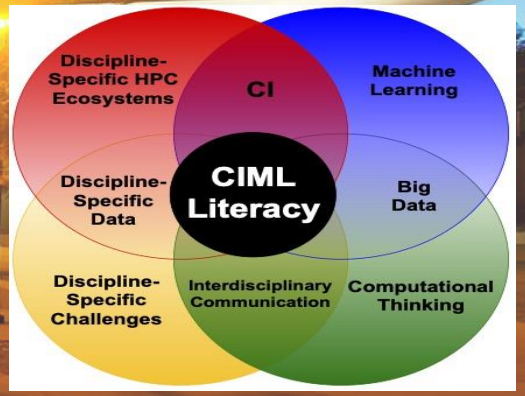
Scalable Machine Learning: R (and other packages), HPC, and Scaling

Paul Rodriguez, PhD

08/09/2023

f



What is Machine Learning?

We often say something like:

Programs that learn from data (as opposed to being given rules)

Statistical Learning (as opposed to Statistical Inferencing)

Or, we often just use certain terms:

Supervised/Unsupervised Learning;

Classification/Regression tasks;

Overfitting/Regularization; etc..

Or, we often talk about algorithm functions like:

fit(); predict(); evaluate(); etc…

The HPC & Machine Learning landscape

Why use HPC?

Big data and/or Big computation

Possibilities for parallelization; eg distributing data and/or computation

For ML models parallelization often depends on the algorithm to fit parameters

Schedule overview

Mai Nguyen, Paul Rodriguez

- **Today - Scaling**
  - R on HPC
  - Spark

- **Tomorrow – Deep Learning**
  - Intro to NN/CNN/Deep Learning
  - Intro to Multinode execution
  - DL Layers and Models
  - DL Transfer Learning
  - DL Functional API, Special Connections, Transformers

Outline

- R and Scaling
- Parallel R
- Embarrassingly Parallel R
- Going big with R?

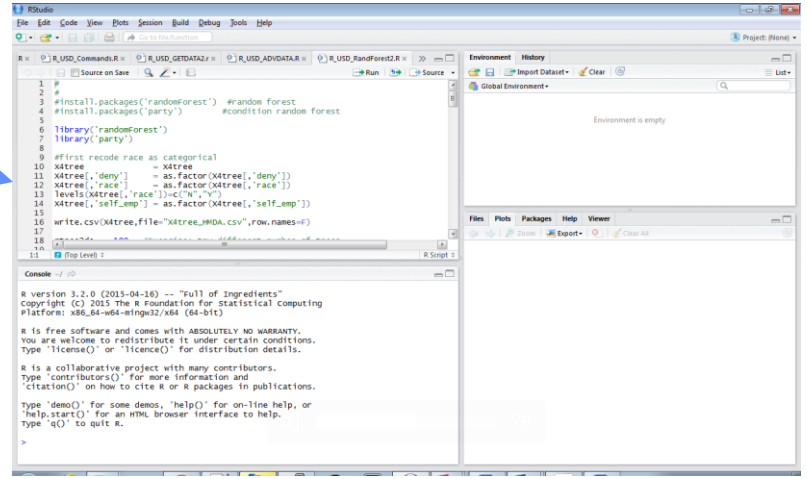
A typical R development workflow

- **R studio: An Integrated development environment for R**

Menu tab

Edit window to Build scripts

R console



Environment Information on variables and command history

Plots, help docs, package lists

Typical R code workflow

```
#READ DATA
X                    =read.csv('hmda_aer.csv',header=T,stringsAsFactors=T)

#SUBSET DATA
indices_2keep =which(X[,'s13'] %in% c(3,4,5)))
X                    =X[unique(indices_2keep),]

#CREATE/TRANSFORM VARIABLES
pi_rat               = as.numeric(X[,'s46']/100)              #debt2income ratio


#RUN MODEL and SHOW RESULTS
lm_result            =lm(deny~pi_rat)              #lm is 'linearmodel'
summary(lm_result)
```

R strengths for HPC (IMHO)

- **Data Wrangling –**

- **Particular statistical procedure implementations -**
  Imputation methods (for missing data)
  Sampling methods
  Instrument Variable (2 stage) Regression
  Matching subjects for pairwise analysis
  MCMC routines (but Stan is likely better package)
  Generalized Linear Model
  Some ML model (e.g. randomForest)

R Scaling In a nutshell

- R uses BLAS/LAPACK math libraries for operations on vectors

  [Same for Matlab and Python]

- R packages provide multicore, out-of-core, multinode, or distributed data (SparkR) options

  [Same for Matlab and Python]

- Some ML model implementations may be built to use parallel backends (review the available options)

Consider Regression Computations

- **Linear Model:** $Y = X * B$

  where Y=outcomes , X=data matrix

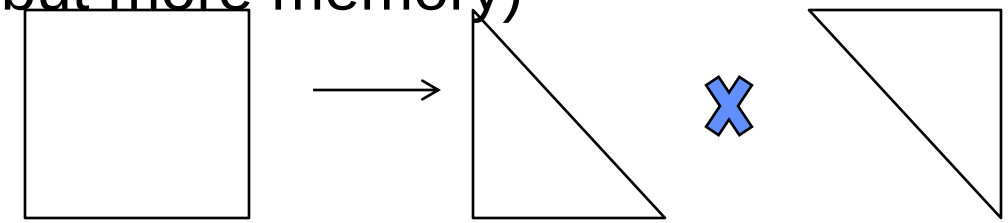
- **Algebraically, we could**:

  take "inverse" $of\ X * Y = B$ (time consuming)

  use derivatives to search for solutions (very general)

