.mapping.ini
```

This command shows how to turn on the `mapping.in.base` parameter for the `genomic.resequencing.frag` workflow:

```
set mapping.in.base 1 secondary/fragment.mapping.ini
```

See Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information on running standard workflows.

As a demo analysis

The optional examples download also includes examples of how to run the mapping module by itself, as an individual analysis that is not part of a standard workflow. Fragment and paired examples are included. The use of a standard workflow is preferred.

See Appendix H, "Demo Analyses" on page 567 for more information on running a demo analysis.

Input files

The mapping module accepts as input one or more input XSQ files. The input reads can have different read lengths but they must be of the same library type. For example, LMP read-sets of length 50 and 60 can be processed together. The XSQ files processed in a single analysis can not have different library types. In one mapping analysis, the input must be of only one library type, either Fragment, LMP, or PE.

Plan your input read-sets

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. The following factors control how your input data is analyzed:

- **Index (barcode) IDs –** Using an index ID restricts your input to the reads data of one or more indices.
- **Grouping of reads –** Each group of reads is analyses together as one specimen. The output data for a group is combined into one set of results files.
- **Multiple sample runs –** Unrelated reads can be processed together in one run of LifeScope™ Software, but analyzed separately as separate input data.

See "Define input data" on page 98 for more information on designing adding input data to your analysis.

Mapping one tag of paired data is not supported.

Legacy data

If the data you want to process with LifeScope™ Software is in CSFASTA and QUAL files, these files must be converted to the XSQ file format, through one of the following methods:

- The LifeScope™ Software UI handles converting these files to the XSQ format before mapping.

- If you are using the LifeScope™ Software command shell, you must first convert the CSFASTA and QUAL files to the XSQ format. See Appendix D, "XSQ Tools" on page 519 for information on the standalone XSQ converter.

Stages of mapping

The mapping algorithm involves following stages:

- **Scatter –** The process of distributing reads to the available compute nodes, to achieve parallelization.
- **Mapping –** Each tag is aligned to the reference.
- **Pairing –** With paired-end data, two tags are individually mapped using the same mechanism as fragment mapping. Then pairing is performed. Fragment data skips this stage.
- **BAM generation –** The process of converting the output of mapping and pairing to BAM format. For color-space data, the base translation step is also included. Due to the scattering step, the BAM files at each compute node are a sub-set of the final BAM file and are referred to as baby BAMs.

  BAM generation involves these steps:
  - **Ma2BAM –** A conversion step, from MA format to BAM format, for fragment data only.
  - **Pa2BAM –** A conversion step, from PA format to BAM format, for paired data only.
  - **Refcor –** Reference-assisted color-to-base translation.
  - **Sort BAM –** Sorting the BAM file from bead-id order to coordinate order.
  - **Merge BAM –** Merging the mini-BAM files created in the sort phase into the final output BAM files.

Figure 1 shows the phases involved in mapping fragment data. *Tag1* refers to the input data in the F3 strand.

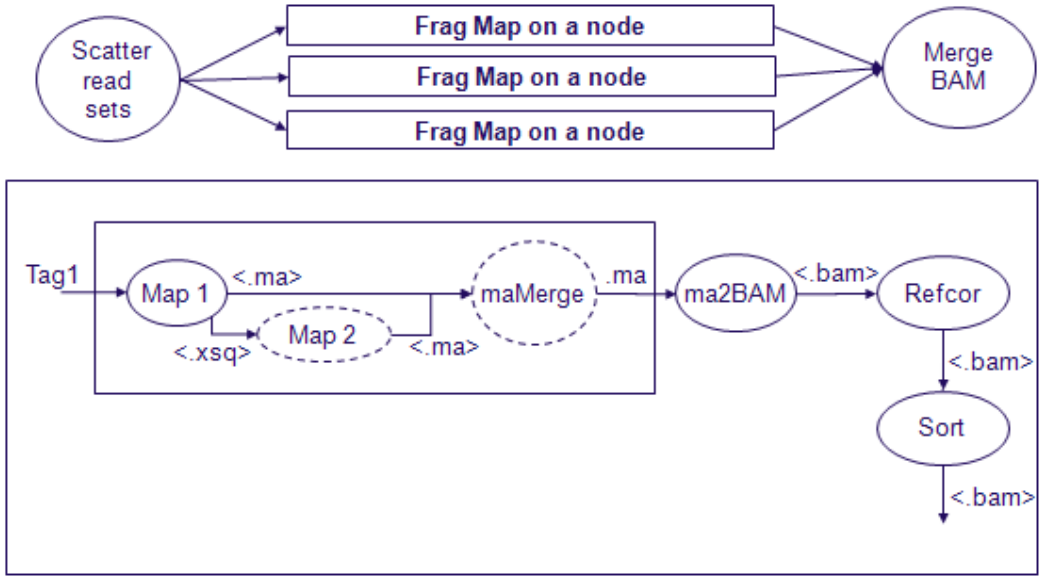


Figure 1 The phases within the mapping module, when mapping fragment data

Figure 2 shows the phases involved in mapping paired data. *Tag1* and *Tag2* refer to the input data on the two different strands.

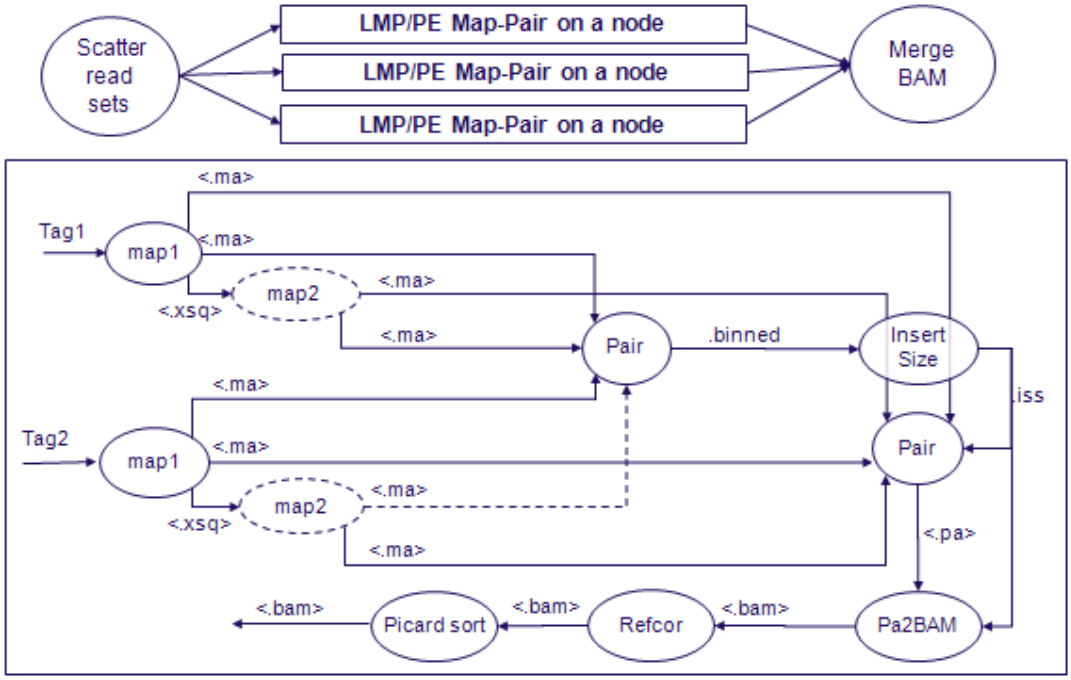


Figure 2 The phases within the mapping module, when mapping mate-pair or paired-end data

Scatter

The scatter stage distributes reads to the available compute nodes, in order to gain the efficiency benefits of parallel computing.

The mapping module can accept multiple read-sets, which are allowed to have different read lengths. The input set of read-sets from multiple XSQ files is first split based on read length. Fragment read-sets are partitioned into separate categories based on these read length categories:

- 0 ≤ Length < 25 (uses first | second.map.scheme.unmapped | repetitive.1)
- 25 ≤ Length < 35 (uses first | second.map.scheme.unmapped | repetitive.25)
- 35 ≤ Length < 50 (uses first | second.map.scheme.unmapped | repetitive.50)
- 50 ≤ Length < 60 (uses first | second.map.scheme.unmapped | repetitive.60)
- 60 ≤ Length < 75 (uses first | second.map.scheme.unmapped | repetitive.75)
- 75 ≤ Length (uses first | second.map.scheme.unmapped | repetitive.75)
- Trimmed reads, with variable read lengths (uses first | second.map.scheme.unmapped | repetitive.variable)

Mate-pair read-sets are partitioned into separate categories based on the pair length. Example categories are:

- (0.0) ≤ (Length1,Length2) < (25,25)
- (25.25) ≤ (Length1,Length2) < (50,50)
- (50.50) ≤ (Length1,Length2) < (60,60)
- (60.25) ≤ (Length1,Length2) < (75,35)

The reads from each category are scattered into multiple jobs based on the parameters `fragmap.minreads.per.node` and `fragmap.number.of.nodes`. These parameters are described in the performance section (see "Mapping performance" on page 152).

Fragment mapping

The Map phase (also known as mapreads) includes map 1, map 2, and maMerge. Map combines information from reads in XSQ-formatted input files and a FASTA-format reference file, to generate alignment information in an intermediate file format known as match file format (.ma). Unmapped or poorly mapped reads from the first stage of mapping (map 1) can enter another mapping stage (map 2). Both these mapping stages have independent controls with respect to the mapping scheme and the gapped alignment algorithm. Since a poorly-mapped read can have alignments reported from map 1 and map 2, the maMerge is invoked to report the best alignments from the two mapping stages.

In the default setting, two mapping stages are invoked. The first one does ungapped alignment only. The second one does gapped alignments only. The second stage is switched off for read-sets with lengths less than the value of the parameter `min.length.for.aggressive.gapped.mapping` (default is 50). The maMerge step is not invoked if only one stage of mapping is run.

Alignment finding and gap finding use a proprietary algorithm. The presence of a gap can not be investigated in the anchor region. In general, for read length L and anchor defined by A.B.C,

- If A is zero, mapping looks for gap from position A+1 to L.
- If C is zero, mapping looks for gap only at position 0 to C.

Each map step is a multi-threaded executable that can handle one or more XSQ files of one library type. The unmapped and poorly reads from the first mapping stage are XSQ-formatted so that they can be processed by subsequent stages of mapping. The match file format is an internal file format that stores alignment and gap information. The MA files from two stages of mapping and maMerge are all generated in the scratch folder. The parameters to control mapping and gap finding algorithms are in "Mapping algorithm" on page 153. For details of the algorithm, see "Mapping schemes" on page 159.

Mate-pair mapping

Each tag is mapped using a mapping protocol similar to fragment mapping. The match files (MA files) are provided to pairing stage.

Pairing

The pairing module matches pairs of reads from the F3 and R3 mapping results of a mate-pair run. Pairing also matches reads from the F3 and F5-P2 files of a paired-end mapping run.

Pairing is divided in two stages. In the first stage, an estimate of the insert size distribution is made. Pairing module accepts the MA files, calculates the insert size from uniquely mapped beads and writes out a BINNED file (*.binned) containing insert size distribution for each read-set. A perl script is invoked on the collection of BINNED files to calculate the insert size statistics (max, min, average, standard deviation) per read set. Output of the perl module is a tab-separated file (ISS, *.iss) containing insert size statistics per read set. This ISS file is passed to pa2BAM module that loads the average and standard deviation information to the appropriate BAM file. Pairing executable is invoked one more time with this ISS file and the MA files. This time pairing, rescue, and indels are called. Output of this pairing stage is a PA file, in an internal format.

The pairing algorithm performs the following steps for each pair of reads:

1. It finds all good AAA pairs based on the order, orientation, and distance between the two reads. (See the genomic classifications listed in Tables 34 and 35 on page 182 and 183 for information about AAA and other three-letter categories.)

2. If no AAA pair is found, and a reference sequence is available, the algorithm performs mate-pair rescue using hits to one tag as an anchor, and then scanning for the "sister tag" in the region predicted by the library insert size.

   Note: Either F3 or R3 tags can serve as the anchor, as long as the number of hits is below the *Z* threshold. The *Z* threshold is determined by the value entered for the `pairing.gapped.max.hits` parameter. If the F3 tag has *x* alignments, the R3 tag has *y* alignments, and both tags have fewer than Z hits, then the *x*+*y* anchor candidates are examined (see Figure 3, Mate-pair rescue example).

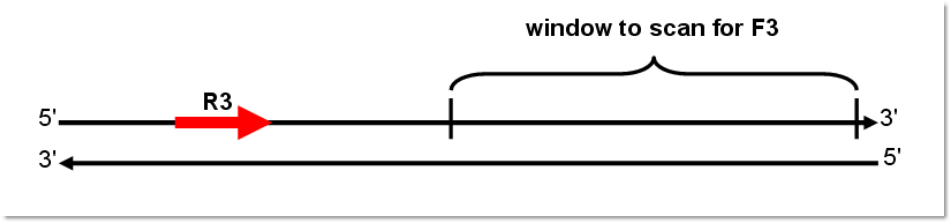


Figure 3 Mate-pair rescue example

3. It determines if a AAA pair is unique. If multiple AAA pairing candidates exist for a given pair of tags, then the pair whose best pairing candidate scores at least *x* higher than the second-highest scoring pairing candidate is still considered unique, and all other AAA candidates with lower scores are discarded. The value of *x* is determined by the setting of the `pair.uniqueness.threshold` parameter.

4. For non-AAA pairs where both tags have unique hits, the algorithm performs additional classifications based on the strand, distance, and orientation of the tags.

   If either of the two tags has multiple mapping locations, but the highest mapping score is more than half of *x* higher than the second-highest score, then the location with the highest score is still considered unique and all other locations are discarded.

The pairing module supports paired-end experiments in addition to mate-pair experiments. In paired-end experiments, the actual sequencing is done on the same strand in opposite directions. However, because the representation in the sequencing output file follows the convention of representing the sequence from 5' to 3', the matching and pairing modules attempt to match the F5 and F3 tags on different strands, facing each other. In addition, a distance constraint determined by insert size is satisfied (see Figure 4). The pairing algorithm follows the same steps as the mate-pair algorithm. Steps 1 and 2 are modified to enforce the different order and orientation requirement. If you convert matching positions for F3 tags to the opposite strand, then paired-end pairing is performed the same way as it is with mate-pair. The three-letter classification is defined the same way in mate-pair and pair-end. (See Table 34 on page 182 and Table 35 on page 183 for the genomic classification tables.)

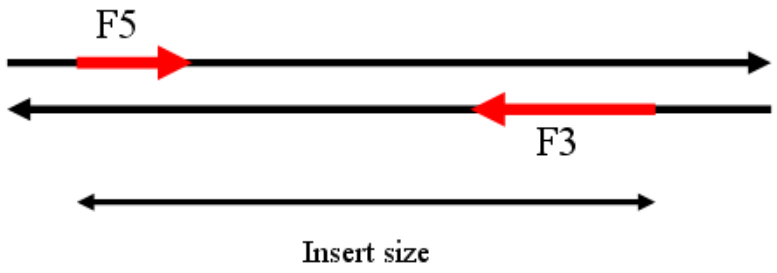


Figure 4 Paired-end tags example

Mark duplicates

For paired data only, the mark duplicate module checks for and marks duplicate bead ids. Except for the highest quality alignment among the duplicates, the duplicate alignments are marked for filtering, and are by default ignored by tertiary modules. Each duplicate alignment is marked by setting to 1 the `0x400` bit of its FLAG field.

For analyses with grouped data, the modules considers all BAM files for one sample, and finds duplicate beads across the multiple BAM files for the sample.

The mark duplicate module is enabled by default in whole genome workflows with paired data.

Mark duplicate parameters

Table 25 lists mark duplicate parameters.

Table 25 Mark duplicate parameters

Parameter name	Default value	Description
Optional parallelization parameters		
process.one.bamfile.only	true	Process BAM files separately for marking duplicates. Set to false to process all BAM files of a sample together. See "Mark duplicates on multiple BAM files" on page 145. Allowed values: • **true**: Process one BAM file at a time. • **false**: Process all BAM files of a sample together.
markduplicates.input.dir	—	The directory containing mapping output files, which also are the BAM files input to mark duplicates. You typically do not change this parameter.
Optional resource parameters		

Table 25 Mark duplicate parameters

Parameter name	Default value	Description
memory.request	15gb	The total memory, in gigabytes, that is available for map reads, per compute node. Include the units gb in the setting.
java.heap.space	1024	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.
markduplicates.max.nodes	4	The number of compute nodes available for this step.
markdup.delete.mapping.bam.files	false	We strongly recommend that you keep this parameter set to *false*. If true, this parameter causes the BAM output of mapping to be deleted (leaving only the BAM output of the mark duplicates module).

Mark duplicates on multiple BAM files

Number of duplicates called

If you process multiple BAM files together and the same bead ID occurs in different BAMs, those beads are treated as duplicates. This is the behavior with the setting `process.one.bamfile.only=false`. See the mark duplicates FAQ "Why does LifeScope™ Software report more duplicate beads than Picard reports?" on page 184.

The `process.one.bamfile.only` parameter

The setting of this parameter affects your processing time and memory usage.

The multiple BAM option (`process.one.bamfile.only=false`) is appropriate for multiple barcodes from the same sample run on the same lane. This option typically takes more memory but less time, and potentially affects the number of duplicates marked. With `process.one.bamfile.only=false`, the mark duplicates module processes input in the following phases:

1. Scan all BAM files.
2. Only after all scanning is complete, then mark duplicates for all BAM files concurrently.

With `process.one.bamfile.only=true`, mark duplicates typically requires less memory but more process time. With this setting, the mark duplicates module phases are the following:

1. Scan one BAM file.
2. Only after the scan of the first BAM is complete, concurrently do these:
   a. Scan the other BAM files.
   b. Mark duplicates for only the first BAM file.
3. Only after mark duplicates for the first BAM file is complete, the module marks duplicates for the remaining BAM files, *one at a time.*

BAM file generation

This section describes the phases involved in BAM file generation.

Ma2BAM, pa2BAM

Ma2BAM and pa2BAM are format converters that change match files to a pro-BAM format. These pro-BAM files do not contain sequence or quality information but contain relevant BAM meta data. The pro-BAM files are sorted by incoming bead-id order. Relevant meta data from XSQ files is captured in pro-BAM files at this stage. The parameters `bamgen.mqv.threshold` and `create.unmapped.bam.files` control the functionality of the ma2BAM and pa2BAM utilities. The pro-BAM files are generated on the scratch folder.

Ma2BAM takes output of mapping (MA files) and converts them to pro-BAM format. The pa2BAM takes output of pairing (PA files) and converts them to pro-BAM format. During BAM generation, the mapping and pairing quality values are also computed.

For reads with multiple ungapped alignments, the read with the highest mapping or pairing quality value is chosen as the primary alignment for the read, and is reported to the BAM file. In cases where there are multiple alignments with the same quality value, the primary alignment is chosen at random from among the alignments with the same quality value.

Mapping and pairing quality values

This section describes mapping quality value (MQV) and pairing quality value (PQV) scores. Quality values are Phred-scaled quality scores. For any given alignment, an MQV is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. A PQV represents the confidence in both alignments of a pair, and is a combined quality score for both reads. Quality values are within the range 0–100:

- **0 –** The highest probability of error
- **100 –** The lowest probability of error

The following factors increase confidence and increase an alignment's quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce an alignment's quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

The pairing algorithm reports multiple sets of possible alignments for any given pair of reads (F3/R3 tags for a mate-pair run and F3/F5-P2 tags for a paired-end run). The pairing quality algorithm uses a Bayesian approach to calculate the quality of a given alignment for a pair of reads and the alignment with the highest pairing quality value (PQV) is chosen as the primary alignment for the pair of reads. The PQVs represent the Phred-scaled quality score, and are useful for downstream variant detection modules such as diBayes, small indels, large indels, and CNV.

In order to be consistent with the Phred quality score (-10*log10[prob(error)]) used widely in literature, the quality is computed as the negative log odds of misaligning the read (or pairs of reads, for PQV). The resulting quality values are normalized by the maximum possible value to ensure that the quality values are within the range 0–100.

Gapped alignments

The pairing algorithm searches for gapped alignments (indels) when one of the tags (F3/R3/F5-P2/F5-BC) maps to the reference genome and the other tag does not map to the genome within the insert-size range, or it maps within the insert-range but the alignment is short. If both an ungapped and a gapped alignment are found for a given read, then, due to the low prior probability of 10^-4 assigned to the gapped alignments, the PQV for gapped alignments is zero.

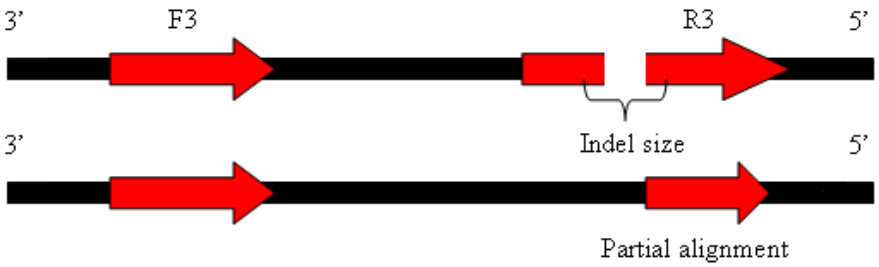


Figure 5 Example of a gapped alignment and a partial alignment

In calculating the PQV for gapped alignments, the alternative hypothesis tested is the probability of finding the partial ungapped alignments. The read with the gapped alignment is treated as two partial reads on either side of the indel start point. The partial read with the greater length is used as the partial alignment length for the alternate hypothesis.

Refcor

Reference Assisted Color-to-Base Translation (also known as refcor) combines information from color calls, optional ECC extra primer round calls, and the reference sequence, to enhance the quality of base-space reads by detecting and fixing color errors that convert into multiple base-space errors.

Refcor accepts a set of pro-BAM files, which contain alignments information without base or color sequence, and generates a corresponding set of BAM files containing alignments with base-space and color-space calls (on the scratch folder). These BAM files are sorted by bead-ID and contain dummy base calls and QVs, but are otherwise compatible with the BAM specification. The refcor module produces bead-ID-sorted BAM files that have valid calls and QVs.

For each alignment, the module constructs the primer-transition graph with vertices corresponding to all possible *k*-mers (*k*=4 for 2BE+4BE, *k*=2 for 2BE) in each read position. Two adjacent vertices are connected with an edge if they overlap in *k*–1 bases. A score corresponding to the (1–*Pe*), where *Pe* is the probability of being erroneous, is assigned to each of the vertices. Scores are derived from quality values corresponding to enumerated calls. When making a base call in a position, the current algorithm considers the cumulative probability of all sequences that have called base in that position. The evidence from the reference (base) is assigned to the vertex with *k*-mer ending in corresponding base only if the color transition between current and following reference bases matches the color call in a read. Such assignment reduces the reference over-correction in SNP positions. In order to reduce reference over-correction in positions with color errors, we assign to reference base a very low weight corresponding to QV of 8. The weight can be reduced to QV of 0, completely reducing

the reference guiding in the color-to-base translation. This results in a smaller skew between reference to non-reference allele ratio. Increasing the weight results in false positive variant calls reduction. This process restores the phase shift caused by color errors or completely eliminates the errors.

The parameters `bamgen.refcor.softclip`, `bamgen.refcor.addcs`, `refcor.reference.weight`, and `refcor.base.filter.qv` control the refcor program. The defaults provided are validated for a wide range of data.

- If the XSQ file contains base-space data only, the XSQ base call and quality value are transferred to the BAM file. In this case, no color calls are present in the BAM file, irrespective of the value of `bamgen.refcor.addcs`.
- If the XSQ file contains 2BE color-space data only, refcor uses aligned 2BE and reference to generate base translation for the aligned part of the read. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, bases are determined by naively translating color to base. For these unaligned reads, the QV for the base at position k is the minimum color QV for all positions $\leq k$.
- If the XSQ file contains ECC data, then reference with 2BE and 4BE color are used to generate the new base sequence. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, the original base-space data from ECC is preserved.

Sort BAM

The output of refcor is a set of BAM files in bead-id sorted order. These files are sorted in this step using Picard sort into coordinate sort order before merge. These BAM files are referred to as mini-BAM files and are stored in the analysis temporary directory.

Merge BAM

This phase accepts a set of mini-BAM files and outputs a set of BAM files. These BAM files are stored in the analysis output directory.

One BAM file is generated per read-set. For a 96-barcoded dataset in a single XSQ file, either 96 or 97 BAM files are created. The 97th BAM contains reads that have the default barcode index.

Quality alignments and unique alignments

We strongly recommend using mapping quality values, rather than uniqueness, as the standard for recognizing or filtering good alignments. A mapping quality value estimates the probability that the alignment is correct, and takes several indications into account, including the number of alignments or uniqueness of an alignment.

The parameter `bamgen.mqv.threshold`, (introduced in release 2.0), is available to filter alignments written to your BAM output files. In order to limit your BAM files to good quality reads, set `bamgen.mqv.threshold` to a positive integer such as 8 or higher. This technique ensures that you do not see poor quality reads in your BAM file, and that your tertiary analysis is limited to good quality reads.

How a quality value is derived

The following factors increase confidence and increase an alignment's quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce an alignment's quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

How a quality value is interpreted

For any given alignment, a quality value is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. Quality values are within the range 0–100:

- **0 –** The highest probability of error
- **100 –** The lowest probability of error

Quality values are Phred-scaled quality scores, logarithmically linked to error probabilities. A mapping QV of 10 means that we calculate there is a 90% chance that the alignment is correct. A mapping QV of 20 means that we calculate there is a 99% chance that the alignment is correct.

Unique alignments

Unique alignments might be confused with high quality alignments, but uniqueness is not by itself a good criteria to judge an alignment. Uniqueness only means that one step of secondary mapping, the aligner, reports one alignment. This information is a part of the total information generated during mapping, but does not contain all of the information known about the alignment. In contrast, the mapping quality value is a numerical representation of all considerations about the alignment.

For example, a read might have multiple alignments. If the quality value for only one of those multiple alignments is high, that probability of that alignment being good is high. That alignment should be used in your analysis, even though it is not unique.

For a unique alignment, its quality value must be examined to determine if the alignment should be taken or filtered out. Its uniqueness by itself is irrelevant to the decision about whether it is a good read.

Fragment mapping parameters

This section describes the runtime parameters which control the behavior of your fragment mapping analysis. Table 26 lists these parameters.

In order to change a parameter value, use the `set param` shell command to replace the line `[optionally change mapping defaults after this line]` in the example commands in "Examples of how to run a mapping analysis" on page 138. For instance, to have the unmapped BAM file generated, change the parameter `create.unmapped.bam.files` with this command:

```
set create.unmapped.bam.files 1 secondary/fragment.mapping.ini
```

Fragment mapping parameters table

Table 26 lists the fragment mapping parameters.

See Table 30 on page 168 for mapping statistics (BAMStats) parameters.

Table 26 Mapping parameter description

Parameter name	Default value	Description
Optional resource parameters		
wall.time	120	Total time for the process to complete.
java.heap.space	1500	Dynamic memory requirement, in megabytes.
mapping.memory	15gb	The total memory, in gigabytes, that is available for map reads, per compute node. Include the units gb in the setting. This number is set to 15 GB because the minimum hardware requirement for memory is 16 GB. **Note:** Whenever possible, run mapping with 24 GB. When a mapping analysis is run with less than 19 GB of memory, the mapping module splits the reference. During gapped alignment of the second half of the genome, some alignments are reported at incorrect positions. The incorrect gapped alignments cause incorrect results (missing indels) during small indel analysis. **Note:** Typically, the job scheduler will allocate on nodes that have the requested memory available. For mixed hardware, it is suggested that different queues are created based on the memory available. Mapping jobs should be launched on high memory systems. For human references (hg18, hg19) and a 25.2 scheme, 45 GB is recommended for fastest performance. The smallest recommended RAM for human reference is 19 GB.
Optional parallelization parameters		
fragmap.number.of.nodes	4	The number of compute nodes available for this analysis.
fragmap.max.number.of.jobs	100	The maximum number of jobs allowed for mapping. Allowed values: Integers 24–1000.
mapping.np.per.node	8	The number of processors per node use for mapping. The reads are divided into the number of chunks specified for this parameter.
fragmap.minreads.per.node	4000000	If the total number of fragments exceeds this amount, then the analysis is distributed across the available nodes. Allowed values: Integers 1–125000000 (125 Million)
fragmap.maxreads.per.node	150000000	The maximum number of fragments that can be processed on a single node. Allowed values: Integers 1–150000000 (150 Million)
Optional mapping parameters		

Table 26 Mapping parameter description *(continued)*

Parameter name	Default value	Description
mapping.in.base	false	Whether to map in base space (true) or color space (false). Set to true if input data has base space available. Allowed values: • **false**: Map in color space. • **true**: Map in base space. If only color space is available, then the module fails when base space mapping is turned on.
second.map.gapped.algorithm	GLOBAL	Whether or not to do indel finding, and controls the behavior of indel finding. Allowed values: • **GLOBAL**: Mapping reports global alignment up to one indel. • **LOCAL**: Mapping reports local alignment up to one indel.
Optional BAM generation parameters		
bamgen.refcor.softclip	0	For cases when the unaligned portion of a read needs to be presented in the BAM record, we use soft clipping in the CIGAR string. If a base-space sequence generated by ECC decoder is available, we use that sequence to represent corresponding part of the read. If the ECC decoded base-space is not available, then we naively translate using 2BE and/or +4BE evidence (if available). The soft clipped part of the read can be permanently clipped if either of these conditions is true: • The ECC base QV is less than 4. • The ECC base and 2+4 naive propagation have less than 50% similarity. Allowed values: 0,1.
bamgen.refcor.addcs	1	Whether or not to add the color sequence to BAM records. If the XSQ file does not contain color space, this parameter is ignored, and the resulting BAM file output does not contain color space. See "Refcor" on page 147 for information about refcor. Allowed values: 0,1.
refcor.reference.weight	8	Reference weight. This parameter is used during base translation. In the read reconstruction process, multiple signals are combined to generate the final base call. This parameter adds weight (in terms of Phred score) only to signals which are compatible with reference. Color combinations that result in a variant are considered compatible with reference. Additional weight helps to eliminate base propagation errors caused by color error(s) during base translation. Allowed values: Integers 0–100.
refcor.base.filter.qv	10	Bases with a quality value below the value of this parameter provided are replaced with 'N'. Allowed values: Integers 0–1000.
bamgen.mqv.threshold	0	Provides control over the contents written to the output BAM file depending on the quality value of the alignment. To preserve only high quality alignments, set this value to a positive integer. Allowed values: Integers 0–1000.

Table 26 Mapping parameter description *(continued)*

Parameter name	Default value	Description
create.unmapped.bam.files	FALSE	If set to TRUE, creates an additional BAM output file containing unmapped reads. Allowed values: • **FALSE**: Do not create the unmapped reads output file. • **TRUE**: Create a BAM output file containing the unmapped reads.

Mapping performance

Mapping speed depends on hardware properties as well as on the scattering logic. Mapping runs are split into multiple jobs for processing efficiency. The following parameters affect how a mapping run is split:

- **`fragmap.number.of.nodes`** – Specifies the number of jobs that are created, only if a split is necessary. The size of each split job is approximately the total number of reads divided by the number of jobs.
- **`fragmap.max.number.of.jobs`** – Determines the maximum number of jobs that can be launched per analysis. Setting this parameter to a very large value can cause the scheduler to behave incorrectly.
- **`fragmap.minreads.per.node`** – Determines the threshold of beads beyond which splitting occurs.
- **`fragmap.maxreads.per.node`** – Determines the largest number of beads that can be mapped per node. The default is based on the minimum scratch space requirements. If you scratch space is smaller or larger, this number can be made smaller or larger respectively.
- **`fragmap.processors.per.node`** – This is the number of cores available for mapping analysis on each node.
- **`mapping.memory`** – For human mapping and 25.2.0 scheme, there are 3 schema lines. Simultaneous handling of schema lines reduces I/O and therefore improves mapping speed.

Memory requirements for mapping jobs are the following:

- Reference: ~6 GB
- 1 schema line: ~13 GB
- 1 schema line and reference: ~19 GB
- 2 schema lines and reference: ~32 GB
- 3 schema lines and reference: ~45 GB

Below 19 GB, the reference is split up and I/O increases significantly. Pre-built hash tables are not used for memory less than 24 GB. LifeScope™ Software supports the reuse of hash tables for human genome reference files (hg18 and hg19) and the default mapping schemes 35.2, 25.2, and 20.1. Pre-calculated color-space hash tables are included in the reference directory available with LifeScope™ Software. If system RAM is 24 GB or higher, the hash tables are loaded into memory. Depending on the pipeline, this hash table reuse typically saves 2–3 hours in analysis processing time.

For genomes without pre-calculated hash tables in the reference repository, the mapping module creates hash tables but does not save them for reuse.

In general, set the `mapping.memory` parameter to be 1 GB less than the total system memory (RAM) available.

Mapping only scales when the number of reads that are mapped together is greater than 100 M. Splitting such that the total number of fragments per node is significantly less than 100 M is not recommended. The splitting can be explained using the following pseudo-code.

```
if (num.reads <= fragmap.minreads.per.node ) { num.jobs = 1}
reads.per.job = num.reads / fragmap.number.of.nodes
if (reads.per.job > fragmap.maxreads.per.node) {
    reads.per.job = fragmap.maxreads.per.node}
num.jobs = celing (num.reads / reads.per.job)
if (num.jobs > fragmap.max.number.of.jobs ) {
    throw exception; }
```

In above pseudo-code, the input read-sets (possibly from multiple XSQ files) have `num.reads` that fall in one read length category. `Num.jobs` is the total number of jobs launched for mapping analysis of `num.reads` beads.

In general, the mapping parameters allow mixing and matching single-anchor and multiple-anchor schemes, and also mixing and matching different seed lengths and different numbers of mismatches. For example, an unmapped scheme of 25.2.0, 35.2.0:15:25, 20.1.0, with a repetitive scheme of 35.2.0:30, 45.2.0, 60.2.0, is supported, but not recommended. The impact on performance can be significant.

Mapping algorithm

Internal mapping parameters

Table 27 lists mapping parameters that we do not recommend changing. The "Mapping schemes" section which follows describes how these parameters affect the behavior of the mapping module.

Table 27 Internal mapping parameters

Parameter name	Default value	Description
fragment.mapping.run	1	Indicates whether or not to run the mapping module. Allowed values: • **0**: Do not run the mapping module. • **1**: Run the mapping module. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
mapping.use.iub.reference	FALSE	Whether or not to support reference sequences that contain IUB codes. Allowed values: • **FALSE**: Do not support IUB codes in the reference file. • **TRUE**: Mapping allows matching to either alleles of bi-allelic IUB codes in the reference file.

Table 27 Internal mapping parameters *(continued)*

Parameter name	Default value	Description
bamgen.primary.output.filter.type	primary_only	Filters the content written to the output BAM file depending on whether the alignment is gapped or not. Note that unmapped reads are considered primary. Allowed values: • **no_filtering**: Report all alignments. • **primary_only**: Report all primary alignments (both gapped and ungapped). • **primary_ungapped_all_gapped**: Report all gapped alignments (even if not primary) and report ungapped alignments only if they are primary. Default.
fragmap.num.mapping.stages	2	The number of mapping passes to perform in this analysis. Allowed values: • **1**: Perform mapping in one pass. Gapped alignment finding is off by default. • **2**: Perform mapping with two passes. Recommended for increased accuracy of indel finding, but with an increase in processing time. The second stage of mapping performs only gap alignments. When set to 2, do not set second.map.gapped.algorithm to NONE.
mapping.fragment.tag	F3	When mapping XSQ files with more than one tag, this parameter selects the tag to map. The value must match one of the tag names in the TagDetails group of the XSQ file. Allowed values: F3, R3, F5-P2, F5-BC, F5-DNA, F5-RNA. This feature should not be used unless for data quality analysis. If a LMP/PE XSQ file is processed for fragment mapping, tertiary analysis is not supported.
mapping.schema.file	—	Specifies the full path to the file containing the mapping schema. If this parameter is not set, the default installed file is used.
mapping.gap.min.non.matched.length	9	The minimum number of non-matches allowed during indel finding (if performed).
min.length.for.aggressive.gapped.mapping	50	Overrides some of other parameters' settings, depending on the relation between this value and read length. If the read length of tag1 is less than this value, the following parameters are set: • num.mapping.stages.tag1=1 • first.map.gapped.algorithm.tag1=NONE • second.map.gapped.algorithm.tag1=NONE • pairing.gap.max.mismatches.tag1=2 If the read length of tag2 is less than this value, the following parameters are set: • num.mapping.stages.tag2=1 • first.map.gapped.algorithm.tag2=NONE • second.map.gapped.algorithm.tag2=NONE • pairing.gap.max.mismatches.tag2=2
First map parameters		

Table 27 Internal mapping parameters *(continued)*

Parameter name	Default value	Description
first.map.gapped.algorithm	NONE	Whether or not to do indel finding, and controls the behavior of indel finding. Allowed values: • **NONE**: Turns off indel finding. • **GLOBAL**: Mapping reports global gapped alignments up to one indel. • **LOCAL**: Mapping reports local gapped alignments up to one indel. first.map.gapped.algorithm is set to NONE if the read length is less than min.length.for.aggressive.gapped.mapping.
first.map.gapped.alignments.only	0	Whether or not the first pass outputs *only* gapped alignments. Allowed values: Integers 0 or 1. • **0**: Do not restrict the output to only gapped alignments. • **1**: Restrict the output to only gapped alignments.
first.map.gapped.deletion	19	The maximum deletion size in a gapped alignment in the first pass.
first.map.gapped.insertion	4	The maximum insertion size in a gapped alignment in the first pass. Allowed range: 0=< insertion size <= read length. The module fails if the insertion size is not within this range.
first.map.gapped.penalty	10	The penalty for a single gap in alignment, for the first mapping pass. Allowed values: Integers >= 0. When the penalty is 10, either side of the gap must have a score of at least 10 for the gapped alignment to exist. Gapped alignment is effectively turned off if the penalty is set greater than the read length. For a gapped alignment, scoring is determined by three factors: • The matching score, which is 1. • The mismatch score, which is set by first.map.mismatch.penalty and is -2, by default. • This parameter (first.map.gapped.penalty), which sets the score associated with a gap.
first.map.gapped.error.indel	3	The number of mismatches allowed for gap alignments on the first mapping pass. Reasonable range: 1 <= value <= 15% of the read length. Allowed values: Integers, from 1 to the read length. Only used if first.map.gapped.algorithm is set to GLOBAL.
first.map.gapped.edge.length.insertions	12	The minimum edge length required for insertions, on the first pass. Reasonable range: 1–15% of the read length. Lower values may lead to more false positives. Allowed values: Integers, from 1 to the read length. Only used if first.map.gapped.algorithm is set to GLOBAL.
first.map.gapped.edge.length.deletions	12	The minimum edge length required for deletions, on the first pass. Reasonable range: 1–15. Lower values may lead to more false positives. Only used if first.map.gapped.algorithm is set to GLOBAL.

Table 27 Internal mapping parameters *(continued)*

Parameter name	Default value	Description
first.map.gapped.seed.window.left	40	The window size allowed to the left of the anchor alignment, on the first pass. Reasonable range: 5–50. Only used if first.map.gapped.algorithm is set to GLOBAL.
first.map.gapped.seed.window.right	80	The window size allowed to the right of the anchor alignment, on the first pass. Reasonable range: (5 + read length) - (50 + read length). Only used if first.map.gapped.algorithm is set to GLOBAL.
first.map.mask.positions.	—	A string of 0s and 1s that indicate the read positions to be masked during the first mapping pass. There is no penalty in mapping towards positions that are masked. Default: blank Leaving the parameter blank results in no masking. Leave the parameter blank for global style mapping (previously called "classic" mapping in earlier versions). (Masking replaces read trimming.) If set, the length of the string must match the anchor length or seed length (the first value of scheme parameter such as 25.2.0). With a 25-mer read and a 25.2.0 scheme, an example mask is 1111111111111111111100000 (20 1s, then 5 0s). The bases corresponding to the 0s are ignored. Under the 0s portion of the mask, any mismatches do not count toward the mismatch penalty.
first.map.max.hits	50	The maximum number of alignments for a read on the first pass. This setting gives the maximum number of hits that are reported in the mapping output (even if more hits are found). Allowed values: Integers >= 1.
first.map.mismatch.penalty	-2	A scoring penalty for mismatches. Used in local alignment mode. Must be a negative number or zero. Allowed values: Integers <= 0. If this penalty is set to a value greater than the read length, effectively no mismatches are allowed on the first pass.
first.map.scheme.unmapped.1	0.0.0	The scheme used for unmapped reads of a length less than 25, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.repetitive.1	—	The scheme used for repetitive reads of a length less than 25, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.unmapped.25	25.2.0	The scheme used for unmapped reads of a length 25–34, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.repetitive.25	—	The scheme used for repetitive reads with a length 25–34, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.unmapped.35	25.2.0	The scheme used for unmapped reads with a length 35–49, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.repetitive.35	—	The scheme used for repetitive reads with a length 35–49, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.unmapped.50	25.2.0:15	The scheme used for unmapped reads with a length 50–59, for the first mapping stage. See "Mapping schemes" on page 159.

Table 27 Internal mapping parameters *(continued)*

Parameter name	Default value	Description
first.map.scheme.repetitive.50	—	The scheme used for repetitive reads with a length 50–59, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.unmapped.60	25.2.0:20	The scheme used for unmapped reads with a length 60–74, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.repetitive.60	—	The scheme used for repetitive reads with a length 60–74, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.unmapped.75	25.2.0:20	The scheme used for unmapped reads with a length of 75 or larger, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.repetitive.75	—	The scheme used for repetitive reads with a length of 75 or larger, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.unmapped.variable	25.2.0:20	Use for unmapped reads of variable length coming from ECC trimmed base-space reads, for the first mapping stage. See "Mapping schemes" on page 159.
first.map.scheme.repetitive.variable	—	Use for repetitive reads of variable length coming from ECC trimmed base-space reads, for the first mapping stage. See "Mapping schemes" on page 159.
Second map parameters		
second.map.gapped.alignments.only	1	Whether or not the second pass outputs *only* gapped alignments. Allowed values: • **0**: Do not restrict the output to only gapped alignments. • **1**: Restrict the output to only gapped alignments.
second.map.gapped.deletion	19	The maximum deletion size in a gapped alignment in the second mapping pass.
second.map.gapped.insertion	4	The maximum insertion size in a gapped alignment in the second mapping pass. Allowed range: 0=< insertion size <= read length. The module fails if the insertion size is not within this range.
second.map.gapped.penalty	10	The penalty for a single gap in alignment, for the second mapping pass. Allowed values: Integers >= 0. When the penalty is 10, either side of the gap must have a score of at least 10 for the gapped alignment to exist. Gapped alignment is effectively turned off if the penalty is set greater than the read length. For a gapped alignment, scoring is determined by three factors: • The matching score, which is 1. • The mismatch score, which is set by second.map.mismatch.penalty and is –2, by default. • This parameter (second.map.gapped.penalty), which sets the score associated with a gap.
second.map.gapped.error.indel	3	The number of mismatches allowed for gap alignments on the second mapping pass. Reasonable range: 1 <= value <= 15% of the read length. Maximum value: the read length. Only used if second.map.gapped.algorithm is set to GLOBAL.

Table 27 Internal mapping parameters *(continued)*

Parameter name	Default value	Description
second.map.gapped.edge.length.insertions	12	The minimum edge length required for insertions, on the second mapping pass. Reasonable range: 1–15% of the read length. Lower values may lead to more false positives. Only used if second.map.gapped.algorithm is set to GLOBAL.
second.map.gapped.edge.length.deletions	12	The minimum edge length required for deletions, on the second mapping pass. Reasonable range: 1–15. Lower values may lead to more false positives. Only used if second.map.gapped.algorithm is set to GLOBAL.
second.map.gapped.seed.window.left	40	The window size allowed to the left of the anchor alignment, on the second pass. Reasonable range: 5–50. Only used if second.map.gapped.algorithm is set to GLOBAL.
second.map.gapped.seed.window.right	80	The window size allowed to the right of the anchor alignment, on the second pass. Reasonable range: (5 + read length) - (50 + read length). Only used if second.map.gapped.algorithm is set to GLOBAL.
second.map.masked.positions	—	A string of 0s and 1s that indicate the read positions to be masked during the second mapping pass. There is no penalty in mapping towards positions that are masked. Default: blank Leaving the parameter blank results in no masking. Leave the parameter blank for global style mapping (previously called "classic" mapping in earlier versions). (Masking replaces read trimming.) If set, the length of the string must match the anchor length or seed length (the first value of scheme parameter such as 25.2.0). With a 25-mer read and a 25.2.0 scheme, an example mask is 1111111111111111111100000 (20 1s, then 5 0s). The bases corresponding to the 0s are ignored. Under the 0s portion of the mask, any mismatches do not count toward the mismatch penalty.
second.map.max.hits.tag1	50	The maximum number of alignments for a read on the second mapping pass. This setting gives the maximum number of hits that are reported in the mapping output (even if more hits are found). Allowed values: Integers >= 1. The module fails if the value is not within this range.
second.map.max.hits.tag2	50	The maximum number of alignments for a read on the second mapping pass. This setting gives the maximum number of hits that are reported in the mapping output (even if more hits are found). Allowed values: Integers >= 1. The module fails if the value is not within this range.
second.map.mismatch.penalty	-2	A scoring penalty for mismatches. Used in local alignment mode. Must be a negative number or zero. Allowed values: Integers <= 0. If the absolute of the penalty is set to greater than the read length, effectively no mismatches are allowed on this pass.
second.map.scheme.unmapped.1	0.0.0	The scheme used for unmapped reads of a length 1–24, for the second mapping stage. See "Mapping schemes" on page 159.

Table 27 Internal mapping parameters *(continued)*

Parameter name	Default value	Description
second.map.scheme.repetitive.1	—	The scheme used for repetitive reads of a length 1–24, for the second mapping stage. See "Mapping schemes".
second.map.scheme.unmapped.25	20.1.0	The scheme used for unmapped reads with a length 25–34, for the second mapping stage. See "Mapping schemes".
second.map.scheme.repetitive.25	—	The scheme used for repetitive reads with a length 25–34, for the second mapping stage. See "Mapping schemes".
second.map.scheme.unmapped.35	20.1.0	The scheme used for unmapped reads with a length 35–49, for the second mapping stage. See "Mapping schemes".
second.map.scheme.repetitive.35	—	The scheme used for repetitive reads with a length 35–49, for the second mapping stage. See "Mapping schemes".
second.map.scheme.unmapped.50	20.1.0:30	The scheme used for unmapped reads with a length 50–59, for the second mapping stage. See "Mapping schemes".
second.map.scheme.repetitive.50	—	The scheme used for repetitive reads with a length 50–59, for the second mapping stage. See "Mapping schemes".
second.map.scheme.unmapped.60	20.1.0:30	The scheme used for unmapped reads with a length 60 to 74, for the second mapping stage. See "Mapping schemes".
second.map.scheme.repetitive.60	—	The scheme used for repetitive reads with a length 60–74, for the second mapping stage. See "Mapping schemes".
second.map.scheme.unmapped.75	20.1.0:30	The scheme used for unmapped reads with a length of 75 or larger, for the second mapping stage. See "Mapping schemes".
second.map.scheme.repetitive.75	—	The scheme used for repetitive reads with a length of 75 or larger, for the second mapping stage. See "Mapping schemes".
second.map.scheme.unmapped. variable	20.1.0:30	The scheme used for unmapped reads of variable length coming from ECC trimmed base-space reads, for the second mapping stage. See "Mapping schemes".
second.map.scheme.repetitive. variable	—	The scheme used for repetitive reads of variable length coming from ECC trimmed base-space reads, for the second mapping stage. See "Mapping schemes".

Mapping schemes

The mapping scheme parameters, listed in Table 27, determine the behavior of the mapping module. The LifeScope™ Software mapping module supports two types of mapping, global mapping and seed-and-extend mapping. These two approaches are described in more detail below.

- **Global mapping** – With this approach the seed length matches the full read length, and the seed anchors at the beginning of the read. For example, global mapping for 50-bp reads uses a scheme such as `50.6.0`. (In this example, 50, the seed length, matches the read length, and 0, the seed start position, starts at the beginning of the read.)
- **Seed-and-extend** – The initial alignment step that locates short matches between a read and the reference sequence. The seed specifies the length of the attempted match. This approach is also called local alignment.

Global mapping

To map a 50 bp read using global mapping, a typical scheme to use is 50.6.0. This allows for up to 6 mismatches. Because the seed length equals the length of the read, there is no extension.

Seed-and-extend

With the seed-and-extend approach, a mapping scheme parameter value such as `25.2.0` is interpreted as follows:

- **First value** – The seed length (`25`).
- **Second value** – The quantity of allowed mismatches in the seed (`2`).
- **Third value** – The start site of the seed within the read (`0`).

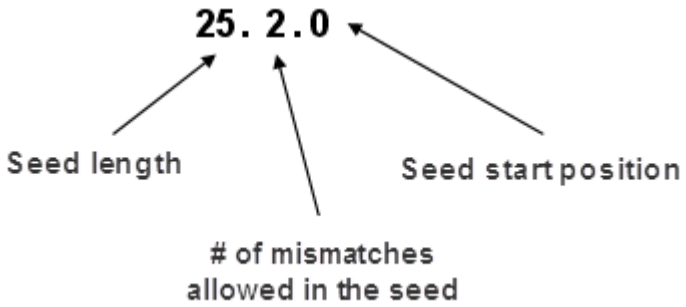


Figure 6 Seeds for local alignments

For example, with a `25.2.0` seed extend scheme, the mapping module examines the first 25 bases of a 50 bp read, and if that portion aligns to the reference with two or fewer mismatches, then the mapping module attempts seed extension. For any alignment, we try two different extensions and report the one with higher score. The two extension approaches are described below:

- **Ungapped extension** – Extends from the anchor alignment, to find the best ungapped local alignment with a score *S*. The alignment length in this case can be shorter than the read length.
- **Gapped extension** – Looks for one indel, following two different algorithms that are controlled by the parameters `first.map.gapped.algorithm` and `second.map.gapped.algorithm`. Only one of the algorithms can be used at a time. These algorithms are:
  - **Local indel finding extension** – Extends from the anchor alignment, doing gapped extension with a modified Smith-Waterman-like algorithm. In this algorithm, we use a matching score (same as in ungapped extension) and gap penalty (`*.map.gapped.penalty`), to find a best local alignment within user specified size restriction (`*.map.gapped.insertion/deletion`). If this alignment has a score larger than `S` (the best ungapped alignment extended from the same anchor alignment), the alignment is reported. The alignment length in this case can be shorter than the read length.
  - **Global indel finding extension** – Extends from anchor alignment, does a full-length gapped extension with the allowed number of mismatches (`*.map.gapped.error.indel`), and within user-specified size restrictions (`*.map.gapped.insertion/deletion`), window restrictions (`*.map.gapped.seed.window.*`), and edge length restrictions

(*`.map.gapped.edge.length.insertions/deletions`). If this alignment has a score larger than *S*, it is reported. This algorithm is the same as what was used with the frag indel module in earlier versions of the software.

For information on sliding periodic seeds, refer to the publication at this site:

http://bioinformatics.oxfordjournals.org/content/25/19/2514.full

Unmapped and repetitive schemes

Mapping schemes have two categories, a scheme for unmapped reads (*`.map.scheme.unmapped.`*) and a scheme for repetitive reads (*`.map.scheme.repetitive.`*). Either type of scheme can also be either single-anchor or multi-anchor.

A single-anchor scheme is represented by a comma-separated list of values, where each value is in the format *seed_length.mismatches.seed_start_position* (see Figure 6 on page 160). The following apply to both unmapped and repetitive single-anchor schemes:

- **Scheme 25.2.0,25.2.20** – First maps using scheme 25.2.0 and only for those reads that do not have an alignment, maps a second round using the scheme 25.2.20.
- **Scheme 25.2.0,25.2.20,25.2.30** – First maps using scheme 25.2.0 and only for those reads that do not have an alignment, maps a second round using the scheme 25.2.20. The reads that do not map to 25.2.20 either are then tried with scheme 25.2.30.
- When both repetitive and unmapped are specified, the repetitive schemes are applied only to the repetitive reads from first unmapped scheme. Figure 7 depicts the logic followed for an unmapped scheme of 25.2.0, 25.2.20 and a repetitive scheme of 35.2.0, 35.2.30.

Multi-anchor schemes are represented by multiple colon-separated seed start positions, in the format *seed_length.mismatches.seed_start_pos1:seed_start_pos2*. The following apply to both unmapped and repetitive multi-anchor schemes:

- **Scheme 25.2.0:20** – Simultaneously maps using scheme 25.2.0 and 25.2.20, and reports all alignments.
- **Scheme 25.2.0:20:30** – Simultaneously maps using scheme 25.2.0, 25.2.20 and 25.2.30, and reports all alignments.
- Figure 8 depicts the logic followed for an unmapped scheme of 25.2.0:20 and a repetitive scheme of 35.2.0:30.

The Figures 7 and 8 describe how the mapping scheme parameters are used in the mapping module. su1+su2 and sr1+sr2 are in the form of (L,m,b1)+(L,m,b2), where *L* is the seed length, *m* is number of mismatches allowed in the seed, *b1* and *b2* are the seed starting positions, *H* is the number of hits, and *Z* is the maximum number of alignments reported by the mapping program.

In these figures, *su* refers to a mapping scheme for matching unmapped reads. And *sr* refers to a mapping scheme for matching repetitive reads.

When H < Z, the read is not in a repetitive region. When H >= Z, the read matches too many locations on the reference, so it is considered a repetitive read, or sequenced from a repetitive region.

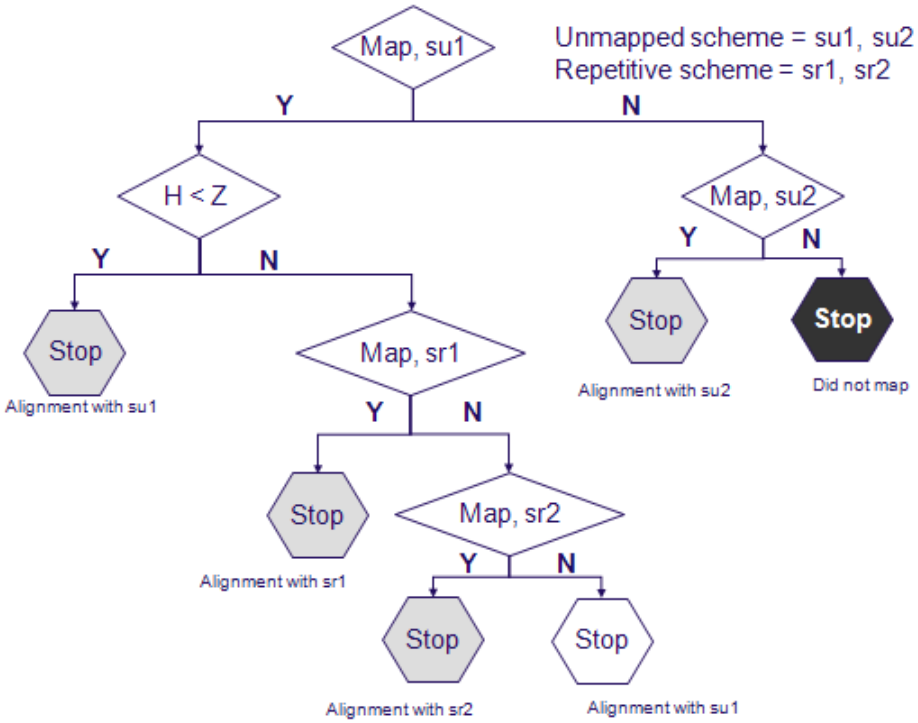


Figure 7 Single-anchor mapping for unmapped and repetitive reads

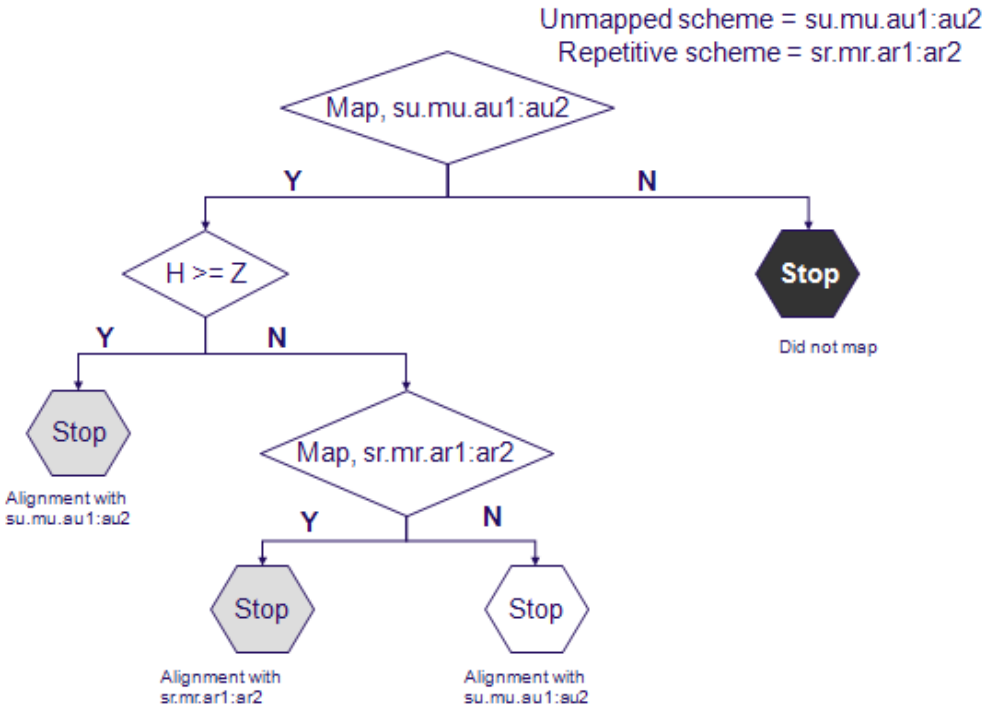


Figure 8 Multi-anchor mapping for unmapped and repetitive reads

Pairing algorithm

Pairing parameters

Table 29 lists parameters for the pairing algorithm.

In order to change a parameter value, use the `set param` shell command to replace the line `[optionally change mapping defaults after this line]` in the example commands in "Examples of how to run a mapping analysis" on page 138. For instance, to adjust the default minimum insert size, change the parameter `default.minimum.insert.size` with this command:

```
set default.minimum.insert.size 200 secondary/pair.mapping.ini.
```

Table 28 Pairing parameters

Parameter name	Default value	Description
default.minimum.insert.size	0	The minimum insert size allowed for the insert size range estimate. This value is used if this measurement fails. Allowed values: Integers 0–10000.
default.maximum.insert.size	20000 for LMP 2000 for PE	The maximum insert size allowed for the insert size range estimate. This value is used if this measurement fails. Allowed values: Integers 0–10000.
minimum.points.for.size.stat	1000000	Minimum points in insert size distribution points to do compute statistics.
memory.per.bam.merge	3	Gigabytes of memory for one BAM merge thread.

Internal pairing parameters

Table 29 lists pairing parameters that we do not recommend changing.

The first stage of pairing does pairing without finding gaps. This stage is followed by 4 stages of pairing that allow gaps. Values in a format such as `0:3:14:14:0` are a list of one value for each indel pass, separated by colons (:).

Table 29 Internal pairing parameters

Parameter name	Default value	Description
pairing.gap.on	1	Enables indel searching in pairing analysis. Allowed values are: • **0**: Do not include indel search. • **1**: Include indel search.
pairing.anchor.max.mismatches	4, if indel searching is off 4:5:5:3:5, with indel searching	Total mismatches for both reads of a pair. If indel searching is on, then this value is specified as a list of one number for each indel pass, separated by colons (:).
pairing.gap.min.non.matched.length	9	Used to determine whether the pair goes through an indel pass. If the regular pairing finds a good pair of alignments that are both longer than read length minus this value, then the pair does not need an indel finding run. Allowed values: Integers >= 0.

Table 29 Internal pairing parameters *(continued)*

Parameter name	Default value	Description
pairing.gap.max.mismatches.tag1	5	Number of mismatches allowed in the extension part of the tag 1. (The extension typically is 5% to 10% of the read length.) pairing.gap.max.mismatches.tag1 is set to 2 if the read length is less than the value of the parameter min.length.for.aggressive.gapped.mapping.
pairing.gap.max.mismatches.tag2	5	Number of mismatches allowed in the extension part of the tag 2. pairing.gap.max.mismatches.tag2 is set to 2 if the read length is less than the value of the parameter min.length.for.aggressive.gapped.mapping.
pairing.gapped.max.hits	50:10	For ungapped and gapped alignment determination during rescue, this value determines the maximum number of rescues attempted for each read. If a read has more than this number of hits, rescue is not performed on this bead. The first value is used for ungapped rescue. The second value is used for gapped rescue
pairing.output.uniqueness.type	–1	Whether to output non-unique hits. Allowed values: • **–1**: A clear zone determines uniqueness. 1 or 2 best pairs are output, depending on whether it is unique. • **0**: All good pairs are output. • **1**: Only unique good pairs are output.
pairing.annotation.type		Whether to output non-AAA pairs. Allowed values: • **0**: Do not output non-AAA pairs. • **1**: Output non-AAA pairs, only if pairing and rescue do not find good pairs.
pairing.mismatch.penalty	–2.0	A single penalty applied to alignments, to compare the significance between alignments. Helps evaluate the alignments. Allowed values: Floats <0.
pair.uniqueness.threshold	10.0	This value defines the clear zone. This value is considered only when pairing.output.uniqueness.type is set to –1. If the score of the best alignment is this threshold more than the score of the second best alignment, the best alignment is considered uniquely placed. A threshold of zero means that this uniqueness is strictly adhered to, regardless of the alignment's relative scores. Allowed values: Floats >=0.
General pairing indel parameters For these parameters each value is a colon-separated list of five numbers, which are used, in order, for the five pairing passes. The first pass of pairing does pairing without finding gaps. This pass is followed by four passes of pairing that allow gaps. These values are potentially overridden by size-specific indel parameters in other sections of this table.		
pairing.gap.max.deletion.size	0:11:0:0:500	The maximum deletion size allowed during pairing.
pairing.gap.max.insertion.size	0:4:15:21:0	The maximum insertion size allowed during pairing.

Table 29 Internal pairing parameters *(continued)*

Parameter name	Default value	Description
pairing.gap.edge.length.deletions	0:3:0:0:20	The gap edge length for deletions during pairing.
pairing.gap.edge.length.insertion	0:3:14:14:0	The gap edge length for insertions during pairing.
Pairing indel parameters for tag lengths 75–35		
For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with F3 lengths of 75 and above, and R3 or F5 lengths of 35–59.		
pairing.gap.max.deletion.size.75-35	0:11:0:0:500	The maximum deletion size allowed for these lengths.
pairing.gap.max.insertion.size.75-35	0:4:15:21:0	The maximum insertion size allowed for these lengths.
pairing.gap.edge.length.deletions.75-35	0:3:0:0:20	The gap edge length for deletions for these lengths.
pairing.gap.edge.length.insertion.75-35	0:3:14:14:0	The gap edge length for insertions for these lengths.
Pairing indel parameters for tag lengths 60–60		
For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with both tags with lengths of 60–74.		
pairing.gap.max.deletion.size.60-60	0:11:0:0:500	The maximum deletion size allowed for these lengths.
pairing.gap.max.insertion.size.60-60	0:4:15:21:0	The maximum insertion size allowed for these lengths.
pairing.gap.edge.length.deletions.60-60	0:3:0:0:20	The gap edge length for deletions for these lengths.
pairing.gap.edge.length.insertions.60-60	0:3:14:14:0	The gap edge length for insertions for these lengths.
Pairing indel parameters for tag lengths 50–50		
For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with both tags with lengths of 50–59.		
pairing.gap.max.deletion.size.50-50	0:11:0:0:500	The maximum deletion size allowed for these lengths.
pairing.gap.max.insertion.size.50-50	0:4:15:21:0	The maximum insertion size allowed for these lengths.
pairing.gap.edge.length.deletions.50-50	0:3:0:0:20	The gap edge length for deletions for these lengths.
pairing.gap.edge.length.insertions.50-50	0:3:14:14:0	The gap edge length for insertions for these lengths.
Pairing indel parameters for tag lengths 35–35		
For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with both tags with lengths of 35–49.		
pairing.gap.max.deletion.size.35-35	0:11	The maximum deletion size allowed for these lengths.
pairing.gap.max.insertion.size.35-35	0:4	The maximum insertion size allowed for these lengths.
pairing.gap.edge.length.deletions.35-35	0:3	The gap edge length for deletions for these lengths.
pairing.gap.edge.length.insertions.35-35	0:3	The gap edge length for insertions for these lengths.
Pairing indel parameters for tag lengths 25–25		
For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with both tags with lengths of 25–34.		
pairing.gap.max.deletion.size.25-25	0:11	The maximum deletion size allowed for these lengths.
pairing.gap.max.insertion.size.25-25	0:4	The maximum insertion size allowed for these lengths.
pairing.gap.edge.length.deletions.25-25	0:3	The gap edge length for deletions for these lengths.
pairing.gap.edge.length.insertions.25-25	0:3	The gap edge length for insertions for these lengths.

Table 29 Internal pairing parameters *(continued)*

Parameter name	Default value	Description
Pairing indel parameters for tag lengths 50–35		
For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with F3 lengths of 50–74, and R3 or F5 lengths of 35–59.		
The first pass of pairing does pairing without finding gaps. The second pass of pairing allows gaps.		
pairing.gap.max.deletion.size.50-35	0:11	The maximum deletion size allowed for these lengths.
pairing.gap.max.insertion.size.50-35	0:4	The maximum insertion size allowed for these lengths.
pairing.gap.edge.length.deletions.50-35	0:3	The gap edge length for deletions for these lengths.
pairing.gap.edge.length.insertions.50-35	0:3	The gap edge length for insertions for these lengths.
Pairing indel parameters for tag lengths 50–25		
For pairs of these specific lengths, parameters in this section are used instead of the general pairing indel parameters. This section applies to pairs with F3 lengths of 50–74, and R3 or F5 lengths of 25–34.		
The first pass of pairing does pairing without finding gaps. The second pass of pairing allows gaps.		
pairing.gap.max.deletion.size.50-25	0:11	The maximum deletion size allowed for these lengths.
pairing.gap.max.insertion.size.50-25	0:4	The maximum insertion size allowed for these lengths.
pairing.gap.edge.length.deletions.50-25	0:3	The gap edge length for deletions for these lengths.
pairing.gap.edge.length.insertions.50-25	0:3	The gap edge length for insertions for these lengths.

Indel sizes

The following mapping parameters affect the indel sizes that can be detected in fragment libraries during tertiary processing:

- `first.map.gapped.deletion`, default value 19
- `first.map.gapped.insertion`, default value 4
- `second.map.gapped.deletion`, default value 19
- `second.map.gapped.insertion`, default value 4

The following pairing maximum insertion and deletion parameters affect the size limits of indels that can be discovered in paired data during tertiary processing:

- `pairing.gap.max.deletion.size`
- `pairing.gap.max.insertion.size`
- `pairing.gap.max.deletion.size.75-35`, etc
- `pairing.gap.max.insertion.size.75-35`, etc

For example, the default value for is `pairing.gap.max.deletion.size` is `0:11:0:0:500`. This default sets to maximum supported small indel deletion to 500. The parameters ending in library sizes (`pairing.gap.max.deletion.size.75-35`, etc) can be used to reset the value for a specific library size.

Mapping output files

The resequencing mapping module generates a BAM file containing alignments in coordinate order. For information about the BAM file, see Appendix C, "File Format Descriptions" on page 493.

The number of BAM files generated depends on the input XSQ files. One BAM file is generated per read-set. If the input data is an XSQ file containing 96 barcodes, then the mapping module generates at least 96 output BAM files. Beads that are unclassified in any barcode are output into a separate additional BAM file. If `create.unmapped.bam.files` is set to `TRUE`, then an additional 96 output BAM files corresponding to the unmapped reads are also generated.

The filenames for the output BAM files are created using information from the input XSQ file, including: file base name, file id, index name, and index id. The BAM files are named according to the following patterns:

- Non-indexed BAM files: *xsqname-fileID*-1.bam
- Indexed BAM files: *xsqname-fileID*-idx_*bcIndex-bcID*.bam

The fields in the filenames are:

- **xsqname** – The XSQ file base name, without the .xsq extension.
- **fileID** – The file ID for internal XSQ file tracking.
- **bcIndex** – The barcode index.
- **bcID** – The barcode identifier for internal barcode tracking.

The directory structure for mapping output is as follows:

```
outputs/${analysis.output.dir}/
      <mappingIniFileBasename>/
            <sampleName1>/bamfilename1.bam
            <sampleName2>/bamfilename2.bam
             ...
```

The directory name <*mappingIniFileBasename*> is the basename of the mapping INI file. The defaults are `fragment.mapping` and `pair.mapping`.

The string <*sampleName**> is determined by the sample description for the particular read-set.

The mapping output files are used by the BAMStats mapping statistics module and by the LifeScope™ Software tertiary analysis modules.

Mapping statistics

Mapping statistics occur after mapping as an optional post-processing step named BAMStats. BAMStats is on by default in all standard workflows. BAMStats accepts the output of the mapping step and generates statistics files to provide an in-depth understanding of the experimental data and to better detect the presence of anomalies. LifeScope™ Software shell users can display the mapping statistics output data as a chart with a spreadsheet program or other third-party program. A subset of the output from mapping can be visualized in the LifeScope™ Software UI as a series of line and bar charts, if the analysis is run in the projects repository.

BAMStats does not accept 1.x BAM files as input. 1.3 BAM files must be converted to 2.0 BAMs before being used with BAMStats.

Mapping statistics parameters

Table 30 lists the parameters which control the BAMStats output.

Table 30 BAMStats parameter description

Parameter name	Default value	Description
Optional parameters		
bamstats.maximum.coverage	10000	Defines the maximum coverage allowed for locations in the reference. Locations with coverage more than the maximum coverage value are ignored during coverage calculations. Allowed values: Integers >= 1.
bamstats.maximum.isize	100000	The maximum inset size for LMP and PE libraries. Reads with an insert size more then the specified value are ignored for Insert Range Report calculation. Allowed values: Integers 1–100000.
bamstats.wig.primary.only	1	Use only primary alignments for coverage in WIG file format. Allowed values: • **0**: Do not restrict coverage in WIG file format to only primary alignments. • **1**: Restrict coverage in WIG file format to only primary alignments.
bamstats.wig.binsize	100	The bin size for coverage in WIG file format. Allowed values: Integers 1–100000.
bamstats.bin.isize	100	The bin size for insert range distribution. Allowed values: Integers 1–100000.
bamstats.wig.combined.report.both.strands	1	Whether to generate a WIG format coverage file with combined data from both the strands. Allowed values: • **0**: Do not generate wig files. • **1**: Generate the combined coverage file.
Optional resource parameters		
wall.time	120	Total time for the process to complete.
java.heap.space	13000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.

Table 31 lists BAMStats parameters that we do not recommend changing.

Table 31 BAMStats internal parameter description

Parameter name	Default value	Description
bamstats.run	1	Whether or not to run the BAMStats module. Allowed values: • **0**: Do not run the BAMStats module. BAMStats statistics are not generated. • **1**: Run the BAMStats module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
bamstats.group.stats	1	Enable combined statistics covering all read-sets in the group or sample. Allowed values: • **0**: Only statistics for BAM files are generated. • **1**: Also generate combined statistics for the group or sample.
bamstats.maximum.mismatches	100	The maximum mismatches allowed in the alignments. Any alignment with more then the specified number of mismatches is ignored while generating reports related to number of mismatches. Allowed values: Integers 0–100.
bamstats.maximum.baseqv	100	Max base quality values. Any base with base quality value more then the specified value is ignored while generating reports. Allowed values: Integers 0–100.
bamstats.maximum.mappingqv	255	Maximum mapping quality value. Any alignment with mapping quality value more than the specified value is ignored. Allowed values: Integers 0–255.
bamstats.wig.minimum.mappingqv	2	Defines the minimum mapping quality allowed for coverage in WIG format. Any alignment with a mapping quality value less than this value is ignored. Allowed values: Integers 0–100.
bamstats.enable.probe.position	0	Enable probe and position error reports. Allowed values: • **0**: Do not generate probe and position error reports. • **1**: Also generate probe and position error reports. **Note:** Because these reports are recommended for SNPs analysis, bamstats.ini files in whole genome and targeted resequencing standard workflows have bamstats.enable.probe.position=1.

Summary of mapping statistics output

File formats

For every input BAM file, a set of statistics files are generated. These files are in CHT, CSV, TXT, and WIG formats. Each CHT file corresponds to one displayed chart. A CHT file specifies the type of chart, the displayed range of each axis, and the data points, without using external references.

The CHT file format is an internal file format based on the CSV file format, with addition header information included. CHT header information is the following:

```
# name:
# type: scatter2d | pie | vbar | line
# title:
# xaxisname:
# yaxisname:
```

```
# xrange: <min>:<tickinterval>:<max>
# yrange: <min>:<tickinterval>:<max>
XAXISNAME, SERIES1NAME, SERIES2NAME, ...
x1, y1.1, y1.2, ...
x2, y2.1, y2.2, ...
x3, y3.1, y3.2, ...
```

The wiggle format (.wig) is a public format typically used for coverage. Visit their site for more information:

hgdownload.cse.ucsc.edu/goldenPath/help/wiggle.html

A genome browser such as the Integrative Genomics Viewer (IGV) can be used to visualize the coverage. For information is available from their site:

www.broadinstitute.org/igv/

For a collection of input BAM files that belong to a sample, a set of cumulative statistics files are generated. The cumulative statistics files are also in CHT, CSV, TXT, and WIG formats. The cumulative statistics can be visualized in the LifeScope™ Software UI.

Directory structure

The output of the BAMStats module has the following directory structure:

```
bamstats/
      <sampleName>/*.cht, *.tbl
      <sampleName>/<bam>/.*cht
      <sampleName>/<bam>/Misc/*.csv, *.txt, *.wig
      <sampleName>/Misc/*.csv, *.txt
```

The position error files and probe errors files are created in the BAM file directories.

Overview

BAMStats generates a comma-separated summary file (BAMfilename-summary.tbl) that summarizes key mapping quality statistics per input BAM file. The summary report contains a snapshot of statistics in all BAM files present in this sample. This file contains one row for each input BAM file in the sample. The summary file is displayed in the LifeScope™ Software UI. See "Summary file" on page 175.

```
<sampleName1>/BAMfilename-summary.tbl
```

Following directories and reports contain the cumulative statistics from all BAM files that belong to this sample. The `Misc` folder is not displayed in UI.

```
<sampleName1>/*.cht
<sampleName1>/Misc/*.csv, *.wig, *.txt
```

Following statistics are generated per BAM file in the mapping directory. These reports are not displayed in the UI.

```
<sampleName1>/<BAMfilename>/.*cht
<sampleName1>/<BAMfilename>/Misc/*.csv, *.txt, *.wig
```

In a sample, some of the BAM files possibly represent unhealthy DNA or RNA, causing the cumulative statistics to look poor. The summary tbl file is the unified location for examining the quality of data of all BAM files in the sample.

If the data for a particular BAM file is not as expected, look at the directory-level reports for that BAM file, for details.

Mapping statistics output files

This section describes the mapping statistics files generated by the BAMStats module. When statistics reports are separated by tag type, the report's file name includes the tag in the file name. The tags used in file names are:

Table 32 Tags in mapping statistics output file names

Library type	Tag
Fragment	F3
Mate-pair	F3
	R3
Paired-end	F3
	F5-P2

The output files generated by BAMStats include the name of the BAM file. The file name pattern is:

BAMFileName-fileID-barcodeID.StatisticsReportName.tag.extension

This string is referred to as *prefix* in Table 33, the mapping statistics output table. In the description of mapping statistics output file names in Table 33, the string *tag* is used to refer to the tags listed in Table 32.

Table 33 Mapping statistics output files

Report	Description, axis information, filename
Alignment Length Distribution	
	A bar plot giving the distribution of alignment lengths found in various tags used in alignment. The report is separated by tag type to provide visibility into the accuracy of individual tags. Only the primary alignment for each bead is considered in calculating the distribution.
	Y axis: Frequency X axis: Alignment length (from 0 to the maximum read length)
	Output file name: *prefix*.Alignment.Length.Distribution.*tag*.cht
Base Mismatch Distribution	
	A bar plot giving the distribution of total number of mismatches found in various tags used in alignment. The bins are summed over all alignment lengths. Only the primary alignment for each bead is considered in calculating the distribution.
	Y axis: Frequency X axis: Number of mismatches (from 0 to the maximum mismatches allowed)
	Output file name: *prefix*.BaseMismatch.Distribution.*tag*.cht
Distribution of Alignment Length and Number of Mismatches in Tags	

Table 33 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
	A report providing a simultaneous picture of the alignment length and number of mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. **Note:** This report is located in the `Misc` folder, and is not displayed in the UI.
	Y axis: Alignment length (from 0 to the max read length) X axis: Number of mismatches (from 0 to the maximum mismatches allowed) Z axis: Frequency
	Output file name: *prefix*.AlignmentLength.Mismatch.*tag*.csv
Distribution of Mismatches by Position	
	A bar plot providing a distribution of errors (mismatches to reference) by position within the read. Only the primary alignments for each bead are used to generate this distribution. The base-space report is generated on every run.
	Y axis: Frequency X axis: Position (from 0 to maximum read length)
	Output file name: *prefix*.Mismatches.By.Position.BaseSpace.*tag*.cht, *and* (only if color-space data is present in the BAM file) *prefix*.Mismatches.By.Position.ColorSpace.*tag*.cht
Color Mismatch Distribution	
	A bar plot providing a distribution of the total number of color mismatches in various tags used in alignment. Only the primary alignments for each bead are used to generate this distribution. This report is only generated if mapping is done in color space.
	Y axis: Frequency X axis: Number of mismatches (from 0 to the maximum mismatches allowed)
	Output file name: *prefix*.ColorMismatches.Distribution.*tag*.cht
Distribution of Alignment Length and Color Mismatches	
	A report providing a simultaneous picture of the alignment length and number of color mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. This report is only generated if mapping is done in color space. **Note:** This report is located in the `Misc` folder, and is not displayed in the UI.
	Y axis: Alignment length (from 0 to the max read length) X axis: Number of mismatches (from 0 to the maximum mismatches allowed) Z axis: Frequency
	Output file name: *prefix*.AlignmentLength.ColorMismatches.*tag*.csv
Base QV Distribution	
	A bar plot providing a distribution of base quality values generated using the reference assisted error correction/base conversion algorithm. The base QV distributions are separated for each tag as quality of individual tags could be very different.
	Y axis: Frequency X axis: Base QV (from 0 to the maximum QV)
	Output file name: *prefix*.BaseQV.*tag*.cht
Base QVs by Position	

Table 33 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
	A report providing a distribution of base quality values by individual base positions. This report identifies if certain base positions (particularly towards the end of the read) have poor base quality values. All the tags (F3/R3/F5-P2) used in the alignment are combined into a single bin.
	Y axis: Base QV X axis: Base position (from 0 to read length) Z axis: Frequency
	Output file name: *prefix*.BaseQV.by.Position.csv
Distribution of Mismatches by Base QV	
	A line plot providing a distribution of errors (mismatches to reference) by base quality values bins.
	Y axis: Percentage of total mismatches X axis: Base QV (from 0 to maximum QV) **Note:** This report gives, for each QV, a measure of the raw mismatches at positions with that QV. This information does not measure whether the base QVs generated are well calibrated to the probability of error in that particular base position.
	Output file name: *prefix*.Mismatches.By.BaseQV.*tag*.cht
Distribution by Mapping QVs by Tag	
	A bar plot providing a distribution of mapping quality values for individual tags (F3/R3/F5-P2). Only the primary alignment for each bead is used in calculating this distribution.
	Y axis: Frequency X axis: Mapping QV (from 0 to maximum mapping QV)
	Output file name: *prefix*.MappingQV.*tag*.cht
Distribution by Pairing QVs	
	A bar plot providing a distribution of pairing quality values. Only the primary alignment for each bead is used in calculating this distribution.
	Y axis: Frequency X axis: Pairing QV (from 0 to maximum pairing QV)
	Output file name: *prefix*.PairingQVs.cht
Coverage Report	
	A line plot providing a distribution of coverage obtained after mapping/pairing. Only the primary alignment for each bead is used in this calculation.
	Y axis: Frequency X axis: Coverage (from 0 to number of reads)
	Output file name: *prefix*.Coverage.cht
Coverage Report by Chromosome (Contig) and Base Windows	
	A line plot providing a distribution of coverage within each reference window. The coverage is calculated within each window along a reference chromosome. The window size can theoretically be anywhere from a single base to the contig length. However the calculations of base level coverage are computationally expensive and less interpretable as the window size increases. Only the primary alignment for each bead is used in this calculation.
	Y axis: Frequency X axis: Coverage (from 0 to number of reads)
	Output file name: *prefix*.Coverage.By.Chromosome.contig*nn*.cht

Table 33 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
Coverage by Strand	
	A line plot providing the distribution of coverage within each reference window separated by reference strand (+/-). Only the primary alignment for each bead is used in this calculation.
	Y axis: Frequency X axis: Coverage (from 0 to number of reads)
	Output file name: *prefix*.Coverage.By.Strand.Positive.cht, *prefix*.Coverage.By.Strand.Negative.cht
Coverage files	
	Coverage reports in wiggle format. For each genome position, reports the number of reads that cover (map to or span) the position. Because reporting coverage for each position results in very large files, coverage is reported for bins, with each bin spanning a user-defined number of bases. For each bin, the mean coverage of all the positions in that bin is reported. The parameter `bamstats.wig.binsize` controls the size of the bins in this file. Each coverage file includes a header as the first line. The header lines follow this pattern: track type=wiggle_0 name=<*chrname*> description=<*coverage from positive/negative/both strand*> visibility=full color=0,0,255 fixedStep chrom=<*chrname*> start=<*startpos*> step=<*binsize*> span=<*binsize*> See also "Coverage files" on page 177.
	Output file name: coverage_chr*nn*.pos.wig, coverage_chr*nn*.neg.wig
Insert Range Distribution	
	A line plot providing the distribution of insert sizes for paired data (PE and LMP library types).
	Y axis: Frequency X axis: Insert Range bins
	Output file name: *prefix*.Insert.Range.Distribution.cht
Distribution of Read Pair Types	
	For paired data, this report calculates the distribution of read pair types (AAA, AAB, C**, etc.).
	Y axis: Frequency X axis: Read pair type
	Output file name: prefix.ReadPair.Type.cht
Pairing Statistics	
	A pie chart showing the percentages of the following F3, R3 (or F5) combinations, for paired data: • Mapped, Mapped • Mapped, Unmapped • Unmapped, Mapped • Unmapped, Unmapped • Mapped, Missing • Missing, Mapped • Unmapped, Missing • Missing, Unmapped
	Output file name: *prefix*.Pairing.Stats.cht
Unique Start Positions	

Table 33 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
	This report lists the following: • The total number of unique start positions. • The average number of unique start positions per reference position. See also "Unique Start Positions" on page 176.
	Output file name: *BAMfilename*-summary.txt, *Samplename*.-summary.txt
Summary Report	
	The summary file provides statistics for each BAM file in the sample. See "Summary file".
	Output file name: *filename*.-summary.tbl
Position error files	
	One or more tab-delimited text files generated in the directory containing the input BAM file. This file is used by the diBayes module for SNP analysis. The file records the frequencies of dicolor mismatches between reads and the reference at different positions in a read. The following position error files are generated, based on library type: • **Fragment runs** – An F3 position error file is created. • **Mate-pair runs** – Both F3 and R3 position error files are created. • **Paired-end runs** – Both F3 and F5 position error files are created. See also "Position error file" on page 178.
	Output file names: *BAMfilename*_positionErrors_*tag*_basespace.txt, *BAMfilename*_positionErrors_*tag*_colorspace.txt
Probe error files	
	One or more tab-delimited text files generated in the directory containing the input BAM file. This file is used by the diBayes module for SNP analysis. The probe error file records the frequencies of dicolor mismatches between the reads and the reference as a function of different 6-mer probes. The following probe error files are generated, based on library type: • **Fragment runs** – An F3 probe error file is created. • **Mate-pair runs** – Both F3 and R3 probe error files are created. • **Paired-end runs** – Both F3 and F5 probe error files are created. See also "Probe error files" on page 178
	Output file names: *BAMfilename*_probeErrors_*tag*_colorspace.txt

Mapping statistics example output

This section provides example output of some mapping statistics output files.

Summary file

The summary file provides statistics for each BAM file in the sample. This list describes labels used in the summary file:

- **NumFragmentsTotal** – The total number of beads, before any filtering on the instrument. This value is also the fragment count in the input XSQ file.
- **NumFragmentsPassingFilters** – The number of beads that pass filtering on the instrument.
- **NumMapped** – The number of reads with primary alignment.

- **% filtered mapped** – The number of primary reads divided by the number of beads passing instrument filtering (`NumMapped` / `NumFragmentsPassingFilters`).
- **% total mapped** – The number of primary reads divided by the number of total beads (`NumMapped` / `NumFragmentsTotal`).
- **Megabases of coverage** – The sum of alignment lengths, for primary alignments only.

The number of beads filtered out in the instrument is found by subtracting the number of beads that pass filtering from the total number of beads:

`NumFragmentsTotal – NumFragmentsPassingFilters`

The following are column headings for a summary file. The `PredictedInsertSize` and `Tag2` fields only are present with paired data.

```
BamFileName,IsColorInBam,IsBaseInXSQ,IsECC,LibraryType,
ReadLength,Megabases of Coverage,PredictedInsertSize,
NumFragmentsPassingFilters,NumFragmentsTotal,
Tag1-NumMapped,
Tag1- % total Mapped,
Tag1- % filtered mapped,
Tag2-NumMapped,
Tag2- % total Mapped,
Tag2- % filtered mapped,
(Tag1-AlignmentLength;Min;Max;Avg;Median;StdDev),
(Tag1-NumBaseMismatches;Min;Max;Avg;Median;StdDev),
(Tag1-MappingQV;Min;Max;Avg;Median;StdDev),
(Tag1-BaseQV;Min;Max;Avg;Median;StdDev),
(Tag1-NumColorMismatches;Min;Max;Avg;Median;StdDev),
(Tag2-AlignmentLength;Min;Max;Avg;Median;StdDev),
(Tag2-NumBaseMismatches;Min;Max;Avg;Median;StdDev),
(Tag2-MappingQV;Min;Max;Avg;Median;StdDev),
(Tag2-BaseQV;Min;Max;Avg;Median;StdDev),
(Tag2-NumColorMismatches;Min;Max;Avg;Median;StdDev)
(Coverage;Min;Max;Avg;Median;StdDev)
```

The following is an example line of output in the summary file:

```
example.bam Y N N FRAG 50 2184653 100000 100000 51536 51.54
51.54 (F3;25;50;42.39;45;6.48) (F3;0;50;2.98;0;9.24)
(F3;0;100;65.43;72;25.31) (F3;1;41;35.09;41;11.65)
(F3;0;9;1.20;1;1.39) (0;399;0.43;0;5.96)
```

Unique Start Positions

The following is an example of a Unique Start Position report, for fragment data:

```
Unique Start Positions F3
Total Number of Unique Start Positions: 16536 (16.043154 %
of primary alignments)
Average Number of Unique Start Positions per Reference
Position: 0.001630
```

Unique Start Position reports have the following naming patterns:

- *BAMfilename*-summary.txt

- *Samplename*-summary.txt

Coverage files

Example contents for the file `coverage_chr1_positive.wig` are:

```
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1,
alignmentFilteringMode=PRIMARY, filterOrpha-nedMates=false,
track name="BAM Coverage positive strand" description="BAM
Coverage positive strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
336171
336181
336191
336201
336211
336221
336231
336241
336251
336261
336271
…
```

Example contents for the file `coverage_chr1_negative.wig` are:

```
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1,
alignmentFilteringMode=PRIMARY, filterOrpha-nedMates=false,
track name="BAM Coverage negative strand" description="BAM
Coverage negative strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
57432
57442
57452
57463
57473
57483
57493
57503
```

```
57513
57523
57533
…
```

Position error file

The following is a truncated example of position error file contents:

```
# Generated by: BAM Stats
Plugin(20110415_F12_KF_FC1_1x50frag_AN_02-1-1.bam-t f3,r3  -o /
data/whist/Lifescope_Results/projects/cafe/pHRECC88020/a1/
outputs/bamstats -r /share/lifescope/Reference_Data/2.0/
lifetech/hg18/reference/human_hg18.fa ) (Base space)
# Version: 1.0
# Date: May 2, 2011
#? Total read positions = 201701987
#? Missing base calls = 34330
#? Unresolved reference positions = 4
##position_[refCall][readCall]  nErrors  nReadOccurrences  Erro
r frequency
1_AC   93033  1272759    0.0044
1_AG   25231  1255139    0.0059
1_AT   184621 37994592   0.0049
1_CA   66929  37196998   0.0018
```

Probe error files

The following is an example of probe error file contents:

```
# Generated by: CountErrorsByPrefix(-t,F3,-d,-1,-k,6,--
mask1=01111,--maskn=11111,--tmfile=/share/apps/corona-r33455M/
etc/analysis/6mers.Tm,/data/results/RegressionDriver/
CaseManager/results/latestBuild/2011-02-19_101137_r3
3455M3.5-trunk_sanityTest/mutationST/case0009/
F3_alignment.v2.gff)
# Version: 1.0.29869
# Date: 02/19/2011 06:41 PM
##ref context   num errors      num occurences  err frequency
Tm
AAAAAA  0       16      0.000000        -24.04
CAAAAA  0       6       0.000000        -24.53
GAAAAA  0       18      0.000000        -25.87
TAAAAA  0       0       0.000000        -37.81
ACAAAA  0       0       0.000000        -24.77
```

Run BAMStats standalone

This section describes requirements to generate mapping statistics outside of the context of the mapping module. These requirements are:

- **Directory structure** – The input directory for BAMStats module must mimic the output directory structure of mapping:

```
Input.BAMStat.dir/
    <sampleName1>/*.bam, *.bai
    <sampleName2>/*.bam, *.bai
```

- **Index files** – For every BAM file, there must be a corresponding BAI-formatted indexed file. BAI files can be generated using the following command:

  ```
  samtools index <bamfile>
  ```

  This command generates index sorted alignment for fast random access. The index file `<bamfile>.bai` is created. For details, please refer to the samtools site:

  http://samtools.sourceforge.net/samtools.shtml
- **Write permission** – The user must also have write permission for the input directory, because the BAMStats module generates position and probe error files in that directory.

FAQ – Mapping

1

How does the local alignment approach affect mapping?

Using the local alignment approach removes the constraint of a whole-read alignment, and mapping rate is significantly increased. It is not uncommon for a poor dataset with a 30% mapping rate at 50_6, to reach more than a 60% mapping rate with the mapping algorithm used in LifeScope™ Software.

While the majority of alignments are full length, some alignments can vary in length. If a read has many errors towards the end, the final alignment will not include these positions if a shorter alignment receives a better overall score.

2

How do I increase mapping rate?

Mapping rate increases with each additional round of mapping. However, the gain in rate comes at the cost of increased runtime and disk requirement. Testing has found sets of mapping schemes that strike a good balance between mapping rate and runtime for 25-, 35,- and 50-bp reads. These schemes are shipped as LifeScope™ Software defaults and should work well for most purposes.

For 50-bp reads, two keys in the mapping.ini file define how many rounds of mapping are run, and what happens in each round. The default values of the keys are shown below, and their meanings are further explained in Table 26 on page 150.

Note: Repetitive schemes are empty by default.

- mapping.scheme.unmapped.50 = 25.2.0,25.2.15
- mapping.scheme.repetitive.50 = 38.3.0,25.2.0

The repetitive scheme does not affect the number of mapped reads, and setting it only increases the chance of finding better alignments for repetitive reads. If your main goal is to improve throughput, first try to add more iterations at the unmapped scheme. For example, add another step of 25.2.25 at the end of the unmapped scheme above.

3

What is a good seed for my application?

Consider the following factors when picking seed parameters:

- Mapping is slower when more mismatches are allowed in the seed. However, allowing more mismatches improves the mapping rate.
- Shorter seeds have higher mapping rates, but also lead to more spurious alignments.
- Color-calls at the beginning of the read are more reliable than those at the end. Therefore, you have the option to anchor the seed near the front of the read. However, applications such as transcriptome sequencing are scenarios where it is an advantage to anchor the seed near the end of the read.

For 50-bp reads, the default setting of 25.2.0 for the first round, followed by 25.2.15 in the second round, delivers good results in most cases. For a single round, 30.3.0 is recommended.

4

What reads scenario achieves the best accuracy?

The best performance is achieved by considering 2BE and 4BE reads, their alignment, and reference. Typically, base-space reads generated by ECC have lower mapping throughput. Base-space reads generated from combined 2BE and 4BE calls and no reference have lower accuracy, compared to ECC calls.

5

How do I interpret a mapping quality value?

First of all, it is worth noting that the mapping quality value (QV) is not like the p-value in BLAST. In BLAST, the p-value is used to estimate the likelihood that the aligned sequences are related. In general, reads come from the same source and it is known that the two are related.

The purpose of mapping QV is to estimate the probability that the read originates from the mapped genomic location. The estimate is determined mainly by the difference in significance between the best- and second-best hits.

The numeric interpretation of mapping QV is the same as the base call QV. A mapping QV of 10 means that there is a 90% chance that the alignment is correct. A mapping QV of 20 means that there is a 99% chance that the alignment is correct.

6

How much RAM do I need to analyze human samples?

For optimal performance, 24 GB of RAM per cluster node is recommended for human samples. If the cluster node has less than 24 GB of available RAM, mapreads can split the genome into smaller segments, and align to each segment sequentially. Splitting the genome into smaller segments is implemented inside mapreads and is completely transparent to the user.

Mapping with less than the recommended amount of RAM slows down performance. If the cluster node has 16 GB of RAM, mapping human samples will be roughly 30% slower compared to a machine with 24 GB of RAM.

7

In the mapping statistics genome coverage calculation report, how are Ns in the reference counted?

Ns in the reference are counted as missing coverage. For example, in the human genome, about 7% of the reference sequence consists of Ns. This means that the coverage calculation will never be reported as higher than 93%. If the frequency distribution says 7.49% of the genome is uncovered, this means that about 0.34% of the non-N reference sequence does not have reads mapped to it, and that also the 7.15% of the genome that consists of N does not have reads mapped to it.

8

What should I do to ensure balanced allele ratios at heterozygous positions?

The base-translation algorithm, refcor, uses instrument color-space data (ECC or non-ECC) and the reference sequence to produce the most likely base calls, along with a Phred-scale quality value. The degree to which the reference influences the base call is controlled by the `refcor.reference.weight` parameter. A position is converted to reference only if the `refcor.reference.weight` setting is higher than the difference between the quality values of a correct color call and of an adjacent erroneous color call that supports reference.

The `refcor.reference.weight` default setting, 8, is chosen to be low enough to ensure balanced allele ratios, and also be large enough both to distinguish correct from incorrect color calls and to maximize base accuracy.

In the event an incorrect reference base call is made, the disagreement between evidence sources causes that call's QV to be low. Incorrect calls can be filtered out based on QV threshold, and this filtering improves allele ratios.

Another way of filtering low quality base-calls is to adjust the parameter `refcor.base.filter.qv`, that converts all base calls with a quality value less than `refcor.base.filter.qv` into Ns. This filtering has the effect of setting QV to zero in the calls with low confidence. To decrease the number of N base calls, reduce this threshold (for example, from the default of 10 to 5).

Note: Setting the `refcor.reference.weight` parameter to zero completely eliminates the reference as a guide for base-translation (completely eliminates reference bias), but also considerably reduces the quality of base-translation.

FAQ – Pairing

1

How is the uniqueness of a pair determined?

A pair of reads is unique when there is exactly one good AAA pair. With the advent of local alignment, the likelihood of finding only one good pair is decreased. As a result, a different heuristic is used to determine "uniqueness".

Consider an alignment of length *L* with *M* mismatches. If the local score is defined as *L+(m-1)M*, where *m<0* is the mismatch penalty, then the score of each good pair candidate is the sum of its two constituent local scores.

A pair of tags is considered unique if it has only one good pair, or if the score of the best pair is at least X greater than the score of the second best pair. X is specified by the `pair.uniqueness.threshold` key, and has a default value of 10.0. See Table 29 on page 163 for a description of this and all pairing parameters.

2

What do the 3-letter pair classifications mean?

Table 34 lists a summary of genomic code classifications for mate-pair pairing runs. See Table 35 on page 183 for a summary of genomic code classifications for paired-end runs.

Table 34 Genomic code classifications, for mate-pair pairing

Class name	Strand	Orientation	Insert size	Preference
AAA	Same	R3 to F3	Normal	1
AAB	Same	R3 to F3	< min	2
AAC	Same	R3 to F3	> max	2
ABA	Same	F3 to R3	Normal	2
ABB	Same	F3 to R3	< min	3
ABC	Same	F3 to R3	> max	3
BAA	Different	Outward	Normal	3
BAB	Different	Outward	< min	4
BAC	Different	Outward	> max	4
BBA	Different	Inward	Normal	3
BBB	Different	Inward	< min	4
BBC	Different	Inward	> max	4

The first letter determines whether the two tags match the same strand:

- **A** – Match the expected strand.
- **B** – Match different strands.

The third letter determines if the distance between two hit positions is within the correct range:

- **A** – Within the correct range.
- **B** – The distance is less than the insert size measurement minimum (set with `default.minimum.insert.size` and denoted by "< min" in the table above).
- **C** – The distance is greater than the insert size measurement maximum (set with `default.maximum.insert.size` and denoted by "> max" in the table above).

The default minimum insert size is 0. For paired-end and long mate pair workflows, the default maximum insert sizes are 2000 and 20,000, respectively. An insert size measurement automatically determines the actual values using the defaults as the largest interval, and the defaults are used if the insert size measurement fails.

The middle letter has different meanings, depending on whether the first letter is A or B. If the first letter is A, the second letter indicates if the order of the two tags is correct:

- **A** – The order is correct, and R3 is upstream of F3.
- **B** – The order is incorrect, and F3 is upstream of R3.

When the two tags are on different strands, the second letter indicates whether the two tags are pointing toward or away from each other

- **A** – Pointing away from each other.
- **B** – Pointing toward each other.

In addition, C** classification is assigned to a pair of reads that have one hit each, but are on different chromosomes. If both F3 and R3 tags are present, but only one has a unique hit while the other one is unmapped, the pair is labeled D**. Finally, if one tag from a pair has a unique hit but the other one is missing, as opposed to unmapped, then the pair is classified as E**.

The Preference value indicates likelihood of the variation can occur in nature. Larger values indicate the variation is less likely to occur in nature.

Table 35 shows the classifications for paired-end runs. Summary statistics of these categories are calculated in the optional BAMStats Read Pair Type Statistics Report (`.ReadPair.Type.cht`).

Table 35 Genomic code classifications, for paired-end pairing

Class name	Strand	Orientation	Insert size	Preference
AAA	Different	Inward	Normal	1
AAB	Different	Inward	< min	2
AAC	Different	Inward	> max	2
ABA	Different	Outward	Normal	2
ABB	Different	Outward	< min	3
ABC	Different	Outward	> max	3
BAA	Same	F3 to F5	Normal	3
BAB	Same	F3 to F5	< min	4
BAC	Same	F3 to F5	> max	4
BBA	Same	F5 to F3	Normal	3
BBB	Same	F5 to F3	< min	4

Class name	Strand	Orientation	Insert size	Preference
BBC	Same	F5 to F3	> max	4

3

Why does LifeScope™ Software report more duplicate beads than Picard reports?

LifeScope™ Software reports significantly more duplicates than does Picard. There are a number of underlying reasons why our mark duplicates module is more aggressive in marking duplicates than Picard:

- By design, LifeScope™ Software marks reads with the same read start position, even if they have soft or hard clipping at the start of the read, as duplicates. We consider this to be desirable behavior.
- There are also some differences in the handling of reads mapped to different chromosomes, and reads that are not "proper pairs" (because of orientation, strand, or distance).
- In the case of reads from multiple BAM files and with non-unique bead IDs across multiple files, LifeScope™ Software marks reads with the same bead ID as duplicates.
- LifeScope™ Software by default marks duplicates only within a single BAM file, with an option to mark duplicates across multiple BAM files. The multiple BAM option is appropriate for multiple barcodes from the same sample run on the same lane. For these runs, our mark duplicates module behaves similarly to Picard and marks around the same number of duplicates as Picard. The duplicates detected are not exactly the same, because the algorithms differ slightly.

FAQ – BAMStats

1

In the BAMStats summary table, why do the `NumFragmentsPassingFilters` and `NumFragmentsTotal` fields have the same totals?

`NumFragmentsTotal` reports the total number of fragments (read pairs or reads) passed from the instrument. `NumFragmentsPassingFilters` reports the number of fragments that pass the instrument filter. For the 5500 instrument, all reads pass filters when using color space, and so these two numbers are the same. If the run is performed with ECC, then the base space reads have filters applied to remove those of lowest quality. For an ECC run mapped in base space using LifeScope™ Software, the filtered reads are reported in LifeScope™ Software, and the Total and Filtered numbers are different.

2

What changes are required for a job with a large number of contigs?

For a job with a large (more than a few thousand) number of contigs, change the system property `max.head.node.java.heap.space` from the default of 64 to 128 or 256.

Modify this parameter in the <*installdir*>`/etc/analysis/system.properties` file.

Because changes to system properties affect all runs of all users, you could consider changing the parameter back to its original value after the large contig run completes. Setting `max.head.node.java.heap.space` to 128 or 256 could present a problem if your head node is extremely loaded.

10 Run a SNPs Analysis

This chapter covers:

- Overview 187
- SNPs input files. 188
- Examples of running a SNPs analysis 190
- SNPs runtime parameters 191
- SNPs output files 198
- SNPs algorithm description 206
- FAQ – SNPs 209

Overview

The LifeScope™ Genomic Analysis Software SNPs analysis module uses the diBayes algorithm to find Single Nucleotide Polymorphisms (SNPs). The diBayes package performs independent SNP analysis at each position in the reference, using either a Bayesian or Frequentist algorithm.

The SNPs module is used to call Single Nucleotide Polymorphisms (SNPs) from mapped and processed SOLiD® System reads. The module takes the reads, quality values, the reference sequence, and error information of each SOLiD® System slide as its input, and calls SNPs.

The SNPs module takes the mapped and processed SOLiD® System reads, quality values, the reference sequence, and error information of each SOLiD® System slide as its input, and calls SNPs.

The SNPs module creates these results files:

- A list of SNPs.
- A quartile file with coverage as well as color and base quality value distribution.
- (*Optional*) A consensus FASTA file with the same number of bases as the reference sequence.
- (*Optional*) A consensus calls file with a list of all covered positions.
- (*Optional*) A collection of annotated files.

The consensus calls file and the quartile file are each generated as one file per contig. The list of SNPs and the consensus FASTA file are each generated both as one file per contig and also as a consolidated file for the entire run. Annotated files are generated for the entire run, not for individual contigs. See "SNPs output files" on page 198.

SNPs input files

The diBayes input files include the reference file, the regions of interest file (BED), the BAM file (the output of the mapping-pairing or enrichment modules), and position error and probe error files (the output of the BAMStats mapping statistics module). When you run a standard workflow that includes a SNP analysis, the wiring of these input files is handled automatically by the LifeScope™ Software command shell. These input files are described in Table 36.

Table 36 diBayes input file description

Input File	Description
BAM file input	The BAM input files are generated by LifeScope™ Software mapping or enrichment modules. The SNPs module accepts one or more BAM files as input. Each BAM file must have only one read group and only one library type. The SNPs module does not support merged BAM files with multiple read groups or with different library types (within one BAM file). For reads containing Ns, the SNPs module uses all information in the reads except for the missing bases. The SNPs module accepts both color space BAM files (from regular two base color-encoding runs) and base space BAM files (from ECC runs), but does *not* support combining both color space and base space BAM files in one run. The module determines what algorithms to use automatically based on the input files. Multiple base space BAM files can be processed as combined input using base space algorithms. Multiple color space BAM files are processed as combined input using color space algorithms. If color space BAM files are missing color space information (for instance when users choose not to output color information into the BAM files.), then base space algorithms are used. **IMPORTANT!** Do not concatenate BAM files from different run types. **IMPORTANT!** Do not merge the BAM files before running the SNPs module. The module can accept multiple BAM files of different run types as its input. **Note:** If you move the input BAM file from its default location (not recommended), you must also move its corresponding position and probe error files to the same directory (the BAM file and its corresponding position and probe error files have the same file name prefix).
Reference genome	The sequence to which the reads are aligned and mapped. The recommended format for the reference file is the FASTA format, for example, chr20.validated.fasta. **Note:** When you run the SNPs module, you must use the same reference that was used for mapping. If different reference files are used, during the SNPs module analysis the coordinates in the BAM file might not correctly represent the coordinates in the reference file. The SNPs module might reject the input BAM files if the reference information in headers of BAM files is not matched to the reference provided. The reference sequence might have multiple chromosomes or contigs, and might contain IUB codes at positions of known SNPs.

Table 36 diBayes input file description *(continued)*

Input File	Description
Regions of interest file (BED file)	The regions of interest file, in BED format, contains a list of defined regions. This file is required input file for target resequencing workflows and optional for whole genome resequencing workflows. If users want to use a BED file in whole genome resequencing workflows, the file must be sorted based on the start positions for defined regions. The BED files for target resequencing workflows do not have this requirement. Instead the enrichment module within the target resequencing workflow sorts the BED file and passes the sorted BED file to the SNPs module. We recommend users to choose target resequencing workflows for analyses that include a BED file.
F3 (R3/F5) position error file	F3 (R3/F5) position error files are tab-delimited text files created during secondary analysis. The files record the frequencies of color/base mismatches between reads and the reference at different positions in a read. For fragment runs, only an F3 position error file is needed. Both F3 and R3(F5) position error files are created for mate-pair (paired-end) runs, and both are needed to run diBayes. Position error files are automatically generated during mapping statistics if BAMStats is enabled (recommended). If position error files are not available, the SNPs module uses internal default position errors for SNP calling. Using default position error files might slightly affect the module's accuracy. Position error files are located in the mapping results folder, and share the same name prefix as the output BAM files. You do not specify their names or location when you run SNPs in a standard workflow. If you run SNPs outside of a workflow and also move the mapping output BAM files, always copy the position and probe error files to the same directory as the BAM files.
F3 (R3/F5) probe error file	F3 (R3/F5) probe error files are tab-delimited text files that record the frequencies of dicolor mismatches between the reads and the reference as a function of different 6-mer probes. LifeScope™ Software calculates the probe error files. For fragment runs, only F3 probe error files are needed. Both F3 and R3(F5) probe error files are created for mate-pair (paired-end) runs, and both are needed to run diBayes on paired data. Different SOLiD® System runs of the same sample might generate different F3 (R3/F5) probe error files for each run, because of the random nature of probe errors. Probe error files are automatically generated during mapping statistics if BAMStats is enabled (recommended). However, probe error files are not generated for base space data. If the probe error file is not found, the SNPs module uses internal default probe error files for SNP calling. Using default probe error files might slightly affect the module's accuracy. Probe error files are located in the mapping results folder, and share the same name prefix as the output BAM files. You do not need to specify their name or location when you run SNPs in a standard workflow. If you run SNPs outside of a workflow and also move the mapping output BAM files, always copy the position and probe error files to the same directory as the BAM files.

Examples of running a SNPs analysis

Examples in a standard workflow

The following standard workflows provide examples of running the SNPs module:

- genomic.resequencing.frag
- genomic.resequencing.lmp
- genomic.resequencing.pe
- targeted.resequencing.frag
- targeted.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# create a project and open it
mk ecoli
cd ecoli
# create an analysis and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.pe
# define the input data
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
# specify the reference
set reference hg19
# optionally change SNPs defaults after this line
# list the analysis configuration
ls
# start the analysis
run
# list progress information for the run
ls
```

Dummy file names for XSQ files are shown here.

See for Chapter 7, “Run a Standard Workflow Analysis” on page 111 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 83 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an example of how to run the SNPs module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, “Demo Analyses” on page 567 for information on running the SNPs module as a standalone analysis.

SNPs runtime parameters

Table 38 describes the SNPs module runtime parameters.

In order to change a parameter value in your analysis, use the `set param` shell command to replace the line `# optionally change SNPs defaults after this line` in the example commands in "Examples in a standard workflow". For instance, to change the call stringency, use this shell command:

```
set dibayes.call.stringency high tertiary/dibayes.ini
```

These workflows use the INI file name `dibayes.ini`:

- `genomic.resequencing.lmp`
- `genomic.resequencing.pe`
- `targeted.resequencing.pe`

For the workflow `targeted.resequencing.frag`, use `dibayes.targeted.frag.ini`. For the workflow `genomic.resequencing.frag`, use `dibayes.genome.ini`.

Certain parameters affect only the module's base-space or color-space algorithm. These parameters are noted in Table 38. Changes to a base-space parameter's setting have no effect if the module runs its color-space algorithm. Similarly, changes to a color-space parameter's setting have no effect if the module runs its base-space algorithm.

Table 37 describes a workflow parameter, `analysis.space`, that is used in the SNPs module. The `analysis.space` parameter is set in the `tertiary/global.ini` file.

Table 37 Analysis space parameter used in the SNPs module

Parameter name	Default	Description
analysis.space	auto	Specifies your *preference* for base space or color space analysis. Allowed values: • **auto**: The analysis space is determined based on the mapping space and the presence of color space data in the analysis BAM files. • **base**: Prefer base space analysis. • **color**: Prefer color space analysis. See "Analysis space" on page 77 for a description of other factors determining the analysis space used for SNPs.

Table 38 describes SNPs parameters.

