R on Biowulf

This is not an introductory R class. This is for users who are familiar with R on their desktop or other servers.

This class aims to help you to:

- migrate smoothly to use R on Biowulf
- submit batch R jobs to the cluster
- run your R code efficiently

through some case studies.
Case 1

Fry has a R job that needs larger memory than is available on his laptop. He plans to transfer his code and data to Biowulf to run his job on the cluster.

What does he need to finish his job on biowulf?

- Biowulf storage (/data - individual or shared directory)
- Temporary local scratch lscratch
- Select an R version
- R package library management
- R job submission
Storage

Fry should make sure his Biowulf account has enough storage space in /data for inputs and results.

```
$ checkquota

<table>
<thead>
<tr>
<th>Mount</th>
<th>Used</th>
<th>Quota</th>
<th>Percent</th>
<th>Files</th>
<th>Limit</th>
<th>Pe</th>
</tr>
</thead>
<tbody>
<tr>
<td>/data:</td>
<td>256.0 KB</td>
<td>100.0 GB</td>
<td>0.00%</td>
<td>0</td>
<td>31457280</td>
<td></td>
</tr>
<tr>
<td>/home:</td>
<td>24.0 KB</td>
<td>16.0 GB</td>
<td>0.00%</td>
<td>7</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>
```

default:

/home (16GB, fixed size)

/data (100GB, expandable -- request quota increase by filling out the online form)

There are several ways to transfer data
Using lscratch for R

Fry should request temporary local scratch for R interactive sessions or batch jobs on HPC.

He allocates 20G lscratch for interactive session:

```
$ sinteractive --gres=lscratch:20
```

He checks if lscratch is used in R:

```
> Sys.getenv('TMPDIR')
"/lscratch/63315403"  # Good
"/tmp"   # Without lscratch
```

1. R will automatically use lscratch for temporary files if it has been allocated on Biowulf.
2. lscratch > 2GB is recommended.
3. lscratch directories are job-specific and cleaned up when a job is done.
R versions

R is installed as a module on Biowulf, and updated frequently. `module load R` will load the default version (currently R/4.2).

$ module spider "R"
R/3.4, R/3.4.3, R/3.4.4, R/3.5.0, R/3.5.2, R/3.6.0,
R/3.6.1, R/3.6.3, R/4.0.0, R/4.0.3, R/4.0.5, R/4.1.0,
R/4.1.3, R/4.2.0

Each version includes many common R packages.

If Fry needs different version of packages for multiple projects, it's better to use `packrat`: a dependency management system or `renv`.

He checks the R version and loaded packages:

```r
> sessionInfo()
```

1. `sessionInfo()` is very helpful for troubleshooting.
Managing R packages

A large number of R packages are installed with R on Biowulf. Users can also install packages themselves. By default, Fry's own R packages will be installed to `~/R/%v/library`. He has to create this directory himself in bash. He will need to know the R version. For R/4.2*, the code would be:

```
$ mkdir -p ~/R/4.2/library
```

R will search libraries in this order:

```
> .libPaths()
[1] '/home/Fry_username/R/4.2/library'
[2] '/usr/local/apps/R/4.2/site-library_4.2.0'
[3] '/usr/local/apps/R/4.2/4.2.0/lib64/R/library'
```

1. All the local R/4.2* will share the same library location.
2. The library for each version of R only need to be created once.
Moving R packages to /data directory

If Fry wants to change the R library from /home to /data because his /home directory is full, he needs to create his own library:

```r
> dir.create('/data/Fry_username/R/4.2/library', recursive = TRUE)
```

Then he adds the following line to `.bashrc`:

```bash
export R_LIBS_USER=/data/Fry_username/R/%v/library
```

In general, /home is fine, since most packages on CRAN (~87%) are less than 1MB. Although some biological packages might depends on database, which could be surprisingly big.
Reinstalling R packages

To replicate his local environment, Fry first check the version of Bioconductor to make sure it is the same as cluster.

```r
> BiocManager::version()
[1] ‘3.15’
```

Then Fry gets the list of installed packages on his local computer:

```r
> packages <- installed.packages()[,"Package"]
> save(packages, file="Rpackages.rda")
```

1. Two Bioconductor versions are associated with one specific R versions.
2. Attempting to install a version of Bioconductor that is not supported by the version of R in use leads to an error.
3. Using the most recent version of Bioconductor may require installing a new version of R.
Reinstalling R packages

He transfers Rpackages.rda to Biowulf.