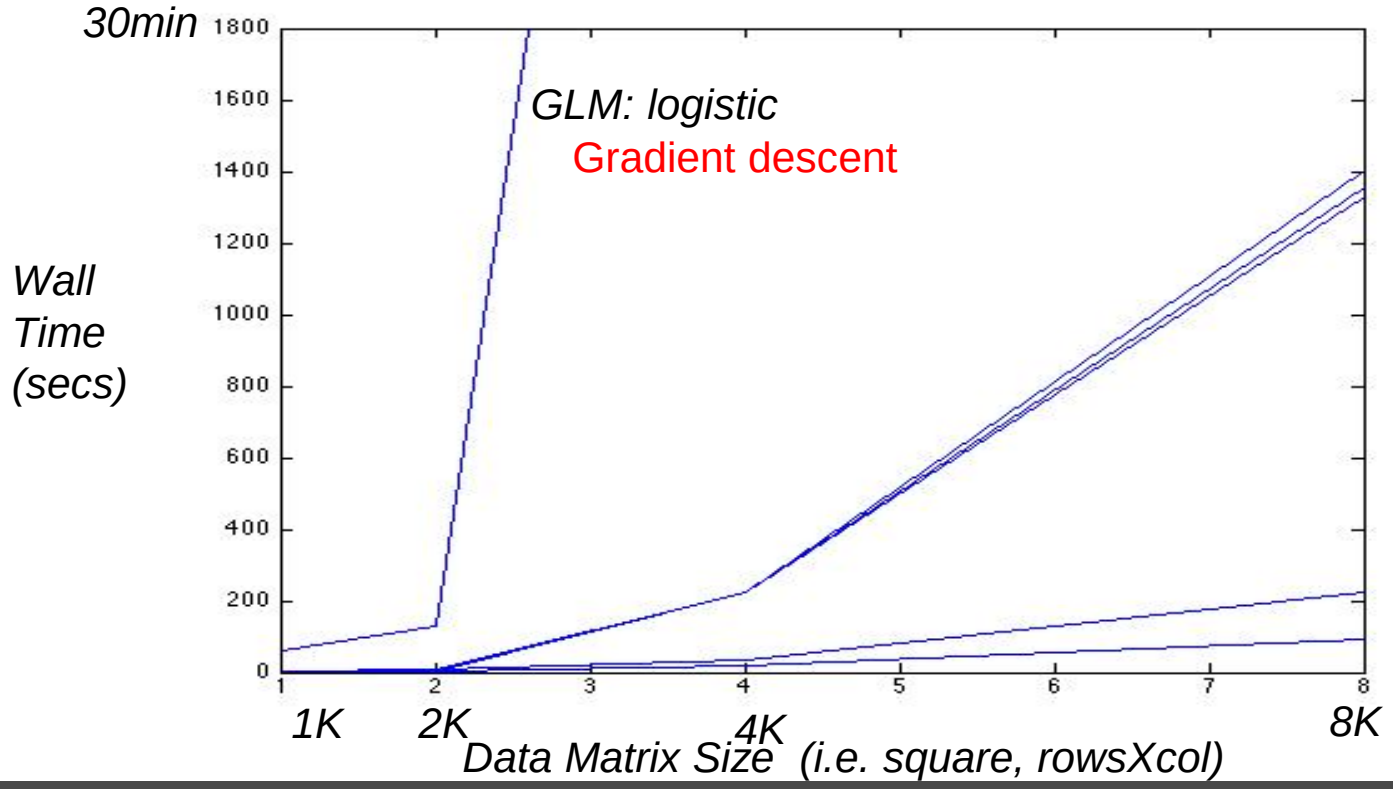
- **Or, better:**

  QR decomposition of *X* into triangular matrices (easier to solve but more memory)



Solving Linear Systems
Performance with R, 1 compute node

R:
glm(Y~X,family=gaussian) #gaussn regrssn (like lm)
glm(Y~X,family=binomial) # logistic regrssn (Y=0 or 1)



Matrix inverse solutions

glm: Gaussian
lm()
Inverse()

QR solutions

Solve(a,b)
QR

Outline

- R and Scaling
- Parallel R
- Embarrassingly Parallel R
- A big data exploration of R

R multicore processing

- ‘doParallel’ package – provides the back end to the ‘for each’ parallel processing command
- uses threads across cpu cores to pass data & commands
- It also works for multinode (runs on top of RMPI)

See https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)
```

1. allocate workers

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(
```

1. allocate workers

2. Make ‘foreach’ loop

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(i=1:24,.combine=rbind)
```

1. allocate workers

2. Make ‘foreach’ loop

3. specify how to combine results

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(i=1:24,.combine=rbind) %dopar%
      {   …
```

1. allocate workers

2. Make ‘foreach’ loop

3. specify how to combine results

4. %dopar% runs it across cores, (%do% runs it serially)

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(i=1:24,.combine=rbind) %dopar%
      {   …
            your code here

          return(  a variable or object )
    })
```

1. allocate workers

2. Make ‘foreach’ loop

3. specify how to combine results

4. %dopar% runs it across cores, (%do% runs it serially)

R multicore coding

```
install.packages(doParallel)
library(doParallel)
registerDoParallel(cores=24)

    my_data_frame = …..

    my_results = foreach(i=1:24,.combine=rbind) %dopar%
      {  …
            your code here

          return(  a variable or object )
    })
```

1. allocate workers

2. Make 'foreach' loop

3. specify how to combine results

4. %dopar% runs it across cores, (%do% runs it serially)

BEWARE: foreach will copy data to every core if its seems necessary

R multinode: parallel backend

```
library(doParallel)

cl <- makeCluster(48)
registerDoParallel(cl)
```

1. allocate cluster as parallel backend

R multinode: parallel backend

```
library(doParallel)

cl <- makeCluster(48)
registerDoParallel(cl)

my_data_frame = …..

results = foreach(i=1:48,.combine=rbind) %dopar%
  {   … your code here


        return(  a variable or object )
})
stopCluster(cl)
```

1. allocate cluster as parallel backend

2. %dopar% puts loops across cores and nodes

R multinode: parallel backend

```
library(doParallel)

cl <- makeCluster(48)
registerDoParallel(cl)

my_data_frame = …..