Table 38 SNP (diBayes) parameter description

Parameter name	Default	Description
Optional general parameters		
dibayes.reads.min.mapping.qv	8	Requires that the mapping quality value of the read be higher than this minimum mapping/pairing QV. Allowed values: Integers 0–100. See "The reads mapping QV parameter" on page 197.
dibayes.detect.2.adjacent.snps	0	Whether or not to detect two adjacent SNPs. Allowed values: • **0**: Do not call a second adjacent SNP as a SNP. • **1**: If two adjacent SNPs are found, call both as SNPs.

Table 38 SNP (diBayes) parameter description *(continued)*

Parameter name	Default	Description
dibayes.polymorphism.rate	0.001	The expected frequency of heterozygotes in the population, for example, 0.001 in humans. Allowed values: Float 0.0–1.0.
dibayes.het.skip.high.coverage	1	Do not call SNPs when the coverage of position is too high compared to the median of the coverage distribution of all positions. Allowed values: • **0**: Call a position a SNP under these conditions. • **1**: Do not call a position a SNP under these conditions. **Note:** Enable this filter for whole genome resequencing application. Disable it for whole transcriptome or target resequencing. See "The skip high coverage filter" on page 197.
Required parameter		
dibayes.call.stringency	medium	Specifies the SNPs call stringency. Each call stringency setting specifies a set of filters. An individual filter parameter setting overrides the call stringency setting, in case of conflict. Allowed values: • **highest**: Recommend when a very low false positive tolerance is allowed. • **high**: Requires the allele on both strands. • **medium**: No both-strand requirement. • **low**: Very aggressive. • **lowest**: Even more aggressive. Very minimal filtering. • **low_frequency**: For use with the low frequency variant workflows. Lower stringency settings result in more SNP calls, but more false positives are also expected. Higher settings result in fewer SNP calls, with fewer false positives. See "The call stringency parameter" on page 196.
Optional read filters The entire read is excluded from analysis if a read fails to pass one of these filters.		
dibayes.reads.with.unmapped.mate.include	1 for PE data; 0 for other data	Whether or not to include reads that only have one tag mapped (their mate tags are either unmapped or missing). Allowed values: • **0**: Do not include these reads. • **1**: Include these reads.
dibayes.reads.with.indel.exclude	1	Whether or not to exclude reads that have indels. Allowed values: • **0**: Do not exclude reads that have indels. • **1**: Exclude reads that have indels.
dibayes.reads.only.uniquely.mapped.allow	0	Whether or not to require that reads be uniquely mapped. A very stringent filter. Allowed values: • **0**: Do not require that reads be uniquely mapped. • **1**: Require uniquely mapped reads. Filter out reads that are not uniquely mapped.
dibayes.reads.max.mismatch.alignlength.ratio	1.0	The threshold of mismatch/alignment-length ratio. The reads whose mismatch/alignment length ratio is higher than this specified threshold are filtered (are ignored). Allowed values: Float 0.0–1.0.

Table 38 SNP (diBayes) parameter description *(continued)*

Parameter name	Default	Description
dibayes.reads.min.alignlength.readlength.ratio	0.0	The threshold of alignment-length/read-length ratio. The reads whose alignment-length/read-length ratio is less than this specified threshold are filtered (are ignored). Allowed values: Float 0.0–1.0.
Optional general position filters The set of pile-up color/base calls at a given position is excluded from both heterozygous and homozygous SNP analysis if the position pile-up fails to pass one of these filters.		
dibayes.snps.min.filteredreads.rawreads.ratio	0.0	The minimum ratio of the filtered reads and raw reads. A position whose ratio of filtered reads to raw reads is below this value is not considered. Allowed values: Float 0.0–1.0.
dibayes.snps.min.color.qv	7	Require that the candidate allele have at least this base quality value. A color call that fails this filter is excluded from the analysis, but the color pile-up is still considered for SNP calling. Allowed values: Integers 0–62. Color space only.
dibayes.snps.min.base.qv	26	Require that the candidate allele have at least this base quality value. A base call that fails this filter is excluded from the analysis, but the base pile-up is still considered for SNP calling. Allowed values: Integers 0–41. Base space only. See "The minimum base QV filters" on page 197.
dibayes.snps.min.nonref.base.qv	26	Require that the non-reference allele have at least this base quality value. A base call that fails this filter is excluded from the analysis, but the base pile-up is still considered for SNP calling. Allowed values: Integers 0–41. Base space only. See "The minimum base QV filters" on page 197.
dibayes.snp.both.strands	0	Whether or not to require that the novel allele is present on both strands and statistically similar represented on both strand for both heterozygous and homozygous SNPs. Allowed values: • **0**: Do not make the above requirement. • **1**: Make the above requirement.
Optional heterozygous position filters The set of pile-up color/base calls at a given position is excluded from only heterozygous SNP analysis if the position pile-up fails to pass one of these filters.		
dibayes.het.min.allele.ratio	0.15	The less-common allele must be at least this proportion of the reads of the first two most common valid alleles. Allowed values: Float 0.0–0.5.
dibayes.het.min.coverage	2	Require at least this coverage to call a heterozygous SNP. Allowed values: Integers >= 0.
dibayes.het.min.start.pos	2	The minimum number of unique start positions required to call a heterozygote. Allowed values: Integers >= 0.
dibayes.het.min.nonref.base.qv	26	Requires the non-reference allele to have at least this average base quality value to call a heterozygous SNP. Allowed values: Integers 0–41. Base space only.

Table 38 SNP (diBayes) parameter description *(continued)*

Parameter name	Default	Description
dibayes.het.min.nonref.color.qv	7	Requires the non-reference allele to have at least this average color quality value to call a heterozygous SNP. When you know whether your input reads contain ECC data (from ICS 1.1 or higher) or not, use one of the following recommended values: • **Non-ECC data**: 7 • **ECC data (from ICS 1.1 or higher)**: 20 Allowed values: Integers 0–62. Color space only.
dibayes.het.min.validreads.totalreads.ratio	0.65	The proportion of the total reads containing either of the two candidate alleles. Filters positions with high raw-error rates. Allowed values: Float 0.0–1.0.
dibayes.het.min.adjacentsnp.allele.count	2	Minimum valid tricolor counts (Het). Allowed values: Integers >= 0. Color space only.
Optional homozygous position filters The set of pile-up color/base calls at a given position is excluded from only homozygous SNP analysis if the position pile-up fails to pass one of these filters.		
dibayes.hom·min.coverage	1	Requires at least this coverage to call a homozygous SNP. Allowed values: Integers >= 0.
dibayes.hom.min.nonref.allele.count	2	Requires at least this non-reference allele count to be called a homozygous SNP. Allowed values: Integers >= 0.
dibayes.hom.min.nonref.color.qv	7	Requires the non-reference allele to have at least this average color quality value to call a homozygous SNP. When you know whether your input reads contain ECC data (from ICS 1.1 or higher) or not, use one of the following recommended values: • **Non-ECC data**: 7 • **ECC data (from ICS 1.1 or higher)**: 8 Allowed values: Integers 0–62. Color space only.
dibayes.hom.min.nonref.base.qv	26	Requires the non-reference allele to have at least this average base quality value to call a homozygous SNP. Allowed values: Integers 0–41. Base space only.
dibayes.hom.min.nonref.start.pos	2	Requires the non-reference allele to have at least this minimum number of unique start positions, to call a homozygote. Allowed values: Integers >= 0.
Optional low frequency variant filters For use with the low frequency variant (LFV) workflows. When dibayes.call.stringency is set to low_frequency, the filters in this section are set to their defaults. Other stringency settings effectively render these filters inactive.		
dibayes.het.lca.both.strands	0	The less common allele on both strands. Allowed values: 0,1. Recommended value for LFV workflows: 1.
dibayes.het.min.lca.start.pos	0	The minimum unique start position of the less common allele. Allowed values: Integers >= 0. Recommended value for LFV workflows: 7.
dibayes.het.min.lca.color.qv	0	The minimum average color QV of the less common allele. Allowed values: Integers >= 0. Recommended value for LFV workflows: 30.

Table 38 SNP (diBayes) parameter description *(continued)*

Parameter name	Default	Description
dibayes.het.max.diff.color.qv	99	The maximum difference of color QVs of the most common and less common alleles. Allowed values: Integers -100–100. Recommended value for LFV workflows: 10.
Output file processing		
dibayes.write.fasta	1	Whether or not to create FASTA files. Allowed values: • **0**: Do not create the output FASTA files. • **1**: Create the individual and combined output FASTA files of all contigs.
dibayes.write.consensus	1	Whether or not to create individual consensus calls files of each contig. Allowed values: • **0**: Do not create individual consensus calls files of each contig. • **1**: Create individual consensus calls files of each contig. Neither setting creates a combined consensus calls file of all contigs.
dibayes.compress.consensus	0	Whether or not to zip the generated consensus files. Allowed values: • **0**: Do not compress the consensus files. • **1**: Compress the individual consensus file of each contig (within the individual contig folders).
Resource parameters		
memory.request	7gb	Memory request for process. Include the units gb in the setting.
java.heap.space	14000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.
maximum.workers	24	Maximum number of nodes that can be used for this analysis.

The call stringency parameter

The `dibayes.call.stringency` parameter controls the SNP calling results. Each setting represents a different combination of filters. Table 39 explains the call stringency parameter settings. The setting `low_frequency` (not shown in the table) is for use only with low frequency variant workflows.

Table 39 Empirical dibayes.call.stringency settings for diBayes

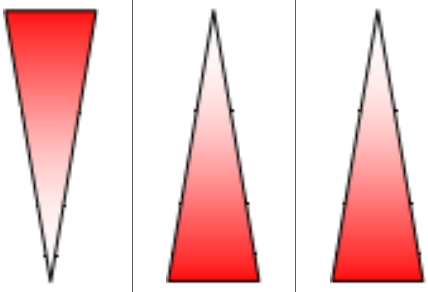
Stringency setting	Stringency	SNPs	False positives	Recommended coverage	Comments
highest				>80x	Recommend when very low false positive tolerance is allowed.
high				20x ~ 80x	Requires the allele on both strands.
medium				1x ~ 25x	No both-strand requirement.
low				—	Very aggressive.
lowest				—	Very aggressive. Minimal filtering.

Table 40 shows the non-default filter parameter values for each level of stringency. The medium stringency level is not listed because it calls for default values of the filtering parameters. If you explicitly set one of the listed filter parameters, the value that you specify overrides the value called for by the call stringency level.

Table 40 Filter parameters affected by call stringency level

Call stringency setting	Filter parameter	Parameter setting change	
		Default value	New value
highest	dibayes.snp.both.strands	0	1
	dibayes.het.min.start.pos	2	3
	dibayes.het.min.validreads.totalreads.ratio	0.65	0.70
	dibayes.het.min.adjacentsnp.allele.count	2	3
	dibayes.hom.min.nonref.start.pos	2	3
high	dibayes.snp.both.strands	0	1
low	dibayes.het.min.start.pos	2	1
	dibayes.hom.min.nonref.start.pos	2	1
lowest	dibayes.het.min.adjacentsnp.allele.count	2	1
	dibayes.het.min.nonref.color.qv	7	0
	dibayes.het.min.start.pos	2	1
	dibayes.het.min.validreads.totalreads.ratio	0.65	0.0
	dibayes.hom.min.nonref.allele.count	2	1
	dibayes.hom.min.nonref.base.qv	26	0
	dibayes.hom.min.nonref.color.qv	7	0
	dibayes.snps.min.nonref.base.qv	26	0
	dibayes.reads.min.mapping.qv	8	2

Table 40 Filter parameters affected by call stringency level *(continued)*

Call stringency setting	Filter parameter	Parameter setting change	
		Default value	New value
low_frequency	The low_frequency stringency in effect sets some filter parameters that are not related to low frequency variants to somewhat extreme levels, so that those parameters do not filter reads out (during normal operation of low frequency variant analyses).		
	dibayes.reads.min.mapping.qv	8	16
	dibayes.het.min.allele.ratio	0.15	0.01
	dibayes.reads.min.alignlength.readlength.ratio	0.0	0.9
	dibayes.hom.min.nonref.base.qv	28	0
	dibayes.het.skip.high.coverage	1	0

The skip high coverage filter

Enable the `dibayes.het.skip.high.coverage` filter to skip false-positive SNP positions with high coverage. If you enable this filter, the SNP position is skipped if the coverage of a position is too high compared to the median of the coverage distribution of all positions. Enable the `dibayes.het.skip.high.coverage` filter for whole genome resequencing. Disable the filter for transcriptome or targeted resequencing. The filter is enabled by default.

The reads mapping QV parameter

The parameter `dibayes.reads.min.mapping.qv` is a read-level filter. Reads with mapping QVs lower than this value are filtered out. The empirical default of the mapping QV is 8, which gives a balanced performance between sensitivity (more SNPs to be called) and specificity (fewer false positive SNPs) on multiple known data sets.

Raising the mapping QV reduces the number of false positive SNP calls. However, because more reads are filtered out, the total number of SNP calls decreases also.

The minimum base QV filters

The `dibayes.snps.min.base.qv` and `dibayes.snps.min.nonref.base.qv` parameters work like a read filter at the position level. These parameters are very effective in removing bad base calls for SNP analysis. Increasing the value of these parameters causes only high-quality base calls to be considered, and results in higher SNP calling accuracy with a slight sacrifice of the total number of SNPs. When data coverage is high, we recommend using a higher value for higher accuracy.

Lowering the value of the `dibayes.snps.min.base.qv` and `dibayes.snps.min.nonref.base.qv` parameters results in more SNP calls.

Other filtering parameters

These other optional diBayes filters provide additional freedom for different datasets and applications. All filtering is applied before a position is considered by the module's algorithms.

Table 44 on page 202 also describes how the optional tuning parameters are used in filtering reads.

SNPs internal parameters

Table 41 describes SNPs internal parameters that we do not recommend changing.

Table 41 SNPs internal parameter description

dibayes.run	1	Enables the SNPs module. Allowed values: • **0**: Do not run a SNPs analysis. • **1**: Run the SNPs module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.

SNPs output files

There are multiple output subfolders (for example, `chr_1`, `chr_2`, ... `chr_n`), corresponding to individual chromosomes or chromosome/contigs. Each subfolder contains up to 4 files:

- <*prefix*>`_SNP.gff`
- <*prefix*>`_quartiles.txt`
- (*Optional*) <*prefix*>`_Consensus_Calls.txt`
- (*Optional*) <*prefix*>`_Consensus_Basespace.fasta`

The remainder of this section describes these files and provides examples.

In the root of output folder, the individual GFF and FASTA file in the chromosome subfolders are concatenated into the final GFF and FASTA file for the whole genome. Because of the large size of individual consensus calls files, we do not consolidate them into a summarized copy to save space.

The output layout is listed below. The first two files are consolidated data for the entire run. The files in contig subfolders are data for an individual contig.

```
results_directory/
    <prefix>_SNP.gff3
    <prefix>_Consensus_Basespace.fasta.
    contig1/
        <prefix>_SNP.gff3
        <prefix>_Consensus_Calls.txt
        <prefix>_Consensus_Basespace.fasta
        <prefix>_quartiles.txt
    contig2/
        <prefix>_SNP.gff3
        <prefix>_Consensus_Calls.txt
        <prefix>_Consensus_Basespace.fasta
        <prefix>_quartiles.txt
    ...
```

GFF file format

The <*prefix*>`_SNP.gff` file is the list of output SNPs. This file is described in Table 42.

Table 42 <diBayes.output.prefix>.gff file format description

Column name	Description	Example
##	Header comment lines.	Input files and algorithm parameters
#	Header of the results.	—
seqid	The string ID of the sequence to which the start and end coordinates refer.	chr1
source	The source of the data.	SOLiD_diBayes
type	Sequence ontology derived type for this variation. For diBayes, this is always SNP.	SNP
start	Start position of the SNP.	420
end	End position of the SNP.	420
score	Calculated p-value of the SNP.	0.000000
strand	—	—
phase	—	—
Attributes:		
• reference	The base of the reference sequence at the current position.	reference=c
• allele-call	Genotype in alleleA/alleleB (in alphabet order) format for bases observed in all valid reads that cover the position.	allele-call=A/C
• coverage	The number of valid reads that cover the current position.	coverage=52
• rawCoverage	The number of raw reads that cover the current position.	rawCoverage=52
• refAlleleCounts	The number of reads of the reference allele at the current position.	refAlleleCounts=22
• refAlleleStarts	The number of different start positions of reads having the reference allele at the current position.	refAlleleStarts=15
• refAlleleMeanQV	The mean of quality values of all reference allele reads at the current position.	refAlleleMeanQV=15
• novelAlleleCount	The number of reads of the most abundant non-reference allele at the current position.	novelAlleleCounts=24
• novelAlleleStarts	The number of different start positions of reads having the most abundant non-reference allele at the current position.	novelAlleleStarts=14
• novelAlleleMeanQV	The mean of quality values of all novel allele reads.	novelAlleleMeanQV=17
• diColor1	The most abundant allele in the reads (not necessarily the reference allele) in dicolor encoding (for example, one of 00, 01, ... 32, 33 of 16 possible dicolors). Does not apply to the results of base-space BAM files.	diColor1=00
• diColor2	The second-most abundant allele in the reads. Does not apply to the results of base-space BAM files.	diColor2=22

Table 42 <diBayes.output.prefix>.gff file format description *(continued)*

Column name	Description	Example
• zygosity	Heterozygosity flag. Values are: **homozygous**: Homozygous SNP **heterozygous**: Heterozygous SNP	zygosity=heterozygous
• flag	Reports why the SNP is not called as a homozygous SNP or as a heterozygous SNP, when information is available. Values are: • **m***: Flags beginning with the letter "m" report why the SNP is not called as a Hom. • **t***: Flags beginning with the letter "t" report why the SNP is not called as a Het. • **—**: No information. The SNP can be either a Het or a Hom. The m* and t* flag values are listed in Table 44 on page 202. A position that is not called as a heterozygous SNP (due to failing current filter settings) may still be called as a homozygous SNP. In this case, the flag reports the reason for not calling the position a heterozygous SNP. Similarly, a position that is not called as a homozygous SNP (due to failing current filter settings) may still be called as a heterozygous SNP. In this case, the flag reports the reason for not calling the position a homozygous SNP. See also the consensus calls Flag field, which reports the reason why a position is not called a SNP, and adds which filter causes this result.	flag=t4,t10

GFF file example

The following is example content of the GFF file generated by the SNPs module.

```
##gff-version 3
##source-version LifeScope(tm) 2.0 diBayes
##feature-ontology http://song.cvs.sourceforge.net/*checkout*/
song/ontology/sofa.obo?revision=1.141
##type DNA
##date 2011-03-15
##time 15:56:41
##genome-build /local/xux1/validataion/HuRef/hg18/reference/
chr22.validated.fasta
##input-files /local/xux1/validataion/HuRef/hg18/input/
solid0107_20100514_PE_launch_verification_F3.sowmi-1_0309.bam
##options call-stringency: medium; dibayes-base-space: 0;
experiment-name:
Test_huref_4.0_base_newbam0309_1of3_color2.0_newout_chr22; het-
skip-high-coverage: 1; index: 1; log-dir: /local/xux1/
validataion/HuRef/hg18/log/newbam0309_1of3_color2.0; output-
dir: /local/xux1/validataion/HuRef/hg18/outputs/
```

```
newbam0309_1of3_color2.0; poly-rate: 0.003; reference: /local/
xux1/validataion/HuRef/hg18/reference/chr22.validated.fasta;
working-dir: /local/xux1/validataion/HuRef/hg18/tmp/
newbam0309_1of3_color2.0;
#ChrSourceTypePos_StartPos_EndScoreStrandPhaseAttributes
chr22SOLiD_diBayesSNP14432513144325130.000984..
reference=G;allele-call=C/
G;coverage=6;rawCoverage=9;refAlleleCounts=3;refAlleleStarts=3;
refAlleleMeanQV=22;novelAlleleCounts=3;novelAlleleStarts=3;nove
lAlleleMeanQV=29;diColor1=01;diColor2=32;zygosity=heterozygous;
flag=-
chr22SOLiD_diBayesSNP14433730144337300.002050..
reference=C;allele-call=A/
C;coverage=8;rawCoverage=8;refAlleleCounts=5;refAlleleStarts=4;
refAlleleMeanQV=20;novelAlleleCounts=3;novelAlleleStarts=3;nove
lAlleleMeanQV=26;diColor1=20;diColor2=31;zygosity=heterozygous;
flag=-##
```

Combined GFF for all contigs

The combined GFF file is a consolidated version of individual GFF files.

This file is named according to the following pattern:

```
${task.output.dir}/${analysis.name}_ SNP.gff3
```

The following is an example of the summarized header of the combined GFF file:

```
##gff-version 3
##List of SNPs. Date Thu Oct  7 14:32:53 2010    Stringency:
medium Mate Pair: 0 Read Length: 50 Polymorphism Rate: 0.001000
Bayes Coverage: 60 Bayes_Single_SNP: 1 Filter_Single_S
NP: 1 Quick_P_Threshold: 0.997000 Bayes_P_Threshold: 0.040000
Minimum_Allele_Ratio: 0.150000
Minimum_Allele_Ratio_Multiple_of_Dicolor_Error: 100
##Data source:    Experiment: test_case0003
##chr1  1// chr${index} chromosome name
##chr2  2// chr${index} chromosome name
##chr3  3// chr${index} chromosome name
```

The portion in bold is the chromosome index and name table, which combine the information from individual GFF files.

Consensus_Calls file format

The <*prefix*>_Consensus_Calls.txt is a tab-separated file that covers all positions that have coverage and provides general information about each position (see Table 43). Its flag column shows a list of codes of filters (see Table 44 on page 202) that the position fails to pass to be called as a SNP. It is very useful for identifying the reason of false negative SNP calls (the known SNPs that are not called by diBayes). By adjusting the value of keys listed in Table 44, user can loosen the filter stringency and make the positions that originally fail the filters to be considered in the SNP analysis.

Consensus calls files are created with file names according to the following pattern:

```
${task.output.dir}/contig${index}/${analysis.name}_ Consensus_Calls.txt
```

Table 43 <diBayes.output.prefix>_Consensus_Calls.txt file format description

File Name/Column	Description	Example
Chr	Chromosome/Contig number.	chr1
Position	Location of the SNP on the reference sequence.	442
Allele_DiColor1	The most abundant allele in the reads (not necessarily the reference allele) in dicolor encoding (for example 00, 01....32, 33) of 16 possible dicolors. Color space only.	03
Allele_DiColor2	The second most abundant allele in the reads in dicolor encoding (for example 00, 01....32, 33) of 16 possible dicolors. Color space only.	03
Reference	The base of the reference sequence at the current position.	C
Genotype	Genotype in the form of IUB codes for bases observed in all the reads.	C
P-value	Calculated p-value of the SNP. (P-values are generated from the Bayesian and Frequentist algorithms.)	1.00000
Coverage	The number of the reads that cover the current position.	2
Unfiltered Coverage	The number of all unfiltered reads that cover the current position.	2
nCounts_1st_allele	The number of the most abundant allele at the current position.	2
nCounts_Reference_allele	The number of reads having the reference allele at the current position.	2
nCounts_NonReference_allele	The number of reads having the most abundant non-reference allele reads at the current position.	0
Ref-Avg-QV	The mean of quality values of all reference allele reads at the current position.	29
Novel-Avg-QV	The mean of quality values of all novel allele reads.	0
Heterozygous	Heterozygosity flag. Values are: **0**: Homozygous SNP. **1**: Heterozygous SNP.	0
Algorithm	The algorithm used to call the current SNP. Values are: • **–1**: Not a SNP. • **0**: Homozygous position. • **1**: Heterozygous SNP, called by the Bayesian algorithm. • **2**: Heterozygous SNP, called by the Frequentist algorithm.	–1
Flag	Flag indicating why a location is not called a SNP. See Table 44 for a description of Flag values.	m4

The flags in Table 44 explain why a read is filtered out.

Table 44 <diBayes.output.prefix>_Consensus_Calls.txt flag column description

Flag	Filter meaning	Related parameter
Heterozygote		
t1	Insufficient coverage for a heterozygous position.	dibayes.het.min.coverage.

Table 44 <diBayes.output.prefix>_Consensus_Calls.txt flag column description *(continued)*

Flag	Filter meaning	Related parameter
t2	Insufficient unique start positions for a heterozygous position.	dibayes.het.min.start.pos
t3	The coverage is too high, compared to the coverage distribution of all positions.	dibayes.het.skip.high.coverage
t4	The fraction of the second-most common *valid* color (or base allele) in the total of top two valid colors (or the top two base alleles) is higher than the threshold (usually a function of raw error squared).	dibayes.het.min.allele.ratio
t5	Genome positions with sufficient coverage (20x) at which there is only 1 unique read position for all the reads. It could be a PCR error.	—
t6	The candidate SNP is evenly distributed over positions (not used).	—
t7	The non-reference allele is not on both strands (counting the reads).	dibayes.snp.both.strands
t8	Both alleles not evenly represented on both strands (doing statistical test on read distribution of the both strands).	dibayes.snp.both.strands
t9	The second-most common valid dicolor is not more frequent than the third-most. Color space only.	—
t10	There are no other valid SNPs, or the second-most common valid SNP (as a proportion of all valid SNPs) is less than half the 2-dibase error frequency. Color space only.	dibayes.het.min.allele.ratio
t11	Sum of the first and second-most common alleles must be at least this proportion, for example, 0.5, of all reads at this position.	dibayes.het.min.ratio.validreads.totalreads.ratio
t12	The dicolor alleles are inconsistent to the reference allele. Color space only.	—
t13	The difference of quality values of the non-reference allele and the reference allele is much lower the quality value of the reference allele.	—
t14	The average of p-values of color/base quality values of the non-reference allele is lower than one standard deviation from the mean of p-value distribution, when the non-reference allele has to be at low frequency (rare variant).	—
t15	The average of p-values of color/base quality values of the position is lower than one standard deviation from the mean of the p-value distribution.	—
t16	Insufficient coverage of the reference allele.	—
t17	Insufficient coverage of the non-reference allele.	—
t18	Zero coverage.	—
t19	Insufficient number of start positions of non-reference allele.	—
t20	Insufficient coverage for either of two alleles, when neither allele is same as the reference.	—
t21	Quality value of non-reference allele too low (lower than a relative threshold depend on the distribution of all color quality values).	—
t22	Quality value of non-reference allele too low (lower than an absolute threshold).	dibayes.het.min.nonref.color.qv
t23	The less common allele is not on both strands.	dibayes.het.lca.both.strands
t24	Insufficient number of start positions of the less common allele.	dibayes.het.min.lca.start.pos

Table 44 <diBayes.output.prefix>_Consensus_Calls.txt flag column description *(continued)*

Flag	Filter meaning	Related parameter
t25	Quality value of the less common allele too low (lower than an absolute color quality value).	dibayes.het.min.lca.color.qv
t26	The color QV difference between the most common and less common allele is larger than the user defined threshold.	dibayes.het.max.diff.color.qv
t27	The ratio of filtered reads and raw reads at a position is lower than a user defined threshold.	dibayes.snps.min.filteredreads.rawreads.ratio
Homozygote		
m1	Insufficient coverage for a homozygous SNP.	dibayes.hom.min.coverage
m2	Insufficient proportion of the count of the most common valid dicolor (base) allele to all valid reads.	—
m3	Non-reference allele not on both strands.	dibayes.snp.both.strands
m4	Reserved for future filters (not used).	—
m5	Second-most common allele too close in coverage to the first most common allele.	—
m6	Insufficient coverage (as a fraction of the median of the coverage distribution of all positions) for a homozygous call.	—
m7	Dicolor inconsistent with reference. Color space only.	—
m8	Insufficient number of non-reference alleles	dibayes.hom.min.nonref.allele.count
m9	Same as the m2 flag.	—
m10	Insufficient number of start positions of non-reference alleles.	dibayes.hom.min.nonref.start.pos
m11	No coverage.	—
m12	Quality value of non-reference allele too low (lower than an absolute threshold).	—
m13	Quality value of non-reference allele too low (lower than a relative threshold that depends on the distribution of all color quality values).	—

Consensus calls file example

The following is example output of a SNPs consensus calls file.

```
##consensus-call
##source-version LifeScope(tm) 2.0 diBayes
##type DNA
##date 2011-03-10
##time 11:38:51
##genome-build /local/xux1/validataion/HuRef/hg18/reference/
chr1.validated.fasta
##input-files /local/xux1/validataion/HuRef/hg18/input/
solid0107_20100514_PE_launch_verification_F3.sowmi-1_0309.bam
##options call-stringency: medium; dibayes-base-space: 0;
experiment-name:
Test_huref_4.0_base_newbam0309_1of3_color2.0_chr1; het-skip-
high-coverage: 1; index: 1; log-dir: /local/xux1/validataion/
HuRef/hg18/log/newbam0309_1of3_color2.0; output-dir: /local/
```

```
xux1/validataion/HuRef/hg18/outputs/newbam0309_1of3_color2.0;
poly-rate: 0.003; reference: /local/xux1/validataion/HuRef/
hg18/reference/chr1.validated.fasta; working-dir: /local/xux1/
validataion/HuRef/hg18/tmp/newbam0309_1of3_color2.0;
#Chr   Position     Allele_DiColor1 Allele_DiColor2 Reference
Genotype        P-value Coverage        Unfiltered_Coverage
nCounts_1st_allele      nCounts_Reference_allele
nCounts_NonReference_allele Ref_Avg_QV      NonRef_Avg_QV
Zygousity       Algorithm       Flag
chr11   50262   10      10      T       T       1.000000        1
1       1       1       0       25      0       homozygous      -
1       t4,t10,t1,t2,t21,t22,m4
chr11   50263   00      00      T       T       1.000000        1
1       1       1       0       25      0       homozygous      -
1       t4,t10,t1,t2,t21,t22,m4
```

Contig base space FASTA file

The <*prefix*>_Consensus_Basespace.fasta file is a base space translation of the genotype column of its corresponding consensus call file. The bases are encoded with IUB codes. All reference positions that have 0 coverage are replaced by "N".

The individual base space FASTA files are created with file names according to the following pattern:

```
${task.output.dir}/contig${index}/${analysis.name}_\
Consensus_Basespace.fasta
```

The following is example content of a contig base space FASTA file. The title line is always a ">" sign followed by the chromosome name. The length of each sequence line is 70.

```
>chromosome_name
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNN
NNNNNNNNNNNNNNNNNNACGGGGGGGGGGGGGGGTACCGAAAAAAAAAAAAAAGGTCCCCCCCCCC
CCCCCCC
```

Combined FASTA for all contigs

The combined FASTA file for all contigs is a consolidated version of individual base space FASTA files. The combined file is named according to the following pattern:

```
${task.output.dir}/${analysis.name}_Consensus_Basespace.fasta
```

For field information and example content, see "Contig base space FASTA file".

Quartiles file

The <*prefix*>_quartiles.txt file lists the quartile and percentile information about the coverage and color quality value distribution of all positions of the chromosome. Quartiles files are created with file names according to the following pattern:

```
${task.output.dir}/contig${index}/${analysis.name}_ quartile.txt
```

The following shows an example output of a SNPs quartiles output file.

```
25 percentile of the coverage distribution (the first quartile): 4
50 percentile of the coverage distribution (median):            11
```

```
75 percentile of the coverage distribution (the third quartile): 23

0.1 percentile of the average color quality value per position
distribution: 4
0.2 percentile of the average color quality value per position
distribution: 4
1   percentile of the average color quality value per position
distribution: 5
50  percentile of the average color quality value per position
distribution (median): 19
```

SNPs algorithm description

LifeScope™ Software uses the diBayes package to find Single Nucleotide Polymorphisms (SNPs). The diBayes package performs independent SNP analysis at each position in the reference, using either a Bayesian or Frequentist algorithm (see Figure 9 on page 207 and Figure 10 on page 208).

Frequentist algorithm

The Frequentist algorithm is used when coverage is high for a given position, for example, 60x. Given the assumption that the errors follow a Poisson distribution, the Frequentist algorithm calculates the probability of a null hypothesis that the observed valid dicolor base mismatches are errors. If the probability of the null hypothesis is too low, the hypothesis is rejected and the position is called a SNP.

Bayesian algorithm

The SNP finding problem is to predict the most likely genotype (G) at a genome position from a list of reads (R) that are mapped to that position. To do that, for each genome type G, we want to estimate the posterior probability of $P(G|R)$. Using Bayesian theorem, $P(G|R)=P(R|G)P(G)/P(R)$, where $P(G)$ is the prior probability of the genotype G, $P(R|G)$ is the posterior probability of observing the given read set at the position assuming the true genome type is G, and $P(R)$ is the prior probability of the read sets. $P(R)$ is a constant for a given set of R and can be ignored from further consideration. To estimate $P(R|G)$, the algorithm takes into consideration the color (or base) quality values, position error profiles, and probe error profiles. (The Frequentist algorithm also uses these information.)

Define $S(G)=P(R|G)P(G)$ and $T = \sum_{G} S(G$, the likelihood of each genotype, $L(G)$,

can be estimated by $S(G)/T$. To simplify the computation, our Bayesian algorithm compares two hypotheses; the first, that the true genotype is homozygous for the most common allele (G_0) in the reference, and the second, that the true genotype is heterozygous (SNP) for the top two alleles (G_a) in the reference. Therefore, T can be simplified to $S(G_0)+S(G_a)$, and $L(G_0)=S(G_0)/T$, and $L(G_a)=S(G_a)/T$. The algorithm reports the genotype with the highest likelihood.

Data flow

Figure 9 shows how diBayes can take multiple BAM files that are sorted by genome positions as input to make SNP calls at positions with coverage. Running diBayes requires the position error and probe error files of each BAM file. The position error files and probe error files are generated by the mapping statistics module. In a multi-CPU computing cluster, diBayes automatically distributes parallel jobs for individual chromosomes to different computing nodes. The final results for the whole genome are merged after the individual jobs are complete.

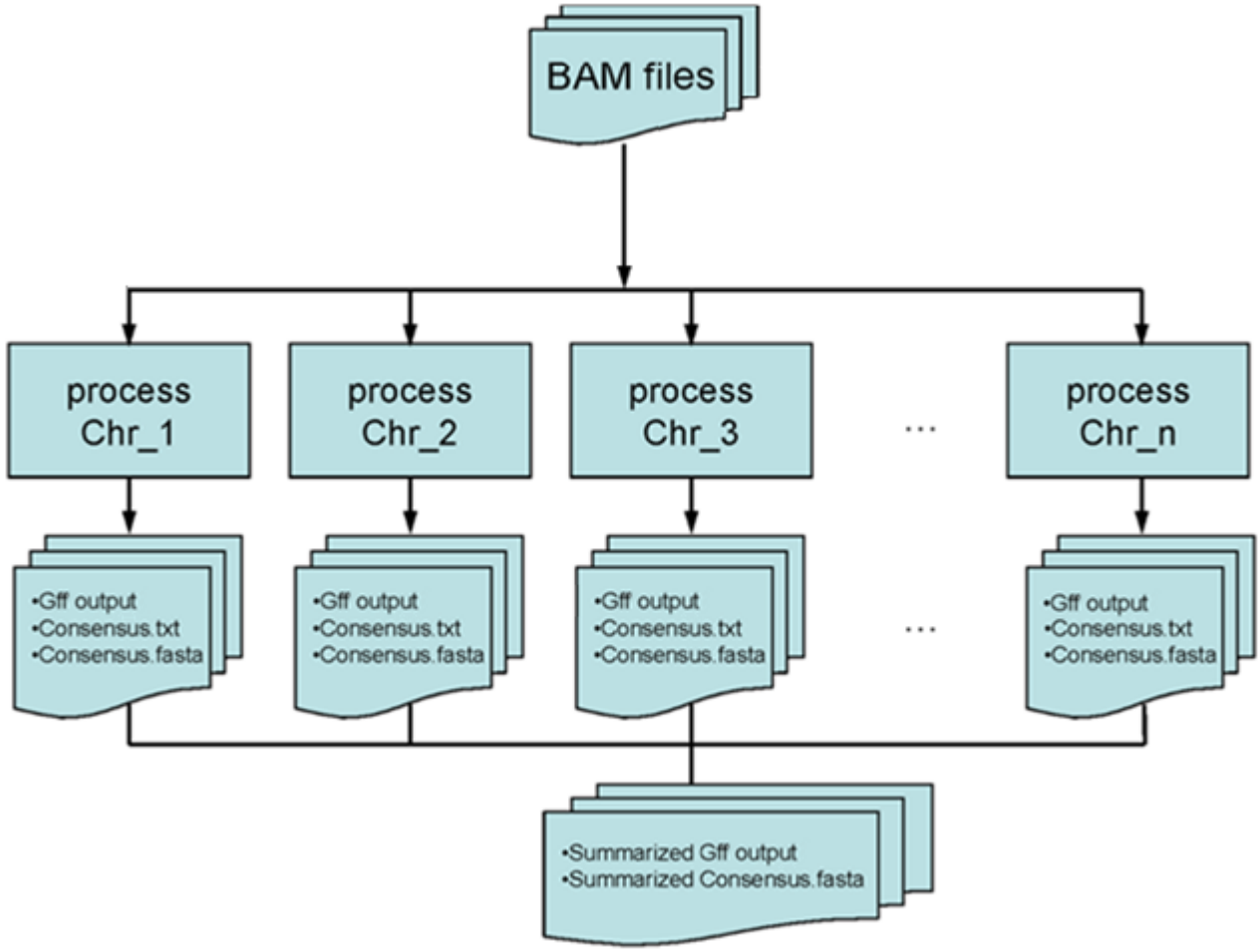


Figure 9 diBayes data flow

Internal module flow

The SNPs module first validates the input BAM files, then determines whether to use its internal color-space or base-space processing.

BAM file validation

The SNPs module requires the following:

- Each BAM file must have only one read group and only one library type.
- Each BAM file is sorted by genomic coordinates.
- The reference sequence lengths in BAM file header and in reference FASTA file are identical.

- The BX and BY fields in the BAM headers of all input BAM files must be consistent. (The BX and BY fields record whether the original XSQ reads file had base space present on tag 1 and tag 2, respectively.)

If BAM files that do not meet these criteria are included in the analysis, the analysis fails.

Color-space and base-space processing

The SNPs module uses its color-space processing algorithm if the following are both true:

- For paired data, BX and BY are both 0, or, for fragment data, BX is 0.
- Color space information is present in the BAM data (CS and CQ fields are present in the first 10 read records).

For other conditions, the SNPs module uses its base-space processing algorithm. (The exception is BAM files with inconsistent BX and BY fields. These BAM files are rejected previously during BAM file validation.)

The type of processing selected is important because certain filtering and other parameters are only effective on either base space or color space, but not both. The parameters that only apply to one type of processing are noted in the parameters table, Table 38 on page 191.

Both types of processing follow the steps shown in Figure 10.

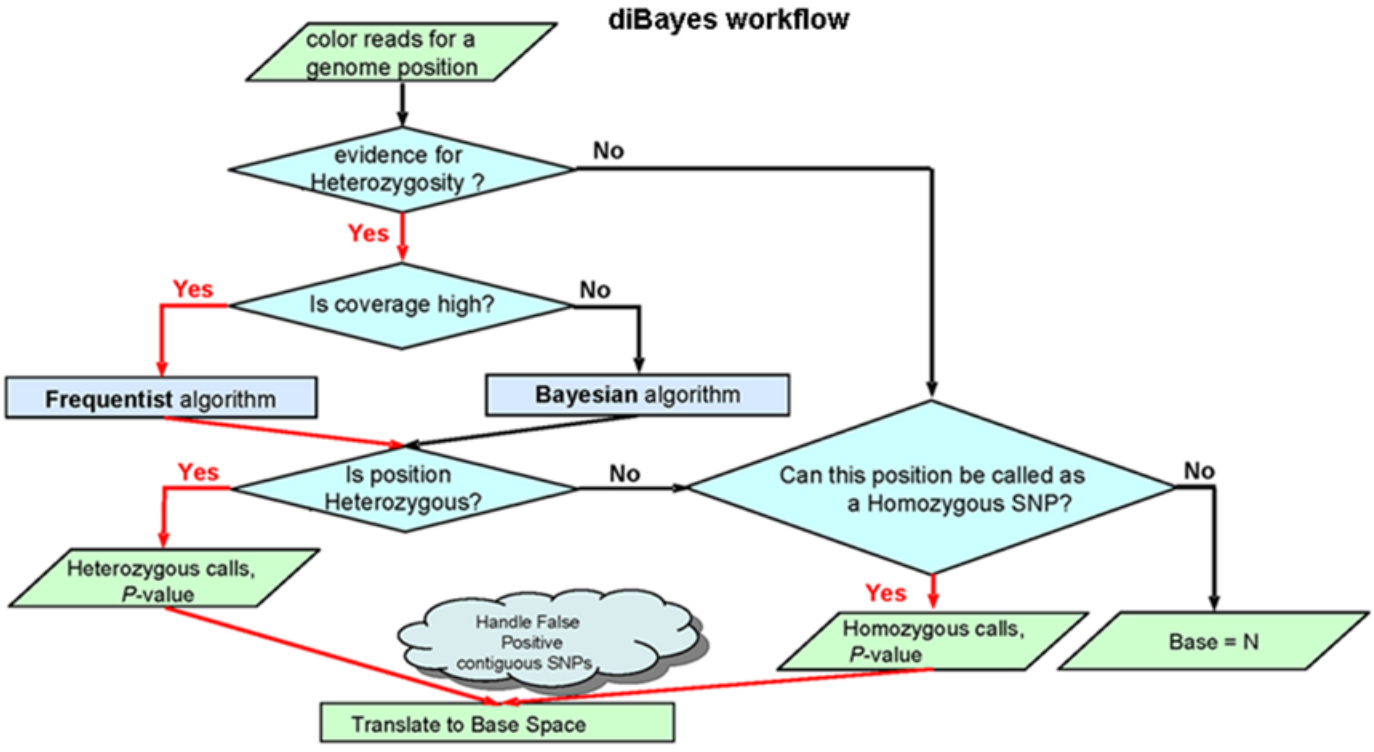


Figure 10 diBayes algorithm flowchart

FAQ – SNPs

1

SNP calling is taking too long. What can I do?

Parallelization is the best solution for running diBayes on large data sets. LifeScope™ Software has implemented the parallel distribution of diBayes jobs for individual chromosomes. It can help the users to achieve the best running efficiency. The diBayes uses BAM files as its input and removes all large temporary files (improving run times).

2

I seem to be missing SNPs — how do I troubleshoot?

First, try repeating the analysis with a lower stringency level (for example, changing `dibayes.call.strengency` from `medium` to `low`). Look at the positions that you expect to be SNPs in the consensus calls file. Typically, you should find a list of flags that describe the reasons the position was not called as a SNP (see Table 42 on page 199). Look at the properties of this position. Visualizing the reads might be helpful here.

Is the coverage much higher than average? The filter `dibayes.het.skip.high.coverage` can remove heterozygous SNPs at positions of extremely high coverage because these have previously been observed to be variants in repeat regions rather than truly heterozygous SNPs at a single position.

Are the reads strongly biased towards one strand, or is the non-reference allele missing from one strand? You probably want to perform a run with `dibayes.call.stringency=medium`, or switch off the "both strands" requirement (`dibayes.snp.both.strand=0`).

Do all the reads have the same start position because of the way the sample was prepared? You need to reduce the number of unique start positions required to call a SNP (`dibayes.het.min.start.pos` and `dibayes.hom.min.start.pos`).

3

I seem to be finding too many SNPs — how do I troubleshoot?

Look at the properties of the SNPs and compare them to the properties of all the positions in the consensus_calls.txt file. Is the coverage of these SNPs much lower or much higher than average? Is the color quality value of the non-reference allele much lower than average? You might want to post-filter the results if you find these kinds of patterns. For example, you might want to remove SNPs with very low coverage, or very low color quality values, or p-values close to one. You might want to repeat the analysis with a more stringent setting for example, changing the parameter `dibayes.call.stringency` from medium to high.

Filtering by p-values is not recommended for homozygous SNPs. See also FAQ 4, "How do I interpret the SNPs p-values?" on page 210.

4

How do I interpret the SNPs p-values?

P-values close to 0 mean that the algorithm has strong confidence that the genotype call is correct. P-values close to 1 mean that the algorithm does not have very strong confidence that the genotype call is correct.

Note that the p-value refers to the probability that the *genotype* call is correct. This meaning is not the same as the probability as "there is a SNP at this position". A homozygote SNP can have a p-value close to 1, but it is still highly likely that a SNP is present at this location.

Most commonly, a p-value close to 1 occurs when there is good evidence that a SNP is present, *but* it is difficult to confidently decide whether the genotype is heterozygote or homozygote. For example, take a case where there is coverage=3 and all of these are high QV non-reference allele calls (e.g., the reference is A and there are three C's observed). In this case, it is clear there is a SNP in this location. However, it is possible this might really be a heterozygous SNP, and if we observed a fourth allele, it might be an A. So the p-value on this homozygous SNP call might be close to 1, since we are not very confident of the genotype call, even though we are confident that there is a SNP in this location. In this case, we are still confident that a SNP is present at this location.

For this reason, it is recommended not to filter out homozygous SNPs with high p-value, as this will result in removing many real SNPs.

5

Can the SNPs module find two adjacent SNPs?

Yes, in some circumstances. You must set the `dibayes.detect.2.adjacent.snps` parameter to 1. With this setting, diBayes detects and properly calls these scenarios:

- Two adjacent heterozygous SNPs
- Two adjacent homozygous SNPs
- A homozygous SNP adjacent to a heterozygous SNPs (base-space algorithm only)
- A heterozygous SNP adjacent to a homozygous SNPs (base-space algorithm only)

However, diBayes does not call a homozygous SNP adjacent to a heterozygous SNP in color-space BAM files (even with `dibayes.detect.2.adjacent.snps` on). In general, adjacent SNPs calls may have more false positives than single SNPs calls do.

6

Why does SNPs output sometimes contain a blank dicolor value?

Occasionally, the color sequence is not reported and the base sequence is reported as an 'N', before and after a SNP. This reporting usually occurs when the non-reference allele is more abundant than the reference allele. In these cases, the bases flanking the SNP are the normal reference sequence. (The color sequence appears as a space character in the output file. The space character may affect pattern-matching tools.)

7

How can I control the sensitivity and specificity of SNP calling?

Different call stringency settings and filter settings may help users to achieve different sensitivity and specificity requirements. The more stringent the filters are, the less sensitivity the SNP call and the higher specificity in general. Two filters that are very flexible for users' different needs are the `dibayes.reads.min.mapping.qv` (MQV) and `dibayes.reads.min.alignlength.readlength.ratio` (ARR) parameters. Changing one or both of these parameters filters out low confident reads and gets SNP calls with high accuracy. The following figures show that changing MQV and ARR may affect the total number of SNP calls and the dbSNP concordance of the predictions.

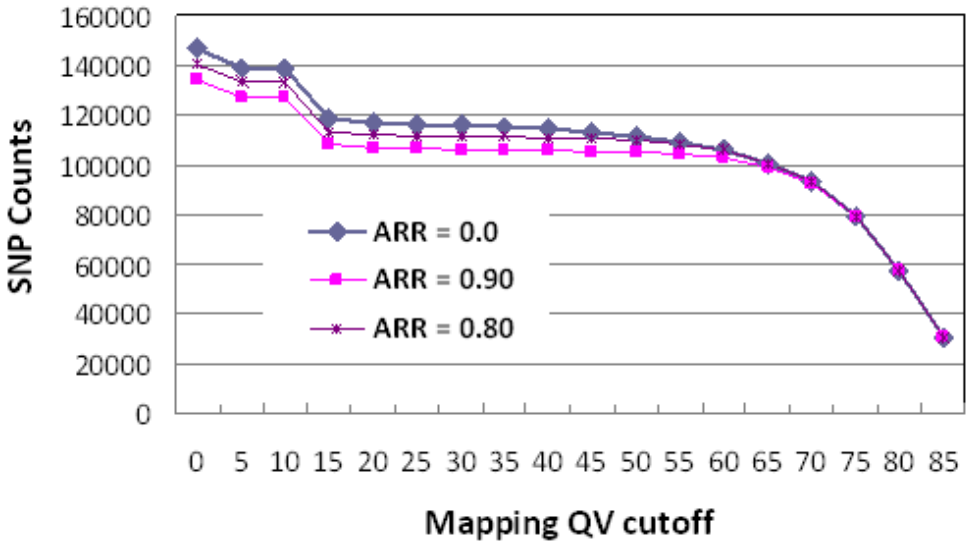


Figure 11 Total SNP calls on chromosome 1 of a HuRef long mate-pair data as a function of mapping QV cutoffs and ratios of alignment length and read length (ARR)

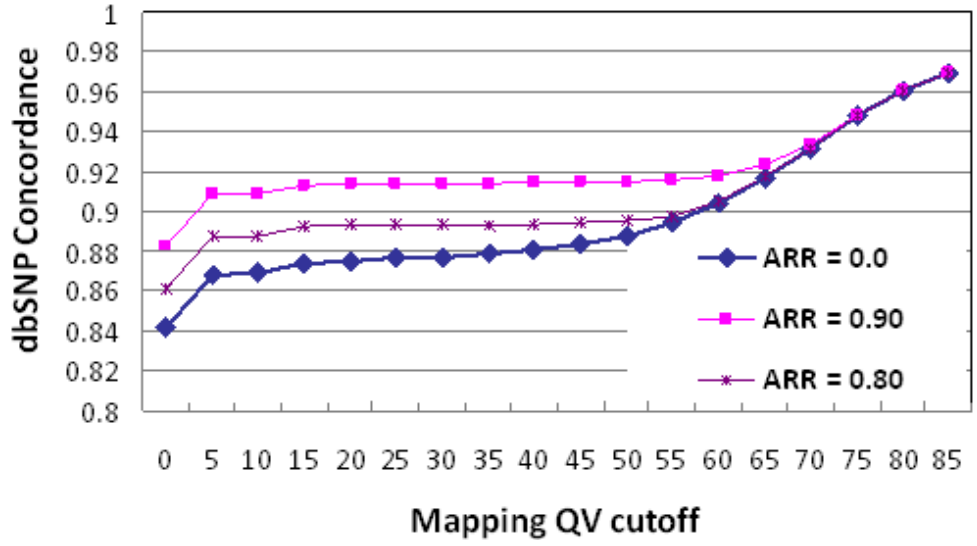


Figure 12 dbSNP concordance of SNP calls on chromosome 1 of a HuRef long mate-pair data as a function of mapping QV cutoffs and ratios of alignment length and read length (ARR)

8

Why are the coverages reported by diBayes for individual genomic positions sometimes different from the coverage derived from input BAM files?

The difference happens because diBayes reports the coverage of filtered reads instead of raw coverages derived from input BAM files. The diBayes module calls SNPs only for these reads that are:

- Based on reads that are primary alignments.
- Are classified as an AAA pair (See "FAQ – Pairing" on page 182 for a description of the 3-letter genomic code classifications).
- Are not duplicates.
- Do not contain indel.
- Have high mapping quality values (default is 8).
- Pass other read level filters (such as optional tuning parameters such as `dibayes.reads.max.mismatch.alignlength.ratio` and others described in Table 38 on page 191).

The SNPs module reports both coverage and raw coverage at every position, for users' convenience.

Users can find detailed information about the numbers of reads filtered by read level filters and the total number of raw reads at the end of each algorithm log file.

9

Why is a SNPs analysis not recommended with whole transcriptome data?

The reverse transcription process that is part of the WT library prep intrinsically adds a percentage of sequencing errors. In addition, alternative splicing and other alignment complexities around exon junctions may cause local misalignments, which can lead to false positive SNP calls. Although the SNPs module can run successfully on BAM files generated from the WTA module, using those BAM files may cause a higher number of false positive SNPs.

11 Run a Human CNVs Analysis

This chapter covers:

- Overview 213
- Examples of running a CNV analysis 214
- CNV module parameter descriptions 215
- Human CNVs results file format description. 218
- Human CNVs results file examples. 220
- Algorithm for the Human CNV module. 221
- FAQ – Human CNVs 224

Overview

The LifeScope™ Software Copy Number Variation (Human CNV) module detects copy number variations in a data sample that is mapped to the reference genome. The analysis method is based on depth of coverage. The module performs these phases in the analysis:

- Samples the genome sequence into non-overlapping windows
- Identifies the regions with significantly higher or lower coverage
- Assigns copy number calls to the identified regions, based on the scale of coverage

The CNV module requires predicted mappability files, which are provide with the LifeScope™ Software reference repository. These pre-computed files are available for the human reference sequences hg18 and hg19. The module currently supports only human CNV detection. By default, the module does not call CNVs within 1 MBase of the chromosomes' centromeres and telomeres. These regions are highly repetitive and tend to contain many apparent CNVs that are not of general interest. The size of the blackout regions is a user-configurable parameter, `cnv.trim.distance`. The module is designed to detect larger CNVs (which can take up whole chromosomes or large portions of chromosomes) as well as smaller CNVs, by doing either global or local normalization, respectively. By default, and if the number of valid chromosome arms is greater than 6, the module does global normalization. The parameter `cnv.local.normalization` is provided to switch to local normalization.

The major components of the CNV module include the following:

- Coverage calculation

- Sampling into windows
- Normalization
- Segmentation
- Post-processing

Examples of running a CNV analysis

In a standard workflow

The following standard workflows provide examples of running the CNV module:

- genomic.resequencing.frag
- genomic.resequencing.lmp
- genomic.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# create a project and open it
cd /projects
mk proj1
cd proj1
# create an analysis and open it
mk run1
cd run1
# define the workflow
set workflow genomic.resequencing.pe
# define the input (dummy file names are used in the example)
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
# define the reference
set reference hg19
# optionally change CNV defaults here
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
```

See for Chapter 7, “Run a Standard Workflow Analysis” on page 111 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 83 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an example of how to run the CNV module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, “Demo Analyses” on page 567 for information on running the CNV module as a standalone analysis.

CNV module parameter descriptions

This section describes the parameters that control the runtime behavior of the CNV module. See also Table 46 on page 217, which lists the effect of the `cnv.stringency.setting` value on other stringency parameters.

In order to change a parameter value in your analysis, use the `set param` shell command to replace the line `# optionally change CNV defaults here` in the example commands in "In a standard workflow". For instance, to change the call stringency, use this shell command:

```
set cnv.stringency.setting High tertiary/cnv.ini
```

Table 45 CNV module parameter descriptions

Parameter name	Default value	Description
CNV parameters		
cnv.window.size	5000	Size of the window block to be considered as a region. Allowed values: Integers >= 100.
cnv.trim.distance	1000	Distance in kilo bases to be trimmed from the extreme ends of the chromosome arms. Allowed values: Integers 0–100000.
cnv.min.quality	2	Minimum mappability quality value of the alignments. Allowed values: Integers 0–100.
cnv.ploidy	2	General ploidy of the genome. Allowed values: Integers >= 1.
cnv.ploidy.exception	None	List of all the contigs whose ploidy is different to the general ploidy of the genome. Entries in the list are in the format {contig id: ploidy of the contig}, and are separated by commas. Use the string "None" to indicate no entries.
cnv.local.normalization	0	Whether or not genome-wide normalization or local normalization should be performed. Allowed values: • **0**: Perform genome-wide normalization. • **1**: Perform chromosome-arm local normalization.
cnv.write.coverage	0	Whether or not to create coverage output files. Allowed values: • **0**: Do not create coverage output files. • **1**: Create coverage output files, in WIG format
cnv.coverage.wsize	1000	Size of the window block to be considered as a region for writing coverage output. The mean coverage of all bases in each of these windows is output. Allowed values: Integers 0–100000.
cnv.gender	Male	Set the ploidy of sex chromosomes in Human correctly. Allowed values: Male, Female.
CNV stringency parameters		

Table 45 CNV module parameter descriptions *(continued)*

Parameter name	Default value	Description
cnv.stringency.setting	Medium	Set the stringency setting for calling CNVs. Allowed values: • **High**: Recommend when a very low false positive tolerance is allowed. • **Medium**: Default values. • **Low**: Aggressive CNV calling. Lower settings results in more CNV calls, but with more false positives. Higher settings result in fewer CNV calls, but with fewer false positives. See also Table 46 on page 217. **Note:** For GUI users, changing the stringency setting resets the values of certain other filtering parameters (described in Table 46). On the command line, the other parameters are not reset automatically. You must set each parameter.
cnv.deletions.min.mappability	50	Minimum mappability percentage for regions to be shown as copy number deletions. Allowed values: Floats 0.0–100.0. If this parameter is not specified, and • cnv.stringency.setting is set to *High*, then this parameter is set to 75. • cnv.stringency.setting is set to *Low*, then this parameter is set to 10.
cnv.insertions.min.mappability	10	Minimum mappability percentage for the regions to be shown as copy number insertions. Allowed values: Floats 0.0–100.0. If this parameter is not specified, and • cnv.stringency.setting is set to *High*, then this parameter is set to 25. • cnv.stringency.setting is set to *Low*, then this parameter is set to 0.
cnv.deletions.min.windows	2	Minimum number of windows for the regions to be shown as copy number deletions. Allowed values: Integers >= 0. If this parameter is not specified, and • cnv.stringency.setting is set to *High*, then this parameter is set to 4. • cnv.stringency.setting is set to *Low*, then this parameter is set to 1.
cnv.insertions.min.windows	2	Minimum number of windows for the regions to be shown as copy number insertions. Allowed values: Integers >= 0. If this parameter is not specified, and • cnv.stringency.setting is set to *High*, then this parameter is set to 4. • cnv.stringency.setting is set to *Low*, then this parameter is set to 1.
cnv.deletions.max.pval	1.0	Maximum p-value for regions to be shown as copy number deletions. Allowed values: Floats 0.0–1.0. If this parameter is not specified, and cnv.stringency.setting is set to *High*, then this parameter is set to 0.1.
cnv.insertions.max.pval	1.0	Maximum p-value for regions to be shown as copy number insertions. Allowed values: Floats 0.0–1.0. If this parameter is not specified, and cnv.stringency.setting is set to *High*, then this parameter is set to 0.1.

Table 45 CNV module parameter descriptions *(continued)*

Parameter name	Default value	Description
cnv.deletions.max.ratio	0.5	Maximum ratio between the coverage of the region and the expected coverage, for a region to be called as CNV deletion. Allowed values: Floats 0.0–1.0. If this parameter is not specified, and • cnv.stringency.setting is set to *High*, then this parameter is set to 0.3. • cnv.stringency.setting is set to *Low*, then this parameter is set to 0.6.
cnv.insertions.min.ratio	1.25	Minimum ratio between the coverage of the region and the expected coverage, for a region to be called as CNVs insertion. Allowed values: Floats >= 1.0. If this parameter is not specified, and • cnv.stringency.setting is set to *High*, then this parameter is set to 1.5. • cnv.stringency.setting is set to *Low*, then this parameter is set to 1.1.
Resource parameters		
memory.request	15gb	Memory request. Include the units gb in the setting.
java.heap.space	3000	Dynamic memory request, in megabytes. Do not enter a unit with the numeric value.
processors.per.node	8	Number of processors per node. The maximum number of CPUs that all simultaneous processes are expected to use to full capacity.

Table 46 shows the non-default filter parameter values for each level of stringency that is set with the `cnv.stringency.setting` parameter. For GUI users, setting the `cnv.stringency.setting` parameter sets the other filter parameters with the non-default values shown in this table. For command shell users, changes to the `cnv.stringency.setting` parameter *do not* affect other filtering parameters, and you must set each filtering parameter individually. The medium stringency level is not listed because it calls for default values of the filtering parameters. If you explicitly set one of the listed filter parameters, that value overrides the value called for by the CNV stringency level.

Table 46 Filter parameters affected by the CNV stringency level

Call stringency setting	Filter parameter	Parameter setting change	
		Default value	New value
High	cnv.deletions.min.mappability	50	75
	cnv.insertions.min.mappability	10	25
	cnv.deletions.min.windows	2	4
	cnv.insertions.min.windows	2	4
	cnv.deletions.max.pval	1.0	0.1
	cnv.insertions.max.pval	1.0	0.1
	cnv.deletions.max.ratio	0.5	0.3
	cnv.insertions.min.ratio	1.25	1.5

Table 46 Filter parameters affected by the CNV stringency level *(continued)*

Call stringency setting	Filter parameter	Parameter setting change	
		Default value	New value
Low	cnv.deletions.min.mappability	50	10
	cnv.insertions.min.mappability	10	0
	cnv.deletions.min.windows	2	1
	cnv.insertions.min.windows	2	1
	cnv.deletions.max.ratio	0.5	0.6
	cnv.insertions.min.ratio	1.25	1.1

Table 47 describes CNV parameters that we do not recommend changing.

Table 47 Annotation internal parameters description

Parameter name	Default value	Description
cnv.run	1	Enables the CNV module. Allowed values: • **0**: Do not run a CNV analysis. • **1**: Run the CNV module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
cnv.mappability.dir	${analysis.mappability.dir}	Path to the directory of mappability files. This parameter is set automatically by the shell when you use the default hg18 or hg19 assemblies in the reference repository.

Human CNVs results file format description

This section describes the file format of the OUT files and the GFF file created by the Human CNVs module run.

OUT files

The OUT files and their contents are:

- **AllSegments.out** – Contains all the segments, including segments with normal copy number.
- **AllCNVs.out** – Contains all the segments with copy number deletions or duplications (CN != 2).
- **OutputCNVs.out** – Contains all the segments with copy number deletions or duplications which pass the specified filtering criteria. These segments are the output CNVs, and are the same as those reported in a different format in the GFF file.

The three OUT files share a common file format, which is described in Table 48. You can view the files in a text editor or a spreadsheet application.

Wiggle file

The CNV module optionally creates, per chromosome, one wiggle format file named `Coverage_chr**.wig`. This file contains the coverage in specified window sizes. You can visualize the file in a browser such as the Integrative Genomics Viewer (IGV). You can download the IGV browser from the Broad Institute website:

www.broadinstitute.org/igv

The wiggle file specification is available from this site:

http://genome.ucsc.edu/goldenPath/help/wiggle.html

GFF file

The module creates one `OutputCNVs.gff` file. The file formats are described in Table 49. You can view the `OutputCNVs.gff` file in a text editor or a spreadsheet application. You can also visualize the file in a browser such as the Integrative Genomics Viewer (IGV), or in a browser such as the UC Santa Cruz genome browser. You can download the IGV browser from the Broad Institute website. For more information, go to these sites:

www.broadinstitute.org/igv

http://genome.ucsc.edu

Table 48 CNV *.out file format descriptions

Column Title	Description	Example
Chrom	Chromosome number.	chr1
start	Start location of the CNV region.	1636395
end	End location of the CNV region.	1780200
mappability	Fraction of mappable bases in the CNV region.	83.745453
coverage	Mean coverage of the windows in the CNV region.	4.080000
log2Ratio	Mean of Log2Ratios of the windows in the CNV region.	–1.126154
copy number	Copy number of the region.	The copy number is relative to a diploid genome, with a normal copy number of 2.
numWindows	Number of windows in the CNV region.	22
p-val	p-value of the CNV call for the region.	0.00001 is very confident. 0.99 is not at all confident.

Table 49 OutputCNVs.gff file format

Column Title	Description	Example
Chr	The ID of the sequence to which the start and end coordinates refer.	chr1
Source	Free text-qualifier indicating the algorithm or method that generated the feature.	AB_CNV_PIPELINE
Type	Sequence ontology derived type for this variation.	repeat_region
Pos_Start	Start position of the CNV Region.	1144255
Pos_End	End position of the CNV Region.	0.04223
Score	p-Value of the CNV Region.	—
Strand	Not used for this output.	—

Table 49 OutputCNVs.gff file format *(continued)*

Column Title	Description	Example
Phase [attribute]	Not used for this output.	—
Attributes		
copynumber	Copy number of the region.	1
logRatio	Mean of Log2Ratios of the windows in the CNV region.	–1.258771
numWindows	Number of windows in the Human CNV region.	23 Usually, the larger this number is, the more confident the CNV call is.
coverage	Mean coverage of the windows in the CNV region.	4.080000
mappability	Fraction of mappable bases in the Human CNV region.	91.421745 The maximum value is 100. A low number may indicate that this region is difficult to map, increasing the likelihood of a false positive CNV call.

Human CNVs results file examples

This section provides examples of the *.out files and the GFF file created when you run the Human CNVs module. This output can be viewed in a text editor or a spreadsheet application.

The following is example *.out file output, from an `OutputCNVs.out` file.

```
chrom   start   end   mappability   coverage   logRatio   copynumber   numWindows
p-val
chr12   50772857  50798807  59.6667  6.13873  -1.23401  1  3  0.00012201
chr12   51001962  51067993  98.3077  23.8285  0.628887  4  13 1.53379e-16
chr12   52032318  52051850  76.6667  7.27728  -1.09105  1  3  0.120583
chr12   52756738  52806997  30       5.85307  -1.3758   1  3  1.4977e-05
chr12   62211177  62434738  31.5385  5.84658  -1.21177  1  13 0.256666
```

The following is example output of the file OutputCNVs.gff:

```
##gff-version 3
##source-version LifeScope(tm) 2.0 {CNVcaller}
##feature-ontology http://song.cvs.sourceforge.net/*checkout*/song/ontology/
sofa.obo?revision=1.141
##type DNA
##date 2011-02-10
##time 15:44:31
##genome-build NCBI B36
##input-files /local/gottimrk/cnv_test_datasets/solid0122_20100412_chr12Q_chr16Q_CNVs-
Paired.bam
#Chr    Source  Type    Pos_Start  Pos_End  Score    Strand    Phase    Attributes
chr12   AB_CNV_PIPELINE     repeat_region   50772857  50798807   0.00012201  .      .
copynumber=1;numWindows=3;coverage=6.13873;logRatio=-1.23401;mappability=59.6667
chr12   AB_CNV_PIPELINE     repeat_region   51001962  51067993   1.53379e-16  .      .
copynumber=4;numWindows=13;coverage=23.8285;logRatio=0.628887;mappability=98.3077
```

```
chr12   AB_CNV_PIPELINE    repeat_region   52032318  52051850    0.120583   .      .
copynumber=1;numWindows=3;coverage=7.27728;logRatio=-1.09105;mappability=76.6667
chr12   AB_CNV_PIPELINE     repeat_region    52756738  52806997   1.4977e-05   .      .
copynumber=1;numWindows=3;coverage=5.85307;logRatio=-1.3758;mappability=30
```

Algorithm for the Human CNV module

The Human CNV module algorithm has six steps:

- Preprocessing
- Coverage calculation
- Sampling into windows
- Normalization
- Segmentation
- Post processing

Preprocessing

During preprocessing, user-defined configuration parameters are validated and initialized. The BAM file headers are validated against the reference sequence. Information about the chromosomes (including chromosome name, start and end locations of arms) is loaded from the mappability file. The chromosomes that are present in both the mappability file and in the reference sequence (with exactly the same name) are selected for the analysis.

Coverage calculation

A position's coverage is the number of alignments spanning the position in the chromosome. The coverage at every position is computed from the BAM file. By default, all alignments with a quality value of at least 8 are used to calculate coverage. Low quality alignments can be filtered out by modifying the mapping quality threshold parameter (`cnv.min.quality`).

Optionally, the coverage computed for every chromosome in this step is output in WIG file format. The `cnv.coverage.wsize` parameter controls the window sizes for this coverage output.

Sampling into windows

The algorithm divides the chromosomal region into windows of variable size, depending upon the mappability of the region. The window sizes are determined dynamically so that exactly the same number of mappable positions are in each window. The program distinguishes between mappable and unmappable positions using the precomputed mappability files.

The coverage mean for every window is computed by taking the average of the coverage values from all the mappable positions in each window. The log ratios between the coverage mean of every window and the expected coverage are computed. The mean of all windows in the whole chromosome arm is used as the expected coverage.

GC Correction

GC content is the number of G or C bases compared to the total number of bases in a particular region. In the regions of the genome where the percentage of GC content is either high or low, the coverage is observed to be decreased (see Figure 13 on page 222).

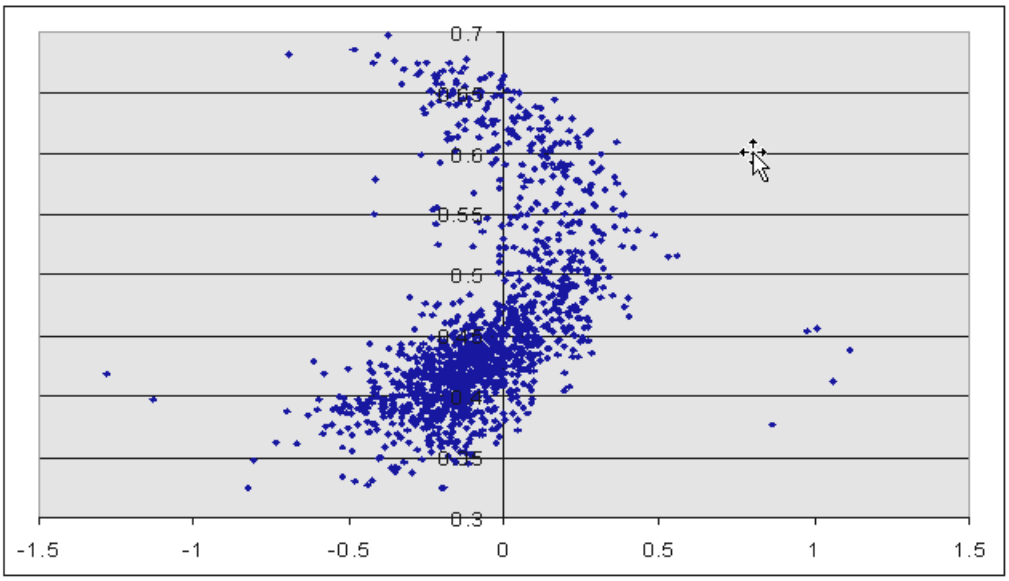


Figure 13 Extreme GC contact reduces coverage

In Figure 13, the X-axis is the log ratio (coverage of the window or coverage of the whole chromosome arm). The Y-axis is the percentage of GC content in each window.

The algorithm normalizes this effect of GC contact by scaling the coverage of the windows with extreme GC content to the median coverage. The scaling factors for the windows are computed inline for every chromosome arm during runtime by the algorithm.

Normalization

The previous section explained how the log ratios computed in the previous step use the mean coverage of the local chromosomal arm as the expected value. However, if the algorithm is used with these settings, it cannot detect large CNV regions spanning more than half the length of the chromosome arm. To detect large CNVs, which can take whole chromosome arms or large portions of chromosome arms, the algorithm performs global normalization. The algorithm computes log ratios by using the median of the coverage means of all chromosome arms as the expected value (see Figure 14).

To skip the normalization step, set the local normalization parameter `cnv.local.normalization` to 1. By default, the algorithm performs global normalization if the number of valid chromosome arms provided for the analysis is nine or higher. In some cases, you might perform an initial run using Global Normalization, and then perform a second run using local normalization to detect more fine-grained Human CNVs.

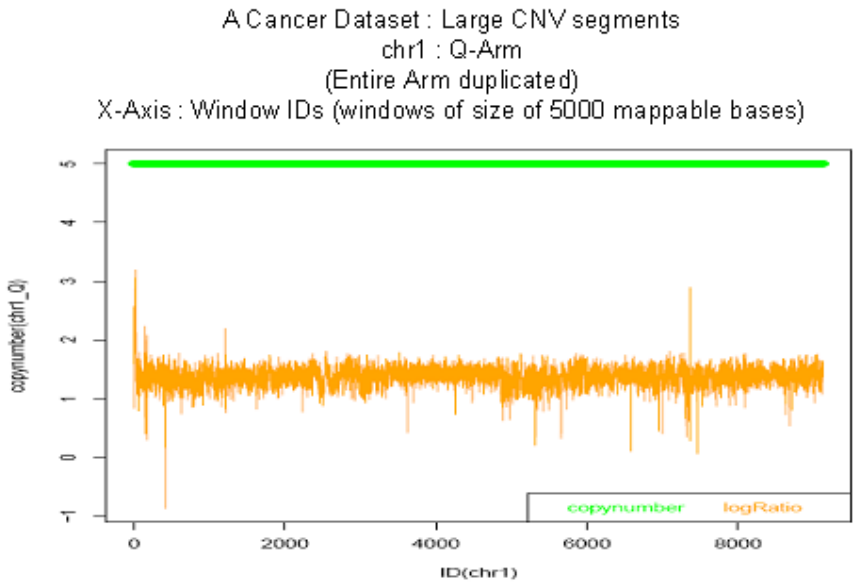


Figure 14 Large CNV segments example

Segmentation

The algorithm uses the Finite First Order Bayesian Hidden Markov Model ("the model") to take the normalized log ratios of the windows as input and convert the continuous log ratio values into discrete copy number states (see Figure 15).

The model consists of 10 states {0, 1, 2, 3, 4,…9}, where each state represents a copy number. For diploid species, state 2 is the normal state, states less than 2 are copy number deletions, and states greater than 2 are copy number amplifications.

The prior probabilities and transition probability matrix for the model are trained using the Baum-Welch EM algorithm.

A copy number state with a p-value is assigned to each window using the Viterbi decoding algorithm. The p-value is a measurement of statistical significance. In general, the lower the p-value, the more likely it is that the prediction will be true.

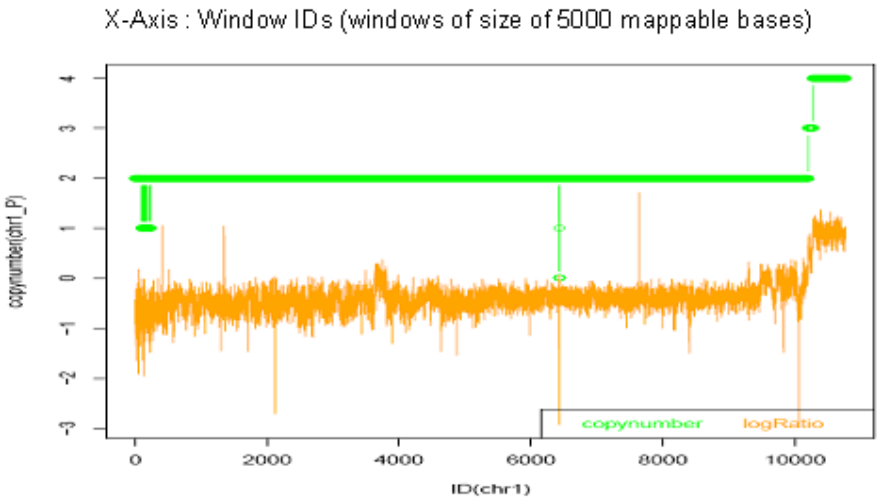


Figure 15 Segmentation example

Post processing

Adjacent windows with the same copy number are merged into a segment.

The algorithm applies filtering criteria on the Human CNV calls according to parameter settings (see Table 45 on page 215 for descriptions of these parameters). The filtering criteria can include parameters defined for minimum mappability, number of continuous blocks with copy number greater than two, and so forth. The algorithm uses the GFF format for structural variants to format the Human CNV segments that pass all of the filtering criteria. See Table 49 on page 219 for a description of the CNV GFF file format.

FAQ – Human CNVs

1

Does the Human CNVs module work on any species?

No. The current version of the module works only for the human hg18 and hg19 reference. The module cannot work on other species because LifeScope™ Software does not have predicted mappability files for any species other than human.

2

How do various parameters affect the sensitivity and specificity of the CNV calls?

The default configuration is designed such that CNV calls are made with a balance between sensitivity and specificity.

To increase the specificity, increase the `cnv.window.size` value, use only high quality alignments by filtering out the low quality ones using `cnv.min.quality`, and change the `cnv.stringency.setting` to `High`.

To increase the sensitivity, work at higher resolution by using a smaller value for `cnv.window.size`, by using a lower value for `cnv.min.quality`, and by changing the `cnv.stringency.setting` to `Low`.

3

What do the mappability files contain?

The mappability files are essentially a representation of reads from the reference that are mapped back to the reference. The mappability files have one row per every position, indicating whether that position is or is not uniquely mappable. A *mer* is the number of the bases per read. These files are in a binary format (HDF).

4

What mappability files are provided?

LifeScope™ Software provides mappability files for 50.4 fragment mapping.

5

Can users generate mappability files if they have FASTA files?

LifeScope™ Software does not provide any applications that allows users to generate their own customized mappability files. Contact your Life Technologies BioInformatics FAS for additional questions.

6

When should we change the default value of the `cnv.window.size` parameter?

The size of Human CNV segments detected by the module is directly dependent on the value of `cnv.window.size`. Typically, the smallest Human CNV segment that can be detected is at least twice the value of the `cnv.window.size` setting. The smaller the window size, the smaller the CNV that can be detected. However very small CNVs may be more likely to be false positives (or at least, under-represented in existing public databases).

As the `cnv.window.size` value decreases, the time taken for the Human CNV analysis and the sizes of the files generated increases significantly.

7

When should we use the `cnv.local.normalization` parameter?

To detect smaller Human CNVs (kbp scale) with tumor samples, normalize by the local chromosome context to detect these smaller Human CNVs. If you do not use `cnv.local.normalization`, large perturbations in ploidy across the whole genome might confuse detection. Ploidy is the number of complete sets of chromosomes in a biological cell. In humans, the somatic cells that compose the body are diploid (containing two complete sets of chromosomes, one set derived from each parent).

The module applies various user-configurable filtering criteria, such as minimum mappability, number of continuous blocks, and so forth on the Human CNV calls.

8

How are the p-values for Human CNV segments computed?

The p-values for Human CNV segments are computed using probabilities from the Finite First Order Bayesian Hidden Markov Model (the model). Sequence of log ratios of coverage per window are given as input to the model for segmentation. The module calculates the most probable Human CNV-state, that is, the hidden state, for each window using “Forward-Backward” Algorithm on the model. For example, if window 'W' is assigned Human CNV state 'c' with a probability 'p', then the p-value of that window is '1-p'. The calculations control Human CNV states and p-values for all windows. In the next step, the CNV algorithm merges the neighboring windows with similar copy number states into a Human CNV segment. The p-value of that Human CNV segment is given by the minimum p-value of all merged windows.

9

How do I interpret a CNV with a copy number value of 2?

CNVs with a copy number value of 2 can occur in male samples because these samples have only one X chromosome and one Y chromosome. Because the normal copy number is 1 for these chromosomes in males, a CNV with copy number 2 on these chromosomes is a gain in copy number. All CNVs that have a copy number value of 2 are present on either chromosome X or Y, and not on other chromosomes.

Run an Inversions Analysis

This chapter covers:

- Overview 227
- Examples of running an inversion analysis 228
- Inversion module parameters 228
- Input files 230
- Inversion output files 231
- Inversion algorithm 234

Overview

An inversion is defined by its two breakpoints. The numbers of mate pairs supporting occurrence of the starting and ending inversion breakpoints are counted for each base pair. The genomic ranges corresponding to local peaks of these counts, if above a score threshold, are called as candidate breakpoint ranges. To define an inversion, its starting and ending breakpoints are paired up only if they are the reciprocal nearest neighbor of each other in the correct order. The score for the inversion is the harmonic mean of its two breakpoints. Each breakpoint range can be scanned for coverage of normal (AAA) mate pairs to identify a sub-range with the lowest normal mate pair coverage as the most probable location of a breakpoint, and to differentiate homozygous inversions from heterozygous ones. The supporting evidence of all inversions can be visually inspected using a genomic browser.

Examples of running an inversion analysis

In a standard workflow

The following standard workflows provide examples of running the inversion module:

- genomic.resequencing.lmp
- genomic.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
lscope.sh shell -u username -w password
cd /projects
mk ecoli
cd ecoli
mk run1
cd run1
set workflow genomic.resequencing.pe
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
set reference hg19
# optionally change inversion defaults here
ls
run
```

See for Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information on running standard workflows. Chapter 6, "Run a Command Shell Analysis" on page 83 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an example of how to run the inversion module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, "Demo Analyses" on page 567 for information on running the inversion module as a standalone analysis.

Inversion module parameters

Table 50 describes the parameters that control the inversion algorithm.

In order to change a parameter value in your analysis, use the `set param` shell command to replace the line `# optionally change inversion defaults here` in the example commands in "In a standard workflow". For instance, to change recover small inversion to on, set the `inversion.recover.tiny.inversions` parameter to 1 with this command:

```
set inversion.recover.tiny.inversions 1 tertiary/inversion.ini
```

Table 50 Inversion parameter description

Parameter name	Default value	Description
Optional parameters		

Table 50 Inversion parameter description *(continued)*

Parameter name	Default value	Description
library.type	matepair	The library type for alignment data. Use when the BAM LB: header field is not populated. Allowed values: matepair, pairedend.
inversion.calculate.mp.coverage	0	Whether to calculate normal mate-pair coverage around inversion breakpoints. Allowed values: • **0**: Do not use this calculation. • **1**: Use this calculation.
inversion.number.of.chromosomes	25	The number of chromosomes used to distribute jobs on a cluster (one job per chromosome). Allowed values: Integers >= 0.
inversion.abx.score	0	The ABX score. ABX is ABA/ABB/ABC, which are mates with both tags on the correct strands but in reverse order. Allowed values: • **0**: Do not use ABX types as candidate inversions. • **1**: Use ABX types as candidate inversions.
inversion.force.update.intermediate.files	0	Whether to force updating of all intermediate files. Allowed values: • **0**: Do not force updating of all intermediate files. • **1**: Force updating of all intermediate files.
inversion.down.weight.mp.mismatches	0	Whether to down-weight mate-pairs with mismatches exponentially. Allowed values: • **0**: Do not down-weight mate-pairs with mismatches exponentially • **1**: Down-weight mate-pairs with mismatches exponentially.
inversion.max.bxx.mp.length	3000000	The maximal mapped length of BXX mate-pairs. Allowed values: Integers >= 0.
inversion.max.inversion.length	100000	The maximum mapped length of inversions. Allowed values: Integers >= 0.
inversion.min.pairing.quality	10	Minimum pairing quality threshold. Allowed values: Integers 0–100.
inversion.score.run.individually	0	Whether to score every run individually. The parameter is used when rerunning the inversions module with a different breakpoint score threshold, and reusing the inversion evidences and score calculated from previous run. Allowed values: • **0**: Do not score every run individually. • **1**: Score every run individually.
inversion.breakpoint.rescue	0	Whether to pair breakpoints with rescue. Allowed values: • **0**: Do not pair breakpoints with rescue • **1**: Pair breakpoints with rescue.
inversion.recover.tiny.inversions	0	Whether to recover small inversions. Allowed values: • **0**: Do not recover small inversions. • **1**: Recover small inversions.
inversion.max.length.tiny.inversions	0	The maximum mapped length of small inversions. Allowed values: Integers >= 0.
inversion.breakpoint.score.threshold	4	The break point score threshold. Allowed values: Integers 0–100.

Table 50 Inversion parameter description *(continued)*

Parameter name	Default value	Description
inversion.sab.gff.score.threshold	0	The output score threshold. Allowed values: Integers 0–100.
inversion.breakpoint.peak.width	100	The break point peak width. Allowed values: Integers 0–100.
Resource parameters		
memory.request	21gb	Memory request. Include the units gb in the setting.
java.heap.space	3000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.

Table 50 describes the inversion parameters that we do not recommend changing.

Table 51 Inversion internal parameter description

Parameter name	Default value	Description
inversion.run	1	Whether or not to run the inversion module. • **0**: Do not run the inversions module. • **1**: Run the inversions module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
inversion.max.anchor.mismatch	0	Filter out mates with more anchor mismatches on either tag. Allowed values: Integers 0–10.
inversion.min.alignment.length	0	Filter out mates with shorter alignment length on either tag. Allowed values: Integers 0–100.
inversion.max.alignment.start	0	Filter out mates with farther alignment start position on either tag. Allowed values: Integers 0–15.

Input files

The inversion module takes the BAM file output of the mapping module as input. Because the inversion module specifically selects for pairs that are in the incorrect orientation, it is important to know the correct orientation. As a result, either the library type information in the LB field or the `library.type` parameter must be properly set.

Libraries in the input file or files must be either:

- All mate-pair
- All paired-end

When different library types are provided as input on the same run, the inversion module fails with an error.

Read lengths

Input BAM files may contain multiple read lengths.

Inversion output files

Table 52 describes the inversion GFF output file format.

Table 52 The inversion output file format

Column title or number	Description	Example
1	Chromosome.	chr10
2	Method.	AB_SOLiD
3	Feature keywords.	inversion
4	Inversion start coordinate.	46443097
5	Inversion end coordinate.	46479578
6	Inversion score.	129.8
7	Not used.	—
8	Not used.	—
9 Attributes, *see below*:		
left‡	Starting breakpoint range.	left=chr10:46443097-46443161
right	Ending breakpoint range.	right=chr10:46479540-46479578
leftscore	Left breakpoint range score.	leftscore=185
rightscore	Right breakpoint range score.	rightscore=100
count_AAA_further_left	The number of AAA mate-pairs spanning the genomic region to the immediate left of the starting breakpoint range.	count_AAA_further_left=1
count_AAA_left	The number of AAA mate-pairs spanning the starting breakpoint range.	count_AAA_left=1
count_AAA_right	The number of AAA mate-pairs spanning the ending breakpoint range.	count_AAA_right=2
count_AAA_further_right	The number of AAA mate-pairs spanning the genomic region to the immediate right of the ending breakpoint range.	count_AAA_further_right=1
left_min_count_AAA	Sub-range within starting breakpoint range that has the minimal AAA coverage.	left_min_count_AAA=chr10:46443 097-46443112
count_AAA_min_left	Minimal AAA coverage in starting breakpoint range.	count_AAA_min_left=0
count_AAA_max_left	Maximal AAA coverage in starting breakpoint range.	count_AAA_max_left=4
right_min_count_AAA	Sub-range within ending breakpoint range that has the minimal AAA coverage.	right_min_count_AAA=chr10:4647 9576-46479578
count_AAA_min_right	Minimal AAA coverage in ending breakpoint range.	count_AAA_min_right=4
count_AAA_max_right	Maximal AAA coverage in ending breakpoint range.	count_AAA_max_right=11
homozygous	Whether AAA coverage at both breakpoints ranges are lower than one-fifth of their neighboring ranges.	homozygous=YES

‡ This and the remaining entries are allowed values for Attributes in column 9.

The following is an example of the file `inversions.s2.w100.100000.GFF`, whose format is described in Table 56 on page 233:

```
##gff-version 3
##generated by SOLiD inversion tool
chr10 AB_SOLiD    inversion   46442583    46479737    4     .     .
left=chr10:46442583-46443522;right=chr10:46479431-
46479737;leftscore=6.0;rightscore=3.0;count_AAA_further_left=0;count_AAA_left
=0;count_AAA_right=0;count_AAA_further_right=0;left_min_count_AAA=chr10:46442
583-
46443522;count_AAA_min_left=0;count_AAA_max_left=0;right_min_count_AAA=chr10:
46479431-46479737;count_AAA_min_right=0;count_AAA_max_right=0;homozygous=YES
chr21 AB_SOLiD    inversion   26295366    26297167    2.7   .     .
left=chr21:26295366-26296041;right=chr21:26296491-
26297167;leftscore=2.7;rightscore=2.7;count_AAA_further_left=0;count_AAA_left
=0;count_AAA_right=0;count_AAA_further_right=0;left_min_count_AAA=chr21:26295
366-
26296041;count_AAA_min_left=0;count_AAA_max_left=0;right_min_count_AAA=chr21:
26296491-26297167;count_AAA_min_right=0;count_AAA_max_right=0;homozygous=YES
chr6  AB_SOLiD    inversion   168834702   168837413   2.1   .     .
left=chr6:168834702-168835567;right=chr6:168836564-
168837413;leftscore=2.2;rightscore=2.1;count_AAA_further_left=0;count_AAA_lef
t=0;count_AAA_right=0;count_AAA_further_right=0;left_min_count_AAA=chr6:16883
4702-
168835567;count_AAA_min_left=0;count_AAA_max_left=0;right_min_count_AAA=chr6:
168836564-
168837413;count_AAA_min_right=0;count_AAA_max_right=0;homozygous=YES
```

Table 53 Inversion all.chr, all.chr*, pair.orphan, pair.txt file format description

Column	Description	Example
1	Display label. For internal use.	—
2	GFF name, type of breakpoint.	InvStart
3	GFF type.	exon
4	Range start coordinate.	1297
5	Range end coordinate.	1362
6	Breakpoint range score, number of supporting mate-pairs.	1
7	Strand.	-
8	—	—
9	—	g=.1
10	Chromosome.	chr1

Table 54 inversion rank.txt file format description

Column	Description	Example
1	Chromosome.	chr1
2	Inversion start coordinate.	1632874
3	Inversion end coordinate.	1706252
4	Inversion score.	4.8
5	Inversion length.	73379
6	Starting breakpoint range.	chr1:1632874-1633166
7	Ending breakpoint range.	chr1:1705726-1706252
8	Left breakpoint range score.	6.0
9	Right breakpoint range score.	4.0

Table 55 Inversion AAA GFF file format description

Column	Description	Example
1	Mate-pair bead ID.	1_11_215_288
2	GFF name.	LEFT
3	GFF type.	exon
4	Mate-pair start coordinate.	3971
5	Mate-pair end coordinate.	4020
6	Score, number of mismatches.	1
7	Strand.	+
8	—	—
9	—	g=

Table 56 Inversion coords.inversions.s*.* and coords.orphan.* file format description

Column	Description	Example
1	The genomic coordinate of an inversion.	chr10:46443097-46479578
2	Inversion score.	129.8

The following is an example of the file `coords.inversions.s2.w100.100000-`, whose format is described in Table 56:

```
chr10:46442583-46479737          4
chr21:26295366-26297167          2.7

chr6:168834702-168837413         2.1
```

Table 57 Inversion AAA/*.txt and AAA.txt file format description

Column	Description	Example
1	Chromosome.	chr10
2	Inversion start coordinate.	46443097
3	Inversion end coordinate.	46479578
4	Inversion score.	129.8
5	Inversion length.	36482
6	Starting breakpoint range.	chr10:46443097-46443161
7	Ending breakpoint range.	chr10:46479540-46479578
8	Left breakpoint range score.	185
9	Right breakpoint range score.	100
10	The number of AAA mate-pairs spanning the genomic region to the immediate left of the starting breakpoint range.	1
11	The number of AAA mate-pairs spanning the starting breakpoint range.	1
12	The number of AAA mate-pairs spanning the ending breakpoint range.	2
13	The number of AAA mate-pairs spanning the genomic region to the immediate right of the ending breakpoint range.	1

Table 58 Inversion rescore.txt file format description

Column	Description	Example
1-13	Same as AAA.txt	See Table 57.
14	Sub-range within starting breakpoint range that has the minimal AAA coverage.	chr10:46443097-46443112
15	Minimal AAA coverage in starting breakpoint range.	0
16	Maximal AAA coverage in starting breakpoint range.	4
17	Sub-range within ending breakpoint range that has the minimal AAA coverage.	chr10:46479576-46479578
18	Minimal AAA coverage in ending breakpoint range.	4
19	Maximal AAA coverage in ending breakpoint range.	11

Inversion algorithm

Overview

The inversion module exploits the large insert size of SOLiD® System mate-pair libraries to detect inversion polymorphisms that are important but often poorly characterized. The large insert size is critical for spanning the repeat regions that are associated with inversion break points.

In its simplest form, the algorithm collects evidence for an inversion by observing accumulations of pairs with correct relative positioning, but with an improper orientation. For SOLiD® System mate-pair libraries, this algorithm requires tags to be mapped to opposite strands. For example, if the R3 tag is to the left of an F3 tag, but the R3 tag maps to the top strand and the F3 tag maps to the bottom strand, this pair provides inversion evidence, specifically that the R3 tag is to the left of the inversion starting break point and that the F3 tag is to the right of the break point. Once the evidence is collected, candidate inversion break points are scored, paired, and ranked. Additional evidence is provided by a scan for drops in the coverage by normal mate-pairs (see Figure 16).

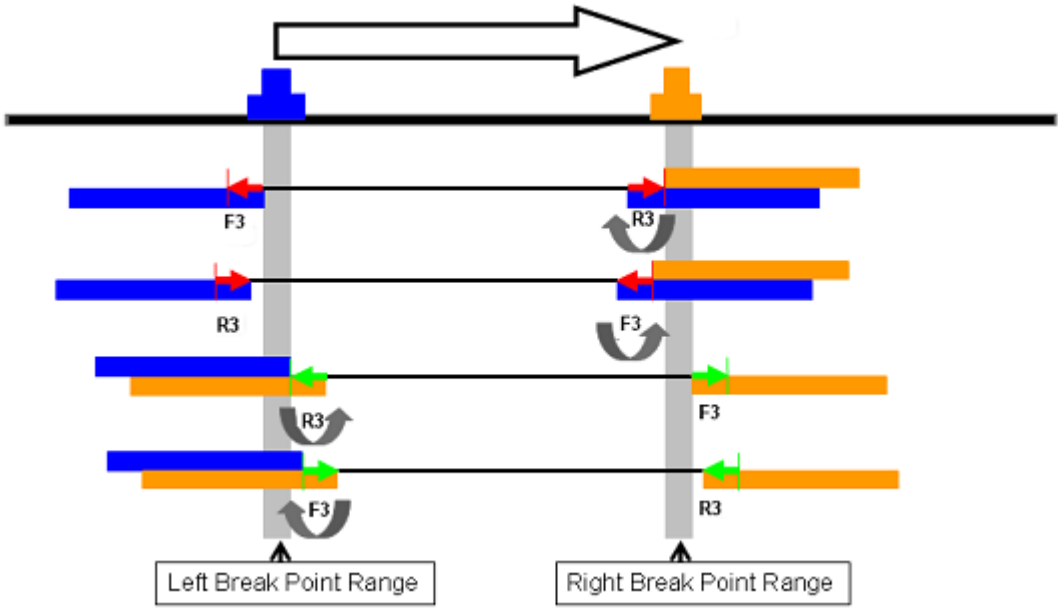


Figure 16 Inversions and break point ranges

Figure 16 depicts the following elements:

Thick black line	The sequenced genome.
Thin black lines	Mate-pairs.
Red and green arrows	Two ends of an inverted mate-pair.
Blue and orange bars	The maximum distance separating the two ends of a normal mate-pair (AAA mates).

By definition, an inversion has two breakpoints: a starting breakpoint and an ending breakpoint. The inversion module plug-in uses SOLiD® System mate files or mate-pair BAM files as input (using paired-end BAM files is not recommended due to their small insert size). The number of mate-pairs supporting an occurrence (of both starting and ending inversion breakpoints) are counted for each base pair as breakpoint scores. Candidate breakpoint ranges are genomic ranges corresponding to local peaks of counts above a predetermined score threshold. The clone insert size constraints specify a minimum and a maximum (blue and orange horizontal bars) distance separating the two ends (small green or red arrows) of a mate-pair (thin black lines) in the sequenced

genome (thick black line). Each green mate-pair suggests a starting breakpoint of an inversion occurring to the left, and an ending breakpoint between its two tags. The green pairs then define the range of the starting breakpoint by contributing positive counts to all base pairs to the left of their left tags within 1 max. The green pairs also help refine the ending breakpoint range by contributing negative counts to all base pairs 1 max to the left and 1 max to the right. The red mate-pairs contribute counts in a similar fashion.

When small inversion detection is enabled (with the parameter key `inversions.recover.tiny.inversions`), a second pass of the algorithm is performed, but with a smaller window for the scoring function, allowing the detection of inversions of 200 base pairs, or even lower, depending on coverage.

Like other SOLiD® System modules, the inversion module outputs results in GFF format for easy viewing in common genome browsers. Because of the inherently ambiguous nature of the detection, the start and end locations represent the widest possible range. More details are provided by the GFF attributes, including the break point ranges and scores.

While the same algorithm can be applied to SOLiD® System paired-end libraries, in practice, the smaller insert size makes inversions more difficult to detect. Without the large insert size of mate-pair libraries, both tags would place into repeat regions that typically flank inversions, making tag placement ambiguous.

Input data

The first step in the detection of inversions is the collection of inverted mates. These are filtered from the BAM file by selecting for mates of opposite orientation. In pairing category terms, this filtering includes BA*, BB*, and AB*. These records are written to an intermediate directory, so that the filtered set can be reused. It is important that either the `library.type` parameter be set or that the LB field be properly constructed to indicate a mate-pair library.

Non-redundant mates are selected from BAM file input using the PCR duplicates flag (`0x0400`). Pairing quality values can be used to select unique records via the `inversion.min.pairing.quality` parameter key. Values greater than 20 should be unique, but criteria may be different for different applications.

The Pairing “SV” output filter is specifically designed to capture more of the deviant pairs used for structural variant programs such as the inversion module. A lower `inversion.min.qv` value (~10) and a BAM file generated with the `"SV"` output filter can potentially find a larger number of candidate inversions.

Workflow

Figure 17 shows the inversion module workflow. The inversion module begins with a long mate-pair BAM file and proceeds through scoring, pairing, ranking, rescoring, and GFF output.

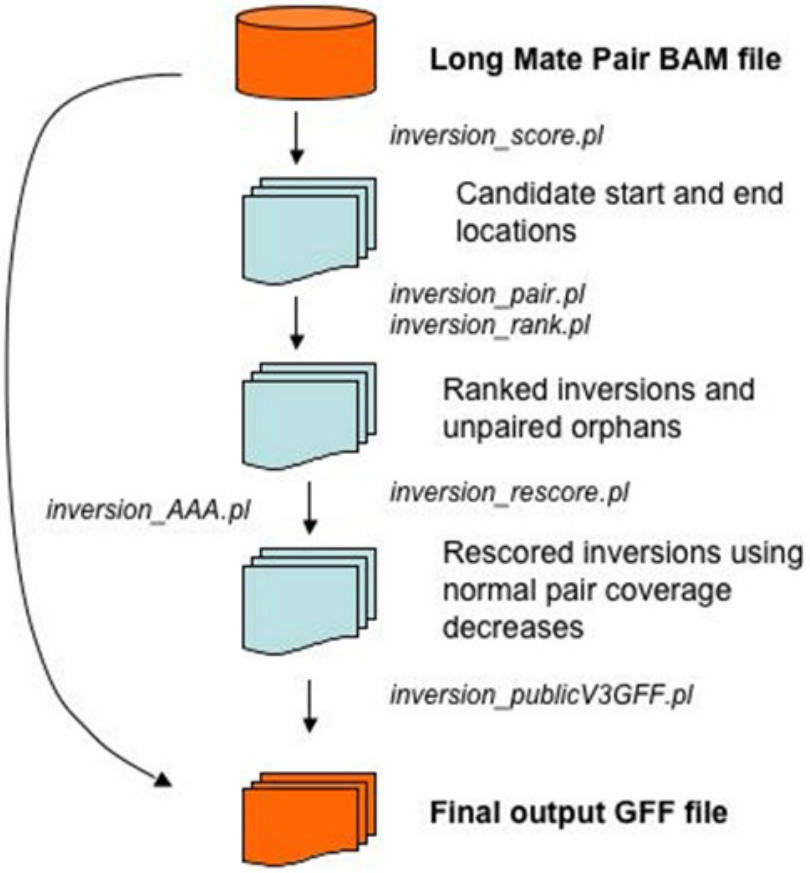


Figure 17 Inversion module algorithm workflow

Scoring

Candidate breakpoints are scored by first determining the tag of a pair that is properly oriented. The inverted mate of the pair is then used to select reference locations that are in the vicinity of the inversion breakpoint. All reference locations that are between the inverted mate and the maximum possible insert size accumulate a positive score, resulting in a list of high scoring locations that represent candidate breakpoints.

Pairing

The high scoring candidate breakpoints are then recombined into inversion candidates by matching start and end locations that are defined by an analysis of the high scoring locations. A user-definable window size (`inversion.breakpoint.peak.width`) is used to determine regions with a peak score greater than the threshold (`inversion.breakpoint.score.threshold`). These peaks are combined with their reciprocal nearest neighbor to create a set of possible inversions.

In some cases there will be a peak whose nearest neighbor is slightly below the threshold. A rescue analysis will retrieve those that are significant peaks, but below the threshold cutoff. This is controlled by the `inversion.breakpoint.rescue` flag.

A correct threshold for inversion breakpoint scoring depends on clone coverage of sequencing data, its nature, and its mapping quality. The default value 4 is set for 100–500X clone coverage of sequencing from a normal genome. As clone coverage increases, the inversion score threshold must be raised to filter out mismapping noise. Because more inversions can be expected in cancer samples, the threshold may be lowered when running on those samples.

Ranking

Inversions are ranked based on the harmonic mean of the scores of the start and end breakpoints. Additionally, inversions are separated out by the user specified maximum inversion length (`inversion.max.inversion.length`).

Inversion length thresholds

Table 59 describes the how the parameters `inversion.max.inversion.length` and `inversion.max.bxx.mp.length` affect inversion calling. The parameter `inversion.max.inversion.length` also affects the output file name, according to these patterns:

- `coords.inversions.s4.w100.<max.inversion.length>-`
- `coords.inversions.s4.w100.<max.inversion.length>+`

The example output file names in this table are based on the parameter setting `inversion.max.inversion.length=100000`.

Table 59 Inversion threshold parameters

Length	Inversion calling behavior
From 1 to inversion.max.inversion.length	Inversions are called, into an output file named, for example, coords.inversions.s4.w100.100000-. **Note:** The minus sign ("-") is part of the file name.
From inversion.max.inversion.length to inversion.max.bxx.mp.length	Inversions are called, into an output file named, for example, coords.inversions.s4.w100.100000+. **Note:** The plus sign ("+") is part of the file name.
Greater then inversion.max.bxx.mp.length	Inversions are not called.

Tiny inversions

If the tiny inversions flag is set (`inversion.recover.tiny.inversions`), the pairing and ranking components are rerun with a window size that is sufficiently small to detect inversions with a smaller size (controlled by the parameter `inversion.max.length.tiny.inversions`).

Normal pair coverage

Additional evidence is provided for breakpoints by examining the relative coverage of unique proper pairs. These pairs are selected from the BAM input using the proper pair flag (0x0002), and they are used to reduce the score of reference positions covered by proper pairs.

Run a Small Indels Analysis

This chapter covers:

- Overview 239
- Examples of running a small indels analysis 240
- Small indel parameter description 241
- Small indel module output file formats 245
- Gap alignment mapping 256
- Small indel calling from gap alignments 260
- FAQ – Small indels 272

Overview

Small indel detection is a tertiary module in LifeScope™ Genomic Analysis Software. Data in BAM format from the fragment and paired mapping modules serve as direct inputs for small indel discovery. Gap alignments, those with a CIGAR string containing `D` or `I`, serve as the primary evidence for this. In the `second.map` part of the mapping module, these evidences are found by taking partial read anchor alignments, and then either extending allowing for a gap, called LOCAL alignment, or as seed placement for a full read, called GLOBAL, very similar to what is done in the pairing phase. Gap alignments are also found separately in the pairing phase of the mapping module. Here, these evidences are found in mate pairs by using the placement of one tag such that the other tag can be used to search for the presence of a small indel. Specifically, with the unplaced tags, the algorithm finds all possible placements of the beginning and ends of the read. Then, these are joined together only if the resulting gap size is within the limits set in the pairing module. All successful joinings form the set of gap alignments.

The LifeScope™ Software small indel module allows flexible processing of these gap alignments. LifeScope™ Software takes input gap alignments from one or more input BAM files and collects them (based on their proximity from each other) to form pileups. The module, in the process of making the pileups or in evaluating these pileups, employs a number of heuristics, including map QV filtering, insertion sequence determination, and assessment of zygosity using ungapped alignment read counting and clustering of gap sizes. In the end, the small indel module produces quality indel calls in a public GFF version 3 format.

