Using R at CHPC

Brett Milash

Center for High Performance Computing

07/18/2024

Today’s agenda:

- Choosing the best way to connect to CHPC for <u>your</u> R application
- Installing R packages
- Using R in parallel

Methods of using R at CHPC

Command	Attributes
R	Interactive, command line style
RStudio	Interactive, either web browser or X-Windows GUI
Jupyter Lab or Notebook	Interactive, document based, web browser GUI
Rscript	Non-interactive, batch script oriented
R Markdown	Non-interactive, document based

Methods to access resources at CHPC

Method	Attributes	Resource
ssh to interactive node	command line or GUI	interactive node
FastX to interactive node	command line or GUI, persistence	interactive node
OnDemand cluster shell access	convenient, like ssh, command line only	interactive node
SLURM sbatch command	non-interactive (batch mode)	compute node(s)
SLURM salloc command	interactive command-line or GUI	compute node(s)
OnDemand system installed applications	web-based access	compute node(s)

Remember the appropriate uses for interactive and compute nodes:

- Interactive nodes: writing code, installing code, small-scale testing, debugging, managing SLURM jobs
- Compute nodes: heavy-duty computing (simulations, stats, data visualization) whether interactive or not

R use methods vs. CHPC access methods

	R	RStudio	Jupyter	RScript	R Markdown
ssh (to interactive node)					
FastX (to interactive node)					
OnDemand cluster shell access (runs on interactive node)					
SLURM sbatch (compute node)					
SLURM salloc (compute node)					
OnDemand system installed applications (runs on compute nodes)					

R use methods vs. CHPC access methods

	R	RStudio	Jupyter	RScript	R Markdown
ssh (to interactive node)	✅ good for testing	✅ but slow	Inefficient - not recommended	✅ good for testing	✅ good for testing
FastX (to interactive node)	✅ good for testing	✅	Inefficient - not recommended	✅ good for testing	✅ good for testing
OnDemand cluster shell access (on interactive node)	✅ but no graphics	❌ - requires X windows	❌ - requires X windows	✅ good for testing	✅ good for testing
SLURM sbatch (compute node)	❌	❌	❌	⭐⭐⭐⭐⭐	⭐⭐⭐
SLURM salloc (compute node)	✅	✅	Inefficient - not recommended	✅	✅
OnDemand system installed applications (on compute nodes)	❌	⭐⭐⭐⭐⭐	⭐⭐⭐⭐⭐	❌	❌

“✅ good for testing” means software works well within computing limits of interactive node

ssh, FastX, and OnDemand shell access

- These methods provide a terminal window on an interactive node
- Graphics (whether GUI or graphical output) requires X-forwarding
  - On Mac use “ssh -Y *username@hostname”*
  - On Windows use Xming (https://xming.en.softonic.com/)
  - X-forwarding can be slow without some help
- FastX accelerates X-forwarding
  - Web interface and desktop clients are available
  - Graphics performance much improved over ssh X-forwarding
  - Keyboard copy and paste can be a problem

SLURM sbatch and salloc demo

- Both methods provide access to compute nodes
- sbatch is batch oriented - therefore non-interactive
  - Submits a shell script to cluster for non-interactive execution
- salloc starts an interactive shell session on a compute node
  - Session started by salloc inherits your environment from interactive node

OnDemand demo

- Web portal
- Access to compute nodes (and interactive nodes)
- Great for running web and GUI applications on compute nodes
- Access to compute nodes via Slurm system

Writing R code for interactive vs batch jobs

- Some tasks are inherently interactive
  - Coding
  - Debugging
  - Data visualization
- Some tasks are inherently batch oriented
  - Large or long-running simulations
  - Processing lots of data files
- To write R code that adapts to both use cases:
  - Write lots of functions (potentially in a separate source code file)
  - Use `interactive()` to test whether job is interactive or batch

solaR: Radiation and Photovoltaic Systems
Calculation methods of solar radiation and performance of photovoltaic systems from daily and intradaily irradiation data sources.

https://cran.r-project.org/web/packages/solaR/index.html

```
$ module load R
$ R

R version 4.4.0 (2024-04-24) -- "Puppy Cup"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

…

> library(solaR)
Error in library(solaR) : there is no package called ‘solaR’
```

Strategies for handling missing R libraries

1. Tell R where to find the already-installed package
2. Find a version of R that has the package installed
3. Install the package yourself

Strategy 1: Tell R where to find the already-installed package

- When loading a library R searches the path returned by .libPaths()

```
> .libPaths()
[1] "/uufs/chpc.utah.edu/sys/installdir/r8/RLibs/4.4.0"
[2] "/uufs/chpc.utah.edu/sys/installdir/r8/R/4.4.0/lib64/R/library"
```