results = foreach(i=1:48,.combine=rbind) %dopar%
  {   … your code here


      return(  a variable or object )
})
stopCluster(cl)
```

1. allocate cluster as parallel backend

2. %dopar% puts loops across cores and nodes

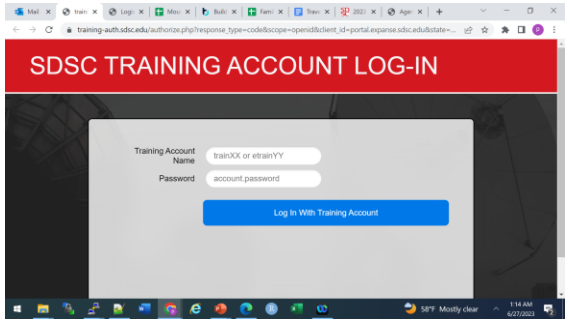
BEWARE: foreach will copy data to every core in every node if its seems necessary

Testing/Evaluating R parallel

- **Exercise: 'TestdoParallel' R script**

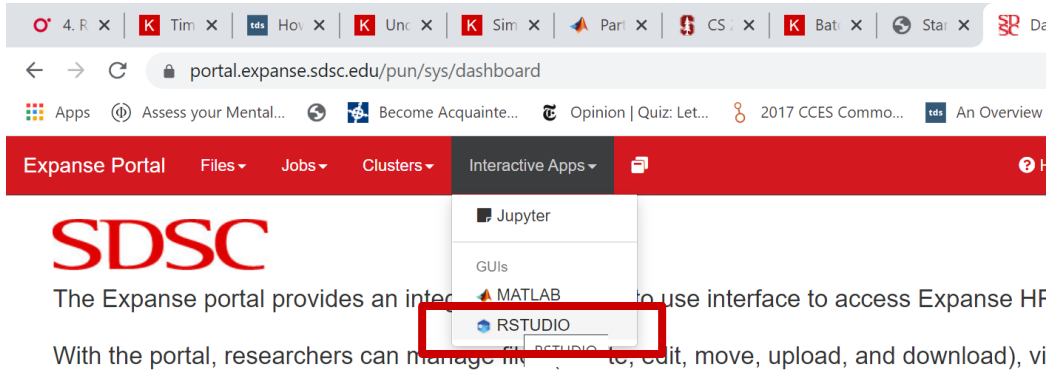
1. **Log into expanse portal and start R studio**

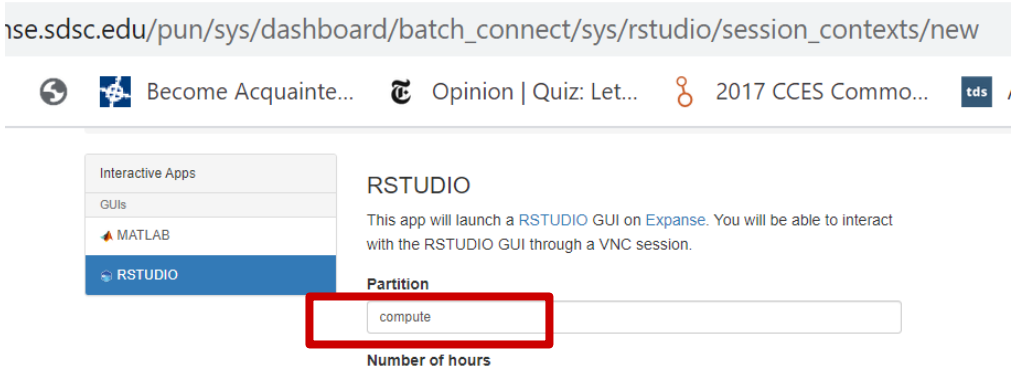
   goto URL: *https://portal.expanse.sdsc.edu/training*

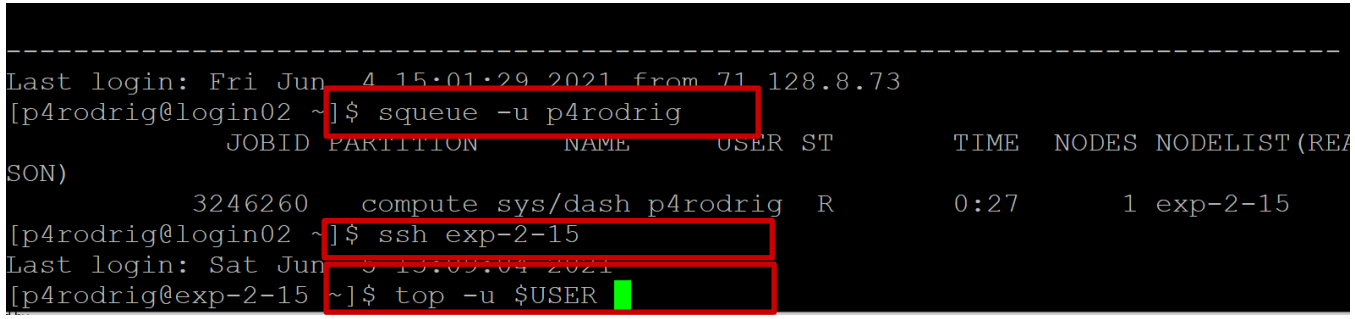


2. **Also log into expanse command line and ssh to compute node**

3. **run 'top –u username' to see performance**
   - look for tradeoffs in memory vs execution as matrix size varies (see next slides)







1 Open portal ->