Examples of running a small indels analysis

In a standard workflow

The following standard workflows provide examples of running the small indels module:

- genomic.resequencing.frag
- genomic.resequencing.lmp
- genomic.resequencing.pe
- targeted.resequencing.frag
- targeted.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk proj1
cd proj1
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.pe
# define the input (this example uses dummy XSQ filenames)
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
# specify the reference to be used
set reference hg19
# optionally change parameter defaults here
# list the analysis configuration
ls
# run the analysis
run
# list progress information for the analysis
ls
```

In order to change a parameter value in your analysis, add a `set param` shell command to after the line `# optionally change parameter defaults here`. For instance, to change the filtering parameter, use this shell command:

```
set small.indel.perform.filtering 0  tertiary/small.indel.ini
```

See for Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information on running standard workflows. Chapter 6, "Run a Command Shell Analysis" on page 83 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an sample of how to run the small indel module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, "Demo Analyses" on page 567 for information on running the small indel module as a standalone analysis.

Small indel parameter description

Table 60 describes a workflow parameter, `analysis.space`, that is used in the small indel module. The `analysis.space` parameter is set in the `tertiary/global.ini` file.

Table 60 Analysis space parameter used in the small indel module

Parameter name	Default	Description
analysis.space	auto	Specifies your *preference* for base space or color space analysis. Allowed values: • **auto**: The analysis space is determined based on the mapping space and the presence of color space data in the analysis BAM files. • **base**: Prefer base space analysis. • **color**: Prefer color space analysis. See "Analysis space" on page 77 for a description of other factors determining the analysis space used by the small indel module.

Table 61 describes the small indel module parameters.

Table 61 Small indel module file parameters

Parameter name	Default value	Description
Optional input parameters		
analysis.regions.file	—	BED format file indicating the target regions for the analysis. Used for targeted analysis. The parameter is managed by the framework. In targeted resequencing and low frequency variant workflows, this parameter is set in the both secondary and tertiary global.ini files.
General options		
small.indel.detail.level	3	For BAM file inputs, the level of detail in output: • **0**: Keeps only position information about the anchor read and no information for the ungapped alignment. • **1–8:** Keeps only some of the alignment's anchor alignment but none of the ungapped alignment. • **9**: Is most detailed, but also the slowest.
small.indel.zygosity.profile.name	max-mapping	Zygosity profile name. • **classic**: Profile for classically (full read) mapped reads. • **max-mapping**: Profile for seed and extend ungapped alignments. • **max-mapping-v2**: Reserved for future use. • **gap-align-only**, and **no-calls**: Force all zygosity calls to be homozygous calls.

Table 61 Small indel module file parameters *(continued)*

Parameter name	Default value	Description
small.indel.genomic.region	—	Names a specific genomic region to be selected from the BAM file. Only full chromosomes are guaranteed not to alter results. Specifying partial chromosomes is allowed but may result in the loss of indels near the edges of that region. For example, chr1:2945-9659 causes a reduction of coverage for approximately a read length after position 2945.
small.indel.display.base.qvs	0	Whether or not to display base quality value scores in the GFF file. Allowed values: **0**: Do not add QV scores in the output file. **1**: Displays the FASTQ base QV scores for all of the reads used for each indel in the GFF file. FASTQ strings contain semi-colons, so adding these strings may produce a GFF file that is not compatible with certain applications.
small.indel.num.aligns.per.pileup	1000	For pileups with more than this number of alignments, sets the expected number of alignments per pileup. Values much higher than 1000 may result in a significant increase in computational time. After position 9659, indels may be called with partial coverage, with reads contributing to this coverage which also start at position 9659 and earlier. Allowed values: Integers >= 0.
small.indel.random.seed	94404	The random seed value used to determine which pseudo random set of reads to use when there are greater than 1000 reads in a pileup. The random number generator used is the Mersenne Twister MT19937 algorithm**.** Allowed values: Integers >= 0.
Pileup options		
small.indel.min.num.evid	2	Minimum number of evidences required for an indel call. This parameter does not have an upper limit, but a value higher than the default of 2, even if it is lower than the average coverage level, may cause a significant reduction in sensitivity.
small.indel.max.num.evid	–1	Maximum number of evidences. Use –1 for no maximum. Setting this value to some multiple of the average coverage could remove indels found in abnormally high coverage areas.
small.indel.consGroup	1	Indel grouping method. Allowed values: **1**: Conservative grouping of indels with 5bp max between consecutive evidences. **2**: Lax grouping. Groups indels that are at maximum the higher of 15 or 7 times the indel size. **9**: No grouping. Makes every indel evidence a separate pileup.
Mapping quality filtering		

Table 61 Small indel module file parameters *(continued)*

Parameter name	Default value	Description
small.indel.max.reported.alignments	–1	Only uses those alignments where the NH field (the number of reported alignments; from the BAM record) is this value or lower. A value of –1 is to have no upper limit. The range where this is effective depends on the input BAM file's range of values of the NH tag.
small.indel.min.mapping.quality	8	Keeps only reads that have this or higher pairing qualities. For paired tags, mapping quality is for the pair (pairing quality), and for fragment, it is the single tag's map quality. Reads that are lower than this value are filtered out. Allowed values: Integers -1 to 256.
small.indel.min.best.mapping.quality	10	For a particular indel called with a set of reads, at least one pairing quality in this set must be higher than this value. Allows for supporting evidences to have a lower mapping quality threshold then the best read. Allowed values: Integers -1 to 256.
small.indel.min.anchor.mapping.quality	–1	Minimum mapping quality for a non-indel (anchor) tag. Effective only for paired reads, for the number of anchors queried as defined by small.indel.detail.level. Uses the SM tag of the BAM record from the mate alignment. Allowed values: Integers -1 to 256. **Note:** The BAM spec allows values 0—255. LifeScope™ Software mapping uses values in the range 0—100. A value of -1 or 0 filters no reads, and 256 filters all reads.
small.indel.ungapped.bam.flag.filter	ProperPair,Primary	For ungapped alignments, specifies the BAM flag properties that a read must have to be included. Allowed values: A comma-separated string of one or more of these values: • ProperPair • UniqueHit (requires the BAM NH field to be set to 1) • NoOptDup • Primary • None If None is present in the string, then all filters, even those explicitly set, are turned off.
small.indel.gapped.bam.flag.filter	Primary	For gapped alignments, specifies the BAM flag properties that a read must have to be included. Allowed values: A comma-separated string of one or more of these values: • ProperPair • UniqueHit (requires the BAM NH field to be set to 1) • NoOptDup • Primary • None If None is present in the string, then all filters, even those explicitly set, are turned off.

Table 61 Small indel module file parameters *(continued)*

Parameter name	Default value	Description
small.indel.edge.length.deletions	0	Gap alignments that do not have this minimum length on either side of the indel will not be considered. Allowed values: Integers >= 0.
small.indel.edge.length.insertions	0	Gap alignments that do not have this minimum length on either side of the indel will not be considered. Allowed values: Integers >= 0.
Heuristic filtering		
small.indel.perform.filtering	1	Whether or not to perform filtering in each pileup. Allowed values: • **0**: Do not perform filtering on pileups. • **1**: Perform heuristic filtering on pileups. Parameters (such as small.indel.min.num.evid) that change the makeup of pileups contained in the PAS.SUM file are still active.
small.indel.indel.size.distribution.allowed	can-cluster	Indel sizes in a pileup are allowed to have certain indel size distributions. Allowed values: • **similar-size**: 75% of the reads of a pileup must have exactly the same size. • **similar-size-any-large-deletions**: Any pileups with at least 2 large deletion alignments, the other pileups must have similar sizes. • **can-cluster**: Allowed if at least one cluster of any indel size is found. [Default]. • **can-cluster-any-large-deletions**: Any pileups with at least 2 large deletion alignments; other pileups must be able to cluster (will have indels with two more reads with larger deletions, even if they don't form good clusters). • **any**: Can have any size distribution (same as small.indel.require.called.indel.size=false).
small.indel.remove.singletons	1	Whether or not to remove the singletons that occur when different alignment methods are combined based on identical bead ids and read sequence. Allowed values: • **0**: Do not remove the singletons. • **1**: Remove singletons.
small.indel.alignment.compatibility.filter	1	Alignment compatibility level. Checks color space compatibility around the gap. Allowed values: • **0**: Disable the check. • **1**: Enable the check.
small.indel.max.coverage.ratio	12	Maximum allowed value for the ratio of number of reference alignments allowed by the ungapped.bam.flag.filter over the number of non-redundant indel variant reads selected with the gapped.bam.flag.filter. Use –1 for no limit (no coverage ratio filtering). Allowed values: Integers.

Table 61 Small indel module file parameters *(continued)*

Parameter name	Default value	Description
small.indel.max.nonreds-4filt	2	Maximum number of non-redundant reads where read position filtering is applied. Allowed values: Integers.
small.indel.min.from.end.pos	9.1	Minimum average number of base pairs from the end of the read required of the pileup, when there are at most a certain number of reads defined by small.indel.max.nonreds-4filt. Allowed values: Floats.
Indel size filtering		
small.indel.min.insertion.size	0	Minimum insertion size to include. Allowed values: Integers.
small.indel.min.deletion.size	0	Minimum deletion size to include. Allowed values: Integers.
small.indel.max.insertion.size	1000000000	Maximum insertion size to include. Allowed values: Integers.
small.indel.max.deletion.size	1000000000	Maximum deletion size to include. Allowed values: Integers.
Resource parameters		
memory.request	2gb	Memory request. Include the units gb in the setting.
java.heap.space	30000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.
maximum.workers	24	Maximum number of nodes that can be used for this analysis.
processors.per.node	1	On a node, the number of parallel threads used to run this process.
wall.time	336	The maximum total time (in hours) allowed for the process to complete. The default is equivalent to 14 days.

Small indel module output file formats

The main output file of this module is the produced GFF_3 file. This list describes the files generated by the small indel module:

- **`.gff`** – Contains the resulting small indel calls with extensive annotation in the attribute column.
- **`.pas.sum`** – An internal pileup format which represents collections of gap alignments extracted from the BAM file.
- **`.align`** – Displays the alignments of reads of the gaps in a human readable format.
- **`.ungapped`** – For each pileup, contains the list of beads ids from reads that are aligned without gaps but also span the location of the indel.
- **`.pas`** – A legacy file produced but not used by the caller.
- **`.txt`**, **`.sql`** – Alternate output format that contain only a subset of the information contained in the GFF file.

Small indel GFF format

As shown in the small indel output file example below (after Table 62), the GFF file created by the small indel module begins with the General File Format Version 3 headers. The section following the GFF headers displays BAM header and read group information. The lines containing information about each indel follows. Table 62 describes each column and attribute contained in the file. Table 63 on page 252 describes the text format, which contains similar information as the GFF file.

Table 62 Small indel GFF file format descriptions

Column/Attribute tag name	Description	Example
seqid	The ID of the sequence to which the start and end coordinates refer, such as a chromosome number.	chrV
source	Free-text qualifier that indicates the algorithm or method that generated the feature.	AB_SOLiD Small Indel Tool
type	SOFA feature. For indels, possible values are: • insertion_site • deletion • combination	deletion
start	Start position of the feature.	200587060
end	End position of the feature. 1-based integer coordinates of the feature, relative to the sequence in column 1. For zero-length features, such as insertion sites, "start" equals "end" and the implied site is to the right of the indicated base in the direction of the landmark. For deletions, the start and end indicate the positions in the reference that are not present in the sample.	200587063
score	Floating-point value representing the quality of the evidence for the feature. "Score" is not currently calculated, so all indels are reported with a score of 1.	1
strand	The strand of the feature. The type of indels detected are not stranded, because they are found with sequence reads that are on either or both strands.	.
phase	Translation frame; relevant only for CDS features.	.
Attributes		
ID	Unique indel id. Non-sequential ids are due to filtering.	21
ins_len	Number of bases inserted relative to the reference.	7
del_len	Number of bases missing from the reference.	2
len	The size of the indel, if both an insertion and a deletion are detected. Deletions are indicated by a negative value.	7, –2
clustered-indel-sizes	The average indel size of each cluster rounded to the nearest integer. Deletions are indicated by a negative value.	4, –298

Table 62 Small indel GFF file format descriptions *(continued)*

Column/Attribute tag name	Description	Example
reference	The reference sequence at the position of the allele call. An event that is an insertion and is not a substitution, is indicated with a -.	ag
allele-call	The allele sequences detected. The reference sequence is only present if sufficient reads span the first breakpoint of the indel. The string ag/- indicates a hemizygous deletion where one allele is the reference, while -/- indicates a homozygous deletion. A third allele is possible if sufficient reads indicate its presence.	ag/
context-pos	Position of the first base of the sequences reported with additional surrounding sequence context. If the first allele is missing, it is the position immediately before the insertion.	713660
context-reference-seq (ref part of alleles in v1.3)	Reference sequence at the context position.	acagagagaag
context-variant-seq (var part of alleles in 1.3)	A complete list of non-reference variant sequences found by all gap alignments in the pileup. Using the ambiguity information from the aligner (XA tag), this contains the context around the indel representing a possible short tandem repeat. "NO_CALL" indicates reads where a misalignment is likely (that is, color-space alignment is not in agreement with base-space alignment).	aCAGAGAAG/ acagagagagaag
reference-reads	Number of reads detected to span the first indel breakpoint location by 5bp.	5
context-variant-reads (was allele-counts in v1.3)	The number of indel reads found for each of the above variants. If the number of gap alignments found is greater than the parameter small.indel.num.aligns.per.pileup, then an estimated counts is displayed followed by the read counts of the sample in parentheses.	3,1 15464,1497(899,87)
gap-total-reads	Total number of gap alignments in the pileup.	4
gap-nonred-reads	Total number of reads that are not marked as PCR duplicates in the BAM file.	4
tight_chrom_pos	Conservative estimate of the chromosome start position range of the feature.	713662–713667
loose_chrom_pos	Estimate of the maximum chromosome start position range of the feature.	713662–713667
no_nonred_reads	Number of reads with unique start positions (non-redundant reads).	3

Table 62 Small indel GFF file format descriptions *(continued)*

Column/Attribute tag name	Description	Example
coverage_ratio	Clipped ungapped coverage divided by the number of non-redundant gap alignment reads. Clipped coverage means the last 5 bp at either end are not counted as a part of coverage.	1.6667
zygosity (v1.3 name: experimental-zygosity)	Experimental zygosity call. Allowed values: • **HEMIZYGOUS-RE**F: enough ungapped alignments overlap the indel break point to indicate that the reference allele is present. Also, all the indels have the same size and the same sequence. • **HOMOZYGOUS-NON-REF**: occurs when there is no indication that the reference allele is present. Also, all the indels have the same size and the same sequence. • **HEMIZYGOUS-REF-MULTI-INDEL-ALLELE**: occurs if the reference is detected, the variant has a single clustered indel size, but multiple variant sequences are detected. • **HETEROZYGOUS-NON-REF**: occurs if the reference is not detected, the variant has a single clustered indel size, but multiple variant sequences are detected. • **MULTI-HEMIZYGOUS-REF**: occurs if the reference, multiple clustered indel sizes, and multiple variant sequences are all detected. • **HEMIZYGOUS-NON-REF**: occurs if the reference is not detected, but multiple clustered indel sizes and multiple variant sequences are.	HEMIZYGOUS-REF
zygosity-score (v1.3 name: experimental-zygosity-score)	Experimental zygosity score. The range is 0–1. The closer to 1, the more likely the zygosity is hemizygous with the reference allele, and the closer to 0, the less likely that it is.	0.9656
rg_id_nums (was contained in run_names in v1.3)	The list of runIdNum's that correspond to the ##@HD header line of the GFF file. This information indicates which input BAM file the particular read came from.	1,1,2,1
bead_ids	Bead IDs for each read. The run_names and all the comma-separated attributes below are in the same order as these.	984_536_1054, 1431_2007_1567, 116_364_1582
overall_qvs	Alignment quality values for each bead.	26,47,66
no_mismatches	List of number of mismatches for each read.	–1,–1,–1
read_pos	Position in each read at which there was a gap in the alignment.	20,17,10
from_end_pos	Same as above, except that the value is the number of base pairs from the end of the read.	30,33,40

Table 62 Small indel GFF file format descriptions *(continued)*

Column/Attribute tag name	Description	Example
strands	Strand for each read.	+,+,+
tags	Tags where the indel was found. Possible values are F3, R3, and FRAG.	F3,F3,F3
indel_sizes	List of indel sizes found for each evidence.	–2,–2,–2
non_indel_no_ mismatches	Number of mismatches of the other tag that was matched without a gap. Values of NIL occur if that particular bead-id is from a fragment analysis.	1,3,3
unmatched-lengths	For a particular bead-id, the length of the read that was left unmatched (equal to the read length if no ungapped match was found). This read length is relevant in extended read alignments.	50,50,50
ave-unmatched	Average of the unmatched lengths.	50
anchor-match-lengths	Anchor tag's match lengths in extended read alignments. If the detail level is not set to 9, some of these values may be reported as 99.	49,44,49
ave-anchor-length	Average of the anchor match lengths.	47.3333
read_seqs	The read sequence where the indel was found for each bead.	T302000, T120320, T232132
base_qvs	The color call QVs for the read sequence where the indel was found. This value is not currently displayed.	—
non_indel_seqs	Sequences of the non-indel anchor tag.	G100220, G313000, G203330
non_indel_qvs	The color call QVs for the non-indel anchor tag sequence. This value is not currently displayed.	—

Example output file

A composite example of a small indel output file is shown below. The file contents are as follows:

1. The general GFF version 3 headers.
2. The BAM header and read group information.
3. Lines containing information about each indel call.

Table 62 on page 246 describes each tag of the attributes column. The following is an example header:

```
##gff-version 3
##solid-gff-version 0.3
##source-version LifeScope(tm) 2.0 SmallIndel
##other-source-version small-indel-tool.pl/splitread-pileup/process-small-indels v 1.4.0,
2011-03-11 17:23:21
##type DNA
##date 2011-04-09
##time 21:16:33
##feature-ontology http://song.cvs.sourceforge.net/*checkout*/song/ontology/
sofa.obo?revision=1.141
```

```
##options reference=/reference/human_hg18.fa output.dir=. output.prefix=baseFilename
bam.file=output/bam/bamA.bam,output/bam/bamB.bam,output/bam/bamC.bam
##reference /reference/human_hg18.fa
##input-files output/bam/bamA.bam,output/bam/bamB.bam,output/bam/bamC.bam
##run-path /path/of/ini/ini-filename/
##genomic-region chr1
##Filter-settings: max-ave-read-pos=none,min-ave-from-end-pos=9.1,max-nonreds-4filt=2,min-
insertion-size=none,min-deletion-size=none,max-insertion-size=none,max-deletion-
size=none,indel-size-distribution-allowed=CAN-CLUSTER,max-coverage-ratio=12,min-mapping-
quality=none,min-best-mapping-quality=5,remove-singletons?=T
##BAM header:
##CODE_VERSION splitread-pileup 1.4.0
##BAM-input='output/bam/bamA.bam' bam_file_num:1
##@HD VN:1.0    SO:2
##@RG rg_id_num:1       LT:MP   ID:20100629040351402    SM:HuRef        LB:50x50MP
PU:bioscope-pairing     PI:1645 DT:2010-06-28T21:03:51-0700     PL:SOLiD
##@SQ SN:chr1   LN:247249719    UR:file:/reference/human_hg18.fa
...
##@SQ SN:chrM   LN:16571        UR:file:/reference/human_hg18.fa
##BAM-input='output/bam/bamB.bam' bam_file_num:2
##@HD_ VN:1.0   SO:2
##@RG_ rg_id_num:2      LT:MP   ID:20100617182840485    SM:HuRef        LB:50x50MP
PU:bioscope-pairing     PI:1575 DT:2010-06-17T11:28:40-0700     PL:SOLiD
##@SQ SN:chr1  LN:247249719    UR:file:/reference/human_hg18.fa
...
##@SQ SN:chrM   LN:16571        UR:file:/reference/human_hg18.fa
##BAM-input='output/bam/bamC.bam' bam_file_num:3
##@HD__ VN:1.0  SO:2
##@RG__ rg_id_num:3     LT:MP   ID:20100627170940231    SM:HuRef        LB:50x50MP
PU:bioscope-pairing     PI:1645 DT:2010-06-27T10:09:40-0700     PL:SOLiD
##@SQ SN:chr1  LN:247249719    UR:file:/reference/human_hg18.fa
...
##@SQ SN:chrM   LN:16571        UR:file:/reference/human_hg18.fa

##Annotation Apr 13, 2011; LifeScope_resources//lifetech/hg18/GTF/refGene.20090513.gtf;
LifeScope_resources/lifetech/hg18/dbSNP/dbSNP_b130.tab
##dbSNP function codes:  Locus region = 1; Coding = 2; Coding-synon = 3; Coding-nonsynon = 4;
mRNA-UTR = 5; Intron = 6; Splice-site = 7; Contig-reference = 8; Coding-exception = 9;
NearGene-3 = 13; NearGene-5 = 15; Codi
ng-nonsynonymous nonsense = 41; Coding-nonsynonymous missense = 42; Coding-nonsynonymous
frameshift = 44; Coding-nonsynonymous cds-indel = 45; UTR-3 = 53; UTR-5 = 55; Splice-3 = 73;
Splice-3 = 75.
##Filtering no filtering.
#Chr    Source  Type    Pos_Start       Pos_End Score   Strand  Phase   Attributes
```

Base space example:

```
chr9    AB_SOLiD Small Indel Tool       insertion_site  127466840       127466840       1.0000
.       .       ID=37171;ins_len=15;clustered-indel-sizes=15;allele-call-pos=127466840;refere
nce=aggaaaa;allele-call=aggaaaa/TAGGGCC/AGGGCCTCTAAAAAAAGGAAAA;rightmost-allele-call-pos=127
466840;rightmost-reference=aggaaaa;rightm o st-allele-call=aggaaaa/TAGGGCC/AGGGCCTCTAAAAAAAG
GAAAA;context-pos=127466839;context-reference-seq=aaggaaaa;context-variant-seq=ATAGGGCC/AAGG
GCCTCTAAAAAAAGGAAAA/AAGCTTCTCTAAAAAAAGGAAAA/NO_CALL/GATCTAAAAAAAAAAAAAAAAAAAGGAAAA;reference-
reads=13;context-variant-reads=1;context-variant-nonPCRdup-reads=1;gap-total-reads=5;gap-non
red-reads=3;gap-nonPCRdup-reads=5;tight_chrom_pos=none;loose_chrom_pos=127466838-127466846;c
overage_ratio=4.3333;zygosity=HEMIZYGOUS-REF-MULTI-INDEL-ALLELE;zygosity-score=1.0000;rg_id_
nums=1_1;bead_ids=705_312_1080,585_1149_1247,486_211_1170,674_1449_36,1453_413_82;overall_qv
s=100,78,82,22;no_mismatches=3,4;read_pos=24,29,27,26,39;from_end_pos=36,31,33,34,21;strands
```

```
=-;tags=R3,F3;indel_sizes=21,15,-369;non_indel_no_mismatches=0,1;unmatched-lengths=0;ave-unm
atched=0.0000;anchor-match-lengths=59;ave-anchor-length=59.0000;read_seqs=AAACAAGGCAAGAAAAAG
AAAAGGAAAAAAAAAAAAAAAAAAAATCTAGAAACAAAAANAAAA,ACAAGAGAACAAGGCAAGAAAAAGAAAAGGAAAAAAATCTCCCNNNAA
AAAAAAAAAAAA,ACAAGAGAACAAGGCAAGAAAAAGAAAAGGAAAAAAATCTCTTCGAAAAGAAAGACCAAAA,ACAAGAGAACAAGGCA
AGAAAAAGAAAAGGAAAAAAATCTCCGGGAAAAAAAAAAAAAAAA,AAATTTTTTGTAAAAATCATCTCCGTCCCAAAACAGTACAACCGGG
ATAAAAAAAAAAAAA;base_qvs=;non_indel_seqs=A221203132333211111320000213300232123123303030030
2012032330,A030011330311130122031001300310313102220203333230120130200000,A333030221113212232
223202202123113120320222202210220210232021;non_indel_qvs=;rsID=72266485,72010320,56370851;fu
nctionCode=72266485:MAPKAP1:6,72010320:MAPKAP1:6,56370851:MAPKAP1:6
```

Color space example:

```
chr1    AB_SOLiD Small Indel Tool       deletion        4865598 4865605 1.0000  .       .
ID=3743;del_len=8;clustered-indel-sizes=-8;allele-call-pos=4865598;reference=CTCCACAGC;allel
e-call=CTCCACAGC/A;rightmost-allele-call-pos=4865598;rightmost-reference=CTCCACAGC;rightmost
-allele-call=CTCCACAGC/A;context-pos=4865596;context-reference-seq=GGCTCCACAGCT;context-vari
ant-seq=GGAT;reference-reads=17;context-variant-reads=8;context-variant-nonPCRdup-reads=8;ga
p-total-reads=8;gap-nonred-reads=7;gap-nonPCRdup-reads=8;tight_chrom_pos=4865598-4865598;loo
se_chrom_pos=4865598-4865598;coverage_ratio=2.4286;zygosity=HEMIZYGOUS-REF;zygosity-score=0.
9450;rg_id_nums=3_3,4_4,1_1,5_5,2_2;bead_ids=1043_1773_368,1779_878_646,979_899_1763,1753_11
03_184,1838_1695_1151,650_1885_304,1389_2001_1593,2219_1326_1756;overall_qvs=72,51,54,36;no_
mismatches=4,5;read_pos=11,20,21,22,32;from_end_pos=39,30,29,28,18;strands=-;tags=F3,R3;inde
l_sizes=-8;non_indel_no_mismatches=0,-1;unmatched-lengths=0;ave-unmatched=0.0000;anchor-matc
h-lengths=49,99;ave-anchor-length=67.7500;read_seqs=T020001220113200201111011100111122220003
23100111220,T10102101132000122011320020111101110011112222000323,G000102101132000122011320020
111101110011112122000 32,G000102101132000122011320020111101110011112222000 32,T320010210113200
012201132002011110111001111222200 03,T223020211002001021011320001220113200201111011100 11;base
_qvs=;non_indel_seqs=G00100020131231013223231211310101000313020212312010,,T11131231001013213
121331131333102331130110010213111,G021310223001021022011231222301022011000320001220 22,G20211
023021003013123202010031022220210123101021322;non_indel_qvs=;rsID=34028970
```

Interpreted example output

This section gives examples of how a few output snippets are interpreted.

Example 1: A two-base insertion

The output below shows that an insertion of GT occurs after position 32658062:

```
chr1 AB_SOLiD Small Indel Tool insertion_site 32658062 32658062 1.0000 . .
ID=4;ins_len=2;clu stered-indel-sizes=2;allele-call-pos=32658062;reference=-;allele-call=GT/
GT;rightmost-allele -call-pos=32658062;rightmost-reference=-;rightmost-allele-call=GT/GT;con
text-pos=32658060;co ntext-reference-seq=GGG;context-variant-seq=GGGGT;reference-reads=0;con
text-variant-reads=2; context-variant-nonPCRdup-reads=2;gap-total-reads=2;gap-nonred-reads=
2;gap-nonPCRdup-reads=2 ;tight_chrom_pos=32658059-32658062;loose_chrom_pos=32658059-32658062
;coverage_ratio=0.0000; zygosity=HOMOZYGOUS-NON-REF;zygosity-score=0.0039;rg_id_nums=1_1;bea
```

Example 2: A four-base insertion replaces one reference base

The output below shows that an insertion ATCT replaces the reference base G at position 34725658. Note the following fields:

- The field `ins_len=3` shows a net gain of 3 bases (4 bases inserted, 1 replaced).
- The field `reference=G` shows the reference base that is lost due to the insertion.

```
chr6 AB_SOLiD Small Indel Tool insertion_site 34725658 34725658 1.0000 . .
```

```
ID=151;ins_len=3;clustered-indel-sizes=3;allele-call-pos=34725658;reference=G;allele-call=
ATCT/ATCT;rightmost-allele-call-pos=34725658;rightmost-reference=G;rightmost-allele-call=
ATCT/ATCT;context-pos=34725658;context-reference-seq=GTCTT;context-variant-seq=ATCTTCTT;ref
erence-reads=1;context-variant-reads=6;context-variant-nonPCRdup-reads=6;gap-total-reads=6;g
ap-nonred-reads=6;gap-nonPCRdup-reads=6;tight_chrom_pos=none;loose_chrom_pos=34725657-34725
662;coverage_ratio=0.1667;zygosity=HOMOZYGOUS-NON-REF;zygosity-score=0.0013;
```

Example 3: A heterozygous insertion

The reference sequence AGGGGG at position 7002407 has a single base insertion of A after position 7002412. The other allele of this heterozygous indel is GAAAAAA at position 7002407. Note the following fields:

- The field `ins_len=1` shows one base is inserted.
- The field `allele-call=AGGGGGA/GAAAAAA` shows that this insertion is heterozygous.
- Comparing the reference field with the first strand of the allele call shows that A is inserted as the 7th base:
  - `reference=AGGGGG`
  - `allele-call=AGGGGGA`
- The second strand of the allele call field shows how that strand differs from reference (GAAAAAA compared to AGGGGG).

```
chr18 AB_SOLiD Small Indel Tool insertion_site 7002407 7002407 1.0000 . .
ID=256;ins_len=1;clustered-indel-sizes=1;allele-call-pos=7002407;reference=AGGGGG;allele-
call=AGGGGGA/GAAAAAA;rightmost-allele-call-pos=7002407;rightmost-reference=AGGGGG;rightmost-
allele-call=AGGGGGA/GAAAAAA;context-pos=7002407;context-reference-seq=AGGGGG;context-variant
-seq=AGGGGGA/GAAAAAA;reference-reads=1;context-variant-reads=1;context-variant-nonPCRdup-rea
ds=1;gap-total-reads=2;gap-nonred-reads=2;gap-nonPCRdup-reads=2;tight_chrom_pos=7002412-700
2412;loose_chrom_pos=7002406-7002412;coverage_ratio=0.5000;zygosity=HETEROZYGOUS-NON-
REF;zygosity-score=0.0113;
```

Small indel TXT and SQL formats

The GFF file serves as the primary output format for the small indel caller. However, the TXT output file is an alternative format of the data contained in the GFF file. The information contained in the TXT file is generally only a subset of that of the GFF. The SQL file contains a CREATE TABLE command for MySQL, and the contents of the SQL file can be imported into MySQL.

Table 63 describes specifically what information the small indel TXT file contains.

Table 63 Small indel TXT file format description

Column name	Description	Example
chrom	Chromosome number of indel.	1
min-chrom-pos	Start position of the indel.	713662
max-chrom-pos	End position of the indel.	713663
called-range	Range of chromosome position range of the feature. **Note:** This ambiguity is resolved in the final indel call of the GFF.	713661–713661
tight-range	Conservative estimate of chromosome position range of the feature.	713662–713667
loose-range	Estimate of the maximum chromosome position range of the feature.	713662–713667
num-pos-strand	Number of reads that were mapped to the positive strand.	3
num-neg-strand	Number of reads that were mapped to the negative strand.	0

Table 63 Small indel TXT file format description *(continued)*

Column name	Description	Example
num-r3-hits	Number of reads where the indel was found on the R3 or F5 tag.	0
num-f3-hits	Number of reads where the indel was found in the F3 tag.	3
num-frag-hits	Number of reads where the indel was found from a Fragment tag.	0
indel-type	The type of indel: INSERTION, DELETION, or COMBINATION. For COMBINATION, there were reads indicating both an insertion and a deletion, and no indel size was called.	DELETION
unique-indel-size	Called indel size.	–2
indel-size range	The indel sizes reported from each of the reads.	–2 to –2
num-uniq-align	Number of non-redundant alignments in the pileup.	3
num-tot-align	Total number of reads in the pileup.	3
average-read-position	Average read position where the gap occurred.	15.6667
ave-from-end-read-position	Average from end read position where the gap occurred.	34.3333
indel-read-pos-list	List of read positions where gap occurred.	20;17;10
dbsnp-indel	Legacy. Not used.	—
uw-hgsv-indel	Legacy. Not used.	—
read-lengths	Lengths of the reads (full read sequence, not the extended match length).	50;50;50
paired-distances	The clone sizes of each of the bead pairs (NIL for fragments).	1426;1370;1454
ave-pair-dist	Average of the paired distances.	1416.667
tags-R3-F3	Tags where the indel was found. Possible values are F3, R3, and FRAG. **Note:** For PE data, R3 represents the F5 tag.	F3;F3;F3
chrom-pos-s	Chromosome positions of the indel tag's match location.	713641;713644;713651
strands	Strand for each bead id.	+;+;+
indel-sizes	List of indel sizes found for each read.	-2;-2;-2
nums-mismatches	List of number of mismatches for each read.	–1;–1;–1
ave-numb-mismatches	Average of the number of mismatches found in the indel tag.	–1
indel-lower-pos-s	List of lower ranges of the indel.	20;17;10
indel-upper-pos-s	List of upper ranges of the indel. The lower and upper ranges represent the ambiguity of the gap alignment, for example, `AT/--` in the context of `ATAT/AT`.	25;22;15
run-names	Run names (one for each input file) for each read. For BAM files, the 1 in 50_1_r is the runIdNum in the ##@HD header line of the GFF file.	L1_1_50_1_r;L1_1_50_1_r ;L1_1_50_1_r
clone-ids	Bead IDs for each read.	984_536_1054;1431_2007_ 1567;116_364_1582
read-seqs	The read sequences for each read.	T302000;T120320;T232132

Table 63 Small indel TXT file format description *(continued)*

Column name	Description	Example
ref-allele	Reference allele.	acagagagaag
var-allele1	Most common variant allele (if it passes the color error correction, otherwise it is NULL).	aCAGAGAAG
other-var-alleles	Other alleles of the variant, if present.	—
var-counts1	Number of reads that have the most common indel allele.	3
other-var-counts	The list of the other allele counts.	NO_CALL
ungapped-unmatched-lengths	For a particular bead-id, the length of the read that was left unmatched (equal to the read length if no ungapped match was found or if the detail level is not set to 9).	50,50,50
ave-ungapped-unmatched	Average of the ungapped, unmatched lengths.	50
anchor-match-lengths	Anchor tags match length.	49,44,49
ave-anchor-length	The average of the anchor match lengths.	47.3333
coverage-ratios	Clipped normal coverage/number of non-redundant reads.	1.6667
zygosity-call	Experimental zygosity call.	HEMIZYGOUS
zygosity-p-value	Experimental zygosity score.	0.9656

Example TXT file content

The following is an example of small indel text file output. See Table 63 on page 252 for an explanation of the fields in this file.

```
1       3534096 3534096 3534095-3534095 3534096-3534098 3534096-
3534098 1       2       0       0       3       DELETION        -
1     -1 to -1       2      3      24.6667 25.3333 30;22;22
\N  \N       50;50;50        \N      \N      FRAG;FRAG;FRAG
3534065;-3534119;-3534119       +;-;-   -1;-1;-1        0;1;0
0.3333  30;20;20        32;22;22
L1_1_50_1_r;L1_1_50_1_r;L1_1_50_1_r
894_1039_1506;1267_2006_1555;275_1513_1309
T33222211112120130220221120311133033230333300000331;T0211213300
00033330323303311130211220220310112111112;T02112133000003333032 3
303311130211220220310212111112   ATAAATA    ATAATA      3
50,50,50        50.0000 NIL,NIL,NIL     NIL     7.0000
HEMIZYGOUS      1.0000
```

Small indel ALIGN format

The ALIGN file contains a text visual of each pileup that makes an indel call. Below is an example record from this file, with approximately 30 base pairs trimmed from each side for clarity.

```
> 64,chr1:16765166-16765167(),DELETION,-2,;allele-call-pos=1676
5166;allele-call=CT/;allele-pos=16765164;alleles=CACTCTGA/CACTG
A;allele-counts=REF,15
CGAAGGGGTCAAAGGACACTCTGAGTTAGTG  16765118        16765208
320200012100202111222122103211 3
                CACTCTGA         16765164        16765170
```

```
                1122212
   T10012100202111--21221032113  1122212/11212+  CACTCTGA/CACTGA
   200012100202111--21221032113  1122212/11212+  CACTCTGA/CACTGA
T30200012100202111--21221032113  1122212/11212+  CACTCTGA/CACTGA
  02000021002022l1--21221032133  1122212/11212+  CACTCTGA/CACTGA
 20200022100202211--21222032113  1122212/11212+  CACTCTGA/CACTGA
 20200002100202111--21221032113  1122212/11212+  CACTCTGA/CACTGA
02020001210020211l--21221030113  1122212/11212+  CACTCTGA/CACTGA
02220001210020211l--21221032113  1122212/11212+  CACTCTGA/CACTGA
12020001210020211l--21221032113  1122212/11212+  CACTCTGA/CACTGA
12020001210020211l--21221032113  1122212/11212+  CACTCTGA/CACTGA
12020001210020211l--21221032113  1122212/11212+  CACTCTGA/CACTGA
12020001210020211l--2122103213T  1122212/11212+  CACTCTGA/CACTGA
12020001210020211l--2120T        1122212/11212+  CACTCTGA/CACTGA
       12100002111--21221032113  1122212/11212+  CACTCTGA/CACTGA
       12100202111--21222032113  1122212/11212+  CACTCTGA/CACTGA
```

Table 64 Small indel ALIGN file format

Line	Name	Format and description
1	Indel	> ID,chr#:start-end(),INDEL_TYPE,INDEL_SIZE. The rest of the line contains several tags that are in common with the GFF file.
2-5	Reference	REFERENCE_SEQUENCE START_POS END_POS Reference sequence in base and color space for entire pileup region, and region around the indel. The coordinates are given after the sequence.
6+	Reads	COLOR_SPACE_READ REF_COLORS/READ_COLORS+ REF_BASES/ READ_BASES Contains all the reads of the pileup, extracted colors around the indel, and the corresponding bases. The sign after the extracted colors indicates color space compatibility (+) or incompatibility (-).

UNGAPPED and PAS.SUM formats

The UNGAPPED file contains more details of the reads counted towards the number of ungapped alignments used for the coverage ratio. For every pileup there are as many header lines as number of BAM files as input. The header line has this format:

```
# INDEL_ID BAM_CHROM_NUM POS1 POS2
```

Each line after the header is a list of bead ids along with properties of that bead extracted from the BAM file.

The PAS.SUM file is an intermediate the pileup file extracted from the BAM file. The file starts with BAM header information from each of the inputs. Following that, each pileup has a separate line. An example and format are given in Table 65.

Table 65 Column descriptions of the PAS SUM file

Name	Components	Example
Position, id, and ungapped	{CHROM_NUM}:{POS}-{POS2}:{INDEL_ID}:{NUM_UNGAPPED_READS}	7:315505-315512:347316:2

Table 65 Column descriptions of the PAS SUM file

Name	Components	Example
# reads	{NUM_NON_REDUNDANT}:{NUM_NON_OPT_DUPS}({NUM_TOTAL_READS})	10:9(13)
Indel size	{MIN_INDEL_SIZE}-{MAX_INDEL_SIZE}	2-2
Reference	{REF_POSITION}:{REF_SEQUENCE}	315455:ACGCA...GGAGCA
Align 1	{BEAD_ID} {UNMATCHED-LENGTH}__{ISDUP}_{READ_LENGTH}_{BAMID}_{READGROUPID},{PAS_ALIGNMENT} {BASE_QVS_TAG1} {BASE_QVS_TAG2} {MAP_QV_SCORE}	1524_856_1942 0__NonDup_60_1_1,7_-315519.57.2(12:12_13)[!TAA...CT]|7_-314783.-1:(59.2.0):q255[!CCA...CG] AB@...>:9 AA<9...%%* 22
Align 2 and higher	[same as above]	1970_1821_1324 0__Dup_60_1_1,7_-317273.-1:(59.0.0):q255[!CTG...AG]|7_-315522.57.0(12:12_12)[!GCA...AC] >BA...7?4> BB?...9B; 100

CHROM_NUM is 1-based. Position 1 and position 2 are the approximate position of the indel from the alignments. The position is recalculated later from the actual alignments. The number of non-redundant reads is calculated by the module and is used for the coverage ratio calculation, while the number of non-optical duplicates is counted from the BAM `0x0400` flag. The individual alignments follow either TAG2|TAG1 for paired libraries, or just TAG1 for fragment ones. In each tag, the format for a gapped alignment is:

```
chromNum_ReadPos.MatchLen.MisMatch(IndelPos:Start_End)[!Base or Color Seq]
```

The format for an ungapped alignment is:

```
chromNum_ReadPos.MatchLen.MisMatch(MatchLength.0.0)[!Base or Color Seq]
```

Base sequences start with `!`, while color start with the primer base.

The following is an example from base space indel calling using ECC mate-pair library.

```
7_-317273.-1:(59.0.0):q255[!CTG…AAG]|7_-315522.57.0(12:12_12)[!GCA…CAC]
```

Gap alignment mapping

Detection of indels variants with a split-read technique is achieved by using LifeScope™ Software's small indel caller on BAM files produced from long mate-pairs, mate-pair, fragment, and pair-end library types. The combination of multiple libraries of any combination of these types is also possible. For pair-end and long mate-pair libraries, the small indel module, by default, is able to determine sizes up to 500 for deletions and 20 for insertions. For all fragment libraries, the default size range is up to 19 for deletions and up to 4 for insertions. Also, 75mers from either pair-end and fragment libraries are capable of insertions up to at least size 29. Furthermore, the module allows for detection of more complex variants such as indel-SNP combinations.

The small indels module determines high-quality calls for insertions and deletions in two stages. In the first stage, the LifeScope™ Software mapping module (see Chapter 9, "Run a Resequencing Mapping Analysis" on page 137) produces gapped alignments on a bead-by-bead basis and writes them to the resulting BAM file. For all libraries, this determination occurs during the mapping phase of this using a single tag approach. For paired tag libraries, this determination additionally occurs in the pairing phase using a paired tag approach. In the second stage, the indel caller takes these gap alignments, extracts them out of the BAM file, forms pileups, filters the pileups based on certain heuristics, determines zygosity, and annotates the indel sequences. This results in concisely annotated and highly accurate indel calls.

Paired tag approach (paired libraries only)

Figure 18 illustrates the F3 with the indel. The algorithm also determines indels in the R3/F5 tag.

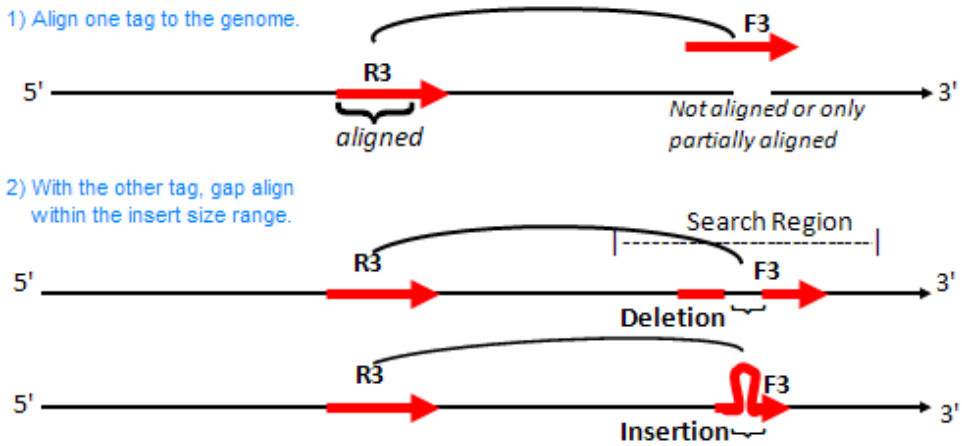


Figure 18 Gap alignment detection using the paired reads from mate-pair and paired-end data

For paired tags, the algorithm surveys only those indels with one-end anchored (OEA) mate-pairs (see Figure 18). It does so by realigning the OEA pairs using an anchor tag (that is one tag which can be aligned to the genome by itself) and by performing a more aggressive alignment with the other tag in a several kbp window (depending on the orientation of the tags and on the minimum and maximum insert sizes set in mapping's pairing phase) around the anchored mate. A pair is considered OEA if one read of the tag fails to align or if its aligned length was not higher than a certain length threshold, specified by the parameter `pairing.gap.min.non.matched.length`. In addition, for anchor tags, only alignments above a minimum length are used (those alignments are close to full length). Finally, the anchor tag may only have a maximum number of anchor alignments (close to full length alignments) in the genome, which by default is set to 10 by the second value of `pairing.gapped.max.hits`.

Using the unanchored or non-fully extended tag, the module starts aligning both ends of the read localized by the other tag's match location. With the region of the genome determined by this match location and insert size distribution, the module makes a catalog of end locations by extending the read starting with a minimum value (specified by `pairing.gap.edge.length.insertions` and `pairing.gap.edge.length.deletions` parameters rescue for insertions and deletions, respectively) until the alignment hits the maximum of number of mismatches allowed. These maximums are specified by these parameters:

- pairing.anchor.max.mismatches: the number of mismatches in both tags

- pairing.gap.max.mismatches.tag1: mismatches for the F3 tag
- pairing.gap.max.mismatches.tag2: mismatches for the R3 or F5 tag

The default value for the single tag number of mismatches is 5 for read lengths 50 and higher (50 set by `min.length.for.aggressive.gapped.mapping`), and 2 for shorter read lengths. Then the single tag number of mismatches for paired-end libraries is 5 and 2 by default for the F3 and F5 tags, respectively.

With this catalog, the process attempts to join the ends of the read to find a single gapped alignment within a certain size range. The maximum gap size allowed depends on parameters specified in the mapping's pairing phase parameters. Furthermore, it identifies it as a gapped alignment if the above joining could be done with the fewest number of mismatches. Ambiguity of the location of this joining is common, mainly due to the presence of short tandem repeats. This is represented by the XA field in the BAM file. Later on downstream, the indel caller (as described later) using the consensus of reads, produces a left most, concise representation of the variant. A rightmost representation is also produced. Furthermore, gap alignments must also satisfy a minimum edge length which is the number of base pairs before the end of the read.

For determining small indels (deletions to size 11 and insertions to size 4), the algorithm disallows indels within 3 bases from either end of the read. The module identifies if it is able to piece together both ends, allowing only for a single gap of up to 4 base pairs inserted (present in read but not in reference), or up to 11 base pairs deleted. Larger indels up to size 500 are also found using different edge length criteria. These sizes and criteria are specified in several mapping parameters as shown in the Table 66. For example, the default value for `pairing.gap.max.deletion.size` is `0:11:0:0:500`, corresponding to the first row of Table 66. The values that effect gap alignments are the second through fourth ones, and are specified in the table as small indels, medium insertions, larger insertions, and larger deletions, respectively. `s1` and `s2` are specific tag lengths of the library used. For example, the parameters `pairing.gap.max.deletion.size.75-35` and `pairing.gap.max.insertion.size.75-35` affect a 75x35 PE library, but not other size libraries.

Larger indels are found for 60-60, 50-50, and 75-35 read length libraries, but not found for smaller 50-35, 50-25, 35-35, or 25-25 read length libraries.

Table 66 Parameters controlling gap size ranges, with default values shown

Parameter	Small indels	Medium insertions	Larger insertions	Larger deletions
`pairing.gap.max.deletion.size` `pairing.gap.max.deletion.size.s1-s2`	11	0	0	500
`pairing.gap.max.insertion.size` `pairing.gap.max.insertion.size.s1-s2`	4	15	21	0
`pairing.gap.edge.length.deletions` `pairing.gap.edge.length.deletions.s1-s2`	3	0	0	20
`pairing.gap.edge.length.insertions` `pairing.gap.edge.length.insertions.s1-s2`	3	14	14	0

Single tag approach (for both paired and fragment libraries)

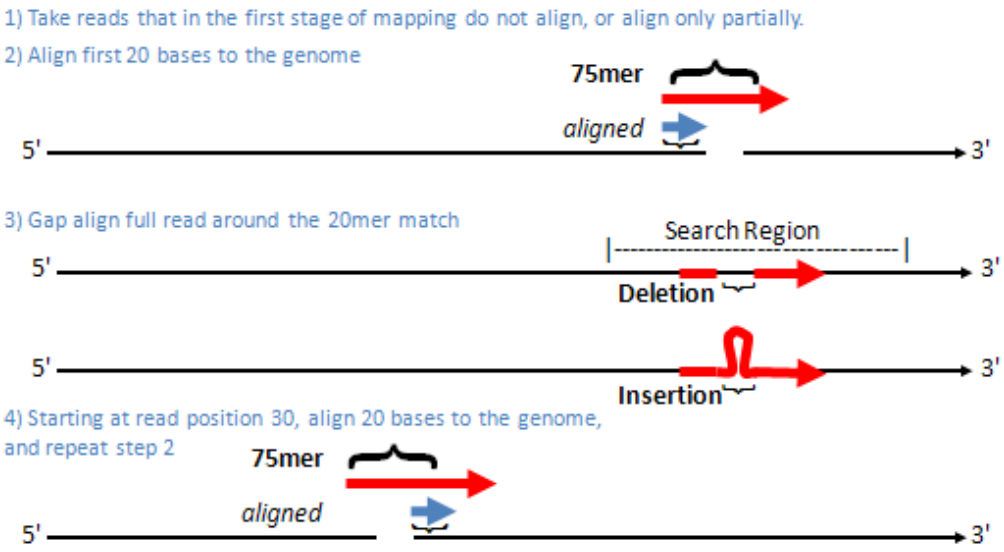


Figure 19 Global full read gap alignment detection using the single read technique used in all libraries

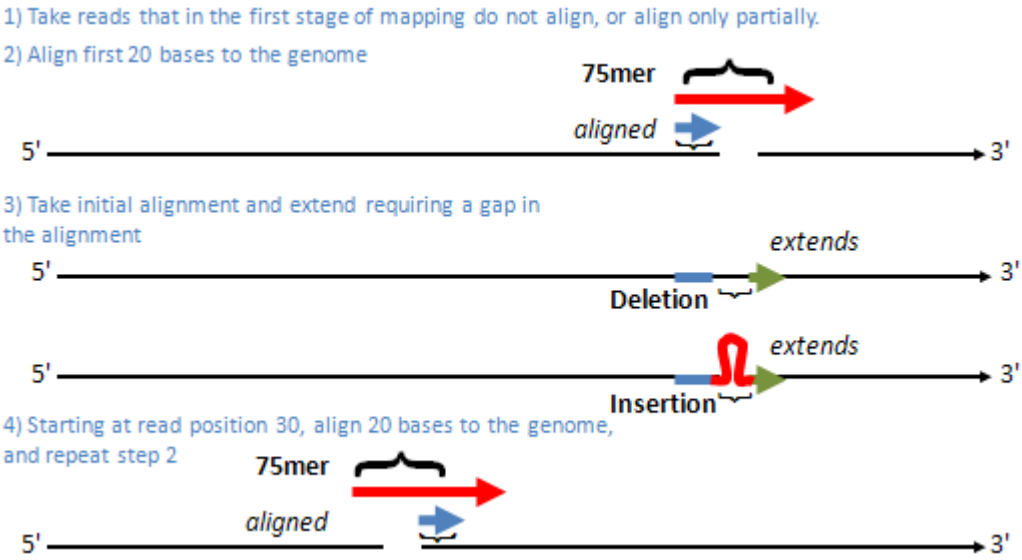


Figure 20 Local gap alignment detection using the single read technique used in all libraries. Reads here may not necessarily extended to the full read length.

Using only a single tag, gap alignments are determined in the map phase of resequencing mapping. Although gap alignments can be found along with ungapped alignments in the first stage of mapping, the process here describes the default process of finding gap alignments only in the second stage of mapping. This process is illustrated in Figure 19 (global) and Figure 20 (local). As with mate-pair libraries, whole genome searches for gapped alignments is prohibitively expensive. Similar to the paired tag approach, a read is only considered for this process for both local and global alignment methods if the tag fails in the first stage of mapping to align or if its alignment length did not extend past a certain length threshold, specified by the gap

aligner parameter `mapping.gap.min.non.matched.length`. Then each read is localized by performing a 20 base pair ungapped alignment allowing for 1 mismatch. The 20mer taken is from both the beginning and 30 bases pairs into the read. The anchor properties are specified by the `second.map.scheme.unmapped.xx` mapping parameters, and for 50mers and longer, the value is `20.1.0:30`.

Two algorithms are available in the system, GLOBAL (default) and LOCAL, specified by `second.map.gapped.algorithm`. For GLOBAL (full read) gap alignments, each 20mer alignment then defines a search region of [A-40, A+80], where *A* is the position of the alignment. The downstream range (80) is settable using the parameter `second.map.gapped.seed.window.right`, and upstream (40), using `second.map.gapped.seed.window.left`. With this region, the same full read alignment strategy done with mate-pairs is performed: a catalog of partial begin and end read hits is formed, and an attempt is made to join them with a gap of some size. For LOCAL gap alignments, full read length matches are not required, but rather a gap is allowed during extension of the original anchor hit with a gap and mismatch penalty specified by the parameters `second.map.gapped.penalty` (default of 5) and `second.map.mismatch.penalty` (default of -2.0), respectively. Once the penalty threshold is met, the extension of the read terminates.

As in the paired approach, all gap alignments are restricted by both maximum size and minimum edge length requirements. By default, this approach finds deletions to size 19 and insertions to size 4. Larger insertions up to size 29 have been found effectively with 75mers using more restrictive parameters, which at the same time reduce sensitivity for smaller insertions and do not perform deletions. Insertions specifically at size 30 were set, but not evaluated. Both the default and larger insertion parameters are specified in the Table 67.

Table 67 Various size and edge length requirements for the small indels and larger insertions

Parameter	Small indels (default)	Medium insertions (optional for 75mers)

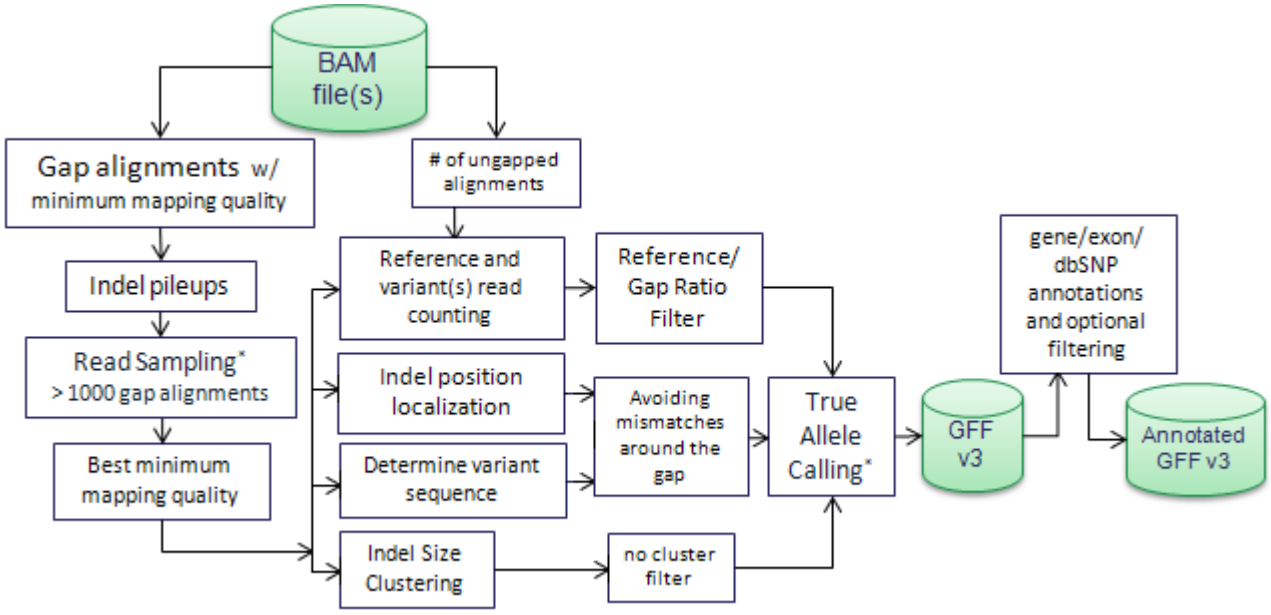
`second.map.gapped.deletion`
`second.map.gapped.deletion.tag1`
`second.map.gapped.deletion.tag2`
	19	0

`second.map.gapped.insertion`
`second.map.gapped.insertion.tag1`
`second.map.gapped.insertion.tag2`
	4	30

`second.map.gapped.edge.length.deletions`	12	0
`second.map.gapped.edge.length.insertions`	12	18

Small indel calling from gap alignments

The small indel caller uses alignments contained in the BAM file, which in the LifeScope™ Software mapper have two general types, ones that contain a single gap in the alignment (those that have an `I` or `D` in the CIGAR string) and ones that do not, which is referred to as ungapped alignments. The caller uses a variety of algorithms to produce the indel calls present in the GFF file. These algorithms are discussed in the sections below, and are summarized in this figure.



Pileup handling

The gap alignments contained in the BAM files form the basis for calling indels. The gap alignments go through a series of processes before being reported in the final GFF output. Those alignments that have a minimum overall mapping quality (`small.indel.min.mapping.quality`, 8 by default) are extracted from the BAM file. The exception to meeting this quality threshold is the class of F3 gap alignments from pair-end libraries (see "Mapping quality adjustment (paired-end libraries only)" on page 263).

Properties of the gap alignment tag (`small.indel.min.non.matched.length`) and anchor tag (`small.indel.min.anchor.mapping.quality`) are also assessed and are affected by `small.indel.detail.level`. Only the first 6 (detail level x 2) anchor reads are considered for the default setting of 3. Alternative alignments for the non-matched length filter are considered only for a detail level of 9. Alternative alignments are those that are found without a gap at the same chromosome location. Furthermore, anchor reads are present only in paired approaches, but alternative alignments could be present in either paired or single tag approaches.

Next the gap alignments are grouped together by genomic location to form pileups of reads. Because of the positional ambiguity of indels, pileups are formed by proximity; specifically, alignments that are within 5 base pairs between consecutive evidences are combined together. For pileups that have 6 or more non-redundant reads, the sixth and additional reads after that are only grouped together if the pileup is 2 or fewer base pairs from the last gap position. The default behavior is set by the parameter `small.indel.consGroup` (1 by default). A value of 9 makes every gapped alignment into a separate pileup.

Finally, the pileup is taken if the number of non-redundant reads is between `small.indel.min.num.evid` and `small.indel.max.num.evid`, inclusively. By default, pileups with two or more reads are taken at this stage. Non-redundant reads are those reads that have unique F3 and R3/F5 positions for paired tags, or unique F3 positions for single tag analysis. Mixed single and paired tag reads are

considered unique from each other. With each pileup extracted from gap alignments, ungapped alignments that span the gap by at least 5 base pairs are counted, and are used in the normal vs. indel coverage ratio. The module stores the pileup information extracted from the BAM file into an intermediate PAS.SUM file for further analysis.

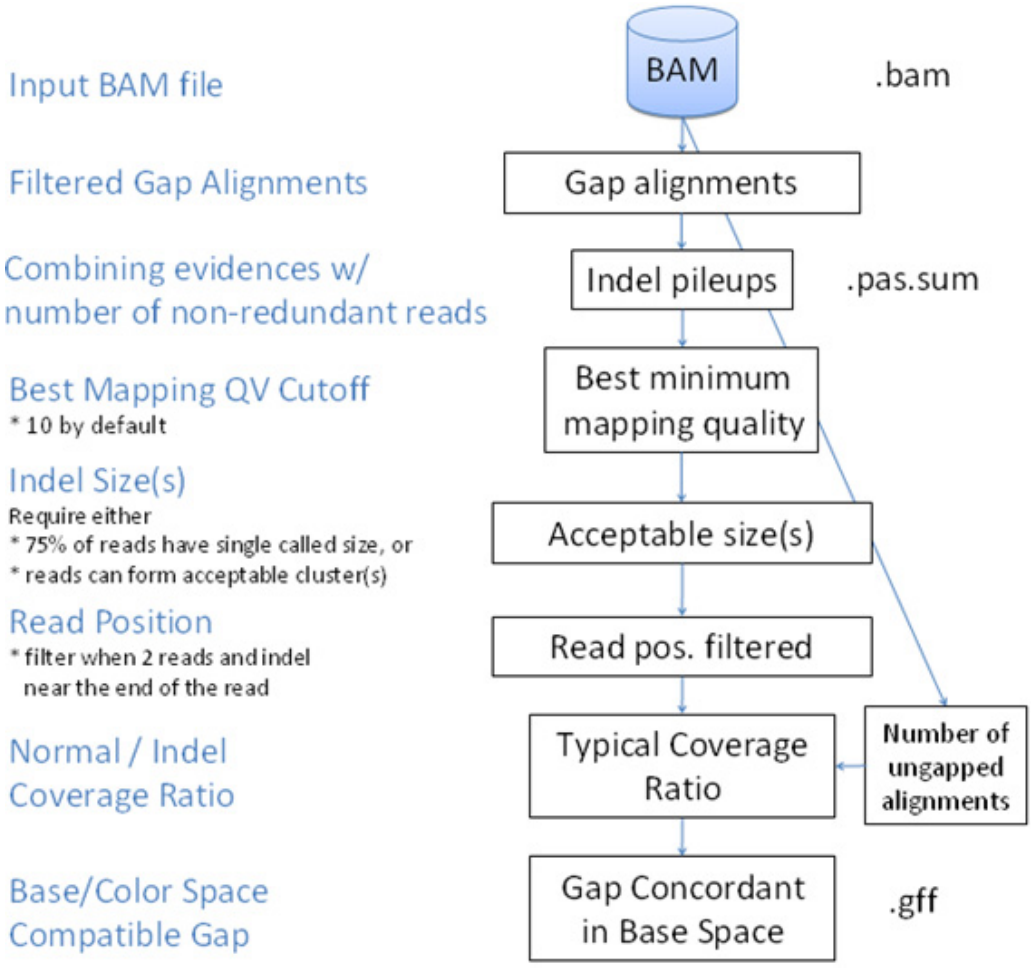


Figure 21 Small indel caller heuristics

The pileups go through additional filters, as illustrated in Figure 21, each with corresponding optional parameters in Table 61 on page 241. The best mapping quality cutoff (specified by `small.indel.min.best.mapping.quality`, with a default of 10) is the cutoff for the highest overall mapping quality from the pileup. For paired tags, this cutoff is the highest pairing quality value; for fragment, the highest mapping quality; and for mixed, the highest from any of the reads of either. Filtering for the maximum value allows lower quality reads to act as supportive evidence. Ambiguous indel size filtering requires that either 75% of the alignments have the same size or that they are able to form at least one valid clustered indel size. The latter method, in the case of multiple clustered sizes, allows including non-reference hemizygote indels. This method is described in more detail in "Indel size determination" on page 267.

If only two non-redundant reads are required to make an indel call, false positives can become prevalent at higher coverage levels (for instance, 100x). By using the number of ungapped alignments queried from the BAM file, the software can determine a coverage ratio of ungapped (with 5 base pairs clipped) coverage over a number of non-redundant indel supports. A high ratio is indicative of a false positive and are filtered out at values higher than the setting of `small.indel.max.coverage.ratio` (12 by default). Even in low coverage situations, there are regions of high coverage where this filter is active and further helps to reduce false positive indel calls.

Finally, the pileup must also be comprised of reads where the majority of reads have gaps that are color space compatible. The parameter `small.indel.alignment.compatibility.filter` sets filtering based on this test, where a setting of 0 indicates no color space-based filtering and 1 filters out pileups were NO_CALL is most prevalent (1 is the default for LifeScope™ Software and recommended for max mapping). NO_CALL for the allele occurs in reads where the gap is not color space compatible.

Mapping quality adjustment (paired-end libraries only)

Certain paired alignments may have poor overall mapping quality because of poor mapping of one tag. However the gap alignment of the better tag may still constitute strong evidence for indels and could be used for increased sensitivity. In order to effectively use these alignments in BAM files missing the single tag mapping quality (SM), the tool performs an adjustment to the mapping quality for the sole purpose of reducing the quality required to meet the minimum overall mapping quality (`small.indel.min.mapping.quality`) for certain otherwise high quality reads. This adjustment has no effect in either the best mapping quality filter or the reported output of quality scores in the GFF3 file. The QV adjustment follows this formula:

$$QV_{adj} = elc \cdot EL - mmc \cdot MM + cc$$

where *elc*, *mmc*, and *cc* are the coefficients for edge length, number of mismatches, and constant, respectively and EL and MM are the actual minimum edge match length and number of mismatches, respectively for that read. By default this is done only for the F3 tag of a pair-end library, with *elc* = 0.25, *mmc* = 0.50, and *cc* = 5. These settings are not readily available, as it may require editing the small indel XML file to add `small.indel.mapping.quality.adjustment.libraries.tag1` (default value of PE) and `small.indel.mapping.quality.adjustment.coefficients.tag1` (default of 0.25,0.50,5) parameters (editing the file is not recommended). This adjustment has no effect on the second (F5) tag. To turn off this adjustment, specify `None` for the adjustment coefficients parameter.

Color space considerations

When an indel occurs in a sequence, and that sequence is measured using color space, the color-space sequence not only has a gap of the same size of the indel, but also leaves a signature that can indicate if there is a measurement error within the gap. This is especially important in the case of insertions when you have a small number of evidences and there is a disagreement in the bases of the inserted sequence. With methods that directly measure bases, there would be no indication, based on the inserted sequence alone, on which inserted bases is more trusted.

In color space, this signature can be used to see if the color that spans the gap is compatible with the set of colors that go through the gap. For example, the alignment AACG/A--G would be 013/2-- in color space. Here the color 2 spans the gap (measuring both A and G), while 013 goes through the gap (measuring AACG). The color 2 is compatible with the sequence 013 because both would end with a G in base space. However, an alignment of 213/2-- would not be compatible because, using the

same starting base A as the above example, 213 would measure AGTA. Because the rest of the color-space sequence beyond this would be aligned, 213/2-- would be indicative of a measurement error within the insertion if 213 is the sample, or if the sequence is the reference, the color 2 that spans the deletion. The alignment's color space compatibility can be calculated for any sequence using color space addition. This signature for color space compatibility is used to more accurately call the inserted sequence of the insertion, important if only a small number of reads are used to call an indel. Also an entire pileup is more likely to be a false positive if most of the reads indicate a gap that is not color space compatible.

Allele calling and sequence context determination

For every gap alignment, the LifeScope™ Software aligner reports the position of the gap and an ambiguity of that placement using the XA BAM field. The caller takes that information and the color-space reads to determine the base-space sequence for each of the reads, reporting the reference and all the sequences found. The module takes these calls further by taking the reference and the most common allele(s) present and then making a concise representation.

Context sequence extraction accounting for ambiguous placement

The small indel caller has three representations of the reference and variant sequences that contain the indel:

- **Context sequences** – Represents the sequence context around the indel.
- **Leftmost sequences** – Represents the sequence after trimming all common bases between the variant reads and the reference. Trimming goes from the right to the left first, and then left to the right.
- **Rightmost sequences** – Same as leftmost, except that trimming goes from left to right first.

The context region is specified by taking the `loose_chrom_pos` (which is contained in the output GFF), expanding the start position by 1 and the end position by 2, and then trimming the first two positions and the last two positions, only if the bases at those particular positions are identical in all of the reads and the reference. In addition, if any of this trimming process produces no bases for the reference sequence, the position is represented as that of the base immediately before the insertion.

An example of this is the following indel call that has a `loose_chrom_pos=94446948-94446955` and is annotated in dbSNP as `rs58864345`. The reference sequence at positions `94446948` to `94446956`, and the corresponding reads is:

```
Ref     A---CTTCTTCCC9444694894446956
        1---20220200
Read1   ACTTCTTCTTCCC
Read2   ACTTCTTCTTCCC
Read3   ACTTCTTCTTCCC
Read4   ACTTCGTCTTCCC
```

Because the beginning A and the ending CC are all the same, these would be trimmed off. This would result in the context sequence related tags being reported as:

```
context-pos=94446949;context-reference-seq=CTTCTTC;context-variant-seq=CTTCTTCTTC
```

The leftmost trimming of this result is represented here:

CTTCTTC Reference @ 94446949

CTTCTTCTTC Sequence found in reads

This results in the following positioning and allele call:

```
allele-call-pos=94446948;reference=-;allele-call=-/CTT
```

Since the reference here was entirely trimmed, the position reported, as per the GFF standard, is the one immediately before the CTT insertion. Finally, the rightmost trimming is represented here:

CTTCTTC Reference @ 94446949

CTTCTTCTTC Sequence found in reads

and results in a different position:

```
rightmost-allele-call-pos=94446955;rightmost-reference=-;rightmost-allele-call=-/TTC
```

All three of these representations are present to preserve the CTT repeat context and have positioning compatible with both VCF (leftmost) and dbSNP (rightmost) positioning.

Making concise allele calls

The allele-call and rightmost-allele-call tags also represent information about the sequences detected. It always contains the most common indel variant sequence. Also, if a large enough number of ungapped alignments are present, it includes the sequence taken from the reference. Finally, if the second most common variant sequence has at least 75% of the reads as the most common, it too is included.

The following indel call from chromosome 2 at position 220,378,748 in hg18 illustrates the allele calling. The sequence context of this detected indel is:

```
context-reference-seq=atatttttttttttt;context-variant-seq=aTATATTTTTTTTT/
aTTATTTTTTTTTT/aTATTTTTTTTTAT
```

and the read counts is `reference-reads=18;context-variant-reads=4,4,1`. From the read counts, both the reference (ATTT) and the two different variant sequences (ATA and TAT) are called. However, the variant with just one read is be called. All in all, the resultant allele call would be

```
allele-call-pos=220378748;reference=attt;allele-call=attt/ATA/TAT
```

VCF format

This section describes the differences between indels in GFF format with indels in VCF format.

Insertion VCF record

The following is an example of a VCF record with an insertion:

```
20     3 .          C       CTAG     .    PASS   DP=100
```

This record contains an insertion since the reference base C is being replaced by C (the reference base) plus three insertion bases TAG. This example contains only two alleles so we have the two following segregating haplotypes:

```
Ref: a t C - - - g a
```
// C is the reference base

```
   : a t C T A G g a
```
// following the C base is an insertion of 3 bases

The following shows how the LifeScope™ Software small indels module represents the same insertion, in GFF3 format:

```
gff start/end position:  3/3
reference = -
variant = TAG
```

Deletion VCF record

The following is an example of a VCF record with a deletion:

```
20     2 .          TCG       T     .    PASS   DP=100
```

This record contains a deletion of two reference bases since the reference allele `TCG` is being replaced by just the `T` (the reference base). This example contains only two alleles so we have the two following segregating haplotypes:

`Ref: a t C g a` // `C` is the reference base

`: a t - - a` // following the `C` base is a deletion of 2 bases

The following shows how the small indels module represents the same deletion, in GFF3 format:

```
gff start/end position:  3/4
reference = CG
variant = -
```

Mixed VCF record for a microsatellite

The following is an example of a mixed VCF record:

```
20     2 .          TCGCG       TCG,TCGCGCG     .    PASS   DP=100
```

This example is a mixed type record containing a 2-base insertion and a 2-base deletion. This example contains three segregating alleles so we have the three following haplotypes:

`Ref: a t c g c g - - a` // `C` is the reference base

`: a t c g - - - - a` // following the `C` base is a deletion of 2 bases

`: a t c g c g c g a` // following the `C` base is a insertion of 2 bases

In all of these examples dashes have been added to make the haplotypes clearer, but of course the equivalence among bases is not provided by the VCF. Technically the following is an equivalent alignment:

`Ref: a t c g - - c g a` // `C` is the reference base

`: a t c g - - - - a` // following the `C` base is a deletion of 2 bases

`: a t c g c g c g a` // following the `C` base is a insertion of 2 bases

The following discussion describes how the small indels module represents the same mixed record, in GFF3 format.

Leftmost representation of the indel actually represents the haplotypes as:

`Ref: a t - - c g c g a` // `C` is the reference base

`: a t - - - - c g a` // following the `C` base is a deletion of 2 bases

`: a t c g c g c g a` // following the `C` base is a insertion of 2 bases

Given the above representation, the small indels module represents the deletion as:

```
gff start/end position: 3/4
reference=cg
variant= -
```

The insertion is represented as:

```
gff start/end position: 2
reference=-
variant=cg
```

When both the insertion and deletion occur at the same time, the small indels module represents them as:

```
gff start/end position:  3/4
reference = CG
variant = -/CGCG
```

Note: The indel caller may be missing some of the heterozygous indel sequences when there are multiple sizes present in a single indel call. In these cases, the allele-call is represented by the `possibleOthers` tag. This occurs when the second most common size does not have 75% of the reads as the most common indel size. The indel sequence is also missing with the third most common and rarer indel sizes, if present.

Indel size determination

Indel sizes are determined by two methods. The first method is if the pileup has a single called indel size, where three-fourths or more of the alignments have a particular size. The GFF file reports this as `ins_len` or `del_len` for an insertion or deletion, respectively. The second method, that is performed on all pileups, is clustering based on the indel size. The list of indel sizes for a particular pileup is collected. Insertions and deletions, if they both occur, are split into two clusters. If both insertions and deletions do not occur, the starting point is a single cluster of all the sizes.

From those clusters that are present, the algorithm makes a decision whether or not to split that cluster based on the size differences contained in that cluster. For small indel clusters, those whose average size is less than 20, single base pair differences are split. For example, a size list of 1,1,1,2,2 are split into two clusters of 1,1,1 and 2,2. For the other clusters, they are split if the ratio of the largest size difference over the smallest indel size is greater than 1/4. For instance, a size list of -300,-340,-499,-500 results in two clusters of -300,-340 and -499,-500. If a cluster is not split by either of these, it still can be split if both the highest distance minus the average distance of the rest is greater than 6, and the highest distance is greater than 5/4 of the average distance of the rest.

Clusters are split recursively until any resulting cluster cannot be split further, and this process forms a list of clusters. Any cluster that has a single alignment is removed. For small indels (those less than 20 in size), the algorithm uses the single size that is present in that cluster. For a cluster of larger indels, the algorithm uses an average size found within that cluster. All cluster sizes are reported by the `clustered-indel-sizes` tag in the GFF. In addition, if there is not an indel size called with 3/4 agreement in size, then the cluster size(s), if at least one exists, is reported in the GFF as a comma-separated list after `ins_len`, `del_len`, or `len`. The first two are for when all the sizes are insertions or deletions, respectively, and the last (`len`) is used if there is both an insertion and deletion, where deletions indicated by a negative value.

Indels are also filtered based on whether these methods were successful in determining a consistent indel size or set of indel sizes, depending on the `small.indel.indel.size.distribution.allowed` setting. The default setting (`can-cluster`) requires that at least one size cluster is formed, and also allows for multiple clusters of different sizes. Overall, this method allows for more variety in the types of indels called. For sample long mate-pair and targeted resequencing paired-end datasets, this method does not significantly decrease concordance to dbSNP. A more restrictive setting of `similar-size` does not allow for this variety, by requiring that the pileup has at least 75% of the reads having the exact same size. A setting of `similar-size-any-large-deletions` allows for larger deletions to have different sizes. With `similar-size-any-large-deletions`, the caller includes additional indels that are discordant in size but have at least two reads with deletions of size 20 or higher. The setting `any` turns off filtering based on indel size, but this setting usually results in greatly increasing false indel calls.

Reference allele calling

The coverage ratio (described in "Pileup handling" on page 261) also serves to call an indel hemizygous, specifically the type of zygosity detected here is one where the reference is present. Because the coverage ratio is the number of ungapped alignments over the number of the gapped ones, the ratio is an indicator of this localized hemizygosity. The length scale of this hemizygous call is that of the gap size range present in the input from the aligner, mostly in the small indel range. The size range is much smaller in length scale than the whole chromosome hemizygosity present in, for example, human males for chromosome X. The software calls the presence of reference alleles by utilizing a table of coverage ratios, by utilizing a table of coverage ratios located in a file located at `<installdir>/etc/small-indels/zygosity-calibration.conf`. The table was derived by matching DH10B reads to an indel introduced reference to simulate the non-hemizygous state (indels of the same length on both alleles). For the hemizygous state, reads that occurred over a simulated indel had a 50% chance of being altered to contain that simulated indel. Because indel alignments are generally less sensitive, hemizygous situations occur frequently above coverage ratios values above 1. Different situations were simulated and are available using the parameter `small.indel.zygosity.profile.name`. The DH10B data comes from a 50mer long mate-pair sequencing run, using both tags for the paired approach and using only the F3 tag for a single tag approach.

Zygosity Calling

Determination of the presence of the reference allele is analogous to heterozygous calling in SNPs where one allele is the reference. However, indels have greater complexity, because a single indel can have different inserted base pair sequences or it can have different gap sizes. For example, an indel can be the same with respect to size, but different with respect to the actual inserted sequence (see "Example 7, multiple inserted alleles" on page 271). Also, indels can also have different distinct sizes that are both not the reference (see "Indel size determination" on page 267).

In the end, three factors are used to call zygosity:

- Presence of the reference allele
- Presence of different instructed sequences
- Presence of multiple indel sizes

Non-diploid organisms or regions of higher than two in copy number can even have two or all three of the above factors. To take into consideration all the possible combinations of the above factors, the final zygosity call has values of HEMIZYGOUS-REF, HOMOZYGOUS-NON-REF, HEMIZYGOUS-REF-MULTI-INDEL-ALLELE, HETEROZYGOUS-NON-REF, MULTI-HEMIZYGOUS-REF, and HEMIZYGOUS-NON-REF.

Note: Having multiple different sizes, without having multiple different sequences, is not possible.

Sampling of gap alignments

Sampling of gap alignments occurs when the number of gap alignments contained in a single pileup is greater than `small.indel.num.aligns.per.pileup`. A larger list than this is initially kept, but that size is limited to 30 times this value. Future gap alignments coming in order from the BAM file are ignored. From this larger list, each item has a probability of being included in the sample list of `small.indel.num.aligns.per.pileup` divided by the size of the larger list.

This manifest itself in the `context-variant-reads` tag of the gff. For example,

```
context-variant-seq=GG,AT; context-variant-reads=15464,1497(899,87)
```

means that in the sample, 899 reads had `GG` in the actual sample, but based on the ratio of the sample size to the total size, an estimated 15,464 reads had that allele in the full sample.

Examples

Below are several examples that illustrate the allele calling. The positions reported here are from chromosome 1 of the human hg18 reference.

Example 1, a simple insertion:

```
ins_len=1;allele-call-pos=55076169;reference=-;allele-call=A/A;reference-reads=5;context-
variant-reads=5;coverage_ratio=1.000;zygosity=HOMOZYGOUS-NON-REF;zygosity-score=0.1000
```

An insertion of `A` is reported here at 55,076,169 because it occurs between position 55,076,169 and 55,076,170. The allele call here is both alleles are variant. Even though there are as many reference reads as variants, it is called homozygous because the generally greater sensitivity of finding ungapped alignments.

```
> 4223,chr1:55076169-55076169(),INSERTION,1,;allele-call-pos=55076169;allele-call=/A;allele-
pos=55076169;alleles=/A;allele-counts=REF,5
AATCTATACAGATCATTTCATCTTTTTCTGGCATTGAGT-TATATCTGTACTGGATACCTATGTTTAAGGCTATG 55076131 55076204
032233311223213002132200002210313012210-333322113121023310233110030203233 1
                              Ref   GT-TA   55076168       55076170
                                    10-3
                              Reads GTATA
                                    1333
  T033311223213002132200002210313012213333333221131210
                   32200002210313012213333332211312102331023311003022T
                   32200002210313012213333332211332102331023311003022T
                       00022103130122133333322113121023310233110030203230T
                       T00221031301221333333221131210233102331100302032331
```

Example 2, a simple deletion:

```
del_len=3;allele-call-pos=91763033;reference=AAA;allele-call=AAA/-;rightmost-allele-call-
pos=91763033;rightmost-allele-call=AAA/-;zygosity=HEMIZYGOUS-REF;
```

This example shows a deletion of `AAA` where position 91,763,033 would be the start of the deletion and 91,763,035 the end. It is a hemizygous reference call, so the reference is present in the allele call, and there is no difference in this case between the rightmost and leftmost representations.

```
> 7305,chr1:91763033-91763035(),DELETION,-3,;allele-call-pos=91763033;allele-call=AAA/;allele-
pos=91763031;alleles=ATAAAGA/ATGA;allele-counts=REF,2
TGGTGCTGGTGCTCTTAACAATTTTGTAAATAAAGAAGATAATTTCCTTTTCTAGAGGTACAT     91763002       91763064
10113210113222030110300011300330022022330300202000223222013113
                        Ref   ATAAAGA  91763031       91763036
                              330022
                        Reads AT---GA
                              31---2  (Color 1 spans the gap)
  T13210113222030110300011300 31---2022330300202000223222
        T1322203011030001130031---2022330300202000223222013113
```

Example 3, an ambiguous insertion:

```
ins_len=3;allele-call-pos=2045476;reference=-;allele-call=GTA/GTA;rightmost-allele-call-
pos=2045478;rightmost-reference=-;rightmost-allele-call=AGT/AGT;context-pos=2045477;context-
reference-seq=gt;context-variant-seq=GTAGT
```

In this example there are two possibilities, `-/GTA` where the insertion is after position 2,045,476 or `-/AGT`, where the insertion is after 2,045,478. Following the left-most rule, the allele call and allele call position are reported as the lowest chromosome position. The full representation is reported here as `gt/GTAGT`.

```
> 167,chr1:2045476-2045476(),INSERTION,3,;allele-call-pos=2045476;allele-call=/GTA;allele-
pos=2045477;alleles=gt/GTAGT;allele-counts=REF,2
cacggcggtg---gttagggtcacggctgtag     2045467 2045495
1130330110---103200121130321132
  Ref   tg---gtta     2045475 2045479
        10---103
  Reads tGGTAGTTA
        10132103
  T331011013210320012113032 1
     T11101321032001211003211 32
```

Example 4, insertion of a short tandem repeat:

```
ins_len=3;allele-call-pos=5274096;reference=-;allele-call=-/TGT;
rightmost-allele-call-pos=5274100;rightmost-reference=-;rightmost-allele-call=-/GTT;
context-pos=5274097;context-reference-seq=tgtt;context-variant-seq=TGTTGTT
```

In this example there is a repeat in the sample that is not in the reference, so the full reference is `TGT/TGTTGT`, where `TGT` in the reference starts at position 5,274,097. The small indel caller trims this and then takes the position immediately before the insertion yields a position of 5,274,096 for the leftmost, and 5274100 for the rightmost representations.

```
> 532,chr1:5274096-5274096(),INSERTION,3,;allele-call-pos=5274096;allele-call=/TGT;allele-
pos=5274097;alleles=tgtt/TGTTGTT;allele-counts=REF,6
gttcccatctcctaactggggctaattatcatccctc---tgtttgagtgtttcgaggatgaattgag       5274060 5274124
10200132220230121000323030332132002 22---11001221110023220231203012 2
                              Ref   tc---tgtttg       5274095 5274101
                                    22---11001
                              Reads tCTGTTGTTTG
  T20132220230121000323030332132002221101100122111002
013222023012100032303033213200222110110012211100 21T
     1322202301210003230303321320022211011001221110232T
T2023012100032303033213200222110110012211100232202 3
               T10032303033213200222110110012211100232202312030122
               T10032303033213200222110110012211100232202312030122
```

Example 5, deletion of short tandem repeats:

```
del_len=4;allele-call-pos=24115702;reference=agag;allele-call=-/-;rightmost-allele-call-
pos=24115707;rightmost-reference=gaga;rightmost-allele-call=-/-;context-
pos=24115700;context-reference-seq=acagagagagaac;context-variant-seq=aCAGAGAAC
```

The AG repeat occurs 4 times in the reference and only twice in the sample, or more concisely, AGAGAGAG/AGAG. Because it is a deletion, the left-most position of this repeat is reported here at position 24,115,702.

```
> 2130,chr1:24115702-24115705(),DELETION,-4,;allele-call-pos=24115702;allele-call=agag/;allele-
pos=24115700;alleles=acagagagagaac/aCAGAGAAC;allele-counts=REF,5
agctctgattatgctactgcactccaggctgggtgacagagagagaaccttgacttgaaaaacaaaaCCCCaaaacacagat 24115665       24115746
2322212303313231213112201203210011211222222220102012120120000110001000100011112223
                            ref    acagagagagaac     24115700       24115711
                                   112222222201
                            reads  aC----AGAGAAC
  T22120331323121311220120321001111----2222010201212012
        T331323121311220120321001121----22220102012120120000011
                    1122012032100112l1----222201020121201200001100010001000100T
                           T2210011211----222201020121201200001100010001000010003
                           T2210011211----222201020121201
```

Example 6, an insertion/SNP combination variant:

```
allele-call-pos=5658680;reference=a;allele-call=a/CT
```

At position 5,658,680 the reference has an A. The sample however has a CT, so this complex variant is simultaneously a SNP and an insertion.

```
> 593,chr1:5658680-5658680(),INSERTION,1,;allele-call-pos=5658680;allele-call=a/CT;allele-
pos=5658680;alleles=atttt/CTTTTT/CTTTTA/NO_CALL;allele-counts=REF,2,1,1
gtgctgatcagtatttagctgaagactctggaga-tttttgttttgtgactttgtccttttc     5658647 5658707
113212321213300323212022122210223-000011000111212001120200002
                           ref    aga-tttttg    5658678 5658685
                                  223-00001
                           reads  aGCTTTTTTG
  221232121330032321202212221022320000011000111212031
          13300323212022122210223200000110001112120011202223T
   201121213300323212022122210223200033110001112120031
```

Example 7, multiple inserted alleles

```
ins_len=3;context-pos=60612131;context-reference-seq=A;context-
variant-seq=TAGA/TTGA/NO_CALL;
```

This example has two main inserted alleles, TAG and TTG.

```
TGGGGGATAAGGTGTTTATTAAGGATGACAGAAACCTCCTGAT---AGAGACAATATCATTCACCTTATAGATCCATCTCTG 60612088
10000233020111003303020231211220010220212330---22221103332130211020333223201322221
                                  Ref   TGAT---AG     60612127       60612131
                                        1233---2
                                        12---332
                                  Read1 TGATTTGAG
                                        12300122
                                  Read3 TGATTAGAG
                                        12303222
  T10233020111003303020231211220010220212300122222110
  T10233020111003303020231211220010220212300122222110
   T12330201110033030202312112200102202123032222221103
               330302023121122001022021230322222211033321302110231
                                  T30102202123032222221103333
                                  T30102202123032222221103332130211020333223201322222
                                        T012300122222110333213021l
```

FAQ – Small indels

1

Can the caller find SNP-Indel events?

Yes. These events, which are also referred to as replacements, are annotated by the allele-call tag in the GFF. For instance, allele-call=AT/G is an A/G SNP followed by a T deletion, which can equally be represented as an A deletion followed by a T/G SNP. Similarly, C/TGA is a C/T SNP followed by a GA insertion, which can also be represented as a TG insertion followed by a C/A SNP. There is no hard limit placed on the complexity of this variant (for example, double, triple, quad SNP, followed by an indel are all allowed), as long as the differences in the number of bases is within the gap size range contained in the BAM file, and the variable region is reported accurately by the aligner in the XA or XW BAM tag.

2

How can I speed up the small indel caller?

You can run the caller on different chromosomes at the same time making multiple INI files, one for every chromosome, and using the `small.indel.genomic.region` parameter to specify each specific chromosome.

3

Can I get the number of reference and variant reads used for each indel call?

Yes. These are directly reported as reference-reads and context-variant-reads tags in the output GFF3 file. Actual bead ids and read sequences are also reported for the variant reads in the GFF3 file using the bead_ids and read_seqs tags. Bead ids for the reference, ungapped reads are reported in the *.ungapped file.

Note that the LifeScope™ Software aligner is generally less sensitive to gap alignments than to ungapped alignments, so two or three times more ungapped alignments are to be expected for hemizygous indels. Also, this calculated number of reference alignments, along with all the other allele counts may include redundant (PCR optical duplicates) alignments. The number of non-redundant reads per allele call is currently not reported.

4

Can I perform gap alignments on all reads for greater sensitivity? Presently only those reads that do not match fully are considered for gap alignments.

The following parameters allow all alignments to be considered for gap alignments:

- For single tag: `mapping.gap.min.non.matched.length=0`
- For paired tags: `pairing.gap.min.non-matched.length=0`

However, processing with these settings could be significantly slower and may produce more false positives.

5

Can I use a legacy PAS file as input to the small indel caller?

All inputs files used with the small indel caller should be BAM files. Legacy files such as PAS and MATES files are not supported as they may give unexpected results, such as missing indel IDs or less accurate zygosity calling.

6

Why is a duplicate insertion or deletion reported?

Occasionally an indel is reported twice in the output file. This situation occurs when an indel is detected separately in two different locations, from different sets of reads. After further processing, the two locations are resolved to the same location. These two entries represent a single insertion or deletion.

7

What does "len=no_call or unknown" mean in small indel GFF output?

The message `len=no_call or unknown` occurs when a deletion is detected and also has reasonably good concordance with dbSNP, but the indel module is not able to determine the deletion size. This message mostly likely occurs with large (> 100bp) deletions.

14 Run a Large Indels Analysis

This chapter covers:

- Overview 275
- Examples of running a large indel analysis 276
- Large indel module parameters descriptions. 276
- Large indel internal parameters 278
- Large indel output files 278
- Large indel analysis algorithm 280
- Identify candidate indels. 281
- Assign statistical significance to candidate indels. 283
- Determine zygosity 284
- FAQ – Large indels. 289

Overview

The large indel module identifies deviations in clone insert size. These deviations indicate intrachromosomal structural variations compared to a reference genome. Insertions and deletions (indels) up to 100 kbp are inferred by identifying positions in the genome in which the pairing distance between mapped mate-pairs deviates significantly from what is expected at the given level of clone coverage.

The module creates a look-up table of standard deviations at each level of clone coverage. The table produces an asymptotic curve in which the minimum size of detectable indels at a given level of significance drops rapidly as the clone coverage increases. The look-up table is used to determine the significance of the deviation in average insert size at each position in the genome.

Regions of the genome that are significantly deviated are selected as candidate indels. Hierarchical clustering is used to segregate the clones into groups in which the difference in the sizes of all clones in a group is less than a specified range. Clusters with too few clones, as specified by the user, are removed and the candidates are assessed to determine if a homozygous or heterozygous population of deviated insert sizes remains. All clones deviated by more than 100 kbp are discarded. Clones from various libraries with various insert sizes contribute to a single indel call by combining the probabilities associated with the clones from each library.

The large indel module only accepts paired data as input.

Examples of running a large indel analysis

In a standard workflow

The following standard workflows provide examples of running the large indels module:

- genomic.resequencing.lmp
- genomic.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
lscope.sh shell -u username -w password
cd /projects
mk ecoli
cd ecoli
mk run1
cd run1
set workflow genomic.resequencing.pe
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
set reference hg19
# optionally change parameter defaults here
ls
run
```

See for Chapter 7, “Run a Standard Workflow Analysis” on page 111 for more information on running standard workflows. Chapter 6, “Run a Command Shell Analysis” on page 83 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an example of how to run the large indel module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, “Demo Analyses” on page 567 for information on running the large indel module as a standalone analysis.

Large indel module parameters descriptions

In order to change a parameter value in your analysis, use the `set param` shell command to replace the line `# optionally change parameter defaults here` in the example commands in “In a standard workflow”. For instance, to change the ploidy parameter, use this shell command:

```
set large.indel.ploidy 0  tertiary/large.indel.ini
```

Table 68 Large indel parameter descriptions

Parameter name	Default value	Description
library.type	matepair	Specifies the library type. Allowed values are: • matepair • pairedend
large.indel.bas.file	—	(*Optional*) If you have a BAS file and specify it with this parameter, the file is not regenerated. The BAS file contains metadata information about the data source(s) and relevant mapping/pairing statistics. The BAS format is a pseudo-standard file format for storing BAM file metadata. For details see this site: **ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/pilot_data/README.bas**
large.indel.max.clone.cov	1000	The maximum physical (clone) coverage for analysis. Loci with clone coverage above this threshold are not analyzed. Allowed values: Integers >= 3. **Note:** You can use this parameter in combination with the parameter large.indel.high.coverage to reduce false positives in high density genomes, for example, bacteria.
large.indel.min.coverage	3	The minimum physical (clone) coverage allowed. Loci with clone coverage below this threshold are not analyzed. Allowed values: Integers >= 1.
large.indel.ploidy	2	Determines zygosity and allele frequencies in eukaryotic genomes. Currently two scenarios are supported haploid (n=1) and diploid (n=2). Allowed values: • **1**: Haploid. • **2**: Diploid.
large.indel.max.insert.size	100000	Maximum insert size, in base pairs.
large.indel.call.stringency	high	Specifies the large indel call stringency. Automates parameter adjustment to the desired stringency level. Allowed values: • **highest**: Recommend when a very low false positive tolerance is allowed. • **high**: Increased filtering. • **medium**: Default values. • **low**: Very aggressive. Lower settings results in more indel calls, but with more false positives. Higher settings result in fewer indel calls, but with fewer false positives.
large.indel.min.pairing.quality	25 (for MP) 10 (for PE)	Paired reads below this threshold are ignored. Allowed values: Integers 0–100.
large.indel.p.value	1e-10	P-value threshold, that is, the raw probability of committing a Type 1 error incorrectly identifying a large indel. Allowed values: Floats 0.0–1.0.

Table 68 Large indel parameter descriptions *(continued)*

Parameter name	Default value	Description
large.indel.high.coverage	0 (off)	Eliminate clone coverage weighting. This option significantly reduces the number of false positives when analyzing very high coverage genomic data. (Very high coverage genomic data is typically greater than 1000x read coverage, and is common for bacterial genomes.) Allowed values: • **0**: No effect on the module. • **1**: Eliminate clone coverage weighting. The algorithm ignores coverage when calculating the structural significance of candidate indels.
Resource parameters		
memory.request	7gb	Memory request. Include the units gb in the setting.
java.heap.space	3000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.

Large indel internal parameters

Table 69 describes parameters which we do not recommend changing.

Table 69 Large indel module internal parameters.

Parameter name	Default value	Description
large.indel.run	1	Whether or not to run the large indel module. • **0**: Do not run the large indel module. • **1**: Run the large indel module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
large.indel.warnings	0	Whether or not to enable warning messages. Allowed values: • **0**: Do not enable warning messages. • **1**: Enable warning messages.

Large indel output files

This section provides descriptions of the large.indel.gff file created by a large indels analysis run (see Table 70).

Table 70 large-indels.gff3 file format description

Column Title[1]	Description	Example
Sequence ID	Chromosome name	chr11
Source	Tool name	AB SOLID Large Indel Tool
Type	Indel type	insertion, deletion, and so forth
Start Position	Estimated 5' breakpoint	1081331
End Position	Estimated 3' breakpoint	1084700

Table 70 large-indels.gff3 file format description *(continued)*

Column Title[1]	Description	Example
Score	Significance of the candidate Indel (p-value)	1e–10
Strand	—	—
Phase	—	—
Attributes[2]		
dev	Indel size, measured in base pair	46
avgDev[3]	Average deviation from the population average	–1.7198
Zygosity[4]	Results from pair partitioning	Homozygous
nRef[5]	Number of reference alleles	0
nDev	Number of deviated alleles	5
refDev[5]	Average deviation of the reference-allele pairs	0
devDev[3]	Average deviation of the deviated-allele pairs	–3.567
refVar[5]	Variance of the reference-allele pairs	0
devVar	Variance of the deviated-allele pairs	0.8972
beadIds[6]	Bead IDs providing support for the candidate indel	1806_975_1088,...

[1] **genome.ucsc.edu/goldenPath/help/customTrack.html#GFF**

[2] Semicolon-separated list of large indel module-specific field names followed by an equal '=' sign, for example dev=46

[3] Insertions have negative values

[4] Either homozygous, heterozygous, or double

[5] Value is always zero for homozygous indels

[6] Comma-separated list

Figure 22 shows an example of an output GFF file displayed in a spreadsheet.

	A	B	C	D	E	F	G	H	I	J
1	chr20	AB_SOLiD Large Indel Tool	deletion	13022198	13022445	2.79E-11	.	.	dev=361;avgDev=0.605991;	
2	chr20	AB_SOLiD Large Indel Tool	deletion	13924362	13924978	1.88E-12	.	.	dev=271;avgDev=1.09822;z	
3	chr20	AB_SOLiD Large Indel Tool	deletion	14000747	14000930	2.29E-12	.	.	dev=193;avgDev=0.78328;z	
4										
5										
6	**Attribute details**									
7	dev=361									
8	avgDev=0.605991									
9	zygosity=HETEROZYGOUS									
10	nRef=55									
11	nDev=62									
12	refDev=-0.361414									
13	devDev=1.46417									
14	refVar=0.529432									
15	devVar=0.605724									
16	beadIds=466_1704_1363,2199_554_1306...									
17										

Figure 22 large-indels.gff3 file example

Large indel analysis algorithm

Large indel detection is a tertiary module in LifeScope™ Software. Data from the pairing phase of the mapping module serve as direct inputs for large indel discovery (see Figure 23).

The large indel module uses pairing statistics are available in BAM headers of the input BAM files. If the pairing statistics are not available, the large indel module estimates the required parameters.

The large indel module processes inputs using an alignment window to hold and analyze sets of locus-spanning pairs. Regions with pairs that have significant insert size deviations are chosen as candidate indel sites, which are processed further to determine zygosity. The output of the analysis is one file in the GFF format. The GFF file contains information about the location, size, and significance of each large indel detected by the module.

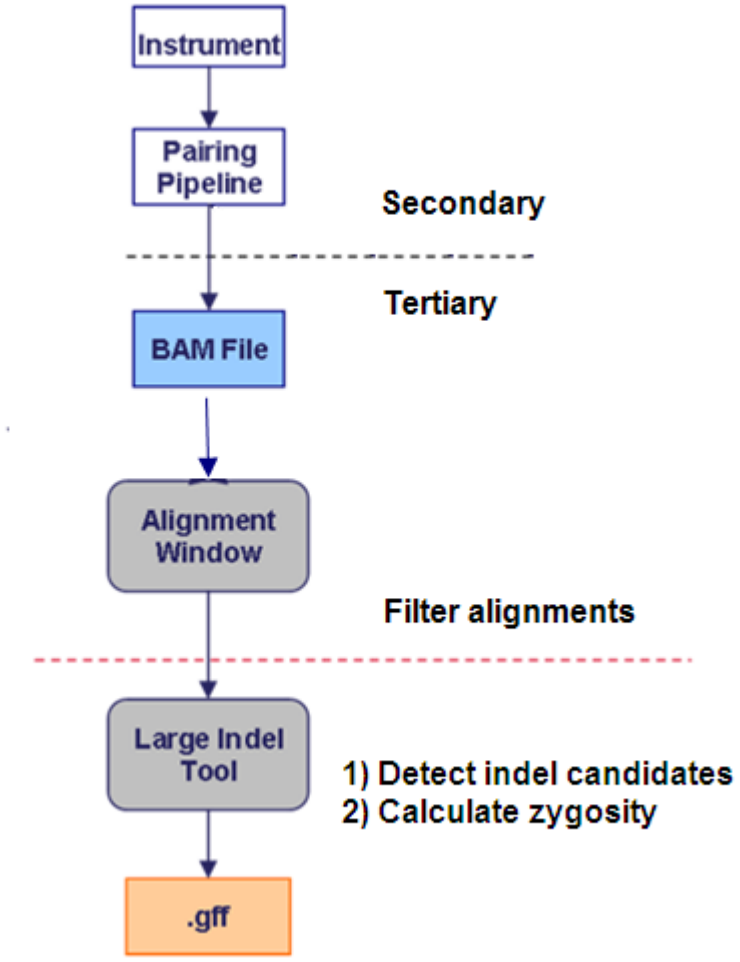


Figure 23 Large indel analysis module

This paragraph refers to Figure 23 on page 280. The large indel module accepts as input BAM files (blue) are generated during the secondary analysis module (mapping and pairing). Pairing statistics are included in the BAM headers. Pairing statistics provide more accuracy and can improve the final results of the large indel module. Final output consists of a GFF file (orange).

Identify candidate indels

Pairing distances (sometimes called insert sizes) for each pair are assigned during the mapping and pairing module and subsequently used by the large indel module to determine indel candidacy.

Note: Insert sizes can be non-unique in the case of multiple feasible mapping/pairing combinations. When insert sizes are non-unique, the primary (optimal) pair is chosen for large indel analysis.

Clones that have been mapped and paired to a reference genome are classified as either concordant or discordant (see Figure 24 on page 281). Concordant pairs are those with insert sizes (sometimes called inter-read distances) that are not significantly deviated from the expected insert size of the library as a whole. Discordant pairs have insert sizes that deviate significantly from the expected value. Discordant pairs containing a putative deletion appear larger when mapped to the reference. Pairs containing a putative insertion appear smaller. Multiple discordant pairs in close proximity provide evidence for a candidate indel within the covered region.

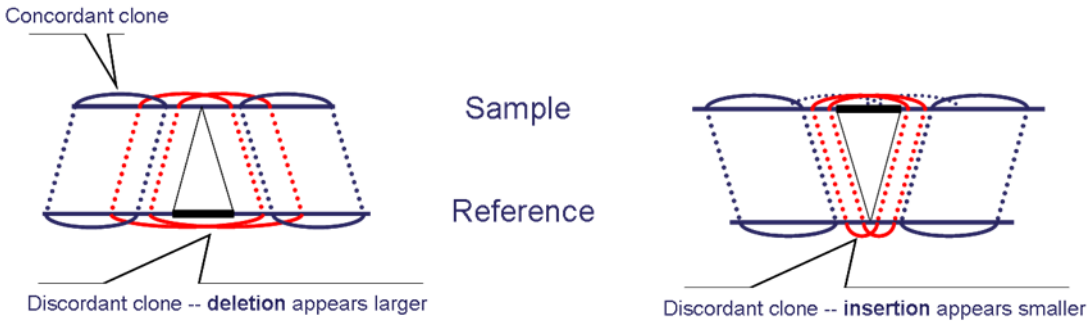


Figure 24 Discordant clones used to identify candidate insertions or deletions

In Figure 24:

Pairs spanning a candidate indel (red) appear distorted when mapped to a reference genome. Insert sizes appear larger for deletions (left) and smaller for insertions (right).

The module moves across individual chromosomes in order of genomic position to generate an alignment window of overlapping locus-spanning pairs (see Figure 25 on page 282). When the window encounters the first read in a pair, the corresponding alignment is incorporated into the alignment window. Simultaneously, several moving statistics, including the number of locus-spanning pairs as well as their average insert size and variance, are updated. When the window encounters the second read in a pair, the corresponding alignment is dropped and the moving statistics are updated appropriately. A genomic region (sometimes called a window) is considered to contain a candidate indel when the average insert size of the set of locus-spanning clones is significantly deviated from the average insert size of the library as a whole.

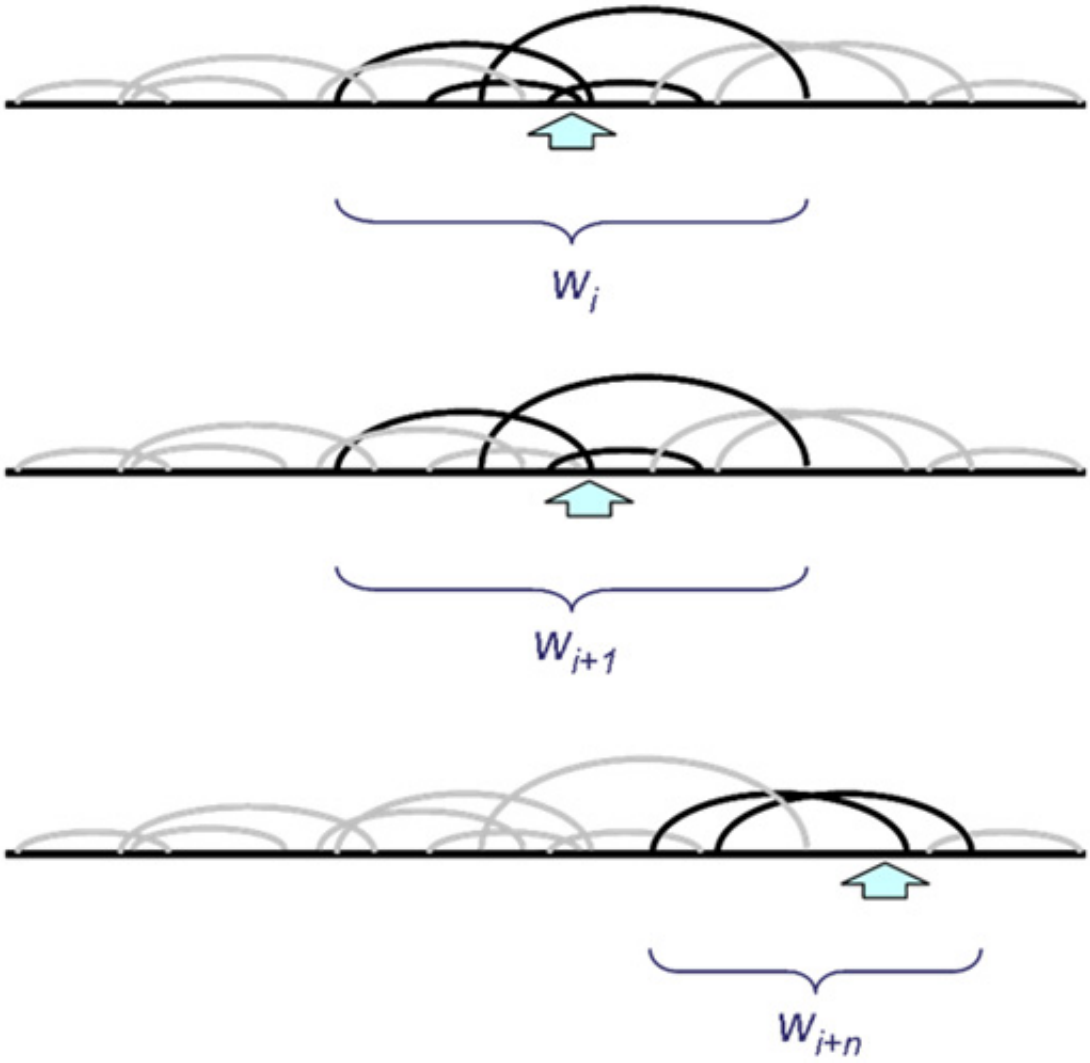


Figure 25 Sets of locus-spanning pairs

The following sections refer to Figure 25.

Top panel

An alignment window, *W*, contains the set of overlapping locus-spanning pairs (black loops) at a particular genomic position *i* (arrow) of the reference genome (black line).

Middle panel

The window advances to the next position by incorporating the next alignment *i+1*. The corresponding alignment is the second in the pair, so the alignment *i* is dropped from the alignment window and the alignment statistics are updated accordingly. When the window encounters the first alignment, the corresponding pair is added to the window.

Lower panel

As this process proceeds along the chromosome, pairs are added or dropped from the window, moving statistics are continuously updated, and regions with significant insert size deviations are detected and analyzed.

Note: Insert size variations are exaggerated for illustrative purposes.

Assign statistical significance to candidate indels

Regional insert size deviations can be calculated directly from the moving statistics associated with each clone window. The deviations that achieve statistical significance indicate relatively large structural variations compared to the reference genome (see Figure 26). Hypothesis testing determines the significance of deviations, where the null hypothesis asserts an insignificant difference between the local average insert size and the population average (H_o: $x = \mu$). Candidate deviations are chosen where the probability of falsely rejecting the null hypothesis in favor of the alternative (H_a: $x \mu$) falls below a user-defined confidence threshold.

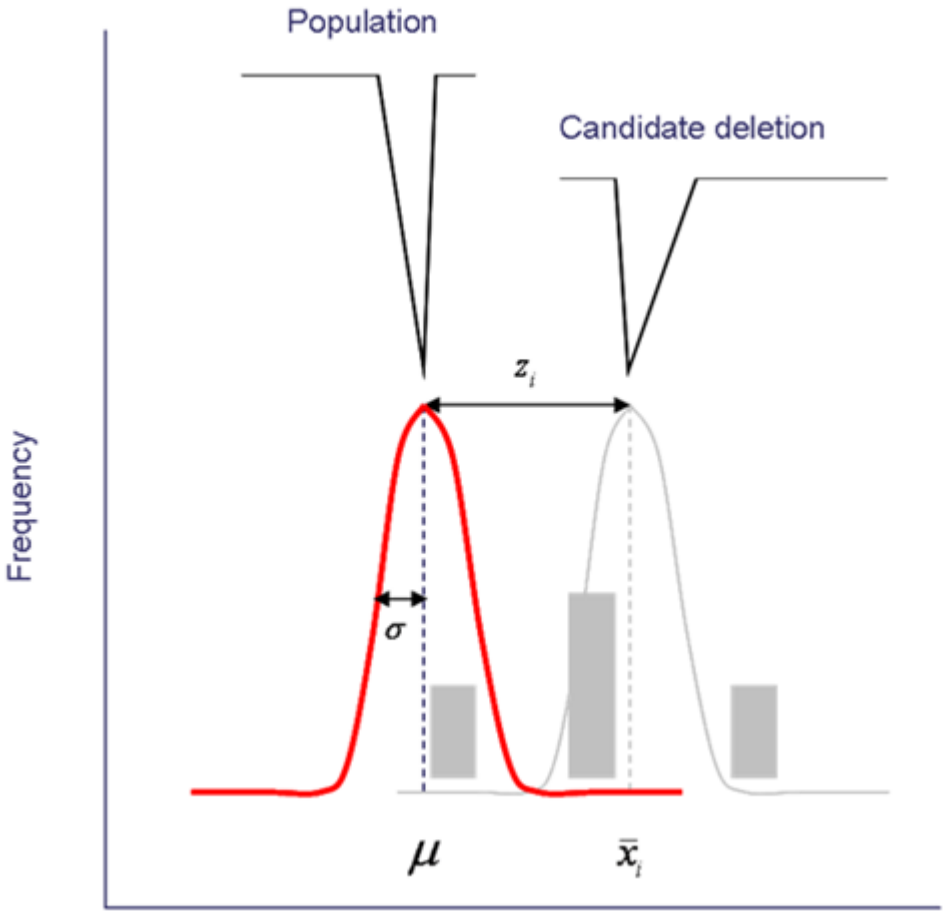


Figure 26 Hypothesis testing example

The following paragraph refers to Figure 26.

The population average insert size μ and standard deviation σ are calculated from the input BAM file. The alignment window contains a very small subset of pairs sorted by insert size (grey bars). Moving statistics including the number of locus-spanning pairs n_i as well as the sample average insert $\bar{x}_i$ size are calculated from this subset of pairs at each genomic position *i*. The parameters are used to z-normalize insert sizes according to $z_i = \frac{|\bar{x}_i - \mu|}{\sigma}$, which measures the absolute insert size deviation between the sample and the population in units of standard deviation. The normalization step allows multiple libraries with variable insert sizes to be combined into one analysis. A candidate indel is considered significant if $p(z_i|zn_i) < \alpha$ where probability values (p-values) are calculated according to the standard normal distribution and is a user-defined threshold parameter, where `large.indel.p.value` default is p=1e–10).

Determine zygosity

After a candidate indel is detected and deemed significant, the alignment window is partitioned to remove erroneous pairs because of mapping/pairing artifacts and to further characterize indel alleles and zygosity (see Figure 27 on page 285). Locus-spanning pairs are partitioned to remove two groups because polyploidy is currently not supported.

Each partition represents a disjoint subset of pairs from the alignment window optimally grouped by insert size so that pairs with similar insert sizes are placed in the same partition. Partitions with only one pair (sometimes called outliers) are removed from consideration. The candidate indel is also removed if the average insert size for pairs in the remaining partition are not significantly deviated from the population average. Removing the candidate indel can occur when mapping or pairing errors are responsible for observed insert-size deviations.

The number of pairs per partition and various summary statistics are calculated for each partition to determine alleles, allele frequencies, and zygosity. Candidate regions with minor allele frequencies less than one-third are removed from consideration.

The large indel module uses a heuristic method to categorize each candidate indel. If both partitions contain pairs with insert sizes that are significantly deviated from the reference but not from each other, the region is designated HOMOZYGOUS. If pairs are significantly deviated from the reference and from each other, the region is designated DOUBLE, which indicates two indel alleles of different types or sizes. DOUBLE regions can be placed into one of several indel/indel categories, for example, insertion/deletion. If one pair is deviated from the reference and the other pair is not, the region is designated HETEROZYGOUS, which indicates the presence of indel and reference alleles.

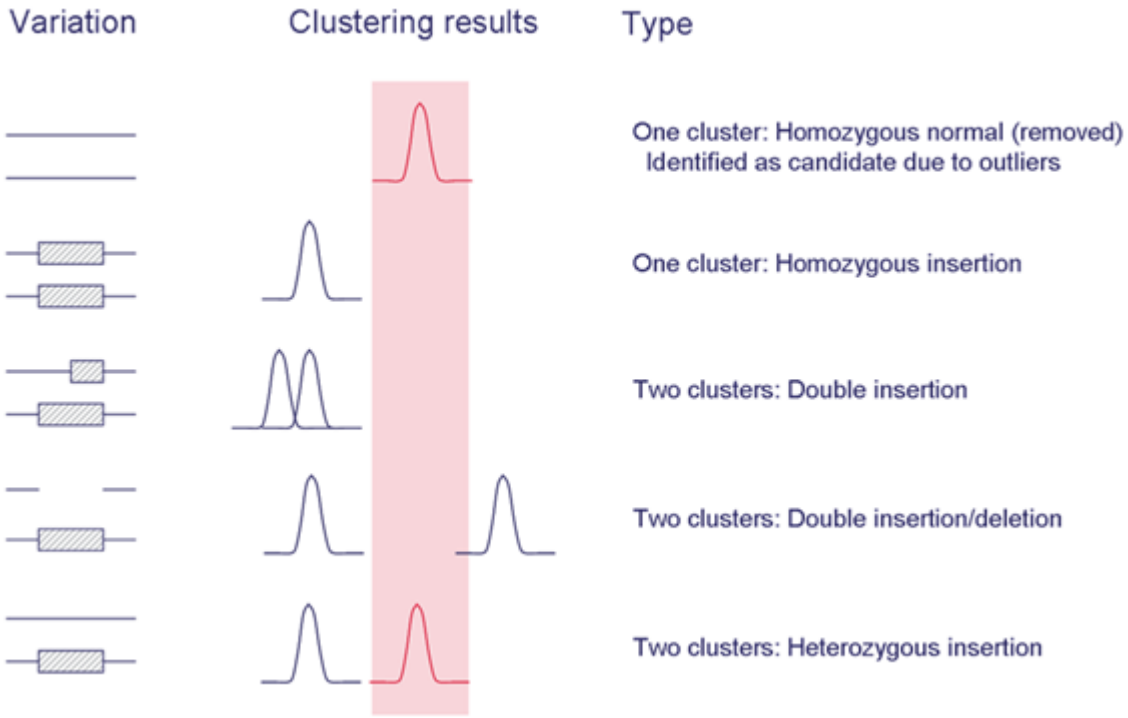


Figure 27 Partitioning the alignment window to determine zygosity

Left panel

Heuristic methods are used to characterize indel alleles including homozygous reference alleles (lines), which are removed from consideration; insertions (hatched boxes), and deletions (broken lines).

Middle panel

Each partition contains pairs with characteristic insert size distributions. Summary statistics describing the distributions are used to determine indel and reference alleles (black and red bell curves, respectively).

Right panel

Based on the results of the analysis, each candidate indel is assigned an appropriate zygosity category.

Filter alignments and parameter optimization

A user-defined pairing quality value (PQV) filtering threshold is set with the parameter `large.indel.min.pairing.quality`. The PQV threshold setting places constraints on which alignments are incorporated into the alignment window and subsequently used to determine large indel candidate regions. The default PQV threshold values are 25 for mate-pair data, and 10 for paired-end data. Adjusting the PQV threshold down (less than the default) to include lower-quality alignments generally improves sensitivity but increases the number of false positives. The opposite is true if you adjust the PQV threshold up (greater than the default).

Adjusting the p-value and PQV thresholds together might be required to optimize the module for analysis of data that differ significantly from normal HuRef samples, for example, non-human or cancer genomes. However, the default settings provide a convenient starting point for this process.

Additional optimization might be required for very high coverage samples. For example, alignments from high-density slides mapped to small prokaryotic genomes typically result in clone coverage values above 1000x. In the case of clone coverage values that are above 1000x, use the high-coverage flag (`large.indel.high.coverage` in LifeScope™ Software), which adjusts the p-value calculation to $p(z_i) < \alpha$, eliminating the conditional coverage parameterization (weighting), and reducing the number of false positives that might otherwise result.

Input files for large indel analysis

The only acceptable inputs for large indel analysis are BAM files containing paired-end or mate-pair data. Other alignment types are not compatible with the large indel module.