- One can append additional *existing* directories to this path:

```
> .libPaths( c( .libPaths(), "~u0253283/R/x86_64-pc-linux-gnu-library/4.4" ) )
> .libPaths()
[1] "/uufs/chpc.utah.edu/sys/installdir/r8/RLibs/4.4.0"
[2] "/uufs/chpc.utah.edu/sys/installdir/r8/R/4.4.0/lib64/R/library"
[3] "/uufs/chpc.utah.edu/common/home/u0253283/R/x86_64-pc-linux-gnu-library/4.4"
```

- However - this doesn’t persist from session to session

Strategy 1 (continued)

- Or use the R_LIBS_USER environment variable:

```
# In bash:
$ export R_LIBS_USER=~u0253283/R/x86_64-pc-linux-gnu-library/4.4
# Or in tcsh:
$ setenv R_LIBS_USER ~u0253283/R/x86_64-pc-linux-gnu-library/4.4
$ R

R version 4.4.0 (2024-04-24) -- "Puppy Cup"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

> .libPaths()
[1] "/uufs/chpc.utah.edu/common/home/u0253283/R/x86_64-pc-linux-gnu-library/4.4"
[2] "/uufs/chpc.utah.edu/sys/installdir/r8/RLibs/4.4.0"
[3] "/uufs/chpc.utah.edu/sys/installdir/r8/R/4.4.0/lib64/R/library"
```

- This strategy works great for research labs with group space (i.e. a shared file system) that want a shared R library collection

Strategy 2: Find a version of R that has the package installed

> "The best R package installation is the one you don't have to do."
> -- Brett Milash

Partial list of R modules at CHPC:

- `R/4.2.1-bioconductor`
- `R/4.2.2`
- `R/4.3.2-basic`
- `R/4.3.2-bioconductor`
- `R/4.3.2-geospatial`
- `R/4.3.2`
- `R/4.4.0-basic`
- `R/4.4.0-bioconductor`
- `R/4.4.0-geospatial`
- `R/4.4.0`

CHPC’s R modules

- Every CHPC R module has useful libraries in addition to the base package:

```
car
caret
data.table
devtools
DiagrammeR
dplyr
DT
dygraphs
foreign
gcbd
ggmap
ggplot2
ggvis
glmnet
googleVis
htmlwidgets
httr
jsonlite
knitr
leaflet
lme4
locfit
lubridate
maps
maptools
mgcv
microbenchmark
multcomp
network3D
nlme
parallel
pryr
qcc
quantmod
randomForest
Rcpp
reshape2
rgl
rmarkdown
RMySQL
RODBC
roxygen2
RPostgresSQL
RSQLite
shiny
sp
stringr
survival
testthat
threeJS
tidyr
vcd
XLConnect
xlsx
XML
xtable
xts
zoo
```

- Certain CHPC R modules have additional library collections:
    - Bioconductor
    - Geospatial packages (e.g. proj4, rgdal, RNetCDF, hdf5r)

Strategy 3: Install the package yourself

Main repositories of R code:

- CRAN: Comprehensive R Archive Network
  - https://cran.r-project.org
  - https://cran.r-project.org/web/packages/available_packages_by_name.html
- Bioconductor: Open-source software for bioinformatics
  - https://www.bioconductor.org
  - https://bioconductor.org/packages/release/bioc/
- Github
  - https://github.com
  - https://github.com/qinwf/awesome-R

Installation method varies depending on repository

Installing packages from CRAN

Use the install.packages() function, with package name in quotes:

```
$ module load R/4.2.2
$ R

R version 4.2.2 (2022-10-31) -- "Innocent and Trusting”
…
> .libPaths()
[1] "/uufs/chpc.utah.edu/sys/installdir/r8/R/4.2.2/lib64/R/library"
[2] "/uufs/chpc.utah.edu/sys/installdir/r8/RLibs/4.2.2”
> install.packages("solaR")
Installing package into
‘/uufs/chpc.utah.edu/sys/installdir/r8/R/4.2.2/lib64/R/library’
(as ‘lib’ is unspecified)
Warning in install.packages("solaR") :
  'lib = "/uufs/chpc.utah.edu/sys/installdir/r8/R/4.2.2/lib64/R/library"' is not
writable
Would you like to use a personal library instead? (yes/No/cancel) yes
```

Installing packages from CRAN (continued)

```
Would you like to create a personal library
‘/uufs/chpc.utah.edu/common/home/u0424091/R/x86_64-pc-linux-gnu-library/4.2’
to install packages into? (yes/No/cancel) yes
--- Please select a CRAN mirror for use in this session ---
(A list of CRAN mirror sites pops up – I selected “USA (IA)(https)”)
also installing the dependencies ‘interp’, ‘latticeExtra’

trying URL 'https://mirror.las.iastate.edu/CRAN/src/contrib/interp_1.1-4.tar.gz’
…
* DONE (solaR)

The downloaded source packages are in
‘/tmp/RtmpdZF0ge/downloaded_packages’
>
```