Interactive Apps -> Rstudio

Enter
Node: "compute"
Account: gue998
Cores: "64"
Memory: 124 Gb
(other fields defaults ok)

2 Also login to Expanse terminal window and Enter

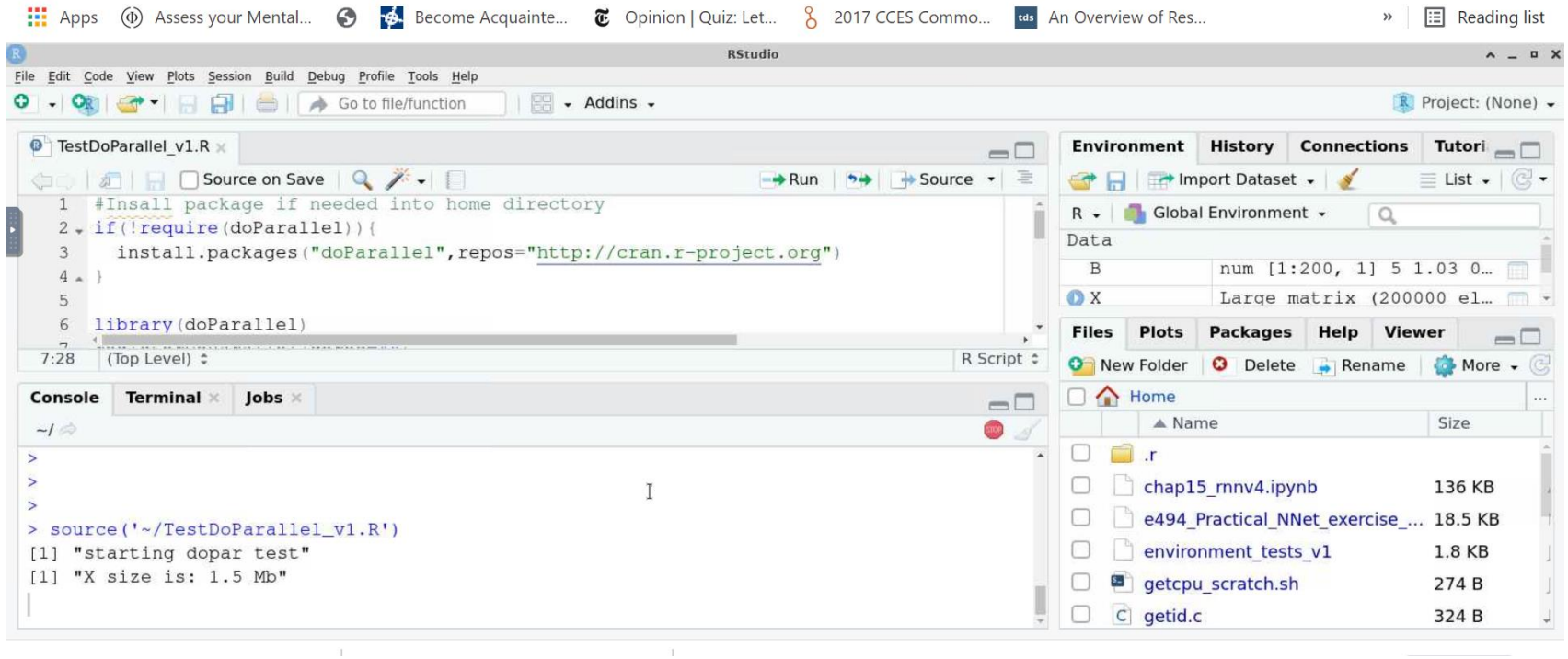
$ squeue –u $USER
$ ssh exp-##-##
$ top –u $USER

'H' will toggle threads
'f', downarrow to P, space, esc.

3 Open the 'Test_doParallel ' Rscript

Select 'source' to run the whole script, it will install 'doParallel' package (if the R installation doesn't have it already)

look for # <<< ----- comments to change data parameters

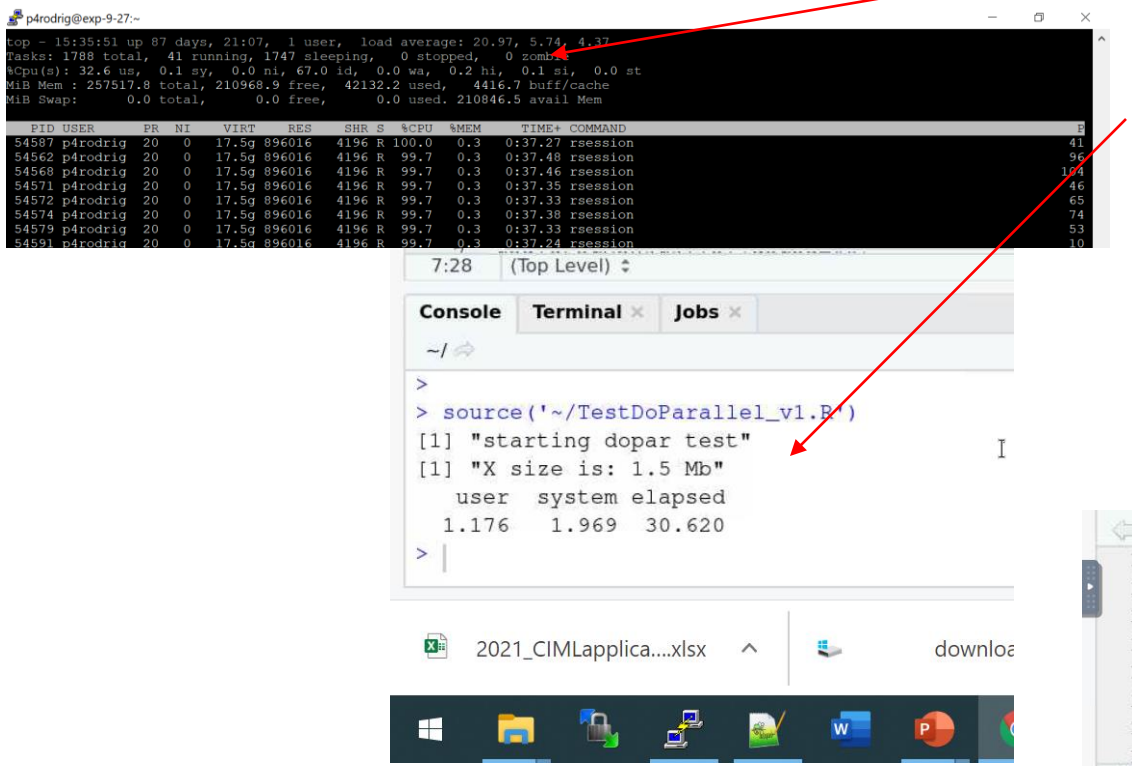


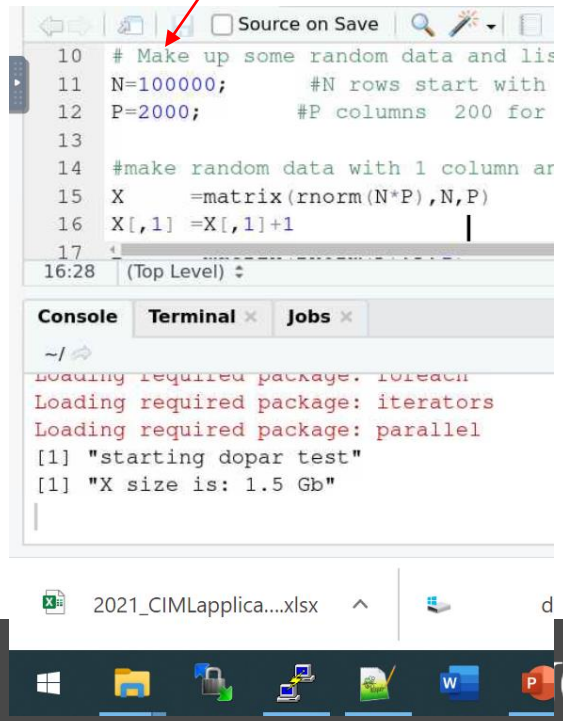
Review the top output

Notice the elapsed time and memory size

Change the NxP matrix size and rerun

(start with N=10K, P=2K)





Try this at home:

Let N=100K , P=2000
Notice the memory used is close to 124Gb we asked for

p4rodrig@exp-9-27:~