Note: LifeScope™ Software does not support combining paired-end and mate-pair data into a single analysis.

Interpreting results from the large indel module

Large indel results are written to a GFF file, which can be uploaded directly into the UCSC Genome Browser (**cbse.ucsc.edu/research/browser**) or other similar visualization tools. See Table 70 on page 278 for a description of the GFF file format. See Figure 28 on page 287 and Figure 29 on page 288 for examples of GFF files generated by the large indel module.

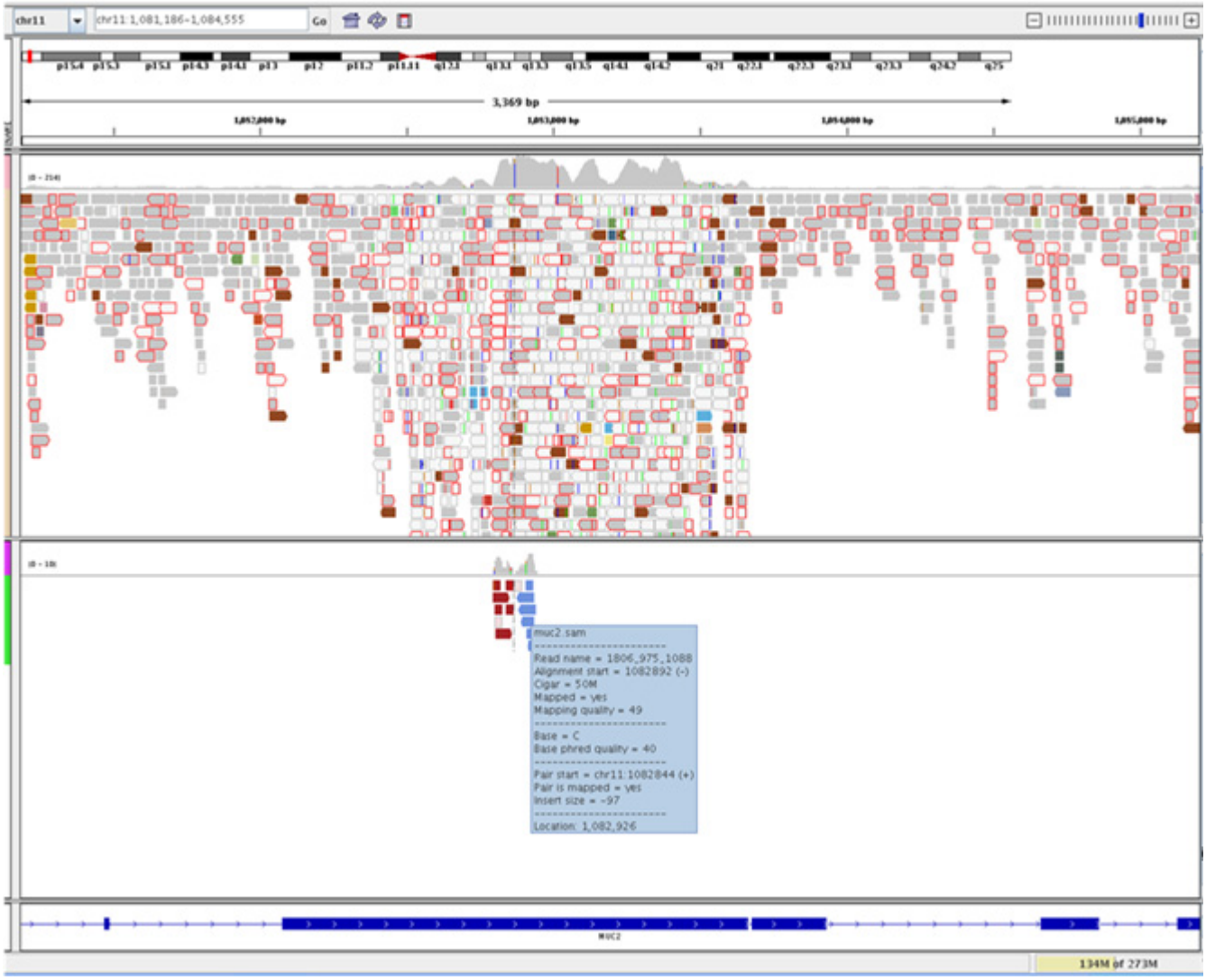


Figure 28 Identifying a 46bp insertion in *MUC2*

In Figure 28, paired-end data was analyzed using the SOLiD® mapping and pairing modules (against a human reference). The resulting BAM file was used for large indel analysis with default settings. The top panel uses the Integrated Genomics Viewer (IGV) to represent all BAM alignments covering the region chr11:1081331–1084700, which contains the partial coding region for the human mucin 2 precursor (MUC2) (bottom panel) as well as an indel previously identified by Levy et al., 2007 (not shown). Further information about IGV, including alignment color-encodings, is available at **broadinstitute.org.** The middle panel represents the subset of alignments used by the large indel module to identify a 46bp homozygous insertion breakpoint at the indicated position (dotted grey line). Forward and reverse strand reads are highlighted red and blue, respectively. The blue rectangle displays various alignment features, including a deviated insert size (97bp) that is significantly smaller than the average insert size of the population (170bp) providing evidence for an insertion at this site. The blue rectangle was generated by mousing over one of the reverse strand reads.

Figure 29 Identifying a 413bp deletion at *IL2RA*

In Figure 29, the top IGV panel represents long mate-pair alignments (mapped to HuRef) covering the region chr10:6135605-6139655, which contains an intron of human interleukin 2 receptor alpha (bottom panel) as well as several indels previously identified by Ahn et al., 2009; Bentley et al., 2008; Wang et al., 2008; Wheeler et al., 2008; and Levy et al., 2007 (not shown). Several alignment features are consistent with the presence of a deletion at this site, including six gapped alignments (dotted lines) flanking the putative deletion. The region between the six gapped alignments contains very few reads with high mapping quality, indicating sequence that is present in the reference but lacking in the sample. Surrounding the gapped alignments is the set of deviated pairs identified by the large indel module as supportive evidence for a 413bp homozygous deletion (middle panel). Reads with significantly deviated insert sizes, > 2000bp compared to the population average (1575bp), are highlighted (in pink).

FAQ – Large indels

1

Why are there so many more deletions than insertions?

Insertion detection is limited by the average insert size of the library, typically around 1500 base pair for mate-pairs. The average insert size of paired-end libraries is much smaller (around 150 base pairs), and only insertions smaller than the insert size can be detected.

2

What is the resolution of large indel detection?

This depends on clone coverage, insert size variability, statistical threshold, and library type. For example, mate-pair data detects large insertions and deletions ranging from 30 base pair to 1.2 kbp and 86 bp to 100 kbp, respectively, in human hg18 (McKernan et al, 2009). For paired-end data, deletion sizes range from 100 base pairs to 2 kbp, and insertions are essentially undetectable.

3

Why are there so many large indels around 300 base pair and deletions around 6 kbp?

Alu elements (SINEs) and LINEs, respectively (McKernan et al, 2009).

4

How many large indels can I expect?

The quantity of large indels to expect depends on:

- Genome size
- Clone (read) coverage
- Average insert size
- Significance threshold
- Pairing quality value threshold
- Library type

Note: The Human hg18 has 4075 deletions and 1515 insertions.

5

Why do pooled samples take so long to analyze?

In areas of higher coverage, more significant candidate regions are found, causing more clustering. Analysis at CNVs can slow down as the coverage exceeds 2x.

6

How long does the module take to run and how much space is required?

Significant algorithmic improvements have cut processing time. Additional parameters and better implementation have reduced lag times associated with too much coverage and zygosity calculation (clustering).

- Non-parallelized human genome runs consisting of approximately 500 million reads typically take two to four hours to run, depending on platform and resource load.
- Running jobs in parallel at one job per chromosome, significantly reduces run time. Intermediate file generation is negligible for BAM file inputs.

15 Add Genomic Annotations to Analysis Results

This chapter covers:

- Overview 291
- Workflows 295
- Annotation sources. 296
- Use custom reference and custom dbSNP files 297
- Annotation input files 299
- Prepare to run annotation processing 300
- Annotation parameters 301
- Output annotation 303
- Annotation output files 307
- About annotations and LifeScope™ Software modules 322
- FAQ - Annotations 323

Overview

The annotations module is an optional post-processing step available with select LifeScope™ Genomic Analysis Software analysis modules. The annotations processing makes a copy of the analysis module's GFF file and adds new attributes to the GFF entries in the copied file. The new attributes are taken from information in publicly available sources about the variants in the input, about features intersecting the variants in the input file, or about any biological function potentially changed by the variants.

Annotations processing does not modify the original LifeScope™ Software analysis module's output file.

Several output files are potentially generated by the annotations processing, depending on the analysis modules involved. One annotations output file contains all of the content of the GFF file generated by the modules, with added attributes for annotations. Another annotations output file is a subset of the previous file, and it contains only those entries that pass filtering requirements that you specified. Annotation processing also produces output files containing only mutated genes, or only SNPs. Several statistics files are also produced.

Two sources of annotation are used:

- A UCSC or Ensembl GTF file, used to determine whether a variant overlaps a gene or exon.
- The National Center for Biotechnology Information (NCBI) dbSNP database, which contains information on SNPs and indels already found by other studies. We use the following fields of the of the dbSNP entries:
  - The reference SNP identifier (rsID)
  - The functional code
  - The corresponding locus ID (the gene)

If a conflict is found between the dbSNP data and the GTF data, both annotations are reported. The annotation module does not attempt to resolve these conflicts.

Annotation sources can be either the refGene GTF file with respect to the human hg18 reference build and the dbSNP 130 build or the human hg19 reference build and the dbSNP 132 build (both available with LifeScope™ Software), or a source that you provide. Only GTF files and dbSNP data can be used as sources. The data from these three tables in dbSNP is required:

- SNPChrPosOnRef
- SNPContigLoc
- SNPContigLocusId

Annotations processing supports these annotation types:

- DNA features: genes and protein-coding features
- Verified variants in existing database

Post-processing types include:

- Annotate genomic variants with the annotation types that apply
- Filter based on annotation
- Report a list of gene and protein-coding annotations involved the variants
- Report statistics

The annotations module is available with the following modules:

- Resequencing:
  - SNPs
  - Small indels
  - Large indels
  - Human CNV
- Targeted resequencing:
  - SNPs
  - Small indels
- Low frequency variant:
  - SNPs

Input file handling

Annotations processing uses the GFF output of an analysis module, but does not modify the GFF file. Annotations are added to a copy of the GFF file.

Filters

Annotation processing optionally generates an output file containing only those entries that fulfill *all* the conditions you specify with annotation filtering options. Annotation filters are described in Table 71.

By default, the filtering options report all variants (regardless of whether or not they appear in dbSNP and whether or not they occur in exons or genes).

Table 71 Annotation filtering options

Filtering option	Default
Report *only* variants in exons	Off
Report *only* variants in genes	Off
Report *only* variants that appear in dbSNP	Off
Report *only* variants that do not appear in dbSNP	Off

Table 72 shows the annotation filtering supported in select LifeScope™ Software modules.

Table 72 Support for annotation filtering in each variant type

Annotation type	SNP	CNV	Small indel	Large indel
geneID	Yes	Yes	Yes	Yes
exonID	Yes	Yes	Yes	Yes
rsID	Yes	—	Yes	—
functionCode	Yes	—	—	—

Statistics

Table 73 lists the statistics included in the annotations output files and the LifeScope™ Software modules that support these annotations. See Table 73 for information about SNPs transitions and transversions.

Table 73 Statistics generated in annotations output files

Analysis type	Statistics
All supported modules (SNPs, CNVs, Indels)	Number of variants Number of variants per chromosome Number of heterozygous variants Number of homozygous variants Number of heterozygous SNPs per chromosome Number of homozygous SNPs per chromosome
SNPs	Number of heterozygous SNPs that are transitions, transversions Number of homozygous SNPs that are transitions, transversions (compared to the reference)
Indels	Indel variant length distribution (negative for deletion, positive for insertion)
CNVs	Copy number distribution CNV length distribution

Table 73 Statistics generated in annotations output files *(continued)*

Analysis type	Statistics
Annotations from dbSNPs	Number of SNPs or indels in dbSNP Number of homozygous SNPs or indels in dbSNP Number of heterozygous SNPs or indels in dbSNP Overall dbSNP concordance (percentage of SNPs or indels in dbSNP) Heterozygous dbSNP concordance (the percentage of heterozygous SNPs or indels found in dbSNP) Homozygous dbSNP concordance (the percentage of homozygous SNPs or indels found in dbSNP)
Annotations from GTF file content	Number and percentage of variants in exons Number and percentage of heterozygous variants in exons Number and percentage of homozygous variants in exons Number and percentage of variants in genes Number and percentage of heterozygous variants in genes Number and percentage of homozygous variants in genes

Table 74 describes how transitions and transversions are determined based on the reference allele and the diBayes output genotype.

Table 74 Transitions and transversions

diBayes genotype	Reference allele | second allele	A	C	G	T	Other
A	A	0	TV	TS	TV	0
C	C	TV	0	TV	TS	0
G	G	TS	TV	0	TV	0
T	T	TV	TS	TV	0	0
M	A|C	TV	TV	TV	TV	TV
R	A|G	TS	TS	TS	TS	TS
W	A|T	TV	TV	TV	TV	TV
S	C|G	TV	TV	TV	TV	TV
Y	C|T	TS	TS	TS	TS	TS
K	T|G	TV	TV	TV	TV	TV
	Other	0	0	0	0	0

Workflows

Annotation is integrated into the resequencing, targeted resequencing, and low frequency variant standard workflows. Table 75 shows the modules with integrated annotation support, in these workflows.

Table 75 Standard workflows and analysis modules with integrated annotation support

Workflow	Library type	SNP	CNV	Small indel	Large indel
Genomic resequencing	Fragment	Yes	Yes	Yes	—
Genomic resequencing	Mate-pair	Yes	Yes	Yes	Yes
Genomic resequencing	Paired-end	Yes	Yes	Yes	Yes
Targeted resequencing	Fragment	Yes	—	Yes	—
Targeted resequencing	Paired-end	Yes	—	Yes	—
Low frequency variant	Fragment	Yes	—	—	—
Low frequency variant	Paired-end	Yes	—	—	—

Example shell commands

The following are example shell commands to run an analysis with annotations. This example uses reads and references from the LifeScope™ Software repository.

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.frag
# define your input data
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
# specify the reference to be used
set reference hg18
# optionally change parameter defaults after this line or
# set analysis-specific parameters
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
```

Because the `set reference hg18` command uses a known assembly, reference-related parameters including annotation parameters are set automatically. (Confirm these settings with the LifeScope™ Software shell `ls` command before starting your analysis.)

To specify your own annotations files, use the `set param` command, as shown in these examples:

```
set annotation.dbsnp.file /path/to/dbsnp/file tertiary/INIfile
set annotation.gtf.file   /path/to/gtf/file tertiary/INIfile
```

Place your `set param` commands after the set reference command (after the line `# optionally change parameter defaults after this line`). The INI file to use with the `set param` command depends on the type of LifeScope™ Software analysis you are running. Replace the string *INIfile* with one of these INI file names:

- `annotation.cnv.ini`
- `annotation.dibayes.ini`
- `annotation.large.indel.ini`
- `annotation.small.indel.ini`

Annotation sources

Table 76 lists common annotation sources supported by LifeScope™ Software. Files required to support certain annotations are available with LifeScope™ Software. These files support the following:

- Annotations based on the hg18 and the dbSNP 130 build
- Annotations based on the hg19 and the dbSNP 132 build

  Note: The dbSNP 132 file used is a VCF file containing 1000 Genome data.

Table 76 Annotation sources

Type of annotation	Location and notes
Variants in existing databases	NCBI Variation dbSNP resources: includes single nucleotide polymorphisms, microsatellites, and small-scale insertions and deletions. dbSNP contains population-specific frequency and genotype data, experimental conditions, molecular context, and mapping information for both neutral polymorphisms and clinical mutations. (This text is taken from NCBI website http://www.ncbi.nlm.nih.gov/guide/variation.) **http://www.ncbi.nlm.nih.gov/snp**
Genes and protein-coding features **Note:** Supported by LifeScope™ Software only if the data is downloaded in GTF format.	RefGene data from UCSC: **ftp://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz**
	The European Bioinformatics Institute's Alternative Splicing Database Project: **http://www.ebi.ac.uk/asd**

Note: While every effort is made to provide correct web addresses, Applied Biosystems is not responsible for the changes to these addresses, nor for the data provided by these locations.

Use custom reference and custom dbSNP files

Note: LifeScope™ Software supports only one format for GTF files. Public sources may have a variety of formats. To use custom GTF files, it may be necessary to download and modify these files into our supported GTF format. This section contains instructions to convert files into the supported GTF format

This section describes the steps required to use a custom reference with LifeScope™ Software. These steps are an alternative to using the reference-related files available with LifeScope™ Software.

Two main processes are required (listed here and described below):

1. Download and convert or reformat a GTF file. Choose one of the following:
   - Download and convert a UCSC GTF file.
   - Download and reformat an Ensembl GTF file.
2. Download and unzip dbSNP files.

The UCSC GTF file

Follow either the steps in this section or in the section "The Ensembl GTF file" on page 297.

After download, a conversion step is required to normalize the file by gene ID as required by LifeScope™ Software. Genome annotations that are downloaded from the UCSC Genome Browser and converted by the annotation conversion script are optimal because they contain Human Genome Organization (HUGO)-style gene names. HUGO-style gene names allow interpretation when using a genome browser or reading reports.

The annotation conversion script `refgene2gff.sh` works with the most recent format of `refGene.txt` files (as of the time of this writing). Some assemblies such as the rat genome use an alternate format for the `refGene.txt` file. The `refgene2gff.sh` script does not convert these alternate formats.

Follow these steps to download and convert the UCSC GTF file:

1. Download the file from the following URL:
   ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/refGene.txt.gz
2. Unzip the file with the following command:
   ```
   gunzip refGene.txt.gz
   ```
3. Use this command to convert the file to GTF format:
   ```
   refgene2gff.sh -i refGene.txt -o refGene.gtf
   ```
4. Specify the file, `refGene.gtf`, with the `annotation.gtf.file` parameter.
   (If you use the `set reference` *`assembly`* command and specify a known assembly, the `annotation.gtf.file` parameter is automatically set.)

The Ensembl GTF file

Follow either the steps in this section or in the section "The UCSC GTF file" on page 297.

Ensembl GTF files have the following characteristics that apply to their use with LifeScope™ Software.

- They are properly normalized by gene and transcript IDs.

- They use gene accession numbers instead of HUGO-style gene names.
- They use unprefixed sequence identifiers, such as 1, 2, 3, ….X, Y, MT.
- They are incompatible with genome reference FASTA files that have UCSC-style sequence IDs with the prefix "chr", for example, chr1, chr2, chr3, ….chrX, chrY, chrM.
- In order to be used as input for annotations with LifeScope™ Software, they must be reformatted to use UCSC-style gene IDs.

Follow these steps to download and reformat the Ensembl GTF file:

1. Download the file from the following URL:

   http://www.ensembl.org/

2. Reformat the GTF file with the `reformat_ensembl_gtf.pl` command, and redirect the output to a new file with `.reformatted` in the file name.

   `For Example:`

   ```
   reformat_ensembl_gtf.pl Homo_sapiens.GRCh37.63.gtf > \
       Homo_sapiens.GRCh37.63.reformatted.gtf
   ```

   The input GTF file name typically includes the NCBI build number and the Ensmbl release number in its name, according to the following pattern:

   `Homo_sapiens.<reference_ncbi_build_no>.<Ensembl_release_no>.`
   `.gtf`

3. Specify the reformatted file with the `annotation.gtf.file` parameter.

   (If you use the `set reference assembly` command and specify a known assembly, the `annotation.gtf.file` parameter is automatically set.)

The dbSNPs files

Annotations based on dbSNPs require the data from these three dbSNPs tables: SNPChrPosOnRef, SNPContigLoc, SNPContigLocusId. LifeScope™ Software is verified to work with the format of these files as of February 2011. The dbSNP files follow this naming pattern:

`b<dbSnp_build_no>_<table_name>_<reference_ncbi_build_no>.bcp.gz`

For example, for build 131 and reference hg19 (NCBI build 37.1), the files required are shown in Table 77:

Table 77 Example files for dbSNP

dbSNPs table	File name
SNPChrPosOnRef	b131_SNPChrPosOnRef_37_1.bcp.gz
SNPContigLoc	b131_SNPContigLoc_37_1.bcp.gz
SNPContigLocusId	b131_SNPContigLocusId_37_1.bcp.gz

Follow these steps to use hg19 dbSNPs files with LifeScope™ Software annotations:

1. Download the dbSNP files from this site:

   http://www.ncbi.nlm.nih.gov/snp

2. Specify the information from these files in your analysis with the `annotation.dbsnp.file` parameter.

(If you use the `set reference assembly` command and specify a known assembly in the reference repository, the `annotation.dbsnp.file` parameter is automatically set.)

3. Increase the memory and heap space for your analysis, if necessary. See "Set your memory requests" on page 301 for instructions.

"Annotation parameters" on page 301 provides more instructions on running annotations.

Annotation input files

This section describes the input files used during annotation processing.

GFFv3 variant file

This file is the output file of a previous analysis module such as SNPs, small indels, large indels, or CNV. The variant file must be in GFFv3 format.

GTF

The input genome annotation file is in GTF format. Refer to the following websites for information on the GTF file format:

genes.cse.wustl.edu/GTF2.html

genome.ucsc.edu/FAQ/FAQformat.html#format4

The GTF file must match the genome reference to ensure that the analysis modules work correctly.

IMPORTANT! Make sure that the GTF file is for the same genome assembly as the genome FASTA reference file, and that matching sequence identifiers are used. Gene and transcript identifiers in the GTF file must be properly normalized. Identifier normalization is a known issue in GTF files from the UCSC Genome Browser, which is a popular source of GTF-formatted annotation. The UCSC GTF files always report the same value for gene and transcript IDs.

Use the LifeScope™ Software scripts to transform a GTF file into the format required for use with LifeScope™ Software modules. (See "The UCSC GTF file" and "The Ensembl GTF file" on page 297 for more information.)

dbSNP

Data downloaded from the publicly-available dbSNP database is included in the LifeScope™ Software reference repository. This data is based on human reference builds hg18 dbSNP version 130 and hg19 dbSNP version 132.

LifeScope™ Software automatically determines your dbSNP data file from the assembly name you specify, when you use the `set workflow assembly` command. For example:

```
set workflow hg18
set workflow hg19
```

You also have the option to specify a different dbSNP database (that you must supply) for your annotations. See "Add new reference files" on page 547 in Appendix E, The Reference Repository.

The following information is parsed from the dbSNP data:

- The refSNP identifier, rsID
- Chromosome number
- Chromosome start coordinate
- Chromosome end
- Whether or not the variant is a SNP
- Function code (consequence to transcript, if any)
- The gene name, locus_symbol (if available)

hg19

The hg19 dbSNP file is `dbSNP_b132_00-All.vcf`.

hg18

Table 78 describes the NCBI hg18 files used for annotations. These are a set of three tab-separated files.

Table 78 dbSNP hg18 annotation files

File	Description	Fields used
b<*dbSNP build number*>_ SNPChrPosOnRef_<*reference build version*>.bcp	A 4-column, tab-separated file, corresponding to the table SNPChrPosOnRef	The first 3 columns: • The rsID • Chromosome • Position
b<*dbSNP build number*>_ SNPContigLocusId_<*reference build version*>.bcp	A 21-column, tab-separated file, corresponding to the table SNPContigLocusId	• snp_id: The rsID • fxn_class: The function class code • (*Optional*) locus_symbol: The gene name
b<*dbSNP build number*>_ SNPContigLoc_<*reference build version*>.bcp	A 22-column, tab-separated file, corresponding to the table SNPContigLoc	• snp_id: The rsID • asn_from, asn_to: Used to determine the feature length • loc_type: Used to determine whether an entry is a SNP or an indel

Prepare to run annotation processing

Select the required input files

This section only applies to references *not* in the reference repository.

If you supply your own annotation source files, you must know the absolute path to those files.

Note: This section does not apply if you specify the human hg18 or hg19 assembly, because this option uses the annotation sources available with LifeScope™ Software.

To include gene and exon annotations, you must know the absolute path to:

- Your GTF file for gene and exon annotations

To include dbSNP annotations, you must know the absolute path to your dbSNP annotation file or files. Your dbSNP annotations might be in a single VCF file, or in a set of three BCP files:

- b<*dbSnp_build_no*>_SNPChrPos_<*reference_ncbi_build_no*>.bcp
- b<*dbSnp_build_no*>_SNPContigLoc_<*reference_ncbi_build_no*>.bcp
- b<*dbSnp_build_no*>_SNPContigLocusId_<*reference_ncbi_build_no*>.bcp

You control which types of annotation are generated:

- To turn off gene and exon annotations, do not provide a GTF file.
- To turn off dbSNP annotations, do not provide the dbSNP files.

Note: If any one of the three required dbSNPs files is missing from a set of three BCP files, no dbSNPs annotation is generated.

Set your memory requests

The memory requirements for dbSNPs annotations vary by reference build and dbSNPs build. Table 79 lists the hardware memory requirements for dbSNPs annotations for supported hg and dbSNPs builds.

Table 79 Hardware memory requirements for dbSNPs annotations

	hg18	hg19
dbSNP 130	22GB	Not supported
dbSNP 131	Not supported	22GB
dbSNP 132	Not supported	22GB

These parameters must be added to your module's properties file. Parameters in a module properties file affect all runs of that module, for all users.

These requirements are for full annotations (dbSNPs as well as exon-based annotations). The annotation module requires considerably less memory when run with exon annotation only (that is, with a GTF file but without a dbSNP file). However, the requirement increases linearly with the number of exons in the GTF file.

Complete the prerequisites

Additional prerequisites specific to each type of analysis are described in the instructions for each analysis module.

Annotation parameters

Table 80 describes the annotation parameters used with LifeScope™ Software analyses.

For filtered output files, all annotation filter options that are turned on must be fulfilled by the variant entry, for the variant to appear in the output file.

Table 80 Annotation parameters description

Parameter name	Default value	Description
Annotation source parameters		
annotation.dbsnp.annotate.snp **Note:** applies only to SNPs and small indels modules	0	Whether or not to annotate variants with SNP entries from the dbSNP file. Allowed values: • **0**: Do not annotate variants with SNP entries from the dbSNP file. • **1**: Annotate variants with SNP entries from the dbSNP file.

Table 80 Annotation parameters description *(continued)*

Parameter name	Default value	Description
annotation.dbsnp.annotate.indel **Note:** applies only to SNPs and small indels modules	0	Whether or not to annotate variants with indel entries from the dbSNP file. Allowed values: • **0**: Do not annotate variants with indel entries from the dbSNP file. • **1**: Annotate variants with indel entries from the dbSNP file.
annotation.indel.border.slack	5	An indel is considered to be matched to a dbSNP indel if the two overlap or if the distance between them is less than or equal to the value of annotation.indel.border.slack. Allowed values: Integers >= 0.
Annotation filtering parameters		
annotation.show.only.variants.in.exons	0	When set to 1, filters the annotated output: restricts the Filtered Variant annotation output file (see "Annotation output files" on page 307) to only the variants that overlap any exon. Allowed values: • **0**: Do not filter the output for variants that overlap any exon. • **1**: Restrict output to variants that overlap any exon.
annotation.show.only.variants.in.genes	0	When set to 1, filters the annotated output: restricts the Filtered Variant annotation output file (see "Annotation output files" on page 307) to only the variants that overlap any gene, even if it does not overlap any exon. Allowed values: • **0**: Do not filter the output for variants that overlap any gene. • **1**: Restrict output to variants that overlap any gene. **Note:** if the parameter annotation.show.only.variants.in.exons is set to 1, then the setting for the parameter annotation.show.only.variants.in.genes is ignored.
annotation.show.only.variants.in.dnsnp **Note:** applies only to SNPs and small indels modules	0	When set to 1, filters the annotated output: restricts the Filtered Variant annotation output file (see "Annotation output files" on page 307) to only SNPs or small indels that exist in dbSNP. Allowed values: • **0**: Do not filter the output for variants that *appear* in dbSNP. • **1**: Restrict output to SNPs or small indels variants that *appear* in dbSNP.
annotation.show.only.variants.not.in.dnsnp **Note:** applies only to SNPs and small indels modules	0	When set to 1, filters the annotated output: restricts the Filtered Variant annotation output file (see "Annotation output files" on page 307) to only SNPs or small indels that do not exist in dbSNP. Allowed values: • **0**: Do not filter the output variants that *do not appear* in dbSNP. • **1**: Restrict output to SNPs or small indels variants that *do not appear* in dbSNP.
Resource parameters		
memory.request	22gb	Memory request. Include the units gb in the setting.

Table 80 Annotation parameters description *(continued)*

Parameter name	Default value	Description
java.heap.space	21500	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.

Annotations parameters are by convention defined in INI files with the following names and location:

- `<analysis>/tertiary/annotation.cnv.ini`
- `<analysis>/tertiary/annotation.dibayes.ini`
- `<analysis>/tertiary/annotation.large.indel.ini`
- `<analysis>/tertiary/annotation.small.indel.ini`

Table 81 describes annotations parameters that are automatically specified by LifeScope™ Software when run as part of a standard workflow.

Table 81 Annotations internal parameter description

Parameter name	Default value	Description
annotation.run	1	Specifies whether or not to add annotations. Allowed values: • **0**: Do not generate annotations. • **1**: Generate annotations during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
annotation.dbsnp.file	—	The path to your input dbSNP file or files.
annotation.gtf.file	—	The path to your input genome annotation file.
annotation.output.file.name.stem	The base name of the input GFF file (without directory path and without file extension).	The common prefix for all output file names. Supplied by the system when run in a standard workflow.

Output annotation

This section describes the new attributes that annotation adds to the GFF entries from the input file. The added information comes from publicly-available sources about variants in the input, features intersecting the variants in the input file, or biological function potentially changed by the variants.

Table 82 describes the new attributes added as output annotation.

Table 82 New attributes added as annotations

Attribute		Description
geneID		Adds a geneID attribute to applicable entries in the output GFF file. The following apply to the geneID attribute: • The attribute is valid for all variants. • The attribute's value is the gene descriptor taken from the input GTF file. • If there are multiple overlapping genes, the attribute has multiple values (in a comma-separated list).
	Supported modules	**Requirement to qualify for annotation**
	SNPs Small indels Large indels CNVs	The variant must be within a gene (including introns and UTRs).
exonID		Adds exonID attributes to applicable entries in the output GFF file. The following apply to the exonID attribute: • The attribute is valid for all variants. • The attribute's value is of the form <*geneID*>-<*Exon#*>, where *Exon#* is the number of the exon counted within the gene (not within each transcript) in the transcription direction. • If there are multiple overlapping exons, the attribute has multiple values (in a comma-separated list). • If two exons start from the same position, the shorter one is listed first.
	Supported modules	**Requirement to qualify for annotation**
	SNPs Small indels Large indels CNVs	The variant must overlap an exon.
rsID		Adds rsID attributes to applicable entries in the output GFF file. The following apply to the rsID attribute: • The attribute is valid for SNPs and Indels. • The attribute's value is the rsID value of the dbSNP entry. • The attribute cannot have multiple values.
	Supported modules	**Requirement to qualify for annotation**
	SNPs	The SNP must appear in dbSNP.
	Small indels	Indels must appear in dbSNP and also must be labeled as InDel in dbSNP.

Table 82 New attributes added as annotations *(continued)*

Attribute		Description
functionCode		Based on functional codes from dbSNP, annotate SNP entries in the output GFF file. The following apply to the SNPType attribute: • The attribute is valid for only for SNPs. • The attribute's value is a pair in which the first value is the locus ID and the second the function code from dbSNP. Function codes are listed in Table 83. • The attribute can have multiple values, one for each SNP type that applies to the variant.
	Supported modules	**Requirement to qualify for annotation**
	SNPs	Must be a SNP.
	Small indels	Must be a small indel.

Table 83 describes function codes for refSNPs in gene features. This table is taken from the dbSNP website

:**http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=handbook&part=ch5**

Table 83 Function codes for refSNPs in gene features

Functional class	Description	Database code
Locus region	Variation is within 2 kbp 5′ or 500 bp 3′ of a gene feature (on either strand), but the variation is not in the transcript for the gene. This class is indicated with an "L" in graphical summaries. **Note:** As of build 127, function code 1 has been modified into a *two-digit code* that more precisely indicates the location of a SNP. The two-digit code has function code 1 as the first digit, which keeps the meaning as described above, and 3 or 5 as the second digit, which indicates whether the SNP is 3′ or 5′ to the region of interest. See function codes 13 and 15 in this table.	1
Coding	Variation is in the coding region of the gene. This class is assigned if the allele-specific class is unknown. This class is indicated with a 'C' in graphical summaries. **Note:** This code was *retired* as of dbSNP build 127.	2
Coding-synon	The variation allele is synonymous with the contig codon in a gene. An allele receives this class when substitution and translation of the allele into the codon makes no change to the amino acid specified by the reference sequence. A variation is a synonymous substitution if all alleles are classified as contig reference or coding-synon. This class is indicated with a 'C' in graphical summaries.	3

Table 83 Function codes for refSNPs in gene features *(continued)*

Functional class	Description	Database code
Coding-nonsynon	The variation allele is nonsynonymous for the contig codon in a gene. An allele receives this class when substitution and translation of the allele into the codon changes the amino acid specified by the reference sequence. A variation is a nonsynonymous substitution if any alleles are classified as coding-nonsynon. This class is indicated with a "C" or "N" in graphical summaries. **Note:** As of build 128, function code 4 has been modified into a *two-digit code* that more precisely indicates the nonsynonymous nature of the SNP. The two-digit code has function code 4 as the first digit, which keeps its original meaning, and 1, 2, or 4 as the second digit, which indicates whether the SNP is nonsense, missense, or frameshift. See function codes 41, 42, and 44 in this table.	4
mRNA-UTR	The variation is in the transcript of a gene but not in the coding region of the transcript. This class is indicated by a "T" in graphical summaries. **Note:** As of build 127, function code 5 has been modified into a *two-digit code* that more precisely indicates the location of a SNP. The two digit code has function code 5 as the first digit, which keeps its original meaning, and 3 or 5 as the second digit, which indicates whether the SNP is 3′ or 5′ to the region of interest. See function codes 53 and 55 in this table.	5
Intron	The variation is in the intron of a gene but not in the first two or last two bases of the intron. This class is indicated by an "L" in graphical summaries.	6
Splice-site	The variation is in the first two or last two bases of the intron. This class is indicated by a "T" in graphical summaries. **Note:** As of build 127, function code 7 has been modified into a *two-digit code* that more precisely indicates the location of a SNP. The two-digit code has function codes 7 as the first digit, which keeps its original meaning, and 3 or 5 as the second digit, which indicates whether the SNP is 3′ or 5′ to the region of interest. See function codes 73 and 75 in this table.	7
Contig-reference	The variation allele is identical to the contig nucleotide. Typically, one allele of a variation is the same as the reference genome. The letter used to indicate the variation is a "C" or "N", depending on the state of the alternative allele for the variation.	8
Coding-exception	The variation is in the coding region of a gene, but the precise location cannot be resolved because of an error in the alignment of the exon. The class is indicated by a "C" in graphical summaries.	9
NearGene–3	Function Code 13, where: 1: locus region (see function code 1 in this table) 3: SNP is 3′ to and 0.5 kbp away from gene	13
NearGene–5	Function Code 15, where: 1: locus region (see function code 1 in this table) 3: SNP is 5′ to and 0.5 kbp away from gene	15
Coding-nonsynonymous nonsense	Function Code 41, where: 4: Coding-nonsynonymous (see function code 4 in this table) 1: Nonsense (changes to the Stop codon)	41

Table 83 Function codes for refSNPs in gene features *(continued)*

Functional class	Description	Database code
Coding-nonsynonymous missense	Function Code 42, where: 4: Coding-nonsynonymous (see function code 4 in this table) 2: missense (alters codon to make an altered amino acid in the protein product)	42
Coding-nonsynonymous frameshift	Function Code 44, where: 4: Coding-nonsynonymous (see function code 4 in this table) 4 (as the second digit): frameshift (alters codon to make an altered amino acid in protein product)	44
UTR–3	Function code 53, where: 5: UTR (untranslated region: see function code 5 in this table) 3: SNP located in the 3′ untranslated region	53
UTR–5	Function code 55, where: 5: UTR (untranslated region: see function code 5 in this table) 5 (as the second digit): SNP located in the 5′ untranslated region	55
Splice–3	Function code 73, where: 7: splice site (see function code 7 in this table) 3: 3′ acceptor dinucleotide	73
Splice–5	Function code 75, where: 7: splice site (see function code 7 in this table) 5: 5′ donor dinucleotide	75

Annotation output files

These types of output files are potentially generated by the annotations processing, depending on the LifeScope™ Software modules being run:

- Annotated Variants
- Filter Variants
- Variant Statistics
- Mutated Genes
- diBayes tab-delimited output file
- diBayes annotated tab-delimited output file

Not all output files are generated on each run. Which files are produced depends on the type of analysis modules being run and on the annotation options you select. See "About annotations and LifeScope™ Software modules" on page 322 for more information on annotations available with the different modules, and "Annotation parameters" on page 301 for a description of annotation options.

The output file content is typically labeled within each file, or as described in this chapter in the sections concerning each annotation output file type.

For filtered output files, all annotation-filter options that are turned on must be fulfilled by the variant entry for the variant to appear in the output file.

Annotated Variants and Filtered Variants output files

The Annotated Variants and Filtered Variants output files are the input GFF files (see "Annotation input files" on page 299) with annotations added.

An Annotated Variants output file contains all the entries of the input GFF file. A Filtered Variants output file contains only those entries that fulfill *all* the conditions you specify with the annotation UI options.

The Annotated Variants and Filtered Variants output files are:

- In GFF v3 format
- Named according to these filename patterns:
  `<annotation.output.file.name.stem>_annotated.gff`
  `<annotation.output.file.name.stem>_annotated_filtered.gff`
  where `<annotation.output.file.name.stem>` is a common prefix generated by the annotations module and used with all output files from the same run.

Table 84 describes the additional annotation attributes added to the last (ninth) column of the Annotated Variants and Filtered Variants files. Except for rsID, attributes can have multiple values listed as a comma-separated list.

Table 84 The Annotated Variant and Filtered Variant output file added fields

Attribute/ Metadata	Description
Metadata keyword	
Annotation	A metadata line beginning with the word "Annotation" and followed by these tab-separated fields: • The date and the sources of annotation used • The path to the flat files • The dbSNP version, if available • The human reference build, if available
Filtering	A metadata line beginning with the word "Filtering" and followed by these tab-separated fields: • The values "no filtering" or "show only" • One or more of the values for "coding variants", "variants in genes", "variants in dbSNP", and "variants not in dbSNP"
Attributes	
geneID	Add a geneID attribute to applicable entries in the output GFF file. The following apply to the geneID attribute: • The attribute is valid for all variants. • The attribute's value is the gene descriptor taken from the input GTF file. If there are multiple overlapping genes, the attribute has multiple values (in a comma-separated list).

Table 84 The Annotated Variant and Filtered Variant output file added fields *(continued)*

Attribute/ Metadata	Description
exonID	Adds exonID attributes to applicable entries in the output GFF file. The following apply to the exonID attribute: • The attribute is valid for all variants. • The attribute's value is of the form <*GeneID*>-<*Exon#*>, where *Exon#* is the number of the exon counted within the gene (not within each transcript) in the transcription direction. • If there are multiple overlapping exons, the attribute has multiple values (in a comma-separated list). • If two exons start from the same position, the shorter one is listed first.
rsID	Adds rsID attributes to applicable entries in the output GFF file. The following apply to the rsID attribute: • The attribute is valid for SNPs and Indels. • The attribute's value is the rsID value of the dbSNP entry. The attribute cannot have multiple values.
functionCode	Based on functional codes from dbSNP, annotate SNP entries in the output GFF file. The following apply to the functionCode attribute: • The attribute is valid for only for SNPs. • The attribute's value is a pair in which the first value is the locus ID and the second the function code from dbSNP. Function codes are listed in Table 83. The attribute can have multiple values, one for each SNP type that applies to the variant.

Variant Statistics output file for SNPs

A Variant Statistics file is a text file containing the statistics about the variants in the input GFF file. The file is named according to this filename pattern:

```
<annotation.output.file.name.stem>.stats
```

where `<annotation.output.file.name.stem>` is a common prefix generated by the annotations module and used with all output files from the same run.

The following is an example of the content of a Variant Statistics file for SNPs:

```
************************************************************
LIFESCOPE REPORT
************************************************************

Input File:/data/swdev/minita_test/
solid0064_20101210_MP_2x60_T3_1/2.0/color_lmp/outputs/dibayes/
color_lmp_SNP.gff3
Date:Apr 14, 2011 2:17:54 PM
Annotation file, dbSNP:/data/analysis/LifeScope_resources/
lifetech/hg18/dbSNP/dbSNP_b130.tab
```

```
Annotation file, Genes and Exons, GTF:/data/analysis/
LifeScope_resources//lifetech/hg18/GTF/refGene.20090513.gtf

************************************************************
Statistics Overview
************************************************************

-- Basic Statistics ----------------------------------------

Number of variants                                  3072881
Number of heterozygous variants                     1639649
Number of homozygous variants                       1433232

-- Variant-Specific Statistics (SNP) -------------------------

Number of transition SNPs                           2070418
Number of transition heterozygous SNPs              1103589
Number of transition homozygous SNPs                 966829
Number of transversion SNPs                         1002463
Number of transversion heterozygous SNPs             536060
Number of transversion homozygous SNPs               466403
Transition:Transversion ratio                 2.065 : 1.000


-- dbSNP Annotation Statistics -------------------------------

Number of              SNPs in dbSNP                2913081
Number of heterozygous SNPs in dbSNP                1506740
Number of homozygous   SNPs in dbSNP                1406341
dbSNP concordance                                     94.80%
dbSNP heterozygous concordance                        91.89%
dbSNP homozygous concordance                          98.12%

-- Per-Chromosome Statistics ---------------------------------

nVar: Number of variants
nHetVar: Number of heterozygous variants
nHomVar: Number of homozygous variants

contig          nVar        nHetVar        nHomVar
chr1          233502         125860         107642
chr2          247682         132717         114965
chr3          210137         115661          94476
chr4          225996         118326         107670
chr5          185137         102825          82312
chr6          191793         104138          87655
chr7          170500          94092          76408
chr8          160126          86553          73573
chr9          128205          70291          57914
chr10         157345          85408          71937
chr11         154271          80792          73479
chr12         139569          72915          66654
chr13         118842          60095          58747
chr14          93526          48742          44784
```

```
chr15          90141          46673          43468
chr16          95819          53400          42419
chr17          79447          44311          35136
chr18          85824          43117          42707
chr19          59142          31904          27238
chr20          65136          38340          26796
chr21          45656          26098          19558
chr22          36633          20355          16278
chrX           76085          18909          57176
chrY           22328          18127           4201
chrM              39              0             39
```

Variant Statistics output file for small indels

A Variant Statistics file is a text file containing the statistics about the variants in the input GFF file. The file is named according to this filename pattern:

*<annotation.output.file.name.stem>*_stats

where *<annotation.output.file.name.stem>* is a common prefix generated by the annotations module and used with all output files from the same run.

The following is an example of the content of a Variant Statistics file for small indels:

```
************************************************************
LIFESCOPE REPORT
************************************************************

Input File:/data/swdev/minita_test/
solid0064_20101210_MP_2x60_T3_1/2.0/color_lmp/outputs/
small.indel/color_lmp.gff3
Date:Apr 14, 2011 10:01:01 AM
Annotation file, dbSNP:/data/analysis/LifeScope_resources/
lifetech/hg18/dbSNP/dbSNP_b130.tab
Annotation file, Genes and Exons, GTF:/data/analysis/
LifeScope_resources//lifetech/hg18/GTF/refGene.20090513.gtf

************************************************************
Statistics Overview
************************************************************

-- Basic Statistics ----------------------------------------

Number of variants                                   296183
Number of heterozygous variants                      194752
Number of homozygous variants                        100934

-- Variant-Specific Statistics (InDel) ----------------------
Distribution of InDel length

Length(bases)                Number
(-Inf:-500]                       0
(-500:-490]                      14
(-490:-480]                       8
(-480:-470]                      13
(-470:-460]                      21
```

(-460:-450]	14
(-450:-440]	18
(-440:-430]	17
(-430:-420]	18
(-420:-410]	18
(-410:-400]	15
(-400:-390]	24
(-390:-380]	21
(-380:-370]	24
(-370:-360]	30
(-360:-350]	39
(-350:-340]	62
(-340:-330]	115
(-330:-320]	202
(-320:-310]	181
(-310:-300]	80
(-300:-290]	43
(-290:-280]	52
(-280:-270]	42
(-270:-260]	56
(-260:-250]	49
(-250:-240]	35
(-240:-230]	39
(-230:-220]	35
(-220:-210]	33
(-210:-200]	37
(-200:-190]	34
(-190:-180]	30
(-180:-170]	40
(-170:-160]	49
(-160:-150]	42
(-150:-140]	55
(-140:-130]	78
(-130:-120]	55
(-120:-110]	68
(-110:-100]	51
(-100:-90]	60
(-90:-80]	61
(-80:-70]	94
(-70:-60]	107
(-60:-50]	115
(-50:-40]	229
(-40:-30]	332
(-30:-20]	1064
(-20:-19]	220
(-19:-18]	392
(-18:-17]	268
(-17:-16]	568
(-16:-15]	436
(-15:-14]	624
(-14:-13]	541
(-13:-12]	1126
(-12:-11]	1009
(-11:-10]	1636

```
(-10:-9]                      1276
(-9:-8]                       2367
(-8:-7]                       1566
(-7:-6]                       3498
(-6:-5]                       5458
(-5:-4]                      15589
(-4:-3]                      11836
(-3:-2]                      22210
(-2:-1]                      77896
(-1:0]                           0
(0:1]                        65663
(1:2]                        19317
(2:3]                        14044
(3:4]                        25082
(4:5]                         3749
(5:6]                         2199
(6:7]                         1011
(7:8]                         1596
(8:9]                          892
(9:10]                        1169
(10:11]                        705
(11:12]                       1289
(12:13]                        803
(13:14]                       1264
(14:15]                       3547
(15:16]                        192
(16:17]                        164
(17:18]                        123
(18:19]                        141
(19:20]                        268
(20:30]                        530
(30:+Inf)                        0


-- dbSNP Annotation Statistics ------------------------------

Number of              InDels in dbSNP               246619
Number of heterozygous InDels in dbSNP               150203
Number of homozygous   InDels in dbSNP                96105
dbSNP concordance                                     83.27%
dbSNP heterozygous concordance                        77.13%
dbSNP homozygous concordance                          95.22%

-- Per-Chromosome Statistics --------------------------------

nVar: Number of variants
nHetVar: Number of heterozygous variants
nHomVar: Number of homozygous variants

contig          nVar        nHetVar        nHomVar
chr1           22753          15323           7385
chr2           24215          16211           7975
chr3           20327          13613           6686
chr4           22322          14798           7485
```

```
chr5          18224          12212          5989
chr6          19046          12807          6213
chr7          16887          11032          5824
chr8          14530           9625          4889
chr9          11539           7926          3601
chr10         14864           9836          5000
chr11         14951           9502          5428
chr12         14000           9180          4793
chr13         11885           7685          4184
chr14          9439           6105          3323
chr15          8945           5912          3022
chr16          7705           5304          2382
chr17          7849           5323          2508
chr18          8605           5530          3066
chr19          5866           3825          2026
chr20          6107           4120          1977
chr21          5067           2977          2072
chr22          3587           2301          1280
chrX           6351           2650          3675
chrY           1118            954           151
chrM              1              1             0
```

Variant Statistics output file for large indels

A Variant Statistics file is a text file containing the statistics about the variants in the input GFF file.

The file is named according to this filename pattern:

> *<annotation.output.file.name.stem>*_stats
>
> where *<annotation.output.file.name.stem>* is a common prefix generated by the annotations module and used with all output files from the same run.

The following is an example of the content of a Variant Statistics file for large indels:

```
************************************************************
LIFESCOPE REPORT
************************************************************

Input File:/data/swdev/minita_test/
solid0064_20101210_MP_2x60_T3_1/2.0/color_lmp/outputs/
large.indel/large-indels.gff3
Date:Apr 14, 2011 10:00:05 AM
Annotation file, dbSNP:/data/analysis/LifeScope_resources/
lifetech/hg18/dbSNP/dbSNP_b130.tab
Annotation file, Genes and Exons, GTF:/data/analysis/
LifeScope_resources//lifetech/hg18/GTF/refGene.20090513.gtf

************************************************************
Statistics Overview
************************************************************

-- Basic Statistics ----------------------------------------

Number of variants                                        21
```

```
Number of heterozygous variants                              2
Number of homozygous variants                               19

-- Variant-Specific Statistics (InDel) -----------------------
Distribution of InDel length

Length(bases)                Number
(-Inf:-500000]                    0
(-500000:-100000]                 0
(-100000:-50000]                  0
(-50000:-20000]                   0
(-20000:-10000]                   0
(-10000:-5000]                    0
(-5000:-1000]                     0
(-1000:-500]                      0
(-500:-200]                       7
(-200:-100]                       5
(-100:0]                          2
(0:100]                           2
(100:200]                         4
(200:500]                         1
(500:1000]                        0
(1000:5000]                       0
(5000:10000]                      0
(10000:20000]                     0
(20000:50000]                     0
(50000:100000]                    0
(100000:500000]                   0
(500000:+Inf)                     0


-- dbSNP Annotation Statistics -------------------------------

Number of             InDels in dbSNP                        0
Number of heterozygous InDels in dbSNP                       0
Number of homozygous   InDels in dbSNP                       0
dbSNP concordance                                       0.00%
dbSNP heterozygous concordance                          0.00%
dbSNP homozygous concordance                            0.00%

-- Per-Chromosome Statistics ---------------------------------

nVar: Number of variants
nHetVar: Number of heterozygous variants
nHomVar: Number of homozygous variants

contig          nVar        nHetVar        nHomVar
chr2               1              0              1
chr9               1              0              1
chr10              7              2              5
chr18              4              0              4
chr19              2              0              2
chrY               4              0              4
chrM               2              0              2
```

Variant Statistics output file for CNVs

A Variant Statistics file is a text file containing the statistics about the variants in the input GFF file.

The file is named according to this filename pattern:

*<annotation.output.file.name.stem>*_stats

where *<annotation.output.file.name.stem>* is a common prefix generated by the annotations module and used with all output files from the same run.

The following is an example of the content of a Variant Statistics file for CNVs:

```
************************************************************
LIFESCOPE REPORT
************************************************************

Input File:/data/swdev/minita_test/
solid0064_20101210_MP_2x60_T3_1/2.0/color_lmp/outputs/cnv/
OutputCNVs.gff3
Date:Apr 14, 2011 9:58:12 AM
Annotation file, dbSNP:/data/analysis/LifeScope_resources/
lifetech/hg18/dbSNP/dbSNP_b130.tab
Annotation file, Genes and Exons, GTF:/data/analysis/
LifeScope_resources//lifetech/hg18/GTF/refGene.20090513.gtf

************************************************************
Statistics Overview
************************************************************

-- Basic Statistics ----------------------------------------

Number of variants                                      120

-- Variant-Specific Statistics (CNV) -------------------------

Distribution of CNV copy number
Copy_Number          Number
          0               5
          1              20
          2              23
          3              20
          4              22
          5               8
          6               9
          7               1
          8               2
          9              10
         >9               0

Distribution of CNV length
Length(bases)                 Number
(0:1000]                           0
(1000:2000]                        0
```

```
(2000:5000]                      0
(5000:10000]                     0
(10000:20000]                   41
(20000:50000]                   54
(50000:100000]                  11
(100000:500000]                 10
(500000:+Inf)                    4


-- Per-Chromosome Statistics ---------------------------------

nVar: Number of variants

contig          nVar
chr1               7
chr2               5
chr3              11
chr4               4
chr5               7
chr6              11
chr7               4
chr8               5
chr9               3
chr10              3
chr11              8
chr12              4
chr13              3
chr14              2
chr15              4
chr16              2
chr17              4
chr19              4
chr20              1
chr22              2
chrX              17
chrY               9
```

Mutated Genes output file

The Mutated Genes output file contains the list of genes whose coding regions were modified by at least one of the variants in the input file. The Mutated Genes file is a text file in tab-separated format. The file contains an entry per row, and one gene per entry. The first 12 columns are the same as the columns in the BED file (`genes.bed`). Additional columns are added by the annotations processing.

The file is named according to this filename pattern:

`<annotation.output.file.name.stem>_genes.tab`

where `<annotation.output.file.name.stem>` is a common prefix generated by the annotations module and used with all output files from the same run.

Table 85 lists the information included in the Mutated Genes output file.

Table 85 Description of the Mutated Genes output file

Field number	Field name	Description
1	Chromosome/Contig	The contig of the gene.
2	Start	The left-most coordinate of the gene.
3	End	The right-most coordinate of the gene.
4	GeneID	The gene identifier as provided by the GTF file.
5	Score	A score between 0 and 1000. The score is a normalized value for the number of coding variants such that all numbers scale between 0 and 1000. The score is equal to the number of coding variants for this entry divided by the maximum number of coding variants for any gene.
6	Strand	The strand of the gene.
7	Start	The left-most coordinate of the gene.
8	End	The right-most coordinate of the gene.
9	Variant color	The color depends on the type of the variant that overlaps this gene: Red: SNP Green: Indel (small indels, large indels) Cyan: CNV
10	Number of exons	The number of exons of the gene.
11	Exon start position list	A comma-separated list of start-coordinates for exons in the gene. All coordinates are relative to the start position of the gene.
12	Exon length list	A comma-separated list of the lengths of the exons in the gene.
13	Total exon length	The sum the length of all exons.
14	Number of coding variants	The number of variants overlapping at least one exon.
15	Number of variants	The number of variants overlapping the gene.
16	Variant type	The type of the variant. (One type is listed per line.)
17	Variant start list	A comma-separated list for start coordinates of variants overlapping the gene. All coordinates are relative to the start position of the gene.
18	Variant length list	A comma-separated list of lengths of the variants overlapping the gene.

Potential uses of a Mutated Genes file

You can use a spreadsheet application or scripts to remove unwanted columns. First save a backup of your output file. The following are examples of uses for a Mutated Genes file:

- The first 12 columns provide information for visualization of modified genes and exons.
- Column 4, the GeneID, provides a list of modified genes. This column provides a gene list to focus on for enrichment analysis.
- A combination of the first 10 and the last 2 fields provides information useful for visualizing variants that overlap the gene.

- Fields 14 and 15 provide the number of variants that overlap exons and the number of variants that overlap the gene. With these columns you can filter by the number of variants overlapping the exons in the gene. With the GeneID in column 4, this information identifies genes to focus on for further analysis.

(*Optional*) View a Mutated Genes output file

Open the BED file in the genomic browser.

Example of a Mutated Genes output file

The following is a truncated example (3 lines) of a Mutated Genes output file:

```
Chromosome/Contig Start End GeneID Score Strand Start End
Variant color Number of Exons Exon start position list Exon
length list Total exon length Number of coding variants Number
of variants Variant type Variant start list Variant length list
chr1  752927    779603    LOC643837  0    +    752927    779603
255,0,0    7    0,1319,19970,24243,24987,25707,25893
92,102,153,184,96,132,784    1543  2    15    SNP
3041,5189,6765,12362,13482,14058,15505,17721,20240,21986,22925,
24198,24335,25649,26192    1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
chr1  850984    869824    SAMD11     0    +    850984    869824
255,0,0    14   0,181,4414,5298,10031,13299,13534,15403,16395,1
6669,16818,17512,17957,18167    60,92,182,51,125,90,186,163,116
,79,500,125,111,674    2554  1    3    SNP  1003,6463,18566
1,1,1
chr1  945366    981355    AGRN       0    +    945366    981355
255,0,0    360,2078,15154,20542,21050,21355,21833,23116,23415,2
3700,23986,24211,25038,25236,25610,25841,26037,26274,26697,2720
4,27450,27653,27889,28744,29113,29443,29780,30110,30304,30603,3
1130,31330,31605,33630,34325,34701    251,262,48,216,225,225,20
7,219,195,201,149,106,117,165,144,125,106,339,138,128,115,120,3
54,193,216,230,135,97,165,112,117,193,88,225,104,1289    7319
    1    8    SNP    16631,17955,18722,18991,31697,32807,33000,
34914 1,1,1,1,1,1,1,1
```

diBayes tab-delimited output file

The first eight columns of this output file are the same as the input GFF file. The ninth column in the input file is replaced by individual columns for each attribute, as shown in Table 86. This output file is generated only on SNPs module runs.

This is a tab-separated text file. The file is named according to this filename pattern:

`<annotation.output.file.name.stem>.tab`

where `<annotation.output.file.name.stem>` is a common prefix generated by the annotations module and used with all output files from the same run.

Table 86 Attributes in the ninth column

Attribute	Description
Seqid	The string ID of the sequence to which the start and end coordinates refer.
Source	The source of the data.
Type	Sequence ontology derived type for this variation. For diBayes, this is always SNP.
Start	Start position of the SNP.

Table 86 Attributes in the ninth column *(continued)*

Attribute	Description
End	End position of the SNP.
Score	Calculated p-value of the SNP.
Strand	Not used.
Phase	Not used.
Genotype	Genotype in the form of IUB codes for bases observed in all the reads. The base of the reference sequence at the current position.
Coverage	The number of the reads that cover the current position.
refAlleleCounts	The number of reads of the reference allele at the current position.
refAlleleStarts	The number of different start positions of reads having the reference allele at the current position.
refAlleleMeanQV	The mean of quality values of all reference allele reads at the current position.
novelAlleleCount	The number of reads of the most abundant non-reference allele at the current position.
novelAlleleStarts	The number of different start positions of reads having the most abundant non-reference allele at the current position.
novelAlleleMeanQV	The mean of quality values of all novel allele reads.
mostAbundantAlleleDiColor2	The most abundant allele in the reads (not necessarily the reference allele) in dicolor encoding (for example, 00, 01, ... 32, 33), of 16 possible dicolors.
secondAbundantAlleleDiColor3	The second most abundant allele in the reads.
Het	Heterozygosity flag. Allowed values: 0,1 • **0**: Homozygous SNP • **1**: Heterozygous SNP

The following is a truncated example of diBayes tab-delimited output file content:

```
Seqid Source  Type  Start End  Score Strand  Phase genotype
reference  coverage refAlleleCounts refAlleleStarts
refAlleleMeanQV novelAlleleCounts novelAlleleStarts
novelAlleleMeanQV mostAbundantAlleleDiColor2
secondAbundantAlleleDiColor3 het
chr1 SOLiD_diBayes SNP 10007  10007  0.806886   .   .   A   G
3   1   1   16   2   2   25   21   21   0
chr1 SOLiD_diBayes SNP 224472 224472 0.879526   .   .   C   T
3   1   1   26   2   2   24   10   10   0
chr1 SOLiD_diBayes SNP 553488 553488 0.0625     .   .   C   T
3   0   0   0    2   2   25   20   20   0
chr1 SOLiD_diBayes SNP 556955 556955 0.095335   .   .   Y   T
10  7   6   14   2   2   26   13   31   1
chr1 SOLiD_diBayes SNP 558326 558326 0.093729   .   .   R   A
8   6   6   26   2   2   22   03   21   1
chr1 SOLiD_diBayes SNP 558554 558554 0.0761     .   .   Y   T
14  11   9  20   3   3   22   03   21   1
```

diBayes annotated tab-delimited output file

This output file is the same as the diBayes tab-delimited output file (see "diBayes tab-delimited output file" on page 319), with four additional columns added. As with the diBayes tab-delimited output file, the first eight columns of this output file are the same as the input file. The ninth column in the input file is replaced by individual columns for each attribute, as shown in Table 87. This output file is generated only on SNPs module runs.

This is a tab-separated text file. The file is named according to this filename pattern:

*<annotation.output.file.name.stem>*_annotated.tab

where *<annotation.output.file.name.stem>* is a common prefix generated by the annotations module and used with all output files from the same run.

Table 87 Additional columns in the diBayes annotated tab-delimited output file

Field	Description
geneID	The gene id as provided by the GTF file.
exonID	The exon id formed by concatenating the GeneID and the exon index number in the list of exons of the gene sorted in transcription order. Transcription order ties are broken listing by the shortest exon first (<GeneID>-<exon #>).
rsID	The dbSNP id of the SNP.
functionCode	The dbSNP functional code.

The following is a truncated example of diBayes annotated tab-delimited output file content:

```
Seqid Source  Type  Start End  Score Strand  Phase genotype
reference  coverage refAlleleCounts refAlleleStarts
refAlleleMeanQV novelAlleleCounts novelAlleleStarts
novelAlleleMeanQV mostAbundantAlleleDiColor2
secondAbundantAlleleDiColor3 het geneID exonID rsID functionCode
chr1  SOLiD_diBayes SNP 10007  10007  0.806886   .   .   A   G
    3   1   1   16   2   2   25   21   21   0   WASH5P
chr1  SOLiD_diBayes SNP 224472 224472 0.879526   .   .   C   T
    3   1   1   26   2   2   24   10   10   0
chr1  SOLiD_diBayes SNP 553488 553488 0.0625     .   .   C   T
    3   0   0   0    2   2   25   20   20   0
chr1  SOLiD_diBayes SNP 556955 556955 0.095335   .   .   Y   T
    10  7   6   14   2   2   26   13   31   1   9326622
chr1  SOLiD_diBayes SNP 558326 558326 0.093729   .   .   R   A
    8   6   6   26   2   2   22   03   21   1   2153587
chr1  SOLiD_diBayes SNP 558554 558554 0.0761     .   .   Y   T
    14  11  9   20   3   3   22   03   21   1   8179256
chr1  SOLiD_diBayes SNP 658055 658055 0.0625     .   .   T   C
    3   0   0   0    2   2   22   20   20   0
```

diBayes filtered annotated tab-delimited output file

This file is the same as the diBayes annotated tab-delimited output file, except that it contains only those entries that fulfill all the conditions set by the your annotation filtering settings (described in "Annotation parameters" on page 301).

This is a tab-separated text file. The file is named according to this filename pattern:

*<annotation.output.file.name.stem>*_annotated_filtered.tab

where *<annotation.output.file.name.stem>* is a common prefix generated by the annotations module and used with all output files from the same run.

This output file is generated only on SNPs module runs.

The following is a truncated example of diBayes filtered annotated tab-delimited output file content:

```
Seqid Source  Type  Start End  Score Strand  Phase genotype
reference  coverage refAlleleCounts refAlleleStarts
refAlleleMeanQV novelAlleleCounts novelAlleleStarts
novelAlleleMeanQV mostAbundantAlleleDiColor2
secondAbundantAlleleDiColor3 het geneID exonID rsID functionCode
chr1 SOLiD_diBayes SNP 556955 556955 0.095335   .   .   Y   T
10   7   6   14   2   2   26   13   31   1         9326622
chr1 SOLiD_diBayes SNP 558326 558326 0.093729   .   .   R   A
8    6   6   26   2   2   22   03   21   1         2153587
chr1 SOLiD_diBayes SNP 558554 558554 0.0761     .   .   Y   T
14   11  9   20   3   3   22   03   21   1         8179256
chr1 SOLiD_diBayes SNP 708249 708249 0.003906   .   .   G   A
5    0   0   0    3   3   23   30   30   0          10900602
chr1 SOLiD_diBayes SNP 708418 708418 0.0625     .   .   C   T
2    0   0   0    2   2   15   13   13   0         10751453
chr1 SOLiD_diBayes SNP 710103 710103 0.003906   .   .   C   T
3    0   0   0    3   3   17   01   01   0          3121393
chr1 SOLiD_diBayes SNP 713754 713754 0.003906   .   .   C   G
4    0   0   0    3   3   21   33   33   0          2977670
```

About annotations and LifeScope™ Software modules

This section describes which annotation functionality is available with the various LifeScope™ Software modules.

Table 88 shows which LifeScope™ Software modules support the annotation attributes.

Table 88 Support for annotation labels

Label	SNPs	CNV	Small indel	Large indel
geneID, exonID	Yes	Yes	Yes	Yes
rsID - SNPs	Yes	—	—	—
rsID – Indels	—	—	Yes	—
functionCode	Yes	—	Yes	—

Table 89 lists which LifeScope™ Software modules support annotation filtering.

Table 89 Support for the annotation filtering options

Label	SNPs	CNV	Small indel	Large indel
Only in exons	Yes	Yes	Yes	Yes
Only in genes	Yes	Yes	Yes	Yes

Label	SNPs	CNV	Small indel	Large indel
Only *not* in dbSNP	Yes	—	Yes	—
Only in dbSNP	Yes	—	Yes	—

Table 90 lists which LifeScope™ Software modules support for the various annotation statistics. Heterozygous and homozygous statistics are included for applicable variants.

Table 90 Support for annotation statistics

Label	SNPs	CNV	Small indel	Large indel
Number of variants (total and per chromosome)	Yes	Yes	Yes	Yes
Transitions and transversions	Yes	—	—	—
Variant length distribution	—	Yes	Yes	Yes
Copy number distribution	—	Yes	—	—
dbSNP concordance	Yes	—	Yes	—
Overlapping exons (number and percent)	Yes	Yes	Yes	Yes
Overlapping genes (number and percent)	Yes	Yes	Yes	Yes

FAQ - Annotations

1

How are errors with input files handled?

If an entry in the GTF file is malformed then the entry would be skipped. If there is a problem with the file then the entire file would be skipped. If no file is provided or is skipped then the GTF annotation is turned off. Similarly, if one of the dbSNP files is missing then the dbSNP annotation is turned off. The *.stats files may still show zeros in the dbSNP fields if the dbSNP annotation is turned on.

2

Can I add annotations to my GFF file from an earlier version of the software?

Yes. Run the annotations as a standalone module. Set the parameter to point to your existing GFF file.

3

Why is the function code missing locus information in hg19 while the information is in hg18? Can I do anything to get the locus information?

While doing annotation with hg19 we use the VCF file, which does not have locus information yet. You can use the dbSNP b132 BCP files to get the locus information, but please note that the indels are not completely annotated in that file. Also, using the BCP files requires more memory than the VCF file.

4

What is the definition of dbSNP concordance?

The dbSNP concordance reported is positional concordance. The dbSNP concordance measures the percentage of SNP calls that are at the same position as a variant in dbSNP. Exact comparison of genotypes is not performed, since typically genotypes for the sample of interest are not available in dbSNP. There is no comparison of observed alleles with the alleles reported at this position in dbSNP.

By default, only variants of the same type are counted:

- SNPs are annotated against SNPs in dbSNP.
- Indels are annotated against InDels in dbSNP.

PART V

Targeted Resequencing and Low Frequency Variants

16 Run Targeted Resequencing Analyses

This chapter covers:

- Overview 327
- Example of running targeted resequencing analyses 328
- Targeted resequencing read selection algorithm 329
- Run targeted resequencing modules 330

Overview

Targeted resequencing is a set of analyses designed for target-enriched sequencing data. Given a set of coordinates representing a region of interest in a genome, a library may be enriched for reads within those regions prior to sequencing.

The workflow for analysis of target-enriched data is nearly identical to a standard resequencing workflow (see Chapter 9, "Run a Resequencing Mapping Analysis" on page 137). In fact, adequate results can be obtained using the standard workflow. The targeted resequencing application consists of many of the same modules as standard resequencing analysis. However, two additional modules are enabled by default for targeted resequencing analysis, and several modules have specific parameters set that are optimized for analysis of targeted resequencing data.

The SAET module is enabled by default for enriched runs, as the smaller target size makes spectral error correction viable, even if it may not be practical for unenriched sequencing runs on a large reference sequence.

Targeted resequencing mapping produces a filtered BAM file for variant calling. The variant calling modules diBayes and small indel are run on the filtered data sets instead of the entire read population, and limit SNP and indel calls to the target region.

The input list of regions of interest (ROI) is provided by the user. Supported ROI input files include third-party files of baits, probes, or tiles. The input ROI files can be in BED format or in a text file as a list of "chr:start-end" values. The targeted resequencing mapping module sorts the input ROI file by chromosome and start position, and makes the resulting file available as output.

Figure 30 describes the targeted resequencing analyses.

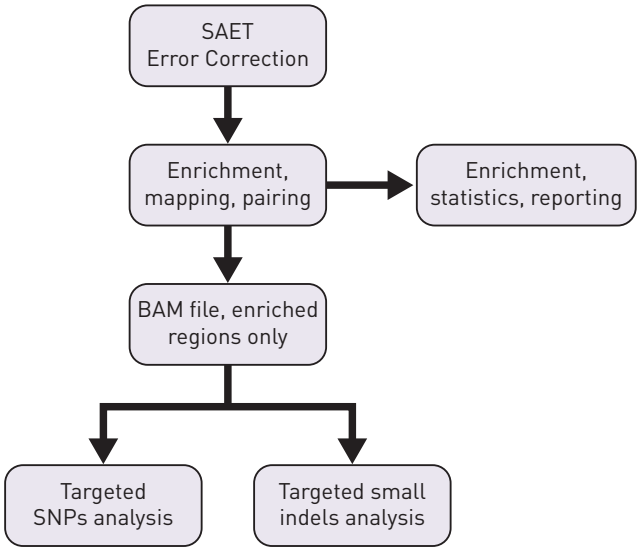


Figure 30 Components of targeted resequencing analyses

The targeted resequencing analyses support data from these library types:

- Fragment
- Paired-End
- Mate-Pair
- Mate-pair libraries are accepted, but pairing information is not considered for mate-pair libraries.
- Libraries generated with up to 96 barcodes

  Individual sequencing and enrichment reports are provided for each barcoded library.

Targeted resequencing output files are compatible with third-party genome browsers, including the IGV browser.

The targeted resequencing mapping module provides analysis of sequencing runs that have been run through targeted enrichment.

Targeted resequencing support optional enrichment statistics, which provide a means to assess enrichment platform performance. For information about the enrichment module, see Chapter 18, "Run Enrichment Analyses" on page 339.

Example of running targeted resequencing analyses

The following standard workflows provide examples of running targeted resequencing:

- targeted.resequencing.frag
- targeted.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
lscope.sh shell -u username -w password
# cd to the projects repository
```

```
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow targeted.resequencing.frag
# define the input data
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
# specify the reference for the analysis
set reference hg19
# optionally change parameter defaults here
set analysis.regions.file /data/results/readsdata/dh10b.bed
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
```

The command to set the `analysis.regions.file` does not include the optional INI file argument. As written, the command sets the `analysis.regions.file` parameter in both the secondary and tertiary global.ini files. In this way the parameter is available to all analysis modules in the workflow. If the parameter is instead set in only one INI file, for example the `enrichment.ini` file, then the regions of interest file is available to the enrichment module, but not available to the SAET module and other modules in the workflow.

See for Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information on running standard workflows. Chapter 6, "Run a Command Shell Analysis" on page 83 for information on shell commands and syntax.

Targeted resequencing read selection algorithm

Reads are classified as "on-target" by a set of user-definable criteria. First, all reads are filtered for a minimum mapping quality. Reads which align to multiple places in the genome are assigned a lower mapping quality. Reads with very low mapping qualities may be false positive hits arising from repetitive sequence in the reference genome.

The default target definition criteria used in targeted resequencing mapping are:

- Reads must have a minimum mapping quality of 8. Recommended values are between 0 (in which case there may be a population of repetitive reads which inflate enrichment statistics and may contribute to false positives in variant calling), and 20 (which ensures only high quality mapping reads are used, but results in a loss of sensitivity in regions with low mappability).
- Alignments may be marked as duplicate by the alignment algorithm. These reads are typically excluded from variant calling, but are counted for purposes of assessing enrichment.

- Only primary alignments are considered. Alignments marked as secondary (alternative placements) are excluded. Note that the mapping quality threshold may excluded reads with multiple secondary hits.
- Target regions are considered as provided. Reads aligning adjacent to targets may be included in the analysis by specifying the number of bases flanking the target regions that each target should be extended by.
- Mapping to target region is defined as having at least one base overlapping the target region.
- For paired-end runs, by default the reverse read must align 100%, as these are often shorter and less accurate. The default 100% alignment requires that all bases align to coordinates that are fully contained by the target region. For this test, trimmed bases are not considered. The overlap percentage required is controlled by the `enrichment.minimum.target.overlap.reverse` parameter.

Both alignment thresholds are user-adjustable, down to $1x10^{-4}$ (meaning at least one base, for reads up to 1000bp) or up to 1.0, meaning the read and target must completely overlap.

While only one list of target regions can be provided by the user, multiple target region sets may be analyzed by re-running the enrichment analysis, and specifying a different output folder or prefix for each target list. Similarly, to assess near-target alignments, reanalyze while specifying a larger number of bases by which to extend the target.

The targeted resequencing module by default runs the SAET utility prior to mapping.

Run targeted resequencing modules

See Chapter 18, “Run Enrichment Analyses” on page 339 for information about the enrichment module.

The instructions for other targeted resequencing modules are the same as for the resequencing tools, except that the targeted modules require an additional input file, the regions of interest (ROI) file. You must provide the ROI file, in any of the following supported formats:

- a bait, probe, tile, or other file in BED format
- a text file as a list of “chr:start-end” values

To run resequencing pairing, find SNPs, and find small indels analyses, complete the instructions in the following chapters and also specify your the regions of interest file:

- Chapter 9, “FAQ – Pairing” on page 182
- Chapter 10, “Run a SNPs Analysis” on page 187
- Chapter 13, “Run a Small Indels Analysis” on page 239

The differences when these analyses are run for targeted resequencing, compared to standard resequencing, are:

- The find SNPs and find small indels analyses are both run with an input file containing only on-target reads. Additionally, find SNPs takes the processed target list as input, and restricts SNP calls to events occurring within the target regions.
- SNPs analysis is also passed the `dibayes.het.skip.high.coverage=0` parameter, to avoid excluding regions of very high coverage, which are common in enriched regions.

- The SAET module is enabled by default.

Note: If you turn off SAET, comment out the `analysis.input.readset.file` line, if it appears in your mapping INI file:

```
analysis.input.readset.file = ${analysis.output.dir}/saet/*.rrs
```

17 Run Low Frequency Variant Analyses

This chapter covers:

- Overview 333
- Examples of running low frequency variant analyses 334
- Low frequency variant read selection algorithm 335
- SNPs optimizations 336
- Run low frequency variant modules 338

Overview

The low frequency variant detection workflow uses the diBayes (SNPs) algorithm (and highly optimized parameters) to reveal SNPs present in a targeted resequencing experiment. The detection limit can range from 1%-5% based on experimental conditions, acceptable sensitivity and specificity, and sequencing coverage. This workflow is ideal for detecting low frequency variants in cancer-related research projects.

The low frequency variant workflows use unique defaults for SNPs filters and also uses these new SNPs filters:

- The color QV of the less common allele
- The color QV difference between the most common allele and the less common allele
- The unique start positions of the less common allele
- The requirement that the less common allele be present on both strands

The workflow for analysis of target-enriched data is nearly identical to a standard targeted resequencing workflow. The low frequency variant application consists of many of the same modules as standard resequencing analysis. However, several modules have specific of target-enriched data.

Low frequency variant workflow uses the enrichment module to produce a filtered BAM file for variant calling. The variant calling module diBayes is run on the filtered data sets instead of the entire read population, and limits SNP calls to the target region. SNPs module parameters are optimized for low frequency variant detection.

The input list of regions of interest (ROI) is provided by the user. Supported ROI input files include third-party files of baits, probes, or tiles. The input ROI files can be in BED format or in a text file as a list of “chr:start-end” values. The mapping module sorts the input ROI file by chromosome and start position, and makes the resulting file available as output.

The low frequency variant analyses support data from these library types:

- Fragment
- Paired-End
- Libraries generated with up to 96 barcodes

  Individual sequencing and enrichment reports are provided for each barcoded library.

Low frequency variant output files are compatible with third-party genome browsers, including the IGV browser.

Low frequency variant workflows optionally produce enrichment statistics, which provide a means to assess enrichment platform performance. For information about the enrichment module, see Chapter 18, "Run Enrichment Analyses" on page 339.

Examples of running low frequency variant analyses

The following standard workflows provide examples of running low frequency variant analyses:

- low.frequency.variant.frag
- low.frequency.variant.pe

The following are example shell commands using reads and references from the LifeScope™ Software repositories.

```
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk liversample1
cd liversample1
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow low.frequency.variant.frag
# define the input data
add xsq Sample_TarReseq_Unbarcoded_50_Frag.xsq
# specify the reference for the analysis
set reference hg19
# optionally change parameter defaults here
set analysis.regions.file /data/results/readsdata/dh10b.bed
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
```

The command to set the `analysis.regions.file` does not include the optional INI file argument. As written, the command sets the `analysis.regions.file` parameter in both the secondary and tertiary global.ini files. In this way the parameter is available to all analysis modules in the workflow. If the parameter is instead set in only one INI file, for example the `enrichment.ini` file, then the regions of interest file is available to the enrichment module, but not available to the SAET module and other modules in the workflow.

See for Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information on running standard workflows. Chapter 6, "Run a Command Shell Analysis" on page 83 for information on shell commands and syntax.

Low frequency variant read selection algorithm

Reads are classified as "on-target" by a set of user-definable criteria. First, all reads are filtered for a minimum mapping quality. Reads which align to multiple places in the genome are assigned a lower mapping quality. Reads with very low mapping qualities may be false positive hits arising from repetitive sequence in the reference genome.

The default target definition criteria used in low frequency variant mapping are:

- Reads must have a minimum mapping quality of 8. Recommended values are between 0 (in which case there may be a population of repetitive reads which inflate enrichment statistics and may contribute to false positives in variant calling), and 20 (which ensures only high quality mapping reads are used, but results in a loss of sensitivity in regions with low mappability).
- Alignments may be marked as duplicate by the alignment algorithm. These reads are typically excluded from variant calling, but are counted for purposes of assessing enrichment.
- Only primary alignments are considered. Alignments marked as secondary (alternative placements) are excluded. Note that the mapping quality threshold may excluded reads with multiple secondary hits.
- Target regions are considered as provided. Reads aligning adjacent to targets may be included in the analysis by specifying the number of bases flanking the target regions that each target should be extended by.
- Mapping to target region is defined as having at least one base overlapping the target region.
- For paired-end runs, by default the reverse read must align 100%, as these are often shorter and less accurate. The default 100% alignment requires that all bases align to coordinates that are fully contained by the target region. For this test, trimmed bases are not considered. The overlap percentage required is controlled by the `enrichment.minimum.target.overlap.reverse` parameter.

Both alignment thresholds are user-adjustable, down to $1x10^{-4}$ (meaning at least one base, for reads up to 1000bp) or up to 1.0, meaning the read and target must completely overlap.

While only one list of target regions can be provided by the user, multiple target region sets may be analyzed by re-running the enrichment analysis, and specifying a different output folder or prefix for each target list. Similarly, to assess near-target alignments, reanalyze while specifying a larger number of bases by which to extend the target.

SNPs optimizations

SNP Finding parameters are optimized for Low Frequency Variant analysis. Parameters specific to LFV workflows are used, and parameters shared with other analyses have different default values for LFV workflows.

The next sections describes optimizations for Low Frequency Variant analysis.

Set the minimum allele ratio

Use the parameter dibayes.het.min.allele.ratio to set the allele ratio to a desired allele frequency.

In most experiments, setting the minimum allele ratio to be one-fifth of the lowest desired variant frequency ensures high sensitivity. For example, if you want to detect >90% of variants down to 10% frequency, set the minimum allele ratio to 2%. (This setting, of course, also generates variants with lower frequency, for example about half of variants at frequency in the sample of 2% are detected.) If you want to detect all variants down to 1% frequency, set the minimum allele ratio to 0.2%. If your coverage is not extremely high, or to increase specificity with some reduction in sensitivity, the minimum allele ratio may be set to one-third or one-half of the desired variant frequency. For example, if you are looking for variants at 1% frequency, you can set the minimum allele ratio to 0.5% or 0.33%.

Adjust sensitivity and specificity

To increase the sensitivity (detect more low frequency variants), or if the coverage is not extremely high, change the following parameters:

- *Reduce* the minimum number of unique start points of the less common allele (for example, to 3 or 4), with `dibayes.het.min.lca.start.pos`.
- *Reduce* the threshold for the difference of color QVs of the most common and less common alleles (for example, to 4), with `dibayes.het.max.diff.color.qv`.

Other parameters that can be changed to increase sensitivity include:

- *Remove* the requirement to observe the less common allele on both strands, with `dibayes.het.lca.both.strands`.
- *Reduce* the minimum average color QV of the less common allele (for example, to 20), `with dibayes.het.min.lca.color.qv`.
- *Reduce* the minimum map QV (for example, to 8), with `dibayes.reads.min.mapping.qv`.

To increase the specificity (call fewer false positives), parameters that can be changed include:

- *Increase* the minimum allele ratio, with `dibayes.het.min.allele.ratio`.
- *Increase* the minimum number of unique start points of the less common allele, with `dibayes.het.min.lca.start.pos`.
- *Include* the requirement to observe the less common allele on both strands, with `dibayes.het.lca.both.strands`.
- *Increase* the minimum average color QV of the less common allele, with `dibayes.het.min.lca.color.qv`.

Low frequency variant detection SNPs parameter table

The following table lists SNP Finding parameters and their default values in low frequency variant detection workflows.

Table 91 SNPs parameter changes for low frequency variant detection workflows

Parameter name	Default value	Comments
General parameters		
Call stringency	low_frequency	The default for Targeted Resequencing is `medium`. Always use `low_frequency` with LFV workflows.
Skip high coverage positions (Het)	0	—
Minimum mapping quality value	16	The default for Targeted Resequencing is 8.
Detect adjacent SNPs	0	—
Polymorphism rate	0.001	—
Low frequency variant filters		
Minimum average color qv of the less common allele	30	The default for Targeted Resequencing is 0.
Minimum unique start position of the less common allele	7	The default for Targeted Resequencing is 0.
The less common allele on both strands	1	The default for Targeted Resequencing is 0.
Maximum difference of color qvs of the most common and less common alleles	10	The default for Targeted Resequencing is 0.
Read filters		
Include reads with unmapped mate	True (Paired) False (Frag)	—
Exclude reads with indels	True	—
Require only uniquely mapped reads	0	—
Ignore reads with a higher mismatch count to alignment length ratio	1.0	—
Ignore reads with a lower alignment length to read length ratio	0.9	The default for Targeted Resequencing is 1.0.
General position filters		
Require alleles to be present in both strands	False	—
Minimum base quality value for a position	28	—
Minimum base quality value of the non-reference allele of a position	28	—
Heterozygous position filters		
Minimum allele ratio (Het)	0.01	The default for Targeted Resequencing is 0.15.
Minimum coverage (Het)	2	—
Minimum unique start position (Het)	2	—
Minimum non-reference color QV (Het)	7	—
Minimum non-reference base QV (Het)	28	—
Minimum ratio of valid reads (Het)	0.65	—

Table 91 SNPs parameter changes for low frequency variant detection workflows

Parameter name	Default value	Comments
Minimum valid tricolor counts (Het)	2	—
Homozygous position filters		
Minimum coverage (Hom)	1	—
Minimum count of the non-reference allele (Hom)	2	—
Minimum average non-reference base QV (hom)	28	—
Minimum average non-reference color QV (hom)	7	—
Minimum unique start position of the non-reference allele (Hom)	2	—

Run low frequency variant modules

See Chapter 18, "Run Enrichment Analyses" on page 339 for information about the enrichment module.

The instructions for other low frequency variant modules are the same as for the resequencing tools, except that the targeted modules require an additional input file, the regions of interest (ROI) file. You must provide the ROI file, in any of the following supported formats:

- a bait, probe, tile, or other file in BED format
- a text file as a list of "chr:start-end" values

To run resequencing pairing and find SNPs, complete the instructions in the following chapters and also specify your the regions of interest file:

- Chapter 9, "FAQ – Pairing" on page 182
- Chapter 10, "Run a SNPs Analysis" on page 187

The differences when these analyses are run for low frequency variants, compared to standard resequencing, are:

- The low frequency variant workflows use these SNPs parameters (which are not used in other SNPs applications):
  - `dibayes.het.lca.both.strands`
  - `dibayes.het.min.lca.start.pos`
  - `dibayes.het.min.lca.color.qv`
  - `dibayes.het.max.diff.color.qv`

  See Table 38 on page 191 for a description of these parameters.
- The find SNPs analyses are run with an input file containing only on-target reads. Additionally, find SNPs takes the processed target list as input, and restricts SNP calls to events occurring within the target regions.
- SNPs analysis is also passed the `dibayes.het.skip.high.coverage=0` parameter, to avoid excluding regions of very high coverage, which are common in enriched regions.

18 Run Enrichment Analyses

This chapter covers:

- Overview 339
- Examples of running the enrichment module 340
- Enrichment parameters 341
- Enrichment statistics input files 343
- Enrichment statistics output files 344

Overview

Enrichment

The targeted resequencing and low frequency variant workflows use the enrichment module to produce target-enriched sequencing data from the input BAM files. Given a set of coordinates representing a region of interest in a genome, a library may be enriched for reads within those regions prior to tertiary analysis. The variant calling modules are run on the filtered data sets instead of the entire read population, and limit variant calls to the target region.

Figure 31 shows the use of enrichment analysis in the targeted resequencing workflow. In low frequency variant workflows, the use of the enrichment module is similar.

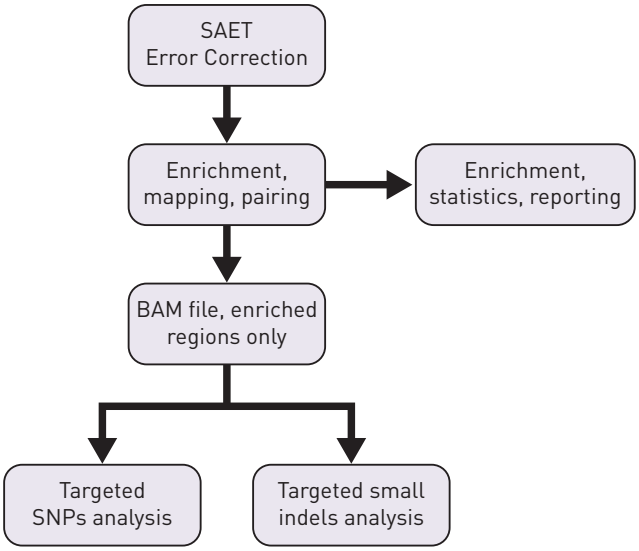


Figure 31 The role of enrichment in targeted resequencing analyses

The input list of regions of interest (ROI) is provided by the user. Supported ROI input files include third-party files of baits, probes, or tiles. The input ROI files can be in BED format or in a text file as a list of "chr:start-end" values. The mapping module sorts the input ROI file by chromosome and start position, and makes the resulting file available as output.

The tertiary analysis modules that support enriched input are the following:

- SNPs, by both targeted resequencing and low frequency variant workflows
- Small indels, by targeted resequencing workflows

Enrichment statistics

Targeted resequencing and low frequency variant workflows support optional enrichment statistics, which provide a means to assess enrichment platform performance by:

- Looking at variations in coverage, both across all targets and on a per-target basis.
- Addressing the uniformity and completeness of coverage within the target region.
- Calculating the degree of enrichment.

Examples of running the enrichment module

In a standard workflow

The following standard workflows provide examples of running the enrichment module:

- targeted.resequencing.frag
- targeted.resequencing.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository.

```
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow targeted.resequencing.frag
# define the input data
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
# specify the reference for the analysis
set reference hg19
# optionally change parameter defaults here
set analysis.regions.file /data/results/readsdata/dh10b.bed
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
```

```
ls
```

The command to set the `analysis.regions.file` does not include the optional INI file argument. As written, the command sets the `analysis.regions.file` parameter in both the secondary and tertiary global.ini files. In this way the parameter is available to all analysis modules in the workflow. If the parameter is instead set in only one INI file, for example the `enrichment.ini` file, then the regions of interest file is available to the enrichment module, but not available to the SAET module and other modules in the workflow.

See for Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information on running standard workflows. Chapter 6, "Run a Command Shell Analysis" on page 83 for information on shell commands and syntax.

As a demo analysis

The optional examples download includes an example of how to run the enrichment module by itself, as a single analysis that is not part of a standard workflow. The demo example is not recommended for general use.

See Appendix H, "Demo Analyses" on page 567 for information on running the enrichment module as a standalone analysis.

Enrichment parameters

Table 92 describes the parameters used with the enrichment statistics module.

In order to change a parameter value in your analysis, use the `set param` shell command to replace the line `# optionally change parameter defaults here` in the example commands in "In a standard workflow". For instance, to turn off the summary report, use this shell command:

```
set enrichment.summary.report 0  tertiary/enrichment.ini
```

Note: Targeted resequencing analysis requires the regions of interest file, but the user interfaces do not enforce this requirement. Be sure you specify a regions of interest file when using a targeted resequencing workflow. An example command is:

```
set analysis.regions.file /data/results/readsdata/dh10b.bed.
```

Table 92 Enrichment statistics parameters

Parameter name	Default value	Description
Required parameter		
analysis.regions.file	—	The full path to your regions of interest file. You must supply the regions of interest file and you *must* explicitly set this parameter. See "Examples of running the enrichment module" on page 340 for an example command. Set this parameter in the global INI files, not in the analysis INI files.
Optional module parameters		
enrichment.run	1	Whether or not to run the enrichment module. Allowed values: • **0**: Do not run the enrichment module. • **1**: Run the enrichment module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.

Table 92 Enrichment statistics parameters *(continued)*

Parameter name	Default value	Description
enrichment.extend.bases	0	The number of bases on either side of the target region in which alignments may be captured. The target sequence is extended by this number of bp on either side. Used by both target capture and target coverage statistics. Example: enrichment.extend.bases=0 Allowed values: Integers >= 0.
enrichment.minimum.mapping.score	8	Minimum mapping quality value (MAPQ) allowed for aligned reads. Reads below this threshold are not used. Allowed values: Integers >= 0–80.
enrichment.minimum.target.overlap	0.0001	The fraction of an alignment that must be overlapped by a target in order to be classified as on target. "0.50" is interpreted as 50%, and implies that the midpoint of the read must fall within target region. Allowed values: 0.0001 < X <= 1.0 Example: enrichment.minimum.target.overlap=0.50
enrichment.minimum.target.overlap.reverse	1.0	The fraction of a Reverse Read (F5) alignment which must be overlapped by the target in order to be classified as on-target. To require 100% alignment, use 1.0 (the default). Example: enrichment.minimum.target.overlap.reverse=.9 Allowed values: 0.0001 < X <= 1.0
Required output parameters		
enrichment.summary.report	true	Whether or not to create the summary statistics file. Allowed values: • **false**, **0**: Do not create the summary statistics. • **true**, **1**: Create the summary statistics file.
enrichment.target.coverage.stats	true	Whether or not to output per-target coverage statistics (min, max, mean) in tabular format. Allowed values: • **false**, **0**: Do not create per-target coverage statistics. • **true**, **1**: Create per-target coverage statistics.
enrichment.target.coverage.frequency	false	Whether or not to create the per-target coverage frequency histogram. Allowed values: • **false**, **0**: Do not create the per-target coverage frequency histogram. • **true**, **1**: Create the per-target coverage frequency histogram.
enrichment.target.coverage.bedgraph	false	Whether or not to create a BEDGRAPH format coverage file for on-target reads. Allowed values: true, false, 1, 0 • **false**, **0**: Do not create this file. • **true**, **1**: Create a BEDGRAPH format coverage file for on-target reads.
enrichment.genome.coverage.frequency	false	Whether or not to create a per-chromosome coverage frequency histogram. Allowed values: • **false**, **0**: Do not create this histogram. • **true**, **1**: Output a per-chromosome coverage frequency histogram.
Resource parameters		

Table 92 Enrichment statistics parameters *(continued)*

Parameter name	Default value	Description
memory.request	15GB	Memory requested per run. Must be at least 13GB to run enrichment statistics on the largest human chromosome (chr1). Specified in GB, MB, or KB. Example usage: memory.request=15GB
java.heap.space	1536	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.

Enrichment statistics input files

This section describes the input files required for by the enrichment statistics module. Parameters mentioned in this section are described in Table 92 on page 341.

Table 93 lists the two input files for the enrichment statistics module.

Table 93 Enrichment Statistics required input files

File	File format	Associated parameters	Description
target regions	BED, GFF v3, or text	analysis.regions.file	A list of target regions, either as regions of interest (ROI) or as target lists provided by vendors of enrichment platforms (such as baits, probes, or tiles files). Required, and supplied by the user.
aligned reads	BAM	analysis.input.bams	Alignments generated by the sequencing run after being matched to the entire reference genome. Managed by the framework, in the tertiary global.ini file.

The input target regions file

The target regions file may be either in BED format or a list of chr:start-end elements. An example of partial content for a target region file is shown in Figure 32.

```
chr13   31787000    31788297
chr13   31788404    31788957
chr13   31789616    31789695
chr13   31790390    31790601
chr13   31790852    31791495
chr13   31791644    31791981
```

Figure 32 Example content for an input target region file

The following information applies to the input target region file:

- Target region coordinates must be specified using the same reference used in mapping.
- Target regions should be sorted by chromosome/position. If not sorted, they will be sorted as the first step in the enrichment statistics module.
- After the first 3 required fields per line, any additional fields are ignored.

- The first field, “chrom”, must have identifiers which match the sequence names used in the BAM file. For example, for the hg18 reference:
  - Use “chr1”, not “1”.
  - Use “chrX”, not “23” or “chr23”.
- This file is in standard BED format, using zero-based indexing and a half-open interval (meaning the specified range is up to, but not including, the last position).

The input can optionally be a list of target regions in the “chr:start-end” format. This format uses one-based, closed intervals to represent genomic regions. An example of this format is:

```
chr1:1234-4321
```

Note: The input target region file and the chr:start-end list use different indexing and different interval formats.

The input aligned reads file

This file is a BAM-formatted file of aligned reads generated by the sequencing run after being matched to the entire reference genome.

The following information applies to the input aligned reads file:

- The BAM file must have a header listing sequence identifiers.

  The header is mandatory in the SAM specification for the binary BAM files.
- These sequence identifiers must be consistent with those used in the target list.
- The BAM file must be sorted by coordinate. The SO annotation in the header is not examined. All alignments within a chromosome must be contiguous. For sequential records, start coordinates within a chromosome must be equal or increasing.
- The BAM file may be from a fragment, paired-end, or mate-pair library.

  All reads are treated independently. No special handing is performed based on library type.
- Reads from paired libraries tend to have improved mapping scores based on unique paired hits, and are more likely to pass uniqueness filtering than unpaired reads.

Enrichment statistics output files

This section describes the enrichment statistics module’s output files. The output files are used in downstream analysis with LifeScope™ Software and are also compatible with external viewers. Genomic reads and annotations in GFFv3, BED, BEDGRAPH, and BAM format can be displayed with third-party graphical genome browsers. Targeted resequencing reports and tabular data can be viewed and graphed in third-party spreadsheet applications.

Table 94 lists the output files for the enrichment statistics module. These files' purpose and content are described in the following pages. The enrichment statistics parameters mentioned in this section are described in Table 92 on page 341.

Table 94 Enrichment Statistics output files

File	File format	Optional	Controlling parameter (if optional)
Output target file	BED	No	—
Output alignment file	BAM	No	—
Target coverage BEDGRAPH file	BEDGRAPH	Yes	enrichment.target.coverage.bedgraph
Genome coverage frequency file	Text	Yes	enrichment.genome.coverage.frequency
Target coverage frequency file	Text	Yes	enrichment.target.coverage.frequency
Target statistics report file	Text	Yes	enrichment.target.coverage.statistics
Enrichment statistics reports	Text	No	—

The output target file

This file is a BED file containing the sorted target list for use by later modules.

The output alignment file

This file is a filtered BAM file which includes only those alignments that overlap target regions. The file is a filtered version of the input file and retains the input file's original characteristics. The BAI index file is also created.

The default file names for the BAM and index file are:

`<input BAM file name>.ontarget.<extend_bases>.bam`

`<input BAM file name>.ontarget.<extend_bases>.bam.bai`

where <*input BAM file name*> is the name of the input file (without path or file name extension), and <*extend_ bases*> is optional and taken from the `enrichment.extend.bases` parameter, if that parameter is non-zero.

The target coverage BEDGRAPH file

This target coverage BEDGRAPH file is a text file in BEDGRAPH format and describes the depth of coverage within targets by on-target alignments. The target coverage file is generated, as determined by the `enrichment.target.coverage.bedgraph` parameter.

By default the target coverage file is created with the name `${enrichment.output.prefix}_target_coverage.bedgraph`.

Example content for a target coverage BEDGRAPH file is shown below:

```
track type=bedGraph name="On-Target Coverage" description="Depth
of Coverage of On-Target Reads"

chr1    2974349 2974349 1

chr1    2974350 2974351 2

chr1    2974352 2974352 5

chr1    2974353 2974353 18

chr1    2974354 2974354 40
```

Table 95 describes the file content. These are tab-separated text fields.

Table 95 Description of the target coverage BEDGRAPH file fields

Field name	Example content	Description
Track annotation headers	track type=bedGraph name="Coverage"	Annotations used for UCSC browser display purposes.
Contig name	chr1	Contig name of the feature. This value comes from the sequence name (SN) in the BAM header. However, in order to be viewed in the UCSC browser, these values must be of the form chr1, chr2, etc.
Feature start	148	Start location of the feature. Zero-based, inclusive.
Feature end	150	End location of the feature. Zero-based, exclusive (half-open ranges).
Coverage	1	Depth of coverage over the given (start-end) range.

The genome coverage frequency file

The genome coverage frequency file is a text file which reports frequency of the coverage depth for each reference sequence contig listed in the BAM file header. The genome coverage frequency file is optionally generated, as determined by the `enrichment.genome.coverage.frequency` parameter.

By default the genome coverage frequency file is named `${enrichment.output.prefix}_genome_coverage.frequency.tab`.

Example content for a genome coverage frequency file is shown below:

track type=bedGraph name="On-Target Coverage" description="Depth of Coverage of On-Target Reads"

chr1 0 233314335 247249719 0.943638

chr1 1 10264341 247249719 0.0415141

chr1 2 2195560 247249719 0.00887993

chr1 3 581693 247249719 0.00235265

chr1 4 216736 247249719 0.000876587

chr1 5 114047 247249719 0.000461262

Table 96 describes the content of the genome coverage frequency file.

Table 96 Description of the genome coverage frequency file fields

Column	Field name	Example content	Type	Description
1	Contig name	chr1	Text	Contig name of the feature. This value comes from the sequence name (SN) in the BAM header.
2	Coverage	0	Text, Integer	Coverage depth value.
3	Coverage count	211562282	Text, Integer	Number of positions with the given depth of coverage.
4	Total contig size	247249719	Text, Integer	Total number of base pairs in the contig.
5	Frequency of depth in contig	0.855662	Text, Float	Fraction of contig covered at given depth. Count / Total = Frequency

The target coverage frequency file

The target coverage frequency file is a text file which reports the frequency of coverage depth for each target. The target coverage frequency file is optionally generated, as determined by the `enrichment.target.coverage.frequency` parameter.

By default the target coverage frequency file name ends in `*_target_coverage.frequency.tab`.

Example content for a target coverage frequency file is shown below:

```
chr1_2974357_2974879   33   4   522   0.00766284
chr1_2974357_2974879   34   3   522   0.00574713
chr1_2974357_2974879   35   2   522   0.00383142
chr1_2974357_2974879   36   2   522   0.00383142
chr1_2974357_2974879   37   4   522   0.00766284
chr1_2974357_2974879   38   5   522   0.00957854
```

Table 97 describes the file content. The target coverage frequency fields are the same as for the genome coverage frequency file, except that location are given as target ranges.

Table 97 Description of the target coverage frequency file fields

Column	Field name	Example content	Type	Description
1	Target range	chr1_2974357_2974879	Text	One of the input target ranges. The contig name and the start and end positions from a single input target file line are concatenated.
2	Coverage	33	Text, Int	Coverage depth value.
3	Coverage count	4	Text, Int	Number of positions with the given depth of coverage.
4	Total target size	522	Text, Int	Total number of base pairs in the target.
5	Frequency of depth in target	0.00766284	Text, Float	Fraction of target covered at given depth. Count / Total = Frequency

The target statistics report file

The target statistics report file is a text file which reports per-target coverage and enrichment details in tab-delimited format, with one line per target. The target statistics report file is optionally generated, as determined by the `enrichment.target.coverage.statistics` parameter.

By default the target statistics report file name ends in `*_target_statistics.txt`.

Example content for a target statistics report file is shown below:

```
Target Name     Target Size     Minimum Coverage        Maximum
Coverage       Average Coverage       Total BP Coverage    BP Covered
at least 1X

chr1_2974357_2974879    522     33      2334    605.515  316079  522
chr1_2974634_2975198    564     0        892     210.183 118543  547
chr1_2975115_2975633    518     22      892      295.367 153000  518
chr1_2975421_2975989    568     0        571     110.783 62925   218
chr1_3092354_3092941    587     347      1875    991.16  581811  587
chr1_3150338_3150930    592     75      1301     546.51  323534  592
```

```
chr1_3291283_3291753    470     468     4436    1705.47 801571  470
chr1_3302721_3303234    513     611     3545    1804.12 925514  513
chr1_3309173_3309758    585     314     2267    940.32  550087  585
chr1_3311082_3311483    401     120     5775    2960.54 1187176 401
```

Table 98 describes the content of a target statistics report file.

Table 98 Description of the target statistics report file fields

Column	Field name	Example content	Type	Description
1	Target name (target range)	chr1_2974357_2974879	Text	One of the input target ranges. The contig name and the start and end positions from a single input target file line are concatenated with underscores.
2	Target Size	522	Text, Integer	Total size of this target range, in base pairs.
3	Min	33	Text, Integer	Minimum fold coverage of position in target.
4	Max	2334	Text, Integer	Maximum fold coverage of position in target.
5	Avg	605.515	Text, Number	Average fold coverage of all positions in target.
6	Total BP Coverage	316079	Text, Number	Enrichment fold of the target range.
7	BP Covered at least 1X	522	Text, Number	Number of bases with at least 1X coverage.

The enrichment statistics report

These reports provide data on the targeted regions in your analysis. The percent of coverage data is for the regions of interest, not for the whole genome reference. The end of the report summaries the number of target bases covered at various depths. The file name pattern is `*enrichment_report.stats`. Example content is given below.

```
Run     Reads On        Percent On      Reads Off       Percent
Off     Enrichment Fold
./outputs/enrichment/output.bam     69260675      88.6216%
8892585 11.3784%        1338.33

Reads In Targets:       69260675        88.6216%
Reads Off Targets: 8892585      11.3784%
Ratio of Percent on/off Target: 7.78859
Total Target BP: 2039803
Total Genome Size: 3080436051
Ratio of target to genome: 0.00066218
Enrichment fold relative to target size: 1338.33

# Targets Not Covered   Target Bases Not Covered        Percent
of Target Bases Not Covered     Percent of Target Covered >= 1X
Percent of Target Covered >= 5X Percent of Target Covered >= 10X
Percent of Target Covered >= 20X       Average Depth of Target
Coverage
1       12061   0.59%   99.41%  98.98%  98.63%  98.22%  2351.24

Number of target regions with no coverage: 1
Percent of target bp not covered: 0.59% (12061 bp)
Percent of target bp covered at >= 1X: 99.41%
```

```
Percent of target bp covered at >= 5X: 98.98%
Percent of target bp covered at >= 10X: 98.63%
Percent of target bp covered at >= 20X: 98.22%
Maximum Depth of Coverage within targets: 49453
Average Depth of Coverage within targets: 2351.24
```

PART VI
Whole Transcriptome Analyses

19 Run a Whole Transcriptome Mapping Analysis

This chapter contains:

- Overview 353
- Examples of running the WT mapping module 355
- Input files 356
- Stages of mapping 359
- Quality alignments and unique alignments 368
- Single-read mapping parameters 369
- Paired-end parameters 375
- Mapping output files 380
- Mapping statistics 386
- FAQ – Whole transcriptome 393
- FAQ – BAMStats 396

Overview

High-throughput sequencing of the transcriptome using the SOLiD® System enables genome-wide expression profiling with high sensitivity and a wider dynamic range than microarray technology. The whole transcriptome library preparation also preserves the strandedness of the RNA transcripts. Preserving the strandedness simplifies data analysis, allows determination of the directionality of transcription and gene orientation, and facilitates detection of opposing and overlapping transcripts.

You can use the 5500 Series Genetic Analysis Systems to sequence RNA prepared with RNA-Seq sample preparation kits. RNA sequencing produces high-depth, short-read sequencing data that can be used to measure RNA expression. Like microarray analysis, RNA-Seq measures expression intensity across many genomic features. Unlike microarray analysis, RNA-Seq can be used to identify novel transcriptome features in a sample.

Like other SOLiD® System applications, RNA-Seq produces short reads in the form of XSQ files. Whole Transcriptome Analysis (WTA) may be applied to these reads to produce alignment, coverage, count, and splice-finding results. WTA is a reference-based analytical method, which means a reference genome is required as input. Much WTA functionality also requires gene annotation associated with the reference genome sequence.

WTA analysis can be divided into secondary and tertiary analysis. Secondary analysis produces the aligned reads using sequences and quality values as input. Tertiary analyses produce biologically meaningful results (gene expression level, feature discovery, etc.) from aligned reads. This chapter describes how WTA is used to align reads to a reference genome, the methods used by WTA, and the configurable parameters of WTA.

The 5500 Series Genetic Analysis Systems produces RNA-Seq reads in both single-read and paired-end configurations. While there is overlap in the analytical techniques, the analysis of single-read and paired-end data is performed with two distinct pipelines. The single-read pipeline consists of alignment, counting, and coverage steps. The paired-end pipeline extends single-read analysis with an alternative alignment step and an additional junction finding step.

The aligned reads produced by WTA can be used with the following tertiary modules:

- **Count annotations –** Computes the number of reads that align within genomic features.
- **Coverage –** Computes the read coverage at each genomic position.
- **Splice finder –** Identifies splice junctions of various types, including fusion junctions from paired-end reads.

The 5500 Series Genetic Analysis Systems™ System produces RNA-Seq reads in both single-read and paired-end configurations. While there is overlap in the analytical techniques, the analysis of single-read and paired-end data is performed with two distinct pipelines. The single-read pipeline consists of alignment, counting, and coverage steps. The paired-end pipeline extends single-read analysis with an alternative alignment step and an additional splice finding step.

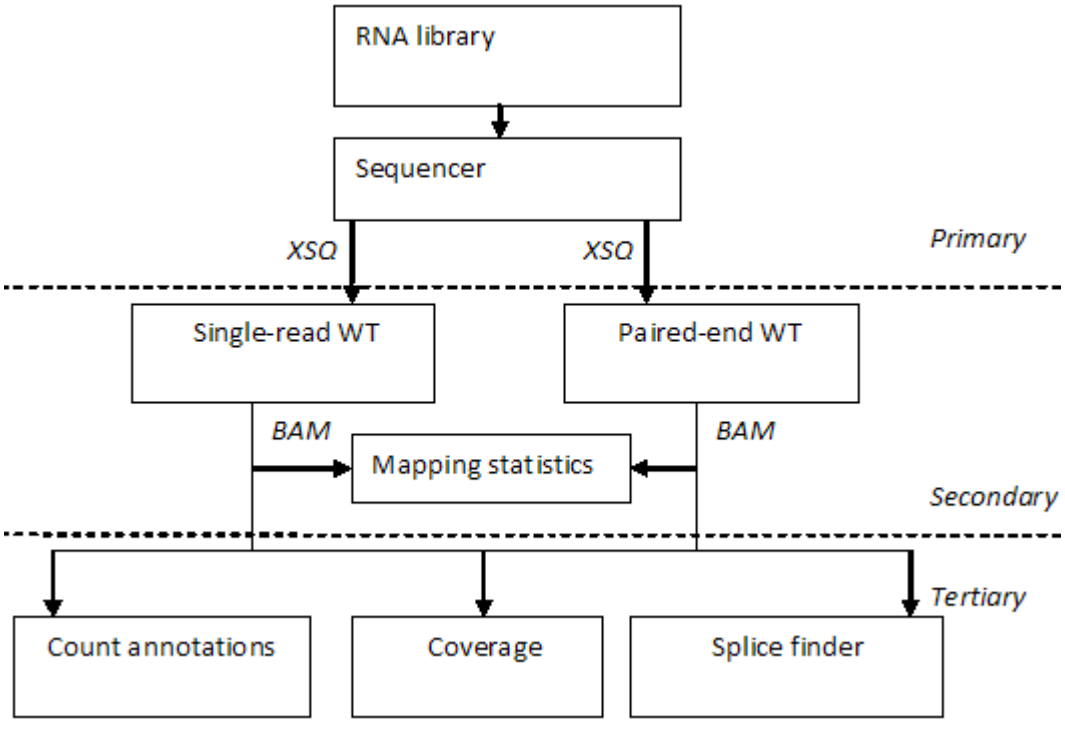


Figure 33 WTA single-read and paired-end internal workflows

Examples of running the WT mapping module

In a standard workflow

The following standard workflows provide examples of running the WT mapping module:

- whole.transcriptome.frag
- whole.transcriptome.pe

The following are example shell commands using reads and references from the LifeScope™ Software repository. This example turns off tertiary modules in the workflow.