Fry then loads the RData file and installs:

```
> load("Rpackages.rda")
> toInstall <- setdiff(packages, installed.packages()[,"Package"])
> BiocManager::install(toInstall)
```

This will download and install the R libraries from Bioconductor and CRAN.

When Fry runs BiocManager, it will always check updates for all packages (including the system packages). Fry should choose "n" since he doesn't have the ability to update the system packages.

```
Update all/some/none? [a/s/n]: n
```

1. Always use BiocManager to install Bioconductor packages
How about other packages?

Install package from GitHub:

```r
> devtools::install_github("trinker/pacman")
```

Biowulf has 1771 R libraries installed. If the package is available on CRAN or Bioconductor, then Fry can email staff@hpc.nih.gov to install it.

`R CMD INSTALL` can be used for installing add-on packages.
Testing a library without modifying the local library

**pacman** is a R package management tool, it conveniently wraps library and package related functions and names them in an intuitive and consistent fashion.

```r
> library(pacman)
> p_temp(rapport)
```

If everything looks good, then he can install with:

```r
> p_install(rapport)
```

Other useful pacman functions: `p_loaded`, `p_unload`, `p_install_gh`, `p_path`, `p_iscran`.
Fry then submits R jobs to the batch scheduler

Rscript (non-interactive variant of R command)

Example: 'rjob.sh'

```
#!/bin/bash
#SBATCH --job-name=R-job    # create a short name for the job
#SBATCH --cpus-per-task=1   # cpu-cores per task (>1 if multi-threaded)
#SBATCH --mem=4G            # memory
#SBATCH --time=04:00:00     # total run time limit (HH:MM:SS)
#SBATCH --gres=lscratch:20   # using local disk
module load R
cd /data/Fry_username/
# Use Rscript
Rscript Rcode.r > Rcode.out
```

He submits the job with `sbatch`:  

```
$ sbatch rjob.sh
```
Questions and Brake

05:00
Case 2

Fry left NIH. His colleague Leela inherits his code, but can't run this code successfully. The error is from Seurat package, so she writes to staff@hpc.nih.gov for help.

What information can Leela provide for troubleshooting?

- Error details and jobID (if it's a batch job)
- R version (sessionInfo())
- Library versions
- Is it a local or system package?
Leela checks the version and location of loaded libraries

```r
> packageVersion('Seurat')
[1] '4.0.1'
> library('Seurat')
> path.package('Seurat')
[1] "/home/Leela_username/R/4.2/library/Seurat"
```

Or use pacman:

```r
> library('pacman')
> p_version(Seurat)
[1] '4.0.1'
> p_path(Seurat)
[1] "/home/Leela_username/R/4.2/library/Seurat"
```

Leela found that her Seurat library is loaded and version 4.0.1.
What should Leela test first?

She switches to the system version instead of her local version:

```r
> p_unload(Seurat)
> library("Seurat", lib.loc="/usr/local/apps/R/4.2/site-library_4.2.0")
```

1) If the error persists, she writes to staff@hpc.nih.gov with the error, a replicable example, and with the output of `sessionInfo()`.

2) If the error is gone, it's time to remove her local packages.
Removing Leela's local installed packages

```r
> remove.packages("test_package")
```

Or with pacman:

```r
> library(pacman)
> p_delete(test_package)
```

Or she deletes the directory with that package:

```bash
$ rm -r ~/R/<ver>/library/test_package
```
Testing R interactively?

Leela plans to plot some figures with the data on Biowulf, she wonders: is there other way (besides command line) to run R interactively?

