

LAB : R

Stampede



David Walling

Texas Advanced Computing Center
The University of Texas at Austin

July 12, 2013

Introduction

What you will learn

- How to perform basic data manipulation
- How to utilize 3rd party packages:
 - tm, caret, klaR, data.table, ggplot2
- How to profile code with Rprof
- How to parallelize code with mclapply
- How to use plots to analyze results
- General performance tips

What you will do

- Follow/modify example code **READ the CODE COMMENTS**
- Use R interactively with RStudio via VNC in 'vis' queue
- Submit R scripts as batch jobs to Stampede

Setup Lab Environment

Accessing Lab Files

- Log on to **Stampede** using your train## account.
- Untar the file lab_R.tar file (in ~train##).
- The new directory (lab_R) contains the scripts for the lab.
- cd into the appropriate subdirectory for an exercise.

You will be assigned this number.

```
$> ssh train##@stampede.tacc.utexas.edu
$> tar zxvf /work/00157/walling/lab_R.tar.gz .
$> cd lab_R
```

Install R Packages

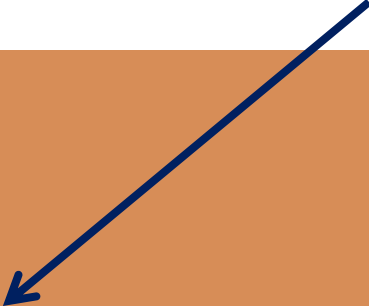
- R packages will be installed in each users home directory.
- Must install from login node. Compute nodes cannot access CRAN.
- From login node:

```
$> cd lab_R
$> source R_sourceme.R
$> R
R> source("InstallPackages.R")
Select mirror 86
R> q()
No save
```

VNC Session

- Start VNC server
 - \$> cd lab_R
 - \$> vim job.vnc
 - \$> vncpasswd (REMEMBER!!!)
 - \$> sbatch job.vnc
 - \$> cat vncserver.out

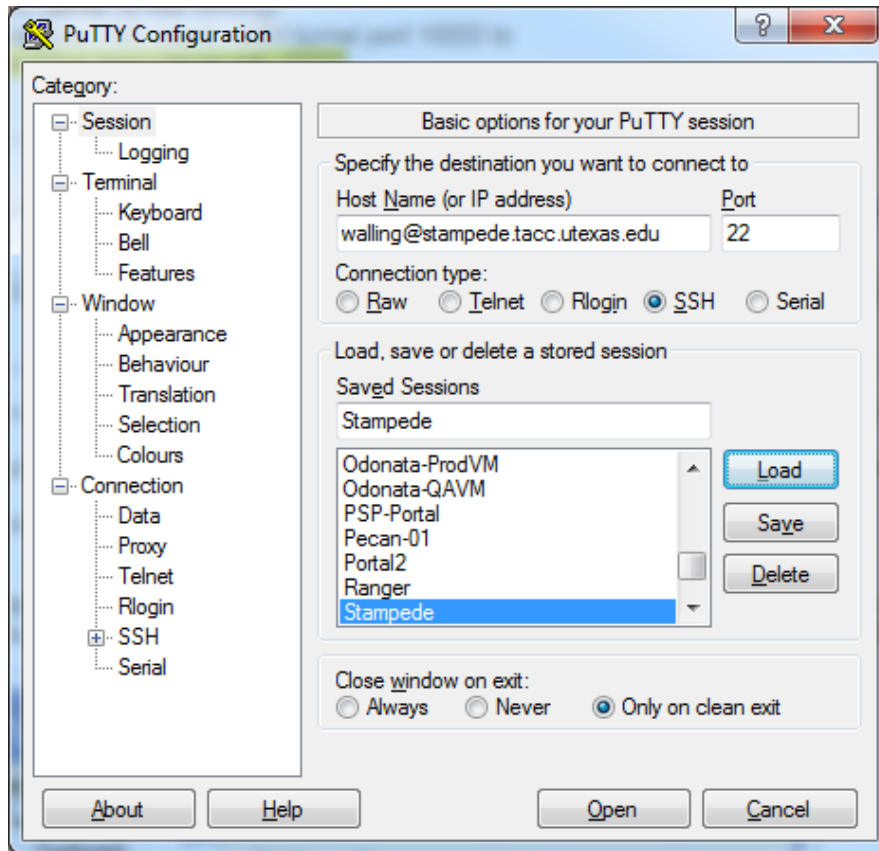
```
c558-103$ ./start.vnc
job execution at: Wed Jul 10 13:40:04 CDT 2013
running on node c558-103
using default UNC server /usr/bin/vncserver
memory limit set to 31194207 kilobytes
set wayness to
got UNC display :1
local (compute node) UNC port is 5901
got login node UNC port 15813
Created reverse ports on Stampede logins
Your UNC server is now running!
To connect via UNC client:  SSH tunnel port 15813 to stampede.tacc.utexas.edu:15813
                            Then connect to localhost::15813
```