```
top - 15:38:40 up 87 days, 21:10,  1 user,  load average: 10.77, 6.29, 4.76
Tasks: 1749 total,  19 running, 1730 sleeping,   0 stopped,   0 zombie
%Cpu(s): 14.0 us,  0.0 sy,  0.0 ni, 85.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 257517.8 total, 130239.0 free, 123199.7 used,   4079.0 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 129947.3 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND                                                P
 55219 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.52 rsession                                              68
 55227 p4rodrig  20   0   24.2g   7.6g   3064 R 100.0   3.0   0:24.55 rsession                                              88
 55235 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.56 rsession                                              80
 55236 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.70 rsession                                             100
 55237 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.50 rsession                                              47
 55242 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.36 rsession                                              32
 55253 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.69 rsession                                             126
 55259 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.00 rsession                                              16
 55261 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:24.25 rsession                                              24
 55265 p4rodrig  20   0   24.2g   7.6g   2696 R 100.0   3.0   0:23.96 rsession                                               6
 55239 p4rodrig  20   0   24.2g   7.6g   2696 R  99.7   3.0   0:24.61 rsession                                              20
 55241 p4rodrig  20   0   24.2g   7.6g   2696 R  99.7   3.0   0:24.43 rsession                                               8
 55243 p4rodrig  20   0   24.2g   7.6g   2836 R  99.7   3.0   0:24.53 rsession                                             104
```

If you ask for 248Gb will it run?
What if you use only 24 cores?

Parallezing for loops

(pseudo code)

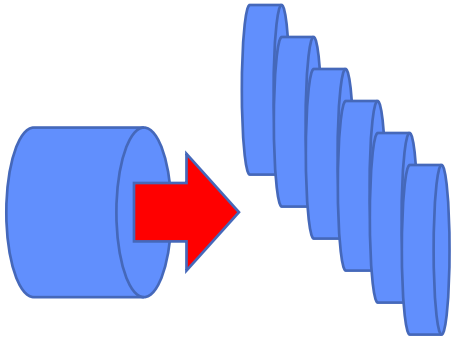
R with doParallel	Matlab with parallel toolbox	Python with dask.distributed
makecluster *registercluster*	*parcluster('local')* *parpool()*	Import delayed, Client Client(numwkrs)
foreach with dopar,	*parfor* *or*	for i in range(numwkrs): A=delayed(my_func)(i)
combine results	*'spmd' with distributed arrays*	Acombine.append(A)
	gather array	Acombined.compute()

Outline

- R and Scaling
- Parallel R
- Embarrassingly Parallel R
- A big data exploration of R

An option for (embarrassingly) Parallel R

1. Split data into N parts

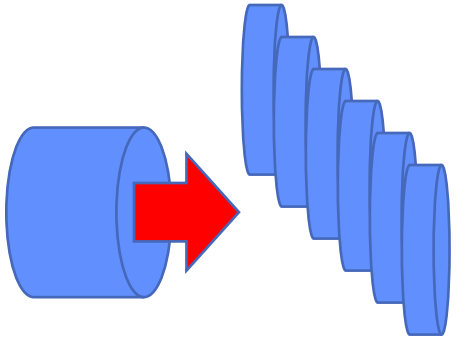


An option for (embarrassingly) Parallel R

1. Split data into N parts

2. In slurm batch script:

mpirun … my-perl-script



My-perl-script:
get cpu-id & pass it to R

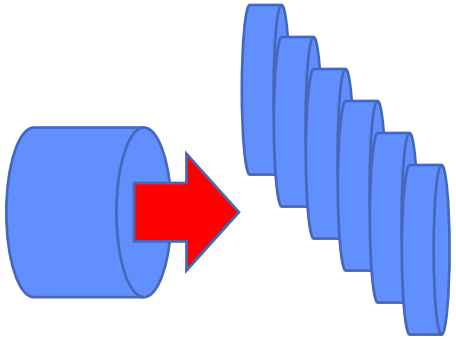
R script:
process
n-th dataset

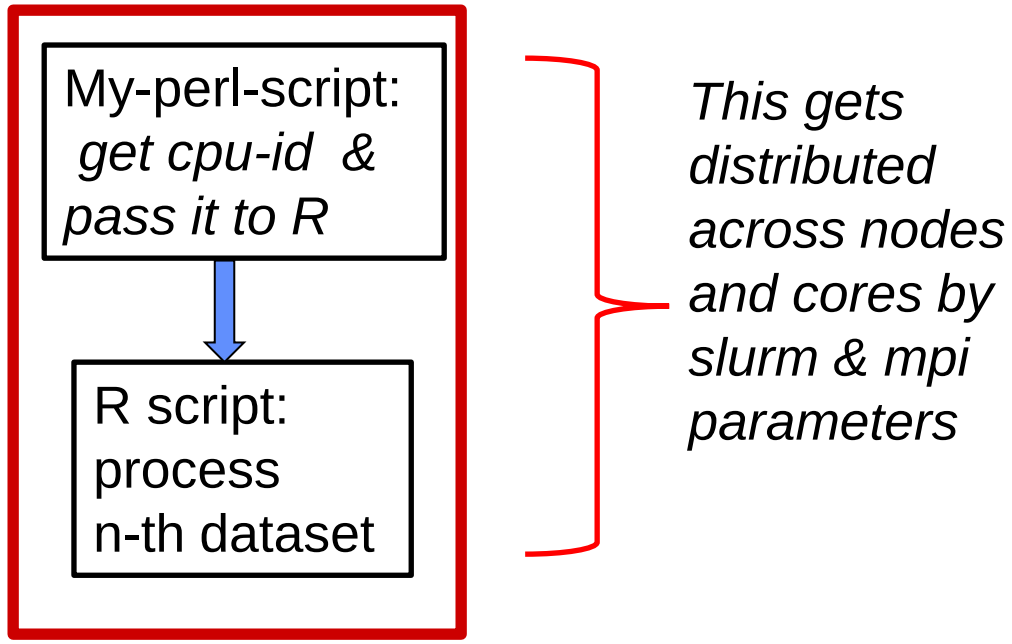
An option for (embarrassingly) Parallel R

1. Split data into N parts

2. In slurm batch script:

mpirun … my-perl-script





Slurm parameters: one R instance per core across all nodes

Normal batch job info