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# make a project, and open it
mk ecoli
cd ecoli
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow genomic.resequencing.pe
# define the input
add xsq run0209a_50_PE.xsq
add xsq run0209b_50_PE.xsq
# specify the reference to be used
set reference hg19
# these commands turn off the tertiary modules
set coverage.run 0 tertiary/coverage.ini
set splice.finder.run 0 tertiary/splice.finder.ini
set wtcounts.run 0 tertiary/wtcounts.ini
# optionally change mapping parameter defaults after this line
# list the configuration of the analysis
ls
# start the run
run
# list progress information about the run
ls
```

In order to change a mapping parameter value in your analysis, use the `set param` shell command after the line `# optionally change mapping parameter defaults after this line`. For instance, to turn on the `mapping.in.base` parameter, use this shell command:

```
set mapping.in.base 1 secondary/wt.pe.map.ini
```

This command shows how to turn on the `mapping.in.base` parameter for the `whole.transcriptome.frag` workflow:

```
set mapping.in.base 1 secondary/wt.frag.map.ini
```

See Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information on running standard workflows.

As a demo analysis

The optional examples download includes an example of how to run the whole transcriptome analyses outside of a standard workflow. A standard workflow is recommended.

See Appendix H, "Demo Analyses" on page 567 for more information on running a demo analysis.

Input files

Reads input files

The WT mapping module accepts as input one or more input XSQ files. The input reads can have different read lengths but they must be of the same library type (fragment or paired-end). For example, paired-end read-sets of length 50x25 and 60x25 can be processed together, but a paired-end read-set cannot be processed with a fragment read-set.

RNA-Seq reads

The RNA-Seq reads used in WTA are different from the genomic reads. For RNA-Seq reads:

- Only transcribed sequences are measured by the system.
- Genome coverage is non-uniform due to variation in transcriptional intensity.
- A large subset of reads originate from uninteresting sequences such as ribosomal RNA (rRNA).
- A subset of the reads originates from splice junctions and do not align contiguously on the genome.

Reads input specification

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. The following factors control how your input data is analyzed:

- **Index (barcode) IDs –** Using an index ID restricts your input to the reads data of one or more indices.
- **Grouping of reads –** Each group of reads is analyses together as one specimen. The output data for a group is combined into one set of results files.
- **Multiple sample runs –** Unrelated reads can be processed together in one run of LifeScope™ Software, but analyses separately as separate input data.

See "Define input data" on page 98 for more information on designing how to add input data to your analysis.

Mapping one tag of paired data is not supported.

Legacy data

If the data you want to process with LifeScope™ Software is in CSFASTA and QUAL files, these files must be converted to the XSQ file format, through one of the following methods:

- The LifeScope™ Software UI handles converting these files to the XSQ format before mapping.

- If you are using the LifeScope™ Software command shell, you must first convert the CSFASTA and QUAL files to the XSQ format. See Appendix D, "XSQ Tools" on page 519 for information on the standalone XSQ converter.

Reference input files

WTA uses several files defining reference information: filter-reference, genome-reference and annotation files. The filter-reference file is a FASTA-formatted file defining uninteresting reference sequences such as Ribosomal RNA, tRNA, or vector sequences. To increase the speed of analysis, reads that align to sequences in the filter reference are excluded from downstream alignment steps. Many of the entries in the filter reference are species-specific. The filter reference file must be appropriate for the species being studied.

The principal output of alignment analysis is a set of alignments between the reads and the genome-reference in BAM format. The genome-reference file is a FASTA-formatted file defining sequences in the reference genome.

The annotation file is a GTF-formatted file defining known genes, transcripts, and exons in the genome reference. See "Annotations input files".

Junction-reference and exon-reference files are generated internally by the software. These files are entirely derived from the GTF and genome-reference files. A user is not required to provide these references. The junction-reference file is a FASTA-formatted file with an entry for every potential intragenic splice-junction defined in the annotation file. The entry contains the position of the junction and the associated flanking sequence. The exon-reference is a FASTA-formatted file with an entry for every exon defined in the annotation file.

Annotations input files

While genomic resequencing relies only on alignment to a reference sequence, WTA also makes use of gene annotations. Gene annotations define the exons, genes, and transcripts used to improve alignment.

WTA modules use genome annotation files in GTF format. The following sites explain the GTF format:

genes.cse.wustl.edu/GTF2.html

genome.ucsc.edu/FAQ/FAQformat.html#format4

The GTF file must match the genome reference to ensure that the WTA modules work correctly.

IMPORTANT! Make sure the GTF file is for the same genome assembly as the FASTA file, and that matching sequence identifiers are used. Gene and transcript identifiers in the GTF file must be properly normalized. Identifier normalization is a known issue in GTF files from the UCSC Genome Browser, which is a popular source of GTF-formatted annotation. The UCSC GTF files always report the same value for gene and transcript IDs.

LifeScope™ Software includes a script (`refgene2gff.sh`) that produces a compliant GTF file from the UCSC Genome browser database.

UCSC genome annotations

Format UCSC Genome Browser Database annotations for WT analyses

The UCSC Genome Browser Database has genome annotations available for many assemblies at

hgdownload.cse.ucsc.edu/goldenPath/

The GTF-formatted annotations available for download are not properly normalized by gene ID. The required content is present for each assembly in the file export of the refGene database table `database/refGene/txt/gz`.

For example, annotation for human genome build 18 is available at:

hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz

Note: The GrefGene.txt.gz file is not in GTF format. You must convert the annotation before using it in WTA.

Convert the refGene.txt.gz file

Run the script `bin/refgene2gff.sh` to convert the `refGene.txt.gz` file:

```
gunzip refGene.txt.gz
refgene2gff.sh -i refGene.txt -o refGene.gff
```

Genome annotations that are downloaded from the UCSC Genome Browser and converted by the annotation conversion script are preferable because they contain Human Genome Organization (HUGO)-style gene names. HUGO-style gene names allow interpretation when using a genome browser or reading reports.

The annotation conversion script works with the latest format of `refGene.txt` files. Assemblies, such as the rat genome, use an alternative format for the `refGene.txt` file. The `refgene2gff.sh` script does not convert alternative formats.

Ensembl GTF files

Format Ensembl GTF files for WTA pipelines

The Ensembl website **ensembl.org**/ has GTF-formatted genome annotations available for many popular assemblies. Unlike the GTF files directly downloadable from UCSC (see "UCSC genome annotations" on page 358), Ensembl GTF files are properly normalized by gene and transcript IDs. Visit the following site to download an Ensembl GTF file:

http://www.ensembl.org/

Ensembl GTF files use gene accession numbers instead of HUGO-style gene names. Ensembl GTF files also use unprefixed sequence identifiers, such as 1,2,3….X,Y,MT. The Ensembl GTF files are incompatible with genome reference FASTA files that have UCSC-style sequence IDs with the prefix "chr", for example, chr1, chr2, chr3….chrX, chrY, chrM.

Reformat the Ensembl GTF file

To reformat the Ensembl GTF file to use UCSC-style gene IDs, run the `reformat_ensembl_gtf.pl` script, and redirect the output to a new file with `.reformatted` in the file name.

```
For Example:
reformat_ensembl_gtf.pl Homo_sapiens.GRCh37.63.gtf > \
    Homo_sapiens.GRCh37.63.reformatted.gtf
```

Stages of mapping

A WT mapping run consists of the following phases:

- **Scatter –** The entire set of reads is divided into small non-overlapping subsets and defines one mapping job per subset.
- **Mapping –** A single computer processes a mapping job, which takes reads as input and produces aligned reads as output.
- **Gather –** The results of multiple mapping jobs are combined to produce final mapping result.

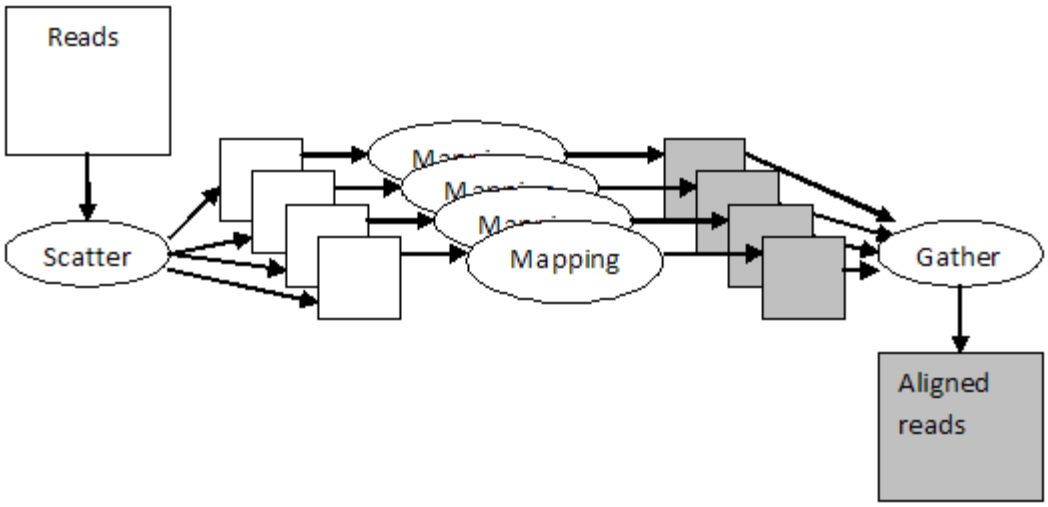


Figure 34 WT mapping phases

The Mapping phase alone computes the alignment records that compose the results. This phase can be broken down into several steps:

- **Filter mapping –** Reads that align to the filter reference are counted and excluded from further analysis.
- **Genome mapping –** Reads are aligned to the genome reference.
- **Junction mapping –** Reads are aligned to the junction reference (F3 reads only).
- **Exon mapping –** Reads are aligned to the exon reference (F5 reads only).
- **Alignment merge –** Results from genome and junction or exon mapping are combined to produce a single set of alignments for each read type (F3 or F5).

- **Exon table rescue –** Supplements the set of alignments by conducting a more sensitive search for a alignments of a read (target) near the alignments found for the other member of the pair (anchor). These regions are selected using the gene structure of the genome. (Paired-end only).
- **Pairing –** Selection of most likely alignments based on alignment quality and the distance between alignments of each member of a pair. (Paired-end only).
- **BAM file generation –** The process producing results in BAM format. For color-space data, the base translation step is also included. Due to the scattering step, the BAM files at each compute node are a sub-set of the final BAM file and are referred to as baby BAMs.

  BAM generation is composed of several substeps.
  - **Ma2BAM**, **Pa2BAM –** A BAM file (lacking actual sequences or quality values) is written. Mapping quality values are computed (MAPQ)
  - **Refcor –** Reference-assisted color-to-base translation. Sequences and quality values are added to the BAM records.
- **Sorting –** Sorting the BAM file by genome coordinate.

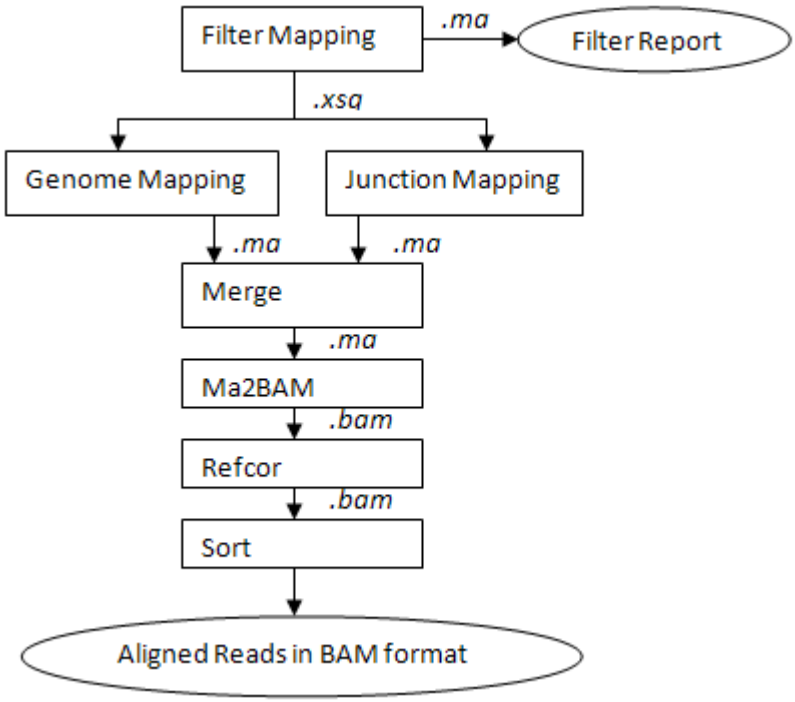


Figure 35 The phases of mapping WT fragment data

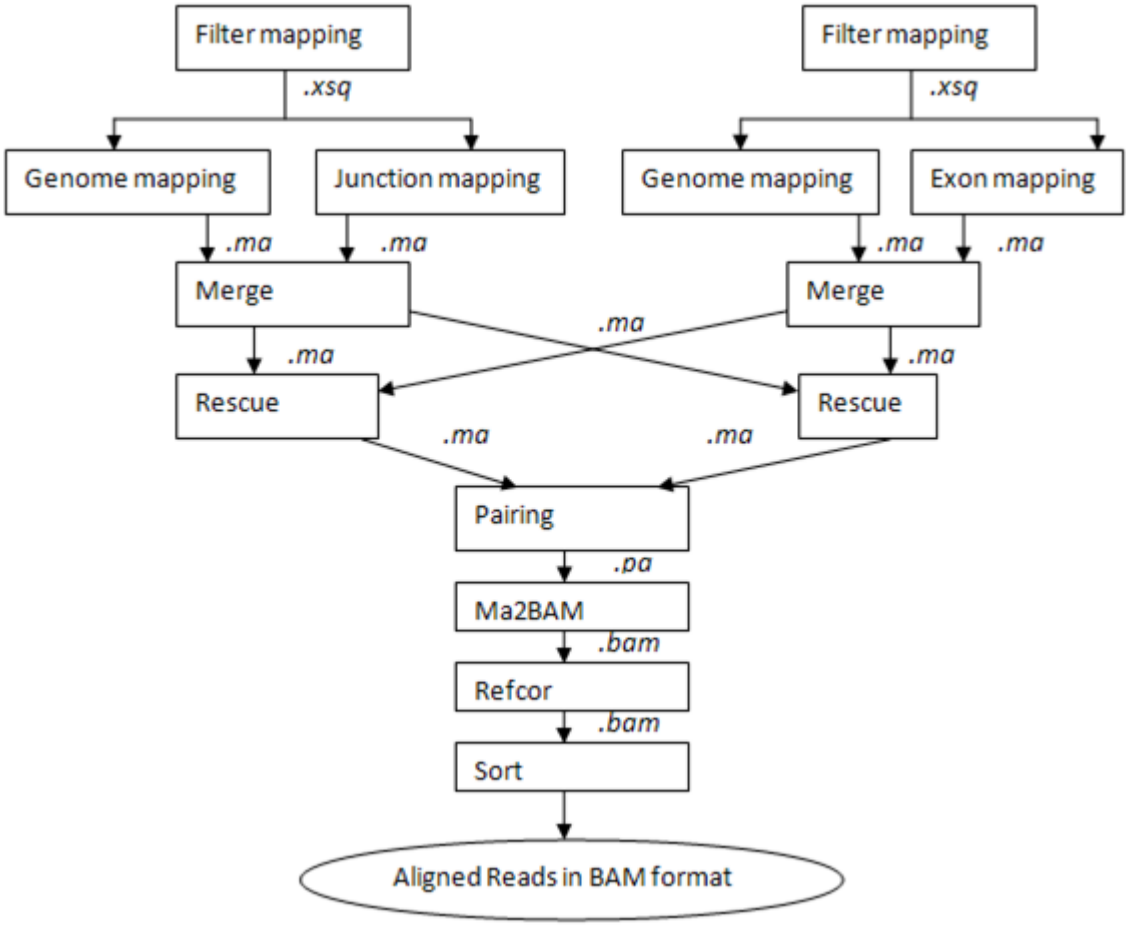


Figure 36 The phases of mapping WT paired-end data

Scatter

The scatter stage distributes reads to the available compute nodes, in order to gain the efficiency benefits of parallel computing.

The mapping module can accept multiple read-sets, which are allowed to have different read lengths. The input set of read-sets from multiple XSQ files is first split based on read length. Fragment read-sets are partitioned into separate categories based on these read length categories:

- $0 \leq \text{Length} < 25$
- $25 \leq \text{Length} < 35$
- $35 \leq \text{Length} < 50$
- $50 \leq \text{Length} < 75$
- $75 \leq \text{Length}$
- Trimmed reads, with variable read lengths

The reads from each category are scattered into multiple jobs based on the parameters `fragmap.minreads.per.node` and `fragmap.number.of.nodes`. These parameters are described in the section "Mapping performance" on page 371.

Single-read mapping

WT Single Read mapping produces a set of aligned reads. Stated briefly, the method performs the following steps (see Figure 35 on page 360).

- Remove reads that align to the filter reference.
- Map reads to genome and junctions.
- Merge results.
- Convert to BAM format.

Alignment finding uses a proprietary algorithm (see Chapter 9, "Run a Resequencing Mapping Analysis" on page 137). The filter mapping step identifies and counts the reads that align to the filter reference. The counts are written to the filter report. Reads that align to entries in the filter reference are omitted from further analysis and do not appear in the set of aligned reads produced by the module.

The genome mapping step finds alignments between the reads and the genome reference. Likewise the junction mapping step finds alignments with the junction reference. Alignments of a read to a genome and to junctions can produce multiple similar alignments at the same location. When multiple similar alignments occur, the alignment with the highest alignment score replaces all others. If there is a tie, the genomic alignment is used. The alignment score is calculated as follows:

$$Score=len-nm\times 1(1+mp)-jp$$

where:

- *len* = number of colors in the alignment
- *nm* = number of color mismatches
- *mp* = mapping mismatch penalty
- *jp* = penalty for alignment to a junction

The parameters to control mapping algorithm are in Table 101 on page 370. For details of the mapping algorithm, see "Mapping schemes" on page 374.

Filter mapping

The reads are mapped to the filter reference in the filter mapping step. Reads that align to the filter reference are called filtered reads. The result of this mapping step is used to populate the filter report. In the single-read pipeline, filtered reads are populated in a separate BAM file. Filtered reads are omitted from the BAM file. A filter reference for use with human reads is available with LifeScope™ Software. This reference contains:

- SOLiD® adapter sequences
- Human ribosomal RNAs (rRNAs)
- Human transfer RNAs (tRNAs)
- Other human sequences
- Single-base-repeat sequences, including Poly-A, Poly-T, and more

Complete the following steps to construct a filter reference for another species.

1. Copy the adapter and single-base-repeat sequences to a new file.
2. Append FASTA-formatted species-specific ribosomal, transfer RNA, and other sequences to the filter reference.

Junction mapping

Reads are mapped to the flanking sequence of junctions defined by the genome annotation in the junction mapping step. Junctions are inferred from the exon, gene, and transcript definitions in the genome annotation. Junction sequences are produced for all splices between pairs of exons within a gene. Junctions present in annotated transcripts are called known junctions. Junctions not present in annotated transcripts are called putative junctions.

Exon mapping

In the exon mapping step, paired-end reads are mapped to the set of exon sequences defined by the genome annotation. This step maps the shorter F5 reads in the paired-end pipeline. By default, more mismatches are allowed in exon mapping of F5 reads in the paired-end pipeline. Single-read mapping skips this stage.

Table 99 describes a human exon sequence database and a human junctions database.

Table 99 Comparison of alternative human references

	Reference	Number of references	Bp size	Mappability	Novelty	Junctions
F3	Genome	25	3 billion	OK	OK	—
	Junctions	2,012,075	201,207,500	OK	—	OK
	Refseq	32,699	99,02,749	OK	—	Partial
F5	Genome	25	3 billion	—	OK	—
	Exons	216,884	99,026,749	OK	—	—
	Refseq	32,699	99,026,749	OK	—	Partial

In a paired-end alignment, the F3 and F5 paired-ends are initially processed through independent mapping paths. Similar to single-read-mapping, the F3 reads are mapped to filter, genome, and junction sequences. The F5 reads are also mapped to filter, genome, and exon sequences. Reads that map to filter sequences are discarded and do not proceed further in the pipeline.

The F3 genomic and junction mapping results are merged into a non-redundant set of alignments. The F5 genomic and exon mappings are merged into a non-redundant set of alignments. At this point in the analysis, the output consists of a set of independently processed F3 and F5 genomic alignments.

Rescue method

You can optionally use the rescue method to find additional alignments. Rescue is an alignment method that is applied to read pairs that have at least one alignment, but no pair of alignments occurring within an expected range (see Figure 37 on page 364). The expected range is set to 100,000 bases by default.

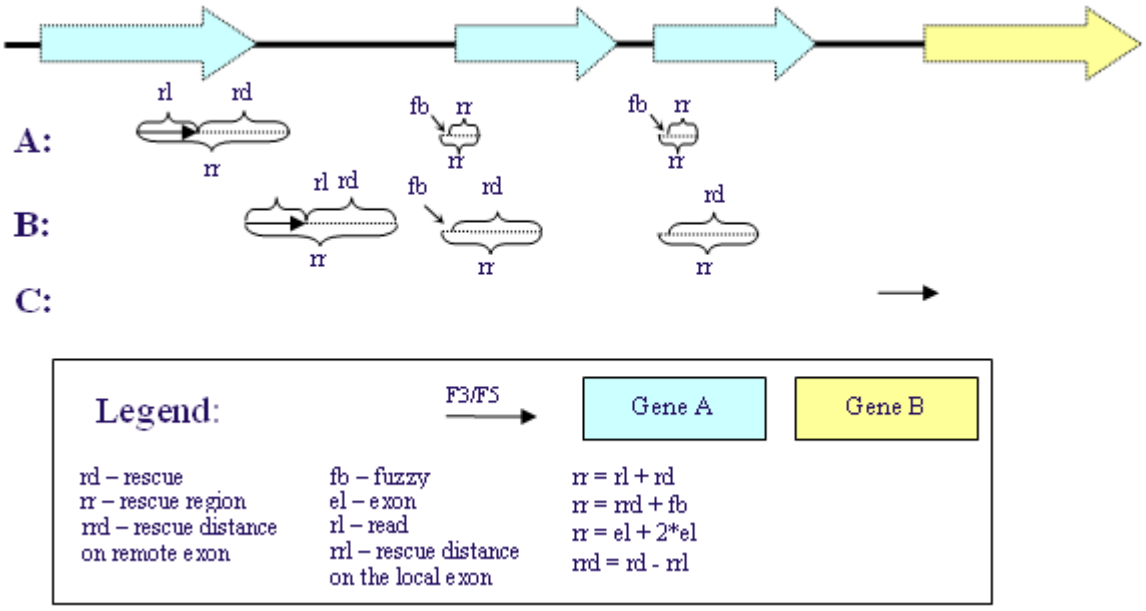


Figure 37 Annotation-aided rescue for WT

The next section refers to Figure 37.

Row A

For alignments that fall on an exon of a gene, and do not have a mate alignment above a certain score threshold, a special exon rescue is performed. Rescue region (rr) is defined as a rescue distance that is downstream of the alignment. The rescue distance is determined by user-defined thresholds. The rescue distance starts from the left-most position of the alignment for overlapping mates. The rescue region also includes certain position range in downstream exons of the same gene. The approach defined in the previous sentences helps rescue mates on different exons.

Row B

A rescue is still performed on downstream exons if a read is mapped on the intron of a given gene.

Row C

No special exon rescue is performed if an alignment falls on an intergenic region. Instead, regular rescue within the defined pairing distance is performed.

Alignments that lack a sibling read aligned nearby are called anchors. If anchors are found, the rescue tool conducts a more sensitive search for the sibling near the anchor, within a limited region of the genome. In the WT paired-end pipeline, the search is limited to anchors that occur within the introns or exons of annotated genes, with an allowance for a few overhanging bases.

Rescue is performed only within a set of expected rescue distances determined by a gene's exon structure and the insert size distribution. Rescue distances are a function of the transcript rescue distance:

transcript rescue distance = mean insert size + 3 standard deviations

This distance is longer than the majority (99.7%) of inserts. The formula above describes the rescue distance without taking into account the presence of introns. Because inserts are very likely to contain introns, a rescue distance is calculated for each potential splice configuration of the gene. For each configuration, the rescue distance is the transcript rescue distance + length of introns within the relevant transcribed sequence interval. Alternatively, rescue can be performed on all exons to improve sensitivity, but with a substantial increase in both false positives and run time. Rescue can be applied using F3 anchors only, F5 anchors only, or both.

Regardless of the rescue steps employed, the rescue results supplement the alignments detected in individual mapping steps. The end result of rescue is a set of F3 and F5 alignments in the same format as that produced by mapping.

Pairing reads

In the pairing step, pairs of reads are evaluated, assigned a mapping quality value, and written to a BAM file. The pairing range is set to 100,000 so that reads in adjacent exons are tagged as proper pairs. Unlike the pairing of genomic resequencing data, the mapping quality of read pairs is a function of genome annotation. Alignment pairs that do not occur within the same gene are penalized. For each pair, the alignment with the highest quality value is designated as the primary alignment. In the case of multiple highest quality, a single-pair alignment is selected randomly. Figure 38 shows an example of pairing range calculation with junction alignments.

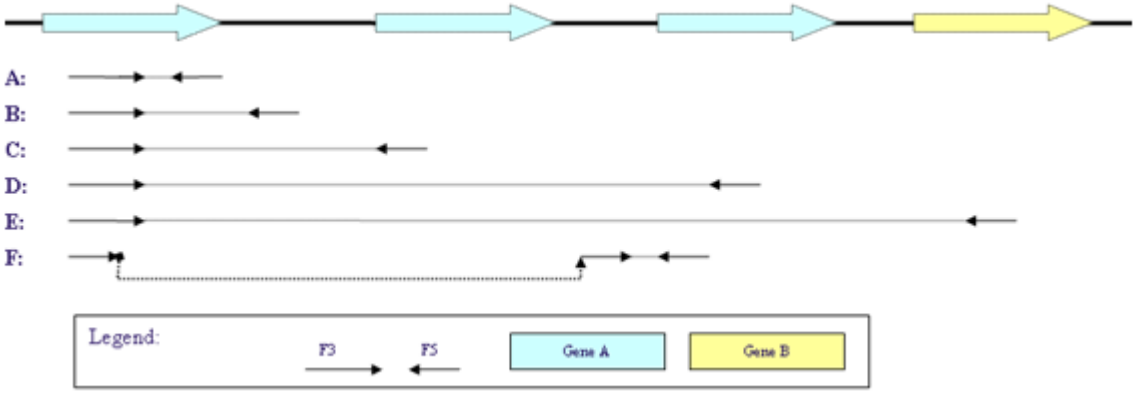


Figure 38 WT annotation-aided alignment Pairing Quality Value (PQV)

See Table 100 for details about rows A to F in Figure 38.

Table 100 Annotation-aided alignment PQV description

Row(s)	Description	PVQ penalized?
A	Mates fall on the same exon.	No
B	One mate falls on an exon and the other falls on an intron.	Yes
C and D	Mates fall on separate exons of the same gene.	No
E	Mates fall on exons of different genes.	Yes
F	Spliced alignment where one mate partially falls on a known gene and the other falls on the exon of the same gene.	No

BAM file generation

Ma2BAM, Pa2BAM

Ma2BAM and pa2BAM are format converters that change match files to a pro-BAM format. These pro-BAM files do not contain sequence or quality information but contain relevant BAM meta data. The pro-BAM files are sorted by incoming bead-id order. Relevant meta data from XSQ files is captured in pro-BAM files at this stage. The parameters `bamgen.primary.output.filter.type`, `bamgen.mqv.threshold`, and `create.unmapped.bam.files` control the functionality of the ma2BAM and pa2BAM utilities. The pro-BAM files are generated on the scratch folder.

Ma2BAM takes output of mapping (MA files) and converts them to pro-BAM format. The pa2BAM takes output of pairing (PA files) and converts them to pro-BAM format. During BAM generation, the mapping and pairing quality values are also computed.

For reads with multiple ungapped alignments, the read with the highest mapping or pairing quality value is chosen as the primary alignment for the read, and is reported to the BAM file. In cases where there are multiple alignments with the same quality value, the primary alignment is chosen at random from among the alignments with the same quality value. The details of calculating mapping and pairing quality values are described below.

Mapping and pairing quality values

This section describes mapping quality value (MQV) and pairing quality value (PQV) scores. Quality values are Phred-scaled quality scores. For any given alignment, an MQV is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. A PQV represents the confidence in both alignments of a pair, and is a combined quality score for both reads. Quality values are within the range 0–100:

- **0 –** The highest probability of error
- **100 –** The lowest probability of error

The following factors increase confidence and increase a read's quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce a read's quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

The pairing algorithm reports multiple sets of possible alignments for any given pair of reads (F3/R3 tags for a mate-pair run and F3/F5-P2 tags for a paired-end run). The pairing quality algorithm uses a Bayesian approach to calculate the quality of a given alignment for a pair of reads and the alignment with the highest pairing quality value (PQV) is chosen as the primary alignment for the pair of reads. The PQVs represent the Phred-scaled quality score, and they are useful for downstream variant detection modules such as diBayes, small indels, large indels, and CNV.

In order to be consistent with the Phred quality score (-10*log10[prob(error)]) used widely in literature, the quality is computed as the negative log odds of misaligning the read (or pairs of reads, for PQV). The resulting quality values are normalized by the maximum possible value to ensure that the quality values are within the range 0–100.

Gapped alignments

The pairing algorithm searches for gapped alignments (indels) when one of the tags (F3/R3/F5-P2/F5-BC) maps to the reference genome and the other tag does not map to the genome within the insert-size range, or it maps within the insert-range but the alignment is short. If both an ungapped and a gapped alignment are found for a given read, then, due to the low prior probability of 10^-4 assigned to the gapped alignments, the PQV for gapped alignments is zero.

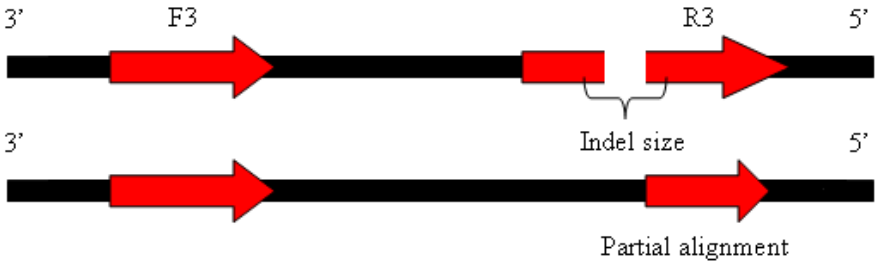


Figure 39 Example of a gapped alignment and a partial alignment

In calculating the PQV for gapped alignments, the alternative hypothesis tested is the probability of finding the partial ungapped alignments. The read with the gapped alignment is treated as two partial reads on either side of the indel start point. The partial read with the greater length is used as the partial alignment length for the alternate hypothesis.

Refcor

Reference Assisted Color-to-Base Translation (also known as refcor) combines information from color calls, optional ECC extra primer round calls, and the reference sequence, to enhance the quality of base-space reads by detecting and fixing color errors that convert into multiple base-space errors.

Refcor accepts a set of pro-BAM files, which contain alignments information without base or color sequence, and generates a corresponding set of BAM files containing alignments with base-space and color-space calls (on the scratch folder). These BAM files are sorted by bead-ID and contain dummy base calls and QVs, but are otherwise compatible with the BAM specification. The refcor module produces bead-ID-sorted BAM files that have valid calls and QVs.

For each alignment, the module constructs the primer-transition graph with vertices corresponding to all possible k-mers (k=4 for 2BE+4BE, k=2 for 2BE) in each read position. Two adjacent vertices are connected with an edge if they overlap in *k*–1 bases. A score corresponding to the (1-Pe), where *Pe* is the probability of being erroneous, is assigned to each of the vertices. Scores are derived from quality values corresponding to enumerated calls. When making a base call in a position, the current algorithm considers the cumulative probability of all sequences that have called base in that

position. The evidence from the reference (base) is assigned to the vertex with *k*-mer ending in corresponding base only if the color transition between current and following reference bases matches the color call in a read. Such assignment reduces the reference over-correction in SNP positions. In order to reduce reference over-correction in positions with color errors, we assign to reference base a very low weight corresponding to QV of 8. The weight can be reduced to QV of 0, completely reducing the reference guiding in the color-to-base translation. This results in a smaller skew between reference to non-reference allele ratio. Increasing the weight results in false positive variant calls reduction. This process restores the phase shift caused by color errors or completely eliminates the errors.

The parameters `bamgen.refcor.addcs`, `refcor.reference.weight,` and `refcor.base.filter.qv` control the refcor program. The defaults provided are validated for a wide range of data.

- If the XSQ file contains base-space data only, the XSQ base call and quality value are transferred to BAM file. In this case, no color calls are present in BAM file irrespective of the value of `bamgen.refcor.addcs`.
- If the XSQ file contains 2BE color-space data only, refcor uses aligned 2BE and reference to generate base translation for the aligned part of the read. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, bases are determined by naively translating color to base. For these unaligned reads, the QV for the base at position *k* is the minimum color QV for all positions ≤ *k*.
- If the XSQ file contains ECC data, then reference with 2BE and 4BE color are used to generate the new base sequence. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, the original base-space data from ECC is preserved.

Sort BAM

The output of refcor is a set of BAM files in bead-id sorted order. These files are sorted in this step using Picard sort into coordinate sorted order before merge. These BAM files are referred to as mini-BAM files and are stored in the analysis temporary directory.

Merge BAM

This phase accepts a set of mini-BAM files and outputs a set of BAM files. These BAM files are stored in the analysis output directory (which is typically a network-attached storage device).

One BAM file is generated per read-set. For a 96-barcoded dataset in a single XSQ file, either 96 or 97 BAM files are created. The 97th BAM contains reads that have the default barcode index.

Quality alignments and unique alignments

We strongly recommend using mapping quality values, rather than uniqueness, as the standard for recognizing or filtering good alignments. A mapping quality value estimates the probability that the alignment is correct, and takes several indications into account, including the number of alignments or uniqueness of an alignment.

The parameter `bamgen.mqv.threshold`, new in release 2.0, is available to filter alignments written to your BAM output files. In order to limit your BAM files to good quality reads, set `bamgen.mqv.threshold` to a positive integer such as 8 or higher. This technique ensures that you do not see poor quality reads in your BAM file, and that your tertiary analysis is limited to good quality reads.

How a quality value is derived

The following factors increase confidence and increase an alignment's quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce an alignment's quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

How a quality value is interpreted

For any given alignment, a quality value is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. Quality values are within the range 0–100:

- **0 –** The highest probability of error
- **100 –** The lowest probability of error

Quality values are Phred-scaled quality scores, logarithmically linked to error probabilities. A mapping QV of 10 means that we calculate there is a 90% chance that the alignment is correct. A mapping QV of 20 means that we calculate there is a 99% chance that the alignment is correct.

Unique alignments

Unique alignments might be confused with high quality alignments, but uniqueness is not by itself a good criteria to judge an alignment. Uniqueness only means that one step of secondary mapping, the aligner, reports one alignment. This information is a part of the total information generated during mapping, but does not contain all of the information known about the alignment. In contrast, the mapping quality value is a numerical representation of all considerations about the alignment.

For example, a read might have multiple alignments. If the quality value for only one of those multiple alignments is high, that probability of that alignment being good is high. That alignment should be used in your analysis, even though it is not unique.

For a unique alignment, its quality value must be examined to determine if the alignment should be taken or filtered out. Its uniqueness by itself is irrelevant to the decision about whether it is a good read.

Single-read mapping parameters

Table 101 describes the algorithm parameters that you can configure for single-read whole transcriptome mapping.

This table lists mapping, resource, parallelization, and BAM-generation parameters. WT tertiary parameters are listed in their chapters:

- Chapter 20, Run a WT Coverage Analysis
- Chapter 21, Run a WT Count Known Genes and Exons Analysis
- Chapter 22, Run a WT Splice Finder Analysis.

Table 101 Single-read WTA mapping parameters

Parameter	Default	Description
Mapping parameters		
enable.filter.mapping	1	Enables filter mapping.
enable.junction.mapping	1	Enables junction mapping.
mapping.in.base	false	Whether to map in base space (true) or color space (false). Set to true if input data has base space available. Allowed values: • **false**: Map in color space. • **true**: Map in base space. If only color space is available, then the module fails when base space mapping is turned on.
wt.merge.min.junction.overhang	8	Alignments to junctions are not reported if they have fewer than this number of bases unaligned on either side of the splice. Allowed values: Integers 0–100.
Optional resource parameters		
java.heap.space	1500	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.
mapping.memory	15gb	The total memory, in gigabytes, that is available for map reads, per compute node. Include the units gb in the setting. This number is set to 15 GB because the minimum hardware requirement for memory is 16 GB. **Note:** Typically, the job scheduler will allocate on nodes that have the requested memory available. For mixed hardware, it is suggested that different queues are created based on the memory available. Mapping jobs should be launched on high memory systems. For human references (hg18, hg19) and a 25.2 scheme, 45 GB is recommended for fastest performance. The smallest recommended RAM for human reference is 19 GB.
mapping.schema.file	—	The mapping schemas file.
Optional parallelization parameters		
fragmap.number.of.nodes	4	The number of compute nodes available for this analysis.
fragmap.max.number.of.jobs	100	The maximum number of nodes allowed for mapping.
fragmap.minreads.per.node	4000000	If the total number of fragments exceeds this amount, then the analysis is distributed across the available nodes. Allowed values: Integers 1–125000000 (125 Million)
fragmap.maxreads.per.node	150000000	The maximum number of fragments that can be processed on a single node. Allowed values: Integers 1–150000000 (150 Million)

Table 101 Single-read WTA mapping parameters *(continued)*

Parameter	Default	Description
processors.per.node	8	The number of processors per node use for mapping. The reads are divided into the number of chunks specified for this parameter.
wt.filter.mapping.np.per.node	8	The number of processors per node use for filter mapping.
wt.genomic.mapping.np.per.node	8	The number of processors per node use for genomic mapping.
wt.junction.mapping.np.per.node	8	The number of processors per node use for junction mapping.
Optional BAM generation parameters		
bamgen.refcor.addcs	1	Whether or not to add the color sequence to BAM records. If the XSQ file does not contain color space, this parameter is ignored, and the resulting BAM file output does not contain color space. See "Refcor" on page 367 for information about refcor. Allowed values: 0,1.
create.unmapped.bam.files	FALSE	If set to TRUE, creates an additional BAM output file containing unmapped reads. Allowed values: • **FALSE**: Do not create the unmapped reads output file. • **TRUE**: Create a BAM output file containing the unmapped reads.
bamgen.mqv.threshold	0	Provides control over the contents written to the output BAM file depending on the quality value of the alignment. To preserve only high quality alignments, set this value to a positive integer. Allowed values: Integers 0–255.
refcor.base.filter.qv	10	Bases with a quality value below the value of this parameter provided are replaced with 'N'. Allowed values: Integers 0–255.
refcor.reference.weight	8	Reference weight. This parameter is used during base translation. In the read reconstruction process, multiple signals are combined to generate the final base call. This parameter adds weight (in terms of Phred score) to the signals which are compatible with reference. Color combinations that result in a variant are considered compatible with reference. Additional weight helps to eliminate base errors caused by color error(s) during base translation. Allowed values: Integers 0–100.

Mapping performance

Mapping speed depends on hardware properties as well as on the scattering logic. Mapping runs are split into multiple jobs for processing efficiency. The following parameters affect how a mapping run is split:

- **`fragmap.number.of.nodes`** – Specifies the number of jobs that are created, only if a split is necessary. The size of each split job is approximately the total number of reads divided by the number of jobs.
- **`fragmap.max.number.of.jobs`** – Determines the maximum number of jobs that can be launched per analysis. Setting this parameter to a very large value can cause the scheduler to behave incorrectly.
- **`fragmap.minreads.per.node`** – Determines the threshold of beads beyond which splitting occurs.

- **`fragmap.maxreads.per.node`** – Determines the largest number of beads that can be mapped per node. The default is based on the minimum scratch space requirements. If you scratch space is smaller or larger, this number can be made smaller or larger respectively.
- **`processors.per.node`** – This is the number of cores available for mapping analysis on each node.
- **`mapping.memory`** – For human mapping and 25.2.0 scheme, there are 3 schema lines. Simultaneous handling of schema lines reduces I/O and therefore improves mapping speed.

Memory requirements for mapping jobs are the following:

- Reference: ~6 GB
- 1 schema line: ~13 GB
- 1 schema line and reference: ~19 GB
- 2 schema lines and reference: ~32 GB
- 3 schema lines and reference: ~45 GB

Below 19 GB, the reference is split up and I/O increases significantly. Pre-built hash tables are not used for memory less than 24 GB. LifeScope™ Software supports the reuse of hash tables for human genome reference files (hg18 and hg19) and the default mapping schemes 35.2, 25.2, and 20.1. Pre-calculated color-space are included in the reference directory available with LifeScope™ Software. If system RAM is 24 GB or higher, the hash tables are loaded into memory. Depending on the pipeline, this hash table reuse typically saves 2–3 hours in analysis processing time.

For genomes without pre-calculated hash tables in the reference repository, the mapping module creates hash tables but does not save them for reuse.

In general, set the `mapping.memory` parameter to be 1 GB less than the total system memory (RAM) available.

Mapping only scales when the number of reads that are mapped together is greater than 100 M. Splitting such that the total number of fragments per node is significantly less than 100 M is not recommended. The splitting can be explained using the following pseudo-code.

```
if (num.reads <= fragmap.minreads.per.node ) { num.jobs = 1}
reads.per.job = num.reads / fragmap.number.of.nodes
if (reads.per.job > fragmap.maxreads.per.node) {
    reads.per.job = fragmap.maxreads.per.node}
num.jobs = celing (num.reads / reads.per.job)
if (num.jobs > fragmap.max.number.of.jobs ) {
    throw exception; }
```

In above pseudo-code, the input read-sets (possibly from multiple XSQ files) have `num.reads` that fall in one read length category. `Num.jobs` is the total number of jobs launched for mapping analysis of `num.reads` beads.

In general, the mapping parameters allow mixing and matching single-anchor and multiple-anchor schemes, and also mixing and matching different seed lengths and different numbers of mismatches. For example, an unmapped scheme of 25.2.0, 35.2.0:15:25, 20.1.0, with a repetitive scheme of 35.2.0:30, 45.2.0, 60.2.0, is supported, but not recommended. The impact on performance can be significant.

WT mapping internal parameters

Table 102 describes the single-read algorithm parameters that we do not recommend changing.

Table 102 Single-read WTA internal parameters

Parameter	Value	Description
wt.frag.map.run	1	Enables fragment mapping. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
wtexonsequenceextractor.secondary.run	1	Enables exon extraction. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
wtsplicejunctionextractor.run	1	Enables splice junction extraction, which is required for junction mapping. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
Algorithm parameters		
mapping.use.iub.reference	FALSE	Whether or not to support reference sequences that contain IUB codes. Allowed values: • **FALSE**: Do not support IUB codes in the reference file. • **TRUE**: Mapping allows matching to either alleles of bi-allelic IUB codes in the reference file.
bamgen.primary.output.filter.type	primary_only	Filters the content written to the output BAM file depending on whether the alignment is gapped or not. Note that unmapped reads are considered primary. Allowed values: • **no_filtering**: Report all alignments. • **primary_only**: Report all primary alignments (both gapped and ungapped).
wt.filter.map.max.hits	10	The maximum number of alignments for a read, during filter mapping. Allowed values: Integers 1–100000.
wt.genomic.map.max.hits	10	The maximum number of alignments for a read, during genomic mapping. Allowed values: Integers 1–100000.
wt.junction.map.max.hits	10	The maximum number of alignments for a read, during junction mapping. Allowed values: Integers 1–100000.
junction.reference.file	—	The junction reference file. Must be a FA or FASTA file.
wt.mask.positions	—	A string of 0s and 1s that show positions to be masked or not. If blank (default), no masking is performed. There is no penalty in mapping towards positions that are masked.
wt.map.mismatch.penalty	-2.0	Specifies a negative scoring penalty for mismatch that is used in local alignment mode. Setting this value to the negative of a number greater than the read lengths has the effect of disallowing mismatches. Allowed values: Floats.
Mapping scheme parameters		

Table 102 Single-read WTA internal parameters *(continued)*

Parameter	Value	Description
map.scheme.unmapped.1	0.0.0	Mapping scheme used for reads with lengths less than 25.
map.scheme.unmapped.25	25.2.0	Mapping scheme used for reads with lengths 25–34.
map.scheme.unmapped.35	25.2.0:10	Mapping scheme used for reads with lengths 35–49.
map.scheme.unmapped.50	25.2.0:20	Mapping scheme used for reads with lengths 51–74.
map.scheme.unmapped.75	25.2.0:20	Mapping scheme used for reads with lengths of 75 and greater.
map.scheme.unmapped.variable	25.2.0:20	Mapping scheme used for trimmed reads.

Mapping schemes

The mapping scheme parameters, listed in Table 102, determine the behavior of the mapping module. The LifeScope™ Software mapping module supports two types of mapping, global mapping and seed-and-extend mapping.

- **Global mapping –** With this approach the seed length matches the full read length, and the seed anchors at the beginning of the read. For example, global mapping for 50-bp reads uses a scheme such as `50.6.0`. (In this example, 50, the seed length, matches the read length, and 0, the seed start position, starts at the beginning of the read.)
- **Seed-and-extend –** The initial alignment step that locates short matches between a read and the reference sequence. The seed specifies the length of the attempted match. This approach is also called local alignment.

Global mapping

To map a 50 bp read using global mapping, a typical scheme to use is 50.6.0. This allows for up to 6 mismatches. Because the seed length equals the length of the read, there is no extension.

Seed-and-extend

With the seed-and-extend approach, a mapping scheme parameter value such as `25.2.0` is interpreted as follows:

- **First value –** The seed length (`25`).
- **Second value –** The quantity of allowed mismatches in the seed (`2`).
- **Third value –** The start site of the seed within the read (`0`).

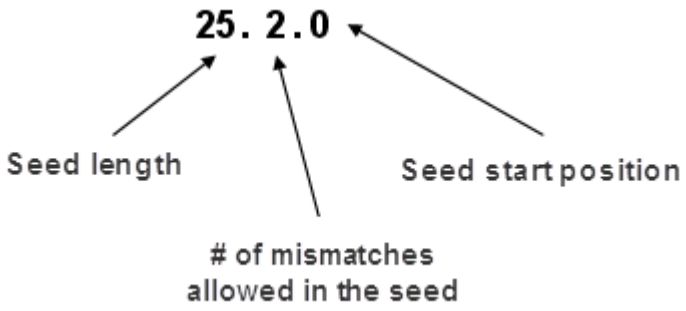


Figure 40 Seeds for local alignments

For example, with a `25.2.0` seed extend scheme, the mapping module examines the first 25 bases of a 50 bp read, and if that portion aligns to the reference with two or fewer mismatches, then the mapping module attempts seed extension. Seed extension begins at the anchor alignment and extends along the read, to find the best ungapped local alignment with a score *S*. The alignment length can be shorter than the read length.

For information on sliding periodic seeds, refer to the publication at this site:

http://bioinformatics.oxfordjournals.org/content/25/19/2514.full

Paired-end parameters

Table 103 describes the algorithm parameters that you can configure for paired-end whole transcriptome mapping.

Table 103 Paired-end WTA mapping parameters

Parameter	Default	Description
Mapping parameters		
enable.exon.mapping	1	Enables exon mapping.
enable.filter.mapping	1	Enables filter mapping.
enable.junction.mapping	1	Enables junction mapping.
wt.f3.rescue.enabled	true	Whether or not to attempt rescue of F3 reads.
wt.f5.rescue.enabled	true	Whether or not to attempt rescue of F5 reads.
std.insert.size	60	The standard deviation of the insert size. Allowed values: Integers 1–100000.
mapping.in.base	FALSE	Whether to map in base space (TRUE) or color space (FALSE). Set to TRUE if input data has base space available. Allowed values: • **FALSE**: Map in color space. • **TRUE**: Map in base space. If only color space is available, then the module fails when base space mapping is turned on.
wt.merge.min.junction.overhang	8	Alignments to junctions are not reported if they have fewer than this number of bases unaligned on either side of the splice. Allowed values: Integers 0–100.
Splice junction extractor parameters		
read.length	75	Read length. Allowed values: Integers 25–100.
Rescue parameters		
avg.insert.size	120	The average the insert size. Used to compute the rescue distance: rescueDistance = avgInsertSize + (3 * stdInsertSize) Allowed values: Integers 1–100000.
wt.f3.rescue.max.mismatches.allowed	8	The maximum number of mismatches allowed in a rescued F3 alignment. Allowed values: Integers 0–15.

Table 103 Paired-end WTA mapping parameters *(continued)*

Parameter	Default	Description
wt.f5.rescue.max.mismatches.allowed	6	The maximum number of mismatches allowed in a rescued F5 alignment. Allowed values: Integers 0–10.
Optional resource parameters		
java.heap.space	1500	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.
mapping.memory	15gb	The total memory, in gigabytes, that is available for map reads, per compute node. This number is set to 15 GB because the minimum hardware requirement for memory is 16 GB. **Note:** Typically, the job scheduler will allocate on nodes that have the requested memory available. For mixed hardware, it is suggested that different queues are created based on the memory available. Mapping jobs should be launched on high memory systems. For human references (hg18, hg19) and a 25.2 scheme, 45 GB is recommended for fastest performance. The smallest recommended RAM for human reference is 19 GB.
Optional parallelization parameters		
fragmap.number.of.nodes	4	The number of compute nodes available for this analysis.
fragmap.max.number.of.jobs	100	The maximum number of nodes allowed for mapping.
fragmap.minreads.per.node	4000000	If the total number of fragments exceeds this amount, then the analysis is distributed across the available nodes. Allowed values: Integers 1–125000000 (125 Million).
fragmap.maxreads.per.node	150 M	The maximum number of fragments that can be processed on a single node. Allowed values: Integers 1–150000000 (150 Million).
processors.per.node	8	The number of processors per node use for mapping. The reads are divided into the number of chunks specified for this parameter.

Table 104 describes WTA paired-end algorithm parameters that we do not recommend changing.

Table 104 Paired-end WTA internal parameters

Parameter	Value	Description
wt.pe.map.run	1	Enables paired-end mapping. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
Splice junction parameters		
wt.f3.splext.output.reference	—	The name of the reference file output by the splice junction extractor stage.
wt.gtf.file	—	The GTF file used to create the genemodel. The default is ${annotation.gtf.file}.
Splice junction extractor parameters		

Table 104 Paired-end WTA internal parameters *(continued)*

Parameter	Value	Description
wtspljunctionextractor.run	1	Enables splice junction extraction, which is required for junction mapping. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
Exon extractor parameters		
wtexonsequenceextractor.secondary.run	1	Enables exon extraction, which is required for F5 pairing. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
wt.f5.exseqext.output.reference		The name of the reference file output by the exon extractor stage.
Algorithm parameters		
mapping.use.iub.reference	FALSE	Whether or not to support reference sequences that contain IUB codes. Allowed values: • **FALSE**: Do not support IUB codes in the reference file. • **TRUE**: Mapping allows matching to either alleles of bi-allelic IUB codes in the reference file.
wt.map.mismatch.penalty	-2.0	Specifies a negative scoring penalty for mismatch that is used in local alignment mode. Setting this value to the negative of a number greater than the read lengths has the effect of disallowing mismatches. Allowed values: Floats.
wt.f3.filter.map.max.hits	10	The maximum number of alignments for an F3 read, during filter mapping. Allowed values: Integers 1–100000.
wt.f3.genome.map.max.hits	10	The maximum number of alignments for an F3 read, during genomic mapping. Allowed values: Integers 1–100000.
wt.f3.junction.map.max.hits	10	The maximum number of alignments for an F3 read, during junction mapping. Allowed values: Integers 1–100000.
wt.f3.map.mask.positions	—	A string of 0s and 1s that show F3 positions to be masked or not. If blank (default), no masking is performed. There is no penalty in mapping towards positions that are masked.
wt.f3.mapping.schema.file	—	Mapping schema file for F3 reads.
wt.f5.filter.map.max.hits	10	The maximum number of alignments for an F5 read, during filter mapping. Allowed values: Integers 1–100000.
wt.f5.genome.map.max.hits	10	The maximum number of alignments for an F5 read, during genomic mapping. Allowed values: Integers 1–100000.
wt.f5.exon.map.max.hits	10	The maximum number of alignments for an F5 read, during exon mapping. Allowed values: Integers 1–100000.

Table 104 Paired-end WTA internal parameters *(continued)*

Parameter	Value	Description
wt.f5.map.mask.positions	—	A string of 0s and 1s that show F5 positions to be masked or not. If blank (default), no masking is performed. There is no penalty in mapping towards positions that are masked.
wt.f5.mapping.schema.file	—	Mapping schema file for F5 reads.
exon.reference.file	exons.fasta	Path to the exon reference file. Must be a FA or FASTA file.
junction.reference.file	—	Path to the junction reference file. Must be a FA or FASTA file.
Pairing parameters		
pair.uniqueness.threshold	10.0	This value defines the clear zone. This value is considered only when pairing.output.uniqueness.type is set to -1. If the score of the best alignment is this threshold more than the score of the second best alignment, the best alignment is considered uniquely placed. A threshold of zero means that this uniqueness is strictly adhered to, regardless of the alignment's relative scores. Allowed values: Floats 0.0–100.0.
pairing.anchor.max.mismatches	4	Total mismatches for both reads of a pair. Allowed values: Integers 0–100.
pairing.gapped.max.hits	100:0	For gapped and ungapped alignment determination during rescue, this value determines the maximum number of rescues attempted for each read. If a read has more than this number of hits, rescue is not performed on this bead. The first value is for ungapped rescue, and second, gapped.
pairing.mismatch.penalty	-2.0	A single penalty applied to alignments, to compare the significance between alignments. Helps evaluate the alignments. Allowed values: Floats –100 to 0.
pairing.output.uniqueness.type	–1	Whether to output non-unique hits. Allowed values: • **-1**: A clear zone determines uniqueness. One or two best pairs are output, depending on whether it is unique. • **0**: All good pairs are output. • **1**: Only unique good pairs are output. Allowed values: Integers –1, 1.
Rescue parameters		
wt.rescue.input.generation.exon.fuzzy.border.width	10	The amount of slack allowed around exon borders, for doing rescue with fuzzy exon borders. Allowed values: Integers 0–2147483647.
wt.rescue.input.generation.min.alignment.distance.for.rescue	100000	When rescuing reads that have already been mapped to the reference, this value is the minimum distance between the anchor alignment and any of the alignments of the rescued read downstream of the anchor alignment. Allowed values: Integers 0–2147483647.

Table 104 Paired-end WTA internal parameters *(continued)*

Parameter	Value	Description
wt.rescue.input.generation.rescue.anchor.alignments.not.overlapping.exons	1	Enables the additional rescue downstream and upstream of alignments that do not overlap exons but do overlap genes. Allowed values: • **0**: No change in behavior. • **1**: Enables the rescue described above.
wt.rescue.input.generation.rescue.fuzzy.exon.borders	1	Enables allowing some slack around exons borders. Allowed values: • **0**: No change in behavior. • **1**: Enables slack around exons borders.
wt.rescue.input.generation.rescue.only.for.the.best.anchor.alignment	0	Limits rescue to only for the best anchor alignment. Allowed values: • **0**: Rescue for all anchor alignments. • **1**: Rescue only for the best anchor alignment.
wt.rescue.input.generation.rescue.only.unaligned.reads	0	Rescue ignores target reads that already have an alignment. Allowed values: • **0**: Rescue all the target reads. • **1**: Rescue ignores target reads that already have an alignment.
wt.rescue.input.generation.rescue.only.within.rescue.distance	1	Rescue only within rescue distance. Rescue only within the rescue region starting from the exon border, to a limit inside the exon. The limit is computed based on the insert size and the location of the anchor alignment inside the previous exon. Allowed values: • **0**: Rescue on the entire exon. • **1**: Rescue only within rescue distance.
wt.rescue.valid.adjacent.mismatches.count.as.one	0	Two adjacent mismatches that can account for a SNP count only as one mismatch. • **0**: No change in behavior. • **1**: Two adjacent mismatches that can account for a SNP count only as one mismatch.
wt.rescue.input.generation.rescue.short.range	1	Limits rescue to all the target reads in the vicinity of the anchor read. Allowed values: • **0**: No change in behavior. • **1**: Limits rescue to all the target reads in the vicinity of the anchor read.
wt.rescue.input.generation.zLimit	100	The Z limit. Allowed values: Integers 0–2147483647.
wt.rescue.standalone.rescue.executable	rescue	The name of the executable file used for rescue.
Mapping scheme parameters		
wt.f3.mapping.scheme.unmapped.1	0.0.0	Mapping scheme for F3 reads with lengths less than 25.

Table 104 Paired-end WTA internal parameters *(continued)*

Parameter	Value	Description
wt.f3.mapping.scheme.unmapped.25	25.2.0	Mapping scheme for F3 reads with lengths 25–34.
wt.f3.mapping.scheme.unmapped.35	25.2.0:10	Mapping scheme for F3 reads with lengths 35–49.
wt.f3.mapping.scheme.unmapped.50	25.2.0:20	Mapping scheme for F3 reads with lengths 51–74.
wt.f3.mapping.scheme.unmapped.75	25.2.0:20	Mapping scheme or F3 reads with lengths of 75 and greater.
wt.f3.mapping.scheme.unmapped.variable	25.2.0:20	Mapping scheme for F3 reads with variable lengths (trimmed reads).
wt.f5.mapping.scheme.unmapped.1	0.0.0	Mapping scheme for F5 reads with lengths less than 25.
wt.f5.mapping.scheme.unmapped.25	25.2.0	Mapping scheme for F5 reads with lengths 25–34.
wt.f5.mapping.scheme.unmapped.35	25.2.0:10	Mapping scheme for F5 reads with lengths 35–49.
wt.f5.mapping.scheme.unmapped.50	25.2.0:20	Mapping scheme for F5 reads with lengths 51–74.
wt.f5.mapping.scheme.unmapped.75	25.2.0:20	Mapping scheme for F5 reads with lengths of 75 and greater.
wt.f5.mapping.scheme.unmapped.variable	25.2.0:20	Mapping scheme for F5 reads with variable lengths (trimmed reads).
wt.f5.exon.mapping.scheme.unmapped.1	0.0.0	Mapping scheme for F5 reads with lengths less than 25.
wt.f5.exon.mapping.scheme.unmapped.25	25.3.0	Mapping scheme for F5 reads with lengths 25–34.
wt.f5.exon.mapping.scheme.unmapped.35	25.3.0:10	Mapping scheme for F5 reads with lengths 35–49.
wt.f5.exon.mapping.scheme.unmapped.50	25.3.0:20	Mapping scheme for F5 reads with lengths 51–74.
wt.f5.exon.mapping.scheme.unmapped.75	25.3.0:20	Mapping scheme for F5 reads with lengths of 75 and greater.
wt.f5.exon.mapping.scheme.unmapped. variable	25.3.0:20	Mapping scheme for F5 reads with variable lengths (trimmed reads).

Mapping output files

Overview

The mapping module generates a BAM file containing alignments in coordinate order. For information about the BAM file, see Appendix C, "File Format Descriptions" on page 493.

The number of BAM files generated depends on the input XSQ files. One BAM file is generated per read-set. If the input data is an XSQ file containing 96 barcodes, then the mapping module generates at least 96 output BAM files. Beads that are unclassified in any barcode are output into a separate additional BAM file. If `create.unmapped.bam.files` is set to `TRUE`, then an additional 96 output BAM files corresponding to the unmapped reads are also generated.

The filenames for the output BAM files are created using information from the input XSQ file, including: file base name, file id, index name, and index id. The BAM files are named according to the following patterns:

- Non-indexed BAM files: *xsqname-fileID*-1.bam
- Indexed BAM files: *xsqname-fileID*-idx_*bcIndex-bcID*.bam

The fields in the filenames are:

- **xsqname** – The XSQ file base name, without the .xsq extension.
- **fileID** – The file ID for internal XSQ file tracking.
- **bcIndex** – The barcode index.
- **bcID** – The barcode identifier for internal barcode tracking.

The directory structure for mapping output is as follows:

```
outputs/${analysis.output.dir}/
      <mappingIniFileBasename>/
            <sampleName1>/*.bam
            <sampleName2>/*.bam
```

The directory name <*mappingIniFileBasename*> is the basename of the mapping INI file. The defaults are `wt.frag.map` and `wt.pe.map`.

The string <*sampleName**> is determined by the sample description for the particular read-set.

The mapping output files are used by the BAMStats mapping statistics module and by the whole transcriptome tertiary analysis modules: WT counts, WT coverage, and WT splice junctions.

BAM file differences

This section describes two types of differences seen in WT BAM files, compared to other LifeScope™ Software BAM files.

Single-read optional fields

A BAM file produced by the WT paired-end pipeline is identical to that produced by the resequencing pipeline. However, the BAM format from the WT single-read pipeline differs from BAM files produced elsewhere in LifeScope™ Software. The single-read pipeline produces separate BAM files for filtered, unmapped, and mapped reads.

Note: The WTA single-read pipeline produces a BAM file that uses the optional fields described in Table 105.

Table 105 WT single-read pipeline BAM file optional fields

Optional field	Description
IH:i:	Number of stored alignments containing the current query.
HI:i:	Query hit index.
NH:i:	Number of reported alignments containing the current query.
CS:z:	Color read sequence.
CQ:z:	Color quality sequence.
CC:z:	Reference name of the next hit.
CP:z:	Coordinate of the next hit.
AS:i:	locationAlignment Score generated by the aligner.
XN:i:	Alignment score of the best non-primary alignment for query in the current record.
XF:z:	T for true or F for false. Set to "T" if this read is filtered.

Table 105 WT single-read pipeline BAM file optional fields *(continued)*

Optional field	Description
XJ:z:	"K" for Known Junction or "P" for Putative Junction.

Clipped records

By design, BAM records produced in WT modules may have soft-clips in the CIGAR string. For records that have soft-clips at the beginning, the correct start position of reported sequence is interpreted as the start position minus the soft-clipped length. However, BAM-consuming external tools that are not BAM-specification complaint may present soft-clipped records as shifted or missed alignments.

WT filtering stats

An example whole transcriptome human filter reference is provided under the `<examplesdir>/demos/WholeTranscriptome/references` folder in the optional LifeScope™ Software examples distribution. This example reference fasta includes contigs from barcode primers, human ribosomal RNAs, tRNAs, and other known targets for filtering. The filter reference may be expanded or removed with more csfasta reference records at the discretion of the user.

The stats for filtered reads are generated after a LifeScope™ Software run. First two lines of the stats file states the number of reads processed, and the number of reads mapped to the filtered reference followed by its percentage. Each following line reports a contig name and the count of reads that aligned to that contig. Paired-end filtering stats may be found in the Intermediate folder. The following is a truncated example of filtering stats output:

```
countOfReadsProcessed: 175,839,941
countOfReadsMapping:    18,783,220 (10.7%)
…
Barcode-047-3-end reverse   83
Barcode-048-3-end reverse   2,350
gi|124517659|ref|NR_003286.1| Homo sapiens 18S ribosomal RNA
(LOC100008588)   5,175,899
gi|142372596|ref|NR_003285.2| Homo sapiens 5.8S ribosomal RNA
(LOC100008587)   78,936
gi|124517661|ref|NR_003287.1| Homo sapiens 28S ribosomal RNA
(LOC100008589)   7,616,212
chr6.trna95-AlaAGC  (58249908-58249836)  Ala (AGC) 73 bp  Sc:
42.26   5
chr6.trna25-AlaAGC  (26859897-26859969)  Ala (AGC) 73 bp  Sc:
46.89   2
chr6.trna94-AlaAGC  (58250620-58250548)  Ala (AGC) 73 bp  Sc:
54.62   4
chr6.trna160-AlaAGC (26881822-26881750)  Ala (AGC) 73 bp  Sc:
54.69   35
```

Mapping statistics

Mapping statistics occur after mapping as an optional post-processing step named BAMStats. BAMStats accepts the output of the mapping step and generates statistics files to provide an in-depth understanding of the experimental data and to better detect the presence of anomalies. LifeScope™ Software shell users can display the mapping statistics output data as a chart with a spreadsheet program or other third-party program. A subset of the output from mapping can be visualized in the LifeScope™ Software UI as a series of line and bar charts, if the analysis is run in the projects repository.

Note: BAMStats does not accept 1.x BAM files as input. 1.3 BAM files must be converted to 2.0 BAMs before being used with BAMStats.

Mapping statistics parameters

Table 106 lists the parameters which control the BAMStats output.

Table 106 BAMStats parameter description

Parameter name	Default value	Description
Mandatory parameters		
Optional parameters		
bamstats.maximum.coverage	10000	Defines the maximum coverage allowed for locations in the reference. Locations with coverage more than the maximum coverage value are ignored during coverage calculations. Allowed values: Integers >= 1.
bamstats.maximum.isize	100000	The maximum inset size for LMP and PE libraries. Reads with an insert size more then the specified value are ignored for Insert Range Report calculation. Allowed values: Integers 1–100000.
bamstats.wig.primary.only	1	Use only primary alignments for coverage in WIG file format. Allowed values: • **0**: Do not restrict coverage in WIG file format to only primary alignments. • **1**: Restrict coverage in WIG file format to only primary alignments.
bamstats.wig.binsize	100	The bin size for coverage in WIG file format. Allowed values: Integers 1–100000.
bamstats.bin.isize	100	The bin size for insert range distribution. Allowed values: Integers 1–100000.
bamstats.wig.combined.report.both.strands	0	Whether to generate a WIG format coverage file with combined data from both the strands. Allowed values: • **0**: Do not generate wig files. • **1**: Generate the combined coverage file.
Optional resource parameters		
wall.time	120	Total time for the process to complete.

Table 106 BAMStats parameter description *(continued)*

Parameter name	Default value	Description
java.heap.space	13000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.
number.of.nodes	4	The number of compute nodes available for this analysis.
memory.request	14gb	Memory request. Include the units gb in the setting.

Table 107 lists BAMStats parameters that we do not recommend changing.

Table 107 BAMStats internal parameter description

Parameter name	Default value	Description
bamstats.run	1	Whether or not to run the BAMStats module. Allowed values: • **0**: Do not run the BAMStats module. BAMStats statistics are not generated. • **1**: Run the BAMStats module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
bamstats.group.stats	1	Enable combined statistics covering all read-sets in the group or sample. Allowed values: • **0**: Only statistics for BAM files are generated. • **1**: Also generate combined statistics for the group or sample.
bamstats.maximum.mismatches	100	The maximum mismatches allowed in the alignments. Any alignment with more then the specified number of mismatches is ignored while generating reports related to number of mismatches. Allowed values: Integers 0–100.
bamstats.maximum.baseqv	100	Max base quality values. Any base with base quality value more then the specified value is ignored while generating reports. Allowed values: Integers 0–100.
bamstats.maximum.mappingqv	255	Maximum mapping quality value. Any alignment with mapping quality value more than the specified value is ignored. Allowed values: Integers 0–255.
bamstats.wig.minimum.mappingqv	2	Defines the minimum mapping quality allowed for coverage in WIG format. Any alignment with a mapping quality value less than this value is ignored. Allowed values: Integers 0–100.
bamstats.enable.probe.position	0	Enable probe and position error reports. Allowed values: • **0**: Do not generate probe and position error reports. Off is recommended for WT. • **1**: Also generate probe and position error reports.

Summary of mapping statistics output

File formats

For every input BAM file, a set of statistics files are generated. These files are in CHT, CSV, TXT, and WIG formats. Each CHT file corresponds to one displayed chart. A CHT file specifies the type of chart, the displayed range of each axis, and the data points, without using external references.

The CHT file format is an internal file format based on the CSV file format, with addition header information included. CHT header information is the following:

```
# name:
# type: scatter2d | pie | vbar | line
# title:
# xaxisname:
# yaxisname:
# xrange: <min>:<tickinterval>:<max>
# yrange: <min>:<tickinterval>:<max>
XAXISNAME, SERIES1NAME, SERIES2NAME, ...
x1, y1.1, y1.2, ...
x2, y2.1, y2.2, ...
x3, y3.1, y3.2, ...
```

The wiggle format (.wig) is a public format typically used for coverage. Visit their site for more information:

hgdownload.cse.ucsc.edu/goldenPath/help/wiggle.html

A genome browser such as the Integrative Genomics Viewer (IGV) can be used to visualize the coverage. For information is available from their site:

www.broadinstitute.org/igv/

For a collection of input BAM files that belong to a sample, a set of cumulative statistics files are generated. The cumulative statistics files are also in CHT, CSV, TXT, and WIG formats. The cumulative statistics can be visualized in the LifeScope™ Software UI.

Summary information

A comma-separated summary file (*BAMfilename*-summary.tbl) is generated that summarizes key mapping quality statistics per input BAM file. This file contains one row for each input BAM file in the sample. The summary file is displayed in the LifeScope™ Software UI. See "Summary file" on page 390.

Directory structure

The output of the BAMStats module has the following directory structure:

```
bamstats/
      <sampleName>/*.cht, *.tbl
      <sampleName>/<bam>/.*cht
      <sampleName>/<bam>/Misc/*.csv, *.txt, *.wig
      <sampleName>/Misc/*.csv, *.txt
```

The position error files and probe errors files are created in the BAM file directories.

Overview

The summary report contains a snapshot of statistics in all BAM files present in this sample. This report is displayed in the LifeScope™ Software UI.

```
<sampleName1>/BAMfilename-summary.tbl
```

Following directories and reports contain the cumulative statistics from all BAM files that belong to this sample. The `Misc` folder is not displayed in UI.

```
<sampleName1>/*.cht
```

```
<sampleName1>/Misc/*.csv, *.wig, *.txt
```

Following statistics are generated per BAM file in the mapping directory. These reports are not displayed in the UI.

```
<sampleName1>/<BAMfilename>/.*cht
<sampleName1>/<BAMfilename>/Misc/*.csv, *.txt, *.wig
```

In a sample, some of the BAM files possibly represent unhealthy DNA or RNA, causing the cumulative statistics to look poor. The summary tbl file is the unified location for examining the quality of data of all BAM files in the sample.

If the data for a particular BAM file is not as expected, look at the directory-level reports for that BAM file, for details.

Mapping statistics output files

This section describes the mapping statistics files generated by the BAMStats module. When statistics reports are separated by tag type, the report's file name includes the tag in the file name. The tags used in file names are:

Table 108 Tags in mapping statistics output file names

Library type	Tag
Fragment	F3
Mate-pair	F3
	R3
Paired-end	F3
	F5-P2

The output files generated by BAMStats include the name of the BAM file. The file name pattern is:

BAMFileName-fileID-barcodeID.StatisticsReportName.tag.extension

This string is referred to by *prefix* in Table 109, the mapping statistics output table. In the description of mapping statistics output file names in Table 109, the string *tag* is used to refer to the tags in Table 108.

Table 109 Mapping statistics output files

Report	Description, axis information, filename
Alignment Length Distribution	
	A bar plot giving the distribution of alignment lengths found in various tags used in alignment. The report is separated by tag type to provide visibility into the accuracy of individual tags. Only the primary alignment for each bead is considered in calculating the distribution.
	Y axis: Frequency X axis: Alignment length (from 0 to the maximum read length)
	Output file name: *prefix*.Alignment.Length.Distribution.*tag*.cht
Base Mismatch Distribution	

Table 109 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
	A bar plot giving the distribution of total number of mismatches found in various tags used in alignment. The bins are summed over all alignment lengths. Only the primary alignment for each bead is considered in calculating the distribution.
	Y axis: Frequency X axis: Alignment length (from 0 to the maximum mismatches allowed)
	Output file name: *prefix*.Mismatch.Distribution.*tag*.cht
Distribution of Alignment Length and Number of Mismatches in Tags	
	A report providing a simultaneous picture of the alignment length and number of mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. This report is located in the `Misc` folder, and is not displayed in the UI.
	Z axis: Frequency Y axis: Alignment length (from 0 to the maximum mismatches allowed) X axis: Number of mismatches (from 0 to the maximum mismatches allowed)
	Output file name: *prefix*.AlignmentLength.Mismatch.*tag*.csv
Distribution of Mismatches by Position	
	A bar plot providing a distribution of errors (mismatches to reference) by position within the read. Only the primary alignments for each bead are used to generate this distribution.
	Y axis: Frequency X axis: Position (from 0 to maximum read length)
	Output file name: *prefix*.Mismatches.By.Position.*tag*.cht
Color Mismatch Distribution	
	A bar plot providing a distribution of the total number of color mismatches in various tags used in alignment. Only the primary alignments for each bead are used to generate this distribution. This report is only generated if mapping is done in color space.
	Y axis: Frequency X axis: Number of mismatches (from 0 to the maximum mismatches allowed)
	Output file name: *prefix*.ColorMismatches.Distribution.*tag*.cht
Distribution of Alignment Length and Color Mismatches	
	A report providing a simultaneous picture of the alignment length and number of color mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. This report is only generated if mapping is done in color space. **Note:** This report is located in the `Misc` folder, and is not displayed in the UI.
	Y axis: Alignment length (from 0 to the max read length) X axis: Number of mismatches (from 0 to the maximum mismatches allowed) Z axis: Frequency
	Output file name: *prefix*.AlignmentLength.ColorMismatches.*tag*.csv
Base QV Distribution	

Table 109 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
	A bar plot providing a distribution of base quality values generated using the reference assisted error correction/base conversion algorithm. The base QV distributions are separated for each tag as quality of individual tags could be very different.
	Y axis: Frequency X axis: Base QV (from 0 to the maximum QV)
	Output file name: *prefix*.BaseQV.*tag*.cht
Base QVs by Position	
	A report providing a distribution of base quality values by individual base positions. This report identifies if certain base positions (particularly towards the end of the read) have poor base quality values. All the tags (F3/R3/F5-P2) used in the alignment are combined into a single bin.
	Y axis: Base QV X axis: Base position (from 0 to read length)
	Output file name: *prefix*.BaseQV.by.Position.csv
Distribution of Mismatches by Base QV	
	A line plot providing a distribution of errors (mismatches to reference) by base quality values bins.
	Y axis: Percentage of total mismatches X axis: Base QV (from 0 to maximum QV) This report gives, for each QV, a measure of the raw mismatches at positions with that QV. This information does not measure whether the base QVs generated are well calibrated to the probability of error in that particular base position.
	Output file name: *prefix*.Mismatches.By.BaseQV.*tag*.cht
Distribution by Mapping QVs by Tag	
	A bar plot providing a distribution of mapping quality values for individual tags (F3/R3/F5-P2). Only the primary alignment for each bead is used in calculating this distribution.
	Y axis: Frequency X axis: Position (from 0 to maximum mapping QV)
	Output file name: *prefix*.MappingQV.*tag*.cht
Distribution by Pairing QVs	
	A bar plot providing a distribution of pairing quality values for individual tags (F3/R3/F5-P2). Only the primary alignment for each bead is used in calculating this distribution.
	Y axis: Frequency X axis: Pairing QV (from 0 to maximum pairing QV)
	Output file name: *prefix*.PairingQVs.*tag*.cht
Coverage Report	
	A line plot providing a distribution of coverage obtained after mapping/pairing. Only the primary alignment for each bead is used in this calculation.
	Y axis: Number of bases X axis: Coverage (from 0 to number of reads)
	Output file name: *prefix*.Coverage.*tag*.cht
Coverage Report by Chromosome (Contig) and Base Windows	

Table 109 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
	A line plot providing a distribution of coverage within each reference window. The coverage is calculated within each window along a reference chromosome. The window size can theoretically be anywhere from a single base to the contig length. However the calculations of base level coverage are computationally expensive and less interpretable as the window size increases. Only the primary alignment for each bead is used in this calculation.
	Y axis: Coverage (from 0 to number of reads) X axis: Contig and window number
	Output file name: *prefix*.Coverage.By.Chromosome.contig*nn.tag*.cht
Coverage by Strand	
	A line plot providing the distribution of coverage within each reference window separated by reference strand (+/-). Only the primary alignment for each bead is used in this calculation.
	Y axis: Number of bases X axis: Coverage (from 0 to number of reads)
	Output file name: *prefix*.Coverage.By.Strand.*tag*.cht
Coverage files	
	Coverage reports in wiggle format. For each genome position, reports the number of reads that cover (map to or span) the position. Because reporting coverage for each position results in very large files, coverage is reported for bins, with each bin spanning a user-defined number of bases. For each bin, the mean coverage of all the positions in that bin is reported. The parameter `bamstats.wig.binsize` controls the size of the bins in this file. Each coverage file includes a header as the first line. The header lines follow this pattern: track type=wiggle_0 name=<*chrname*> description=<*coverage from positive/negative/both strand*> visibility=full color=0,0,255 fixedStep chrom=<*chrname*> start=<*startpos*> step=<*binsize*> span=<*binsize*>
	Output file name: coverage_chr*nn*.POS.wig, coverage_chr*nn*.NEG.wig
Insert Range Distribution	
	A line plot providing the distribution of insert sizes for paired data (PE and LMP library types).
	Y axis: Frequency X axis: Insert Range bins
	Output file name: *prefix*.Insert.Range.Distribution.cht
Distribution of Read Pair Types	
	For paired data, this report calculates the distribution of read pair types (AAA, AAB, C**, etc.).
	Y axis: Frequency X axis: Read pair type
	Output file name: *prefix*.ReadPair.Type.cht
Pairing Statistics	

Table 109 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
	A pie chart showing the percentages of the following F3, R3 (or F5) combinations, for paired data,: • Mapped, Mapped • Mapped, Unmapped • Unmapped, Mapped • Unmapped, Unmapped • Mapped, Missing • Missing, Mapped • Unmapped, Missing • Missing, Unmapped
	Output file name: *prefix*.Pairing.Stats.cht
Unique Start Position	
	This report lists the following: • The total number of unique start positions. • The average number of unique start positions per reference position. See also "Unique Start Position" on page 391.
	Output file name: *BAMfilename*-summary.txt, *Samplename*.-summary.txt
Summary Report	
	The summary file provides statistics for each BAM file in the sample. See "Summary file".
	Output file name: *filename*.-summary.tbl

Mapping statistics example output

This section provides example output of some mapping statistics output files.

Summary file

The summary file provides statistics for each BAM file in the sample. This list describes labels used in the summary file:

- **NumFragmentsTotal** – The total number of beads, before any filtering on the instrument. This value is also the fragment count in the input XSQ file.
- **NumFragmentsPassingFilters** – The number of beads that pass filtering on the instrument.
- **NumMapped** – The number of reads with primary alignment.
- **% filtered mapped** – The number of primary reads divided by the number of beads passing instrument filtering (`NumMapped` / `NumFragmentsPassingFilters`).
- **% total mapped** – The number of primary reads divided by the number of total beads (`NumMapped` / `NumFragmentsTotal`).
- **Megabases of coverage** – The sum of alignment lengths, for primary alignments only.

The number of beads filtered out in the instrument is found by subtracting the number of beads that pass filtering from the total number of beads:

```
NumFragmentsTotal – NumFragmentsPassingFilters
```

The following are column headings for a summary file. The `PredictedInsertSize` and `Tag2` fields only are present with paired data.

```
BamFileName,IsColorInBam,IsBaseInXSQ,IsECC,LibraryType,
ReadLength,Megabases of Coverage,PredictedInsertSize,
NumFragmentsPassingFilters,NumFragmentsTotal,
Tag1-NumMapped,
Tag1- % total Mapped,
Tag1- % filtered mapped,
Tag2-NumMapped,
Tag2- % total Mapped,
Tag2- % filtered mapped,
(Tag1-AlignmentLength;Min;Max;Avg;Median;StdDev),
(Tag1-NumBaseMismatches;Min;Max;Avg;Median;StdDev),
(Tag1-MappingQV;Min;Max;Avg;Median;StdDev),
(Tag1-BaseQV;Min;Max;Avg;Median;StdDev),
(Tag1-NumColorMismatches;Min;Max;Avg;Median;StdDev),
(Tag2-AlignmentLength;Min;Max;Avg;Median;StdDev),
(Tag2-NumBaseMismatches;Min;Max;Avg;Median;StdDev),
(Tag2-MappingQV;Min;Max;Avg;Median;StdDev),
(Tag2-BaseQV;Min;Max;Avg;Median;StdDev),
(Tag2-NumColorMismatches;Min;Max;Avg;Median;StdDev)
(Coverage;Min;Max;Avg;Median;StdDev)
```

Unique Start Position

The following is an example of a Unique Start Position report, for fragment data:

```
Unique Start Positions F3
Total Number of Unique Start Positions: 16536 (16.043154 %
of primary alignments)
Average Number of Unique Start Positions per Reference
Position: 0.001630
```

Unique Start Position reports have the following naming patterns:

- *BAMfilename*-summary.txt
- *Samplename*-summary.txt

Coverage files

Example contents for the file `coverage_chr1_positive.wig` are:

```
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1,
alignmentFilteringMode=PRIMARY, filterOrpha-nedMates=false,
track name="BAM Coverage positive strand" description="BAM
Coverage positive strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
336171
336181
```

```
336191
336201
336211
336221
336231
336241
336251
336261
336271
…
```

Example contents for the file `coverage_chr1_negative.wig` are:

```
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1,
alignmentFilteringMode=PRIMARY, filterOrpha-nedMates=false,
track name="BAM Coverage negative strand" description="BAM
Coverage negative strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
57432
57442
57452
57463
57473
57483
57493
57503
57513
57523
57533
…
```

Run BAMStats standalone

This section describes requirements to generate mapping statistics outside of the context of the mapping module. These requirements are:

- **Directory structure –** The input directory for BAMStats module must mimic the output directory structure of mapping:

  ```
  Input.BAMStat.dir/
      <sampleName1>/*.bam, *.bai
      <sampleName2>/*.bam, *.bai
  ```

- **Index files –** For every BAM file, there must be a corresponding BAI-formatted indexed file. BAI files can be generated using the following command:

  ```
  samtools index <bamfile>
  ```

This command generates index sorted alignment for fast random access. The index file <*bamfile*>.bai is created. For details, please refer to the samtools site:

http://samtools.sourceforge.net/samtools.shtml

- **Write permission –** The user must also have write permission for the input directory, because the BAMStats module generates position and probe error files in that directory.

FAQ – Whole transcriptome

1

Why is SNPs analysis not included in the whole transcriptome pipeline?

A SNPs analysis is not offered for WTA pipelines. The reverse transcription process that is part of the WT library prep intrinsically adds a percentage of sequencing errors. In addition, alternative splicing and other alignment complexities around exon junctions may cause local misalignments, which can lead to false positive SNP calls. Using that data in a SNPs analysis may cause a high number of false positives.

2

What reads scenario achieves the best accuracy?

The best performance is achieved by considering 2BE and 4BE reads, their alignment, and reference. Typically, base-space reads generated by ECC have lower mapping throughput. Base-space reads generated from combined 2BE and 4BE calls and no reference have lower accuracy, compared to ECC calls.

3

In the mapping statistics genome coverage calculation report, how are Ns in the reference counted?

Ns in the reference are counted as missing coverage. For example, in the human genome, about 7% of the reference sequence consists of Ns. This means that the coverage calculation will never be reported as higher than 93%. If the frequency distribution says 7.49% of the genome is uncovered, this means that about 0.34% of the non-N reference sequence does not have reads mapped to it, and that also the 7.15% of the genome that consists of N does not have reads mapped to it.

4

What should I do to ensure balanced allele ratios at heterozygous positions?

The base-translation algorithm, refcor, uses instrument color-space data (ECC or non-ECC) and the reference sequence to produce the most likely base calls, along with a Phred-scale quality value. The degree to which the reference influences the base call is controlled by the `refcor.reference.weight` parameter. A position is converted to reference only if the `refcor.reference.weight` setting is higher than the difference between the quality values of a correct color call and of an adjacent erroneous color call that supports reference.

The `refcor.reference.weight` default setting, 8, is chosen to be low enough to ensure balanced allele ratios, and also be large enough both to distinguish correct from incorrect color calls and to maximize base accuracy.

In the event an incorrect reference base call is made, the disagreement between evidence sources causes that call's QV to be low. Incorrect calls can be filtered out based on QV threshold, and this filtering improves allele ratios.

Another way of filtering low quality base-calls is to adjust the parameter `refcor.base.filter.qv`, that converts all base calls with a quality value less than `refcor.base.filter.qv` into Ns. This filtering has the effect of setting QV to zero in the calls with low confidence. To decrease the number of N base calls, reduce this threshold (for example, from the default of 10 to 5).

Note: Setting the `refcor.reference.weight` parameter to zero completely eliminates the reference as a guide for base-translation (completely eliminates reference bias), but also considerably reduces the quality of base-translation.

5

How is the uniqueness of a pair determined?

A pair of reads is unique when there is exactly one good AAA pair. With the advent of local alignment, the likelihood of finding only one good pair is decreased. As a result, a different heuristic is used to determine “uniqueness”.

Consider an alignment of length *L* with *M* mismatches. If the local score is defined as *L+(m-1)M*, where *m<0* is the mismatch penalty, then the score of each good pair candidate is the sum of its two constituent local scores.

A pair of tags is considered unique if it has only one good pair, or if the score of the best pair is at least X greater than the score of the second best pair. X is specified by the `pair.uniqueness.threshold` key, and has a default value of 10.0. See Table 104 on page 376 and Table 103 on page 375 for a description of this and all pairing parameters.

6

What do the 3-letter pair classifications mean?

Table 110 lists a summary of genomic code classifications for mate-pair pairing runs. See Table 111 on page 396 for a summary of genomic code classifications for paired-end runs.

Table 110 Genomic code classifications, for mate-pair pairing

Class name	Strand	Orientation	Insert size	Preference
AAA	Same	R3 to F3	Normal	1
AAB	Same	R3 to F3	< min	2
AAC	Same	R3 to F3	> max	2
ABA	Same	F3 to R3	Normal	2
ABB	Same	F3 to R3	< min	3
ABC	Same	F3 to R3	> max	3
BAA	Different	Outward	Normal	3
BAB	Different	Outward	< min	4
BAC	Different	Outward	> max	4
BBA	Different	Inward	Normal	3
BBB	Different	Inward	< min	4
BBC	Different	Inward	> max	4

The first letter determines whether the two tags match the same strand:

- **A** – Match the expected strand.
- **B** – Match different strands.

The third letter determines if the distance between two hit positions is within the correct range:

- **A** – Within the correct range.
- **B** – The distance is less than the insert size measurement minimum (denoted by "< min" in the table above).
- **C** – The distance is greater than the insert size measurement maximum (denoted by "> max" in the table above).

The default minimum insert size is 0. For paired-end and long mate pair workflows, the default maximum insert sizes are 2000 and 20,000, respectively. An insert size measurement automatically determines the actual values using the defaults as the largest interval, and the defaults are used if the insert size measurement fails.

The middle letter has different meanings, depending on whether the first letter is A or B. If the first letter is A, the second letter indicates if the order of the two tags is correct:

- **A** – The order is correct, and R3 is upstream of F3.
- **B** – The order is incorrect, and F3 is upstream of R3.

When the two tags are on different strands, the second letter indicates whether the two tags are pointing toward or away from each other

- **A** – Pointing away from each other.
- **B** – Pointing toward each other.

In addition, C** classification is assigned to a pair of reads that have one hit each, but are on different chromosomes. If both F3 and R3 tags are present, but only one has a unique hit while the other one is unmapped, the pair is labeled D**. Finally, if one tag from a pair has a unique hit but the other one is missing, as opposed to unmapped, then the pair is classified as E**.

The Preference value indicates likelihood of the variation can occur in nature. Larger values indicate the variation is less likely to occur in nature.

Table 111 shows the classifications for paired-end runs. Summary statistics of these categories are calculated in the optional BAMStats Read pair Type Statistics Report (.ReadPair.Type.cht).

Table 111 Genomic code classifications, for paired-end pairing

Class name	Strand	Orientation	Insert size	Preference
AAA	Different	Inward	Normal	1
AAB	Different	Inward	< min	2
AAC	Different	Inward	> max	2
ABA	Different	Outward	Normal	2
ABB	Different	Outward	< min	3
ABC	Different	Outward	> max	3
BAA	Same	F3 to F5	Normal	3
BAB	Same	F3 to F5	< min	4
BAC	Same	F3 to F5	> max	4
BBA	Same	F5 to F3	Normal	3
BBB	Same	F5 to F3	< min	4
BBC	Same	F5 to F3	> max	4

FAQ – BAMStats

1

In the BAMStats summary table, why do the `NumFragmentsPassingFilters` and `NumFragmentsTotal` fields have the same totals?

`NumFragmentsTotal` reports the total number of fragments (read pairs or reads) passed from the instrument. `NumFragmentsPassingFilters` reports the number of fragments that pass the instrument filter. For the 5500 instrument, all reads pass filters when using color space, and so these two numbers are the same. If the run is performed with ECC, then the base space reads have filters applied to remove those of lowest quality. For an ECC run mapped in base space using LifeScope™ Software, the filtered reads are reported in LifeScope™ Software, and the Total and Filtered numbers are different.

2

What changes are required for a job with a large number of contigs?

For a job with a large (more than a few thousand) number of contigs, change the system property `max.head.node.java.heap.space` from the default of 64 to 128, or to an intermediate value between 64 and 128.

Modify this parameter in the <*installdir*>`/etc/analysis/system.properties` file.

Because changes to system properties affect all runs of all users, you could consider changing the parameter back to its original value after the large contig run completes. Setting `max.head.node.java.heap.space` to 128 could present a problem if your head node is extremely loaded.

20 Run a WT Coverage Analysis

This chapter covers:

- Overview 399
- WT coverage input files 399
- WT coverage parameters 400
- Coverage output files 401

Overview

The whole transcriptome coverage module calculates read coverage per position.

WT coverage input files

The WT coverage module takes as input one or more BAM files containing mapped data. The following describe the alignment input accepted. This module:

- Accepts only fragment data for RNA libraries.
- Accepts multiple BAM files as input.
- Accepts BAM files with different read lengths.
- Accepts both color space and base-space files, and a combination of color and base-space data.

WT coverage parameters

Table 112 lists the WT coverage parameters.

Table 112 WT coverage parameter description

Parameter name	Default value	Description
coverage.run	1	Whether or not to run the WT coverage module. Allowed values: • **0**: Do not run the coverage module. • **1**: Run the coverage module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
RNA.bam.file	—	A comma-separated list of paths to the input BAM files. Relative and absolute paths are accepted. Generated automatically by LifeScope™ Software during a workflow. Use this parameter only when running this module standalone.
Optional parameters		
RNA.coverage.min.quality	2	(*Optional*) The mapping quality value threshold for selecting the alignments. Alignments with a mapping quality value lower than this threshold are not written to output. Allowed values: Integers 0–100.
RNA.coverage.min.value	0	(*Optional*) The mapping quality threshold value for selecting alignments from the input BAM. In order to be included in the output, a position's coverage must be greater than or equal to this value. Allowed values: Integers >= 0.
RNA.coverage.per.chromosome	true	(*Optional*) Whether coverage output is generated as one file per every chromosome per strand, or as a single file with coverage of all chromosomes per strand. Allowed values: • **false**: Generate a single file with coverage of all chromosomes per strand. • **true**: Generate one file per every chromosome per strand.
RNA.coverage.primary.only	true	(*Optional*) Whether only primary alignments or all the alignments from the input BAM file are used as input. Allowed values: • **false**: Consider all alignments. • **true**: Consider only primary alignments (both gapped and ungapped).
Resource parameters		
memory.request	4gb	Memory request. Include the units gb in the setting.
java.heap.space	4000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.

Coverage output files

The WT coverage module's output files are in the wiggle (*.wig) file format. Wiggle files are visualized in genomics browsers such as the Integrative Genomics Viewer (IGV). You can download the IGV browser from the Broad Institute website:

www.broadinstitute.org/igv

The wiggle file specification is available from this site:

http://genome.ucsc.edu/goldenPath/help/wiggle.html

If the parameter `RNA.coverage.per.chromosome` is set to `false`, the WT coverage module generates two output files with coverage of all chromosomes per strand:

- `coverage_positive.wig`
- `coverage_negative.wig`

If the parameter `RNA.coverage.per.chromosome` is set to `true`, the WT coverage module generates one coverage file per chromosome per strand:

- `coverage_chr**_positive.wig`
- `coverage_chr**_negative.wig`

21 Run a WT Count Known Genes and Exons Analysis

This chapter contains:

- Overview 403
- Input files 404
- WT counts parameter description 404
- WT counts algorithm description 404
- Output files 407
- FAQ – WT counts 408

Overview

The module performs counting of features provided in a GTF file. It counts:

- Exons
- Genes

The module allows filters, generates RPKM, and annotates how many exons are expressed in a gene. This tool annotates only the Exon features in a GTF file

IMPORTANT! Alignments must come from the same strand as the feature to contribute to the count. A non-gapped tag contributes to a feature's count if it overlaps the feature and has no more than three bases outside the feature. A gapped tag contributes to a features count if one of its match regions terminates at a feature boundary.

Input files

Use the WT counts module to count the number of reads that align within genomic features. The WT counts module takes the following as input:

- A set of read alignments (BAM file)
- A set of user-defined (or default) stringency parameters
- A set of genome annotations (GTF file)

Paired-end and fragment libraries may be combined in the BAM input. Unmapped reads in the BAM input are not used.

WT counts parameter description

Table 113 describes parameters that you can configure for the WT counts module.

Table 113 Counts module algorithm parameters

Parameter	Default	Description
Algorithm parameters		
wt.gtf.file	${annotation.gtf.file}	The GTf file used to create the genemodel.
wt.count.features.gene.count	1	Generate whole transcriptome gene count. Allowed values: • **0**: Do not generate whole transcriptome gene count. • **1**: Generate whole transcriptome gene count.
wt.count.features.primary	1	Whole transcriptome counts primary. Allowed values: • **0**: Count all alignments. • **1**: Count only primary alignments. Recommended.
wt.count.features.min.mapq	10	The minimum alignment quality for an alignment to be counted. Allowed values: Integers 0–255.
wt.count.features.overflow.limit	3	Whole transcriptome overflow limit. An alignment may align outside the boundaries of a given exon feature (called overflowing). If an alignment overflows the feature more than the value specified in this parameter, then that read is not counted towards that feature. *Both* sides of the exon are checked for overflows, and the sum of all overflowing bases (on both sides) is compared to this threshold. Allowed values: Integers 0–100.
Optional resource parameters		
java.heap.space	8000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.
memory.request	8gb	Memory request. Include the units gb in the setting.

Table 114 lists WT counts parameters that we do not recommend changing.

Table 114 WT Counts internal parameters

Parameter name	Description
wt.exon.reference	The exon FASTA reference file. Default is ${analysis.output.dir}/wtexonsequenceextractor/exons.fasta.

Table 115 lists parameters that you do not specify when you run the module in a standard workflow.

Table 115 WT count parameters supplied by LifeScope™ Software

Parameter name	Description
wt.counts.run	Enable the WT counts module. Allowed values: • **0**: Do not run the module. • **1**: Run the WT counts module. Default.
wt.exon.reference	The exon FASTA reference file. Default is ${analysis.output.dir}/wtexonsequenceextractor/exons.fasta. This file is defined by the wt.f5.exseqext.output.reference parameter in the WT exon sequence extractor.
wt.gtf.file	The GTf file used to create the genemodel. Default: ${annotation.gtf.file}.

WT counts algorithm description

Filters

Stringency parameters govern the set of alignments in a BAM file. These alignments are considered in counting. Reads that do not meet these stringency parameters are filtered out and do not contribute to the result.

These parameters govern stringency:

- Filter alignment mode (`wt.count.features.primary`)
- Minimum mapping quality (`wt.count.features.min.mapq`)

The filter alignment mode governs how multi-mapping reads are handled. The option settings for filter alignment mode are `all` and `primary` (see Table 116). Both options are supported for single-read and for paired-end data.

Table 116 Alignment filter modes

Mode	Description
all	All alignments are considered.
primary	Only primary alignments are considered.

If a minimum mapping quality is selected, only alignments with the minimum quality or greater are considered. Likewise, if a minimum score is selected, only alignments having at least the minimum score are considered.

Filtering by unique or top criteria is included mainly for legacy reasons. We recommend that you use only default setting of mapping quality to filter reads.

Half (0.5) a count is assigned for each exon of a read that spans an exon-exon junction.

Mapping quality incorporates an assessment of score and uniqueness. To filter using only mapping quality specify a minimum mapping quality, and set the alignment filter mode to all, and specify a minimum mapping quality.

In previous releases of LifeScope™ Software, filtering single-read alignments using the unique alignment filter mode was recommended. When unique filtering is enabled, only alignments that satisfy the following criteria pass the filter:

- The total number of alignments for the read is less than the maximum allowed to be reported in mapping (the Z parameter).
- The alignment must be the single best alignment of the set of alignments for the read.
- The score of the alignment must be sufficiently better than that of any suboptimal hits. If no suboptimal hits are present, a suboptimal hit is assumed to exist with a score one less than the minimum possible score for the anchor size and mismatch level used in mapping. The difference between the score of the alignment score and the best suboptimal alignment must be greater than or equal to the clear zone.

Counts

Each feature defined in the genome annotation is assigned a tag count. A tag count is the total number of alignments that pass the stringency filters and are consistent with the annotation. The criteria for counting an alignment toward a feature are:

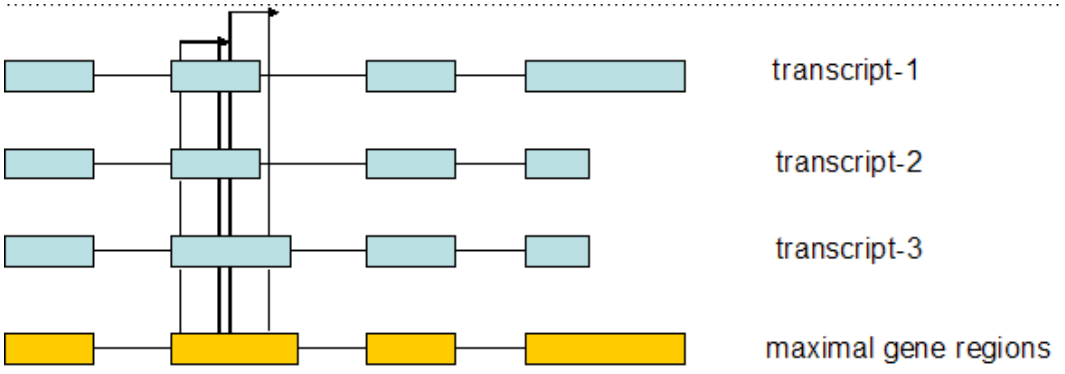
- The alignment must overlap the feature's numeric coordinates (contig, start, end).
- The alignment strand orientation must be consistent with the feature strand orientation.

  Note: F3 reads align to the sense genomic strand; F5 reads align to the antisense genomic strand.

- If the alignment does not span an intron, the alignment must include no more than three bases outside the feature.
- If the alignment spans an intron, the position of the exon-intron boundary must match in the alignment and the feature.

The read count for a given gene is defined as number of reads that map to the maximal region (features) of the merged gene. Maximal features are calculated from the transcripts provided in the GTF file by using the gene model. Reads that map to overlapping exons or features of a given gene are not counted multiple times.

The following figure describes the component flow during exon counting.



RPKM

The RPKM of exons (the Reads Per Kilobase of exon model per Million mapped reads), which is a normalized measure of expression, is also reported. This metric is calculated with the formula:

$$RPKM = 10^9 \times \frac{ExonReadCount}{TotalReadCount \times ExonLength}$$

The number of total uniquely mapped reads on exons is used as the normalization factor. The RPKM value is also reported by the WT splice finder module (see "Splice finder parameters" on page 410).

Output files

This section describes the output of the WT counts module.

GTF output

The output format of the WT counts module follows the same GTF file format as the supplied genome annotation. The score field of the GTF file contains the tag count. RPKM is reported in the attributes field.

The following is example GTF output.