- jupyter (https://hpc.nih.gov/apps/jupyter.html)

**RStudio (IDE for R)**

- Ideal for graphics.
- Easy to write RMarkdown, Shiny web apps or Quarto.
- Since it needs a graphical connection, please use NoMachine, then start an interactive session. See detailed instructions to start RStudio on Biowulf.

```bash
$ sinteractive --cpus-per-task=2 --mem=6g --gres=lscratch:20
$ module load Rstudio R
$ rstudio &
```
Questions and Brake

05:00
Case 3: Will more CPUs and/or more memory = less running time?

Amy has a deadline coming soon. She wonders if allocating more CPUs and memory will speed up her job?

It depends. Base R is mostly single-threaded. Regardless how many cores are allocated, many R scripts can only use one CPU unless the code was explicitly written to use more than one CPU. As for memory, most of the time the additional resources are not used.

```
$ dashboard_cli jobs --compact
jobid     st  submit_time  partition  n  c  m     timelimit  gres
===================================================================== 
16340213  CD  02T09:19:33  norm   1  8  10GB    8:00:00  lscratch:5 
```

Alternatively, check CPUs and memory usage on graphic dashboard.
When to increase memory?

- Amy's job failed with an error indicating that a process was killed:

/var/spool/slurm/slurmd/job10000101/slurm_script: line 82: 39920 Kill

- Or, if Amy was running Rstudio, her R Session was killed.

- Or the dashboard shows:

<table>
<thead>
<tr>
<th>jobid</th>
<th>state</th>
<th>submit_time</th>
<th>partition</th>
<th>n</th>
<th>c</th>
<th>m</th>
<th>timelimit</th>
<th>gres</th>
</tr>
</thead>
<tbody>
<tr>
<td>18019104</td>
<td>COMPLETED</td>
<td>2021-06-28 19:58:04 EDT</td>
<td>interact+</td>
<td>1</td>
<td>2</td>
<td>2GB</td>
<td>8:00:00</td>
<td>-</td>
</tr>
</tbody>
</table>

- Or the dashboard_cli shows MEM_OVER:

  $ dashboard_cli jobs -u apptest1 --since 6/20 --compact
  jobid   st   submit_time   partition  n  c   m      timelimit  gres
  18019104 CD  28T12:45:27 interact+  1  2  2GB     8:00:00   -
How does Amy know if parallel computing is worth to try?

- Her jobs run for a long time (e.g. 9 days) - Yes
- Her R code is written in Rcpp - No
- One part of the code consumes most of the runtime - Yes

How does she identify the "bottleneck" in the program?

- `system.time()`
- `Rprof()` and `summaryRprof()`
- `Rprof()` runs the profiler for performance of analysis of R code.
- `summaryRprof()` summarizes the output of `Rprof` and gives percent of time spent in each function
How does Amy know if her code can run in parallel?

- levels of parallelization: multiprocessing vs multithreads
- Can her job be split to multiple independent processes?
- Does a function in her code support multiple threads?
- Is there 'lapply/sapply' function?
- Is there 'for' loop?
Running multiple (pleasingly parallel) single-thread R jobs concurrently

Amy can submit \texttt{swarm} jobs if each R process is independent.

**Generate R.swarm:**

\begin{verbatim}
Rscript countA.R
Rscript countB.R
Rscript countC.R
\end{verbatim}

**Submit to Biowulf:**

\begin{verbatim}
$ swarm -f R.swarm --gres=lscratch:20 --module R/4.2
\end{verbatim}
Common pitfall: Failed R jobs with "all connections are in use" error

Amy's swarm jobs were submitted to the cluster, but some of them failed.

She wonders why?

- Some of the Biowulf nodes have 128 CPUs (phase6 nodes) or 144 CPUs (largemem nodes), check with `freen`

- Some R packages will detect all cores on a node even if they are not allocated (e.g. `parallel::detectCores()`)

- If not explicitly specifying the number of CPUs R may try to use all CPUs but R can't communicate with more than 128 parallel worker processes

- Solution: use `parallelly::availableCores()` to detect allocated CPUs and use fewer CPUs
Can Amy's code use multiple threads?