```
…
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=128
#SBATCH --cpus-per-task=1

module load slurm
module load cpu
module load gcc
module load intel-mpi

mpirun -genv I_MPI_PIN_DOMAIN=omp:compact ./get_mpirank_runcmd.pl
```

2 x128 = 256 mpi ranks

256 perl script/R instances
1 core each

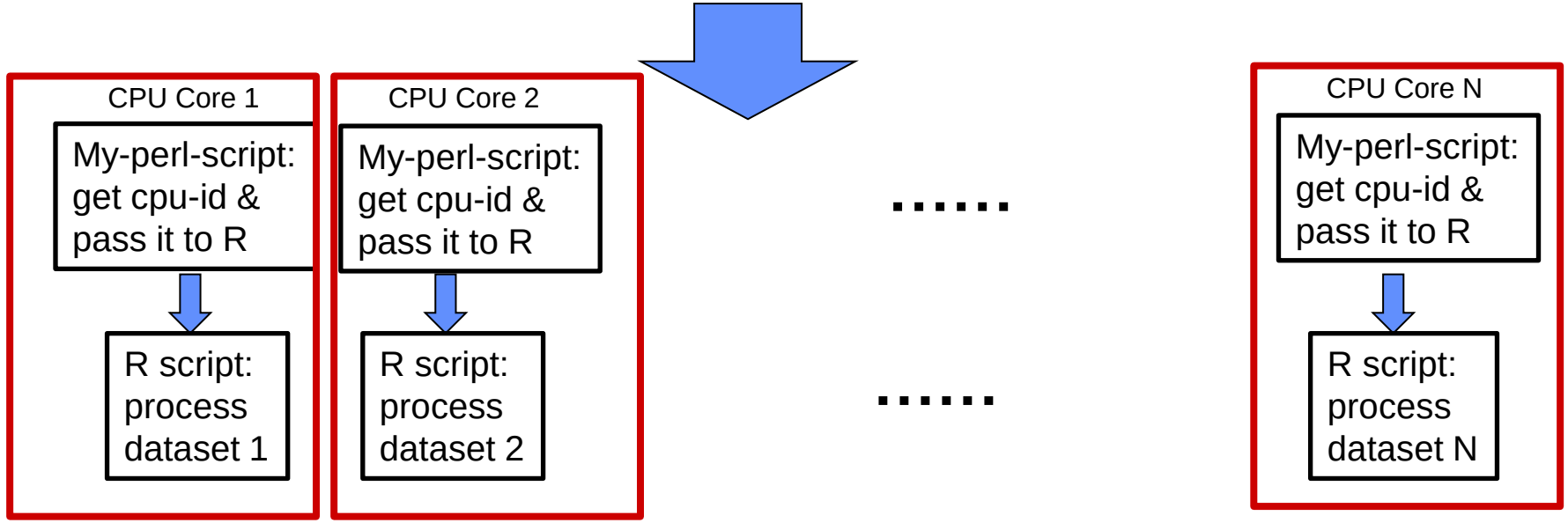
(based on /cm/shared/examples/sdsc/mpi-openmp-hybrid/hybrid-slurm.sb)

one R instance per core across all nodes

1. Split data

2. In slurm batch script:

mpirun … my-perl-script

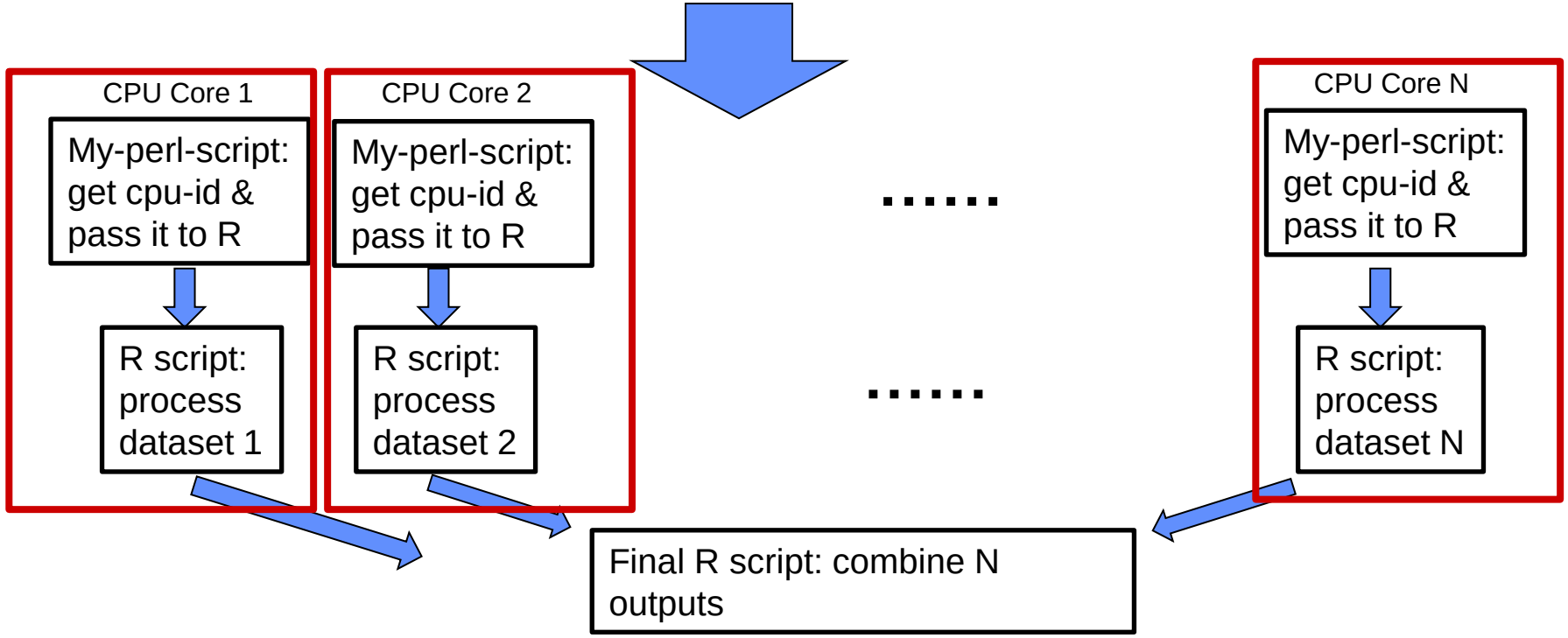


one R instance per core across all nodes

1. Split data

2. In slurm batch script:

mpirun … my-perl-script



More programming but perhaps more useful

Slurm parameters: one R instance per node with 128 cores per R instance

Normal batch job info