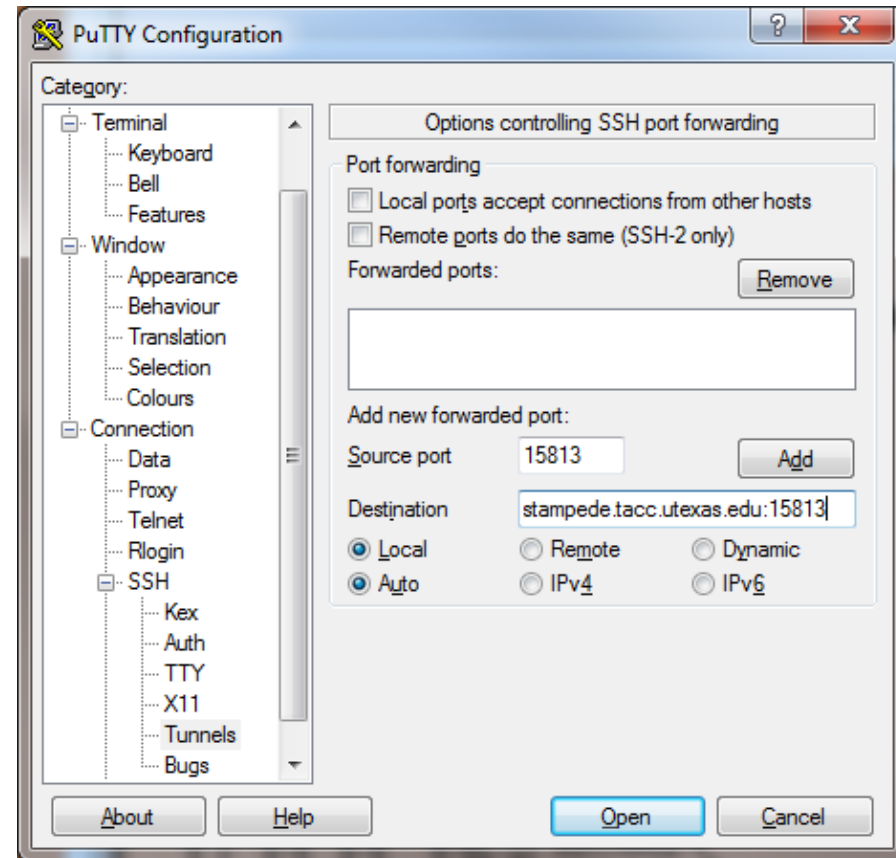
Your port!!!

Windows SSH Tunnel (Putty)

Load Session



Create Tunnel



- Enter Source/Destination
- Click Add
- Click Open

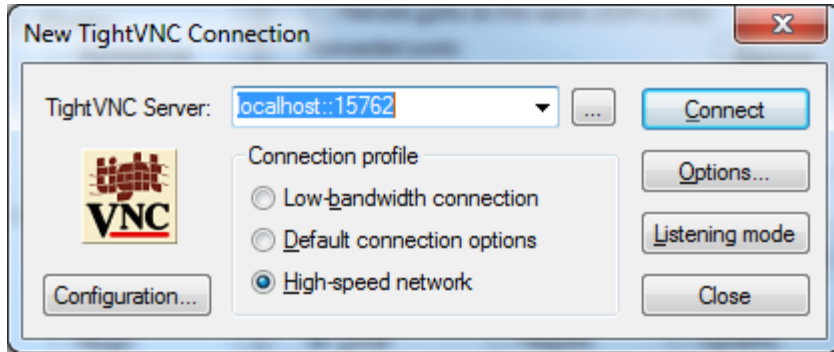
Linux/Mac SSH Tunnel

Create tunnel:

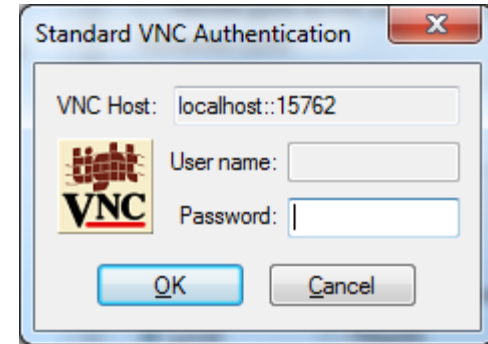
```
$> ssh -f -N -L xxxx:stampede.tacc.utexas.edu:yyyy  
username@stampede.tacc.utexas.edu
```


VNC Client

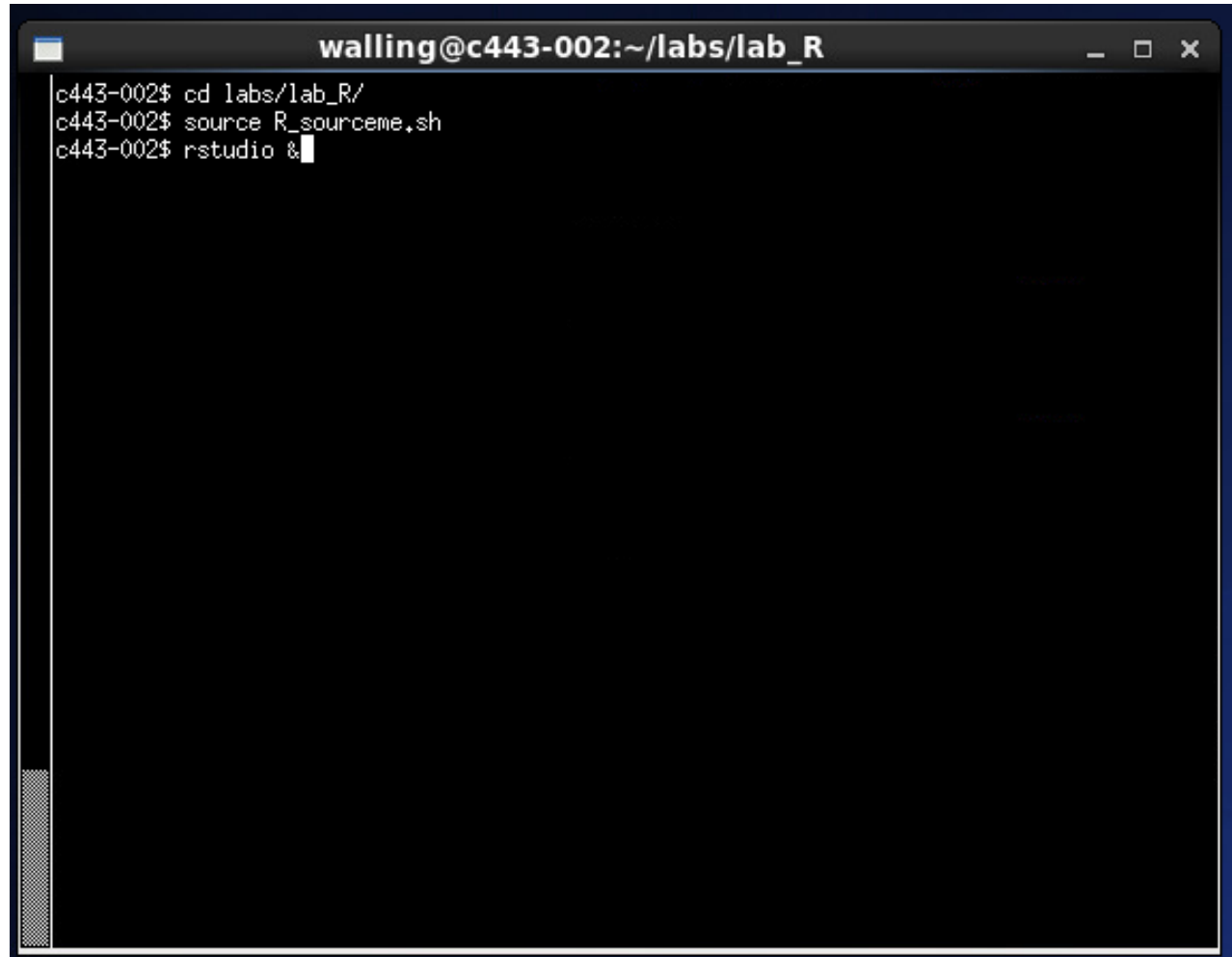
TightVNC



Enter vncpasswd



Open Project



```
walling@c443-002:~/labs/lab_R
c443-002$ cd labs/lab_R/
c443-002$ source R_sourceme.sh
c443-002$ rstudio &
```

RStudio

The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains R code for training a Naive Bayes classifier and analyzing its results. The code includes data loading, model training, prediction, and a custom function to generate a barplot of sensitivity.
- Console:** Shows the execution of the code, including the output of the `confusionMatrix` function and the execution of the `barplot` function.
- Workspace:** Lists the objects in the environment, including `all.data`, `test.data`, `training.data`, `actual`, `classes`, `cm`, `dir.path`, `predicted`, `predictions`, `result`, `trained.model`, `training.indexes`, `training.tdm`, and `values`.
- Plots:** A barplot titled "Sensitivity by Newsgroup" is displayed, showing the sensitivity for three classes: `alt.atheism` (red bar, sensitivity ~0.967), `comp.graphics` (green bar, sensitivity ~0.946), and `misc.forsale` (blue bar, sensitivity ~0.999).

```
163 training.tdm <- get.tdm(training.data$text)
164 training.data <- create.doc.vars.parallel(training.data, training.tdm)
165 trained.model <- train.klar.nBayes(training.data)
166
167 test.data <- create.doc.vars.parallel(test.data, training.tdm)
168 predictions <- classify.klar.nBayes.parallel(test.data, trained.model)
169 result <- table(predictions$class, test.data$class) #view the results. Diagonls = correct predictions
170