R on biowulf uses the Intel MKL math libraries which can use multiple threads for some functions. Amy can find out if her script could speed up by allocating more CPUs and comparing the runtime between single-threaded and multi-threaded:

```bash
$ sbatch --cpus-per-task=4
```

And she uses the `OMP_NUM_THREADS` variable in her script:

```bash
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
Rscript script.R
```

She checks the runtime and CPU utilization on the dashboard and found out she was using 32 CPUs (overloading) instead of 4 CPUs.
Overloading could dramatically slow down your job

So she change the OMP_NUM_THREADS to 1 (which is also the default setting of R module on Biowulf):

```bash
export OMP_NUM_THREADS=1
Rscript script.R
```

She checks the runtime and CPU utilization on the dashboard and found out she was using only 1 CPU instead of 4 CPUs.
Can Amy modify the code to use more CPUs?

Amy replaces 'lapply' with 'mclapply' and setup 'mc.cores' to use multiple worker processes to parallelize work.

Example with 12 CPUs:

```r
> library(parallel)
> ncpus <- parallelly::availableCores()
> options(mc.cores = ncpus) # set a global option for parallel package
# Then run mclapply()
> mclapply(X, FUN, ..., mc.cores = ncpus)
```

1. Similarly, 'sapply' could be replaced by 'mcsapply'.

Performance comparison between `lapply()` and `mclapply()`

```r
> N <- 10^6
> system.time(x<-lapply(1:N, function(i) {rnorm(300)}))
## user  system elapsed
## 36.588   1.375  38.053
> system.time(x<-mclapply(1:N, function(i) {rnorm(300)},mc.cores = no)
## user  system elapsed
## 11.587  14.547  13.684
```

12 workers, 3x speedup, efficiency = $\frac{3}{12} = 25\%$.

Parallel jobs should aim for an efficiency of 70-80%.
More CPUs do not always run code efficiently

- e.g. running 20 loops with 40 CPUs.
- Benchmark tests are highly recommended for parallel jobs.
- The performance of mclappy() with 2-32 CPUs and compared their efficiency:

Run this R code no more than 6 CPUs for efficiency

1. mclapply/mcsapply can use multiple cores on one node, but not on multiple nodes.
Questions and Brake

05:00
Contact the HPC team with questions

- Always check the up-to-date R doc.
- Email: staff@hpc.nih.gov
- Join our monthly walk-in virtual consultation.
What steps need to be modified to Bender's code with multiple CPUs:

```r
library(doParallel)
library(doMC)
registerDoMC(cores=parallel::detectCores())
max.eig <- function(N) {
    d <- matrix(rnorm(N*N), nrow = N)
    E <- eigen(d)$values
    abs(E)[[1]]
}
for (n in 400:520) max.eig(n)
```

batch job:

```bash
#!/bin/bash
module load R
R --vanilla < test.R > test.out
```

submit with:

```
$ sbatch test.sh --ntask=6 --mem=4
```

One solution for case 4:

Code:

```r
library(doParallel)
library(doMC)
registerDoMC(cores=parallelly::availableCores())
max.eig <- function(N) {
  d <- matrix(rnorm(N*N), nrow = N)
  E <- eigen(d)$values
  abs(E)[[1]]
}
foreach(n = 400:520) %dopar% max.eig(n)
```

batch job:

```bash
#!/bin/bash
module load R
Rscript test.R >test.out
```

submit with:

```
$ sbatch --gres=lscratch:20 --mem=4g --cpus-per-task=6 test.sh
```
Working through a case study 5:

- Rscript run_sim_1.R sim_1.out
- Rscript run_sim_2.R sim_2.out
- ...

How would Bender modify the code to be able to run on the cluster:

(hint: check our R doc):

- Rscript run_sim.R 1 sim_1.out
- Rscript run_sim.R 2 sim_2.out
- ...

01:30