```
# filterOrphanedMates=false, minMapq=10, minScore=0, scoreClearZone=5,
alignmentFilteringMode=PRIMARY
#number_of_mapped_reads=190375154
chr1  BioScope  exon  4225  4692  86  -  .  gene_id "WASH5P"; transcript_id
"NR_024540"; RPKM "0.97";
chr1  BioScope  exon  4833  4901  5   -  .  gene_id "WASH5P"; transcript_id
"NR_024540"; RPKM "0.38";
chr1  BioScope  exon  5659  5810  18  -  .  gene_id "WASH5P"; transcript_id
"NR_024540"; RPKM "0.62";
```

Tab output

The following fields are in the `gene_counts.tab` output file:

- Gene name
- Chr
- Start
- End
- Strand
- ReadCount
- Length (the maximal gene region length)
- RPKM
- Number of features (exons)
- Number of expressed features (>10 reads)
- Transcript names

Every gene is reported whether it is expressed or not.

FAQ – WT counts

1

Does the whole transcriptome module count invalid reads?

A: No. If a read is invalid (that is, considered invalid by Picard software), that read is not included in the count of mapped reads.

22 Run a WT Splice Finder Analysis

This chapter contains:

- Overview 409
- Splice finder parameters 410
- SASR splice finder 413
- Output files 416

Overview

Transcripts of eukaryotic genomes are transcribed from exons which are usually separated by one or more introns. The spliceosome machinery removes the introns from pre-messenger-RNAs to generate mature messenger-RNAs. The removed introns are called splice junctions. Alignments from an RNA-Seq experiment may map on these splice junctions, meaning the alignment has two parts that are separated by thousands of bases on the genome. The splice finder software is designed to discover these splice junctions efficiently and annotate the type and evidence for each discovered junction.

A fusion junction is a section of transcribed RNA that maps to an exon from one gene followed by an exon from another gene. It can occur as the result of a translocation, deletion, or chromosomal inversion. A splice junction excludes exon-to-exon boundaries that arise from alternative splicing for a gene.

The module takes as input:

- A BAM file of either paired-end or fragment libraries reads sequenced from transcribed RNA
- A list of exons. Each exon includes its color sequence and the reverse of that sequence.
- A GTF file
- A FASTA file of the reference genome

Unmapped reads in the BAM input are not used. Partially mapped reads are used to find gene fusions.

Splice finder parameters

Table 117 describes a workflow parameter, `analysis.space`, that is used in the splice finder module. The `analysis.space` parameter is set in the `tertiary/global.ini` file.

Table 117 Analysis space parameter used in the splice finder module

Parameter name	Default	Description
analysis.space	auto	Specifies your *preference* for base space or color space analysis. Allowed values: • **auto**: The analysis space is determined based on the mapping space and the presence of color space data in the analysis BAM files. • **base**: Prefer base space analysis. • **color**: Prefer color space analysis. See "Analysis space" on page 77 for a description of other factors determining the analysis space used by the splice finder module.

Table 118 lists the parameters for the splice finder module.

Table 118 Splice finder parameter description

Parameter name	Default value	Description
Required parameters		
wt.splice.finder.first.read.max.read.length	50	F3 read length. This value is auto-populated if possible: Allowed values: Integers 25–150.
wt.splice.finder.second.read.max.read.length	25	F5 read length. This value is auto-populated if possible: Allowed values: Integers 25–150.
Optional parameters		
wt.splice.finder.min.exon.length	25	All exons shorter than this are removed: Allowed values: Integers 0–500.
Optional single read junction evidence collection		
wt.splice.finder.single.read	1	Enable single–read fusion finding. Allowed values: • **0**: Do not run single–read junction finding. • **1**: Enable single–read junction finding.
wt.splice.finder.single.read.min.overlap	10	The minimum number of contiguous color positions that must align in a read–exon map. Allowed values: Integers 1–100.
wt.splice.finder.single.read.max.mismatches	2	The maximum number of mismatches allowed in a read-exon map. Allowed values: Integers 0–25.
wt.splice.finder.single.read.clip.size	2	Progressive unit size for clipping at the end of read. Allowed values: Integers 0–25.

Table 118 Splice finder parameter description *(continued)*

Parameter name	Default value	Description
wt.splice.finder.single.read.clip.total	10	Total size for clipping at the end of read. Allowed values: Integers 0–25.
wt.splice.finder.single.read.ReportMultihit	0	How to count single reads that align to multiple splice junctions. Allowed values: • **0**: Do not count them as evidence to any of the junctions. • **1**: Count them as evidence to all of the junctions. • **2**: Count them as evidence to the first of the junctions. Option 0 provides uniqueness.
wt.splice.finder.single.read.remap	0	Single read remap. 0=false, 1=true. Allowed values: • **0**: SASR registers the evidence, but does not remap. • **1**: SASR remaps reads that have already been mapped to a junction, and registers evidence for that remap.
wt.splice.finder.single.read.clip.5.prime	1	Single read clip 5 prime. Allowed values: • **0**: SASR clips only the 3' end. • **1**: SASR clips both the 5' and the 3' end of the read.
wt.splice.finder.single.read.min.read.length	37	SASR considers shorter reads (in number of colors) to be inadmissible as evidence. Allowed values: Integers 0–100.
Optional paired read junction evidence collection		
wt.splice.finder.paired.read	1	Enable paired-end fusion finding. Allowed values: • **0**: Do not run paired-end junction finding. • **1**: Enable paired-end junction finding.
wt.splice.finder.paired.read.min.mapq	10	A read-pair that has mapping quality lower than this is inadmissible as evidence. Allowed values: Integers 0–225.
wt.splice.finder.paired.read.avg.insert.size	120	Average insert size of the paired-end library. Allowed values: Integers 0–1000.
wt.splice.finder.paired.read.std.insert.size	60	Standard deviation of the insert size for the paired-end library. Allowed values: Integers 0–1000.
Combined caller for junction		
The parameters below allow splice finder to combine evidence from single-reads and from paired-reads to call junctions, alternative splices, and fusions, respectively. In each case, an event (for example, a fusion) is reported if SRE>x and PRE>y and ([SRE+PRE)>z or MAX(SRE,PRE) > w), where SRE = unique single-read evidence, PRE = unique paired-read evidence, and x, y, z, and w are specified by the parameters below.		
wt.splice.finder.junction.split.read.min.evidence	1	x in SRE>x Minimum single read evidence for junction. Allowed values: Integers 0–50.

Table 118 Splice finder parameter description *(continued)*

Parameter name	Default value	Description
wt.splice.finder.junction.paired.end.min.evidence	1	y in PRE>y Minimum paired-end read evidence for junction. Allowed values: Integers 0–50.
wt.splice.finder.junction.combined.min.evidence	2	z in (SRE+PRE)>z Minimum combined evidence for junction. Allowed values: Integers 0–50.
Combined caller for (same gene) alternative splicing		
wt.splice.finder.alt.splicing.split.read.min.evidence	1	x Minimum single read evidence for alternative splicing. Allowed values: Integers 0–50.
wt.splice.finder.alt.splicing.paired.end.min.evidence	1	y Minimum paired-end evidence for alternative splicing. Allowed values: Integers 0–50.
wt.splice.finder.alt.splicing.combined.min.evidence	2	z Minimum combined evidence for alternative splicing. Allowed values: Integers 0–50.
Combined caller for gene fusion		
wt.splice.finder.fusion.split.read.min.evidence	2	x Minimum split read evidence for junction. Allowed values: Integers 0–50.
wt.splice.finder.fusion.paired.end.min.evidence	2	y Minimum paired-end evidence for junction. Allowed values: Integers 0–50.
wt.splice.finder.fusion.combined.evidence	4	z Minimum combined evidence for junction. Allowed values: Integers 0–50.
Other options		
wt.splice.finder.output.format	3	Output format parameter. Allowed values: • **1**: Tabular output format • **2**: BED output format • **3**: BED, tabular, and SEQ output formats • **4**: All formats, including Circos. With option 3, both tabular, BED, and also additional SEQ files are created separately. SEQ files contain 50 base pairs of sequences from each end of a junction exon and are useful for validation.

Table 120 lists parameters that we do not recommend changing.

Table 119 Splice finder internal parameters

Parameter name	Default value	Description
wt.splice.finder.base.space	0	Call junctions in base space. Allowed values: • **0**: Call junctions in color space. • **1**: Call junctions in base space.
Optional resource parameters		
java.heap.space	14000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.
memory.request	15gb	Memory request. Include the units gb in the setting.

Table 120 lists parameters that you do not specify when you run the module in a standard workflow.

Table 120 Splice finder parameters supplied by LifeScope™ Software

Parameter name	Description
splice.finder.run	Enable the splice finder module. Allowed values: • **0**: Do not run the splice finder module. • **1**: Run the splice finder module. Default.
wt.exon.reference	The exon FASTA reference file. Default is ${analysis.output.dir}/wtexonsequenceextractor/exons.fasta. This file is defined by the wt.f5.exseqext.output.reference parameter in the WT exon sequence extractor.
wt.gtf.file	The GTf file used to create the genemodel. Default: ${annotation.gtf.file}.

SASR splice finder

SASR splice finder module summary

1. Read in and process the exon list.
2. For each admissible read:
   a. Let *L* be the set of exons that map to the prefix of the read.
   b. Let *R* be the set of exons that map to the suffix of the read.
   c. For each exon *e* in *L*, and each exon *f* in *R*,
      if *overlap(e, r) + overlap(f,r) + 1 = length(r),* then
      register read *r* as evidence for junction *e-f*.
3. Output the list of junctions and evidence.

SASR splice finder module evidence evaluator

After the single-read and paired-end splice finders fully process the BAM file, the evidence evaluator evaluates the existing evidence and makes junction calls based on user-defined thresholds.

Pairs mapped less than a confidence threshold (set with the parameter `wt.splice.finder.paired.read.min.mapq`) are not used. By default, reads with an original length of less than 37 are not considered for splicing evidence. The parameter `wt.splice.finder.single.read.min.read.length` changes the default. Spliced reads are used as evidence unless remap is enabled (with the parameter `wt.splice.finder.single.read.remap`). When remap is enabled, the spliced read is remapped and then used as evidence if it falls on a junction.

The default thresholds require at least one single-read and one paired-end evidence to call a junction. The calling of special junctions, such as alternative splicing and fusion, is described in the following three paragraphs.

The evidence evaluator also counts the number of unique pieces of evidence. In the case of paired-end evidence, this count is the number of unique mate-alignment start positions, and in the case of single-read evidence, it is the number of unique overlap lengths for the "left" exon.

The candidate's data structure stores the evidence for junctions as properties of its exons. Each exon points to other exons for which there is evidence of a junction, where the current exon is a source, and the pointed-to exon is the destination. The data structure of the candidate corresponds to a directed exon evidence graph (shown in Figure 41 on page 414.

In a junction evidence graph, nodes are exons, and each directed edge corresponds to junction evidence from a source exon to a destination exon. Each edge has two values:

- Unique SASR evidence
- Unique PR evidence

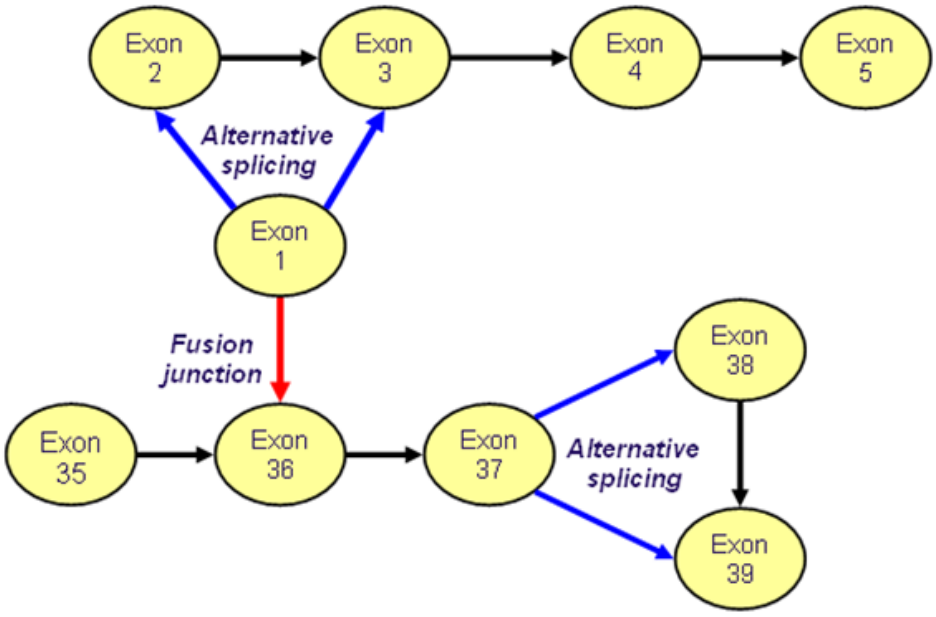


Figure 41 A directed exon evidence graph

Referring to Figure 41, the black arrows correspond to normal junctions, blue arrows correspond to alternative splice junctions, and red arrows correspond to fusions.

Most junctions found are regular junctions, meaning junctions between exons of the same gene in the RefSeq expected order. However, more interesting special junctions are also detected during this evaluation step. Alternatively, spliced junctions are defined as multiple junctions from a given exon to two other exons of the same gene within the given sample. Fusion junctions are defined as junctions between exons of different genes.

Call junctions and fusions with single read only

LifeScope™ Software includes user-defined parameters that allow the software to call junctions and fusions with fragment (F3 only) datasets. However, LifeScope™ Software's fusion caller is designed to work with paired-end reads. Calling fusions with only fragment reads likely results in a large number of false positives. We observed that on the same dataset, LifeScope™ Software calls ten times or more fusions using only single-read evidence compared with calls using both paired-end and single-read evidence. We suggest rigorous post-filtering and sorting by total unique single-read evidence for validating those fusion calls.

For detecting known same-gene junctions we observed that using only single-reads calls 80% of the junctions called by paired-end reads (see Figure 42). However, the remaining 20% usually contains lowly-expressed or harder to detect junctions, and we observed a need to double the total number of reads in order to achieve the same sensitivity. As a result, detection of exon junctions with fragment reads is not recommended due to lower specificity and sensitivity.

Figure 42 compares single-read (SR) vs. paired-end (SR+PE) junction calling sensitivity. The left two bars show SR and SR+PE number of junction calls when using UHR barcode 1. The middle two bars are with UHR barcode 2. The right two bars show the increase in number of calls when the two barcodes are merged and calls are repeated with effectively double the amount of reads from the same library. For both SR and SR+PE, two unique evidences are required to call a junction. For SR+PE, at least one SR and at least one PE evidence is required as well.

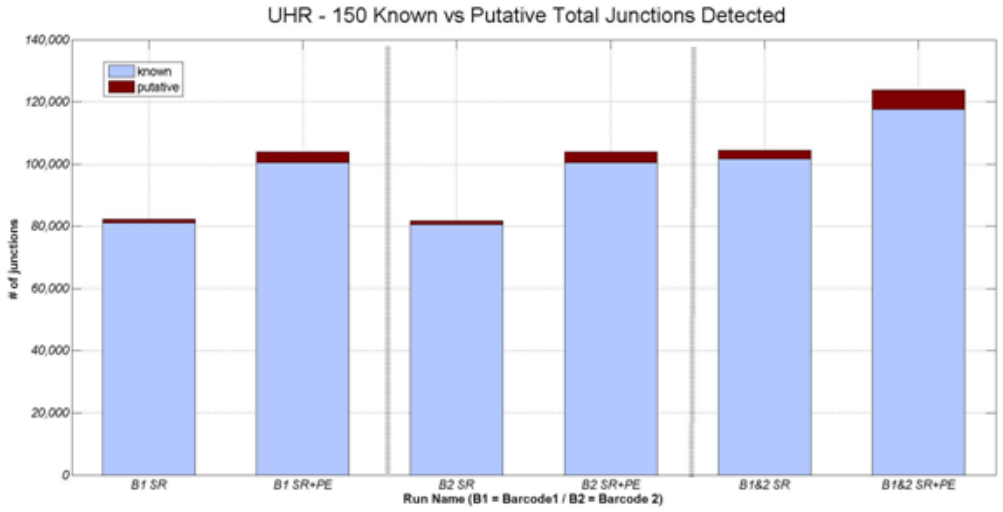


Figure 42 Comparison of single-read (SR) vs. paired-end (SR+PE) junction calling sensitivity with two barcodes

Output files

The following types of output are generated, depending on the parameter `wt.splice.finder.output.format`:

- Tabular (a tab-delimited junction database) files with the following contents:
  - All detected junctions
  - Junctions interpreted as arising from alternate gene splicing
  - Junctions interpreted as arising from inter-gene fusion
- BED
- SEQ
- Circos
- Statistics files, which contain the following parts:
  - Paired-end splice finder statistics
  - SASR splice finder statistics
  - Evidence matrix

These files are described in the remainder of this section.

Tabular junction files

Three output files are generated with the `.tab` extension for regular junctions, alternatively-spliced junctions, and fusion junctions. Table 121 describes the fields for these files.

Table 121 Junction tab-delimited output format

Field name	Description
Exon-1	The gene id followed by the exon order on the gene.
Exon-1-reference	The reference FASTA file.
Exon-1-strand	± strand.
Exon-1-start	The start position.
Exon-1-end	The end position.
Exon-2	The gene id followed by the exon order on the gene.
Exon-2-reference	The reference FASTA file.
Exon-2-strand	± strand.
Exon-2-start	The start position.
Exon-2-end	The end position.
Exon-1-size	The size of the first exon.
Exon-2-size	The size of the second exon.
Exon-1-readcount	The number of reads that map on the first exon.
Exon-2-readcount	The number of reads that map on the second exon.
Exon-1-F3-RPKM	The Reads Per kbp Per Million Reads (RPKM) exon-1 from F3.
Exon-2-F3-RPKM	The RPKM reads (RPKM) exon-2 from F3.
Exon-distance	Distance between two exons. The exon-distance is not applicable if it is on a different chromosome.

Table 121 Junction tab-delimited output format *(continued)*

Field name	Description
Total-PR-evidence	The total paired-end evidence for the junction.
Unique-PR-evidence	The unique paired-end evidence for the junction.
Total-SR-evidence	The total single-read evidence for the junction.
Unique-SR-evidence	The unique single-read evidence for this junction.
JCV	A junction confidence value.
Known	"K" if a junction is known, and "P" if a junction is putative.
E1-all-genes	The list of all genes to which exon-1 was mapped.
E2-all-genes	The list of all genes to which exon-2 was mapped.
Other	Other information provided about the junction. For alternatively spliced junctions, this field reports the length of the alternative exon.

The columns that are most relevant to the splice finder are Unique-PR-evidence and Unique-SR-evidence. Also of interest are the two metrics RPKM and JCV.

The purpose of the RPKM (Reads Per Kilobase of exon sequence, per Million reads) metric is to provide a normalized exon expression level. This metric is calculated with the formula:

$$RPKM = 10^9 \times \frac{ExonReadCount}{TotalReadCount \times ExonLength}$$

The RPKM value is also reported by the WT counts module (see "WT counts parameter description" on page 404).

The purpose of the Junction Confidence Value (JCV) metric is to aid in the detection of false positives and in other decisions that depend on a confidence level. Depending on the coverage of the sequencing experiment and exons being tested, the algorithms might generate results that require different user-defined thresholds to detect the most likely fusions. The major contributor of false positives possibly is highly-expressed genes, which have a considerable random chance of encountering a fusion event due to sequencing errors and mapping to homologous exons. A statistical confidence metric (see below) was developed to detect such false positives and assign a confidence level to the evidenced junction.

$$JCV_{j_{x-y}} = \sum_{i=1}^{n} PQV_i - 10\log_{10}\left(EEM_{j_{x-y}}\right)$$

Equation 1. Junction Confidence Value (JCV)

$$EEM_{j_{x-y}} = \frac{RC_x}{\frac{l_x}{\mu_T + 3\times\sigma_T}} \times \frac{RC_y}{\frac{l_y}{\mu_T + 3\times\sigma_T}}$$

Equation 2. Error expectation metric (EEM)

where PQV_i is the Phred-scale pairing quality value for the *i*'th unique paired read evidence for a candidate junction j_{x-y} and *x* and *y* are the junction exons. For each unique single read evidence, the PQV_i is set to 10. If there are multiple alignments *for* a given unique start point, take the *PQV* of the first such alignment. *RC* is the absolute proper mapped read count for the corresponding exon and is the length of the exon; μ_T and σ_T are the mean and standard deviation of the insert size for the current experiment, *T*.

Error expectation metric (EEM) is used to quantify highly expressed junctions. This metric is hard to calculate due to genome complexity and homology of exons. Our estimation considers the number of reads mapped to the exons, the length of exons, and a conservative insert range.

After the equation is calculated, a JCV that is larger than 100 is set to 100, and those smaller then 0 are set to 0. If a JCV approaches 100, it is more likely to be a real junction.

Junction examples

For a junction detected between exon-1 of size 5,000 and exon-2 of size 400, with mean insert 100 and standard deviation 33.3 bp, assume in case-1, there were three unique junction evidences with PQV 30, 20 and 40 respectively and in case-2 only a single unique evidence with PQV 20. There were 900 properly mapped alignments to exon-1 and 100 such alignments for exon-2. In case-3 has 3 unique evidences similar to case-1, but the exons are assumed to have 100 times more coverage each. The junction confidence value for a junction between these exons would be:

$$30+20+40-10\times\log_{10}\left(\frac{900}{5000}\times\frac{100}{400}\times 200^2\right)=90-10\times\log_{10}(1800)\approx 50\text{ to }60$$

Equation 3. Case-1: 3 unique junction evidence between 9x and 12x exons

$$20-10\times\log_{10}\left(\frac{900}{5000}\times\frac{100}{400}\times 200^2\right)=20-10\times\log_{10}(1800)\approx 0$$

Equation 4. Case-2: 1 unique junction evidence between 9x and 12x exons

$$30+20+40-10\times\log_{10}\left(\frac{90000}{5000}\times\frac{10000}{400}\times 200^2\right)=90-10\times\log_{10}(18{,}000{,}00)\approx 10\text{ to }20$$

Equation 5. Case-3: 3 unique junction evidence between 900x and 1200x exons

A simplified table of outputs is shown in Table 122.

Table 122 Simplified junction tab output

E1	E1-reference	s	E1-start/ E1-end	E2	E2-reference	s	E2-Start/ E2-End	Unique-PR	Unique-SR
AGRN-1	chr1	+	886872/ 886993	KLHL17-4	chr1	+	887069/ 887290	3	2
KLHL17-8	chr1	+	888580/ 888747	KLHL17-9	chr1	+	889163/ 889251	1	1
AGRN-11	chr1	+	969352/ 969500	AGRN-12	chr1	+	969577/ 969682	2	1
AGRN-14	chr1	+	970602/ 970766	AGRN-15	chr1	+	970976/ 971119	2	1
AGRN-17	chr1	+	971403/ 971508	AGRN-15	chr1	+	971640/ 971978	6	2
AGRN-20	chr1	+	972570/ 972697	AGRN-21	chr1	+	972816/ 972930	2	3
AGRN-21	chr1	+	972816/ 972930	AGRN-22	chr1	+	973019/ 973138	2	1
AGRN-26	chr1	+	974809/ 975038	AGRN-27	chr1	+	975146/ 975280	3	0
AGRN-27	chr1	+	975146/ 975280	AGRN-28	chr1	+	975476/ 975572	3	1
PUSL1-4	chr1	+	1234697/ 1234846	PUSL1-5	chr1	+	1234924/ 1235094	2	0
PUSL1-6	chr1	+	1235877/ 1235931	PUSL1-7	chr1	+	1236152/ 1236314	2	0
VWA1-0	chr1	+	1362170/ 1362727	VWA1-3	chr1	+	1364324/ 136600	4	0
MIB2-3	chr1	+	1548632/ 1548942	MIB2-4	chr1	+	1549017/ 1549188	5	1
MIB2-4	chr1	+	1549017/ 1549188	MIB2-5	chr1	+	1550038/ 1550144	1	1
MIB2-6	chr1	+	1550234/ 1550428	MIB2-7	chr1	+	1550529/ 1550671	2	1
MIB2-7	chr1	+	1550529/ 1550671	MIB2-8	chr1	+	1550789/ 1550896	2	1
MIB2-9	chr1	+	1551893/ 1551997	MIB2-10	chr1	+	1552080/ 1552242	2	2
MIB2-10	chr1	+	1552080/ 1552242	MIB2-11	chr1	+	1552317/ 1552450	3	0
MIB2-11	chr1	+	1552317/ 1552450	MIB2-12	chr1	+	1552539/ 1552687	2	0
MIB2-14	chr1	+	1553262/ 1553422	MIB2-15	chr1	+	1553516/ 1553642	3	1

Browser Extensible Display (BED) output

The BED format was developed to extend the UCSC Genome Browser with user-defined tracks. BED is used to visualize the splice and fusion junctions in the UCSC Genome browser and in the IGV browser (see Figure 43 on page 421). For general documentation about the BED format, including information about all of the BED fields, go to:

genome.ucsc.edu/FAQ/FAQformat.html

For information about visualization software, see "Pairing information in a BAM file" on page 505.

Each line in the track defines a junction where chromStart is the smaller of the coordinates of and chromEnd is the greater.

There are two blocks because a junction typically contains two exons. BlockSizes are the lengths of the exons. The block starts the beginning of the exons. When fusions on different strands or chromosomes, two lines are added to the output, with each line representing one chromosome. Different colors are used to color-code different types:

Figure 43 on page 421 shows the Upstream Hypersensitive Region (UHR) gene region displayed with the Integrative Genomics Viewer (IGV) for positions 3,530,193 to 3,548,355 of Human Chr-1. The following sections describe the tracks in Figure 43.

WIG (x2) tracks

The top two tracks show the genomic coverage using the negative strand and positive strand generated by the Bam2Wig tool (Max: 100 coverage).

BAM track

The middle track shows the alignments from the BAM file. For display purposes, reads are filtered with MAPQ threshold of 45 (a stringent filter). Bases with quality value 5 to 20 are shaded.

BED track

The fifth track shows the junctions detected by the splice finder (BED file). As shown in the figure, all junctions detected are "known" and so are shaded in green.

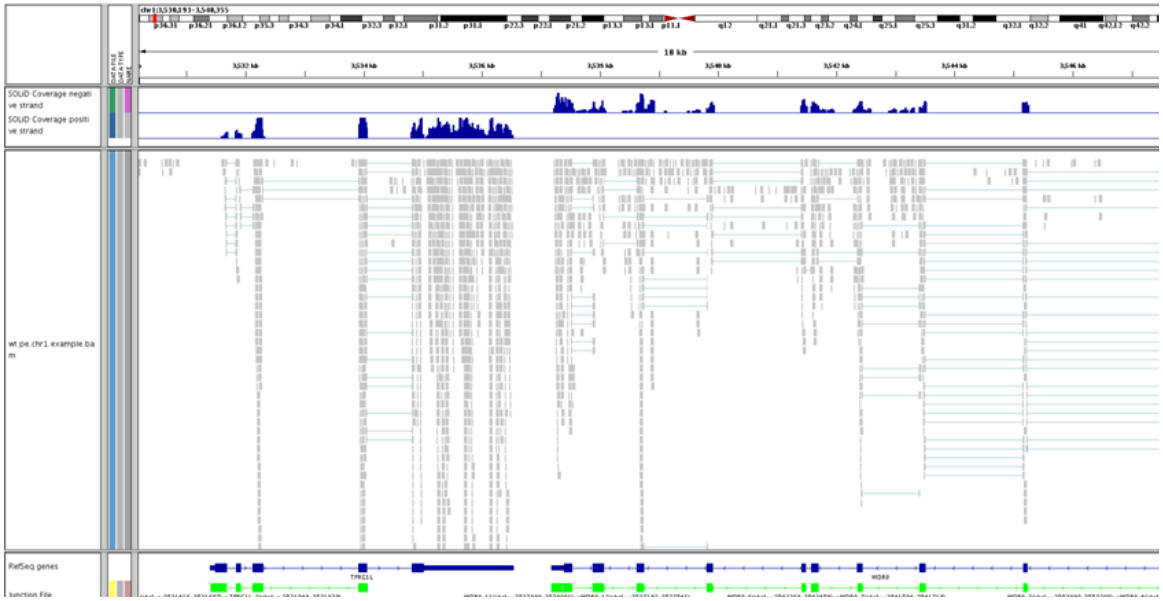


Figure 43 BED-IGV extended track example

SEQ

The optional SEQ output files are in FASTA-format and list regular junctions, alternatively-spliced junctions, and fusion junctions. The entries are 100bp from each end of putative junctions. If any of the exons is less than 100bp in size, the total exon sequence is added.

Circos

The Circos-ready format is generated when the `wt.splice.finder.output.format` parameter is set to 4. Gene fusion calls are output in Circos-compatible format, to enable easy visualization with Circos. Follow these steps to use this file with Circos:

1. Copy the folder splice-finder output directory `circus-fusion` and paste it under the Circos installation directory.
2. By default, only the human karyotype file is distributed. If you are working with another species, you must replace the karyotype file.
3. Run this command to start Circos:

   ```
   bin/Circos -conf circus-fusion/circos.conf
   ```

Output format values

The value of the `wt.splice.finder.output.format` parameter determines which file formats are included in the splice-finder output, as shown in the following list:

- `wt.splice.finder.output.format=1`
  - <bam_prefix>.alternative_splicing.tab
  - <bam_prefix>.fusion.tab
  - <bam_prefix>.junctions.tab
  - <bam_prefix>.stats
- `wt.splice.finder.output.format=2`
  - <bam_prefix>.alternative_splicing.tab
  - <bam_prefix>.fusion.tab
  - <bam_prefix>.junctions.bed

  - <bam_prefix>.junctions.tab
  - <bam_prefix>.stats
- `wt.splice.finder.output.format=3`
  - <bam_prefix>.alternative_splicing.seq
  - <bam_prefix>.alternative_splicing.tab
  - <bam_prefix>.fusion.seq
  - <bam_prefix>.fusion.tab
  - <bam_prefix>.junctions.bed
  - <bam_prefix>.junctions.seq
  - <bam_prefix>.junctions.tab
  - <bam_prefix>.stats
- `wt.splice.finder.output.format=4`
  - <bam_prefix>.alternative_splicing.seq
  - <bam_prefix>.alternative_splicing.tab
  - <bam_prefix>.fusion.seq
  - <bam_prefix>.fusion.tab
  - <bam_prefix>.junctions.bed
  - <bam_prefix>.junctions.seq
  - <bam_prefix>.junctions.tab
  - <bam_prefix>.stats
  - circos-fusion/fusion.genes
  - circos-fusion/fusion.links
  - circos-fusion/circos.conf
  - circos-fusion/ideogram.conf
  - circos-fusion/ticks.conf
  - circos-fusion/karyotype.human.txt

PART VII
Small RNA Analyses

23 Run a Small RNA Mapping Analysis

This chapter covers:

- Overview 425
- Example of running the small RNA mapping module 426
- Small RNA mapping input files 427
- Stages of mapping 428
- Quality alignments and unique alignments 432
- Small RNA mapping parameters 433
- Small RNA mapping output files 440
- Mapping statistics 441
- FAQ - Small RNA mapping 450
- FAQ – BAMStats 454

Overview

The LifeScope™ Software small RNA analysis modules are used to analyze high throughput small RNA sequencing raw data. These modules encapsulate a workflow to map reads to the target genome, to detect small RNAs in the genome, and to determine the expression of the detected small RNAs.

The small RNA mapping module maps small RNA (also known as microRNA or miRNA) reads using the mapping program mapReads. This module maps the small RNA reads to three different references in three steps:

- In the first step, the given set of reads are mapped to filter sequences in order to eliminate the reads generated from irrelevant sources (such as tRNA, adaptor sequences, or others).
- In the second step, the remaining reads from the first step are mapped to the list of known small RNA precursor sequences (miRBase annotations). This list is downloaded from Sanger's website. This lists for human hg18 and hg19 references are available in the LifeScope™ Software reference repository.
- In the third step, the unmapped reads from the second step are mapped to the genome reference sequence, in order to find novel small RNAs in the sample.

The mapped reads outputs from the second and third steps are merged, and the merged file is the primary output of the small RNA mapping module.

The major components of the small RNA mapping module are:

1. Filter reference mapping
2. miRBase reference mapping
3. Genome reference mapping
4. Merge of mapped reads output (MA files)
5. BAM generation (including ma2BAM, refcor, and sort steps) and BAM merge

Parameters are provided to skip any of the three mapping steps.

Usual read lengths from small RNA sequencing are 35–50 bases, and the length of the miRNA fragments is only 18–28 bp. The reads contain both the miRNA sequence and a P2 adapter sequence at the end. The mapReads phase does an extra step of extension onto adaptor, to correctly remove the adaptor sequence from the read and to report only small RNA sequences in the BAM file.

After the miRBase mapping step, the mapping output file (a MA file) has alignments with respect to the smaller miRBase reference locations. Before merging the MA files, the program converts these miRBase alignments to genomic reference locations, using the input GFF file (described in "Small RNA mapping input files" on page 427).

Example of running the small RNA mapping module

The `small.rna` standard workflow provides an example of how to run the small RNA mapping module. Here are example LifeScope™ Software shell commands for this workflow, which includes small RNA coverage and small RNA counts:

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd projects
# make a project, and open it
mk mirna
cd mirna
# make an analysis, and open it
mk run1
cd run1
# define the analysis type
set workflow small.rna
# define the input
add xsq rna.xsq
add xsq rna2.xsq
# specify the reference to be used
set reference hg18
# optionally change parameter defaults after this line
# list the analysis configuration
ls
# run the analysis
run
```

```
# display the run progress
ls
# exit the command shell
exit
```

Dummy XSQ file names are used in this example.

In order to change a parameter value in your analysis, use the `set param` shell command after the line `# optionally change parameter defaults after this line` in the example commands. For instance, to change the `create.unmapped.bam.files` parameter, use this shell command:

```
set create.unmapped.bam.files 0  secondary/smallrnamapping.ini
```

You must know which INI file to use for `set param` command. The `ls` command, executed in an analysis virtual directory after the workflow has been defined, lists the INI files used in the analysis.

Small RNA mapping input files

The small RNA mapping module requires the following input:

- One or more XSQ files containing sequencing data.
- A Genome Coordinates GFF file containing precursor sequences with genomic locations. This file is available the following link:

  http://www.mirbase.org/ftp
- A single multi-fasta reference file for the filter mapping step.
- A single multi-fasta reference file for the genome mapping step.

The mapping modules accepts as input one or more input XSQ files. The input reads can have different read lengths but they must be of the same library type.

RNA-Seq reads

The RNA-Seq reads used in the small RNA modules are different from the genomic reads. For RNA-Seq reads:

- Only transcribed sequences are measured by the system.
- Genome coverage is non-uniform due to variation in transcriptional intensity.
- A large subset of reads originate from uninteresting sequences such as ribosomal RNA (rRNA).

Reads input specification

Plan your analysis input carefully. The way you define your reads input affects the behavior of your analysis. The following factors control how your input data is analyzed:

- **Index (barcode) IDs –** Using an index ID restricts your input to the reads data of one or more indices.
- **Grouping of reads –** Each group of reads is analyses together as one specimen. The output data for a group is combined into one set of results files.
- **Multiple sample runs –** Unrelated reads can be processed together in one run of LifeScope™ Software, while analyzed separately as separate input data.

See "Define input data" on page 98 for more information on designing how to add input data to your analysis.

Legacy data

If the data you want to process with LifeScope™ Software is in CSFASTA and QUAL files, these files must be converted to the XSQ file format, through one of the following methods:

- The LifeScope™ Software UI handles converting these files to the XSQ format before mapping.
- If you are using the LifeScope™ Software command shell, you must first convert the CSFASTA and QUAL files to the XSQ format. See Appendix D, "XSQ Tools" on page 519 for information on the standalone XSQ converter.

Stages of mapping

The mapping algorithm involves following stages:

- **Scatter** – The process of distributing reads to the available compute nodes, to achieve parallelization.
- **Mapping** – Each tag is aligned to the reference.
- **BAM file generation** – The process of converting the output of mapping and pairing to BAM format. For color-space data, the base translation step is also included. Due to the scattering step, the BAM files at each compute node are a sub-set of the final BAM file and are referred to as baby BAMs.

  BAM generation involves these steps:
  - **Ma2BAM** – A conversion step, from MA format to BAM format, for fragment data only.
  - **Refcor** – Reference-assisted color-to-base translation.
  - **Sort BAM** – Sorting the BAM file from bead-id order to coordinate order.
  - **Merge BAM** – Merging the mini-BAM files created in the sort phase into the final output BAM files.

Figure 44 shows the phases involved in small RNA mapping.

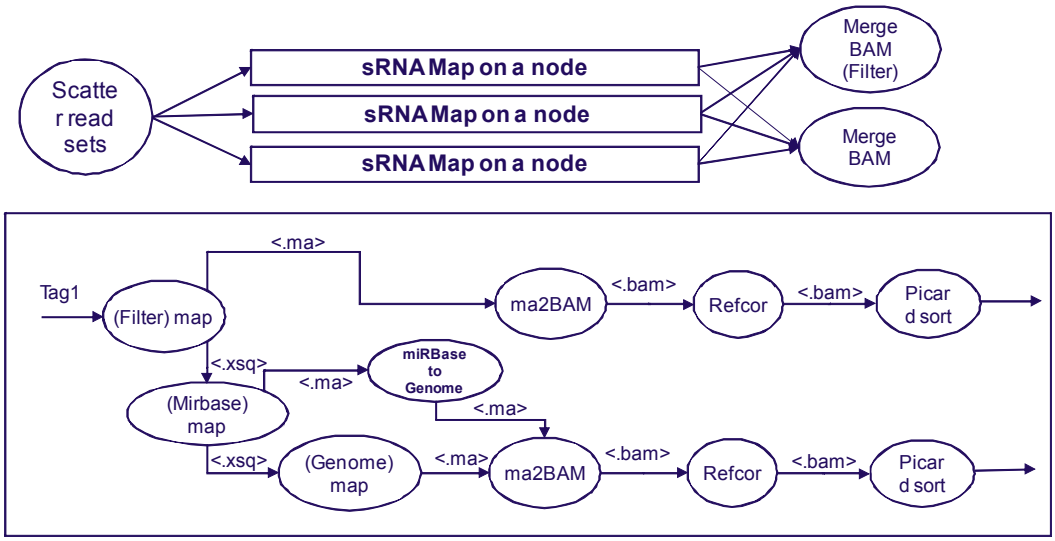


Figure 44 The phases within the small RNA mapping module

Scatter

The scatter stage distributes reads to the available compute nodes, in order to gain the efficiency benefits of parallel computing.

The mapping module can accept multiple read-sets, which are allowed to have different read lengths. The input set of read-sets from multiple XSQ files is first split based on read length. Fragment read-sets are partitioned into separate categories based on these read length categories:

- 0 ≤ Length < 25
- 25 ≤ Length < 35
- 35 ≤ Length < 50
- 50 ≤ Length < 60
- 60 ≤ Length < 75
- 75 ≤ Length
- Trimmed reads, with variable read lengths

The reads from each category are scattered into multiple jobs based on the parameters `fragmap.minreads.per.node` and `fragmap.number.of.nodes`. These parameters are described in the performance section.

Filter map

The reads are mapped to the filter reference in the filter mapping step. Reads that align to the filter reference are skipped from mapping to miRBase and genome references. The filtered reads are populated in separate BAM file inside the `outputs/filtered` directory. A filter reference for use with human reads is available with LifeScope™ Software. This reference contains:

- SOLiD® adapter sequences
- Human ribosomal RNAs (rRNAs)
- Human transfer RNAs (tRNAs)
- Other repetitive human sequences
- Single-base-repeat sequences, including Poly-A and Poly-T

Complete the following steps to construct a filter reference for another species.

1. Copy the adapter and single-base-repeat sequences to a new file.
2. Append FASTA-formatted species-specific ribosomal, transfer RNA, and other sequences to the filter reference.

- (*Optional*) Add the new filter reference file to the reference repository.
  - If you added the assembly for this species to the repository, copy the new filter file to the assembly directory. See "Add new reference files" on page 547. The filter reference is then set automatically when you use the follow shell command: `set reference` *assembly_name*.
  - If you do not have the assembly for this species in the reference repository, use the `analysis.filter.reference` parameter to include the new filter reference in your analysis. For example:

    ```
    set analysis.filter.reference filterfile secondary/global.ini
    ```

    An absolute path is recommended for `filterfile`. A relative path is interpreted from the Linux directory in which the shell was started (when the `lscope.sh shell` command was given). (The `secondary/global.ini` INI file is the correct file for the `analysis.filter.reference` parameter.)

miRBase map

This phase maps to known miRBase sequences. In a pre-process step miRBase database is converted from GFF to FASTA format to create the miRBase reference. Then the map module is invoked with the unmapped reads from the filter map step and the miRBase reference. The outputs of this step are the miRBase match file (or files) and unmapped reads in XSQ format.

Genome map

The unmapped reads from the miRBase map step are mapped to the genome reference in the genome mapping step to produce genome match files.

miRBase2Genome

The output of the miRBase mapping step has alignments with respect to the smaller miRBase reference locations. This phase converts alignments from the miRBase coordinate system to the genome coordinate system required with LifeScope™ Software BAM files.

BAM file generation

Ma2BAM

Ma2BAM is a format converter that changes match files to a pro-BAM format. These pro-BAM files do not contain sequence or quality information but contain relevant BAM meta data. The pro-BAM files are sorted by incoming bead-id order. Relevant meta data from XSQ files is captured in pro-BAM files at this stage. The parameters `smallRNA.bam.primary.align.only`, `smallrna.bam.mqv.threshold`, and `smallrna.create.unmapped.bam.files` control the functionality of the ma2BAM utility. The pro-BAM files are generated on the scratch folder.

Ma2BAM takes output of mapping (MA files) and converts them to pro-BAM format. During BAM generation, mapping quality values are also computed.

For reads with multiple ungapped alignments, the read with the highest mapping quality value is chosen as the primary alignment for the read, and is reported to the BAM file. In cases where there are multiple alignments with the same quality value, the primary alignment is chosen at random from among the alignments with the same quality value. The details of calculating mapping quality values are described below.

Mapping quality values

Quality values are Phred-scaled quality scores. For any given alignment, an MQV is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. Quality values are in the range 0–100:

- **0 –** The highest probability of error
- **100 –** The lowest probability of error

The following factors increase confidence and increase a read's quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce a read's quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

In order to be consistent with the Phred quality score (-10*log10[prob(error)]) used widely in literature, the quality is computed as the negative log odds of misaligning the read. The resulting quality values are normalized by the maximum possible value to ensure that the quality values are within the range 0–100.

Refcor

Reference Assisted Color-to-Base Translation (also known as refcor) combines information from color calls, optional ECC extra primer round calls, and the reference sequence, to enhance the quality of base-space reads by detecting and fixing color errors that convert into multiple base-space errors.

Refcor accepts a set of pro-BAM files, which contain alignments information without base or color sequence, and generates a corresponding set of BAM files containing alignments with base-space and color-space calls (on the scratch folder). These BAM files are sorted by bead-ID and contain dummy base calls and QVs, but are otherwise compatible with the BAM specification. The refcor module produces bead-ID-sorted BAM files that have valid calls and QVs.

For each read, the module constructs the primer-transition graph with vertices corresponding to all possible *k*–mers (*k*=4 for 2BE+4BE, *k*=2 for 2BE) in each read position. Two adjacent vertices are connected with an edge if they overlap in *k*–1 bases. A score corresponding to the (1–*Pe*), where *Pe* is the probability of being erroneous, is assigned to each of the vertices. Scores are derived from quality values corresponding to enumerated calls. When making a base call in a position, the current algorithm considers the cumulative probability of all sequences that have called base in that position. The evidence from the reference (base) is assigned to the vertex with *k*-mer ending in corresponding base only if the color transition between current and following reference bases matches the color call in a read. Such assignment reduces the reference over-correction in SNP positions. In order to reduce reference over-correction in positions with color errors, we assign to reference base a very low weight corresponding to QV of 8. The weight can be reduced to QV of 0, completely reducing the reference guiding in the color-to-base translation. This results in a smaller skew between reference to non-reference allele ratio. Increasing the weight results in false positive variant calls reduction. This process restores the phase shift caused by color errors or completely eliminates the errors.

The parameters `bamgen.refcor.softclip`, `smallRNA.bamgen.refcor.addcs`, `refcor.reference.weight`, and `refcor.base.filter.qv` control the refcor program. The defaults provided are validated for a wide range of data.

- If the XSQ file contains base-space data only, the XSQ base call and quality value are transferred to BAM file. In this case, no color calls are present in BAM file irrespective of the value of `bamgen.refcor.addcs`.
- If the XSQ file contains 2BE color-space data only, refcor uses aligned 2BE and reference to do base translation for the aligned part of the read. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, bases are determined by naively translating color to base. For these unaligned reads, the QV for the base at position *k* is the minimum color QV for all positions ≤ *k*.
- If the XSQ file contains ECC data, then reference with 2BE and 4BE color are used to generate the new base sequence. The base sequence and quality value are generated using a proprietary algorithm. For reads with no alignment, the original base-space data from ECC is preserved.

Sort BAM

The output of refcor is a set of BAM files in bead-id sorted order. These files are sorted in this step using Picard sort into coordinate sorted order before merge. These BAM files are referred to as mini-BAM files and are stored in the analysis temporary directory.

Merge BAM

This phase accepts a set of mini-BAM files and outputs a set of BAM files. These BAM files are stored in the analysis output directory (which is typically a network-attached storage device).

One BAM file is generated per read-set. For a 96-barcoded dataset in a single XSQ file, either 96 or 97 BAM files are created. The 97th BAM contains reads that have the default barcode index.

Quality alignments and unique alignments

We strongly recommend using mapping quality values, rather than uniqueness, as the standard for recognizing or filtering good alignments. A mapping quality value estimates the probability that the alignment is correct, and takes several indications into account, including the number of alignments or uniqueness of an alignment.

The parameter `bamgen.mqv.threshold`, new in release 2.0, is available to filter alignments written to your BAM output files. In order to limit your BAM files to good quality reads, set `bamgen.mqv.threshold` to a positive integer such as 8 or higher. This technique ensures that you do not see poor quality reads in your BAM file, and that your tertiary analysis is limited to good quality reads.

How a quality value is derived

The following factors increase confidence and increase an alignment's quality value:

- Longer alignment
- Fewer possible alignments
- Fewer mismatches within the alignment

These factors reduce confidence and reduce an alignment's quality value:

- Shorter alignment
- More possible alignments
- More mismatches within the alignment

How a quality value is interpreted

For any given alignment, a quality value is a measure of the confidence of a read aligning to the particular location, given all possible alignments for the read. Quality values are within the range 0–100:

- **0 –** The highest probability of error
- **100 –** The lowest probability of error

Quality values are Phred-scaled quality scores, logarithmically linked to error probabilities. A mapping QV of 10 means that we calculate there is a 90% chance that the alignment is correct. A mapping QV of 20 means that we calculate there is a 99% chance that the alignment is correct.

Unique alignments

Unique alignments might be confused with high quality alignments, but uniqueness is not by itself a good criteria to judge an alignment. Uniqueness only means that one step of secondary mapping, the aligner, reports one alignment. This information is a part of the total information generated during mapping, but does not contain all of the information known about the alignment. In contrast, the mapping quality value is a numerical representation of all considerations about the alignment.

For example, a read might have multiple alignments. If the quality value for only one of those multiple alignments is high, that probability of that alignment being good is high. That alignment should be used in your analysis, even though it is not unique.

For a unique alignment, its quality value must be examined to determine if the alignment should be taken or filtered out. Its uniqueness by itself is irrelevant to the decision about whether it is a good read.

Small RNA mapping parameters

LifeScope™ Software sets reference-related parameters for you when you use the `set reference` command with a recognized assembly name and also have the reference files properly added to the reference repository. Table 123 lists miRBase parameters you must either set manually or cause to be set with the `set reference` command.

Table 123 Small RNA miRBase parameters

analysis.mirbase.mature.file	The file containing mature form microRNA database annotations. This file is a tab-delimited text file version of miRBase's mirRNA.xls (*not* miRBase's mirRNA.txt).

Table 123 Small RNA miRBase parameters

analysis.mirbase.precursor.file	The file containing precursor microRNA database annotations. Set this parameter to the miRBase GFF file for your organism of choice.

Table 124 lists small RNA mapping parameters.

Table 124 Small RNA mapping parameters

Parameter name	Default value	Description
Mandatory parameter		
smallrnamapping.run	1	Whether or not to run the small RNA mapping module. Allowed values: • **0**: Do not run the small RNA mapping module. • **1**: Run the small RNA mapping module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
Optional parallelization parameters		
fragmap.number.of.nodes	4	The number of compute nodes available for this analysis.
fragmap.max.number.of.jobs	100	The maximum number of jobs allowed for mapping. Allowed values: Integers 24–1000.
mapping.np.per.node	8	The number of processors per node use for mapping. The reads are divided into the number of chunks specified for this parameter.
fragmap.minreads.per.node	4000000	If the total number of fragments exceeds this amount, then the analysis is distributed across the available nodes. Allowed values: Integers 1–125000000 (125 Million).
fragmap.maxreads.per.node	150000000	The maximum number of fragments that can be processed on a single node. Allowed values: Integers 1–150000000 (150 Million).
Optional resource parameters		
wall.time	120	Total time for the process to complete.
java.heap.space	1500	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.

Table 124 Small RNA mapping parameters *(continued)*

Parameter name	Default value	Description
mapping.memory	15gb	The total memory, in gigabytes, that is available for map reads, per compute node. Include the units gb in the setting. This number is set to 15 GB because the minimum hardware requirement for memory is 16 GB. **Note:** Typically, the job scheduler allocates on nodes that have the requested memory available. For mixed hardware, it is suggested that different queues are created based on the memory available. Mapping jobs should be launched on high memory systems. For human references (hg18, hg19) and a 25.2 scheme, 45 GB is recommended for fastest performance. The smallest recommended RAM for human reference is 19 GB.
Optional mapping parameters		
smallRNA.filter.mapping	true	Whether to turn on or off the filter mapping step. Allowed values: • **false**: Turn off the filter mapping step. • **true**: Turn on the filter mapping step.
smallRNA.genome.mapping	true	(*Optional*) Whether to turn on or off the genome mapping step. Allowed values: • **false**: Turn off the genome mapping step. • **true**: Turn on the genome mapping step.
mapping.in.base	false	Whether or not to include base-space data in the mapping. Allowed values: • **false**: Do not include base-space data in mapping. • **true**: Include base-space data in mapping.
smallRNA.mirBase.mapping	true	(*Optional*) Whether or not to turn on the small RNA miRBase mapping step. Allowed values: • **false**: Turn off the small RNA miRBase mapping step. • **true**: Turn on the small RNA miRBase mapping step.
Optional BAM file generation parameters		
bamgen.refcor.addcs	true	Whether or not to add the color sequence to BAM records. If the XSQ file does not contain color space, this parameter is ignored, and the resulting BAM file output does not contain color space. See "Refcor" on page 431 for information about refcor. Allowed values: • **false**: Do not add color sequence to the output. • **true**: Add color sequence, if color space is present.

Table 124 Small RNA mapping parameters *(continued)*

Parameter name	Default value	Description
refcor.reference.weight	8	Reference weight. This parameter is used during base translation. In the read reconstruction process, multiple signals are combined to generate the final base call. This parameter adds weight (in terms of Phred score) to the signals which are compatible with reference. Color combinations that result in a variant are considered compatible with reference. Additional weight helps to eliminate base errors caused by color error(s) during base translation. Allowed values: Integers 0–100.
refcor.base.filter.qv	10	Bases with a quality value below the value of this parameter provided are replaced with 'N'. Allowed values: Integers 0–255.
bamgen.mqv.threshold	0	Provides control over the contents written to the output BAM file depending on the quality value of the alignment. To preserve only high quality alignments, set this value to a positive integer. Allowed values: Integers 0–100.
create.unmapped.bam.files	false	If set to TRUE, creates an additional BAM output file containing unmapped reads. Allowed values: • **false**: Do not create the unmapped reads output file. • **true**: Create a BAM output file containing the unmapped reads.

Mapping performance

Mapping speed depends on hardware properties as well as on the scattering logic. Mapping runs are split into multiple jobs for processing efficiency. The following parameters affect how a mapping run is split:

- **`fragmap.number.of.nodes`** – Specifies the number of jobs that are created, only if a split is necessary. The size of each split job is approximately the total number of reads divided by the number of jobs.
- **`fragmap.max.number.of.jobs`** – Determines the maximum number of jobs that can be launched per analysis. Setting this parameter to a very large value can cause the scheduler to behave incorrectly.
- **`fragmap.minreads.per.node`** – Determines the threshold of beads beyond which splitting occurs.
- **`fragmap.maxreads.per.node`** – Determines the largest number of beads that can be mapped per node. The default is based on the minimum scratch space requirements. If you scratch space is smaller or larger, this number can be made smaller or larger respectively.
- **`processors.per.node`** – This is the number of cores available for mapping analysis on each node.
- **`mapping.memory`** – For human mapping and 25.2.0 scheme, there are 3 schema lines. Simultaneous handling of schema lines reduces I/O and therefore improves mapping speed.

Memory requirements for mapping jobs are the following:

- Reference: ~6 GB
- 1 schema line: ~13 GB
- 1 schema line and reference: ~19 GB
- 2 schema lines and reference: ~32 GB
- 3 schema lines and reference: ~45 GB

Below 19 GB, the reference is split up and I/O increases significantly. Pre-built hash tables are not used for memory less than 24 GB. LifeScope™ Software supports the reuse of hash tables for human genome reference files (hg18 and hg19) and the default mapping schemes 35.2, 25.2, and 20.1. Pre-calculated color-space hash tables are included in the reference directory available with LifeScope™ Software. If system RAM is 24 GB or higher, the hash tables are loaded into memory. Depending on the pipeline, this hash table reuse typically saves 2–3 hours in analysis processing time.

For genomes without pre-calculated hash tables in the reference repository, the mapping module creates hash tables but does not save them for reuse.

In general, set the `mapping.memory` parameter to be 1 GB less than the total system memory (RAM) available.

Mapping only scales when the number of reads that are mapped together is greater than 100 M. Splitting such that the total number of fragments per node is significantly less than 100 M is not recommended. The splitting can be explained using the following pseudo-code.

```
if (num.reads <= fragmap.minreads.per.node ) { num.jobs = 1}
reads.per.job = num.reads / fragmap.number.of.nodes
if (reads.per.job > fragmap.maxreads.per.node) {
    reads.per.job = fragmap.maxreads.per.node}
num.jobs = celing (num.reads / reads.per.job)
if (num.jobs > fragmap.max.number.of.jobs ) {
    throw exception; }
```

In above pseudo-code, the input read-sets (possibly from multiple XSQ files) have `num.reads` that fall in one read length category. `Num.jobs` is the total number of jobs launched for mapping analysis of `num.reads` beads.

In general, the mapping parameters allow mixing and matching single-anchor and multiple-anchor schemes, and also mixing and matching different seed lengths and different numbers of mismatches. For example, an unmapped scheme of 25.2.0, 35.2.0:15:25, 20.1.0, with a repetitive scheme of 35.2.0:30, 45.2.0, 60.2.0, is supported, but not recommended. The impact on performance can be significant.

Internal parameters

Table 125 lists small RNA mapping parameters that we do not recommend changing.

Table 125 Small RNA internal mapping parameters

Parameter name	Default value	Description
Mapping parameters		
smallRNA.adaptor.sequence	CGCCTTGGCCGTACAGCAG	(*Optional*) The small RNA adaptor sequence. Used by mapreads in the extension step to identify the adaptor in the read.

Table 125 Small RNA internal mapping parameters *(continued)*

Parameter name	Default value	Description
smallRNA.filter.mapping.max.hits	10	The maximum number of hits for a read in the filter mapping step. Allowed values: Integers.
smallRNA.filter.mapping.mismatch.penalty	–2.0	Specifies a negative scoring penalty for mismatch that is used in local alignment mode. Setting this value to a negative number greater than the read lengths has the effect of disallowing mismatches. Allowed values: Floats.
smallRNA.filter.mapping.scheme	20.1.0	The mapping scheme for the filter mapping step. The values are: the size of the seed, number of mismatches allowed in the seed, and the start position of the seed in the read. See "Mapping schemes" on page 439 for information about mapping scheme notation.
smallRNA.genome.mapping.max.hits	30	(*Optional*) The maximum number of hits allowed for a read in the genome mapping step. Allowed values: Integers <= 0.
smallRNA.genome.mapping.mismatch.penalty	–2.0	Specifies a negative scoring penalty for a mismatch. Used in local alignment mode. Allowed values: Floats <= 0.0. If the absolute value of this penalty is set to greater than the read length, effectively no mismatches are allowed on this pass.
smallRNA.genome.mapping.scheme	20.1.0	(*Optional*) The mapping scheme for the filter mapping step. The values are: the size of the seed, number of mismatches allowed in the seed, and the start position of the seed in the read. See "Mapping schemes" on page 439 for information about mapping scheme notation.
smallRNA.mapping.fragment.tag	F3	Small RNA mapping fragment tag name.
smallRNA.mapping.schemas.file	—	Location of the schemas file for small RNA mapping.
smallRNA.mirBase.mapping.max.hits	30	(*Optional*) The maximum number of hits per read in the miRBase mapping step.
smallRNA.mirBase.mapping.mismatch.penalty	–2.0	Specifies a negative scoring penalty for a mismatch. Used in local alignment mode. Allowed values: Floats <= 0.0. If the absolute value of this penalty is set to greater than the read length, effectively no mismatches are allowed on this pass.
smallRNA.mirBase.mapping.scheme	18.2.0	(*Optional*) Mapping scheme for the miRBase mapping step. The values are: the size of the seed, number of mismatches allowed in the seed, and the start position of the seed in the read.
smallRNA.valid.adjacent.mismatch	false	(*Optional*) Indicates how to count adjacent mismatches. Allowed values: • **false**: Treat two valid adjacent mismatches as two. • **true**: Treat two valid adjacent mismatches as one. Required if smallRNA.mirBase.mapping is set to true.

Table 125 Small RNA internal mapping parameters *(continued)*

Parameter name	Default value	Description
task.scratch.dir.delete	true	Whether to keep the scratch files generated during small RNA. Allowed values: • **false**: Do not keep the scratch files. • **true**: Keep the scratch files.
analysis.clean.temp.files	1	Whether or not to keep temporary files generated during small RNA mapping. Allowed values are: • **0**: Do not keep temporary files. • **1**: Keep the temporary files.
BAM file generation parameters		
bamgen.primary.align.only	true	Whether only primary alignments or all the alignments are reported in the BAM files. Allowed values: • **false**: Report all alignments. • **true**: Report only primary alignments (both gapped and ungapped).
bamgen.refcor.softclip	0	Whether to try partial recovering of hard-clipped regions. For cases when the unaligned portion of a read needs to be presented in the BAM record, we use soft clipping in the CIGAR string. If a base-space sequence generated by ECC decoder is available, we use that sequence to represent corresponding part of the read. If the ECC decoded base-space is not available, then we naively translate using 2BE and/or +4BE evidence (if available). The soft clipped part of the read can be permanently clipped if either of these conditions is true: • The ECC base QV is less than 4. • The ECC base and 2+4 naive propagation have less than 50% similarity. Allowed values: • **false**: Do not try partial recovery. • **true**: Attempt partial recovery.

Mapping schemes

The mapping scheme parameters listed in Table 110 on page 385 determine the behavior of the mapping module. The LifeScope™ Software mapping module is based on seed-and-extend mapping, where the initial alignment step locates short matches between a read and the reference sequence. The seed specifies the length of the attempted match. With the seed-and-extend approach, a mapping scheme parameter value such as 25.2.0 is interpreted as follows:

- **First value** – The seed length (25).
- **Second value** – The quantity of allowed mismatches in the seed (2).
- **Third value** – The start site of the seed within the read (0).

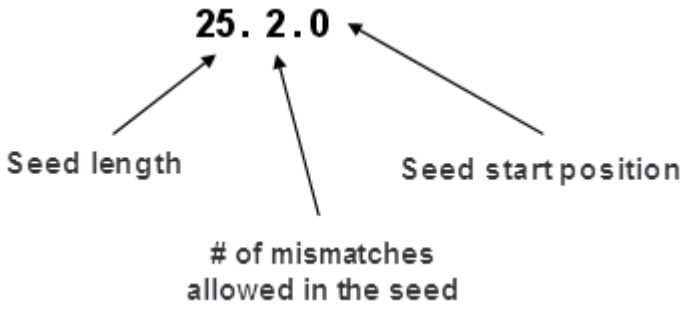


Figure 45 Seeds for local alignments

For example, with a 25.2.0 seed extend scheme, the mapping module examines the first 25 bases of a 50 bp read, and if that portion aligns to the reference with two or fewer mismatches, then the mapping module attempts seed extension. Typically, because small RNA reads are short, the seed lengths also are short, for example 18 bp for miRBase and 20 bp for genomic mapping. The extension step involves identification of the adaptor sequence. For a hit of length 35, once a hit is found, say of length 18, the extension part is evaluated for best match for adaptor starting at 19 and ending at 35 or adaptor starting at 20 and ending at 35 and so forth.

Small RNA mapping output files

The small RNA mapping module generates two BAM files containing alignments in coordinate order:

- A filtered BAM file, from the filter mapping step.
- A mapped BAM file with genomic locations. This content is merged output from the miRBase mapping and genomic mapping steps.

For information about the BAM file format, see Appendix C, "File Format Descriptions" on page 493.

The number of BAM files generated depends on the input XSQ files. One BAM file is generated per read-set. So if the input data is an XSQ file containing 96 barcodes, then the mapping module generates at least 96 output BAM files. Beads that are unclassified in any barcode are output into a separate additional BAM file. If the parameter `smallRNA.create.unmapped.bam.files` is set to `TRUE`, then an additional 96 or 97 output BAM files corresponding to the unmapped reads are also generated.

The filenames for the output BAM files are created using information from the input XSQ file, including: file base name, file id, index name, and index id. The BAM files are named according to the following patterns:

- Non-indexed BAM files: *xsqname-fileID*-1.bam
- Indexed BAM files: *xsqname-fileID*-idx_*bcIndex-bcID*.bam

The fields in the filenames are:

- **xsqname** – The XSQ file base name, without the .xsq extension.
- **fileID** – The file ID for internal XSQ file tracking.
- **bcIndex** – The barcode index.

- **bcID** – The barcode identifier for internal barcode tracking.

The directory structure for mapping output is as follows:

```
outputs/${analysis.output.dir}/
      <mappingIniFileBasename>/
            <sampleName1>/bamfilename1.bam
            <sampleName2>/bamfilename1.bam
```

The directory name <*mappingIniFileBasename*> is the basename of the mapping INI file. The default is `smallrnamapping`.

The string <*sampleName**> is determined by the sample description for the particular read-set.

The small RNA mapping output files are used by the BAMStats mapping statistics module and by the small RNA tertiary analysis modules.

Mapping statistics

Mapping statistics occur after mapping as an optional post-processing step named BAMStats. BAMStats accepts the output of the mapping step and generates statistics files to provide an in-depth understanding of the experimental data and to better detect the presence of anomalies. LifeScope™ Software shell users can display the mapping statistics output data as a chart with a spreadsheet program or other third-party program. A subset of the output from mapping can be visualized in the LifeScope™ Software UI as a series of line and bar charts, if the analysis is run in the projects repository.

Note: BAMStats does not accept 1.x BAM files as input. 1.3 BAM files must be converted to 2.0 BAMs before being used with BAMStats.

Mapping statistics parameters

Table 126 lists the parameters which control the BAMStats output.

Table 126 BAMStats parameter description

Parameter name	Default value	Description
Optional parameters		
bamstats.maximum.coverage	10000	Defines the maximum coverage allowed for locations in the reference. Locations with coverage more than the maximum coverage value are ignored during coverage calculations. Allowed values: Integers >= 1.
bamstats.maximum.isize	100000	The maximum inset size for LMP and PE libraries. Reads with an insert size more then the specified value are ignored for Insert Range Report calculation. Allowed values: Integers 1–100000.

Table 126 BAMStats parameter description *(continued)*

Parameter name	Default value	Description
bamstats.wig.primary.only	1	Use only primary alignments for coverage in WIG file format. Allowed values: • **0**: Do not restrict coverage in WIG file format to only primary alignments. • **1**: Restrict coverage in WIG file format to only primary alignments.
bamstats.wig.binsize	100	The bin size for coverage in WIG file format. Allowed values: Integers 1–100000.
bamstats.bin.isize	100	The bin size for insert range distribution. Allowed values: Integers 1–100000.
bamstats.wig.combined.report.both.strands	0	Whether to combine data from both the strands for coverage in WIG format. Allowed values: • **0**: Do not combine data. • **1**: Combine data from both the strands for coverage in WIG format.
Resource parameters		
wall.time	120	Total time for the process to complete.
java.heap.space	13000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.
number.of.nodes	4	The number of compute nodes available for this analysis.
memory.request	14gb	Memory request. Include the units gb in the setting.

Table 127 lists BAMStats parameters that we do not recommend changing.

Table 127 BAMStats internal parameter description

Parameter name	Default value	Description
bamstats.run	1	Whether or not to run the BAMStats module. Allowed values: • **0**: Do not run the BAMStats module. BAMStats statistics are not generated. • **1**: Run the BAMStats module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
bamstats.group.stats	1	Enable combined statistics covering all read-sets in the group or sample. Allowed values: • **0**: Only statistics for BAM files are generated. • **1**: Also generate combined statistics for the group or sample.
bamstats.maximum.mismatches	100	The maximum mismatches allowed in the alignments. Any alignment with more then the specified number of mismatches is ignored while generating reports related to number of mismatches. Allowed values: Integers 0–100.

Table 127 BAMStats internal parameter description *(continued)*

Parameter name	Default value	Description
bamstats.maximum.baseqv	100	Max base quality values. Any base with base quality value more then the specified value is ignored while generating reports. Allowed values: Integers 0–100.
bamstats.maximum.mappingqv	255	Maximum mapping quality value. Any alignment with mapping quality value more than the specified value is ignored. Allowed values: Integers 0–100.
bamstats.wig.minimum.mappingqv	2	Defines the minimum mapping quality allowed for coverage in WIG format. Any alignment with a mapping quality value less than this value is ignored. Allowed values: Integers 0–100.
bamstats.enable.probe.position	0	Enable probe and position error reports. Allowed values: • **0**: Do not generate probe and position error reports. Off is recommended for small RNA. • **1**: Also generate probe and position error reports.

Summary of mapping statistics output

File formats

For every input BAM file, a set of statistics files are generated. These files are in CHT, CSV, TXT, and WIG formats. Each CHT file corresponds to one displayed chart. A CHT file specifies the type of chart, the displayed range of each axis, and the data points, without using external references.

The CHT file format is an internal file format based on the CSV file format, with addition header information included. CHT header information is the following:

```
# name:
# type: scatter2d | pie | vbar | line
# title:
# xaxisname:
# yaxisname:
# xrange: <min>:<tickinterval>:<max>
# yrange: <min>:<tickinterval>:<max>
XAXISNAME, SERIES1NAME, SERIES2NAME, ...
x1, y1.1, y1.2, ...
x2, y2.1, y2.2, ...
x3, y3.1, y3.2, ...
```

The wiggle format (.wig) is a public format typically used for coverage. Visit their site for more information:

hgdownload.cse.ucsc.edu/goldenPath/help/wiggle.html

A genome browser such as the Integrative Genomics Viewer (IGV) can be used to visualize the coverage. For information is available from their site:

www.broadinstitute.org/igv/

For a collection of input BAM files that belong to a sample, a set of cumulative statistics files are generated. The cumulative statistics files are also in CHT, CSV, TXT, and WIG formats. The cumulative statistics can be visualized in the LifeScope™ Software UI.

Directory structure

The output of the BAMStats module has the following directory structure:

```
bamstats/
      <sampleName1>/*.cht, *.tbl
      <sampleName1>/<bam1>/.*cht
      <sampleName1>/<bam1>/misc/*.csv, *.txt, *.wig
```

Overview

BAMStats generates a comma-separated summary file (*BAMfilename*-summary.tbl) that summarizes key mapping quality statistics per input BAM file. The summary report contains a snapshot of statistics in all BAM files present in this sample. This file contains one row for each input BAM file in the sample. The summary file is displayed in the LifeScope™ Software UI.

`<sampleName1>/`*`BAMfilename`*`-summary.tbl`

Following directories and reports contain the cumulative statistics from all BAM files that belong to this sample. The `Misc` folder is not displayed in UI.

```
<sampleName1>/*.cht
<sampleName1>/Misc/*.csv, *.wig, *.txt
```

Following statistics are generated per BAM file in the mapping directory. These reports are not displayed in the UI.

```
<sampleName1>/<BAMfilename>/.*cht
<sampleName1>/<BAMfilename>/Misc/*.csv, *.txt, *.wig
```

In a sample, some of the BAM files possibly represent unhealthy DNA or RNA, causing the cumulative statistics to look poor. The summary tbl file is the unified location for examining the quality of data of all BAM files in the sample.

If the data for a particular BAM file is not as expected, look at the directory-level reports for that BAM file, for details.

Mapping statistics output files

This section describes the mapping statistics files generated by the BAMStats module. The output files generated by BAMStats include the name of the BAM file. The file name pattern is:

BAMFileName-fileID-barcodeID.StatisticsReportName.tag.extension

This string, without the extension, is referred to by *prefix* in Table 128, the mapping statistics output table. For small RNA reports, *tag* is F3,

Table 128 Mapping statistics output files

Report	Description, axis information, filename
Alignment Length Distribution	
	A bar plot giving the distribution of alignment lengths found in various tags used in alignment. The report is separated by tag type to provide visibility into the accuracy of individual tags. Only the primary alignment for each bead is considered in calculating the distribution.
	Y axis: Frequency X axis: Alignment length (from 0 to the maximum read length)
	Output file name: *prefix*.Alignment.Length.Distribution.*tag*.cht

Table 128 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
Base Mismatch Distribution	
	A bar plot giving the distribution of total number of mismatches found in various tags used in alignment. The bins are summed over all alignment lengths. Only the primary alignment for each bead is considered in calculating the distribution.
	Y axis: Frequency X axis: Number of mismatches (from 0 to the maximum mismatches allowed)
	Output file name: *prefix*.BaseMismatch.Distribution.*tag*.cht
Distribution of Alignment Length and Number of Mismatches in Tags	
	A report providing a simultaneous picture of the alignment length and number of mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. **Note:** This report is located in the `Misc` folder, and is not displayed in the UI.
	Y axis: Alignment length (from 0 to the max read length) X axis: Number of mismatches (from 0 to the maximum mismatches allowed)
	Output file name: *prefix*.AlignmentLength.Mismatch.*tag*.csv
Distribution of Mismatches by Position	
	A bar plot providing a distribution of errors (mismatches to reference) by position within the read. Only the primary alignments for each bead are used to generate this distribution.
	Y axis: Frequency X axis: Position (from 0 to maximum read length)
	Output file name: *prefix*.Mismatches.By.Position.BaseSpace.*tag*.cht
Color Mismatch Distribution	
	A bar plot providing a distribution of the total number of color mismatches in various tags used in alignment. Only the primary alignments for each bead are used to generate this distribution. This report is only generated if mapping is done in color space.
	Y axis: Frequency X axis: Number of mismatches (from 0 to the maximum mismatches allowed)
	Output file name: *prefix*.ColorMismatches.Distribution.*tag*.cht
Distribution of Alignment Length and Color Mismatches	
	A report providing a simultaneous picture of the alignment length and number of color mismatches distributions in various tags used in alignment. Only primary alignments for each tag are considered in calculating this distribution. The bins range from 0 to the maximum read length, and from 0 to the maximum number of mismatches allowed. This report is only generated if mapping is done in color space. **Note:** This report is located in the `Misc` folder, and is not displayed in the UI.
	Y axis: Alignment length (from 0 to the max read length) X axis: Number of mismatches (from 0 to the maximum mismatches allowed) Z axis: Frequency
	Output file name: *prefix*.AlignmentLength.ColorMismatches.*tag*.csv
Base QV Distribution	

Table 128 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
	A bar plot providing a distribution of base quality values generated using the reference assisted error correction/base conversion algorithm. The base QV distributions are separated for each tag as quality of individual tags could be very different.
	Y axis: Frequency X axis: Base QV (from 0 to the maximum QV)
	Output file name: *prefix*.BaseQV.cht
Base QVs by Position	
	A bar plot providing a distribution of base quality values by individual base positions. This report identifies if certain base positions (particularly towards the end of the read) have poor base quality values. All the tags (F3/R3/F5-P2) used in the alignment are combined into a single bin.
	Y axis: Base QV X axis: Base position (from 0 to read length)
	Output file name: *prefix*.BaseQV.by.Position.csv
Distribution of Mismatches by Base QV	
	A line plot providing a distribution of errors (mismatches to reference) by base quality values bins.
	Y axis: Percentage of total mismatches X axis: Base QV (from 0 to maximum QV) This report gives, for each QV, a measure of the raw mismatches at positions with that QV. This information does not measure whether the base QVs generated are well calibrated to the probability of error in that particular base position.
	Output file name: *prefix*.Mismatches.By.BaseQV.*tag*.cht
Distribution by Mapping QVs	
	A bar plot providing a distribution of mapping quality values for the F3 tag. Only the primary alignment for each bead is used in calculating this distribution.
	Y axis: Frequency X axis: Position (from 0 to maximum mapping QV)
	Output file name: *prefix*.MappingQV.*tag*.cht
Coverage Report	
	A line plot providing a distribution of coverage obtained after mapping/pairing. Only the primary alignment for each bead is used in this calculation.
	Y axis: Frequency X axis: Coverage (from 0 to number of reads)
	Output file name: *prefix*.Coverage.*tag*.cht
Coverage Report by Chromosome (Contig) and Base Windows	
	A line plot providing a distribution of coverage within each reference window. The coverage is calculated within each window along a reference chromosome. The window size can theoretically be anywhere from a single base to the contig length. However the calculations of base level coverage are computationally expensive and less interpretable as the window size increases. Only the primary alignment for each bead is used in this calculation.
	Y axis: Frequency X axis: Coverage (from 0 to number of reads)
	Output file name: *prefix*.Coverage.By.Chromosome.contig*nn*.cht

Table 128 Mapping statistics output files *(continued)*

Report	Description, axis information, filename
Coverage by Strand	
	A line plot providing the distribution of coverage within each reference window separated by reference strand (+/-). Only the primary alignment for each bead is used in this calculation.
	Y axis: Frequency X axis: Coverage (from 0 to number of reads)
	Output file name: *prefix*.Coverage.By.Strand.*tag*.cht
Coverage files	
	Coverage reports in wiggle format. For each genome position, reports the number of reads that cover (map to or span) the position. Because reporting coverage for each position results in very large files, coverage is reported for bins, with each bin spanning a user-defined number of bases. For each bin, the mean coverage of all the positions in that bin is reported. The parameter `bamstats.wig.binsize` controls the size of the bins in this file. Each coverage file includes a header as the first line. The header lines follow this pattern: track type=wiggle_0 name=<*chrname*> description=<*coverage from positive/negative/both strand*> visibility=full color=0,0,255 fixedStep chrom=<*chrname*> start=<*startpos*> step=<*binsize*> span=<*binsize*>
	Output file name: coverage_chr*nn*.POS.wig, coverage_chr*nn*.NEG.wig
Unique Start Positions	
	This report lists the following: • The total number of unique start positions. • The average number of unique start positions per reference position. See also "Unique Start Position" on page 448.
	Output file name: *BAMfilename*-summary.txt, *Samplename*.-summary.txt
Summary Report	
	The summary file provides statistics for each BAM file in the sample. See "Summary file".
	Output file name: *filename*.-summary.tbl

Mapping statistics example output

This section provides example output of some mapping statistics output files.

Summary file

The summary file provides statistics for each BAM file in the sample. This list describes labels used in the summary file:

- **NumFragmentsTotal** – The total number of beads, before any filtering on the instrument. This value is also the fragment count in the input XSQ file.
- **NumFragmentsPassingFilters** – The number of beads that pass filtering on the instrument.
- **NumMapped** – The number of reads with primary alignment.
- **% filtered mapped** – The number of primary reads divided by the number of beads passing instrument filtering (`NumMapped` / `NumFragmentsPassingFilters`).
- **% total mapped** – The number of primary reads divided by the number of total beads (`NumMapped` / `NumFragmentsTotal`).

- **Megabases of coverage** – The sum of alignment lengths, for primary alignments only.

The number of beads filtered out in the instrument is found by subtracting the number of beads that pass filtering from the total number of beads:

```
NumFragmentsTotal – NumFragmentsPassingFilters
```

The following are column headings for a summary file.

```
BamFileName,IsColorInBam,IsBaseInXSQ,IsECC,LibraryType,
ReadLength,Megabases of Coverage,
NumFragmentsPassingFilters,NumFragmentsTotal,
Tag1-NumMapped,
Tag1- % total Mapped,
Tag1- % filtered mapped,
(Tag1-AlignmentLength;Min;Max;Avg;Median;StdDev),
(Tag1-NumBaseMismatches;Min;Max;Avg;Median;StdDev),
(Tag1-MappingQV;Min;Max;Avg;Median;StdDev),
(Tag1-BaseQV;Min;Max;Avg;Median;StdDev),
(Tag1-NumColorMismatches;Min;Max;Avg;Median;StdDev),
(Coverage;Min;Max;Avg;Median;StdDev)
```

Unique Start Position

The following is an example of a Unique Start Position report, for fragment data:

```
Unique Start Positions F3
Total Number of Unique Start Positions: 16536 (16.043154 %
of primary alignments)
Average Number of Unique Start Positions per Reference
Position: 0.001630
```

Unique Start Position reports have the following naming patterns:

- *BAMfilename*-summary.txt
- *Samplename*-summary.txt

Coverage files

Example contents for the file `coverage_chr1_positive.wig` are:

```
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1,
alignmentFilteringMode=PRIMARY, filterOrpha-nedMates=false,
track name="BAM Coverage positive strand" description="BAM
Coverage positive strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
336171
336181
336191
336201
```

```
336211
336221
336231
336241
336251
336261
336271
…
```

Example contents for the file `coverage_chr1_negative.wig` are:

```
browser position chr1:1-200000000
browser hide all
browser pack refGene encodeRegions
# minimumMapq=25, minimumCoverage=1,
alignmentFilteringMode=PRIMARY, filterOrpha-nedMates=false,
track name="BAM Coverage negative strand" description="BAM
Coverage negative strand" visibility="full color 0,0,255"
priority=10 yLineMark=0 type=wiggle_0 yLineOnOff=on
variableStep chrom=chr1 span=1
57432
57442
57452
57463
57473
57483
57493
57503
57513
57523
57533
…
```

Run BAMStats standalone

This section describes requirements to generate mapping statistics outside of the context of the mapping module. These requirements are:

- **Directory structure** – The input directory for BAMStats module must mimic the output directory structure of mapping:

  ```
  Input.BAMStat.dir/
      <sampleName1>/*.bam, *.bai
      <sampleName2>/*.bam, *.bai
  ```

- **Index files** – For every BAM file, there must be a corresponding BAI-formatted indexed file. BAI files can be generated using the following command:

  ```
  samtools index <bamfile>
  ```

  This command generates index sorted alignment for fast random access. The index file `<bamfile>.bai` is created. For details, please refer to the samtools site:

http://samtools.sourceforge.net/samtools.shtml

- **Write permission** – The user must also have write permission for the input directory, because the BAMStats module generates position and probe error files in that directory.

FAQ - Small RNA mapping

1

How does the local alignment approach affect mapping?

Using the local alignment approach removes the constraint of a whole-read alignment, and mapping rate is significantly increased. It is not uncommon for a poor dataset with a 30% mapping rate at 50_6, to reach more than a 60% mapping rate with the mapping algorithm used in LifeScope™ Software.

While the majority of alignments are full length, some alignments can vary in length. If a read has many errors towards the end, the final alignment will not include these positions if a shorter alignment receives a better overall score.

2

How do I increase mapping rate?

Mapping rate increases with each additional round of mapping. However, the gain in rate comes at the cost of increased runtime and disk requirement. Testing has found sets of mapping schemes that strike a good balance between mapping rate and runtime for 25-, 35,- and 50-bp reads. These schemes are shipped as LifeScope™ Software defaults and should work well for most purposes.

For 50-bp reads, two mapping parameters keys define how many rounds of mapping are run, and what happens in each round. The default values of the keys are shown below, and their meanings are further explained in Table 9 on page 76.

Note: Repetitive schemes are empty by default.

- mapping.scheme.unmapped.50 = 25.2.0,25.2.15
- mapping.scheme.repetitive.50 = 38.3.0,25.2.0

The repetitive scheme does not affect the number of mapped reads, and setting it only increases the chance of finding better alignments for repetitive reads. If your main goal is to improve throughput, first try to add more iterations at the unmapped scheme. For example, add another step of 25.2.25 at the end of the unmapped scheme above.

3

What is a good seed for my application?

Consider the following factors when picking seed parameters:

- Mapping is slower when more mismatches are allowed in the seed. However, allowing more mismatches improves the mapping rate.
- Shorter seeds have higher mapping rates, but also lead to more spurious alignments.
- Color-calls at the beginning of a read are more reliable than those at the end. Therefore, you have the option to anchor the seed near the front of the read. However, applications such as transcriptome sequencing are scenarios where it is an advantage to anchor the seed near the end of the read.

For 50-bp reads, the following default settings are recommended:

- 18.2.0 for Sanger alignment
- 20.1.0 for genome alignment

4

What reads scenario achieves the best accuracy?

The best performance is achieved by considering 2BE and 4BE reads, their alignment, and reference. Typically, base-space reads generated by ECC have lower mapping throughput. Base-space reads generated from combined 2BE and 4BE calls and no reference have lower accuracy, compared to ECC calls.

5

How do I interpret a mapping quality value?

First of all, it is worth noting that the mapping quality value (QV) is not like the p-value in BLAST. In BLAST, the p-value is used to estimate the likelihood that the aligned sequences are related. In general, reads come from the same source and it is known that the two are related.

The purpose of mapping QV is to estimate the probability that the read originates from the mapped genomic location. The estimate is determined mainly by the difference in significance between the best- and second-best hits.

The numeric interpretation of mapping QV is the same as the base call QV. A mapping QV of 10 means that there is a 90% chance that the alignment is correct. A mapping QV of 20 means that there is a 99% chance that the alignment is correct.

6

How much RAM do I need to analyze human samples?

For optimal performance, 24 GB of RAM per cluster node is recommended for human samples. If the cluster node has less than 24 GB of available RAM, mapreads can split the genome into smaller segments, and align to each segment sequentially. Splitting the genome into smaller segments is implemented inside mapreads and is completely transparent to the user.

Mapping with less than the recommended amount of RAM slows down performance. If the cluster node has 16 GB of RAM, mapping human samples will be roughly 30% slower compared to a machine with 24 GB of RAM.

7

In the mapping statistics genome coverage calculation report, how are Ns in the reference counted?

Ns in the reference are counted as missing coverage. For example, in the human genome, about 7% of the reference sequence consists of Ns. This means that the coverage calculation will never be reported as higher than 93%. If the frequency distribution says 7.49% of the genome is uncovered, this means that about 0.34% of the non-N reference sequence does not have reads mapped to it, and that also the 7.15% of the genome that consists of N does not have reads mapped to it.

8

What should I do to ensure balanced allele ratios at heterozygous positions?

The base-translation algorithm, refcor, uses instrument color-space data (ECC or non-ECC) and the reference sequence to produce the most likely base calls, along with a Phred-scale quality value. The degree to which the reference influences the base call is controlled by the `refcor.reference.weight` parameter. A position is converted to reference only if the `refcor.reference.weight` setting is higher than the difference between the quality values of a correct color call and of an adjacent erroneous color call that supports reference.

The `refcor.reference.weight` default setting, 8, is chosen to be low enough to ensure balanced allele ratios, and also be large enough both to distinguish correct from incorrect color calls and to maximize base accuracy.

In the event an incorrect reference base call is made, the disagreement between evidence sources causes that call's QV to be low. Incorrect calls can be filtered out based on QV threshold, and this filtering improves allele ratios.

Another way of filtering low quality base-calls is to adjust the parameter `refcor.base.filter.qv`, that converts all base calls with a quality value less than `refcor.base.filter.qv` into Ns. This filtering has the effect of setting QV to zero in the calls with low confidence. To decrease the number of N base calls, reduce this threshold (for example, from the default of 10 to 5).

Note: Setting the `refcor.reference.weight` parameter to zero completely eliminates the reference as a guide for base-translation (completely eliminates reference bias), but also considerably reduces the quality of base-translation.

9

Can I run BAMStats (mapping statistics) on filtered BAM files?

Yes. You can run BAMStats on the filtered BAM files using the command shell. This scenario is not supported in the LifeScope™ Software UI.

These instructions describe creating your own version of the small RNA workflow, and modifying your version to include BAMStats on the filtered BAM file.

1. Follow the instructions in "Create a new workflow" on page 121 to create a workflow based on the `small.rna` standard workflow.
2. Create a new INI file named `filtermapping.bamstats.ini`, with these contents:

```
import global.ini

##Run parameter

bamstats.run=1

##Bin size for insert range distribution.
#bamstats.bin.isize=100

##The input directory for BamStats. There should be one
directory per sample containg the BAM files for that sample.
bamstats.input.dir=${analysis.output.dir}/bam/filtered

##Maximum Coverage of a locations in the reference. Locations
with coverage more than the maximum coverage values will be
ignored during coverage calculations.
#bamstats.maximum.coverage=10000

##Maximum inset size for LMP and PE libraries. Reads with insert
size more then the specified value will be ignored for Insert
Range Report calculation.
#bamstats.maximum.isize=100000

##The path to the output directory where BamStats will write its
.cht files.
#bamstats.output.dir= ${task.output.dir}

##Bin size for Coverage in WIG file format
#bamstats.wig.binsize=100

##Whether to combine data from both the strands for coverage in
WIG format #bamstats.wig.combined.report.both.strands=0

##Use only primary alignments for Coverage in WIG file format.
#bamstats.wig.primary.only=1

analysis.genome.reference = ${analysis.filter.reference}
```

3. Edit the file `secondary/secondary.pln` in your workflow to add the `filtermapping.bamstats` step. The contents of the revised PLN file are:

```
smallrnamapping.ini
bamstats.ini < smallrnamapping.ini
```

This PLN file runs the BAMStats module on filtered BAM files also as part of workflow.

10

What is the difference between the small RNA alignment method and the other LifeScope™ Software alignment methods?

Genome alignment algorithm is modified to accommodate reads particularities of small RNA libraries. The main difference between reads generated from DNA or RNA libraries and small RNA libraries is that only the beginning of the reads is represented in reference genome (mature forms of miRNA with variable length), followed by the P2 adaptor sequence.

Alignment (against Sanger or genome reference) comprises two steps:

1. **Global alignment –** Read seed alignment.

   Global alignment is identical to the one used in all other LifeScope™ Software alignment methods (that is, genomic resequencing and whole transcriptome mapping).

2. **Local alignment –** Search for adaptor sequence start position (within read) using seed alignment location.

   During local alignment step, we use the following to estimate the adaptor start position:

   - For every possible adaptor start position `n`, we compare the actual read with the hypothetical one obtained by concatenating `n-1` bases extracted from the genomic location of the read seed and read length `-n` bases of the adaptor sequence.
   - All adaptor start locations are tested. The adaptor start location giving the smallest number of mismatches is considered to be the beginning of the adaptor.
   - The number of mismatches for the full length of the read is used to generate the alignment confidence metric (the mapping QV).

FAQ – BAMStats

1

In the BAMStats summary table, why do the `NumFragmentsPassingFilters` and `NumFragmentsTotal` fields have the same totals?

`NumFragmentsTotal` reports the total number of fragments (read pairs or reads) passed from the instrument. `NumFragmentsPassingFilters` reports the number of fragments that pass the instrument filter. For the 5500 instrument, all reads pass filters when using color space, and so these two numbers are the same. If the run is

performed with ECC, then the base space reads have filters applied to remove those of lowest quality. For an ECC run mapped in base space using LifeScope™ Software, the filtered reads are reported in LifeScope™ Software, and the Total and Filtered numbers are different.

2

What changes are required for a job with a large number of contigs?

For a job with a large (more than a few thousand) number of contigs, change the system property `max.head.node.java.heap.space` from the default of 64 to 128, or to an intermediate value between 64 and 128.

Modify this parameter in the <*installdir*>`/etc/analysis/system.properties` file.

Because changes to system properties affect all runs of all users, you could consider changing the parameter back to its original value after the large contig run completes. Setting `max.head.node.java.heap.space` to 128 could present a problem if your head node is extremely loaded.

24 Run a Small RNA Coverage Analysis

This chapter covers:

- Overview 457
- Example of running the small RNA coverage module 457
- Small RNA coverage input files 458
- Small RNA coverage parameters 458
- Small RNA coverage output files 459

Overview

The small RNA coverage module calculates read coverage per position.

Example of running the small RNA coverage module

The `small.rna` standard workflow provides an example of how to run the small RNA coverage module. Here are example LifeScope™ Software shell commands for this workflow, which includes small RNA mapping and small RNA counts:

```
lscope.sh shell -u username -w password
cd projects
mk mirna
cd mirna
mk run1
cd run1
set workflow small.rna
add xsq rna.xsq
set reference hg18
# optionally change parameter defaults here
run
ls
exit
```

In order to change a parameter value in your analysis, use the `set param` shell command after the line `# optionally change parameter defaults here` in the example commands. For instance, to change the `RNA.coverage.min.value` parameter, use this shell command:

```
set RNA.coverage.min.value 0  secondary/srCoverage.ini
```

Small RNA coverage input files

The small RNA coverage module takes as input one or more BAM files containing mapped data. The following describe the alignment input accepted. This module:

- Accepts only fragment data for RNA libraries.
- Accepts multiple BAM files as input.
- Accepts BAM files with different read lengths.
- Accepts both color space and base-space files, and a combination of color and base-space data.

Small RNA coverage parameters

Table 129 lists the small RNA coverage parameters.

Table 129 Small RNA coverage parameter description

Parameter name	Default value	Description
Optional parameters		
RNA.coverage.min.quality	2	The mapping quality value threshold for selecting the alignments. Alignments with a mapping quality value lower than this threshold are not written to output. Allowed values: Integers 0–100.
RNA.coverage.min.value	0	The mapping quality threshold value for selecting alignments from the input BAM. In order to be included in the output, a position's coverage must be greater than or equal to this value. Allowed values: Integers >= 0.
RNA.coverage.per.chromosome	true	Whether coverage output is generated as one file per every chromosome per strand, or as a single file with coverage of all chromosomes per strand. Allowed values: • **false**: Generate a single file with coverage of all chromosomes per strand. • **true**: Generate one file per every chromosome per strand.
RNA.coverage.primary.only	true	Whether only primary alignments or all the alignments from the input BAM file are used as input. Allowed values: • **false**: Consider all alignments. • **true**: Consider only primary alignments (both gapped and ungapped).
Resource parameters		
memory.request	4gb	Memory request. Include the units gb in the setting.
java.heap.space	4000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.

Table 130 lists the small RNA coverage parameters that we do not recommend changing.

Table 130 Small RNA coverage internal parameter description

Parameter name	Default value	Description
coverage.run	1	Whether or not to run the small RNA coverage module. Allowed values: • 0: Do not run the coverage module. • 1: Run the coverage module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
analysis.input.bams	—	A comma-separated list of paths to the input BAM files. Relative and absolute paths are accepted. Generated automatically by LifeScope™ Software during a workflow.

Small RNA coverage output files

The small RNA coverage module's output files are in the wiggle (*.wig) file format. Wiggle files are visualized in genomics browsers such as the Integrative Genomics Viewer (IGV). You can download the IGV browser from the Broad Institute website:

www.broadinstitute.org/igv

The wiggle file specification is available from this site:

http://genome.ucsc.edu/goldenPath/help/wiggle.html

If the parameter `RNA.coverage.per.chromosome` is set to `false`, the small RNA coverage module generates two output files with coverage of all chromosomes per strand:

- `coverage_positive.wig`
- `coverage_negative.wig`

If the parameter `RNA.coverage.per.chromosome` is set to `true`, the small RNA coverage module generates one coverage file per chromosome per strand:

- `coverage_chr**_positive.wig`
- `coverage_chr**_negative.wig`

25 Run a Small RNA Counts Analysis

This chapter covers:

- Overview 461
- Example of running the small RNA count module 461
- Small RNA counts 462
- Small RNA counts input files 462
- Small RNA counts parameters 464
- Small RNA counts output files 465

Overview

This module generates tag counts for precursor and mature miRNA sequences. Run this analysis module to generate miRNA expression profiles.

IMPORTANT! Alignments must come from the same strand as the feature to contribute to the count. A non-gapped tag contributes to a feature's count if it overlaps the feature and has no more than three bases outside the feature. A gapped tag contributes to a feature's count if one of its match regions terminates at a feature boundary.

Example of running the small RNA count module

The `small.rna` standard workflow provides an example of how to run the small RNA counts module. Here are example LifeScope™ Software shell commands for this workflow, which includes small RNA mapping and small RNA coverage:

```
# log into the shell
lscope.sh shell -u username -w password
# cd to the projects repository
cd /projects
# create a project and open it
mk mirna
cd mirna
# create an analysis and open it
```

```
mk run1
cd run1
# define the analysis type
set workflow small.rna
# define the input data
add xsq rna.xsq
add xsq rna2.xsq
# specify the reference
set reference hg18
# optionally change parameter defaults here
# list the analysis configuration
ls
# start the analysis
run
# list progress information about the run
ls
# exit the command shell
exit
```

Dummy XSQ file name are given in the example. In order to change a parameter value in your analysis, use the `set param` shell command after the line `# optionally change parameter defaults here` in the example commands. For instance, to change the parameter `SmallRNA.counts.min.quality`, use this shell command:

```
set SmallRNA.counts.min.quality 0  secondary/srCount.ini
```

You must know which INI file to use for the `set param` command. The `ls` command, executed in an analysis virtual directory after the workflow has been defined, lists the INI files used in the analysis.

Note: The shell command `set workflow hg18` (or `hg19`) incorrectly sets the `analysis.mirbase.mature.file` parameter to README.txt. You must manually set the parameter correctly *after* the `set workflow` command.

Small RNA counts

The LifeScope™ Software small RNA counts module calculates counts per precursor or mature sequence, where counts are the number of reads mapped. This module is similar to the WT counts module, which computes counts per exon.

This module requires an annotation features file, which contains either precursor sequences information in a GFF file, or mature sequences information in a text file. The counts module uses the annotation features file and alignments in the form of one or more BAM files, and computes the number of alignments that match every feature. Non-primary and low quality alignments can be filtered by setting parameters appropriately.

Small RNA counts input files

The small RNA mapping module generates two BAM files containing alignments in coordinate order. These two BAM files are input to the small RNA counts module for separate counts processing:

- The filtered BAM file, from the filter mapping step. The small RNA counts module uses this file to provide counts for the number of reads mapped to the sequences in the filter reference. See **"Filtered BAM file parameters" on page 465** and **"Filtered counts output" on page 467**.
- One or more mapped BAM files with genomic locations. The module uses these files as input with the mapped BAM files:
  - A Genome Coordinates GFF file containing precursor sequences with genomic locations.
  - (*Optional*) A file containing mature form sequences with locations relative to precursor sequences. If the mature sequences file is not provided as input, the module computes counts per precursor sequence.

  These files are described in the following sections.

Alignments files

The following describe the alignment input accepted by the small RNA counts module. This module:

- Accepts only fragment data for RNA libraries.
- Accepts multiple BAM files as input.
- Accepts BAM files with different read lengths.
- Accepts both color space and base space files, and a combination of color-space and base-space data.

Precursor sequences

This file is a Genome Coordinates GFF file containing precursor sequences with genomic locations. You must use the same file that was used during the small RNA mapping step.

This file is included in the LifeScope™ Software reference repository and is also available the following link:

http://www.mirbase.org/ftp

Mature form sequences

The mature form sequences file provides locations relative to precursor sequences. This tab-separated file is included in the LifeScope™ Software reference repository and is also available the following link:

ftp://mirbase.org/pub/mirbase/CURRENT/miRNA.xls

If you download a new file, you must save the file locally in tab-delimited format. Only this specific mature forms file is supported.

If the mature sequences file is not provided as input, the module computes counts per precursor sequence.

Small RNA counts parameters

Mapped file parameters

Table 131 lists the small RNA counts parameters used to process the mapped BAM file.

Table 131 Small RNA counts parameter description

Parameter name	Default value	Description
SmallRNA.counts.min.quality	2	The mapping quality value threshold for selecting the alignments. Alignments with a mapping quality value lower than this threshold are not written to output. Allowed values: Integers 0–100.
SmallRNA.counts.overflow.limit	3	The maximum allowed offset on either side of an alignment or feature, when looking for overlap between an alignment and a feature. If this limit is set to 0, then the start and end coordinates of the alignment record and feature must match exactly. Allowed values: Integers 0–20.
SmallRNA.counts.per.feature	true	Controls the aggregated count output. Allowed values: • **false**: The aggregated count of all the reads mapped to a feature are output on the same line. • **true**: Reads of different starts and ends are counted separately for every feature and are output on separate lines. The primary key in the output GTF file is: {Feature, start, end}.
SmallRNA.counts.primary.only	true	Whether only primary alignments or all the alignments from the input BAM file are used as input. Allowed values: • **false**: Consider all alignments. • **true**: Consider only primary alignments (both gapped and ungapped).
Resource parameters		
memory.request	4gb	Memory request. Include the units gb in the setting.
java.heap.space	3000	Dynamic memory requirement, in megabytes. Do not enter a unit with the numeric value.

Table 132 lists the small RNA counts parameters that we do not recommend changing.

Table 132 Small RNA counts internal parameter description

Parameter name	Default value	Description
counts.run	1	Whether or not to run the small RNA counts module. Allowed values: • 0: Do not run the small RNA counts module. • 1: Run the small RNA counts module during this analysis. The run parameter is set automatically by the shell during a standard workflow. Accept the default for most use.
SmallRNA.counts.mismatches	3	The maximum number of mismatches used when computing statistics. Allowed values: Integers 0–10.
SmallRNA.matureForms.file	—	Path to the tab-separated file with mature form sequences in every precursor sequence. Relative and absolute paths are supported.

Table 132 Small RNA counts internal parameter description *(continued)*

Parameter name	Default value	Description
SmallRNA.precursor.gff.file	—	Path to the input GFF file with genomic coordinates of the precursor sequences. Relative and absolute paths are supported.

Filtered BAM file parameters

Table 133 lists the small RNA counts parameters related to the filtered BAM file.

Table 133 Small RNA filter counts parameter description

Parameter name	Default value	Description
filtermapping.counts.run	1	Whether or not to run the small RNA counts on the filtered BAM file. Allowed values: • 0: Do not run small RNA counts on the filtered BAM file. • 1: Run small RNA counts on the filtered BAM file. The run parameter is set automatically by the shell during a standard workflow.
analysis.filter.reference	—	The path to the filter reference file. This parameter is set automatically by the shell when you use the `set reference assembly` command.
SmallRNA.counts.primary.only	true	Whether only primary alignments or all the alignments from the input BAM file are used as input. Allowed values: • **false**: Consider all alignments. • **true**: Consider only primary alignments (both gapped and ungapped).

Small RNA counts output files

Mapped output

This section describes the output files the small RNA counts module creates from the mapped BAM input.

GTF files

The small RNA counts module creates a GTF file for precursor counts and one for mature counts (if mature parameter is provided). Each input GFF file contains the counts per feature in column 5. Table 134 describes the counts file fields. The names of these files are:

- `precursor_counts.gtf`
- `matureForms_counts.gtf`

The following is example content from a `counts.gtf` file:

```
chr1     .       mirBase 3044539  3044599 0  - . Precursor_ID
"hsa-mir-4251"; RPM "0.00";
```

Table 134 Small RNA counts GTF output file format description

Column name	Description	Examples
seqname	The name of the sequence. Must be a chromosome or a scaffold.	chr1
source	The program that generated this feature.	–
feature	The name of this type of feature.	*miRNA
start	The starting position of the feature in the sequence. The first base is numbered 1.	3044539
end	The ending position of the feature (inclusive).	3044599
score	The tag count for the feature.	0
strand	The strand containing the feature. Values: • '+': The feature is on the 3′ strand. • '-': The feature is on the 5′ strand. • '.': Unknown or not required.	-
frame	(Not applicable to miRNAs.)	–
attributes	A list of attributes delimited by semi-colons. Each attribute is a type/value pair, with type and value separated by a single space character. Type is a string matching the pattern [A-Za-z1-9_]+. Value is a number or a double-quoted string. Every attribute must end with a semi-colon.	ACC="MI000000": ID "hsa-mir-4251";

contig_counts.txt file

This section describes the output file, `contig_counts.txt`, generated from the mapped BAM file. This output file is a tab-separated file listing the following for each the sequence id in the filter reference:

- The sequence id
- The number of reads on the positive strand that map to that sequence
- The number of reads on the negative strand that map to that sequence
- The size of the reference sequence

The following is an example of a few lines from the mapped `contig_counts.txt` file:

```
ID      Counts_Pos_Strand    Counts_Neg_Strand   RefSeq_size
chr20   2932     1        63025520
chr21   1312     0        48129895
chr22   159      2        51304566
```

Mismatches stats files

The small RNA module generates two stats files based on the mapped BAM input:

- `precursor_mismatch_stats.txt`
- `matureForms_mismatch_stats.txt`

These files list the sequences from either the input precursor and or the mature forms file, and provide the number of alignments that map to each sequence with 0, 1, 2, and 3 mismatches. The total number of alignments that map to the sequence is also provided. The following is an example of a few lines from a `mismatch_stats.txt` file:

	0MM	1MM	2MM	3MM	Total
hsa-let-7a	48(64.00%)	15(20.00%)	3(4.00%)	5(6.67%)	75(100.00%)
hsa-let-7a*	1(16.67%)	0(0.00%)	4(66.67%)	0(0.00%)	6(100.00%)
hsa-let-7a-2*	0(0.00%)	0(0.00%)	0(0.00%)	0(0.00%)	0(-%)
hsa-let-7b	17(33.33%)	7(13.73%)	16(31.37%)	7(13.73%)	51(100.00%)
hsa-let-7b*	1(33.33%)	0(0.00%)	1(33.33%)	1(33.33%)	3(100.00%)

Filtered counts output

This section describes the output file, `contig_counts.txt`, generated from the filtered BAM file. This output file is tab-separated and lists the following for each the sequence id in the filter reference:

- The sequence id
- The number of reads on the positive strand that map to that sequence
- The number of reads on the negative strand that map to that sequence
- The size of the reference sequence

The following is an example of a few lines from the filtered `contig_counts.txt` file:

```
ID       Counts_Pos_Strand   Counts_Neg_Strand   RefSeq_size
3000072055953=Tigger4a#DNA/MER2_type   0   0   236
chr16.trna19-GlyGCC                   69   0   71
chr17.trna14-ThrCGT                    2   0   72
3000072056044=tRNA-Tyr-TAC#tRNA        0   0   76
chr2.trna27-GlyCCC                   128   0   71
```

PART VIII

Additional Mapping Analyses

26 Run a ChIP-Seq Mapping Analysis

This chapter covers:

- Overview 471
- Run ChIP-Seq mapping 471
- Run ChIP-Seq as an individual analysis 474
- Mapping algorithm 474
- Use results files 474

Overview

LifeScope™ Genomic Analysis Software provides the ability to map data and create an output file type compatible with a variety of third-party Chromatin Immunoprecipitation Sequencing (ChIP-Seq) data analysis tools. The ChIP-Seq application has publicly available analysis software that can be used with LifeScope™ Software output.

The ChIP assay is a method for analyzing epigenetic modifications and genomic DNA sequences bound to specific regulatory proteins. ChIP-Seq is a combined assay and sequencing technique for identifying and characterizing elements in protein-DNA interactions. It typically examines transcription factors (TF) bound to DNA and finds DNA sequence motifs common to binding sites.

Using the MAGnify™ ChIP-Seq kit with the SOLiD® sequencing system enables you to generate sequence read data from a ChIP-Seq experimental approach. LifeScope™ Software gives you the option to map the read data.

Run ChIP-Seq mapping

ChIP-Seq analysis is available as workflow in the LifeScope™ Software command shell.

Run in the lscope command shell

This section describes how to run the fragment ChIP-Seq mapping workflow in the LifeScope™ Software command shell.

In the LifeScope™ Software command shell, running the ChIP-Seq workflow requires the following steps:

- Create a LifeScope™ Software command shell project and analysis.
- Identify your input data (your reads).
- Identify the reference genome.
- Select the ChIP-Seq the workflow as the analysis to be executed on your data.
- Issue the `run` command to start your analysis.

Example steps to run ChIP-Seq mapping as standard workflow

This section shows the steps to run the ChIP-Seq workflow in the LifeScope™ Software command shell. A description of each step is given in Table 135.

```
# log into the shell
lscope.sh shell -u username -w password
# import your data into the reads repository
cd /reads
import /data/xsq/xsq1.xsq
# cd to the projects repository
cd /projects
# make a project, and open it
cd /projects
mk chip_seq
cd chip_seq
# make a analysis, and open it
mk run1
cd run1
# define the analysis type (the ChIP-Seq workflow)
set workflow chip.seq
# define the input
add xsq xsq1.xsq  lung
# specify the reference to be used
set reference /data/results/references/DH10B.fasta
# list the analysis configuration
ls
# run the analysis
run
# display progress information on your run
ls
```

Dummy XSQ file names are used in the example.

Information about your workflow job

Output of the `ls` command displays the analysis' parameters, INI files, and run completion percentage or completion status.

Table 135 explains the purpose of the steps shown above.

Table 135 Description of sample shell commands to run the ChIP-Seq workflow

Command	Purpose
`lscope.sh shell -u username`	Log into the LifeScope™ Software command shell.

Table 135 Description of sample shell commands to run the ChIP-Seq workflow *(continued)*

Command	Purpose
`cd /reads` `import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq`	Go into the reads repository and import an XSQ file (the input data file). This command in effect brings the reads file into the shell reads repository. Once in the reads repository, the reads file is available as input to your analysis runs.

`cd /projects`
`mk chip_seq`
`cd chip_seq`
`mk run1`
`cd run1`
	In the projects repository, create a project named `ecoli`, and in that project create an analysis named `run1`. Open the `run1` analysis.
	Note: A project name cannot include a dash character ("-").
	Note: The open analysis (`cd run1`, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.

`set workflow chip.seq`	Define the workflow used with this analysis. The ChIP-Seq workflow includes mapping, mapping statistics, and duplicate bead finding.
`add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq`	Define the XSQ file containing the input reads data for this analysis.
`set reference /data/results/results/references/DH10B.fasta`	Define the reference file for this analysis.
`set saet.run 0 secondary/saet.ini`	Turns off the SAET module. SAET IS NOT recommended for ChIP-Seq.
`ls`	Display the configuration of your analysis, including input reads, reference files, workflow, and select parameter settings.
`run`	Start the analysis run.
`ls`	Display status information about the analysis run.

See Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information on running workflows, including the ChIP-Seq workflow.

Do not rename

Do not delete or rename any of the following directories or files for your workflow:

- The `analysis.pln` file
- The `secondary` directory
- The `tertiary` directory
- The `secondary.pln` file

Turn off SAET

We recommend not running SAET with the ChIP-Seq workflow. When you turn off SAET in a workflow, make sure your mapping INI file does not contain the line:

```
analysis.input.readset.file = ${analysis.output.dir}/saet/*.rrs
```

Comment out this line if it appears in your mapping INI file.

Run ChIP-Seq as an individual analysis

The optional examples download includes an example of how to run the ChIP-Seq module by itself, as an individual analysis that is not part of a standard workflow.

See Appendix H, "Demo Analyses" on page 567 for information on the examples.

Mapping algorithm

The ChIP-Seq module uses the resequencing mapping and pairing algorithm. See the following sections for more information:

- "Stages of mapping" on page 140
- "Mapping algorithm" on page 153
- "Mapping statistics" on page 167
- "Mapping output files" on page 167

Use results files

After the mapping steps are complete, the resulting BAM file can be used with compatible third-party commercial and academic ChIP-Seq analysis software tools (see Figure 46 on page 475).

As of this writing, you can download a BAM-to-BED format converter from third-party tools sites, for example:

code.google.com/p/bedtools

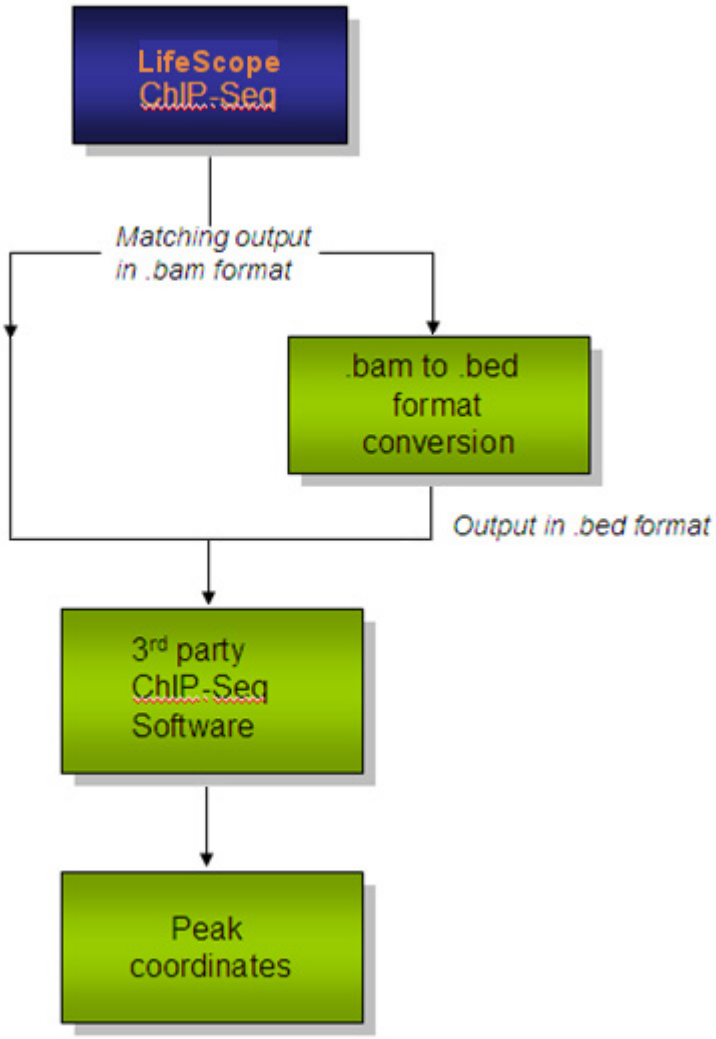


Figure 46 Compatible ChIP-Seq analysis software tools

27 Run a MethylMiner™ Mapping Analysis

This chapter covers:

- Overview 477
- Run MethylMiner™ mapping 478
- MethylMiner™ mapping output files 481
- Mapping algorithm 481
- Further analysis of MethylMiner™ mapping results 481

Overview

DNA methylation is an epigenetic modification crucial for organism development and normal gene regulation. Life Technologies have introduced a versatile methyl-CpG binding protein-based system, the MethylMiner™ Kit, for the enrichment of methylated sequences from genomic DNA, that, with the use of SOLiD® System sequencing, allows for focused evaluation of methylation patterns in genome-wide studies. The enriched reads can also be bisulfite-converted, which would additionally allow determination of methylation status of individual cytosines.

SOLiD® System sequencing can also, of course, be used to study methylation in unenriched, whole-genome bisulfite-converted read data.

The purpose of LifeScope™ Genomic Analysis Software MethylMiner™ module is to provide a data analysis pipeline for mapping and analyzing MethylMiner™ enriched and unenriched fractions of genomic DNA as well as bisulfite-converted and unconverted reads sequenced on the SOLiD® System.

The MethylMiner™ analysis module functionality currently includes:

- Mapping of unconverted and bisulfite-converted reads.
- Mapping statistics and statistics on read coverage and depth.
- Mapped reads output in BAM-format files.
- Visualization of mapped reads on publicly available genome browsers.

Figure 47 shows the MethylMiner™ analysis workflow with SOLiD® software and third-party browsers.

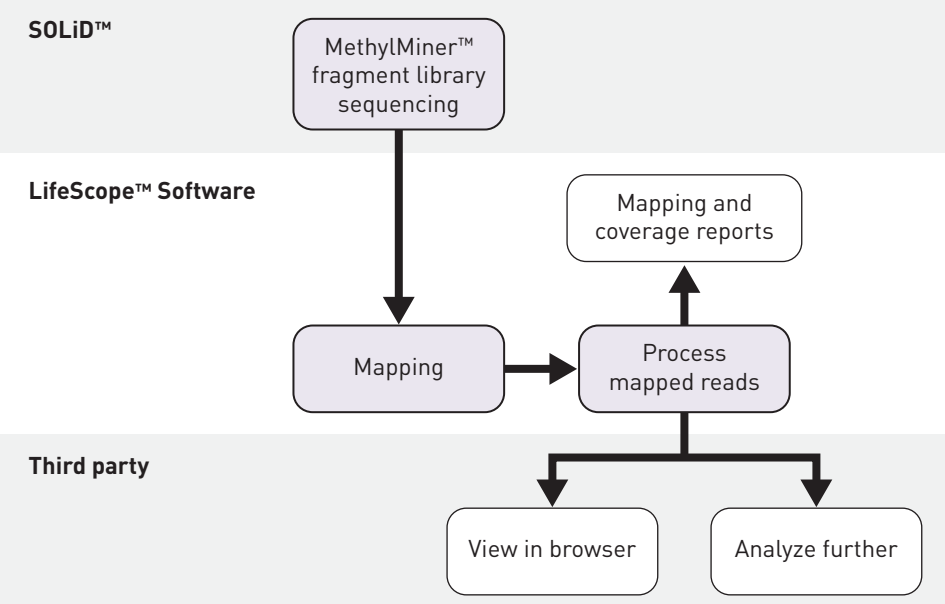


Figure 47 Methylation data analysis module overview

Run MethylMiner™ mapping

Unconverted MethylMiner™ reads must be mapped to a regular (unconverted) reference genome sequence.

Bisulfite-converted reads must be mapped to a pair of appropriately converted reference sequences (forward and reverse conversions), as recommended below.

Note: MethylMiner™ supports fragment and paired-end libraries. Do not use this module with mate-pair libraries.

Follow these recommendations to map MethylMiner™ bisulfite-converted reads:

1. For mapping bisulfite-converted reads, the converted reference sequence pairs below are recommended. Use either Pair 1 or Pair 2 below.
   - Pair 1:
     - Reference with all non-CpG C's converted to T's (the forward conversion)
     - Reference with all non-CpG G's converted to A's (the reverse conversion)

     We have found that Pair 1 is most appropriate for bisulfite-converted reads from MethylMiner™ samples.
   - Pair 2:
     - Reference with all C's converted to T's
     - Reference with all G's converted to A's

     Pair 2 is most appropriate for bisulfite-converted reads from whole-genome data (not MethylMiner™ enriched).

2. The two sets of reference sequences within a given pair must be saved as separate reference files, and the mapping must be performed separately with each of them.

Run MethylMiner™ workflows

MethylMiner™ analysis is available as two standard workflows in the LifeScope™ Software command shell:

- For fragment data: `methyl.miner.frag`
- For paired-end data: `methyl.miner.pe`

These workflows use the resequencing mapping modules to perform mapping to a selected reference and generate standard BAM files.

The additional consolidation of the results from the forward and reverse conversions must be done by the user, and further analysis can be done with third-party tools.

Run in the lscope command shell

This section describes how to run the fragment MethylMiner™ mapping workflow in the LifeScope™ Software command shell.

In the LifeScope™ Software command shell, running the MethylMiner™ workflow requires the following steps:

- Create a LifeScope™ Software command shell project and analysis.
- Identify your input data (your reads).
- Identify the reference genome.
- Specify the MethylMiner™ fragment workflow as the analysis type to be executed on your data.
- Issue the `run` command to start your analysis.

Example steps to run the MethylMiner™ fragment mapping standard workflow

This section shows the steps to run the MethylMiner™ fragment mapping workflow in the LifeScope™ Software command shell. A description of each step is given in Table 136.

```
lscope.sh shell -u username -w password
cd /reads
import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq
cd /projects
mk methyl
cd methyl
mk run1
cd run1
set workflow methyl.miner.frag
add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq  lung
set reference /data/results/references/DH10B.fasta
set saet.run 0 secondary/saet.ini
ls
run
ls
```

Dummy XSQ file names are used in the example.

Information about your workflow job

Output of the `ls` command displays the analysis' parameters, INI files, and run completion percentage or completion status.

Table 136 explains the purpose of the steps shown above.

Table 136 Description of sample shell commands to run the MethylMiner™ workflow

Command	Purpose
`lscope.sh shell -u username`	Log into the LifeScope™ Software command shell.
`cd /reads` `import /data/xsq/DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq`	Go into the reads repository and import an XSQ file (the input data file). This command in effect brings the reads file into the shell reads repository. Once in the reads repository, the reads file is available as input to your analysis runs.

`cd /projects`
`mk methyl`
`cd methyl`
`mk run1`
`cd run1`
	In the projects repository, create a project named `ecoli`, and in that project create an analysis named `run1`. Open the `run1` analysis.
	Note: The open analysis (`cd run1`, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.

`set workflow methyl.miner.frag`	Define the workflow used with this analysis. The MethylMiner™ workflow includes SAET, mapping, mapping statistics, and duplicate bead finding.
`add xsq DH10B_ColorSpaceOnly_Unbarcoded_50_Frag.xsq`	Define the XSQ file containing the input reads data for this analysis.
`set reference /data/results/results/references/DH10B.fasta`	Define the reference file for this analysis.
`set saet.run 0 secondary/saet.ini`	Turns off the SAET module. SAET IS NOT recommended for the MethylMiner™ workflow.
`ls`	Display the configuration of your analysis, including input reads, reference files, workflow, and select parameter settings.
`run`	Start the analysis run.
`ls`	Display status information about the analysis run.

See Chapter 7, "Run a Standard Workflow Analysis" on page 111 for more information on running workflows, including the MethylMiner™ workflows.

Do not rename

When using a standard workflow, do not delete or rename any of the following directories or files:

- The `analysis.pln` file
- The `secondary` directory
- The `tertiary` directory
- The `secondary.pln` file

Visualize MethylMiner™ results

After the mapping steps are complete, the resulting BAM file with mapped reads can be visualized with compatible third-party commercial software tools and publicly-available genome browsers (see Figure 47 on page 478).

To visualize the mapped reads, launch your UCSC genome browser or IGV browser to import and view the mapped read BAM file generated by MethylMiner™ mapping.

MethylMiner™ mapping output files

Mapped reads are output in BAM files.

The output mapping statistics reports generated by MethylMiner™ mapping include:

- The number and percent of all reads mapped and of reads mapped uniquely, in the file `mapping-stats.txt` and also in files ending in `F3.stats` and `F5-P2.stats`.
- Genome coverage tables for at least read depths 0X to 50X, in output files ending in `coverage-histogram-F3.txt` and `coverage-histogram-F5-P2.txt`.

Mapping algorithm

The MethylMiner™ module uses the resequencing mapping and pairing algorithm. See the following sections for more information:

- "Stages of mapping" on page 140
- "Mapping algorithm" on page 153
- "Mapping statistics" on page 167
- "Mapping output files" on page 167

Further analysis of MethylMiner™ mapping results

Besides viewing mapped reads in genome browsers, you can further analyze mapped reads outside of LifeScope™ Software with software available in the SOLiD® development community or with other third-party tools. MethylMiner™ unconverted mapped reads can be processed with peak-finding programs like MACS to identify genome regions of significant methylation.

Similarly, MethylMiner™ bisulfite-converted mapped reads can be processed with peak-finding programs to identify regions of significant methylation. These reads can also be processed at nucleotide resolution to report the methylation status of individual C bases, for bases covered at sufficient read depth.

PART IX
Appendices

A FAQ

This appendix covers:

- FAQ lists for LifeScope™ Genomic Analysis Software modules 485
- General FAQ for LifeScope™ Software 485

FAQ lists for LifeScope™ Genomic Analysis Software modules

Frequently-asked questions for modules are available in the analysis module chapters. See the following sections for module-specific FAQ lists:

- "FAQ – Mapping"
- "FAQ – Pairing"
- "FAQ – BAMStats"
- "FAQ – SNPs"
- "FAQ – Human CNVs"
- "FAQ – Small indels"
- "FAQ – Large indels"
- "FAQ - Annotations"
- "FAQ – Whole transcriptome"
- "FAQ - Small RNA mapping"
- "FAQ - XSQ Tools"

General FAQ for LifeScope™ Software

1

What is the performance increase vs. RAM increase?