```
…
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=128

module load slurm
module load cpu
module load gcc
module load intel-mpi

module load r

mpirun -genv I_MPI_PIN_DOMAIN=omp:compact ./get_mpirank_runcmd.pl
```

2 x1 = 2 mpi ranks

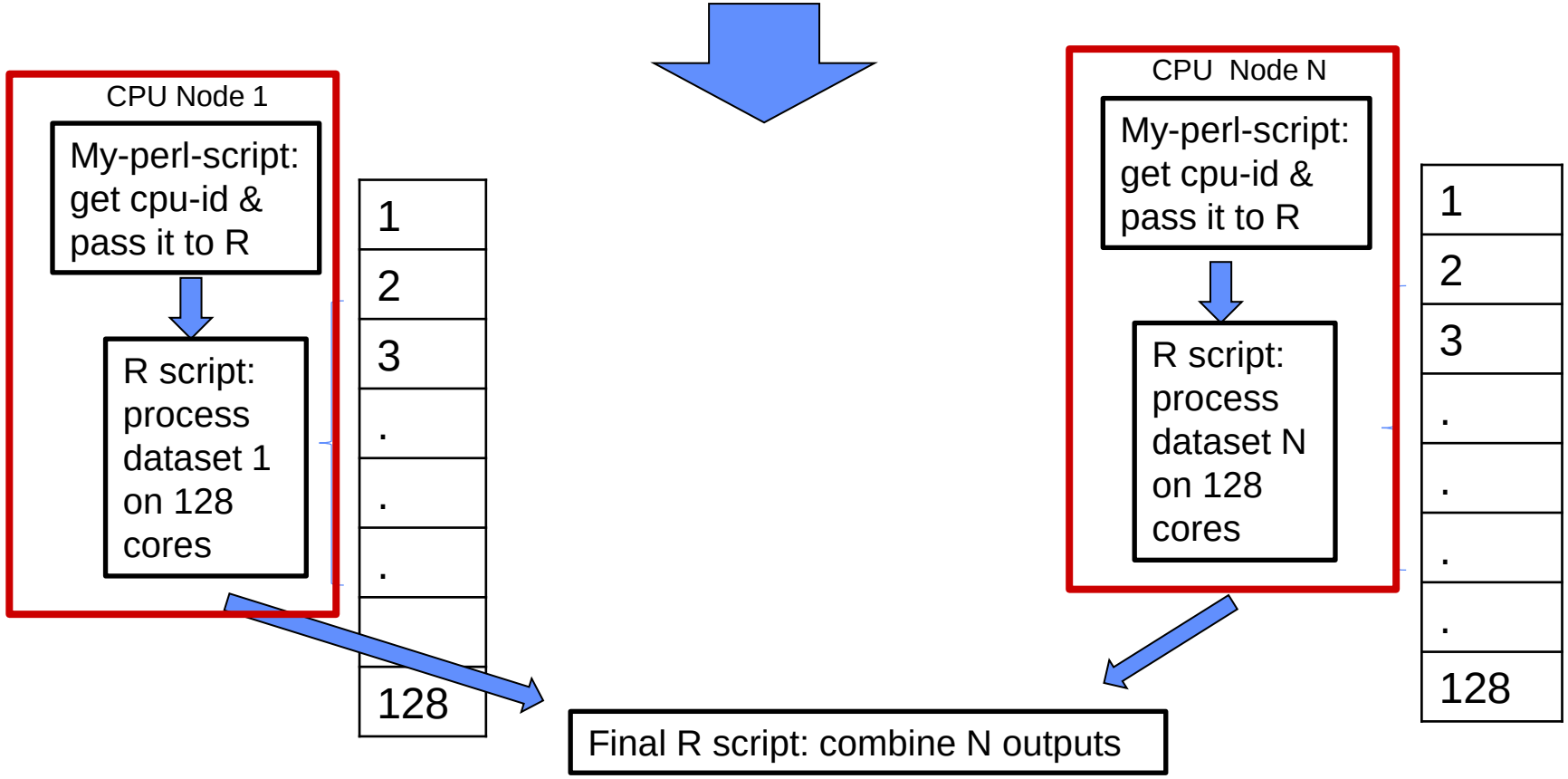
2 perl script/R instances
128 cores each
(doParallel can use them)

Example: One R instance per node, doParallel across all cores in each node

1. Split data

2. In slurm batch script:

mpirun … my-perl-script



Outline

- R and Scaling
- Parallel R
- Embarrassingly Parallel R
- A big data exploration of R

Big Data exploration

- Create large CSV file (117Gb) of X data matrix and Y outcomes:

  Y=X*B + noise (where X is 100K x50K)

- Run R ‘biglasso’ with a file backed ‘bigmatrix’ for X (b/c dataset too big)

- Use 1 compute node (248Gb RAM), use scratch (ssd) space.

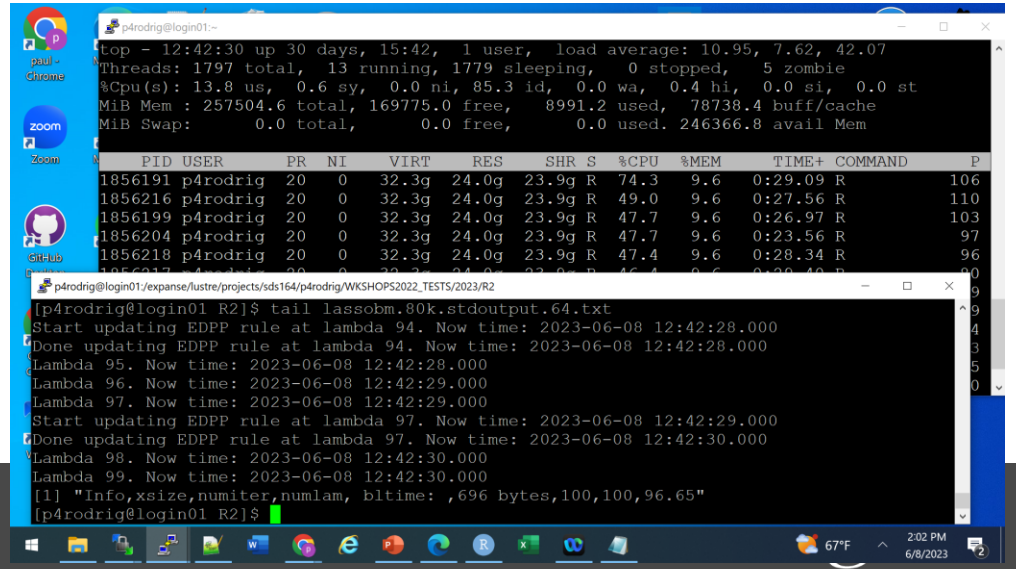
R

- R – biglasso (bigmatrix) package to set up file backed dataframe

https://cran.rstudio.com/web/packages/biglasso/

Issue: the file backend for the big data object path option was hard to get right – ended up just running out of scratch SSD as working directory;

Outcome: R copies everything into binary file backend and descriptor file and got results in about 2hours



Some considerations for big data

- Start with small data with interactive session or notebook – maybe even just use a smaller sample?

- Other packages (Matlab, Dask, Spark, Keras) can also work for many analytic tasks - your choice depends on your application needs, data size, single vs multiple node requirements.