Installing packages from CRAN (continued)

```
> library(solaR)
Loading required package: zoo

Attaching package: ‘zoo’

The following objects are masked from ‘package:base’:

    as.Date, as.Date.numeric

Loading required package: lattice
Loading required package: latticeExtra
Time Zone set to UTC.
> find.package("solaR")
[1] "/uufs/chpc.utah.edu/common/home/u0424091/R/x86_64-pc-linux-gnu-
library/4.2/solaR"
> .libPaths()
[1] "/uufs/chpc.utah.edu/common/home/u0424091/R/x86_64-pc-linux-gnu-library/4.2"
[2] "/uufs/chpc.utah.edu/sys/installdir/r8/R/4.2.2/lib64/R/library"
[3] "/uufs/chpc.utah.edu/sys/installdir/r8/RLibs/4.2.2"
```

Installing packages from Github

DataExplorer: https://github.com/boxuancui/DataExplorer

Use the devtools::install_github() function, with package name in quotes.

Note that “lib=“ is specified! Without that the installation of dependencies will fail.

```
$ module load R/4.2.2
$ R

R version 4.4.0 (2024-04-24) -- "Puppy Cup”
…
> devtools::install_github("boxuancui/DataExplorer",
lib=c("/uufs/chpc.utah.edu/common/home/u0424091/R/x86_64-pc-linux-gnu-
library/4.4"))
Downloading GitHub repo boxuancui/DataExplorer@HEAD
…
** testing if installed package keeps a record of temporary installation path
* DONE (DataExplorer)
> library(DataExplorer)
```

Installing packages from Bioconductor

```
> install.packages("BiocManager")
Installing package into
‘/uufs/chpc.utah.edu/sys/installdir/r8/R/4.2.2/lib64/R/library’
(as ‘lib’ is unspecified)
Warning in install.packages("BiocManager") :
  'lib = "/uufs/chpc.utah.edu/sys/installdir/r8/R/4.2.2/lib64/R/library"' is not
writable
Would you like to use a personal library instead? (yes/No/cancel) yes

…

** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (BiocManager)


The downloaded source packages are in
‘/tmp/RtmpzCb3wb/downloaded_packages’
>
```

Installing packages from Bioconductor (continued)

Like devtools::install_github(), it is safest to specify the ”lib=“ argument.

```
> BiocManager::install("PFAM.db",
lib=c("/uufs/chpc.utah.edu/common/home/u0424091/R/x86_64-pc-linux-gnu-
library/4.4"))

…
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (PFAM.db)


The downloaded source packages are in
‘/tmp/RtmpzCb3wb/downloaded_packages’
>
```

Installing packages from source code

- install.packages, BiocManager::install, and devtools::install_github download the package source code, then compile and install it
- If you have the source code URL you can compile and install it like this:

```
$ wget https://bioconductor.org/packages/3.16/bioc/src/contrib/limma_3.54.2.tar.gz
$ R CMD INSTALL --library=$HOME/R/x86_64-pc-linux-gnu-library/4.2 limma_3.54.2.tar.gz
```

- This method doesn’t handle dependencies however

anyLib: Install and Load Any Package from CRAN, Bioconductor or Github

- “Made to make your life simpler with packages, by installing and loading a list of packages, whether they are on CRAN, Bioconductor or github. For github, if you do not have the full path, with the maintainer name in it (e.g. "achateigner/topReviGO"), it will be able to load it but not to install it.”

- https://cran.r-project.org/web/packages/anyLib/index.html

Running parallel R code at CHPC

- On an interactive node - should not be using multiple cores
- A SLURM job may not have access to all the cores on a node
- To count all the cores on the machine (which is not what we want):

```
# How many cores are on this machine?
> library(parallel)
> detectCores()
[1] 64
```

- To count the cores available to your job:

```
# How many cores are available to me on this node?
> strtoi(Sys.getenv("SLURM_TASKS_PER_NODE"))
[1] 10
# How many cores are available to my potentially multi-node job:
> strtoi(Sys.getenv("SLURM_NTASKS"))
[1] 10
```

CHPC’s R documentation

- https://git.io/CHPC-Intro-to-Parallel-Computing-R
  - Excellent examples of parallel R code, both single- and multi-node
- https://www.chpc.utah.edu/documentation/software/r-language.php
  - A few things are out of date:
    - Setting up a personal library: as we saw, R version 4.X handles (most) of this automatically
    - We no longer use Intel’s compiler icc for R, we use gcc exclusively
    - Older versions of R used the file $HOME/.R/Makevars to specify compile-time options, and this file is largely obsolete (thank goodness!)