For the human genome, 24GB of RAM per node provides approximately 30% to 40% performance improvement over systems that have 16GB of RAM per node.

2

Does LifeScope™ Software support hyper-threading?

Yes, some LifeScope™ Software components can take advantage of hyper-threading if it is present.

3

Can I configure LifeScope™ Software to submit jobs to the queues that are already set up on my cluster?

You can use pre-existing queues. During installation you will be asked which existing queues you would like to use with LifeScope™ Software.

4

What is the expected time for my run to complete?

This depends on many factors, like the number of beads, the reference, the hardware of the cluster in use, the traffic condition if it's a shared cluster, the speed of storage system, etc.

In a typical dedicated three to five nodes cluster, with 8 cores per node and 24GB memory per node, for one (1) full slide in a mate-pair run, it takes about two days to do the mapping, pairing, and generate BAM files for 1 billion reads. Most analyses can be done within one or two days.

5

How do I know if my run is successfully complete?

Use the LifeScope™ Software command shell `ls` command in your analysis, or the `lscope.sh status` command at the Linux prompt.

See "Review job status" on page 107 and "Review logs" on page 108.

6

What should I do if my run never seems to complete?

Verify that your run partition is not full and that the queueing system is working correctly. If your run directory's partition is 100% full, LifeScope™ Software may report that your job has started, but LifeScope™ Software is not able to process the job.

7

Do you have example data to test my install with?

Ask your LifeScope™ Software administrator if the optional Examples directory was installed.

8

Can the "scratch" directory be on a shared NFS storage?

Scratch space can be a local disk or shared storage mounted via NFS, but there will be high I/O activity on the scratch disk. Ensure the storage has high throughput and will not degrade the overall performance of the system. LifeScope™ Software is not tested in an environment that is configured with scratch space that is mounted via NFS.

9

Can multiple users try to install LifeScope™ Software in different locations?

No. There is no need to install multiple copies. You just need to set the environment to point to LifeScope™ Software. See "Set the LifeScope™ Software environment" on page 68.

10

Can I continue to run my BioScope™ Software after I install LifeScope™ Software?

Yes. You must follow the instructions in "BioScope™ Software users' PATH variable" on page 68.

11

What changes are required for a job with a large number of contigs?

For a job with a large (more than a few thousand) number of contigs, change the system property `max.head.node.java.heap.space` from the default of 64 to 128 or 256.

For example, a run with 40,000 contigs required 128 for BAMStats and 256 for SNPs.

Modify this parameter in the <*installdir*>`/etc/analysis/system.properties` file.

Because changes to system properties affect all runs of all users, you could consider changing the parameter back to its original value after the large contig run completes. Setting `max.head.node.java.heap.space` to 128 or 256 could present a problem if your head node is extremely loaded.

B Troubleshooting

This appendix covers:

- Overview 489
- Logs. 489
- Troubleshooting 490

Overview

This appendix describes job log files and troubleshooting tips.

Logs

An analyses' log files are located in the results directory (set during the installation) under the following directory naming scheme:

username/project_name/analysis_name

The following logs contain messages related to your analysis:

- **`analysis.log`** – A full log of entire experiment
- **`summary.log`** – A log containing milestone messages
- **`log/`*`module`*`/*.log`** – Log files generated by the analysis modules in your run (modules such as mapping, SNPs, CNV, etc.)
- **`log/TORQUE/`*`module`*`/*.log`** – Log files from the compute node

  The TORQUE logs contain the following information of interest:
  - The node the job is run on
  - Job kill notification for timeout failures

Server log messages and errors are captured in the following files:

- <installdir>`/server/server.log`
- <installdir>`/server/server.err`.

Troubleshooting

Login issues

- Check that the LifeScope™ Software server is running.
  - The Linux command `ps -ef | grep WebServer` displays the server process, if it is running.
  - The file <installdir>`/server/server.pid` lists the process id and port number for the server.
- Check that the license server is running and is not out of licenses, with the command `lscope-lmgrd.sh status`, on the Linux command line.

Analysis failures

- Use the `lscope.sh status` command or the GUI status information to identify which analysis module failed and to check the error message.
- If the error message is not sufficient to diagnose the problem, use the summary log and the analysis module log to identify more details on the problem.

The server responds with an error

- Check the files <installdir>`/server/server.log` and <installdir>`/server/server.err`.
- Check the analysis configuration for sample grouping errors. In the command shell read-sets are grouped with `add xsq` commands options or with a groupsfile. In the GUI, groups are created in the **Add Data to Project** and **Add Group to Project** windows.

Note: In this release, sample group names and reference file names containing a hyphen or a space are not permitted.

Message: Read-only db connection

If power was lost, causing the LifeScope™ Software server to shut down, and the server generates the message "read-only db connection," type the following commands:

1. `lscope-server.sh stop`
2. `rm /share/apps/lifescope/server/UserDB/db*.lck`
3. `lscope-server.sh start`

If the problem persists, or if the error message is different (indicating a corrupt db), type the following commands to delete the UserDB, which will remove all the current users in the system. Then recreate the users.

1. `lscope-server.sh stop`
2. `rm -rf /share/apps/lifescope/server/UserDB`
3. `lscope-server.sh start`

Sequence name not found in reference

An analysis fails with a reference error, but the reference file is in the reference repository. The error message is similar to the following:

```
12 Apr 2011 16:17:41,750 ERROR [0x2a96825860] reference_sam_check:
49 - sequence name chr6 was not found in reference file;
```

If the error is seen with an analysis that previously ran without error, a conflict in the reference repository is possibly the cause. LifeScope™ Software first looks in the reference repository by assembly name, and then in the assembly folder searches for a reference file that includes the assembly name in its file name. If a file with the assembly name is not found, the software then uses the first file in that folder (alphabetically). The reference selected may or may not contain the required sequence.

The avoid this type of error, when adding a new reference file to the repository, always add it under the subfolder for the correct assembly name. Create a new assembly name subfolder under `referenceData/internal` when adding a new genome. See Appendix E, "The Reference Repository" on page 539 for more information.

To recover from this error, you could use the `set reference` command with an absolute path to the reference file on the Linux file system.

Memory errors

If a job is killed for exceeding its requested memory limit, an error similar to the following is seen (the example memory numbers are unimportant):

```
PBS: job killed: mem job total 4203328 kb exceeded limit 4194304 kb
```

Job schedulers can be configured to kill jobs that exceed their requested memory or to be tolerant as long as memory is available.

Use the `memory.request` or `mapping.memory` parameters to configure the memory request for each analysis module.

Special characters in BAM header fields

The following special characters are not allowed in XSQ file fields that mapped to BAM header fields:

- **Slashes** – /, \
- **Dollar sign** – $
- **Greater than and less than signs** – >, <

Examples of BAM header fields in which these characters are not allowed are the following:

- AT in @CO
- LD in @CO
- PJ in @CO
- SP in @CO
- SO in @CO
- CO in @CO
- SD in @CO
- ID in @PG
- SM in @RG

The presence of unsupported special characters in these fields can cause errors such as null pointer exceptions.

C

File Format Descriptions

This appendix covers:

- Introduction 493
- XSQ file format 494
- BAM header usage 498
- Color-space attributes 504
- Pairing information in a BAM file 505
- Hard clipping of incomplete extensions 506
- Visualize BAM output 507
- Indel alignments 508
- BED file format 510
- BEDGRAPH file format 511
- Reference file data overview 512
- Read-set file format 513
- Legacy format translation 515
- Legacy CMAP file format description 516

Introduction

This appendix describes file formats used with LifeScope™ Software.

Before reading the section about the SOLiD® System BAM file contents, you should be familiar with the general SAM specification and with the SAM specification field definitions.

Before SOLiD® 4.0, the primary SOLiD® read and alignment format was based on the public GFF specification. The specification defined the location of an aligned read on a reference sequence and provided for arbitrary name-value pair attributes. For the SOLiD® GFF, attributes are used to detail color reads and quality values, base space translations, mate-pair information, and other aspects of the aligned read. Increased throughput of second-generation sequencing technologies resulted in the definition of the SAM file format and a compressed, indexed binary format (BAM) that expands the basic alignment information of a GFF file to include paired-read information and a more structured attribute set. A number of tools exist for basic manipulation of BAM files, and an array of viewers and analysis tools are available.

LifeScope™ Software secondary analysis (mapping and pairing) now produces a BAM file as the main alignment format. Mate-pair and paired-end analysis directly produces a BAM file, while a single file conversion is needed for fragment libraries. Depending on the output filter selected, unmapped and secondary alignments can be included.

LifeScope™ Software supports the XSQ sequence data file format introduced with the 5500 Series Genetic Analysis Systems.

XSQ file format

XSQ (eXtensible SeQuence) is an extensible file format for storing sequence data. The XSQ format supports multiple independent reads at the same position in a fragment. XSQ is a binary format based on the open HDF format.

Sequencing run data are automatically exported from the 5500 Series Genetic Analysis Systems in XSQ binary file format. If an ECC primer round has been performed, the XSQ output also includes the sequence information in base space, in addition to color space.

XSQ file content overview

5500 Series Genetic Analysis Systems data

XSQ files generated by the 5500 Series Genetic Analysis Systems contain:

- **Base-space data** – Reference-free base-space data from primary analysis (if the optional ECC round is performed).
- **Color-space data** – Referred to as 2+4 color, as follows:
  - The 2BE (Base Encoding) data is the data from first 5 primer rounds of regular 2BE probes. This data is in the same format as a data created by a SOLiD® 4 System non-ECC run.
  - The 4BE data is the data from the 6th primer round, the optional ECC round.

Data for optional ECC primer rounds is supported in an XSQ file.

Paired-end libraries do not support ECC data. Fragment libraries support ECC data, and mate-pair libraries support ECC data on both tags.

Trimming is allowed with base-space reads, and can occur at the beginning as well as the end of a read. The trimming is typically an output of an ECC run. The mapping of trimmed reads requires that the untrimmed portion be at least as long as the mapping seed/index length.

SOLiD® 4 System data

SOLiD® 4 System CSFASTA and QUAL files can be used with LifeScope™ Software, but first must be converted to XSQ format. Converted XSQ files contain only:

- Color-space 2BE data
- Color for both tags (one tag for fragment and both tags for mate-pair and paired-end data).

See Appendix D, "XSQ Tools" on page 519 for information on the XSQ converter.

Relation to sequencing instrument

This section describes how physical aspects of the 5500 Series Genetic Analysis Systems instrument affect the output XSQ files.

Each lane of the 5500 instrument results in a single XSQ output file. Different models of the 5500 support from 6 to 12 lanes, so typically multiple XSQ files are generated per sequencing run. You can direct your LifeScope™ Software analyses to data from specific barcodes run in different lanes, or group the output of multiple lanes or multiple sequencing runs to be processed as a single specimen.

The most granularity possible for a LifeScope™ Software analysis is the data from one barcode in one lane.

For information on specifying barcodes and specimen groups, see the following material:

- The `add xsq` command in Table 13 "LifeScope™ Software shell commands" on page 89.
- "Run a grouped analysis" on page 101 in Chapter 6, "Run a Command Shell Analysis" on page 83

XSQ file format description

This section describes several tables within an XSQ file. The complete XSQ file format specification is available on the SOLiD® Software Community website. At the time of this writing the specification is this site:

http://solidsoftwaretools.com/gf/project/xsq

The LibraryDetails table describes the properties of each library used in the XSQ file. String fields are limited in length to 255 characters.

Table 137 XSQ library details table

Column name	Description
LibraryName	Describes the name of one library preparation, which may be used across several indexes in an indexing run.
Application	Describes how this sequence is intended to be analyzed. Allowed values are: • Whole genome resequencing • Targeted resequencing • Whole transcriptome (Fragment) • Whole transcriptome (PairedEnd) • Small RNA • ChIP-Seq • Methylation
ProjectName	Describes the project within which this sample is being run.
SampleOwner	Names the owner of this sample.
SampleIdentifier	A unique name for this sample.
Description	Describes the sample.
Comments	Optional additional information about this sample or its preparation.
ERCC	Whether internal controls have been used. Reserved for future use.

Table 137 XSQ library details table *(continued)*

Column name	Description
Species	The intended species for mapping. Example values include: • Homo sapiens • Mus musculus • other
Assembly	The intended assembly for mapping (may be blank if unknown). Example values include: • hg18 • hg19 • mm9 Other valid assembly names are accepted. Spaces are not allowed.
LibraryInsertSize Minimum	The expected minimum insert size in nucleotides. Used with paired-end and mate-pair libraries. Allowed values: Integers >= 0. Set to 0 for fragment libraries, or when the size is unknown.

Each read type (tag) has a tag-level properties in the XSQ file. Table 138 lists the attributes describing the base-space and color-space data for a single type of read. Table 139 on page 497 lists tag names for current library types.

Table 138 XSQ tag details table

Column name	Default	Description
TagSequence	—	The full sequence of the tag used for primer annealing. It is used in conjunction with the read offset to determine the nucleotide or nucleotides for phasing the color-space reads. When the full sequence is not available, the last base(s) of the tag are sufficient as long as they cover the largest offset in the DatasetColorEncoding table. Conditional: Required when color space is present.
IsColorPresent	0	Whether color-space call data is present. Values: • **0**: No calls are reported in color space (causing the ColorCallQV dataset to be missing). • **Non-zero**: Color calls are reported.
IsBasePresent	1	Whether base call data is present. Values: • **0**: No calls are reported in base space (causing the BaseCallQV dataset to be missing). • **Non-zero**: Base calls are reported.
NumBaseCalls	—	The maximum untrimmed read length in base space across all reads for this tag. This length is the width of the BaseCallQV dataset. This field is not used when the BaseCallQV dataset is not present, but is required when BaseCallQV is present.
MinTrimmedReadLength	—	The minimum trimmed read length across all unfiltered reads for this tag. Allowed values: Integers >= 0.
DoesRepresentedStrandMatchSource	1	Whether the reads are represented on the same strand as the source. Values: • **0**: The synthesis strand is not the same as the source strand. • **Non-zero**: The synthesis strand is the same as the source strand.

Table 138 XSQ tag details table *(continued)*

Column name	Default	Description
RelativeOrder	—	Describes the relative order of a read on the sense strand of the source nucleic acid, in relation to all of the other reads from the same fragment. 1 based. Required only when there is more than one tag per fragment.
SynthesisFromThreePrime	0	(*Optional*) Whether the synthesis direction of the synthesized strand is from the 3′ end. Values: • **0** or missing: synthesis from the 5′ end to the 3′ end. • **Non-zero**: The synthesis direction is from the 3′ end to the 5′ end.

Table 139 lists tag names for current library types

Table 139 Tag information for current library types

Instrument	Library type	Indexing	Tag name	Base	Does-Represented-StrandMatch-Source	Relative-Order	Synthesis-FromThree-Prime
SOLiD®	Fragment	Yes | no	F3	T	1	1	1
SOLiD® and 5500	Mate-pair	No	F3	T	1	2	1
			R3	G	1	1	1
SOLiD®	Paired-end	No	F3	T	1	1	1
			F5-P2	T	0	2	0
SOLiD®	Paired-end	Yes	F3	T	1	1	1
			F5-BC	G	0	2	0
5500	Paired-end (DNA)	Yes | no	F3	T	1	1	1
			F5-DNA	T	0	2	0
5500	Paired-end (RNA)	Yes | no	F3	T	1	1	1
			F5-RNA	G	0	2	0

The DatasetColorEncoding table in the XSQ file is required when color space is present. The table describes how reads are encoded as colors by the instrument. Indexing reads are not included in this table. The DatasetColorEncoding table is described in Table 140.

Table 140 The DatasetColorEncoding table

	Column name	Description
1	DataSetName	Must match the name of a read dataset under a tag group. This table describes the encoding scheme used to interpret that read dataset.
2	Offset	Describes the distance and direction of the primer end as compared to the tag end. The offset is used to relate individual calls to the bases they encode.

Table 140 The DatasetColorEncoding table

	Column name	Description
3	Encoding	Contains a set of integers that direct the conversion from nucleotides to colors as described in the full XSQ file format document on the SOLiD® Software Tools website. (2BE is '11', 4BE is '1303').
4	Stride	Used to pack discontinuous reads into a continuous array. Stride is used to determine the relative positioning of the start nucleotides between color calls.
5	NumColorCalls	The maximum untrimmed read length in base space across all reads for this tag. This value is the width of the associated CallQV dataset.

For more information on the XSQ file format refer to this site:

http://solidsoftwaretools.com/gf/project/xsq

BAM header usage

The BAM file generates all of the header information required by the SAM format specification, including @HD, @SQ, and @RG lines.

To view the content of the BAM file header, use the following command:

```
samtools view -H <BAMfilename>
```

Sequence dictionary (@SQ)

Sequence header lines include the reference file URL, for example, `file://share/reference/genomes/hg18.fa` in the optional UR field of the reference file. This value might become invalid if you relocate files.

Read group (@RG)

Read groups receive an arbitrary ID and sample name. The library field (LB) contains information that is important to downstream algorithms that use pairing information. A library name, which is specified by the tool parameter `library.name`, and the library type, are separated by a dash in the LB field. The library type is a structured value that details the nominal length of the two tags and the protocol used as shown in the following syntax example:

```
l1(x12)[F|MP|RR|RRBC]
```

In the syntax example, *l1* is the nominal length of the first read and *l2* is the nominal length of the second read. There will only be one number for fragment libraries. The library types correspond to fragment (*F*), mate-pair (*MP*), reverse read (*RR*), and reverse read-bar coded (*RRBC*).

Detecting structural variations, particularly large insertions and deletions, depends on the range of pairing insert (PI) sizes. The pairing tool generates the information about PI sizes. The information has been used to define the three-letter pairing category, specifically the third letter. (See Table 34 on page 182 and Table 35 on page 183 for descriptions of the three-letter categories.) The PI field in the read group captures the range of pairing insert sizes with a range of the form shown in the following example:

```
PI:low-high
```

In the PI example, `low` is the lower bound of the pairing range in nucleotides, and `high` is the upper bound.

Header (@HD) sort order

The LifeScope™ Software SNPs module requires that BAM headers of its input files contain an @HD line with a SO sort-order field, even if the BAM file is sorted properly. The required header line is:

```
@HD    VN:1.0    GO:none SO:coordinate
```

This header contains the information that the BAM file is sorted by genomic coordinates. BFAST mapping, for example, by default sorts the BAM file by coordinates, but does not update the header to reflect the sort order.

This section describes how to update the @HD SO field. If your BAM file is sorted properly but does not contain the SO sort-order field, follow these instructions before using a BAM file generated outside of LifeScope™ Software with the SNPs module:

1. Write the BAM file header to a file

   ```
   samtools view -H input.bam > input_header.sam
   ```

2. Edit the BAM header file (`input_header.sam` created in the previous step). Insert the following as the first line of the header:

   ```
   @HD    VN:1.0    GO:none SO:coordinate
   ```

   Make sure that the fields are tab-separated.

3. Update your BAM file with this new header. This command requires samtools-0.1.8 or later.

   ```
   samtools reheader input_header.sam input.bam
   ```

These instructions are not required with BAM files generated by LifeScope™ Software mapping modules.

BAM file validation

In order to be used by LifeScope™ Software tertiary analysis tools, the BAM file must have the @RG field populated in the BAM header. If you are using BFAST mapping analysis, you must run it with the -R option in the post-processing step to create the @RG field in the BAM file header. BAM files created outside of LifeScope™ Software must pass Picard validation in order to be used with LifeScope™ Software tertiary analysis tools.

Note: Picard validation and samtools are supported by the open source project SourceForge.net. Applied Biosystems is not responsible for the availability or functionality of the Picard utility.

At the time of this writing the syntax for Picard validation is available at the following URL:

http://picard.sourceforge.net/command-line-overview.shtml#ValidateSamFile

The Picard utility, `picard-tools-`*`version`*`.zip`, (where *version* is the current version number) can be downloaded at:

http://sourceforge.net/projects/picard/files

The suggested Picard validation command is:

```
java -Xmx8g -jar ./ValidateSamFile.jar I=./input.bam \
    IGNORE=MISSING_TAG_NM
```

where *input.bam* is the name of the BAM file to be validated.

XSQ metadata in BAM headers

LifeScope™ Software secondary analysis modules write metadata contained in XSQ reads files to their mapping output files, as metadata in BAM header comment (@CO) lines. The new metadata is not in the BAM specification, but BAM files with this metadata conform to the BAM specification. Table 141 lists the XSQ file fields with both the new @CO field names and the existing BAM header fields that contain the XSQ metadata.

Tie to existing BAM header lines

Each @CO metadata line with metadata information references one of the following types of BAM header lines:

- **Program group** – @PG lines capture information about the program that generated the data originally. The sequencing instrument and primary analysis software are considered to be the initial program. An example @PG line is:

  ```
  @PG ID:Mordor_201103161921_0_3 {...}
  ```

- **Read group** – @RG lines contain information on library type and read length. (See "Read group (@RG)" on page 498 for more information.) An example @RG line is:

  ```
  @RG ID:1 SM:Stooges LB:Larry DS:"75x35PE" PI:225 {...}
  ```

- **General header** – @HD lines are general header lines.

All the attributes on the @CO metadata line then apply to referenced line.

The following example shows how @CO lines containing BAM metadata information refer to read group and program group lines. Except for @CO HD lines, the second field of the @CO metadata lines tie its attributes to either @RG or @PG lines. In the example, the `RG:1` fields tie the metadata in those @CO lines to the @RG read group with an ID of `1`. The `PG:Mordor_201012161921_0_3` field ties the metadata in that @CO line to the @PG program group with an ID of `Mordor_201012161921_0_3`.

```
@CO RG:1 IA:215 IS:5.232 IN:200 IM:250
@CO RG:1 TN:50 TX:75 TB:0 TC:1
@CO RG:1 UN:25 UX:35 TB:0 TC:1
@CO RG:1 BX:0 EC:0
@CO PG:Mordor_201012161921_0_3 AS:"primary5500 1.0.7"
@CO PG:Mordor_201012161921_0_3 \
    CL:"/usr/bin/primary5500 <with full arguments>"
@CO HD UF:"f47ac10b-58cc-4372-a567-0e02b2c3d479"
```

Metadata table

Table 141 lists the existing BAM header fields that contain the XSQ metadata, the new @CO metadata field names, and the XSQ file fields. See also "Input read count fields" on page 502.

Table 141 XSQ-related BAM metadata fields

Field name	Public BAM 1.3 spec field	New field in @CO line	XSQ field name	Optional or mandatory
Reference	@SQ SP	—	Species { 'Homo sapiens' | 'Mus musculus' | 'other' | ...}	Mandatory

Table 141 XSQ-related BAM metadata fields

Field name	Public BAM 1.3 spec field	New field in @CO line	XSQ field name	Optional or mandatory
Assembly	@SQ AS	—	Assembly { 'hg18' | 'hg19' | 'mm9' | ...}	Mandatory
—	—	@CO HD UF	FileUUID	Optional
Program	@PG ID	—	InstrumentName_Date_FlowcellAssignment_ LaneNumber	Mandatory
Analysis Software	—	@CO PG:x AS	AnalysisSoftware	Optional
InstrumentSerial	—	@CO PG:x PS	InstrumentSerial	Optional
InstrumentName	—	@CO PG:x PN	InstrumentName	Optional
InstrumentVendor	—	@CO PG:x PV	InstrumentVendor	Optional
InstrumentModel	—	@CO PG:x PM	InstrumentModel	Optional
ReadGroupID	@RG ID	—	LibraryName or LibraryName_IndexName (if present)	Mandatory
—	—	@CO RG:x IX	IndexName (if present)	Optional
—	—	@CO RG:x II	IndexID (if present)	Optional
Sequencing Center	@RG CN	—	SequencingCenter	Optional
Description	@RG DS	—	l1(xl2)[F|MP|RR|RRBC]	Mandatory
Library Description	—	@CO RG:x LD	LibraryDetails.Description	Optional
Library Type	—	@CO RG:x LT	LibraryType { 'MatePair' | 'PairedEnd' | 'Fragment' }	Mandatory
Application Type	—	@CO RG:x AT	ApplicationType { 'Whole Genome Resequencing' | 'Targeted Resequencing' | 'Whole Transcriptome (Fragment)' | 'Whole Transcriptome (PairedEnd)' | 'Small RNA' | 'ChIP-Seq' | 'Methylation' }	Mandatory
—	@RG DT	—	RunStartDate	Optional
—	—	@CO RG:x DE	RunEndTime	Optional
Library	@RG LB	—	LibraryName	Mandatory
Predicted Insert Size	@RG PI	—	(LibraryInsertSizeMin + LibraryInsertSizeMax) / 2	Conditional
Calculated Average Insert Size	—	@CO RG:x IA	<calculated>	Mandatory
Calculated Insert Size Std Dev	—	@CO RG:x IS	<calculated>	Mandatory
Min Insert Size from User Input	—	@CO RG:x IN	LibraryInsertSizeMinimum	Conditional
Max Insert Size from User Input	—	@CO RG:x IM	LibraryInsertSizeMaximum	Conditional
Input Read Count Passing All Filtering Steps	—	@CO RG:x CU	The sum across all ImageUnits of *ImageUnitID.Fragments.NumFragmentsPassed* (passes all filtering steps)	Optional

Table 141 XSQ-related BAM metadata fields

Field name	Public BAM 1.3 spec field	New field in @CO line	XSQ field name	Optional or mandatory
Input Read Count Total	—	@CO RG:x CT	The sum across all ImageUnits of *ImageUnitID. FragmentCount* (filtered and unfiltered) Not used in data converted from pre-5500 systems. Mandatory for 5500 data.	Mandatory
Specimen		@CO RG:x SP	SampleIdentifier	Optional
Platform name	@RG PL	—	"SOLID"	Mandatory
Lane Id	@RG PU	—	FlowcellAssignment_LaneNumber	Optional
Sample (Pool)	@RG SM	—	SequencingSampleName	Optional
—	—	@CO RG:x SD	SequencingSampleDescription	Optional
Tag1 Was Base Present In XSQ	—	@CO RG:x BX	Tag1.IsBasePresent	Mandatory
Tag1 Min Read Len	—	@CO RG:x TN	Tag1.MinTrimmedReadLength	Mandatory
Tag1 Max Read Len	—	@CO RG:x TX	Tag1.NumColorCalls or Tag1.NumBaseCalls	Mandatory
Tag2 Was Base Present In XSQ	—	@CO RG:x BY	Tag2.IsBasePresent	Mandatory for MP, PE. Optional for fragment
Tag2 Min Read Len	—	@CO RG:x UN	Tag2.MinTrimmedReadLength	
Tag2 Max Read Len	—	@CO RG:x UX	Tag2.NumColorCalls or Tag2.NumBaseCalls	
ECC run	—	@CO RG:x EC	<whether this was an ECC run (inferred)>	Mandatory
ERCC	—	@CO RG:x ER	ERCC	Optional
—	—	@CO RG:x CO	Operator	Optional
—	—	@CO RG:x UU	LibraryIndexUUID	Optional
—	—	@CO RG:x PN	Application	Optional
—	—	@CO RG:x PJ	ProjectName	Optional
—	—	@CO RG:x SO	SampleOwner	Optional

Input read count fields

This section describes aspects of the Input Read Count fields from the viewpoint of the sequencing instrument and the viewpoint of secondary analysis.

The Input Read Counts fields are:

- **CU –** Input Read Count Passing All Filtering Steps, @CO RG:x CU
- **CT –** Input Read Count Total, @CO RG:x CT

The number of *total beads*, before any filtering on the instrument, is given by these fields:

- **NumFragmentsTotal –** In mapping statistics output
- **FragmentCount –** In an XSQ file
- **CT –** In BAM header metadata

The number of *beads that pass filtering* on the instrument is given by these fields:

- **NumFragmentsPassingFilters –** In mapping statistics output
- **NumFragmentsPassed –** In an XSQ file
- **CU –** In BAM header metadata

The number of *beads filtered out* on the instrument is given by:

NumFragmentsTotal – NumFragmentsPassingFilters

Example of @CO metadata in a BAM header

The following is an example of a BAM header with an @CO comment line containing metadata information. The @CO line applies to the @RG ID:12345678 line below it. (The @SQ lines for chromosomes 2–22 are deleted.)

```
@HD  VN:1.0  GO:noneSO:coordinate
@SQ  SN:chr1  LN:247249719  UR:file:/ref/hg18.fasta
  SP:Homosapiens  AS:hg18
  ...
@SQ  SN:chrX  LN:154913754  UR:file:/ref/hg18.fasta
  SP:Homosapiens  AS:hg18
@SQ  SN:chrY  LN:57772954   UR:file:/ref/hg18.fasta
  SP:Homosapiens  AS:hg18
@SQ  SN:chrM  LN:16571    UR:file:/ref/hg18.fasta
  SP:Homosapiens  AS:hg18
@PG  ID:SOLiD_12321_1
@CO  RG:12345678 LT:MatePair  AT:WholeGenomeResequencing
  LN:<50X50MP>  PI:1500  IA:1500  IS:300  IN:1200  IM:1800
  PU:1234  SM:sample  BX:0  TN:50  TX:50  BY:1  UN:50  UX:60  EC:0
@RG  ID:12345678 DS:50x50MP  LB:MP  PI:1500  PU:1234  SM:sample
```

@CO syntax and rules

This section describes the syntax rules and conventions for @CO metadata lines. The @CO metadata rules are:

- The second field of each metadata @CO line either notes that the line is a general header (HD), or refers to a single @RG ID or @PG ID.
- Each metadata @CO line contains one or more tab-separated attributes.
- For a metadata @CO line that refers to either a @RG or @PG line, all of the @CO attributes must apply to the same @RG or @PG.
- The beginning of metadata @CO lines must match the regular expression "^@CO\t[PR]G:" and contain no other non-metadata text. LifeScope™ Software ignores and does not change @CO lines that do not match the regular expression.
- The @CO PG:*id* line describing the instrument is required and must refer to a valid @PG ID:*id* program group line. This @CO line contains information about the primary analysis done on the instrument.

According to the BAM specification, the @HD line is optional. If present, the @HD line is required to be the first line of the header. The order of all other header lines is not defined. The following order of @CO metadata lines is preferred when possible (but not required):

- Place the @CO HD:UF comment immediately after the @HD line.
- Place @CO RG: and @CO PG: comment lines before other @CO lines for ease of visibility.

- Place @CO RG: and @CO PG: comment lines before the @RG and @PG blocks that the comment lines refer to.

Special character restrictions

Table 142 lists special characters that are not allowed in the LibraryName (@RG LB) and the LibraryType (LT in @CO) BAM header fields. Program failures are observed when either the LibraryName or the LibraryType fields contain one of these characters.

Table 142 Special characters not supported in the LibraryName and LibraryType fields

Character name	Character
Forward slash	\
Backwards slash	/
Period	.
Comma	,
Underscore	_
Hyphen	-
Space	" "
Dollar sign	$
Less than sign	<
Greater than sign	>
Number sign	#

The following BAM header fields do allow the characters listed in Table 142:

- AT in @CO
- LD in @CO
- PJ in @CO
- SP in @CO
- SO in @CO
- CO in @CO
- SD in @CO
- ID in @PG
- SM in @RG

Color-space attributes

The SAM format specification includes the attribute tags CS, CQ and CM. All BAM files support color-space reads (see Table 143).

Table 143 Color attribute tag description

Attribute tag	Description
CS	Color-space (CS) read. The CS field contains the original color-space read, which includes the primer base, in the orientation of the CSFASTA file. CS entries are not manipulated to be top-strand relative.

Table 143 Color attribute tag description

Attribute tag	Description
CQ	Color qualities. Color qualities are encoded according to the ascii-33 scheme used for the QUAL field. The orientation is the same as the orientation used for the CSFASTA file.
CM	The number of color-space mismatches.

Pairing information in a BAM file

The BAM file that is produced by the pairing tool supports both mate-pair and paired-end protocols using the standard SAM format fields, in particular the ISIZE and FLAG fields.

Calculation of tag names

The paired libraries use tag names to refer to members of the pair. The mate-pair libraries use F3 and R3 as the tag names. The paired-end libraries use F3 and F5 as the tag names. Use the FLAG field and information from the LB field of the read group to recapitulate tag names (see Table 144).

Table 144 Calculating tag names

FLAG bit	Library type	Tag name
0x0040 (first read in a pair)	LMP	F3
0x0080 (second read in a pair)	LMP	R3
0x0040	PE	F3
0x0080	PE	F5
0x040	F	F3

Proper pairs

Legacy file formats, such as *.mates, and GFF, described pairs using a three-letter category. Pairs in the AAA category correspond to the “proper pair” concept in the SAM format. The pairs reflect pairings that are not altered by a structural variation such as an inversion or deletion (see Figure 48). The BAM file field values for proper pairs are different for mate-pair and paired-end libraries:

Mate-pair libraries

- Strand flag is equal for both mates (both 0 or both 1).
- ISIZE is between the lower and upper limit of the insert range.
- For forward strand hits R3 POS < F3 POS.
- For reverse strand hits F3 POS < R3 POS.

Paired-end

- Strand flag is opposite for the mates.
- ISIZE is between the lower and upper limit of the insert range. In the case of paired-end libraries, the ISIZE might be smaller than the sum of the alignment lengths.
- F3 POS < F5 POS if F3 is on the forward strand.

- F5 POS < F3 POS if F5 is on the forward strand.

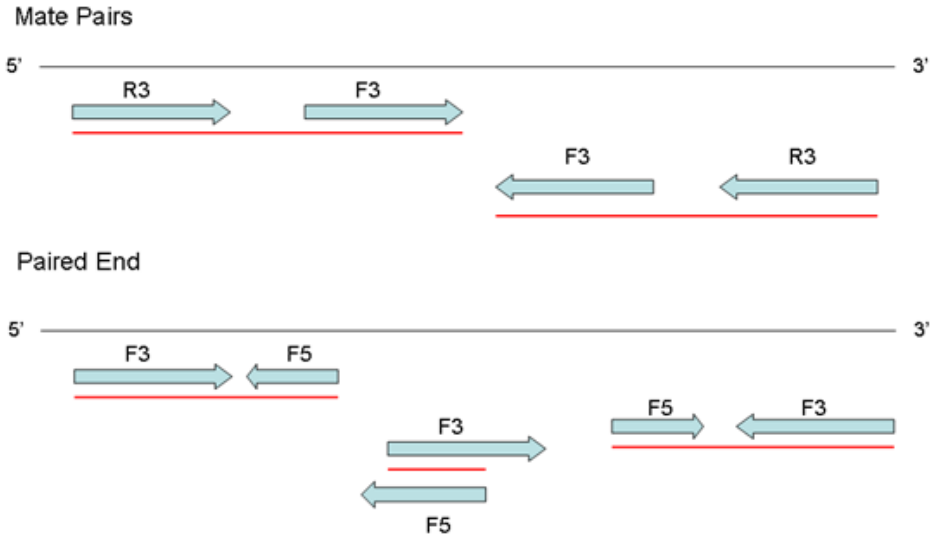


Figure 48 Example of proper pairs for mate-pair and paired-end protocols

Single read mapping quality

As described in the SAM format specification, the MAPQ field for paired results contains a pairing quality value. Under some circumstances. it is valuable to include the original single-read alignment quality value. The original single-read alignment value is maintained in the SM:i attribute in BAM files.

Hard clipping of incomplete extensions

LifeScope™ Software mapping uses a seed-extend algorithm. The algorithm increases mapping throughput by matching a seed, usually 25 bp, and extending the alignment until mismatches drive down the alignment score. Many alignments do not completely cover the color-space read. Because the base-space sequence of color reads cannot be precisely known in the absence of alignment, incomplete extensions are represented as a hard-clip (H) operation in the BAM CIGAR string (see Figure 49).

```
T33232030301212311201322311232302131021221120112222

AAGGCCTCTGAACCCACTCAGGTACTTAGCTGTAGATGGACATCAGTTAATTCGATGAC
      2222110211221201312032321132231021132121030302323 3T
        * *  *

8H42M
```

Figure 49 Example of hard clipping from a color alignment

Figure 34 shows the read in normal orientation (see the top section) and aligned in reverse orientation to the reference top strand (see the middle section). The lines below the alignment show the extent of the seed (top horizontal line) and extension (bottom horizontal line) phases of mapping. The extension only results in 42 bases of

alignment. The remaining portion of the color alignment has a number of mismatches that prevent extension. These are coded as hard-clipped regions. The CIGAR field in the BAM file is top-strand relative, so even though the hard clipping is on the end of the reversed color read, it is on the beginning of the CIGAR string.

Visualize BAM output

You can use third-party software visualization tools to view BAM files in a browser.

Integrative Genomics View (IGV)

The Integrative Genomics Viewer (IGV) available from the Broad Institute is a visualization tool for interactive exploration of large, integrated datasets. The IGV reads BAM files directly, which allows for easy viewing and inspection of alignments against the genome (see Figure 49).

For more information, go to the following site:

www.broadinstitute.org/igv/

If you use IGV to visualize the BAM files, verify that the BAI file is present. The BAI file is the index that is built for BAM files and is a standard part of the public SAM specification. If the pairing and MaToBam tools do not automatically create the BAI file:

1. Log in to the LifeScope™ Software cluster.
2. At a command prompt, enter:

   ```
   samtools index <BAM file name>.bam
   ```

Indexing only works if the file is sorted in coordinate order. If the file is not sorted in coordinate order, at a command prompt, run the following command to sort the file in coordinate order:

```
samtools sort <unsorted BAM name>.bam <sorted BAM name>
```

UC Santa Cruz (UCSC) genome browser

The UCSC Genome Browser serves as an interactive web-based microscope that allows researchers to view all 23 chromosomes of the human genome at any scale, from a full chromosome down to an individual nucleotide.

For more information, go to the Genome Browser website:

www.cbse.ucsc.edu/research/browser

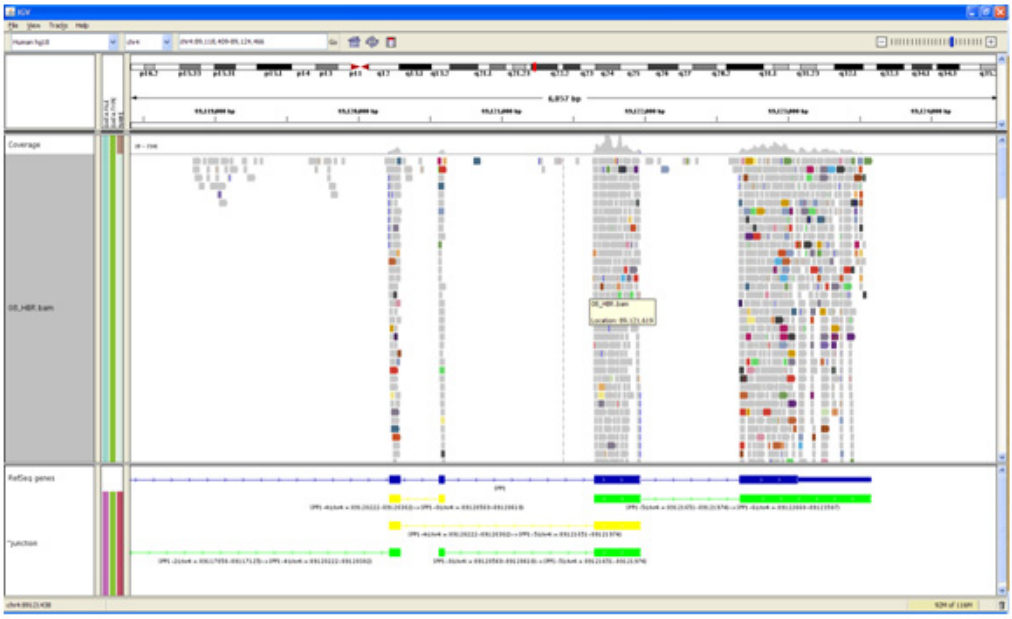


Figure 50 An example of a BAM file visualized in the IGV viewer

Indel alignments

The alignments are included in the secondary analysis BAM file. Indel alignments are included in quality calculations and the selections of primary alignments and pairs.

Nearly all of the fields in the PAS file are represented in SAM format. However, there is no allowance for ambiguous locations in the CIGAR string or elsewhere. Ambiguity occurs when a repeat element is inserted or deleted with respect to the reference sequence. Indel alignments include a user-defined attribute (XW:Z) to specify the range of possible locations within the read for an indel in a repeat region (see Figure 51).

```
GTCTCACCAAGTTTTTTACATTTTCT'
:12221101021.00000211300022
```

Figure 51 Example indel in a repeat region with ambiguous placement

Referring to Figure 51, in the alignment shown of a reference sequence stretch with a color read, the deletion of a single T, which is represented as a dot in the color sequence, cannot be placed precisely because of the repeated Ts. The XW attribute would span the homopolymer region (30_35).

Table 145 PAS file column descriptions

Field Name	Example	Description
Genome position	4294973387	The genome position given by this formula: C * 232 + P – 1 where *C* is the chromosome number and *P* is the position on that chromosome.

Table 145 PAS file column descriptions

Field Name	Example	Description
Indel size	–7	Number of bases in the indel. A negative value means a deletion, a positive value is an insertion, for example, –8 is a deletion of size 8, and 3 is an insertion of size 3.
Number of errors	2	Number of errors in the tag where the indel was found.
Alignment	See below.	Details of the alignment in the form of a number of concatenated fields.

The PAS file is a legacy format containing four tab-separated columns as described in Table 145. The fourth column contains the full alignment information. One example of an alignment is illustrated as follows.

The following is an example of PAS file content:

```
>600_16_579_14_Lib1_1_50_1_runName_sampleName,1_6074.51.2(17:17
_17)[G20]| 1_7297.1:(44.8.0)[T02] !8B52/
:B;*%=7+'539&)4>455++60+0<9.(2
```

The alignment is in the form of a number of concatenated fields. The text represents:

```
<data source>,<R3 tag details>|<F3 tag details> R3_FASTQ F3_FASTQ
```

where the indel may be either F3 or R3, and the other tag would be the anchor, ungapped alignment. The R3_FASTQ and F3_FASTQ is the color quality strings for those tags in FASTQ base quality format. For fragment, there is only one tag:

```
<data source>,<tag details> FASTQ
```

Table 146 explains the fields in the example PAS file content.

Table 146 PAS file field descriptions

Field	Meaning	Notes
Data source fields		
`600_16_579`	The bead ID.	—
`14`	the unmatched length for that bead if an ungapped extended alignment was found for that bead.	The value `99999` is used if there was no ungapped alignment found.
`Lib1_1`	A library indicator.	Deprecated.
`50`	The full read-length.	—
`1`	The run ID.	—
`runName_sampleName`	The concatenation of the run name and the sample name.	—
Indel tag fields		
`1_6074.26.2`	A sequence `1` hit at position `6074`. `26` is the match length with `2` mismatches.	—

Table 146 PAS file field descriptions *(continued)*

Field	Meaning	Notes
(17:17_17)	The read position found was position 17 (zero-based). 17_17 is the range of other possible positions. [G20] is the read sequence with the gap alignment.	insertion size = read_length – match_length – 1 Negative values are deletions, and positive values are insertions. In this example, the read length was 50, so the indel is a deletion of 2. If an indel is not detected in a tag, the Tag indel field contains the mapping hit information: <seq index>_<alignment start>.<num mismatches>
Anchor (non-indel) tag fields		
1_7297.1:(44.8.0)[T02]	The alignment and the sequence of the anchor non-indel tag. 1: The chromosome index (not necessarily the chromosome number). 7297: The chromosome position of the alignment. 1: The number of mismatches in the anchor part of the alignment. 44: The alignment extended length. 8: The overall number of mismatches. 0: The start of the alignment. [T02]: The color-space sequence.	—

Note: The positions in the PAS files are 0-based. This contrasts with the positions reported in the TXT and GFF files, which have been adjusted to be 1-based.

BED file format

The Browser Extensible Display (BED) format was developed to extend the UCSC Genome Browser with user- defined tracks. BED is used to visualize the splice and fusion junctions in the UCSC Genome browser and in the IGV browser (see Figure 52 on page 511). For general documentation about the BED format, including information about all of the BED fields, go to:

genome.ucsc.edu/FAQ/FAQformat.html

For information about visualization software, see "Pairing information in a BAM file" on page 505.

Each line in the track defines a junction where `chromStart` is the smaller of the coordinates and `chromEnd` is the greater.

There are two blocks because a junction typically contains two exons. BlockSizes are the lengths of the exons. The block starts the beginning of the exons. When fusions on different strands or chromosomes, two lines are added to the output, with each line representing one chromosome. Different colors are used to color-code different types:

Figure 52 on page 511 shows the Upstream Hypersensitive Region (UHR) gene region displayed with the Integrative Genomics Viewer (IGV) for positions 3,530,193 to 3,548,355 of Human Chr-1. The following sections describe the tracks in Figure 52.

WIG (x2) tracks

The top two tracks show the genomic coverage using the negative strand and positive strand generated by the Bam2Wig tool (Max: 100 coverage).

BAM track

The middle track shows the alignments from the BAM file. For display purposes, reads are filtered with MAPQ threshold of 45 (a stringent filter). Bases with quality value 5 to 20 are shaded.

BED track

The fifth track shows the junctions detected by the Junction Finder (BED file). As shown in the figure, all junctions detected are "known" and so are shaded in green.

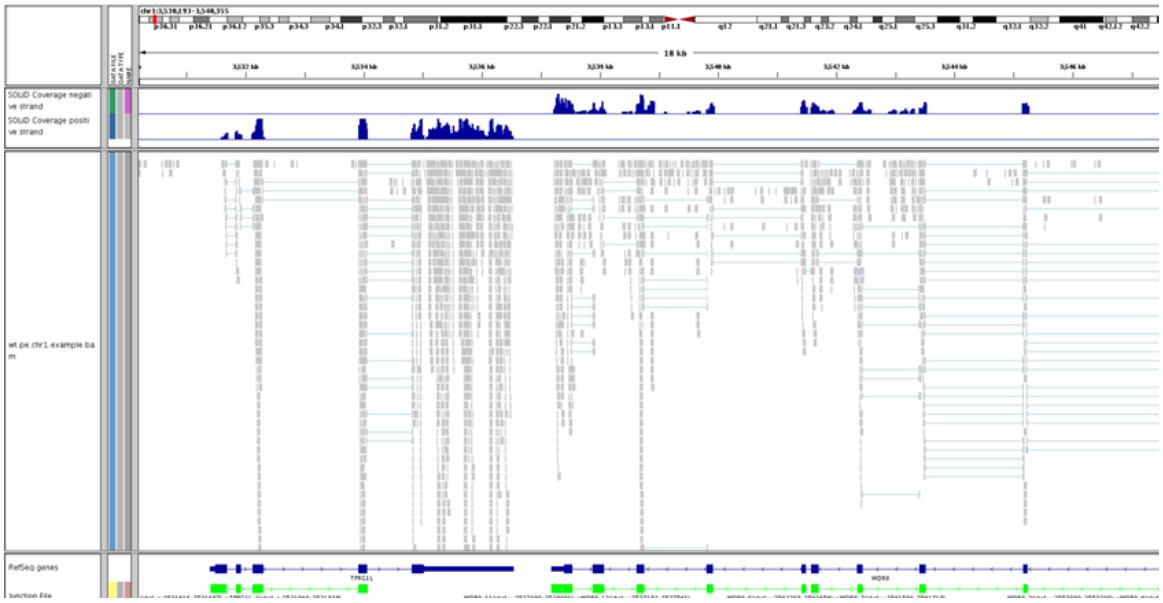


Figure 52 BED-IGV extended track example

BEDGRAPH file format

Table 147 describes the BEDGRAPH format fields. These are tab-separated text fields.

Table 147 Description of BEDGRAPH fields

Field name	Example content	Description
Track annotation headers	track type=bedGraph name="Coverage"	Annotations used for UCSC browser display purposes.

Table 147 Description of BEDGRAPH fields

Field name	Example content	Description
Contig name	chr1	Contig name of the feature. This value comes from the sequence name (SN) in the BAM header. However, in order to be viewed in the UCSC browser, these values must be of the form chr1, chr2, etc.
Feature start	148	Start location of the feature. Zero-based, inclusive.
Feature end	150	End location of the feature. Zero-based, exclusive (half-open ranges).
Coverage	1	Depth of coverage over the given (start-end) range.

Reference file data overview

Nearly all LifeScope™ Software analysis modules use reference sequence data and, in some cases, the modules also use reference annotations and metadata.

Reference data takes a number of forms, which are described in the following sections.

Contig multi-fasta file

A single, multiple-entry FASTA file where each entry corresponds to a genomic contig or chromosome.

Single contig FASTA file

A single-entry FASTA file containing one contig or chromosome.

CMAP file

A tab-delimited text file used in previous versions of the software. Each line specifies the path to a single contig FASTA file. Columns in the CMAP file provide annotation and paths to other chromosome specific support files.

GTF file

A genome annotation file. This file describes gene models used by the whole transcriptome and small RNA modules. It is similar in form to the GTF files downloaded from the UCSC website, but includes some changes for WT or small RNA processing.

Genomic reference files downloaded from UCSC; for example, you can download hg18 from:

http://hgdownload.cse.ucsc.edu/goldenPath/hg18/bigZips/chromFa.zip

The file can be used as a basis for the reference sequence data in LifeScope™ Software tools. A few small transformations are required to prepare the GTF files for use by LifeScope™ Software applications. See "Add new reference files" on page 547 and "Convert a GTF file" on page 550.

A GTF file is an ASCII text file composed of lines of text separated by line feed characters (UNIX-style new lines). Each line of text represents a comment or a tab delimited content about a genomic feature. A comment line always starts with a "#" character. GTF does not specify any format of structured content in comments. Comments containing parsable meta-data should precede any lines containing genomic features. For more detail about the format, refer to the following site:

http://genome.ucsc.edu/FAQ/FAQformat.html#format4

VCF file

The Variant Call Format, in a text format. The file contains meta-information lines, a header line, and data lines that each contain information about a position in the genome. This file format was developed as part of the 1000 Genomes project, and is described here:

http://vcftools.sourceforge.net/specs.html

Recent releases of dbSNP are available in as VCF files. An example is:

ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606/VCF/v4.0/00-All.vcf.gz

Reference sequence data validation

The `reference_validation.pl` script provided with LifeScope™ Software checks and corrects common issues with FASTA sequence files that can cause errors in LifeScope™ Software runs. Examples of errors generated by an uncorrected GTF file can include extra spaces and mixed case.

Concatenation

The contig multi-FASTA reference file can be generated by concatenating the individual chromosome files together.

Select a reference file

Be sure to use the correct reference file type for your requirements. The identifiers in the FASTA definition line are carried through many tools, including the BAM file generated by mapping and pairing, and the GFF files typically written by tertiary analysis tools.

For some downstream viewing and integration tools, for example, the UCSC genome browser, the identifiers can be used to connect sequence references with other sources of annotation.

Read-set file format

An RRS (Read Range Specifier) file, also called a read-set file, is an internal file that defines which reads data is processed in a mapping run. LifeScope™ Software automatically creates the RRS file for your mapping analysis. This format is considered an internal file format and specification for this file format can change in future versions of LifeScope™ Software.

An RRS file allows a mapping analysis to process, in a single run:

- Data from a single XSQ reads file.
- Data from multiple XSQ reads files.
- Data for a range of barcodes (a subset of XSQ file data).
- Data for a range of panels within a barcode (also a subset of XSQ file data).

The RRS file is a tab-separated file with three columns which describe the input XSQ file, the sample name, and optionally a barcode range and panel range (see Table 148 on page 514). The following rules apply to RRS files:

- One read-set (barcode) cannot be part of two samples.
- For a single analysis, mapping can be in color or base. If the mapping is in color, all XSQ files in the RRS file must have color available. If the mapping is in base, all XSQ files in the RRS file must have base available. Mapping fails if these restrictions are not met.
- For barcoded XSQ files, the RRS file can specify a range of barcodes. If some of specified barcodes are missing, mapping will log a warning for the missing barcode IDs.
- One read-set is specified per content line.
- At least one of the read-sets needs to be valid for mapping module to complete.

The requirements for RRS fields are described in Table 148. The read-set file also accepts comments, which contain a '#' character at the beginning of the line.

Table 148 Read-set file description

Column name	Description
Sample name	Multiple read-sets with the same sample name are treated as a single sample (and are processed together). Allowed values: A string of alpha-numeric characters up to 255 chars longs. Spaces are not allowed. Examples: `Human_101`, `Bob`, `Sample19` (This sample name does not need to match any information provided on the instrument.)
File id	A positive integer that distinguishes between XSQ files according to these rules: • Two different XSQ files must have different file IDs. • Two identical XSQ files must have same file ID. (For read-set files created with the UI, LifeScope™ Software assigns this number.)
read-set range URL	A string specifying one or more XSQ data files and optionally also barcode and panel specifiers. The format is a URI scheme: *fileName*?start=*barcodeID.panelstart*&*end=barcodeID.panelend* *fileName*: either a read in the reads repository or the absolute path to an XSQ file on the Linux file system (and that file must be a valid XSQ file for which the user has read permission). *barcodeID*: integers from 1–96. *panelstart*, *panelend*: integers from 1–4000. These fields correspond to panels on the sequencing instrument. The start= section must appear before the end= section. Start *barcodeID* value must be <= the end *barcodeID* value. The *panelstart* value must be <= the *panelend* value.

Example RRS file

The following is a simple example of an RRS file which defines an entire XSQ file as the read-set. The XSQ file in this example has been imported into the reads repository, so an absolute path to the local file system is not required.

```
#SampleName FileID ReadSets
Huref   1      reads/solid0054_20110102_RD_HuRef100_F3.xsq
```

Legacy format translation

Table 149 and Table 150 map the GFF file field name to the corresponding BAM file field name for header fields and alignment data.

Table 149 GFF file header fields

GFF field	Corresponding BAM field	Comments
##reference-name	@SQ UR field	This header field contains the reference file name in the UR field.
##history	—	—
##color-code	—	Only a single color encoding is used.
##primer-base	—	This field is not needed. The primer base is stored with the color read attribute.
##max-num-mismatches	—	—
##max-read-length	@RG LB field	The read lengths in the file are determined from the library type information in the LB field of the read group.
##line-order	@HD SO field	The sort order field in the BAM header indicates the line order. Valid values are: • **None** • **Coordinate** • **Qname**

Table 150 Alignment data

GFF field	Corresponding BAM field	Comments
seqid	RNAME	GFF files use an ordinal number to indicate the reference contig. The ordinal number could be translated to names by a header table. The BAM file uses the @SQ lines to perform a similar lookup.
source	@RG PL	The read group platform (@RG PL field) is used in a manner similar to the SOLiD® GFF source.
type	—	All records are alignments.
start	POS	The 1-based, left-most, top-strand position of the alignment.
end	Calculated from CIGAR	The BAM file does not support an explicit end point. An explicit end point can be calculated from the CIGAR string.
score	—	A read quality is not stored in the BAM alignment record. Mapping and pairing quality is stored in the BAM alignment record.

Table 150 Alignment data *(continued)*

GFF field	Corresponding BAM field	Comments
strand	5th FLAG bit (query strand)	The fifth bit of the FLAG field indicates the strand of the alignment.
GFF attributes		
aID (*bead id*)	QNAME	The bead id.
at (*adaptor type*)	Calculated value	The adaptor type or primer set field can be calculated as described in "Calculation of tag names" on page 505.
b (*bases*)	SEQ	Base space representation of the aligned read. Bases are all uppercase. Equal signs "=" are not used.
c (*category*)	2nd FLAG bit (proper pair) and calculation	The most commonly-used category value, AAA, corresponds to the proper pair bit in the FLAG field. Use POS, MPOS, and strand bits in the FLAG field to calculate other categories. You can find C** pairs (different contigs) by interrogating the mate reference (MRNM). The D** is indicated by the fourth FLAG bit.
g (*color-space read*)	CS attribute	The color read is stored in the CS attribute. However, unlike the 'g' GFF attribute, the color read is stored in an unaltered CSFASTA form that includes the last primer base.
mq (mq [mapping quality])	MAPQ (or SM for pairs)	The mapping quality is stored in the MAPQ field for fragment libraries and in the SM attribute for paired results.
o (*offset*)	CIGAR (hard-clipping)	The offset for an alignment can be computed by checking the number of hard clip (H) operations on the CIGAR string.
p (*mappability ambiguity*)	—	Use mapping quality for uniqueness determinations.
pq (*pairing quality*)	MAPQ (for pairs)	The pairing quality is stored in the MAPQ field for paired results.
q (*color qualities*)	CQ	The CQ attribute stores color quality. The values are not represented as integers, but are represented by ascii-33 encoding.
r (*reference color at mismatches*)	Calculate reference color at mismatches using the samtools `fillmd` command.	CIGAR strings do not distinguish between matches and mismatches. Calculate this information using the samtools `fillmd` command, though in a form that is different from the rb GFF attribute.
s (*color annotations*)	—	—
u (*accumulated mismatch count*)	Per-alignment mismatches supported with CM attribute	The CM attribute specifies the number of color mismatches for the current alignment. You can use a combination of the number of color mismatches across alignments in the BAM file to reconstruct values like the 'u' attribute.

Legacy CMAP file format description

CMAP is a legacy file format normally used to describe the chromosomes of an organism. A CMAP file provides a mapping between:

- Individual FASTA reference files
- An integer index
- A short name

Additional annotation is added in subsequent columns. File paths can be relative or absolute. FASTA reference files must be single entry. See Table 133 for a description of CMAP file fields.

CMAP files generated by the CNV tool contain additional information, pointing to the mappability files for each chromosome arm. See Table 134 on page 445 for a description of the Human CNV CMAP file fields.

The CMAP file may begin with one or more comment lines beginning with a pound sign (#). The comments can include file variables. Variable names use dot separated syntax and must appear immediately after the pound sign. Variable values are separated from the name by an equal sign (=) which may not be surrounded by white-space. The following fields are currently supported:

- data.dir – You can use this optional field to define a root directory. If defined, relative file paths used in the data columns are relative to the root directory.
- Header – This optional field may be used to define column headers for the columns. Column headers should be tab-separated.

Table 151 CMAP file parameters

Column	Field name	Type	Example	Description
1	Seq id	Non-negative integer	2	The seq id is an ordinal index of the sequence.
2	Seq name	String	2	The name of the sequence. The name is typically a chromosome name and might include X, Y,M, etc. The *.chr suffix is added to the sequence name in a number of tools to support the UCSC browser function.
3	Fasta file path	String (path)	—	The path to a FASTA record containing a single sequence entry. The path can be relative or absolute. If the path is relative, you must define the data.dir field in the comments.
4	Double-encoded file path	String (path)	—	If there is a double-encoded version of the FASTA reference file, the path is specified here. Like the FASTA file path, if a relative path is provided, data.dir must be specified.
5, 6, 7, etc.	Additional data	Any	—	Any number of additional fields can be specified after the first four columns. These fields can be integers, strings, file paths, etc. If additional fields are file paths, the rules for relative file paths apply.

D XSQ Tools

This appendix covers:

- Overview 519
- The XSQ Tools package 520
- Conversion to the XSQ format 521
- Conversion from the XSQ format 525
- XSQ file splitting 526
- XSQ index reassignment 526
- HDF5 tools 527
- Internal conversion parameters 528
- Links to resources 533
- FAQ - XSQ Tools 533

Overview

As sequencing technologies evolve and their output capacity increases, large amounts of new sequence data are straining current data storage and analysis systems in the genomics community. One approach to mitigating this pain is to store the instrument data in a more compact format, replacing the older text-based formats — FASTQ, CSFASTA, and QUAL — with more efficient binary encoding. The Extensible Sequence (XSQ) file format has been developed to store each call and quality value in a single byte, which results in file sizes that are up to 75% smaller. There are many other benefits of this format, including embedded metadata, hierarchical structure of the file allowing simplified access to its contents, and the ability to store parallel datasets for easier integration of results.

Each new data format, however, comes with costs. Learning the intricacies of a new format can be easier with full descriptions and simplified examples. Integrating the new format with legacy data or legacy workflows can be more challenging, but this can be mitigated through the use of appropriate data format converters. The goal of the XSQ Tools package is to lower the activation energy associated with adopting the XSQ format by providing documentation, examples, and converters.

For command-line users, the XSQ Tools package is downloaded and run separately from LifeScope™ Software. The LifeScope™ Software UI handles the file conversion automatically.

Utilities in the XSQ Tools package perform these functions:

- For SOLiD® 4 System users, convert CSFASTA format files into the XSQ format required by LifeScope™ Software.
- For 5500 Series Genetic Analysis Systems users running BioScope™ Software or third-party tools, convert XSQ files into the CSFASTA format required by BioScope™ Software.
- For LifeScope™ Software users, optionally split an XSQ file into multiple XSQ files, each containing the data for one library.

More information about XSQ files and the XSQ file format is available on the SOLiD® Software Tools website:

http://solidsoftwaretools.com/gf/project/xsq

This website provides:

- The XSQ file format specification document
- Slides describing XSQ and its relation to the 5500 Series Genetic Analysis Systems
- A link to the XSQ Tools package

Table 152 lists the platforms supported by the XSQ Tools package.

Table 152 Supported platforms for the XSQ conversion tools

Platform	Conversion type	
	To XSQ	From XSQ
Linux® CentOS 4.7	Yes	No
Linux® CentOS 5.5	Yes	Yes

Note: An XSQ script is included in the files installed with LifeScope™ Software. This script is for internal use only, and cannot be used convert your XSQ files.

The XSQ Tools package

Package components

The following components are included in the XSQ Tools package:

- The `convertToXSQ.sh` wrapper script, which sets paths and executes tools for converting from CSFASTA and QUAL to XSQ.
- The `convertFromXSQ.sh` wrapper script, which sets paths and executes tools for the following:
  - Convert from XSQ to CSFASTA and QUAL.
  - Convert from XSQ to FASTQ.
  - Split an indexed (barcoded) XSQ file by library into new XSQ files.
  - Filter an XSQ file.
- The `indexReassignment.sh` wrapper script, which supports library index reassignment for an XSQ file.
- Operating system-specific software libraries to support standalone executables.
- Example data sets, which show the data format using simulated reads. (These do not represent data generated by an instrument). We recommend you use the example data to verify your setup and to practice a conversion. This data is based on chromosome 6.

- Example usage scripts.
- End User License Agreement (EULA).
- Third-party licenses.
- Documentation.

Download instructions

To download the XSQ Tools package, visit the following site:

http://solidsoftwaretools.com/gf/project/xsq

You must accept the EULA and follow the download directions on the site. After downloading the file, follow these steps to install the XSQ Tools package:

1. Move the file to an appropriate location.
2. Change directory to the tarball location.
3. Execute the command `tar -xvfz XSQ_Tools.tgz`. This creates a directory named `XSQ_Tools` with the package contents.
4. Add the `XSQ_Tools` directory to your path, with one of the following commands:
   - bash: `export PATH=$PATH:$`*`tarlocation`*`/XSQ_Tools`
   - csh/tcsh: `set PATH = ( $PATH:$`*`tarlocation`*`/XSQ_Tools )`

   where *tarlocation* is the directory containing the package tarball.

Conversion to the XSQ format

This section describes converting CSFASTA and QUAL files from earlier SOLiD® Systems to the XSQ format. This conversion is also supported in the LifeScope™ Software UI.

The `convertToXSQ.sh` script converts a pair of CSFASTA and QUAL files to an XSQ file. The command options vary by library type, as shown below:

- To convert SOLiD® fragment data:

```
sh convertToXSQ.sh \
    --mode=Fragment \
    --c1=<csfastafile> --q1=<qualfile> \
    --xsqfile=<output XSQ path and filename> \
    --libraryName=<LibraryName> \
    --laneNumber=<LaneNumber> \
    --runStartTime="yyyy-mm-dd hh:mm:ss"
```

- To convert SOLiD® mate-pair *or* paired-end data:

```
sh convertToXSQ.sh \
    --mode=LMP  OR  --mode=PE \
    --c1=<csfastafile> --q1=<qualfile> \
    --c2=<csfastafile> --q2=<qualfile> \
    --libraryInsertSizeMinimum=<Minimum insert size> \
    --libraryInsertSizeMaximum=<Maximum insert size> \
    --xsqfile=<output XSQ path and filename> \
```

```
    --libraryName=<LibraryName> \
    --laneNumber=<LaneNumber> \
    --runStartTime="yyyy-mm-dd hh:mm:ss"
```

- To convert SOLiD® barcoded fragment data (a single barcoded fragment data set):

```
sh convertToXSQ.sh \
    --mode=BC \
    --c1=<csfastafile> --q1=<qualfile> \
    --bc1=<index csfastafile> --bq1=<index qualfile> \
    --xsqfile=<output XSQ path and filename> \
    --libraryName=<LibraryName> \
    --laneNumber=<LaneNumber> \
    --runStartTime="yyyy-mm-dd hh:mm:ss"
```

- To convert SOLiD® barcoded paired-end data (a single barcoded paired-end data set):

```
sh convertToXSQ.sh \
    --mode=BCPE \
    --c1=<csfastafile> --q1=<qualfile> \
    --c2=<csfastafile> --q2=<qualfile> \
    --bc1=<index csfastafile> --bq1=<index qualfile> \
    --libraryInsertSizeMinimum=<Minimum insert size> \
    --libraryInsertSizeMaximum=<Maximum insert size> \
    --xsqfile=<output XSQ path and filename> \
    --libraryName=<LibraryName> \
    --laneNumber=<LaneNumber> \
    --runStartTime="yyyy-mm-dd hh:mm:ss"
```

Table 153 lists the options required by the `convertToXSQ.sh` script. Table 154 on page 523 lists the remaining options.

Table 153 Required arguments for the `convertToXSQ.sh` script (to create an XSQ file)

Option	Required with...	Description
--mode	All	Allowed values are Fragment, LMP, PE, BC, BCPE: • **Fragment**: Convert fragment CSFASTA and QUAL files to XSQ format. • **LMP**: Convert mate-pair CSFASTA and QUAL files to XSQ format. • **PE**: Convert paired-end CSFASTA and QUAL files to XSQ format. • **BC**: Convert single barcoded fragment CSFASTA/QUAL with index information to XSQ format. • **BCPE**: Convert single barcoded paired-end CSFASTA/QUAL with index information to XSQ format.
--libraryName	All	Name of the library preparation.
--laneNumber <*arg*>	All	Lane number in which this sequencing sample was loaded on the flowchip.

Table 153 Required arguments for the `convertToXSQ.sh` script (to create an XSQ file) *(continued)*

Option	Required with...	Description
--runStartTime <*arg*>	All	Time when the instrument run was executed. This information is entered into the output XSQ file. Use one of these formats: • `'yyyy-mm-dd hh:mm:ss'` (including the quotes, either double or single). • `yyyy-mm-dd\ hh:mm:ss` (escape the space with a backslash).
--xsqfile <*arg*> *or* -x <*arg*>	All	Requested name of output XSQ file. The name must include the extension `.xsq`.
--c1 <*arg*>	All SOLiD® System data	The CSFASTA input file for the first tag.
--q1 <*arg*>	All SOLiD® System data	The QUAL input file for the first tag.
--c2 <*arg*>	LMP, PE, or BCPE SOLiD® System data	The CSFASTA input file for the second tag.
--q2 <*arg*>	LMP, PE, or BCPE SOLiD® System data	The QUAL input file for the second tag.
--bc1 <*arg*>	BC or BCPE SOLiD® System data	The index (barcoded) CSFASTA input file.
--bq1 <*arg*>	BC or BCPE SOLiD® System data	The index (barcoded) QUAL input file.
libraryInsertSizeMinimum	LMP or PE SOLiD® System data	The expected minimum insert size, in nucleotides.
libraryInsertSizeMaximum	LMP or PE SOLiD® System data	The expected maximum insert size, in nucleotides.

Table 154 lists options for the `convertToXSQ.sh` script. See Table 153 on page 522 for the required options.

Table 154 Optional arguments for the `convertToXSQ.sh` script (to create an XSQ file)

Option		Description
-a <*arg*>	--application <*arg*>	The type of analysis intended for this input data. Allowed values are listed below. Quotes are required around values that contain spaces. • Whole genome resequencing • Targeted resequencing • Whole transcriptome (Fragment) • Whole transcriptome (PairedEnd) • Small RNA • ChIP-Seq • Methylation

Table 154 Optional arguments for the `convertToXSQ.sh` script (to create an XSQ file) *(continued)*

Option		Description
	--assembly <*arg*>	The intended assembly for mapping. May be blank if unknown. Examples: • hg18 • hg19 • mm9 Other assembly names are allowed. Spaces are not allowed in these names.

-c <*arg*>		Use the configuration file <*arg*> as a list of conversion parameters. The parameters are specified as nested key value pairs. Example file content:
		`q1=/data/results/reads/uhr.150.F3.qual`
		`c1=/data/results/reads/uhr.150.F3.csfasta`
		`x=./out.xsq`
		`mode=Fragment`
		`runStartTime=2011-04-01\ 12:01:02`
		`libraryName=150_UHR`
		Option hyphens ('-', '--') are not used in the configuration file.

	--comments	Additional information about this sample or its preparation.
-d <*arg*>	--description <*arg*>	A description of the sample.
-e	--ercc	Whether internal controls have been used. Reserved for future use.
	--flowCellAssignment <*arg*>	The position of the instrument flowcell in which this flowchip was placed.
-h	--help	Display command usage and options.
	--instrumentName <*arg*>	The name of the instrument.
	--instrumentSerial <***arg***>	The unique serial number identifying the instrument.
-k <*arg*>	--indexKitName <*arg*>	The name of the kit used to add the indexing nucleotides.
	--laneNumber <*arg*>	Lane number in which this sequencing sample was loaded within the flowchip layout.
	--libraryName <*arg*>	The name of library preparation.
-n	--noqualfile	Use this attribute to specify that the QUAL file is not available.
-o <*arg*>	--operator <*arg*>	The name or ID of person who set up the instrument run.
-p <*arg*>	--projectName <*arg*>	Project within which this sample is being run. This value is entered into the projectName field in the libraryDetails table, and may be different from the LifeScope™ Software project name.
	--runEndTime <*arg*>	Time when the instrument run completed. This information is entered into the output XSQ file. Use one of these formats: • `'yyyy-mm-dd hh:mm:ss'` (including the quotes, either double or single). • `yyyy-mm-dd\ hh:mm:ss` (escape the space with a backslash).
	--runName <*arg*>	The name of a flowchip run.
	--sampleIdentifier <*arg*>	The unique name for this sample.
	--sampleOwner <*arg*>	The owner of this sample.
	-sequencingSampleDescription <*arg*>	Description of the sample loaded in a lane on a flowchip.
	--sequencingSampleName <*arg*>	The name of the sample loaded in a lane on a flowchip.

Table 154 Optional arguments for the `convertToXSQ.sh` script (to create an XSQ file) *(continued)*

Option		Description
	--species <*arg*>	The intended species for mapping. Example values include: • Homo sapiens • Mus musculus • other
-V	--version	Print version information.

Conversion from the XSQ format

This section describes converting XSQ format files from the 5500 Series Genetic Analysis Systems into CSFASTA and QUAL files, in the format required by earlier version of BioScope™ Software. This section does not apply to LifeScope™ Software users who are analyzing XSQ data files generated by the 5500 Series Genetic Analysis Systems .

The conversion program is available as a standalone script to be run on the Linux shell. This conversion is not available within the LifeScope™ Software UI or the LifeScope™ Software command shell.

If the input XSQ file includes base-space data, the conversion also exports the base-space data into a FASTQ file.

Conversion syntax

The `convertFromXSQ.sh` script converts an XSQ file into a pair of CSFASTA and QUAL files. The usage for this script is:

```
convertFromXSQ.sh -c xsqfile
```

where *xsqfile* gives an existing XSQ file on the file system.

The converted files are created in the following directory:

./Libraries/<*LibraryName*>/<*TagName*>/reads/

The output files are named according to the following pattern:

<xsqfile>_<LibraryName>_<TagName>.ext

where *ext* has the following values:

- `.csfasta`
- `.QV.qual`
- `.fastq`

FASTQ files are created only if the input XSQ file contains base-space data.

Filter option

Use the `-f` option to filter (remove) marked reads during conversion, with this version of the command:

```
convertFromXSQ.sh -c -f xsqfile
```

In the XSQ file, each read contains a field of filtering flags, which may mark the read for filtering. Only when the `-f` option is used, reads that are marked for filtering are not written to the CSFASTA file.

By default `convertFromXSQ.sh` ignores the filtering flags and writes all reads to the converted output files. With the `-f` option, the converter applies the filtering flags during conversion, which suppresses the filtered reads and quality values from being written in the output files.

Options table

Table 155 lists the options supported by the `convertFromXSQ.sh` script. These options include functionality for XSQ file-splitting and for version information.

Table 155 Options for the `convertFromXSQ.sh` script

Option	Description
-c	Converts the input XSQ file into CSFASTA, QUAL, and FASTQ files.
-f	(*Optional*) During conversion, removes reads that are marked for filtering.
-o	(*Optional*) Specifies an output directory partial path.
-s	Splits the input XSQ file into multiple output files, one for each library in the parent XSQ file.
-v	Displays the package version.

XSQ file splitting

The `convertFromXSQ.sh` script with the `-s` option splits an input XSQ file into multiple XSQ files, one file per library. For example, if a paired-end library is created with indices for each of lib1, lib2, and lib3 and run in the same lane, a single XSQ file will be created. Using the `-s` option creates three separate XSQ files - one for each of lib1, lib2, and lib3. Each XSQ file contains data for both the F3 and F5 tags.

To split an XSQ file, use the following command:

```
convertFromXSQ.sh -s xsqfile -o outputpath
```

The following output files are produced:

```
outputpath/
    lib1.xsq
    lib2.xsq
    ...
```

Only one input XSQ file is allowed per run.

XSQ index reassignment

Purpose

The `indexReassignment.sh` script supports correcting the index information in an XSQ file (for example, in case incorrect information is entered on the instrument). The process involves these steps:

1. Run the `indexReassignment.sh` script to write a comma-separated values (CSV) file.
2. Manually edit the CSV file to enter the correct index name.
3. Run the `indexReassignment.sh` script to read the CSV file and create an XSQ file with the corrected index information.

The `indexReassignment.sh` script also supports optionally changing the mismatch level.

Usage

This section describes the usage of the `indexReassignment.sh` script.

- To *extract* the index information:

  ```
  indexReassignment.sh -e -xsq example.xsq
  ```

  This command creates a CSV file with the same file name as the input XSQ file but with a `.csv` extension. (You then manually edit the CSV file to correct the index information, before running `indexReassignment.sh` to update the information.)

- To *update* with the corrected index information:

  ```
  indexReassignment.sh -csv example.csv -xsq example.xsq \
      -o example-fixed.xsq
  ```

  This command creates a new XSQ file with the corrected index information. The options have the following meaning:

  `-csv <arg>`: The CSV file edited to correct the index.

  Note: You create the CSV file during the `indexReassignment.sh` extraction step above, and manually edit it to correct the index.

  `-xsq <arg>`: The original XSQ file.

  `-o <arg>`: The output XSQ file, newly-created with the corrected index information.

- To *change* the mismatch level while updating the index:

  ```
  indexReassignment.sh -csv example.csv -xsq example.xsq \
      -o example-fixed.xsq -mm n
  ```

  The `-mm n` option changes the update command to also update the mismatch level. This option redoes the entire indexing assignment and rewrites all library groups. (Allowed values with the `-mm` option are 0 and 1.)

Index reassignment options table

Table 156 lists the options supported by the `indexReassignment.sh` script.

Table 156 Options supported by the `indexReassignment.sh` script

Option	Description
-csv <*arg*>	Update the XSQ file index, reading index information from the CSV file <*arg*>.
-e	Extract current index information.
-mm <*arg*>	Update the mismatch level for index classification.
-o <*arg*>	The output XSQ file, created with the corrected index information.
-xsq <*arg*>	The input XSQ file.

HDF5 tools

The XSQ format conforms to the HDF5 specification, and HDF5 tools work on XSQ files. Figure 53 is an example of how HDFView displays an XSQ file, showing Groups, Tables, and Datasets.

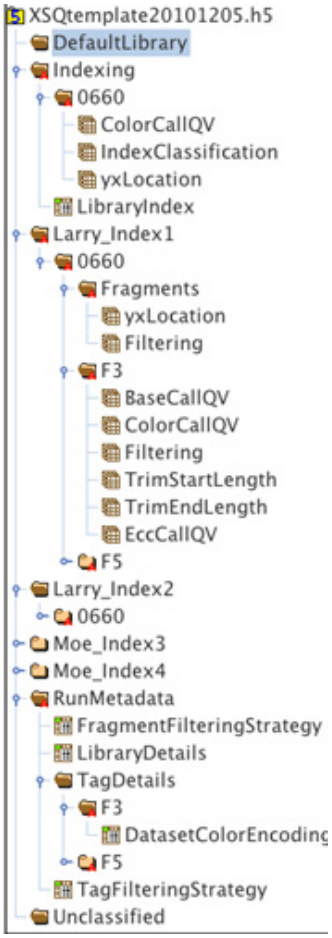


Figure 53 HDFView screenshot of the XSQ file format, showing Groups, Tables, and Datasets

Internal conversion parameters

This section describes the parameters used internally by the LifeScope™ Software UI when converting a file to the XSQ format from the older CSFASTA and QUAL formats used in earlier versions of the software. These parameters are for internal use.

Parameters table

Table 157 lists and describes the conversion parameters used internally by the LifeScope™ Software UI.

Table 157 Internal XSQ conversion parameters

Parameter	Default	Description
xsqconverter.run	—	For internal use.

Table 157 Internal XSQ conversion parameters *(continued)*

Parameter	Default	Description
xsq.application	—	Describes how this sequence is intended to be analyzed. Allowed values are: • Whole genome resequencing • Targeted resequencing • Whole transcriptome (Fragment) • Whole transcriptome (PairedEnd) • Small RNA • ChIP-Seq • Methylation
xsq.assembly	—	The intended assembly for mapping. May be blank if unknown. Example values: • hg18 • hg19 • mm9
xsq.comments	—	Additional information about this sample or its preparation.
xsq.description	—	Sample description.
xsq.ercc	0	Whether or not internal controls have been used. Allowed values: • **0**: No • **1**: Yes
xsq.flowCellAssignment	255	The position of the instrument flowcell in which this flowchip was placed.
xsq.indexKitName	—	Name of the kit used to add the indexing nucleotides.
xsq.input.barcodedCsfasta1	—	Barcoded CSFASTA file (used only in the case of indexed runs).
xsq.input.barcodedQual1	—	Barcoded QUAL file (used only in the case of indexed runs).
xsq.input.csfasta1	—	Input CSFASTA file.
xsq.input.csfasta2	—	Second CSFASTA file (use only in the case of LMP or PE runs).
xsq.input.qual1	—	Input QUAL file.
xsq.input.qual2	—	Second QUAL file (use only in the case of LMP or PE).
xsq.instrumentName	—	Name of the sequencing instrument.
xsq.instrumentSerial	—	Unique serial number to identify a particular sequencing instrument.
xsq.laneNumber	255	Lane number within the flowchip layout in which this sequencing sample was loaded.
xsq.libraryInsertSizeMaximum	—	Expected maximum insert size in nucleotides. (Required for LMP and PE conversion).
xsq.libraryInsertSizeMinimum	—	Expected minimum insert size in nucleotides. (Required for LMP and PE conversion).
xsq.libraryName	—	The name of library preparation.

Table 157 Internal XSQ conversion parameters *(continued)*

Parameter	Default	Description
xsq.mode	—	XSQ mode of operation. Allowed values are: • **Fragment**: Convert fragment CSFASTA and QUAL formats to XSQ. • **LMP**: Convert mate-pair CSFASTA and QUAL formats to XSQ. • **PE**: Convert paired-end CSFASTA and QUAL formats to XSQ. • **BC**: Converting single barcoded fragment CSFASTA and QUAL formats with index information to XSQ. • **BCPE:** Convert single barcoded paired-end CSFASTA and QUAL formats with index information to XSQ.
xsq.noqualfile	false	Specify that the input QUAL file is not available. Allowed values: • **false**: The QUAl file is available. • **true**: The QUAL file is missing.
xsq.operator	—	The name of person or ID who set up the instrument run.
xsq.output.xsqfile	—	The requested name for output XSQ file. Must use the ".xsq" extension.
xsq.projectName	—	Project within which this sample is being run.
xsq.runEndTime	—	Time when the instrument run completed. Required format: `'yyyy-mm-dd hh:mm:ss'` (including quotes).
xsq.runName	—	Name of a flowchip run.
xsq.runStartTime	—	Time when the instrument run started. Required format: `'yyyy-mm-dd hh:mm:ss'` (including quotes).
xsq.sampleIdentifier	—	Unique name for this sample.
xsq.sampleOwner	—	Owner of this sample.
xsq.sequencingSampleDescription	—	Description of the sample loaded in a lane on a flowchip.
xsq.sequencingSampleName	—	Name of sequencing sample loaded in a lane on a flowchip. This value is the bead emulsion loaded on the instrument. The emulsion may be a collection of barcoded libraries from multiple samples.
xsq.species	—	The Intended species for mapping. Example values include: • Homo sapiens • Mus musculus • Other

Internal INI file

The following is an example of an INI file used internally by the LifeScope™ Software UI during a conversion to the XSQ format from the older CSFASTA and QUAL formats used in earlier versions of the software. Refer to the *LifeScope™ Genomic Analysis Software User Guide* (Part no. 4471877) for information about the UI. This INI file is for internal use.

```
##Run parameter
xsqconverter.run=1

##How this sequences is intended to be analyzed
xsq.application=Unknown

##Intended assembly for mapping
xsq.assembly=Unknown
```

```
##Additional information about this sample or its preparation
xsq.comments=Unknown

##Sample description
xsq.description=Unknown

##Whether internal controls have been used (0=No, 1=Yes)
xsq.ercc=0

##The position of the instrument flowcell in which this flowchip
was placed
xsq.flowCellAssignment=255

##Name of the kit used to add the indexing nucleotides
xsq.indexKitName=Unknown

##Barcoded csfasta file (should be used only in case of indexed
runs)
xsq.input.barcodedCsfasta1=

##Barcoded qual file  (should be used only in case of indexed
runs)
xsq.input.barcodedQual1=

##Csfasta file
xsq.input.csfasta1=

##Second csfasta file (should be used only in case of LMP or PE)
xsq.input.csfastac2=

##Qual file
xsq.input.qual1=

##Second qual file (should be used only in case of LMP or PE)
xsq.input.qual2=

##Name of instrument
xsq.instrumentName=Unknown

##Unique serial number to identify a particular instrument
xsq.instrumentSerial=Unknown

##Lane number within the flowchip layout in which this
sequencing sample was loaded
xsq.laneNumber=255

##Expected maximum insert size in nucleotides. (Required for LMP
and PE conversion)
xsq.libraryInsertSizeMaximum=

##Expected minimum insert size in nucleotides. (Required for LMP
and PE conversion)
xsq.libraryInsertSizeMinimum=
```

```
# The name of library preparation
xsq.libraryName=DefaultLibrary

# Allowed values are: Fragment, LMP, PE, BC, BCPE. Use
'Fragement' for converting fragment CSFASTA/QUAL to XSQ. Use
'LMP' for converting mate pair CSFASTA/QUAL to XSQ. Use 'PE' for
converting paired end CSFASTA/QUAL to XSQ. Use 'BC' for
converting  single barcoded fragment CSFASTA/QUAL with index
information to XSQ. Use 'PE' for converting single barcoded
paired end CSFASTA/QUAL with index information to XSQ
xsq.mode=Fragment

##Specify that qual file is not available (Use 'true' - to
specify qual file missing)
xsq.noqualfile=false

##The name of person/ID who set up the instrument run
xsq.operator=Unknown

##Desired name for output XSQ file
xsq.output.xsqfile=pluginfrag.xsq

##Project whithin which this sample is being run
xsq.projectName=Unknown

##Time when run completed. The run end time should be in 'yyyy-
mm-dd hh:mm:ss' format
xsq.runEndTime=

##Name of a flowchip run
xsq.runName=Unknown

##Time when run was executed. The run start time should be in
'yyyy-mm-dd hh:mm:ss' format
xsq.runStartTime=2011-01-11 20:10:20

##Unique name for this sample
xsq.sampleIdentifier=Unknown

##Owner of this sample
xsq.sampleOwner=Unknown

##Description of the sample loaded in a lane on a flowchip
xsq.sequencingSampleDescription=Unknown

##Name of sample loaded in a lane on a flowchip
xsq.sequencingSampleName=Unknown

##Intended species for mapping
xsq.species=Unknown
```

Links to resources

This section lists resources related to XSQ file format and converters.

For the XSQ Tools package and documentation, XSQ file format specification, XSQ webinar slides, and example XSQ files, visit this site:

http://solidsoftwaretools.com/gf/project/xsq/

HDFView is a Java® tool for browsing HDF-based files, including XSQ files. For information about HDFView, visit this site:

http://www.hdfgroup.org/hdf-java-html/hdfview/

HDF5 is a technology suite for managing of large, complex data collections. HDF5 APIs are available at this site:

http://www.hdfgroup.org/HDF5/release/obtain5.html

Exact Call Chemistry (ECC) is a unique ligation-based sequencing methodology, introduced with the 5500 Series Genetic Analysis Systems . The Exact Call Chemistry white paper is available through this link and the instructions that follow:

http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing/publications-literature.html

1. Under the Publication & Literature section, click the Product Literature tab.
2. Scroll down to the White Papers section.
3. Click on the "SOLiD® System Accuracy with the Exact Call Chemistry Module" link.

FAQ - XSQ Tools

1

What software accepts the XSQ file format?

Initially, only LifeScope™ Software supports the new format. Life Technologies Corporation is working with third-party developers to adapt their workflows to support the new chemistry and data format.

2

I have pipelines that require CSFASTA and QUAL files as input. What should I do?

Life Technologies Corporation provides tools on the SOLiD® Software Tools website to convert XSQ files into CSFASTA and QUAL files. See "Links to resources" on page 533, and download the XSQ Tools package.

3

I have pipelines that require FASTQ files as input. What should I do?

When the ECC module is used during sequencing, base-space data is available in the XSQ file and can be exported into a FASTQ file. See "Conversion from the XSQ format" on page 525.

4

Can I use data from both a SOLiD® 4 System and a 5500 Series Genetic Analysis Systems for data analysis?

Yes. There are two options and both involve converting the data into a common format, either XSQ or CSFASTA+QUAL. One way is to convert the SOLiD® 4 System CSFASTA and QUAL output to the XSQ format (see "Conversion to the XSQ format" on page 521). Then supply both the converted XSQ file and the 5500 XSQ file as input to your LifeScope™ Software analysis.

Another method is to convert the XSQ data into CSFASTA+QUAL for use with older analysis tools.

5

Is there a converter to change CSFASTA and QUAL files to the XSQ format?

Yes, standalone converters are provided in the XSQ Tools package at the following site:

http://solidsoftwaretools.com/gf/project/xsq/

6

How does the XSQ format handle multiplexing?

Multiple libraries may be included in a single XSQ file, such as are generated for each lane of a SOLiD® 5500 Series Sequencer run. When multiple different users are providing samples to be run in the same lane, the resulting file may be split by library for distribution to individual users, without sharing all of the data with each user. The `convertFromXSQ.sh` script provides the `-s` splitting option to separate libraries into separate XSQ files when necessary. See "XSQ file splitting" on page 526.

Note: The `-s` splitting option only works for indexing files. Non-indexing files already contain only a single library (and are not split with the `-s` option).

The `convertToXSQ.sh` script is run once for each barcode index, and each run results in a single XSQ file. Multiple barcode indices cannot be joined in a single XSQ file with the current converter.

7

Are APIs available for accessing XSQ data?

Yes. HDF5 APIs work with the XSQ format, and are available at this site:

http://www.hdfgroup.org/HDF5/release/obtain5.html

8

Is an XSQ file viewer available?

Yes. XSQ data is can be viewed with the HDFView browser, which can be obtained at this site:

http://www.hdfgroup.org/hdf-java-html/hdfview/

9

Can XSQ files be converted in the LifeScope™ Software UI and command shell?

Yes. The LifeScope™ Software UI supports the conversion of CSFASTA and QUAL files to the XSQ format. If you are running the LifeScope™ Software command shell, use the standalone XSQ conversion tool that is part of the XSQ Tools package.

10

Is library index correction supported?

Not in the current version. An XSQ library index reassignment tool is under consideration.

11

Why does the converter not work on my variant of Linux.

The table in the "Overview" on page 519 section describes that conversion *to* XSQ requires CentOS 4.7 or later, and conversion *from* XSQ requires CentOS 5.5 or later. Other versions of Linux are not supported.

12

What are the reserved quality values (QVs) in the XSQ file?

From the XSQ File Format Specification:

> For the QV, the range of the 6 bits is between 0 and 63. Valid quality values will be from 3 to 62, represented as PHRED scores [QV = -10*log10P(error)]. 0 and 1 are reserved for future usage; 2 indicates a missing quality value (not trimming); 63 refers to an uninformative or missing call.

13

What values should I use for libraryInsertSizeMinimum and libraryInsertSizeMaximum when they are not known during the conversion of mate-pair and paired-end files?

Any values will work as long as `libraryInsertSizeMinimum` is less than or equal to `libraryInsertSizeMaximum` and both values are nonnegative.

14

How can I collect statistics on an XSQ file?

Not supported currently. An XSQ Statistics utility to extract the number of reads by file, barcode, tag, etc., in addition to library type and read length, is under consideration.

15

How can I tell if my XSQ file is valid or corrupt?

Not supported currently. An XSQ validator utility is under consideration.

16

I have a CSFASTA file to convert to XSQ format, but I do not have the QUAL file. Can I create the XSQ file?

When converting CSFASTA to XSQ, if the QUAL file is missing or not found, all reads are marked for filtering in the output XSQ file. As a result, the reads are ignored (not processed) when used as input into LifeScope™ Software analyses.

The workaround is to create a dummy QUAL file to use as input with the `convertToXSQ.sh` script. Create a QUAL file with the same bead ids (and in the same order) as the CSFASTA file. Use QV values at least high enough to pass default thresholds.

18

Where does the tool get the indexing kit information? Is it stored in the XSQ file or does it come from the XSQ_Tool package?

This information is distributed within XSQ Tools. See `XSQ_Tools/xsqreindexer/conf` for details.

19

If a wrong kit was used to assign the original XSQ file, can the tool reassign it to a different, correct kit? For example, does the tool support reassignment from a 16-index kit to a 96-index kit?

No. Kit reassignment is not supported here because the index calls are of a different length (5 vs. 10 calls).

20

The 5500 ICS v1.0 and v1.1 only uses 4 colors of the 5-color 16 barcode kit. Can the index reassignment tool use all 5 colors for reassignment?

Currently the index reassignment tool is using the same underlying library, so for now the same issue is present for the 16 index library. This is not a problem with the 96 index libraries.

21

How can I tell if an XSQ file was generated from an ECC run?

As this is a property of each tag and may be different between tags, the `IsBasePresent` attribute is most informative. See `/RunMetadata/TagDetails/<TagName>/IsBasePresent` in the XSQ file.

E The Reference Repository

This appendix covers:

- Overview 539
- Repository structure 539
- Repository location 540
- Reference file conflicts 540
- Initial repository contents 541
- Add new reference files 547
- Prepare reference files 550

Overview

This appendix describes the LifeScope™ Genomic Analysis Software reference repository. The repository is created during the pre-installation and installation process, using files shipped on a separate data drive.

You can add new reference files to the repository at any time after installation, but you must follow restrictions and conventions described in this appendix.

Repository structure

The files on disk that make up the reference repository have the following top-level organization:

- **lifetech –** Contains files that have been prepared or formatted to support customizations within LifeScope™ Software
- **external –** Contains files that have been downloaded and placed here verbatim, such as annotations from public repositories.
- **internal –** Intended to hold locally derived files to support new genomes and annotations. The `internal` structure mirrors the `lifetech` directory.

The next directory level are assembly names such as `hg18` and `hg19`.

For more detail see "Initial repository contents" on page 541.

Repository location

The location (path) of the reference repository is set during the pre-installation and installation process. The location is determined by where the data drive contents are copied, and must match the location answered to the installer question "Enter the location for reference files."

The reference repository location after installation is given by the parameter `lifescope.references` in the file `<installdir>/server/server.properties`. For example:

```
# Define reference directory
lifescope.references=/data/results/referenceData
```

The location `/data/results` is an example only. The reference repository location is set during "Copy data drive content" on page 41 in Chapter 2, LifeScope™ Genomic Analysis Software Installation, and is the location `<LSDF>/referenceData`, where `<LSDF>` is the LifeScope™ Software Data Folder that the LifeScope™ Software administrator creates during the "Copy data drive content" step.

Reference file conflicts

In order to avoid conflicts with reference files, follow the instructions in "Add new reference files" to create a new folder when adding references for a new genome to the repository.

The behavior for a `set reference assembly_name` command, such as `set reference hg19`, is the following:

- LifeScope™ Software first looks in the reference repository for the *`assembly_name`* folder, in the `lifetech`, `internal`, and `external` directories, in that order. The search stops when the first match is found.
- The software searches in the *`assembly_name/reference`* folder for a genomic reference files that includes the assembly name in its file name
- In the *`assembly_name`* folder, the software searches for reference files that include the assembly name in its file name.
- If a file with the assembly name is not found, or if multiple files are found that both have the assembly name and have the expected file extension, the software then uses the first file (alphabetically) in that folder.

  If you do not follow the recommended conventions for adding reference files to the repository, the reference selected may or may not be the one intended.

Assembly names and reference parameters

Users are recommended to use the assembly name option with the `set reference` command, when possible. Using a recognized assembly name automates the specification of reference-related parameters. See Table 13 on page 89 for more information on the `set reference` command.

Before starting your analysis, review the reference-related parameters to confirm that they are correctly set. (Run the LifeScope™ Software shell `ls` command in the analysis directory to list the parameter settings.)

Initial repository contents

The initial contents of the repository are created from the data drive during the installation process ("Copy data drive content" on page 41).

The following tables describe the files and directories that make up the initial reference repository:

- `lifetech/hg19` – Table 158
- `lifetech/hg18` – Table 159
- `external/hg18` and `hg19` – Table 160
- `external/uc01` – Table 161
- `internal/uc01` – Table 162

See "Add new reference files" on page 547 to add new reference files to the repository.

Note: Modifying files under the lifetech directory of the reference repository is not supported. Header and other information in the reference file is case sensitive. For example, changing the case of a contig name in a reference file causes program failures. Adding files under the lifetech directory also is not supported. Changing the information in any respect is not supported.

Table 159 describes the reference files in the `lifetech/hg19` directory.

Table 158 Contents and purpose of `lifetech/hg19` reference files and directories

lifetech/hg19 file or directory	Description
hg19/human_filter_reference.fasta	The file human_filter_reference.fasta is a collection of sequences that are used to filter out commonly occurring motifs and contaminating sequences prior to mapping.

Table 158 Contents and purpose of `lifetech/hg19` reference files and directories *(continued)*

lifetech/hg19 file or directory	Description

hg19/reference	This directory contains a validated FASTA file describing the reference sequence for human hg19: `human_hg19.fa`
	An index file with the following name has also been created to describe each entry: `human_hg19.fa.fai`
	This file contains five columns, with one row for each entry:
	1. Chromosome or contig name (no spaces)
	2. Chromosome or contig length
	3. Cumulative chromosome or contig length
	4. Number of characters per line
	5. Line width (including carriage return)
	The other files are all derivatives of the original file and contain the hash tables used for mapping:
	`human_hg19.fa.100111010011010011101`
	`human_hg19.fa.101100111110111101`
	`human_hg19.fa.101101110111011101`
	`human_hg19.fa.110100111010011100100111`
	`human_hg19.fa.111010011010011101001`
	`human_hg19.fa.111010011010011101010011`
	`human_hg19.fa.111011101101111011`
	`human_hg19.fa.1111110101110101101`

hg19/refGene	RefGene files are downloaded directly from UCSC, from this website: **http://hgdownload.cse.ucsc.edu/goldenness/hg19/database/refGene.txt.gz** These files have this filename pattern: refGene.${ORGAMISM}.${DATE}.gtf. RefGene files are converted to GTF (preserving the filename pattern): refGene.hg19.20101221.gtf A refGene file is then used, in conjunction with the reference FASTA file, referenceData/lifetech/hg19/reference/human_hg19.fa to generate new FASTA files for mapping transcripts to exons and junctions: referenceData/lifetech/hg19/refGene/refGene.hg19.20101221.exon.fa referenceData/lifetech/hg19/refGene/refGene.hg19.20101221.junctions.fa
hg19/dbSNP/dbSNP_b132_00-All.vcf	This file is from the National Center for Biotechnology Information (NCBI) dbSNP database, which contains information on SNPs and indels already found by other studies. Recent releases of dbSNP are available in VCF (Variant Call Format), such as this file for Human: **ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606/VCF/v4.0/00-All.vcf.gz** The VCF file format was developed as part of the 1000 Genomes project. The format is described at this website: **http://vcftools.sourceforge.net/specs.html**

Table 158 Contents and purpose of `lifetech/hg19` reference files and directories *(continued)*

lifetech/hg19 file or directory	Description
hg19/mirbase/hsa.gff	The microRNA database annotations are derived directly from the miRBase website: **http://www.mirbase.org/ftp** The species-specific precursor GFF file is downloaded from this page: **http://www.mirbase.org/ftp.shtml** which resolves into a file such as this (for human): **ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff**
hg19/CNV/mappability/fragment_50.h5	The CNV hg19 mappability file.

Table 159 describes the reference files in the `lifetech/`hg18 directory.

Table 159 Contents and purpose of `lifetech/hg18` reference files and directories

lifetech/hg18 file or directory	Description
hg18/human_filter_reference.fasta	The file human_filter_reference.fasta is a collection of sequences that are used to filter out commonly occuring motifs and contaminating sequences prior to mapping.

hg18/reference	This directory contains a validated FASTA file describing the reference sequence for human hg18: `human_hg18.fa`
	An index file with the following name has also been created to describe each entry: `human_hg18.fa.fai`
	This file contains five columns, with one row for each entry:
	1. Chromosome or contig name (no spaces)
	2. Chromosome or contig length
	3. Cumulative chromosome or contig length
	4. Number of characters per line
	5. Line width (including carriage return)
	The other files are all derivatives of the original file and contain the hash tables used for mapping:
	`human_hg18.fa.111011101101111011`
	`human_hg18.fa.101101110111011101`
	`human_hg18.fa.11010011101001110100111`
	`human_hg18.fa.101100111110111101`
	`human_hg18.fa.11101001110100111101001`
	`human_hg18.fa.11101001110100111010011`
	`human_hg18.fa.10011101001110100111101`
	`human_hg18.fa.1111110101110101101`

Table 159 Contents and purpose of `lifetech/hg18` reference files and directories *(continued)*

lifetech/hg18 file or directory	Description
hg18/refGene	RefGene files are downloaded directly from UCSC, from this website: **http://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz** These files have this filename pattern: refGene.${ORGAMISM}.${DATE}.gtf. RefGene files are converted to GTF (preserving the filename pattern): refGene.hg18.20090513.gtf A refGene file is then used, in conjunction with the reference FASTA file, referenceData/lifetech/hg18/reference/human_hg18.fa to generate new FASTA files for mapping transcripts to exons and junctions: referenceData/lifetech/hg18/refGene/refGene.hg18.20090513.exon.fa referenceData/lifetech/hg18/refGene/refGene.hg18.20090513.junctions.fa
hg18/dbSNP/dbSNP_b130.tab	This file is from the National Center for Biotechnology Information (NCBI) dbSNP database, which contains information on SNPs and indels already found by other studies. This file contains the data from dbSNP version 130, and has the following tab-separated fields: 1. rsID, the refSNP identifier 2. Chromosome number 3. Chromosome start coordinate 4. Chromosome end 5. Is SNP? 6. Function code (consequence to transcript if any) The function code is a comma separated list of pairs: locus_id:function code). The FunctionCode attribute is valid only for SNPs and small indels, and are used to annotate SNPs. Codes are based on functional codes from dbSNP, or refSNPs in gene features (from the dbSNP handbook).
hg18/mirbase/hsa.gff	The microRNA database annotations are derived directly from the miRBase website: **http://www.mirbase.org/ftp** The species-specific precursor GFF file is downloaded from this page: **http://www.mirbase.org/ftp.shtml** which resolves into a file such as this (for human): **ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff**
hg18/CNV/mappability/ fragment_25.h5 fragment_50.h5 matepair_50x50.h5	The CNV hg18 mappability files.

Table 161 describes the hg18 and hg19 reference files in the `external` directory.

Table 160 Contents and purpose of `external` `hg18` and `hg19` reference files and directories

external file or directory	Description
hg18/ucsc/refGene.txt.gz, hg19/ucsc/refGene.txt.gz	RefGene files are downloaded directly from UCSC, from these websites: **http://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz** **http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/refGene.txt.gz**
hg18/targetedEnrichment/ targetRegionsHg18.bed hg19/targetedEnrichment/ targetRegionsHg19.bed	Targeted Regions files are specific to each target capture procedure used during the sample preparation process before the library generation step. As there are many ways to enrich for targeted regions, including custom preparations, these files are to be provided by the user. This directory is the default location for such files, and BED format is typically used here. Here are example locations and filenames: external/hg18/targetedEnrichment/targetRegionsHg18.bed external/hg19/targetedEnrichment/targetRegionsHg19.bed The BED file format was developed to support the UCSC Genome Browser, and the format is described here: **http://genome.ucsc.edu/FAQ/FAQformat.html#format1** **Note:** The target regions BED files on the data drive are 0-length placeholders.
hg18/mirbase/miRNA.txt, hg19/mirbase/miRNA.txt	The microRNA database annotations are derived directly from the miRBase website: **http://www.mirbase.org/ftp** The species-specific mature forms are downloaded together as an Excel document: **ftp://mirbase.org/pub/mirbase/CURRENT/miRNA.xls** and written as text files on the data drive: referenceData/external/hg18/mirbase/miRNA.txt referenceData/external/hg19/mirbase/miRNA.txt **Note:** The shell command `set workflow hg18` (or `hg19`) incorrectly sets the `analysis.mirbase.mature.file` parameter. You must manually set the parameter correctly *after* the `set workflow` command. **Note:** The `hsa.gff` files are included in the `lifetech/hg18/mirbase` and `lifetech/hg19/mirbase` directories.

Table 161 describes the uc01 example reference files in the `external` directory.

Table 161 Contents and purpose of `external` `uc01` reference files and directories

external uc01 file or directory	Description
dbSNP/b131_SNPChrPosOnRef_37_1.bcp dbSNP/b131_SNPContigLoc_37_1.bcp dbSNP/b131_SNPContigLocusId_37_1.bcp	This set of three files is available from the NCBI dbSNP database, which contains information on SNPs and indels already found by other studies. These files contain the data from dbSNP version 131, and are available from this website: **http://www.ncbi.nlm.nih.gov/snp** Three files are used together and all three are needed.

Table 161 Contents and purpose of `external uc01` reference files and directories *(continued)*

external uc01 file or directory	Description
uc01/targetedEnrichment/ targetRegionsUc01.bed	Targeted Regions files are specific to each target capture procedure used during the sample preparation process before the library generation step. As there are many ways to enrich for targeted regions, including custom preparations, these files are to be provided by the user. This directory is the default location for such files, and BED format is typically used here. Here is an example location and filename: external/hg18/targetedEnrichment/targetRegionsHg18.bed The BED file format was developed to support the UCSC Genome Browser, and the format is described here: **http://genome.ucsc.edu/FAQ/FAQformat.html#format1** **Note:** The target regions BED files on the data drive are 0-length placeholders.
uc01/mirbase/	The microRNA database annotations are derived directly from the miRBase website: **http://www.mirbase.org/ftp** which resolves into a file such as this (for human): **ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff** The mature forms are downloaded together as an Excel document: **ftp://mirbase.org/pub/mirbase/CURRENT/miRNA.xls**

Table 162 describes the uc01 example reference files in the `internal` directory. The uc01 is an example assembly. Copy this directory structure to add your own locally-derived assembly to the reference repository, and replace the string `uc01` with the name of your new assembly.

Table 162 Contents and purpose of `internal uc01` reference files and directories

internal uc01 file or directory	Description
reference/unicorn_uc01.fa reference/unicorn_uc01.fa.fai	This directory is for a validated FASTA file describing the genomic reference sequence for uc01: uc01.fa (*Optional*) An index file with the following name can be created to describe each entry: uc01.fa.fai This file contains five columns, with one row for each entry: 1. Chromosome or contig name (no spaces) 2. Chromosome or contig length 3. Cumulative chromosome or contig length 4. Number of characters per line 5. Line width (including carriage return)

Add new reference files

You use the Linux file system, not LifeScope™ Software shell commands, to add new reference files. Depending on the local Linux file system configuration, you may need admin permissions to add new reference files to the repository.

WARNING! Do not add new files to the `lifetech` directory or modify files under `lifetech` directory. Instead use the `internal` and `external` folders, as described below.

Add by assembly name

The reference repository is organized by subfolders for each assembly name. The choice of folder name is important because the folder name matches the assembly name and is used with the `set reference` *assembly* command to automate setting reference-related parameters.

The `internal` and `external` folders are available for your new reference files. Follow this convention:

- **external –** Use for files that have been downloaded from public repositories and placed here verbatim.
- **internal –** Use for locally-derived files to support new genomes and annotations.

Example references at `internal/uc01` and `external/uc01`, in the reference repository location, provide a structure to be followed for new references.

Follow these steps to add a new reference file to the reference repository:

1. Find the location of the reference repository on the local Linux file system.
2. If necessary, create a new subfolder under `referenceData/internal` (or `external`) for the new reference being added. Use a unique assembly name as the name for the new subfolder.

   Mimic the structure of the example references at `referenceData/internal/uc01`, or `referenceData/external/uc01`, in the reference repository location, for your new references.

   For example, to add references for an example assembly named uc02, create the following directory: `referenceData/internal/uc02` or `referenceData/external/uc02`.
3. Follow the instructions in "Prepare reference files" to validate and convert your new reference files.

Example steps to add a new assembly

This section lists example steps to add a new assembly or a new species to the reference repository. This example describes how to add the rat assembly rn4 to the repository. In most installations, file permissions require that the LifeScope™ Software administrator perform these steps.

Repository location and directories

1. Identify the reference repository location, which in the installation instructions is `<`*LSDF*`>/referenceData`.
2. Under `<`*LSDF*`>/referenceData`, create these directories for the new assembly:

```
external/rn4
external/rn4/ucsc
external/rn4/reference
external/rn4/dbSNP                (only required for annotations)
external/rn4/mirbase              (only required for small RNA analyses)
external/rn4/targetedEnrichment   (only required for targeted analyses)
```

The tables in "Initial repository contents" on page 541 explain the repository directory structure.

FASTA file

1. Find a FASTA file describing the reference sequence for rat, for instance from an NCBI, UCSC, or Ensembl site.
2. Copy the rat FASTA file to a local directory.
3. Validate the FASTA file with the `reference_validation.pl` script (see "Validate new reference files" on page 550).
4. Copy or move the validated FASTA file to `external/rn4/reference`, for instance, as the file:

   ```
   external/rn4/reference/rat_rn4.fa
   ```

RefGene file

1. Find a RefGene file for the rat assembly. RefGene files are downloaded directly from UCSC, from this website:

   http://hgdownload.cse.ucsc.edu

   The files have this filename pattern: `refGene.${ASSEMBLY}.${DATE}.gtf`.

   Convert the RefGene file to GTF, using the `refgene2gff.sh` described in "Convert a GTF file" on page 550. The output is in the same filename pattern:

   ```
   refGene.rn4.20110517.gtf
   ```

2. This step is only required for whole transcriptome analyses. In conjunction with the reference FASTA file, the WT exon extractor and junction extractor modules generate new FASTA files for mapping transcripts to exons and junctions:

   ```
   external/rn4/reference/rat_rn4.fa
   ```

   Copy the new FASTA files to:

   ```
   external/rn4/refGene/refGene.rn4.20110517.exon.fa
   external/rn4/refGene/refGene.rn4.20110517.junctions.fa
   ```

Regions of interest file

This section is required for targeted resequencing, low frequency variant, and other enrichment analyses.

1. Find or develop a regions of interest file, in BED format (see "BED file format" on page 510).
2. Copy the regions of interest file to the following location and file name:

   ```
   external/rn4/targetedEnrichment/targetRegionsRn4.bed
   ```

miRBase files

This section is only required for small RNA analyses.

1. Find the species-specific microRNA database annotations (precursor file) on the miRBase website:

   http://www.mirbase.org/ftp.shtml

2. Download the mature forms together as an Excel document:

   ftp://mirbase.org/pub/mirbase/CURRENT/miRNA.xls

3. Save the Excel document as a text file in this location and file name:

   `external/rn4/mirbase/miRNA.txt`

Filter reference file

1. Find or develop a filter reference file.
2. Copy the filter reference file to a local directory.
3. Validate the filter reference file with the `reference_validation.pl` script (see "Validate new reference files" on page 550).
4. Copy or move the validated FASTA file to `external/rn4`, for instance, as the file:

   `external/rn4/rat_filter_reference.fa`

Genomic annotations

Annotation source files such as dbSNP are only required to run the LifeScope™ Software genomic annotations module.

1. See "Annotation sources" on page 296 in Chapter 15, "Add Genomic Annotations to Analysis Results" on page 291 for information on obtaining these files.
2. Copy the dbSNP and other annotation source files to the following directory:

   `external/rn4/dbSNP`

Shell command

When you configure your LifeScope™ Software command shell analysis, use the following shell command to specify your new reference files:

```
set reference rn4
```

This command sets the following parameters with default values for rn4:

- `annotation.dbsnp.file`
- `annotation.gtf.file`
- `analysis.mirbase.precursor.file`
- `analysis.mirbase.mature.file`
- `analysis.filter.reference`
- `analysis.assembly.name`
- `analysis.species.name`

Note: If an assembly of the same name exists in the reference repository `lifetech` directory, the software finds the `lifetech` assembly first. In this case you must set the reference parameters individually.

Prepare reference files

You must perform these procedures on new reference files that you add to the reference repository, before you can use them in a LifeScope™ Software analysis. Each of these steps is described below.