- Packages generally work as documented, but often require working through some implementation issues or environment options for the session/job/execution

How to use R directly on Expanse

1. Get an interactive compute node:

2. Try
$ module spider r *(this tells you what modules you need)*

```
[p4rodrig@login02 ~]$ module spider r

-----------------------------------------------------------------
  r: r/4.0.2-openblas
-----------------------------------------------------------------

     Other possible modules matches:
        AMDuProf, amber, aria2, arm-forge, berkeley-db, bism

    You will need to load all module(s) on any one of the li
 "r/4.0.2-openblas" module is available to load.

      cpu/0.15.4  gcc/9.2.0

    Help:
```

3. Enter
$ module load cpu/0.15.4
$ module load gcc/9.2.0
$ module load r/4.0.2-openblas

$ R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
…..
Type 'q()' to quit R.

>

A note on installing R Packages (into your own directories)

- **In R (might help to be on interactive node):**

  install.packages('package-name')

  (see https://cran.r-project.org/ for package lists and reviews)

- **Sometimes you have to be explicit:**

  install.packages('ggmap',
  repos='http://cran.us.r-project.org',dependencies=TRUE)

  If compiling is required and you get an error, call support

  Packages are put into your /home/user/R directory

Other R package possibilities:

- Rspark - R interface to Spark
- R Keras – R interface to Keras
- pdbR - distributed matrix support (better for dense matrices vs Spark)
- Rgputools – GPU support

THE END