**Setup**

bw$ sinteractive -c12 --mem=24g --gres=1scratch:20
...
node$ module load singularity snakemake hisat
node$ cd /data/$USER
node$ git clone https://github.com/NIH-HPC/snakemake-class.git
node$ cd snakemake-class
node$ ./setup.sh
...

+---------------------------------------------------------------------+
| Class materials have been set up successfully                         |
+---------------------------------------------------------------------+
Building a reproducible workflow with Snakemake and Singularity
Slides adapted from

Johannes Koester

http://slides.com/johanneskoester/snakemake-tutorial-2016#/
automatically execute steps in **parallel**
minimize redundant computation when adding/changing data, or resuming interrupted workflows
document tools, versions, parameters, algorithms execute automatically
There Are Many Workflow Tools

make, ninja, scons, waf, ruffus, jug, Rake, bpipe, BigDataScript, toil, nextflow, paver, bcbio-nextgen, **snakemake**, wdl, cwl, Galaxy, KNIME, Taverna, Partek flow, DNAnexus, SevenBridges, Basespace

https://github.com/pditommaso/awesome-pipeline
Snakemake—a scalable bioinformatics workflow engine

Johannes Köster¹,²,* and Sven Rahmann¹

¹Genome Informatics, Institute of Human Genetics, University of Duisburg-Essen and ²Paediatric Oncology, University Childrens Hospital, 45147 Essen, Germany

Associate Editor: Alfonso Valencia

biowulf users: ~100
Rules

rule qc:
  input:
    "seq/{sample}.fastq.gz"
  output:
    "qc/{sample}.qc"
  script:
    "scripts/myscript.py"

rule aln:
  input:
    "seq/{sample}.fastq.gz"
  output:
    bam = "aln/{sample}.bam",
    bai = "aln/{sample}.bai"
  shell:
    ""
    hisat2 -x /ref/genome -U {input}\n    | samtools sort > {output.bam}
    samtools index {output.bam}
    ""
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        bam = "aln/{sample}.bam",
        bai = "aln/{sample}.bai"
    shell:
        "hisat2 -x /ref/genome -U {input} | samtools sort > {output.bam}"
        "samtools index {output.bam}"

refer to input and output in recipe
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        bam = "aln/{sample}.bam",
        bai = "aln/{sample}.bai"
    shell:
        ""
        hisat2 -x /ref/genome -U {input}\n        | samtools sort > {output.bam}\n        samtools index {output.bam}\n        """
input and output can be **single files** or **lists** referred to by name.
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        "aln/{sample}.bam",
        "aln/{sample}.bai"
    shell:
        ":""hisat2 -x /ref/genome -U {input}\n        | samtools sort > {output[0]}\n        samtools index {output[0]}\n        """

Rules
input and output can be single files or lists
referred to by index
rule aln:
  input:
    "seq/{sample}.fastq.gz"
  output:
    "aln/{sample}.bam",
    "aln/{sample}.bai"
  conda:
    "envs/aln.yml"
  shell:
    ""
    hisat2 -x /ref/genome -U {input}\n    | samtools sort > {output[0]}
    samtools index {output[0]}
    ""
rule aln:
    input:
        "seq/{sample}.fastq.gz"
    output:
        "aln/{sample}.bam",
        "aln/{sample}.bai"
    singularity:
        "shub://NIH-HPC/snakemake-class"
    shell:
      "hisat2 -x /ref/genome -U {input}\
      | samtools sort > {output[0]}\
      samtools index {output[0]}"
      """
Dependencies are implicit and 'backwards'

rule a:
- input: "start/{sample}.txt"
- output: "mid/{sample}.txt"
- shell: "sort {input} > {output}"

rule b:
- input: "mid/{sample}.txt"
- output: "final/{sample}.summary"
- shell: "uniq -c {input} > {output}"
Dependencies are implicit and 'backwards'

rule a:
input: "start/{sample}.txt"
output: "mid/{sample}.txt"
shell: "sort {input} > {output}"

rule b:
input: "mid/{sample}.txt"
output: "final/{sample}.summary"
shell: "uniq -c {input} > {output}"

$ snakemake final/ABC.summary
Dependencies are implicit and 'backwards'

rule a:
input: "start/{sample}.txt"
output: "mid/{sample}.txt"
shell: "sort {input} > {output}"

rule b:
input: "mid/{sample}.txt"
output: "final/{sample}.summary"
shell: "uniq -c {input} > {output}"

$ snakemake final/ABC.summary
Dependencies are implicit and 'backwards'

rule a:
input: "start/{sample}.txt"
output: "mid/{sample}.txt"
shell: "sort {input} > {output}"

rule b:
input: "mid/{sample}.txt"
output: "final/{sample}.summary"
shell: "uniq -c {input} > {output}"

$ snakemake final/ABC.summary
Dependencies are implicit and 'backwards'

rule a:
input: "start/{sample}.txt"
output: "mid/{sample}.txt"
shell: "sort {input} > {output}"

rule b:
input: "mid/{ABC}.txt"
output: "final/{ABC}.summary"
shell: "uniq -c {input} > {output}"

$ snakemake final/ABC.summary
Dependencies are implicit and 'backwards'

rule a:
input: "start/{sample}.txt"
output: "mid/{sample}.txt"
shell: "sort {input} > {output}"

rule b:
input: "mid/ABC.txt"
output: "final/ABC.summary"
shell: "uniq -c {input} > {output}"

$ snakemake final/ABC.summary
Dependencies are implicit and 'backwards'

rule a:
input: "start/{sample}.txt"
output: "mid/ABC.txt"
shell: "sort {input} > {output}"

rule b:
input: "mid/ABC.txt"
output: "final/ABC.summary"
shell: "uniq -c {input} > {output}"

$ snakemake final/ABC.summary
Dependencies are implicit and 'backwards'

rule a:
input: "start/{sample}.txt"
output: "mid/{sample}.txt"
shell: "sort {input} > {output}"

rule b:
input: "mid/{sample}.txt"
output: "final/{sample}.summary"
shell: "uniq -c {input} > {output}"

$ snakemake final/ABC.summary
ABC
ABC
ABC
ABC
reads [fastq] → report [html]

transcriptome [fa] → transcriptome index [salmon index] → salmon [quant.sf] → merged data [rdata]


reads [fastq] → alignment [bam] → alignment qc [rseqc out]

alignment [bam] → rseqc [rseqc out]

fastqc → alignment qc [rseqc out]

hisat2 → alignment [bam]

featureCounts → counts [count.tsv]

R → salmon [quant.sf]

R → salmon [count.tsv]

R → merged data [rdata]