1. Validate the reference file to a format that complies with LifeScope™ Software.
2. Convert new GTF files and refGene files that are required for your analyses.
3. Concatenate the reference files into a single file, if you have more than one, such as when the sequence is separated into single chromosome files.

Validate new reference files

Reference validation is not performed automatically. You must validate the reference files that you add to the reference repository so that reference files are presented in the correct format to each LifeScope™ Software module that uses a reference file.

To validate a reference file, follow these steps:

1. Log in to the LifeScope™ Software cluster. Be sure that you log in to Linux with a user name that has execute ('x') privileges on the directory (`<installdir>/bin`) that contains the `reference_validation.pl` script.

   The script is under the bin folder where the LifeScope™ Software is installed.
2. Navigate to the directory that contains the `reference_validation.pl` file.
3. At a command prompt, enter:

   `<installdir>/bin/reference_validation.pl -f <InputFasta.ext>`

   If the input FASTA file contains IUB codes, the default is to replace the IUBs with Ns.

   Preserving IUBs is not recommended for use with some tertiary modules. To keep IUBs unchanged, use the `-i` (keep IUBs) option:

   `reference_validation.pl -i TRUE -f <InputFasta.ext>`

The output file is named `<InputFasta>_validated.fa`.

Convert a GTF file

Not all species have exactly the same format of GTF file. Before using a GTF file from a public source with a LifeScope™ Software module, you must convert the GTF file into a format that is compatible with LifeScope™ Software.

Convert an Ensembl GTF file

The Ensembl website has GTF-formatted genome annotations available for many popular assemblies. Ensembl GTF files are properly normalized by gene and transcript IDs. The site is at:

www.ensembl.org

Ensembl GTF files use gene accession numbers instead of HUGO-style gene names. Ensembl GTF files also use unprefixed sequence identifiers, such as 1,2,3….X,Y,MT. The Ensembl GTF files are incompatible with genome reference FASTA files that have UCSC-style sequence IDs with the prefix "chr", for example, chr1, chr2, chr3….chrX, chrY, chrM.

Reformat a Ensembl GTF file

1. Log in to the cluster where LifeScope™ Software is installed.
2. Reformat the GTF file with the reformat_ensembl_gtf.pl command, and redirect the output to a new file with `.reformatted` in the file name.

   ```
   For Example:
   reformat_ensembl_gtf.pl Homo_sapiens.GRCh37.63.gtf > \
       Homo_sapiens.GRCh37.63.reformatted.gtf
   ```

   The input GTF file name typically includes the NCBI build number and the Ensmbl release number in its name, according to the following pattern:

   `Homo_sapiens.<reference_ncbi_build_no>.<Ensembl_release_no>.`
   `.gtf`

Convert a refGene.txt.gz file

The UCSC Genome Browser has genome annotations available for many assemblies at

hgdownload.cse.ucsc.edu/goldenPath/

The GTF-formatted annotations available for download are not properly normalized by gene ID. The required content is present for each assembly in the file export of the refGene database table `database/refGene.txt.gz`.

For example, annotation for human genome build 18 is available at:

hgdownload.cse.ucsc.edu/goldenPath/hg18/database/refGene.txt.gz

Note: The GTF-formatted annotation is not in the GTF format required by LifeScope™ Genomic Analysis Software. You must convert the annotation before using it in WTA.

Run the `bin/refgene2gff.sh` script to convert the refGene.txt.gz file:

```
gunzip refGene.txt.gz
refgene2gff.sh -i refGene.txt -o refGene.gff
```

Genome annotations that are downloaded from the UCSC Genome Browser and converted by the annotation conversion script are optimal because they contain Human Genome Organization (HUGO)-style gene names. HUGO-style gene names allow interpretation when using a genome browser or reading reports.

The annotation conversion script works with the latest format of `refGene.txt` files. Assemblies, such as the rat genome, use an alternative format for the `refGene.txt` file. The `refgene2gff.sh` script does not convert alternative formats.

F Command Shell Control Files

This appendix covers:

- Overview 553
- Configuration files 554
- Set up your own analysis run 558
- LifeScope™ Software parameters 559
- (Optional) The Linux `mail` command 561

Overview

This chapter describes the configuration files that LifeScope™ Genomic Analysis Software uses to control its execution. These files control which LifeScope™ Software modules are executed and their order of execution, and also define the run-time parameters that customize each module's behavior.

When you use the LifeScope™ Software UI or the LifeScope™ Software command shell, the configuration files are automatically generated for you.

The following types of control files are used with your LifeScope™ Software analyses:

- **Read-only module XML files** – Contain the parameters accepted for each module and default settings for those parameters. The XML files, one per module, are in the `<installdir>/etc/plugins/pipelines` directory.

  Example: `<installdir>/etc/plugins/pipelines/dibayes.xml`
- **Module properties files** – Define settings used in all runs for a particular module. These files are located in `<installdir>/etc/plugins/properties`.

  Example: `<installdir>/etc/plugins/properties/ \ dibayes.properties`
- **A global.ini file** – Defines settings which by convention are imported into other INI files in the current directory only. For example, in a workflow tertiary directory, the other INI files in that tertiary directory import `global.ini` and all its parameters. In the case of conflicting settings, the setting in the `global.ini` files takes precedence over settings in the `<installdir>` module properties file

  Example: `<installdir>/etc/workflows/small.rna/tertiary/ \ global.ini`
- **LifeScope™ Software modules' INI files** – Define settings for a single module run. In the case of conflicting settings, the module INI file setting takes precedence over settings in both your `global.ini` file and the `<installdir>` module properties file.

  Example: `<installdir>/etc/workflows/ \ targeted.resequencing.frag/tertiary/dibayes.ini`

- **PLN files** – Define a sequence of analyses to be executed. A PLN file lists one or more INI or PLN files. Each INI file listed corresponds to a LifeScope™ Software module to be executed.

  Example: `<installdir>/etc/workflows/ \
  targeted.resequencing.frag/tertiary/tertiary.pln`

Configuration files

PLN files

A PLN file defines your analysis through a list of tasks, which is a list of one or more INI or PLN files. For each listed INI file, LifeScope™ Software executes an analysis run of the corresponding module.

The following is an example of a simple PLN file, which causes a SNPs module analysis to be run:

```
diBayes.ini
```

Tasks (INI or PLN files) are listed one per line. When there are multiple tasks listed, every task without a dependency is executed in parallel (assuming there are sufficient compute nodes).

Dependencies

A PLN file specifies a directed acyclic graph of module dependencies. The less-than sign ('<') signifies a dependency. A dependency means that task on the left of the less-than sign is executed only if the task on the right of the sign completes successfully.

For example, the following is a typical workflow PLN file:

```
secondary.pln
tertiary.pln < secondary.pln
```

This PLN file specifies that `tertiary.pln` is dependent on `secondary.pln`. First the modules specified in `secondary.pln` are executed. Only if those modules completed successfully, then the modules specified in `tertiary.pln` are executed.

A task with no dependencies takes this form in a PLN file:

```
x.ini
```

Multiple dependencies are added after the less-than sign, separated by commas:

```
x.ini < a.ini, b.ini, c.ini
```

Forward references are not allowed in the dependency list, and a task cannot list itself as a dependency.

Other syntax

Blank lines and comment lines are ignored in PLN files. Comment lines must begin with a "#" as the first character.

On error

When one module in a sequence fails, the remaining modules specified in the PLN file continue. However, any module that depends in the failed module is not attempted.

INI files

Module INI files

Module INI files are the main source of configuration parameters requiring your customization for LifeScope™ Software runs. A module INI file defines run-time parameters which control the behavior of the module's algorithm.

Many examples of module INI files are included in the LifeScope™ Software standard workflows, under the following directories:

- <*installdir*>/etc/workflows/*/secondary
- <*installdir*>/etc/workflows/*/tertiary

The following is example content of a module INI file, in this case the CNV INI file:

```
import global.ini

##Run parameter
cnv.run=1

#cnv.mappability.dir=${analysis.mappability.dir}

##Size of the window block to be considered as a region for
writing coverage output. Mean coverage of all bases in each of
these windows will be written
#cnv.coverage.wsize=1000

##The maximum p-value for the regions to be shown as copy number
deletions
#cnv.deletions.max.pval=1.0

##The maximum ratio between the  coverage of the region and the
expected coverage, for the region to be called as cnv deletion
#cnv.deletions.max.ratio=0.5

##The minimum mappability percentage for the regions to be shown
as copy number deletions
#cnv.deletions.min.mappability=50

##The minimum number of windows for the regions to be shown as
copy number deletions
#cnv.deletions.min.windows=2

##To set the ploidy of all chromosomes for Humans
#cnv.gender=Male

##The maximum p-value for the regions to be shown as copy number
insertions
#cnv.insertions.max.pval=1.0

##The minimum mappability percentage for the regions to be shown
as copy number insertions
#cnv.insertions.min.mappability=10

##The minimum ratio between the  coverage of the region and the
expected coverage, for the region to be called as cnv insertion
#cnv.insertions.min.ratio=1.25
```

```
##The minimum number of windows for the regions to be shown as
copy number insertions
#cnv.insertions.min.windows=2

##To indicate whether genome wide normalization or local
normalization should be perfomed. false for genome wide and true
for chromosome-arm local normalization
#cnv.local.normalization=0

##The minimum quality value of the alignments
#cnv.min.quality=2

##The general ploidy of the genome
#cnv.ploidy=2

##List of {contig id: ploidy of the contig} of all the contigs
whose ploidy is different to the general ploidy of the genome
#cnv.ploidy.exception=None

##To set the stringency setting for calling CNVs
#cnv.stringency.setting=Medium

##The distance in kilo bases to be trimmed from the extreme ends
of the chromosome arms
#cnv.trim.distance=1000

##The size of the window block to be considered as a region
#cnv.window.size=5000

##To indicate whether coverage files should be output or not.
Enter 1 to write coverage output in *.wig format. Keep the
default value if you do not want to output coverage
#cnv.write.coverage=0
```

Note: The parameters for CNV are described in Table 45 on page 215 in Chapter 11, Run a Human CNVs Analysis.

Import command

The module INI file typically imports the `global.ini` file. The effect of the `import` statement is to include the content of the `global.ini` file, with all its parameters and defined values, into the module INI file.

Note: An imported file must be in the same directory as the INI file. No pathnames are allowed with INI file `import` statement.

.run parameters

Within a INI file, the `*.run` parameters control which modules are run. For example, these .run parameters appear in INI files:

```
saet.run
bamstats.run
```

```
fragment.mapping.ini
dibayes.run
```

The run parameters are set automatically by the shell during standard workflows. Accept the default for most use.

The allowed values for `.run` parameters are 0 and 1:

- **0**: Do not run the module.
- **1**: Run the module analysis (default).

Because mapping is a prerequisite for all other runs, `mapping.run` may not be set to 0 unless it has already been successfully run to completion. You may set the .run parameters for other modules to 0, under these conditions:

- You do not want to run that module analysis.
- That module analysis has already been completed (for instance, if you are restarting the workflow run and repeating a particular module is not necessary).

When you run in the project repository, the LifeScope™ Software command shell `resume` command does not repeat modules that have successfully completed.

The global.ini file

The `global.ini` file defines parameters for one or more INI files.

If a parameter in the global.ini also appears in an analysis module's INI file, then the value set in the global.ini file is ignored and the analysis module's INI's definition is used. The `import global.ini` statement must be placed at the top of the an analysis module's INI file.

Paths are not allowed in the `import global.ini` statement. The `global.ini` file must be in the same directory as the analysis module INI file that imports it.

Properties files

A properties file sets the default parameters used by every run of its module type, for all users of the LifeScope™ Software installation. Properties files are located in the directory `<installdir>/etc/plugins/properties`.

For example, the file `fragment.mapping.properties` contains the settings for genomic mapping runs. Example contents for the `fragment.mapping.properties` file are shown below.

```
wall.time=120
fragmap.number.of.nodes=4
mapping.np.per.node=8
fragmap.minreads.per.node=4000000
fragmap.maxreads.per.node=150000000
fragmap.max.number.of.nodes=100
mapping.memory=15gb
java.heap.space=1500
```

Parameter values defined in a .properties file apply to all runs of that type of module, except that parameter definitions in a INI file override the value set in the properties file. The relationship between the two types of files is explained in "Property files vs. INI files".

Property files vs. INI files

Parameter values set in a properties file affect all runs of that module, but may be reset in the module INI file.

Modifying a setting in the properties file affects all future runs of that module, by all users. Modifying the setting in the `fragment.mapping.ini` file affects only the current run and any subsequent runs using the INI file. Settings present in the `fragment.mapping.ini` file take precedence over settings in the properties file. Table 163 compares the influence of parameters in a .properties file and an INI file. Mapping module files are used as a example to demonstrate the general principle relating .properties files and INI files.

Table 163 Comparing module properties and INI files (using mapping as an example)

File and location	Coverage	Behavior on parameter conflicts
<*installdir*>/etc/plugins/properties/mapping.properties	Affects all mapping runs.	Values may be overridden by settings in fragment.mapping.ini.
your global.ini file	Affects all runs (of all modules) if an INI file imports this global.ini file.	Overrides settings in mapping.properties. Values may be overridden by settings in fragment.mapping.ini.
your fragmant.mapping.ini file	Affects all fragment mapping runs that use this INI file.	The parameter value defined in fragment.mapping.ini is used, overriding settings from the mapping.properties and global.ini files.

Analysis module XML files

In the directory `<installdir>/etc/plugins/pipelines` are module XML files, one per module, which list all parameters accepted by each module, along with the parameters' default values. Use the XML files as a reference for allowed parameters and their default values.

Do not edit these XML files in order to change a parameter's value. Instead, modify one of these two files, depending on your purpose:

- **The module's .properties file** – To change a parameter for all subsequent module runs *for all users*, edit the parameter in the module's properties file under `<installdir>`. See "Properties files" on page 557.
- **Your module INI file** – To change a parameter for only your own runs of a module, edit the parameter in your module's INI file. This change affects only analyses that use that INI file. See "PLN files" on page 554.

In either on the two cases above, you may add a parameter that did not previously exist in the file, only if that parameter appears in the module's XML file.

Set up your own analysis run

The most direct way to customize your own LifeScope™ Software analysis is to copy an existing workflow. You can then change the workflow to execute your preferred module or sequence of modules. See "Create a new workflow" on page 121 and "Run a standard workflow" on page 118 for more information.

The optional examples download provides single-module analyses. See Appendix H, "Demo Analyses" on page 567 for information on these analyses.

The shell `get workflow` command copies a workflow's INI files, PLN files, and their directory structure to the local file system. These files are copied to the directory you are in when you open the shell with the `lscope.sh` command. You can modify the INI and PLN files to customize your own analysis. See Table 13 on page 89 and "Create a new workflow" on page 121.

LifeScope™ Software parameters

This section describes parameters the software uses to control your analysis runs. When you run an analysis in the command shell, these parameters can be set automatically by the software. If you create your own INI files, you often need to specify some of these parameters (depending on the type of analysis you are running).

See also "Analysis space" on page 77 for a description of the `analysis.space` parameter.

General parameters

Table 164 lists parameters that LifeScope™ Software sets when you run an analysis in the command shell.

Table 164 General parameters

Parameter	Description
analysis.input.readset.file	An RRS (Read Range Specifier) file built by the system specifying the XSQ read-sets being analyzed (only applies to secondary analysis). RRS is an internal file format.
analysis.genome.reference	Path to the genome reference file.
analysis.sample.name	User-defined sample name.
analysis.input.bams	Comma-separated paths to input BAM files (for tertiary analysis).

Reference parameters

Table 165 lists reference-related parameters that LifeScope™ Software sets for you when you use the `set reference` command with a recognized assembly name.

Table 165 Reference-related parameters

Parameter	Description
annotation.dbsnp.file	The file used to annotate SNPs and small indels. The initial reference repository includes dbSNP_b130.tab (hg18) and dbSNP_b132_00-All.vcf (hg19).
annotation.gtf.file	The file containing gene and exon annotations corresponding to the genome assembly used in the analysis.
analysis.filter.reference	The file containing a collection of sequences that are used to filter out commonly-occurring motifs and contaminating sequences prior to mapping.
analysis.mirbase.mature.file	The file containing mature form microRNA database annotations.
analysis.mirbase.precursor.file	The file containing precursor microRNA database annotations.
analysis.assembly.name	The name of the genome assembly used in current analysis. Examples: hg18, hg19

Table 165 Reference-related parameters *(continued)*

Parameter	Description
analysis.species.name	The name of the species used in current analysis. Example: human
analysis.mappability.dir	The directory containing binary mappability files used by the Human CNV module.

Before starting your analysis, confirm that the reference-related parameters are correctly set. After the `set reference` command, run the LifeScope™ Software shell `ls` command to review the parameter settings (both commands are executed in the LifeScope™ Software command shell, in the analysis virtual directory).

Note: In the default reference repository, the precursor file `hsa.gff` is not located under the `external` directories. The file is included as `referenceData/lifetech/hg19/mirbase/hsa.gff`.

Table 166 lists a framework parameter, `analysis.regions.file`, that you must explicitly set (in both the GUI and the command shell). The reference repository has placeholders for the targeted regions of interest files, but you must both supply the regions of interest file and explicitly set the `analysis.regions.file` parameter. The framework does not set this parameter automatically with the `set workflow` command. The GUI does not enforce that this parameter be set.

The regions is interest file is required with targeted resequencing and low frequency variant workflows, and with any other analysis that involves the enrichment module.

Table 166 Regions of interest file parameter

Parameter	Description
analysis.regions.file	Full path to your regions of interest file.

Execution control parameters

Table 167 lists parameters used for job management.

Table 167 Job management parameters

Parameter	Description
task.temp.dir.delete	Toggles cleanup of intermediate files created in the temp directory.
task.scratch.dir.delete	Toggles cleanup of intermediate files created in the scratch directory.
scratch.dir	Location of the scratch directory.
*.run	Parameters that toggle the execution of analysis modules. Examples: • cnv.run • dibayes.run

Read-only parameters

Table 168 lists *read-only* parameters that you can use in your INI files to set other parameters. Do *not* set any of these parameters.

Table 168 Read-only parameters

Parameter	Description
analysis.dir	The base analysis directory.
analysis.name	The name of the base analysis.

Table 168 Read-only parameters *(continued)*

Parameter	Description
analysis.output.dir	${analysis.dir}/outputs.
analysis.temp.dir	${analysis.dir}/temp.
task.name	The name of the current task. Same as base name of the task's INI file.
task.working.dir	${analysis.dir}/${task.name}, the task working directory.
task.temp.dir	${analysis.temp.dir}/${task.name}, the task temp directory where task writes temporary files. Deleted at the end of a successful run, unless task.temp.dir.delete is set to false.
task.scratch.dir	The task-specific scratch directory. For cluster installations, this directory is local to the compute node.
task.output.dir	${analysis.output.dir}/${task.name}/${analysis.sample.name}, the task's sample-specific output directory.

(*Optional*) The Linux `mail` command

The LifeScope™ Software command shell protects your currently running analyses as well as the configuration and definition of both your projects and your analyses. The running jobs and the configuration are not affected if you close your LifeScope™ Software shell and exit your Linux session. You do not need to use Linux commands such as `screens` or `nohup` in order to keep your analysis or session window open.

The UNIX `mail` command can be used to send an email notification of a script's completion. Your UNIX mail system must be configured properly. Add the following to your `lscope.sh run` or `lscope.sh resume` command:

```
&& mail -s "subject" <email_address>
```

where *subject* is the subject of the email message to be sent and *email_address* is the email account to be sent the completion message. You must follow UNIX command conventions. Keep the *subject* string short and avoid special characters.

An example of the complete command is:

```
lscope.sh run my.pln -u user && mail -s "done" me@xyz.com
```

G Administration

This appendix covers:

- Overview 563
- Admin tasks. 563
- Admin interfaces. 565

Overview

This appendix describes administrative functions that can be performed on either in the Linux shell or the LifeScope™ Software command shell.

Most administrative functions performed in the LifeScope™ Software UI, with the GUI Admin Portal. Refer to the *LifeScope™ Genomic Analysis Software User Guide* (Part no. 4471877) for more information.

Under some conditions, you can use the LifeScope™ Software command shell (or Linux shell) for all administrative functions, and avoid using the GUI Admin Portal. You do not need the GUI Admin Portal if both of these are true for your installation:

- You select the "host" authentication realm during installation.
- You do not use named licenses.

See "Authentication realm" on page 43 and "Configure the license server" on page 46, in Chapter 2, LifeScope™ Genomic Analysis Software Installation, for information on authentication realms and licenses.

Admin tasks

This section describes typical admin tasks such as user management, scratch space cleanup, reference repository management, and resetting the admin password.

User management

User management tasks vary with the authentication realm used for your installation:

- **LifeScope –** Users are created in the LifeScope™ Software user repository. User account management (including account creation) is performed with the LifeScope™ Software Admin Portal (included in this installation). There is no relationship between the LifeScope™ Software users accounts and Linux® host system users or LDAP users.
- **LDAP –** User accounts on LDAP-compliant authentication server (OpenLDAP, Active Directory, etc.) are eligible to be LifeScope™ Software users. User account management is performed with the LifeScope™ Software Admin Portal (included in this installation). As admin, you select the LDAP users to be allowed LifeScope™ Software access. You do not create user accounts.
- **Host –** Local (on the installation machine or cluster) Linux users are all LifeScope™ Software users. You do not create user accounts or select LifeScope™ Software users.

As admin, in the Admin Portal you can disable a LifeScope™ Software user and later re-enable the user.

Scratch space cleanup

Under normal operation, LifeScope™ Software removes the files and directories it created in scratch space. However, as admin you must periodically check the scratch directory and manually delete old files that LifeScope™ Software did not remove.

The scratch directory is setting during installation. The default is `/scratch/solid/lifescope`.

The files and directories to delete have various names, but contain the name of an analysis module and end in a time stamp. Example names are:

- `wholeTranscriptomeFrag_secondary-human_chr17_6-wt.frag.map.main.20110430091312676`
- `test45_secondary-hg18-saet.main.20110702013417478`
- `test52_20110707030127_secondary-hg18-saet.main.20110707100009799`
- `saet_secondary-hg18-saet.main.20110427011039729`

Note: Be careful to not remove files or directories for a currently-running LifeScope™ Software process.

Reference file conflicts

Depending on the local file permissions, the LifeScope admin might be responsible for adding the reference files to the reference repository. In order to avoid conflicts with other reference files in the reference repository, always add new reference files to the correct assembly subfolder under the `internal` directory. See "Add new reference files" on page 547 in Appendix E, "The Reference Repository" on page 539.

Reset the admin password

The `resetpwd.sh` script is provided as an emergency way to reset the administrator's password to its default value. To reset the admin password, follow these steps (at a Linux prompt):

1. Stop the LifeScope™ Software server:
   `lscope-server.sh stop`
2. Reset the admin password:
   `resetpwd.sh lifescope`
3. Restart the LifeScope™ Software server.

4. In the LifeScope™ Software UI, change the admin password to a secure password, according to your local policy.

The use of the `resetpwd.sh` script is not recommended for any user account except the LifeScope™ Software admin. Passwords reset for other users may become out of sync the authenticating realm.

Read-set repository path: notify users

If you change the path to the read-set repository, notify users to restart LifeScope™ Software so that the repository can be updated to the new path.

License management

Licenses are released during normal exit of the client software. If the client software exits abnormally, the LifeScope™ Software server may not be able to release the license. The only way to free that unreleased license is through stopping and restarting the LifeScope™ Software server module.

Admin interfaces

GUI Admin Portal

The main administrative interface is the GUI Admin Portal, described in the *LifeScope™ Genomic Analysis Software User Guide* (Part no. 4471877). The Admin Portal supports all administrative tasks required for LifeScope™ Software.

The following administrative functions cannot be done in the command shell and can only be done in the GUI Admin Portal:

- Setting a named license per user.
- Enabling and disabling LifeScope™ Software users.

  However, with the "host" authentication realm (selected during installation), all Linux user accounts are enabled as LifeScope™ Software users. With "host" authentication, you avoid the need to enable LifeScope™ Software users.

Shell admin commands

Limited admin functionality is also available in the command shell. Table 169 lists the LifeScope™ Software shell admin commands, which cover user administration and read repository index. These commands require the admin role privilege.

Table 169 LifeScope™ Software shell admin commands

Command	Description
Administrative user commands:	
ls	Lists all users.
mk *name*	Creates a new user named *name*, with the password *name* and the role `user`.
rm *name*	Deletes the LifeScope™ Software shell user named *name*.
loggedin	Shows logged in users.
set username *role* [password]	Sets a user's role and optionally set the user's password. Valid roles are: admin, user
makepasswdfile	Creates an encrypted password file for non-interactive login. The file is named `lscope.passwd`, and is created in the directory where your `lscope.sh shell` command is originally issued.
Other administrative command:	

Table 169 LifeScope™ Software shell admin commands

rebuild	In `/reads`, rebuilds the read repository index, which is used to display the list of read-sets in the repository. Recent changes (additions or deletions) to the read repository are not reflected in the `ls` output until the index is rebuilt.

Only admin users have access to the /`users` category. The admin user commands must be executed at the `/users` level.

Other information

The <*`installdir`*>`/server` directory contains the following files of interest:

- **`server.properties`** – Contains important LifeScope™ Software configuration information and locations, such as the authentication realm, port numbers, projects repository directory, results directory, and others.
- **`server.err`** – Contains error messages generated by the LifeScope™ Software server.
- **`server.log`** – Contains log messages generated by the LifeScope™ Software server.
- **`server.pid`** – Contains the process id and port number used by the LifeScope™ Software server.

See also the following sections:

Section	Page	Chapter or appendix
"Logs"	489	"Troubleshooting"
"Troubleshooting"	490	"Troubleshooting"
"Naming restrictions"	81	"Understand The LifeScope™ Software Shell"
"Error behavior and error messages"	108	"Run a Command Shell Analysis"
"Check LifeScope™ Software servers"	69	"Before You Begin"

H Demo Analyses

This appendix covers:

- Overview 567
- List of demo modules. 567
- Demos location 568
- How a demo is setup 568
- Run all demos 569
- Run a single demo analysis. 570
- Output files and logs 570

Overview

This appendix describes the demo modules included in an optional download. The demos show how to run an individual LifeScope™ Software analysis module, such as mapping or SNPs, using the LifeScope™ Software command shell.

The demo analyses are mostly self-contained. Input data and references files are provided. The exception is that some demos use hg18 references files in the LifeScope™ Software reference repository.

The standard workflows, which are sequences of analyses combined into one run of LifeScope™ Software, are recommended for general use instead of the demos. See Chapter 7, "Run a Standard Workflow Analysis" on page 111 for information on standard workflows. The demo analyses are based on execution methods used in earlier versions of the software, and have been adapted for the LifeScope™ Software command shell.

List of demo modules

The following analyses are included in the demos download:

- **Mapping –** fragmentMapping
- **Paired mapping –** pairMapping
- **SNPs –** diBayes
- **Human CNV –** cnv
- **Inversion –** inversion
- **Large indel –** largeIndel
- **Small indel –** smallIndel
- **SAET –** saet
- **Enrichment –** enrichment

- **WT fragment –** wholeTranscriptomeFrag
- **WT paired-end –** wholeTranscriptomePE

Except for whole transcriptome, the demos are configured to run an individual analysis module. The whole transcriptome demos are similar to the whole transcriptome standard workflows, but the demos include the input files and reference files.

Demos location

The LifeScope™ Software demos are a separate download not included in the main installation. Ask your LifeScope™ Software administrator for the location of the demos.

How a demo is setup

This section describes the mechanics of how each demo and its files and directories are organized.

Each demo analysis directory contains a `run.sh` script, which invokes `lscope.sh` in scripted mode. The scripted `lscope.sh` shell commands are contained in a file named `run.txt`. Here are example `run.txt` contents, for SNPs:

```
cd projects
mk examples
cd examples
rm diBayes
mk diBayes
cd diBayes
set workflow ./analysis.pln
add bam ./bam/F3-R3-Paired.twoSNPs.bam
add bam ./bam/F3-R3-Paired.twoSNPsmerged.bam
set reference hg18
run
wait
ls
exit
```

Table 170 explains the purpose of the steps shown above.

Table 170 Description of sample shell commands to run a workflow

Command	Purpose

`cd /projects`
`mk examples`
`cd examples`
`rm diBayes`
`mk diBayes`
	In the projects repository, create a project named `examples`, and in that project create an analysis named `dibayes`. Open the `dibayes` analysis.
	Note: The open analysis (`cd dibayes`, in this case) command is important. The data, reference, and other configuration commands which follow apply only to the current analysis.
	Note: The demos all delete the current demo analysis if it exists (the `rm diBayes` command in this case).

`set workflow ./analysis.pln`	Define the workflow used with this analysis.
`add bam ./bam/F3-R3-Paired.twoSNPs.bam` `add bam ./bam/F3-R3-Paired.twoSNPsmerged.bam`	Define the BAM input files containing data for this analysis.
`set reference hg18`	Define the reference for this analysis. The string `hg18` matches an assembly name in the reference repository.
`run`	Start the analysis run.
`wait`	Wait until the current analysis run completes.
`ls`	Display status information about the analysis run.

Workflow

The workflow defined in this demo is a PLN file, `analysis.pln`. A PLN file defines your analysis through a list of tasks, which is a list of one or more INI or PLN files. For each listed INI file, LifeScope™ Software executes an analysis run of the corresponding module.

In the SNPs demo, `analysis.pln` calls `tertiary.pln`, which contains the entry `dibayes.ini` to invoke the SNPs analysis module

Input files

The demo uses BAM input, because SNPs is a tertiary analysis. Demos which do mapping analysis instead take XSQ files as input.

Run all demos

This section describes how to run all the demo analyses in one run. The combined run should complete in under 30 minutes.

The demos require the default reference repository, which includes hg18 and hg19 assemblies.

Follow these steps to execute all the demos in one run:

1. Cd to the demo directory.
2. Execute this command in the Linux shell:
   `runall.sh`
   This script runs each demo one at a time.

The `runall.sh` script by default executes the run under the `lifescope` user account. To use a different user name, use the `-u` and `-w` options:

`runall.sh -u` *username* `-w` *password*

Run a single demo analysis

Each demo is executed with the same command. To execute any demo, follow these steps:

1. Cd to the general demo directory, and then cd to the specific analysis directory.
2. Execute this command in the Linux shell:

   `run.sh`

   The `run.sh` invokes `lscope.sh` in scripted mode. The scripted `lscope.sh` shell commands are contained in a file named `run.txt`, and are similar to the commands in Table 170.

   The major differences in `run.txt` commands are the following:

   - The analysis name
   - The input commands, `add xsq` or `add bam`
   - The `set reference` command

The `runall.sh` script by default executes the run under the `lifescope` user account. To use a different user name, use the `-u` and `-w` options:

`runall.sh -u` *username* `-w` *password*

Output files and logs

The results files are generated in the projects repository, which is set during installation. Within the projects repository directory, the demos results are generated in the following directories:

user_name`/examples/`*analysis_name*`/outputs/`*sample_name*

This list describes the folder names

- ***user_name*** – The LifeScope™ Software user name of the person running the demo.
- **`examples`** – The project name used by all demos.
- ***analysis_name*** – The name of the analysis module used by a particular demos. For example: `cnv`, `dibayes`, `enrichment`.
- **outputs** – A hard-coded subfolder name.
- ***sample_name*** – Determined by the sample name in the input data.

In its results directory, each of the demos analyses creates an output file named `summary.log`. Check for a success message in the last statement of the log.

The results files are overwritten on subsequent runs executed by the same user. You can view the results of your own demo runs in the LifeScope™ Software UI.

I LIFESCOPE™ GENOMIC ANALYSIS SOFTWARE v2

END USER LICENSE AGREEMENT

This is a legal agreement between you, the person or entity receiving software products and/or software support ("Licensee"), and Life Technologies Corporation, having offices at 5791 Van Allen Way, Carlsbad California 92008 USA ("Licensor"). This agreement is part of a package that includes one or more software products and certain electronic and/or written materials. This agreement covers your licensing of such software and/or purchase of support.

You must agree to the terms in this End User License Agreement ("EULA") in order to access the software and/or receive support.

BY CLICKING YOUR ACCEPTANCE OF THIS EULA, OR BY INSTALLING OR USING THE SOFTWARE (defined below) OR ANY OTHER COMPONENT OF THE PACKAGE, YOU ACKNOWLEDGE THAT YOU HAVE READ ALL OF THE TERMS AND CONDITIONS OF THIS EULA, UNDERSTAND THEM, AND AGREE TO BE LEGALLY BOUND BY THEM. If you do not agree to the terms of this EULA, you may not install or use the Software, and may return it to Licensor for a refund or product credit. In addition to the restrictions imposed under this EULA, any other usage restrictions contained in the Order (defined below), Software installation instructions or release notes, and Support policies (defined below) shall apply to your use of the Software and receipt of Support.

As used in this EULA: "**Authorized Users**" means, collectively, the personnel authorized by you to use the Software for your benefit, provided you have both purchased a License (as defined below) and paid the corresponding license fees. Unless otherwise expressly allowed by this EULA, Authorized Users may include only your employees and agents having a need to know, and Authorized Users may not be entities or persons in the business of licensing or otherwise providing products or services competitive with the Software. "**Designated Site**" means your facilities or offices located at the postal address provided to Licensor for your billing and invoicing purposes, unless otherwise indicated in a license key provided to you. "**Software**" means the software product(s) accompanying this EULA and the content therein; including the associated data, user manuals, user documentation and application program interfaces (if any), and License Key(s) and File(s) provided, and any patches, updates, upgrades, improvements, enhancements, fixes and revised versions of any of the foregoing that may be provided to you from time to time, and any combination of the foregoing. "**License Key(s))**" means the alphanumeric code(s) provided to you by Licensor to enable you to obtain a License File(s). "**License File(s)** means the file(s) provided by Licensor with which you activate your copy of the Software. "**Order**" means that part of a written or electronic document that identifies (1) the Software to be licensed to you, (2) the Authorized Scope (defined below), (3) any Support purchases, (4) the purchase price, and (5) location for delivery, and in each case as expressly agreed upon by Licensor. "**Affiliate**" means any entity Controlling,

Controlled by, or under common Control with the referenced entity, where the term "**Control**" means the possession, direct or indirect, of the power to direct or cause the direction of the management and policies of an entity, whether through the ownership of voting securities, by contract, or otherwise.

Your Payment Obligations

YOU AGREE TO PAY ALL AMOUNTS DUE OR INCURRED BY YOU, INCLUDING ANY LATE PAYMENT FEES, AS ARE SPECIFIED IN THIS EULA, IN THE ORDER, AND/OR ANY ASSOCIATED INVOICE. ALL FEES AND AMOUNTS DUE LICENSOR ARE EXCLUSIVE OF ALL TAXES, DUTIES SHIPPING FEES, AND SIMILAR AMOUNTS. IF ANY AUTHORITY IMPOSES A DUTY, TAX OR SIMILAR AMOUNT (OTHER THAN TAXES BASED ON LICENSOR'S INCOME), YOU AGREE TO PAY, OR TO PROMPTLY REIMBURSE LICENSOR FOR, ALL SUCH AMOUNTS. UNLESS OTHERWISE INDICATED, ALL INVOICES ARE PAYABLE THIRTY (30) DAYS FROM THE DATE OF INVOICE. OVERDUE AMOUNTS ARE SUBJECT TO A LATE PAYMENT CHARGE, AT THE LOWER RATE OF (I) ONE PERCENT (1%) PER MONTH, OR (II) THE MAXIMUM RATE UNDER APPLICABLE LAW. YOU AGREE TO PROMPTLY PAY OR REIMBURSE LICENSOR FOR ALL COSTS AND EXPENSES, INCLUDING ALL REASONABLE ATTORNEYS' FEES, RELATED TO BREACH OF YOUR OBLIGATIONS UNDER THIS EULA AND/OR LICENSOR'S ENFORCEMENT OF THIS EULA. ALL SHIPMENTS BY LICENSOR OR ITS DESIGNEE ARE FCA POINT OF SHIPMENT (INCOTERMS 2000).]

Acceptance

Except with respect to Software provided under a Trial License (defined below), you will be deemed to have accepted the Software unless you provide written notice of rejection within ten (10) days after receipt of the Software or the corresponding License Key(s) and File(s), if any (whichever event occurs first). Any such notice must state the reason for rejection, and you may only reject the Software if it fails to materially comply with its accompanying documentation. If you reject the Software, Licensor's sole obligation and liability, and your sole and exclusive remedy, shall be for Licensor to use commercially reasonable efforts to deliver to you a replacement for the nonconforming Software, and if Licensor is not able to deliver a replacement for the Software, then Licensor will refund any license fees paid by you for the Software, and in the event of any such refund, the EULA shall terminate. Software provided under a Trial License shall be deemed accepted upon receipt.

Grant of Software License

Subject to the terms and conditions of this EULA, Licensor grants to you a non-exclusive, non-transferable license ("**License**") for Authorized Users to use the Software and Support for your internal operations and internal data processing purposes within and up to the Authorized Scope described on the Order, and for which you have paid the applicable license and support fees and have registered the Software for use. Except as otherwise provided below, this EULA and the License granted hereunder shall be effective until terminated in accordance with Section 7 below. The term "**Authorized Scope**" means the following, and any other capacity, term/duration, or use restrictions indicated by Licensor on the license granted to you:

The "**Concurrent User License**" configuration is comprised of a defined number of simultaneous-use licenses shared among an unlimited number of computers that are on the network at the Designated Site. Each Concurrent User License has a term of one (1) year. For example, the "LifeScope™ Core Server Software C1" product enables simultaneous use of the Software on any of up to five (5) computers for one (1) year at

the Designated Site. In other words, at any given time, any combination of users from up to 5 computers on the network may access the Software at the Designated Site. Each Concurrent User must have a unique username and password to access the Software. Simultaneous use of the Software from additional computers above five (5) can be enabled through the purchase of the desired number of "Supplemental User C1" licenses and payment of the applicable license and support fees. You may not enable concurrent operation of the Software beyond the scope of the Concurrent User Licenses purchased. You may not network the Software for use at any other geographic location. . Termination or expiration of the Concurrent User Licenses shall immediately terminate this EULA.

The "**Named User License**" configuration is comprised of a defined number of licenses shared among the same number of computers that are on the network at the Designated Site. Each Named User License has a term of one (1) year. For example, the "LifeScope™ Core Server Software N1" product enables simultaneous use of the Software on only five (5) specific computers for one (1) year at the Designated Site. In other words, at any given time, only users from the defined list of computers may access the software at the Designated Site. Each Named User must have a unique username and password to access the Software. Additional computers above five (5) may be added to the defined list of computers through the purchase of the desired number of "Supplemental User N1" licenses and payment of the applicable license and support fees. You may not enable operation of the Software beyond the scope of the Named User Licenses purchased. You may not network the Software for use at any other geographic location. Termination or expiration of the Named User Licenses shall immediately terminate this EULA.

The Software may be provided under a "**Trial License**" for your internal evaluation purposes only, pending your purchase of a commercial-use Software license and payment of the applicable license fees. Depending on the mode of delivery of the Trial License, you may (i) access the Software though an online account mechanism, or (ii) you may install and operate the Software on computer hardware located at the Designated Site. You may not network the Software for use at any other geographic location. A Trial License may not be used for any commercial purposes. A Trial License may automatically be converted into a commercial license (*e.g.*, with Authorized Scope converted into a Concurrent User License) upon payment of the applicable license fees to Licensor or issuance of the applicable License Key(s) and Files(s), if any (whichever event occurs first). Upon such conversion, this EULA shall continue in full force and effect, subject to the restrictions applicable to the new Authorized Scope. Software provided under a Trial License is made available to you "AS IS", AND LICENSOR MAKES NO REPRESENTATION OR WARRANTY REGARDING SUCH SOFTWARE.

RESTRICTIONS ON USE. You acknowledge that you are receiving LICENSED RIGHTS only. The Software may only be used internally, by your Authorized Users, with the License Key(s) and File(s) (if any) provided, for your copy(ies) of the Software. If any Software is provided on separate media (*e.g.*, a CD-ROM), you may make a single copy solely for your internal backup purposes. You shall not directly or indirectly: (i) sell, rent, lease, distribute, redistribute or transfer any of the Software or any rights in any of the Software, including without limitation through "charge back" or any other selling, reselling, distributing or redistributing within your organization of any usage capacity you have licensed, without the prior express written approval of Licensor, (ii) modify, translate, reverse engineer (except to the limited extent permitted by law), decompile, disassemble, attempt to discover the source code for, create derivative works based on, or sublicense any of the Software, (iii) use any Software for the benefit of any third parties (*e.g.*, in an ASP, outsourcing or service bureau relationship), or in

any way other than in its intended manner, without the prior express written approval of Licensor, (iv) remove any proprietary notice, labels, or marks on or in the Software, or (v) disable or circumvent any access control or related device, process or procedure established with respect to the Software, including the License Key(s) and File(s) (if any) or any other part thereof. Further, without Licensor's prior express written consent, you may not: (i) network the Software for use at any other geographic location; (ii) enable operation of concurrent "instances" of the Software beyond the scope of the license purchased; (iii) share accounts, e.g., pool multiple users through a single account, or (iv) "multiplex" or "pool" any Software, including through use of third party software products such as those made by Citrix Systems, Inc., or use any terminal applications/emulators to enable use of the software beyond the scope of the license purchased. If the Software design permits modification, then you may only use such modifications or new software programs for your internal purposes and otherwise consistent with the License and Authorized Scope. You are responsible for all use of the Software and for compliance with this EULA; any breach by you or any user of the Software shall be deemed to have been a breach by you. Licensor reserves all rights not expressly granted; no right or license is granted hereunder, express or implied or by way of estoppel, to any intellectual property rights other than as expressly set forth herein; and your purchase of a license to the Software does not by itself convey or imply the right to use the Software in combination with any other product(s). As between you and Licensor, Licensor retains all right, title, and interest in and to the Software, which rights include, but are not limited to, patent, copyright, moral, trademark, trade secret and all other intellectual property rights. You agree and acknowledge that you have been provided sufficient information such that you do not need to reverse engineer the Software in any way to permit other products or information to interoperate with the Software.

NO SEPARATION OF COMPONENTS. The Software is licensed as a single product. Some Licensor software products combine separately available components into a single product (*e.g.*, a software suite product may be comprised of multiple component products). When licensed as a combination product, the component parts may not be separated for use independently of the combination product. You must first purchase a license to each component of the combination product before you may use it independently of the combination product.

ADDITIONAL PURCHASES. Purchase of additional or changed licenses or Authorized Scope is subject to availability and current pricing. Licensor may, from time to time, update the available Authorized Scope plans, and add or delete from available plans. To the extent you purchase an upgrade to your license by expanding the Authorized Scope under a changed or additional plan, Licensor will provide you with the additional license terms and conditions governing your use of the Software under such plans; all other terms and conditions of this EULA shall remain in effect.

TERM AND TERMINATION. Unless otherwise agreed, the term of this EULA shall continue until it expires or is terminated; however, if you are receiving a Trial License, the term shall expire thirty (30) days following the date you receive the Software. Termination or expiration of this EULA shall concurrently terminate all Licenses granted under this EULA. Licensor shall not refund any amounts paid by you hereunder in the event of expiration or termination of this EULA unless expressly provided in this EULA. Licensor may (i) terminate an Order and the Licenses to the Software and/or Support on that Order if you fail to pay any applicable fees due under that Order within fifteen (15) days after receipt of written notice of non-payment; and/or (ii) terminate this EULA (or any License) upon fifteen (15) days written notice if you breach this EULA and do not cure the breach within fifteen (15) days following receipt of written notice of breach. Immediately upon any termination or expiration of this

EULA, you agree to: (a) pay all amounts owed to Licensor; (b) un-install and cease use of the Software for which your rights have been terminated; (c) upon request, return to Licensor (or destroy) all copies of the Software and any other Confidential Information or proprietary materials in your possession for which your rights have been terminated; and (d) upon request, certify in writing your compliance with (b) and (c), above.

CONFIDENTIALITY. You agree to protect Licensor's Confidential Information with the same degree of care used to protect your own confidential information (but in no event less than a reasonable standard of care), and not to use or disclose any portion of such Confidential Information to third parties, except as expressly authorized in this EULA. You acknowledge that the Software, including its content, structure, organization and design constitute proprietary and valuable trade secrets (and other intellectual property rights) of Licensor and/or its licensors. The term "**Confidential Information**" means, collectively, non-public information that Licensor (and its licensors) provide and reasonably consider to be of a confidential, proprietary or trade secret nature, including but not limited to (i) the Software, (ii) Software License and Support prices, (iii) Software License Keys and Files, and (iv) confidential elements of the Software and Licensor's (and its licensors') technology and know-how, whether in tangible or intangible form, whether designated as confidential or not, and whether or not stored, compiled or memorialized physically, electronically, graphically, photographically, or in writing. Confidential Information does not include any information which you can demonstrate by credible evidence: (a) is, as of the time of its disclosure, or thereafter becomes part of the public domain through no fault of yours; (b) was rightfully known to you prior to the time of its disclosure, or to have been independently developed by you without use of Confidential Information; and/or (c) is subsequently learned from a third party not under a confidentiality obligation with respect to such Confidential Information. Confidential Information that is required to be disclosed by you pursuant to a duly authorized subpoena, court order, or government authority shall continue to be Confidential Information for all other purposes and you agree, prior to disclosing pursuant to a subpoena, court order, or government authority, to provide prompt written notice and assistance to Licensor prior to such disclosure, so that Licensor may seek a protective order or other appropriate remedy to protect against disclosure.

WARRANTY AND DISCLAIMER. Licensor warrants that (i) except with respect to Software provided under a Trial License (in respect of which no warranty is made, as described in Section Grant of Software License), for a period of twenty (20) days from the date of acceptance of the Software as described in Section Except with respect to Software provided under a Trial License (defined below), you will be deemed to have accepted the Software unless you provide written notice of rejection within ten (10) days after receipt of the Software or the corresponding License Key(s) and File(s), if any (whichever event occurs first). Any such notice must state the reason for rejection, and you may only reject the Software if it fails to materially comply with its accompanying documentation. If you reject the Software, Licensor's sole obligation and liability, and your sole and exclusive remedy, shall be for Licensor to use commercially reasonable efforts to deliver to you a replacement for the nonconforming Software, and if Licensor is not able to deliver a replacement for the Software, then Licensor will refund any license fees paid by you for the Software, and in the event of any such refund, the EULA shall terminate. Software provided under a Trial License shall be deemed accepted upon receipt., the Software will, under normal use and as unmodified, substantially perform the functions described in its accompanying documentation; and (ii) Licensor will perform Support services during the License term in a professional and workmanlike manner. No warranty is provided for uses beyond the Authorized Scope. THE FOREGOING EXPRESS WARRANTIES

REPLACE AND ARE IN LIEU OF ALL OTHER WARRANTIES AND REPRESENTATIONS BY LICENSOR, WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY IMPLIED OR OTHER WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-MISAPPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, CUSTOM, TRADE, QUIET ENJOYMENT, ACCURACY OF INFORMATIONAL CONTENT, OR SYSTEM INTEGRATION. NO WARRANTY IS MADE THAT ANY SOFTWARE WILL OPERATE IN AN ERROR FREE, UNINTERRUPTED OR COMPLETELY SECURE MANNER, IN COMBINATION WITH THIRD PARTY HARDWARE OR SOFTWARE PRODUCTS, OR THAT ALL DEFECTS CAN BE CORRECTED. YOU ACKNOWLEDGE THAT LICENSOR HAS NO CONTROL OVER THE SPECIFIC CONDITIONS UNDER WHICH YOU USE THE SOFTWARE. ACCORDINGLY, EXCEPT FOR THE FOREGOING EXPRESS WARRANTY, LICENSOR CANNOT AND DOES NOT WARRANT THE PERFORMANCE OF THE SOFTWARE OR ANY PARTICULAR RESULTS THAT MAY BE OBTAINED BY THE USE OF THE SOFTWARE. THE SOFTWARE AND SUPPORT DO NOT REPLACE YOUR OBLIGATION TO EXERCISE YOUR INDEPENDENT JUDGMENT IN USING THE SOFTWARE. The warranties made by Licensor may be voided by abuse or misuse of the Software and/or Support.

EXCLUSIVE REMEDY. Licensor's sole obligation and liability, and your sole and exclusive remedy under the warranties set forth in Section 9, shall be for Licensor to use commercially reasonable efforts to have the problem remedied, to re-perform Support services, to deliver to you a replacement for the defective Software, or to refund fees paid (in each case, as determined by Licensor and as applicable), provided that Licensor is notified in writing of all warranty problems during the applicable warranty period.

Third Party Software and Databases

This Software uses third-party software components from several sources. Portions of these software components are copyrighted and licensed by their respective owners. Various components require distribution of source code or if a URL is used to point the end-user to a source-code repository, and the source code is not available at such site, the distributor must, for a time determined by the license, offer to provide the source code. In such cases, please contact your Life Technologies representative. In addition, various licenses require that the end user receive a copy of the license. Such licenses may be found on-line as supporting files on the download page for the Software. In order to use this Software, the end-user must abide by the terms and conditions of these third-party licenses. You understand that third party products integrated into the Software or provided for use with the Software may be subject to additional terms and conditions and/or license agreements from the applicable third party vendor, which shall govern over conflicting terms of this EULA for purposes of your relationship with the third party vendor. You agree not to use any such third party product on a stand-alone basis independent of the Software, unless you have purchased the appropriate license from the third party vendor for use of such products. NOTWITHSTANDING ANYTHING TO THE CONTRARY IN THIS AGREEMENT, ALL THIRD PARTY SOFTWARE, DATABASES AND OTHER PROGRAMS AND SOFTWARE COMPONENTS ARE PROVIDED "AS IS" WITHOUT ANY WARRANTY WHATSOEVER FROM LICENSOR. ANY DATABASES OR OTHER INFORMATION PROVIDED BY LICENSOR ARE DESIGNED TO SUPPLEMENT OTHER SOURCES OF INFORMATION, ARE NOT INTENDED TO REPLACE YOUR PROFESSIONAL DISCRETION AND JUDGMENT AND

LICENSOR MAKES NO REPRESENTATIONS OR WARRANTIES REGARDING SUCH DATABASES OR INFORMATION, THEIR ACCURACY, COMPLETENESS OR OTHERWISE. Licensor agrees, upon request and as Licensor's sole liability and obligation, and for your convenience only, to have passed through to you (to the extent it may reasonably do so) any warranties and indemnifications provided by the applicable third party vendor of any third party products provided to you. To the extent any problem or liability arises from a third party product, you agree to seek recourse solely from the applicable third party vendor and not Licensor.

LICENSOR INDEMNIFICATION. Subject to the limitations set forth herein, Licensor agrees to defend you against any claims, actions, suits and proceedings brought against you by unaffiliated third parties arising from or related to a claim that the Software (other than any third party or open source components or elements) infringes upon such third party's copyrights, and Licensor agrees to pay all damages that a court finally awards to such third party, and all associated settlement amounts agreed to by Licensor in writing; provided that, Licensor receives from you (i) prompt written notice of the claim; (ii) all necessary assistance, information and authority necessary for Licensor to defend the claim and perform Licensor's obligations under this Section; and (iii) sole control of the defense of such claim and all associated settlement negotiations. If such a claim is made or appears likely to be made, you agree to permit Licensor to enable you to continue to use the affected Software, or to have the Software modified to make it non-infringing, or to have the Software replaced with a substantially functional equivalent. If Licensor determines that none of these options is reasonably available, then Licensor may terminate this EULA in whole or with respect to the affected Software product, and you may be entitled to a credit equal to the price paid for the affected product, less depreciation assuming a three (3) year useful life and straight-line depreciation. THIS SECTION STATES LICENSOR'S AND ITS AFFILIATES' ENTIRE OBLIGATION AND LIABILITY REGARDING INFRINGEMENT OF THIRD PARTY RIGHTS OF ANY KIND OR CLAIMS OF ANY SUCH INFRINGEMENT. Licensor will have no responsibility for (v) any use of any product after you have been notified to discontinue use because of a third party claim of infringement, (w) the alteration of the Software or the combination of the Software with third party materials, products or software, (x) use of the Software by any person or entity other than an Authorized User, (y) any misuse or unauthorized use of the Software, or (z) failure to use provided updated or modified Software to avoid a claim of infringement or misappropriation.

INDEMNIFICATION BY YOU. You agree to indemnify and defend Licensor, its licensors, and its affiliates, against any third party claims arising from or related to your use or misuse of the Software or any breach of the terms and conditions of this EULA, and you agree to pay all costs, losses, damages, and attorneys' fees that a court finally awards, and all associated settlements.

LIMITATION OF LIABILITY. EXCEPT TO THE EXTENT PROHIBITED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR'S OR ITS AFFILIATES' TOTAL, AGGREGATE LIABILITY ARISING FROM OR RELATED TO THIS AGREEMENT, THE SOFTWARE AND/OR SUPPORT (INCLUDING FOR NEGLIGENCE, STRICT LIABILITY, BREACH OF CONTRACT, MISREPRESENTATION, AND OTHER CONTRACT OR TORT CLAIMS), EXCEED THE AMOUNT OF YOUR DIRECT DAMAGES ACTUALLY INCURRED, UP TO THE AMOUNT OF FEES PAID TO LICENSOR UNDER THIS AGREEMENT FOR THE SOFTWARE PRODUCT OR SUPPORT PRODUCT THAT IS THE SUBJECT OF THE CLAIM UNDERLYING THE DAMAGES; OR, IN THE CASE OF SOFTWARE PROVIDED UNDER A TRIAL OR DEMONSTRATION LICENSE, ONE HUNDRED DOLLARS ($100.00), WHICHEVER IS LESS.

EXCLUSION OF DAMAGES. EXCEPT TO THE EXTENT PROHIBITED BY APPLICABLE LAW, UNDER NO CIRCUMSTANCES SHALL LICENSOR, ITS AFFILIATES, OR ANY OF THEIR SUPPLIERS OR LICENSORS BE LIABLE HEREUNDER FOR ANY OF THE FOLLOWING: (I) THIRD PARTY CLAIMS, EXCEPT AS PROVIDED IN SECTION 12, (II) LOSS OR DAMAGE TO ANY SYSTEMS, RECORDS OR DATA, (III) DIRECT DAMAGES FOR BREACH OF WARRANTY (IN RESPECT OF WHICH ANY LIABILITY SHALL BE LIMITED TO RE-PERFORMANCE OR REFUND AS SPECIFIED IN SECTION Exclusive Remedy. Licensor's sole obligation and liability, and your sole and exclusive remedy under the warranties set forth in Section 9, shall be for Licensor to use commercially reasonable efforts to have the problem remedied, to re-perform Support services, to deliver to you a replacement for the defective Software, or to refund fees paid (in each case, as determined by Licensor and as applicable), provided that Licensor is notified in writing of all warranty problems during the applicable warranty period.), AND/OR (IV) INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, PUNITIVE, RELIANCE, OR COVER DAMAGES (INCLUDING WITHOUT LIMITATION FOR LOST PROFITS, LOST SAVINGS AND DAMAGE TO ANY DATA OR SYSTEMS); EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND EVEN IF A LIMITED REMEDY SET FORTH HEREIN FAILS OF ITS ESSENTIAL PURPOSE. YOU ARE SOLELY RESPONSIBLE AND LIABLE FOR VERIFYING THE ACCURACY AND ADEQUACY OF ANY OUTPUT FROM THE SOFTWARE, AND FOR ANY RELIANCE THEREON.

THE FEES FOR THE SOFTWARE AND SUPPORT, THE REMEDIES SET FORTH IN THIS AGREEMENT, THE LIMITS ON LIABILITY SET FORTH IN SECTIONS Limitation of Liability. EXCEPT TO THE EXTENT PROHIBITED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR'S OR ITS AFFILIATES' TOTAL, AGGREGATE LIABILITY ARISING FROM OR RELATED TO THIS AGREEMENT, THE SOFTWARE AND/OR SUPPORT (INCLUDING FOR NEGLIGENCE, STRICT LIABILITY, BREACH OF CONTRACT, MISREPRESENTATION, AND OTHER CONTRACT OR TORT CLAIMS), EXCEED THE AMOUNT OF YOUR DIRECT DAMAGES ACTUALLY INCURRED, UP TO THE AMOUNT OF FEES PAID TO LICENSOR UNDER THIS AGREEMENT FOR THE SOFTWARE PRODUCT OR SUPPORT PRODUCT THAT IS THE SUBJECT OF THE CLAIM UNDERLYING THE DAMAGES; OR, IN THE CASE OF SOFTWARE PROVIDED UNDER A TRIAL OR DEMONSTRATION LICENSE, ONE HUNDRED DOLLARS ($100.00), WHICHEVER IS LESS. AND Exclusion of Damages. EXCEPT TO THE EXTENT PROHIBITED BY APPLICABLE LAW, UNDER NO CIRCUMSTANCES SHALL LICENSOR, ITS AFFILIATES, OR ANY OF THEIR SUPPLIERS OR LICENSORS BE LIABLE HEREUNDER FOR ANY OF THE FOLLOWING: (I) THIRD PARTY CLAIMS, EXCEPT AS PROVIDED IN SECTION 12, (II) LOSS OR DAMAGE TO ANY SYSTEMS, RECORDS OR DATA, (III) DIRECT DAMAGES FOR BREACH OF WARRANTY (IN RESPECT OF WHICH ANY LIABILITY SHALL BE LIMITED TO RE-PERFORMANCE OR REFUND AS SPECIFIED IN SECTION Exclusive Remedy. Licensor's sole obligation and liability, and your sole and exclusive remedy under the warranties set forth in Section 9, shall be for Licensor to use commercially reasonable efforts to have the problem remedied, to re-perform Support services, to deliver to you a replacement for the defective Software, or to refund fees paid (in each case, as determined by Licensor and as applicable), provided that Licensor is notified in writing of all warranty problems during the applicable warranty period.), AND/OR (IV) INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, PUNITIVE, RELIANCE, OR COVER DAMAGES (INCLUDING WITHOUT LIMITATION FOR LOST PROFITS, LOST SAVINGS AND DAMAGE TO ANY DATA OR SYSTEMS); EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND EVEN IF A LIMITED REMEDY SET FORTH HEREIN FAILS OF ITS ESSENTIAL PURPOSE. YOU

ARE SOLELY RESPONSIBLE AND LIABLE FOR VERIFYING THE ACCURACY AND ADEQUACY OF ANY OUTPUT FROM THE SOFTWARE, AND FOR ANY RELIANCE THEREON. AND THE OTHER PROVISIONS IN THIS AGREEMENT REFLECT THE ALLOCATION OF RISKS BETWEEN THE PARTIES. THIS SECTION IS AN ESSENTIAL ELEMENT OF THE BASIS OF THE BARGAIN BETWEEN THE PARTIES.

VERIFICATION. Licensor shall have the right to have on-site audits periodically conducted of your use of the Software. These audits are for license verification purposes only, will generally be conducted during regular business hours, and Licensor will use its reasonable efforts not to interfere unduly with your regular business activities. Licensor may also require you to accurately complete a self-audit questionnaire in a form Licensor may have provided. If an audit reveals unauthorized use, you must promptly purchase sufficient licenses and Authorized Scope to permit all usage disclosed. If material unlicensed use is found (*i.e.*, license shortage of 5% or more), you also shall reimburse Licensor for all costs incurred in connection with the verification, including without limitation reasonable attorneys' fees.

REGULATED USES. You acknowledge that the Software has not been cleared, approved, registered or otherwise qualified (collectively, "Approval") by Life Technologies Corporation with any regulatory agency for use in diagnostic or therapeutic procedures, or for any other use requiring compliance with any federal or state law regulating diagnostic or therapeutic products, blood products, medical devices or any similar product (hereafter collectively referred to as "federal or state drug laws"). The Software may not be used for any purpose that would require any such Approval unless proper Approval is obtained. You agree that if you elect to use the Software for a purpose that would subject you or the Software to the jurisdiction of any federal or state drug laws, you will be solely responsible for obtaining any required Approvals and otherwise ensuring that your use of the Software complies with such laws.

EUROPEAN COMMUNITY END USERS. If this Software is used within a country of the European Community, nothing in this Agreement shall be construed as restricting any rights available under the European Community Software Directive, O.J. Eur. Comm. (No. L. 122) 42 (1991).

LEGAL COMPLIANCE; RESTRICTED RIGHTS. The Software is provided solely for internal research and solely for lawful purposes and use. You shall be solely responsible for, and agree to comply with, all applicable laws, statutes, ordinances, and other governmental authority, however designated. Without limiting the foregoing, this EULA is expressly made subject to any United States government laws, regulations, orders or other restrictions regarding export from the United States and re-export from other jurisdictions of equipment, computer hardware, software, technical data and information or derivatives of such equipment, hardware, software or technical data and information. You agree to comply with all applicable export and re-export control laws and regulations in regard to products (including computer hardware, software, deliverables, technical data, source code, or any other technology, equipment, and/or derivatives of such hardware, software, deliverables, technical data, source code, equipment, or any other technology) received from Licensor. You further certify that you will not, directly or indirectly, without obtaining prior authorization from the competent government authorities as required by those laws and regulations: (1) sell, export, re-export, transfer, divert, or disclose technical data or dispose of any product or technology received from Licensor to any prohibited person, entity, or destination; or (2) use the product or technology for any use prohibited by the laws or regulations of the United States. You will reasonably cooperate with Licensor and will provide to Licensor promptly upon request any certificates or documents as are reasonably

requested to obtain approvals, consents, licenses and/or permits required for any payment or any export or import of products or services under this EULA, at Licensor's expense. Your breach of this provision shall constitute cause for immediate termination of this EULA. You agree to indemnify and hold harmless Licensor, its affiliates, and their respective officers, directors, employees and agents for your noncompliance with this Section. The Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101, consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212. Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4, all U.S. Government End Users acquire the Software with only those rights set forth herein.

GOVERNING LAW; SEVERABILITY. This EULA shall be governed in all respects by the laws of the State of California, USA, without regard to its conflicts of law rules or principles. Any dispute arising out of or related to this EULA shall be resolved only in the state or federal courts having subject matter jurisdiction in California. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods. Each party hereby consents to the exclusive jurisdiction and venue of such courts. If any provision of this EULA is held to be illegal or unenforceable for any reason, then such provision shall be deemed to be restated so as to be enforceable to the maximum extent permissible under law; the remainder of this EULA shall remain in full force and effect.

SOFTWARE MAINTENANCE AND SUPPORT. Licensor offers certain software maintenance and technical support service programs documented in Licensor's then-current Support policies ("**Support**"). Subscription to Support shall be governed by this EULA and by the terms and conditions set forth in such policies. Licensor shall have no obligation to provide Support if (a) the Software is not used in accordance with the Documentation or Authorized Scope; (b) the Software was modified by you; (c) you have not implemented all upgrades that would otherwise correct the problem; or (d) the problem is caused by your misuse, negligence or other cause within your control. Licensor may change its Support policies and prices at any time. Licensor reserves the right to discontinue Support services for any Software where Licensor generally discontinues such services to all Licensees of such Software, in which case such discontinuation shall not automatically terminate this EULA and the License. If you terminate Support and then re-enroll, Licensor may charge you a reinstatement fee.

GENERAL. This EULA, including any Orders, Support policies, and associated Licensor invoices (all of which are incorporated herein), are collectively the parties' complete agreement regarding its subject matter, superseding any prior oral or written communications, representations or agreements. In the event that any prior oral or written communication is in direct conflict with the terms of this EULA, this EULA shall control. You understand and agree that, to the extent Licensor permits you to use a non-Licensor purchase order or other form to order Software and/or Support, Licensor does so solely for your convenience. Any terms in any such forms that purport to vary or are in addition to or inconsistent with any terms in this EULA or in the applicable Order shall be deemed to be void and of no effect. Amendments or changes to this EULA must be in mutually executed writings to be effective. Sections 1, 4, 5, and 7 through 22, inclusive, shall survive the termination or expiration of this EULA. The parties are independent contractors for all purposes under this EULA. Neither party shall be liable for any delay or failure due to force majeure and other causes beyond its reasonable control; provided that the foregoing shall not apply to any of your payment obligations. Any notices under this EULA to Licensor must be personally delivered or sent by certified or registered mail, return receipt requested, or

by nationally recognized overnight express courier, to the address specified herein or such other address as Licensor may specify in writing. Such notices will be effective upon receipt, which may be shown by confirmation of delivery. All such notices shall be sent to the attention of the Chief Legal Officer of Life Technologies Corporation (unless otherwise specified by Licensor). You may not assign or otherwise transfer this EULA or any License without Licensor's prior written consent. This EULA shall be binding upon and inure to the benefit of the parties' successors and permitted assigns. You agree, at Licensor's request and reasonable expense, to provide reasonable assistance and cooperation to Licensor and its designees, and to give testimony and execute documents and to take such further acts reasonably requested by the other to acquire, transfer, maintain, perfect, and enforce Licensor's intellectual property rights as described in this EULA. To the extent you fail to do so, you appoint Licensor's or its affiliates' officers as your attorney in fact to execute documents on your (and your personnel's), successors' and assigns' behalf for this limited purpose.

v1.0

Glossary

alignment	The process of mapping sequencing reads to a reference genome or sequence.
alignment browser	An interactive software in which to view alignments of sequencing reads with the reference genome or sequence.
alignment score	Matching score; an optimal alignment is an alignment giving the highest score of matches.
allele	One of two or more alternative nucleotide sequences at the same location on homologous chromosomes.
analysis	A data analysis run in LifeScope™ Software; may include secondary analysis on data from multiple sequencer runs.
analysis type	An *analysis* based on common genetic-analysis experiment types. LifeScope™ Software provides: Genomic Resequencing Analysis, Targeted Resequencing Analysis, Low Frequency Variant Detection, Small RNA Analysis, and Whole Transcriptome Analysis.
annotated gene	Within one of several reference databases, a gene sequence that has biological attributes attached that describe structure or function, such as coding regions or biochemical function.
annotation	Biological attributes or metadata that are attached to sequence data or files. Examples include: genes and protein-coding features, and verified variants.
BAM file	The compressed binary version of the Sequence Alignment/Map (SAM) format, a compact and index-able representation of nucleotide sequence alignments. The BAM file is generated from the mapping step and has special attributes to support SOLiD® data.
barcode	A short, unique sequence that is incorporated into a library that enables identification of the library during multiplex sequencing.
barcode group, barcode pool	The conceptual grouping used during data analysis if more than one barcode is used during libration preparation from a single biological specimen.
barcoded library	A library that has a unique barcode sequence incorporated that enables identification of the library during multiplex sequencing.
base-space analysis	Analysis of SOLiD® sequencing data that has been converted from color space to a nucleotide sequence.

Bayesian	An algorithm that is used to evaluate the probability of the existence of a heterozygote or a non-reference homozygote when the coverage at the position is not high.
bead	The substrate to which the individual strands of DNA molecules in a SOLiD® library are attached.The covalently attached strand serves as the template for SOLiD® sequencing after the bead is immobilized on the flowchip surface.
BED file	BED format provides a flexible way to define the data lines that are displayed in an annotation track. The required BED fields are: • chrom - name of the chromosome • chromStart - starting position of the feature in the chromosome (the first base in a chromosome is numbered 0). • chromEnd - ending position of the feature in the chromosome
BEDGRAPH	In targeted resequencing, a text file in BEDGRAPH format that describes the depth of coverage within targets by on-target alignments.
bisulfite read	A sequence read of bisulfite-converted DNA, in which unmethylated cytosine residues have been converted to uracil. Used in methylation analysis of genomic DNA.
BLAST	In bioinformatics, Basic Local Alignment Search Tool, or BLAST, is an algorithm for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or the nucleotides of DNA sequences.
call stringency	The criteria for calling a nucleic acid species or genetic variant present in the biological sample.
calling threshold	The criteria for calling a nucleic acid species or genetic variant present in the biological sample.
ChIP-Seq	Sequence analysis of genomic regions associated with DNA-binding proteins.
classic mapping	An algorithm that searches for matches at the full read length, allowing up to a user-specified number of mismatches. Now called global mapping.
clear zone	The threshold to decide whether a read is mapped uniquely in the reference. Clear zones are used in previous versions of the software, but are not used with LifeScope™ Software.
clipping, hard clipping	An operation that codes a portion of a color alignment with a number of mismatches that prevent seed or anchor extension.
clone	A single bead with a clonal population of templates for sequencing, generated during templated bead preparation/emulsion PCR.
color balance	The relative proportion of beads in a given sequencing cycle that are called as each of the four colors.
color call	For each SOLiD® chemistry cycle, the color that is determined for each bead.

color space, color-space analysis	Color space data is a sequence of colors obtained from 2-base encoded probes in SOLiD® System sequencing, representing a series of overlapping dinucleotides. Color space data is converted to a nucleotide sequence by aligning to a reference.
complexity	The size of the genome that is composed of unique sequences. With respect to SOLiD® libraries, the number of independent molecules derived from the original DNA input, not from amplification.
consensus calls	A file that lists the SNPs called with the SNP Finding algorithm, and provides general information about each position.
contig	A nucleotide sequence that has been assembled from shorter, overlapping sequences.
core	A single core is equivalent to a processor on a traditional computer. Multiple cores increase the performance and efficiency of a computer that is running multiple programs.
counting	Expression analysis that focuses on relative or absolute quantification of RNA molecules in a biological specimen.
copy number variation (CNV)	A variation in the number of copies of DNA segments among individuals in a population. The length of the segment can vary, and it is typically more than a few base pairs. Some use a broader definition that includes any insertion, deletion, or duplication longer than a few base pairs.
CountTags	A module to count the number of reads that align within genomic features.
counting	Sequence analysis that generates read or tag counts for annotated regions of a reference sequence.
coverage	• In genomic analysis, the number of aligned (or mapped) sequencing reads that span a position in the reference genome. • In RNA analysis, this term is sometimes used to describe the fraction of the reference sequence that has sequencing reads aligned (or mapped), at a certain calling threshold.
csfasta file	The SOLiD® System produces color-space sequence reads in a fasta-format labeled as CSFASTA. These reads can be retained and analyzed in color space.
dbSNP	A Single Nucleotide Polymorphism database repository for single nucleotide polymorphisms and short insertion and deletion polymorphisms, hosted by the NCBI.
de novo sequencing	The initial generation of the primary genetic sequence of a particular organism, without use of a reference genome or sequence.
deletion	A gap in a nucleotide sequence with respect to a reference genome or sequence.
dibayes	Analysis module to identify SNPs from mapped and processed SOLiD® System color space reads. Also called diBayes and SNP Finding.

discordant	In large indel analysis, mate-paired reads that have been mapped to a reference genome, and whose inter-read distance deviates significantly from the expected insert size of the mate-paired library as a whole. Discordant pairs have insert sizes that deviate significantly from the expected value. Discordant pairs containing a putative deletion appear larger when mapped to the reference.
EBI	European Bioinformatics Institute.
enriched data	Sequencing data that is associated with a region of interest in a targeted resequencing experiment or in a low frequency variant detection workflow.
enrichment statistics	Calculated results derived from the targeted resequencing enrichment module in LifeScope™ Software. Also generated in low frequency variant detection workflows.
ENSEMBL	A genome browser. **www.ensembl.org.**
error correction	(SAET) This step takes each read in the input file and attempts to correct it if any of its k-mers are not present in the spectrum. Use of multi-threaded versioning may be recommended for speed of process.
Error expectation metric (EEM)	A whole transcriptome pipeline, a parameter that is used in the calculation of the Junction Confidence Value metric, to determine the statistical significance of detected variant fusion transcripts.
evidence graph	A visual representation of the data structure of a candidate fusion transcript.
exon	In eukaryotic organisms, a segment of a gene that encodes part or all of a protein. Exons may be separated by introns that are spliced out of the primary transcript to produce a mature mRNA for translation into protein.
exon mapping	A whole transcriptome pipeline, a step in which reads are mapped to the set of exon sequences defined by the genome annotation.
F3 tag	Sequencing data derived from the P1 end of the template in the SOLiD® templated bead, using forward ligation chemistry. The F3 tag is generated using the SOLiD® FWD1 Seq. Primers or the SOLiD® Small RNA Seq. Primers. See the *5500 Series SOLiD® Sequencers: Reagents and Consumables Ordering Guide* (Part no. 4465650) for an illustration.
F5 tag	Sequencing data derived from the P2 end of the template in the SOLiD® templated bead, using reverse ligation chemistry. The F5 tag is generated using the SOLiD® REV1 (DNA) Seq. Primers or the SOLiD® REV1 (RNA) Seq. Primers. See the *5500 Series SOLiD® Sequencers: Reagents and Consumables Ordering Guide* (Part no. 4465650) for an illustration.
false positive	Sequence variants that are present in the output data from a sequencing run but are not present in the biological source.

filter	A reference sequence of interest that is used to select reads that align with or map to the reference sequence for further analysis.
filter mapping	Alignment of sequencing reads to a reference sequence of interest, to select reads for further analysis.
Frequentist	An algorithm for finding SNPs (alternative to SNP Finding algorithm).
function code (dbSNPs)	A code in the dbSNP database that indicates the consequence, if any, of a SNP to the transcript in which it is located.
fusion transcript	An RNA molecule that results from transcription of a gene fusion. See also *gene fusion*
gapped alignment	A read alignment to the reference sequence that indicates an insertion or deletion. The pairing algorithm searches for gapped alignments (indels) when one of the tags (F3/R3/F5-P2) maps to the reference genome and the other tag does not map to the genome within the insert-size range. Small indel analysis calls indels from a consensus of gapped alignments, using the BAM file as input.
gene fusion	A section of the genome that maps to an exon from one gene followed by an exon from another gene. It can occur as the result of a translocation, deletion, or chromosomal inversion. A gene fusion junction excludes exon-to-exon boundaries that arise from alternative splicing of a transcript.
genomic classification table	For mate-paired or paired-end reads, a code that describes the strandedness, distance, and orientation of the two reads.
genomic mapping	See *mapping*.
genomics	Global analysis of the genome to discern elements involved in regulation of gene activity or expression, with an emphasis on genetic variation such as single nucleotide polymorphisms, small and large insertions and deletions, and other structural variants such as translocations and inversions. Some use the term genomics to as an umbrella term that includes transcriptomics, epigenomics, and analysis of the genome.
group	A collection of read-sets that are required to be analyzed together, in groups that you define. Reads that are grouped together are treated as one specimen, even if the reads are in different *.xsq input files.
hemizygous	A small indels tool, a parameter that indicates that the reference allele and the small indel are both present.
hg18, hg19	Reference sequence assemblies of the human genome using the nomenclature used by the UCSC Genome Browser.
highly expressed junctions	A whole transcriptome pipeline, high-count sequencing reads that span an exon-exon junction.
homozygous	In diploid organisms, having two identical alleles in the corresponding genes.

HUGO-style gene names	Gene names following the Human Genome Organization style.
HuRef	A diploid human genome sequence of one individual (J. Craig Venter).
indel	A difference in sequence due to either an insertion or a deletion event; especially used when the evolutionary direction of the change is unspecified.
insert size	The physical size of the genomic DNA segments or RNA molecules represented in a SOLiD® library. • Fragment libraries: the size of the sheared DNA fragments. • Mate-paired libraries: the length of the genomic DNA segment spanned by the corresponding mate-pair tags. • Whole transcriptome libraries: the size of the RNA fragments.
insertion	An insertion of nucleotide sequence with respect to the reference genome or sequence.
intron	The genomic sequence between two exons that is spliced out of a primary transcript prior to translation. See *exon*.
inversion	A segment of DNA that is in its native location but is in the reverse orientation.
junction	A place where two regions that are not contiguous in the genomic sequence are joined in a single sequenced region under consideration.
Junction Confidence Value (JCV) metric	Whole transcriptome pipeline, a statistical confidence metric to detect false positives and assign a confidence level to a detected junction.
junction mapping	Whole transcriptome pipeline, a mapping step that incorporates exon, gene, and transcript definitions in the genome annotation for that region.
junk seed	An initial alignment in a seed-and-extend algorithm that is not aligned with the correct sequence
large indel	An insertion or deletion, with respect to a reference sequence, of more than 200 bp.
LB field	A field in a BAM file that contains library type information.
library	A set of DNA or cDNA molecules prepared from the same biological specimen and prepared for sequencing on the SOLiD® System.
local mapping	The initial alignment step that starts with locating short matches between a read and the reference sequence.
locus-spanning	Pairs of reads derived from mate pairs or paired-end reads that span a relatively large distance in the reference sequence.

Map Fragment data tool	Tool that maps sequence data from fragment libraries.
mappability	In the CNVs analysis module, the fraction of mappable bases in the candidate CNV region.
mapping	The process of aligning sequencing reads to a reference genome or sequence.
mate alignment	The process of mapping paired-end or mate-paired reads to the reference sequence, taking into account the proximity of the reads on a template clone.
mate-paired library	Library consisting of two DNA segments that reside a known distance apart in the genome, linked by an internal adaptor, and with P1 and P2 Adaptors ligated to the 5' and 3' ends of the template strand, respectively.
mates	Sequencing reads that are linked through their origin in a mate-paired library or a paired-end sequencing run.
mer	The number of nucleotides in a sequencing read.
merging (WTA)	Algorithm parameter in paired-end WTA analysis that enables the merging of genomic and junction mappings.
methylation analysis	The study of how methylation of nucleic acids is involved in DNA structure and control of gene expression.
module	A virtual piece of LifeScope™ Software, such as mapping, SNP Finding, or CNV.
NCBI	National Center for Biotechnology Information.
node	A node is a single computer that is connected to the network. A multi-node cluster has several nodes (computers).
normalization	In general, the process of comparing an experimental measurement to a reference measurement. In the CNV analysis module, the process of comparing the relative coverage of a region of interest to the global coverage, based on the human reference genome. In targeted resequencing, the process of adjusting the amplicon amounts after PCR enrichment and before library preparation, so that amplicons are represented equally in the library. In the whole transcriptome pipeline, the parameter RPKM is a normalized measure of gene expression.
paired-end sequencing	Sequencing runs that acquire sequence from each end of the insert in a DNA fragment or whole transcriptome library, using both forward and reverse reads.
pairing	An algorithm that occurs after mapping in the LifeScope™ Software workflow

pairing distances	Pairing distances (sometimes called insert sizes) for each pair are assigned during the mapping/pairing pipelines and subsequently used by the large indel tool to determine indel candidacy.
parallelization	A tertiary analysis tool by which you can split spectrum generation into multiple jobs, where each job generates a subspectrum from the subset of reads.
pipeline	Sets of steps or modules that are used in sequence for different data analysis goals, such as whole transcriptome analysis or methylation analysis.
ploidy	The number of chromosome sets in a cell.
polyploidy	The number of chromosome sets in a cell.
polymorphism	A genetic variant in a population of individuals that may or not may be associated with an observable (phenotypic) trait.
primer set	In the SOLiD® System, the set of primers that are used sequentially to initiate ligation sequence chemistry.
project	A project is a container for your analysis runs. You create your projects. Within each project, you create one or more analyses. Each project is private to an individual LifeScope™ Software user.
p-value	The statistical significance of an alignment score is frequently assessed by its p-value, which is the probability that this score or a higher one can occur simply by chance, given the probabilistic models for the sequences.
quality value	An empirically defined value based on a phred-like score equating to the confidence that the color called for that cycle is the correct one. In general, the brighter the bead, the greater the difference in signal between the primary and secondary colors, the higher the quality value (QV).
R3 tag	The R3 tag applies only to mate-paired libraries; sequencing data derived from the mate-pair tag closer to the P2 end of the SOLiD® templated bead, using forward ligation chemistry. The R3 tag initiates in the IA sequence using the SOLiD® FWD2 Seq. Primers. See the *5500 Series SOLiD® Sequencers: Reagents and Consumables Ordering Guide* (Part no. 4465650) for an illustration.
raw reads	A format in which the nucleotide sequence appears without headers or comments.
read, sequencing read	Sequencing data from a single bead with a single primer set.
read-set	A group of reads belonging either to one .xsq file (in the case of non-indexed data) or to one barcode from one *.xsq file (in the case of indexed data).
read-set group	A collection of similar read-sets designated by a user.

read-set repository	A storage place in LifeScope™ Software for instrument data intended to be input data for LifeScope™ Software analyses.
reference, reference genome, reference sequence	A sequence against which sequencing reads are aligned before further bioinformatics analysis.
reference file	A file containing the reference sequence information, usually in the *.fasta format.
RefSeq	A multi-organism database archive of DNA, RNA, and protein sequences, hosted by the NCBI.
repository	A virtual container with your reference files, reads files, and projects. Repositories include: • Projects, which contains your projects. • Reads (read-sets), which contains your .*.xsq data files. • References, which contains the reference genomes used in your analyses.
rescue	An alignment method that is applied to read pairs that have at least one alignment, but no pair of alignments occurring within an expected range. Use the rescue method to find additional alignments.
rescue distance	The maximum distance x for a pair considered to be properly paired (SAM flag 0x2) is calculated by solving equation Phi((x-mu)/sigma)=x/L*p0, where mu is the mean, sigma is the standard error of the insert size distribution, L is the length of the genome, p0 is prior of anomalous pair and Phi() is the standard cumulative distribution function.
resequencing	The process of genomic sequencing in cases where a reference sequence already exists, and the new sequence is compared to the reference.
resource manager	Software for executing batch and interactive jobs on a cluster of networked computers.
RNA-Seq	Gene expression analysis using sequence-based approaches. RNA-Seq can include whole transcriptome analysis, small RNA analysis, and SOLiD® SAGE™ analysis.
RPKM	The number of reads mapping to a transcript per kilobase of transcript length per million mappable reads. RPKM is used to set a threshold for calling a transcript or new RNA species or isoform "present." 1 RPKM is equivalent to 20 reads mapping to a 1 kb transcript per 20×10^6 mappable reads.
SAET	SOLiD® Accuracy Enhancer Tool for correcting spectral-alignment errors in raw data; reduces color-calling error rate without alignment to reference genome. SAET takes as input a file with SOLiD® reads in .csfasta format and outputs corrected reads and quality values into new .csfasta and .qual files, respectively.

SAGE™ analysis	(Serial Analysis of Gene Expression) Nucleotide sequence analysis seeking to find specific gene expression information using short stretches of cDNA (also known as *tags*) from the 3' ends of RNA molecules. In the SOLiD® System, the SAGE tag is 25–27 bp in length.
sample, barcoded sample	In the 5500 Series Genetic Analysis Systems , the set of templated beads that will be sequenced in a single flowchip lane. A barcoded sample contains templated beads from up to 96 barcoded libraries.
SAMtools	SAMtools provide various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format. SAMtools is hosted by SourceForge.net.
Satay plot	A cross-axis graph displaying the spectral purity and signal intensity of beads in a sequencing run that meet defined thresholds in each parameter (*on-axis beads*). *Best beads* meet the highest stringency thresholds; *good beads* meet less stringent thresholds.
secondary analysis	Sequence alignment and mapping to a reference sequence.
seed	A field: size of seed used in spectrum construction. Seeding: Using a heuristic method, finding homologous sequences, not by comparing either sequence in its entirety, but rather by locating short matches between two sequences.
seed-and-extend	Seed-and-extend algorithms map reads to a reference genome. Certain softwares can map reads with any number of differences or mismatches using the observation that for an r bp read to align to the reference with at most k differences, the alignment must have a region of length s=r/k+1 called a seed that exactly matches the reference.
sense strand	The strand of DNA with the same nucleotide sequence as that of the corresponding mRNA. Also called the *coding strand*.
sequence trimming	See *trimming*.
sequencing sample	A physical pooling of one or more library preparations before templated bead preparation. The sequencing sample can be used on one or more lanes on a flowchip.
single read	Forward-only sequence data that generates the F3 tag.
small indel	An insertion or deletion, with respect to a reference sequence, of less than 10 base pairs in length.
small RNA analysis	Global sequence analysis of the small RNA population of a cellular RNA sample; small RNA includes microRNAs (miRNAs), short interfering RNAs (siRNAs), piwi-interacting RNAs (piRNAs), and repeat-associated siRNAs (rasiRNAs).
small RNA library	A SOLiD® System-compatible library that is prepared from the small RNA fraction of a total RNA sample.
SNP	Single Nucleotide Polymorphisms (SNPs); single base pair variants in genomic DNA or the corresponding RNA transcript.

SNP Finding	An analysis module to identify SNPs from mapped and processed SOLiD® System color space reads. Also called diBayes.
spectrum building	(SAET) When a temporary file with spectrum constructed from all reads generates in the output directory. This process is fast, resource less, and creates temporary swap files when memory usage is above 2GB.
splice junction	Exon-to-exon boundaries that arise from alternative splicing of a transcript. See also *gene fusion*.
splicing	The process whereby introns are removed from a primary mRNA, resulting in a mature mRNA that is ready for translation into protein.
strandedness	The polarity or orientation of a nucleic acid strand with respect to being sense or antisense. Libraries prepared using the SOLiD® Total RNA-Seq Kit preserve the strandedness of the original RNA molecule such that F3 tag reads align to the sense strand and F5 tag reads align to the antisense strand.
stringency	Stringency is one criterion used to filter out junk reads in ReadBuilder stage. Specifically, in the SNP Finding tool, the stringency parameter (call.stringency) defines the criteria by which to report SNPs in analyzing genomic data. There are several main empirical stringency settings (low_frequency, low, medium, high, highest).
tag	A tag is: • Sequencing data from a single bead with a single primer set; sometimes used interchangeably with read, sequencing read. • A length of DNA or cDNA to be sequenced; especially, a relatively short stretch of DNA or cDNA that is used to infer information about the longer native molecule from which it is derived, such as in mate-paired library sequencing and SAGE™ analysis, respectively.
tags: BC, F3, F5, R3	Sequencing data derived from specific locations on theSOLiD® templated bead. See the *5500 Series SOLiD® Sequencers: Reagents and Consumables Ordering Guide* (Part no. 4465650) for an illustration.
targeted resequencing	Comparative sequence analysis of selected candidate genes or regions to discover genetic variants and mutations.
template strand	The strand of each DNA molecule that is covalently attached to SOLiD® P1 beads during templated bead preparation and that serves as a template for SOLiD® System sequencing.
tertiary analysis	Data analysis that takes place after mapping and alignment.
threading	The ability of an operating system to execute different parts of a program, called threads, simultaneously. A multi-threaded software program is designed so that all the threads can execute at the same time without interfering with each other.

transcript rescue distance	See *rescue distance*.
transcriptome	The compilation of all transcribed sequences from a genome, both coding and non-coding.
transition	A single nucleotide (point) mutation that changes a purine nucleotide to another purine nucleotide (for example, A to G), or a pyrimidine nucleotide to another pyrimidine nucleotide (for example T to C).
translocation	A mutation in which a chromosomal segment changes position, usually moving from one chromosome to a different, nonhomologous chromosome.
transversion	A single nucleotide (point) mutation that changes a purine nucleotide to a pyrimidine nucleotide (for example, A to C), or a pyrimidine nucleotide to a purine nucleotide.
trimming	Filtering noise algorithmically which removes 5′ overhangs and shortens 3′ overhangs, typically to 4–5 bases. Trimming reads requires both modifying the .csfasta reads file to replace the color to be trimmed with a dot (“.”), and modifying the mapping parameters to account for the alignment change.
trusted frequency (SAET)	(SAET) Use this developer option (-trustfreq freq) to overwrite estimated frequency cutoff of trusted seeds. All seeds with frequency < “freq” are filtered out of spectrum.
trusted seed	(SAET) If globally computed cutoff for frequency of trusted seeds does not meet your purpose, e.g., it is too low and too many junk seeds are considered correct or it is too high and many correct but low frequency seeds are filtered out, then use -trustfreq option to overwrite estimated frequency cutoff.
ungapped alignment	See *gapped alignment*.
uniquely placed reads	A read that is mapped only once in a genome with a given number of mismatches.
variant	A difference in the nucleotide sequence of interest, with respect to the reference sequence.
visualization	Viewing mapped reads on any number of publicly available genome browsers.
weighting	A weighting scheme can outperform a simple binary scheme traditionally applied in genomic analysis, independent of the organism, and used to improve the quality of the data.
whole transcriptome analysis	Global sequence analysis of coding and non-coding RNA transcripts along their entire length.
whole transcriptome library	A SOLiD® System-compatible library that is prepared from total or poly(A) RNA that enables sequence analysis of the transcripts along their entire length.

workflow	Configuration files that are used for common end-to-end analysis runs, including multiple modules. LifeScope™ Software runs these common workflows: resequencing, targeted resequencing, low frequency variant, whole transcriptome, small RNA, Methyl Miner, and ChIP-Seq.
WTA	See *whole transcriptome analysis*.
*.xsq	eXtensible SeQuence, an extensible file format for storing sequence data. A binary sequence output file generated by the instrument. This file contains primary analysis results for a single lane in a flowchip.
z-normalize	Parameters called moving statistics, such as the number of locus-spanning pairs, and the sample average insert size, are calculated from a subset of pairs at each genomic position, yielding the absolute insert size deviation between the sample and the population in units of standard deviation. This normalization step allows multiple libraries with variable insert sizes to be combined into one analysis.
zygosity	The combination of *alleles* at a site in a nucleotide sequence; for example, homozygous reference allele, homozygous non-reference allele, or heterozygous.

Documentation

Related documentation

For the latest documentation on LifeScope™ Genomic Analysis Software, go to:

http://www3.appliedbiosystems.com/AB_Home/Support/index.htm

Document	Part number	Description
LifeScope™ Genomic Analysis Software User Guide	4471877	Describes the bioinformatics analysis framework for flexible application analysis (data-generated mapping, SNPs, count reads, etc.) from sequencing runs.

Obtaining support

For the latest services and support information for all locations, go to:

www.appliedbiosystems.com

At the Applied Biosystems website, you can:

- Access worldwide telephone and fax numbers to contact Applied Biosystems Technical Support and Sales facilities.
- Search through frequently asked questions (FAQs).
- Submit a question directly to Technical Support.
- Order Applied Biosystems user documents, SDSs, certificates of analysis, and other related documents.
- Download PDF documents.
- Obtain information about customer training.
- Download software updates and patches.

Index

Numerics

2BE 494
4BE 494

A

administrative shell commands 96, 565
alignment 583
  alignment browser 583
  alignment format 493
  alignment score 583
  annotation-aided alignment 365
  basic alignment information 493
  color 506, 507
  format 494
  gapped 147, 367
  gapped alignment 587
  mate 589
  nucleotide sequence 34
  score 506
  unmapped and secondary 494
  viewing and inspection 55, 507
allele 583
analysis 583
  methylation 589
  SAGE 592
  secondary 592
analysis groups 104
analysis type 583
analysis.regions.file 560
  example command 84
  with small indel module 241
annotated gene 583
annotation 291, 583
  CNV output file 316
  conflicts 292
  custom dbSNP build 297
  custom reference 297
  dbSNP 130 build 292
  dbSNP 131 build 292
  dbSNP files 299
  dbSNP tables 292
  download dbSNP files 298
  Ensembl GTF files 297
  error handling 323
  filtered variants output files 308
  filtering options 292
  function codes 305
  gene features 305
  hg18 reference build 292
  hg19 reference build 292
  input files 299
  large indels output file 314
  memory requirements 301
  mutated genes output file 317
  optional annotation sources 296, 297
  options 292
  output annotation 304
  parameters table 301
  refGene.txt 297
  small indels annotated output file 311
  SNPs output file 309, 319, 321
  sources 299
  supported attributes 322
  transitions and transversions 294
  UCSC GTF file 297
  variant overlapping gene or exon 292
  variant statistics output file 309, 314, 316
  variants appearing in dbSNP 293
  variants in exons 293
  variants in genes 293
annotation parameters 301
  annotation parameters table 301
  filtering parameters 302
  source parameters 301
authentication 43

B

BAM file 583
  @RG header field required 499
  header requirement 498
  indel alignments 508

pairing information 505
Picard validation 499
position errors 207
with samtools command 498
WT format differences 381
WT optional fields 381
bamgen parameters 148, 371, 432
bamgen.mqv.threshold 148, 369, 432
BAMStats 452
megabases of coverage 176, 390, 448
NumFragmentsPassingFilters 175
NumFragmentsTotal 175
NumMapped 175
output 169, 171, 384, 443
output description 175, 390, 447
output files 386, 444
overview 167, 383, 441
parameters table 168, 169, 383, 384, 441, 442
report files 171
barcode 94, 148, 583
barcode group 583
barcode pool 583
barcoded library 583
barcoded sample 592
See also index ID
base calls 452
base-space analysis 583
base-translation algorithm. *See* refcor
Bayesian 584
bead 584
BED file 584
BED format 420, 474, 510
bedgraph 342, 345, 584
Binary Alignment Map 34
bisulfite reads 478, 584
BLAST 584
browser, supported versions 68

C

call stringency 584
calling threshold 584
character restrictions 81
ChIP-Seq 471, 584
analysis tools 471
BAM-to-BED format converter 474
ChIP-Seq analysis software tools 475
ChIP-Seq Map Data 471
instructions 471
chromosome files 513
CIGAR string 382
Circos 412, 416, 421
clear zone 584
clipping 584
H operation 506
hard 506
hard clipping 506, 507
hard clipping example 506
refcor parameter 432
splice finder parameter 411
WT clipped records 382
WT soft-clips 382
clone 277, 281, 584
CNV
algorithm description 221
CG correction 221
coverage 222
FAQ 224
output file examples 220
output file format 219
output files 218
CNV parameters 215
CNV parameters table 215
CNV stringency settings 215
color balance 584
color call 584
color space 263, 585
analysis in color space 585
color attribute tags 504
color quality value 585
refcor color sequence parameter 435
command shell
commands table 89
common analysis scenarios 80
concepts 71, 72
grouped analysis 101
groupsfile 101
help 97
man page 97
modes 85
modes of operation 86
naming restrictions 76
options 97
reference files 78
repositories 76
review job status 107
review results 108

run mode 86
run mode commands 87
scripted mode 88
shell mode 89
status mode 87
terminology 71, 72
complexity 585
concordant 585
configuration files 553
consensus calls 585
consensus_calls file 34, 198, 202, 209, 212
contig 585
convertToXSQ.sh 520
copy number variation (CNV) 585
Count Known Exons 404
algorithm description 405
parameters 404
counting 585
Counts tool 404, 407, 417
CountTags 585
CountTags tool. *See also* Count Known Exons
coverage 585
csfasta 34, 504, 505
csfasta file 585

D

dbSNP 585, 587
annotation files 300
build 131 301
de novo sequencing 585
deletion 585
demo examples
instructions 59
not recommended for new users 59
optional 59
diBayes 188, 206, 207, 209
dibayes.call.stringency 196
discordant 586
distance rescue 591
documentation, related 597

E

EBI 586
ECC data 494
not supported on PE R3 tag 494
with SAET 431
email notification 561
enriched data 586
enrichment
in the targeted resequencing workflow 339
regions of interest file 340
enrichment parameters
enrichment parameters table 341
enrichment statistics parameters 342
enrichment statistics 586
aligned reads input 344
genome coverage frequency file 346
input files 343
output alignment file 345
output bedgraph file 345
output files 344
target coverage frequency file 347
target regions file 343
target statistics report file 347
Ensembl 586
Ensembl GTF file 297, 359, 550
reformat_ensembl_gtf.pl 298
error correction 135, 586
See also SAET
Error expectation metric (EEM) 418, 586
evidence graph 586
Example steps to run a standard workflow 84
examples of INI files 555
exon 409, 586
exon mapping 363
Exon-1 419
Exon-2 419
exon-exon boundaries 409
exon mapping 586

F

F3 tag 586
F5 tag 586
false positive 586
FAQ 485
CNV 224
general 485
large indels 289
mapping 179
pairing 182
small indels 272
SNPs 209
file formats 34, 198, 493, 505
file types 34

filter 587
filter mapping 587
filter reference fasta 34
fragmap.max.number.of.jobs 152, 371, 436
fragmap.maxreads.per.node 152, 372, 436
fragmap.minreads.per.node 152, 371, 436
fragmap.processors.per.node 152
framework parameters
  general parameters 559
  read-only parameters 560
  reference-related parameters 433, 559
  with the set reference command 433, 559
Frequentist 587
function code (dbSNPs) 587
fusion junction 354, 416
fusion transcript 587
fusion, gene 587

G

gap alignment 147, 259, 261, 264, 367
  detection 257, 259
gapped alignment 587
gene 550
  gene annotations 357
  gene orientation 353
  HUGO-style gene names 551
gene fusion 587
genome
  annotations 358, 512
  reference 591
genomic browser
  IGV 481, 507
  UCSC Genome Browser 507
genomic classification table 587
genomic classifications 182, 394
genomic mapping 587
genomics 587
get INI files 91, 123
get PLN files 91, 123
group 587
  BAM @RG field 498
  described 71, 73
  group BAM files 104
  read-set groups 93
  running a grouped analysis 101
  set group command 93
groupsfile 73, 103, 104
GTF files 512
  annotation source 292
  converting 551
GUI guide 24, 597

H

hard clipping 584
hemizygous 587
hg18
  and the CNV tool 224
  annotations based on hg18 296
  genomic reference download from UCSC 512
  hg18 reference build 292
  human hg18 reference 289
hg18, hg19 587
hg19
  annotations based on hg19 296
  memory requirements 301
highly expressed junctions 587
homozygous 587
HUGO-style gene names 588
HuRef 588
hyper-threading 486
hyphen, restrictions 76

I

IGV viewer 420, 481, 507, 510
import statement restriction for global.ini files 556
increase Java heap space 487
indel 588
  large 588
  small 592
indel alignments 508
index ID 100
  in a groupsfile 99, 101
  in BAM file name 167
  shown by the shell cat command 94
  shown by the shell ls command 90
  when specifying input data 98, 139
  with the add xsq command 92
  with the set groups command 93
INI files
  analysis module INI files 555
  examples 555
  explained 555, 558
  import statement 556
  import statement restriction 556

parameters 558
run parameters 556
insert size 588
insertion 588
installation
authentication realms 61
cluster 43
computer date 48
copy license file 52
data drive 540
demos as installation verification 59
DNS prerequisites 48
hardware configuration 42
instructions 44
license key 48, 51
license key activation 51
license server configuration 46
MAC address 49, 50, 51
remote submission 47
remote workstation 43
uninstall 64
upgrade 64
user accounts 61
user authentication realms 43
verification 58
verification tests 60
workstation 42
inversion 588
ambiguous tag placement 236
breakpoints 235
candidate breakpoints 237
GFF file format 233
GFF output file format 231
inversion polymorphisms 234
inversion thresholds 238
inversion tool workflow 236
inverted mate-pair 235
inverted mates 236
output files 233
rank.txt file format 233
ranking 238
rescore.txt file format 234
scoring 237
small inversion detection 236
tiny inversions 238
window size 237
inversion parameters
inversion threshold parameters 238
inversion.breakpoint.peak.width 237
inversion.breakpoint.rescue 237
inversion.breakpoint.score.threshold 237
inversion.max.bxx.mp.length 238
inversion.max.inversion.length 238
inversion.max.length.tiny.inversions 238
inversion.min.pairing.quality 236
inversion.min.qv 236
inversion.recover.tiny.inversions 236, 238

J

JCV metric 588
job queues 486
junction 588
splice 593
Junction Confidence Value (JCV) metric 417, 588
junction mapping 588
junk seed 588

L

large indel 588
candidate deviations 283
candidate indels 283
detection 280
determining zygosity 284, 285
FAQ 289
module flow 280
output file formats 278
pairing distances 281
parameter optimization 285
parameter table 277
run time 290
space requirements 290
zygosity 277
large indel parameters 276
parameter table 277
resource parameters 278
LB field 588
legacy format translation
@HD SO field 515
@RG LB field 515
@SQ UR field 515
##color-code 515
##history 515
##line-order 515
##max-num-mismatches 515
##max-read-length 515
##primer-base 515
##reference-name 515

library 588
mate-paired 589
small RNA 592
license activation 51
license file 51
licensing server
check status 53, 69
how to start 53
how to stop 63
when not to stop 63
LifeScope™ Software URL 62
local mapping 588
locus-spanning 588
log files 489
low frequency variant detection 333
read selection algorithm 335
running analyses 334
SNPs optimizations 336
SNPs parameters 194
supported analyses 338
lscope.sh 85
commands 89
options 97
usage 86

M

mail command, for completion notification 561
Map Fragment data tool 589
mappability 589
mapping 589
FAQ 179
filter 587
frag mapping of paired data 154
genomic 587
global mapping 584
junction 588
local 588
local alignment 151, 156, 158, 159, 179, 182
mapping quality value 146
performance 152, 371, 436
quality 405, 516
repetitive scheme 179, 450
schema lines 152, 372, 436
split into multiple jobs 152, 371, 436
mapping parameters
fragmap.mi.reads.per.node 152, 371, 436
fragmap.minreads.per.node 152, 371, 436
fragmap.number.of.nodes 152, 371, 436
mapping parameters table 150, 153
mapping.memory 150, 152, 372, 435, 436
mapping.min.reads 150, 370, 376, 434
parallelization 152
mapping statistics
See BAMStats
mark duplicates
compared to Picard 184
module 144
process.one.bamfile.only parameter 144
matching file, .ma 589
mate alignment 589
mate-paired library 589
mates 589
MaToBam 34
max.head.node.java.heap.space property 487
memory.request 133, 195, 217, 230, 245, 278, 302, 343, 384, 400, 404, 413, 491
annotations 302
CNV 217
inversions 230
large indel 278
SAET 133
small indel 245
small RNA count 464
small RNA coverage 458
small RNA mapping 442
SNPs 195
targeted resequencing 343
whole transcriptome count 404
whole transcriptome coverage 400
whole transcriptome mapping 384
whole transcriptome splice finder 413
mer 589
merging (WTA) 589
methylation analysis 589
MethylMiner™ 477
Methyl Map Data 477, 479
output files 481
sample workflow commands 479
mismatches
allowed in the seed 160, 374, 439
color space 505
dicolor 189, 206
mismatch penalty 182, 394
number of mismatches 233, 248, 249
mixed library types, tertiary only 125
module 589

module properties files 557
multi-group tertiary analysis 104

N

naming restrictions 81
NCBI 292, 542, 544, 545, 589
  annotation sources 300
  dbSNP data 299
node 589
normalization 589

O

order requirements for shell commands 119
output files location 61

P

paired-end sequencing 589
pairing 142, 589
  "SV" output filter 236
  distances 590
  FAQ 182
  mate-pair rescue 143
  paired-end tags example 144
  pairing quality 146, 366
  pairing quality value 418
  quality 516
  uniqueness 182, 394
pairing quality value 146
parallelization 590
parameter table 276
PAS files 245, 273
  legacy format, not used 245
  PAS file format 509
PAS.SUM file 245, 255, 262
penalty 590
Phred-scale 366, 418
pipeline 590
ploidy 590
polymorphism 590
polyploidy 590
position error 207
PQV 146, 147, 285, 365, 366, 367, 418
primary alignment 146
primer set 590
probe error 207
processors.per.node 372, 436
project 590

Q

quality
  bamgen.mqv.threshold parameter 148, 369, 432
  mapping quality values 148, 368, 432
  quality alignments 148, 368, 369, 432
  QV preferred over uniqueness 148, 368, 432
quality value 590

R

R3 tag 590
raw reads 590
read 590
  sibling 592
  single 592
read repository index 97, 566
read-set 590
read-set group 590
read-set repository 591
rebuild command 97, 566
refcor 147, 367, 431
  clipping parameter 432, 439
  reference weight 432, 436, 452
refcor parameters 148, 368, 432
Reference 436
reference
  files 591
  genome 591
  one maximum per analysis run 94
  sequence 591
Reference Assisted Base Translation. *See* refcor
reference files 34, 78
  conflicts 540
  fasta 34
  filter reference FASTA 34
  multi-fasta 513
  overview 512
  reference file types 513
  validation 550
reference repository 540
  add new reference files 42
  initial content 41
  location 540
reference SNP identifier 292, 300
refGene.txt 297
refgene2gff.sh 297

reformat_ensembl_gtf.pl 551
RefSeq 591
refSNP identifier 292, 300
regions of interest file 119
repository 591
requirements 38
rescue 591
  distance 591
resequencing 591
  mapping overview 137
  standard workflows 111
resource manager 591
restart an analysis 106
results directory location 105
results files location 61
resume 106
reuse an analysis 106
RNA library, small 592
RNA sequencing 353
RNA-Seq 591
RPKM 591
RPKM metric 407, 417
rsID 292, 300
run completion 486, 561

S

SAET
  error correction 135
  spectrum building 135
  usage 129
SAET parameters
  SAET parameters table 131
  saet.trustfreq 133
SAGE analysis 592
sample 592
sample, sequencing 592
samtools 592
  fillmd command 516
  index command 507
  sort command 507
  view command 498
SASR. *See* splice finder
Satay plot 592
scratch directory 487
secondary analysis 592
secondary processing
  quality values 148, 368, 432
  unique alignments 149, 369, 433
seed 135, 506, 592
  allowed mismatches 160, 374, 439
  anchoring 180, 451
  for an application 180, 450
  junk 133, 588
  low-frequency 133
  picking seed parameters 180, 450
  seed-extend 506
  shorter seeds 180, 451
  start site of 160, 374, 439
  trusted 133
seed-and-extend 592
sense strand 592
sequence
  trimming 592
sequence, reference 591
sequencing
  paired-end 589
sequencing read 590
sequencing sample 592
Serial Analysis of Gene Expression 592
server
  check status 55, 69
  how to start 54
  how to stop 63
set reference command 433, 559
shell commands
  concepts 72
  example 84
  syntax table 89
  table 89
sibling read 592
Single Nucleotide Polymorphisms 592
Single Nucleotide Polymorphisms. *See also* SNPs
single read 592
small indel 256, 592
  algorithm 257
  ALIGN file 254, 255
  allele calling examples 269
  allele calls 265
  ambiguous insertion example 270
  anchor 257
  caller heuristics 262
  clipped coverage 248
  color space 263
  coverage ratio 268
  deletion and insertion ranges 258

detection 239, 256
example commands 240
FAQ 272
gap alignment detection 257, 259
GFF file format 246
hemizygous 248, 268
hemizygous call 268
indel size determination 267
local alignment strategy 260
multiple inserted alleles example 271
one-end anchored 257
output file example 249
output file formats 245
overview 239
pileups 254, 261, 263
TXT file format 252
with targeted resequencing 330
zygosity call 248
small indel parameters 241
indel size filtering 245
indel.max.mismatches 257
indel.min.non-matched.length 257
pairing.indel.max.mismatch.tag1 258
pairing.indel.max.mismatch.tag2 258
resource parameters 245
small indel parameters table 241
small.indel.colorspace.compatibility.level 263
small.indel.consGroup 261
small.indel.detail.level 261
small.indel.genomic.region 272
small.indel.indel.size.distribution.allowed 268
small.indel.max.coverage.ratio 263
small.indel.max.num.evid 261
small.indel.min.map.qv 261
small.indel.min.mapping.quality 261, 263
small.indel.min.non.matched.length 261
small.indel.min.num.evid 261
small.indel.zygosity.profile.name 268
small RNA
FAQ 450
mapping stages of mapping 428
overview 425
small RNA counts 462
gff output file description 466
input files 462
internal parameters table 464
output files 465
parameters table 464
precursor sequence 463
small RNA coverage 457
BAM metadata 400
coverage per chromosome parameter 401
input files 399, 458
output files 401, 459
parameters table 400, 458, 459
RNA.coverage.per.chromosome parameter 459
small RNA library 592
small RNA mapping 425
components 426
input files 427
miRBase2Genome 430
output files 440
parameters table 434
to known miRBase sequences 430
SNP 592
SNP Finding 593
SNPChrPosOnRef 298
SNPContigLoc 298
SNPContigLocusId 298
SNPs
algorithm 206
call stringency settings 211
consensus calls file 201
consensus calls file flags 202
Consensus_Calls.txt output file format 202
FAQ 209
filtered reads 212
filtering reads 197
frequentist algorithm 206
gff3 output file format 199
multiple BAM files as input 188
output file formats 198
with low frequency variant detection 338
with targeted resequencing 330
SNPs parameters 191
call stringency 196
dibayes.call.stringency 192, 196
dibayes.het.skip.high.coverage 197
dibayes.reads.min.mapping.qv 197
het.skip.high.coverage 330, 338
SNPs parameters table 191
stringency and low fequency variant workflows 194
SOLiD® Accuracy Enhancer Tool. *See* SAET 591
spectrum building 593
splice finder
evidence evaluator 414
evidence graph 414

output 416, 421
overview 409
parameters 410
parameters table 410
remaps 411
statistics 416
splice finder parameters 410
splice junction 593
splicing 593
standard workflows 30, 111, 112
example commands 84
strandedness 593
stringency 593
system requirements 38

T

tags 593
F3 586
F5 586
R3 590
targeted resequencing
algorithm 329, 335
analysis.regions.file required 341, 560
barcode support 328, 334
coverage data 348
overview 327, 339
regions of interest file required 341
SAET 330, 331
supported analyses 327, 330
target coverage bedgraph file 345
terminology 71, 72
tertiary analysis
example commands 103
tertiary-only analysis 103
with a groupsfile 103
with grouped BAM files 104
TORQUE 38, 489
trimming 494
troubleshooting 490

U

UCSC Genome Browser 358, 420, 507, 510
unique alignments 149, 369, 433
Unique-PR 419
Unique-SR 419
UNIX commands, mail 561
user accounts 43, 61
user authentication realms 43

V

Variant Call Format (VCF) 513

W

whole transcriptome 353, 403, 409
exon mapping 363
pairing 365
rescue 363
splice finder 409
whole transcriptome analysis 353
whole transcriptome coverage 399
WTA merging 589

X

XSQ converter 520
XSQ file format 494, 495
library details table 495
tag details table 496
XSQ files
5500 instrument data 76
in reads repository 71
input to mapping 29
the add xsq command 92

Z

zygosity 241, 248, 254, 257, 268, 273, 279, 280, 284, 285, 290

Headquarters
5791 Van Allen Way | Carlsbad, CA 92008 USA | Phone +1 760 603 7200 | Toll Free in USA 800 955 6288
For support visit www.appliedbiosystems.com/support

www.lifetechnologies.com

life technologies™