171 return(result)
172 }
173
174 #-----
175 # 7: Use caret to analyze results
176 #-----
177 analyze.results <- function(predicted, actual) {
178   cm <- confusionMatrix(data=predicted, reference=actual) #using package 'caret'
179   values <- cm$byClass[,1] #first column from all rows
180   classes <- sub('class:', '', rownames(cm$byClass))
181   par(mar=c(15,4,2,2)) #Extra large bottom margin
182   barplot(height=values,
183           names.arg=classes,
184           las=2,
185           ...
186 )
187 }
188
189 # Run the function
190 analyze.results(cm)
```

```
> cm <- confusionMatrix(data=predicted, reference=actual) #using package 'caret'
> str(cm)
List of 5
 $ positive: NULL
 $ table : 'table' int [1:3, 1:3] 8 0 0 0 15 0 0 1 6
 .. attr(*, "dimnames")=List of 2
 .. ..$ Prediction: chr [1:3] "alt.atheism" "comp.graphics" "misc.forsale"
 .. ..$ Reference : chr [1:3] "alt.atheism" "comp.graphics" "misc.forsale"
 $ overall : Named num [1:7] 0.967 0.946 0.828 0.999 0.5 ...
 .. attr(*, "names")= chr [1:7] "Accuracy" "Kappa" "AccuracyLower" "AccuracyUpper" ...
 $ byClass : num [1:3, 1:7] 1 1 0.857 1 0.933 ...
 .. attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:3] "class: alt.atheism" "class: comp.graphics" "class: misc.forsale"
 .. ..$ : chr [1:7] "Sensitivity" "Specificity" "Pos Pred Value" "Neg Pred Value" ...
 $ dots : list()
 - attr(*, "class")= chr "confusionMatrix"
> values <- cm$byClass[,1] #First column from all rows
> classes <- sub('class:', '', rownames(cm$byClass))
> par(mar=c(15,4,2,2)) #Extra large bottom margin
> barplot(height=values,
+         names.arg=classes,
+         las=2,
+         ylab='sensitivity',
+         main='Sensitivity by Newsgroup',
+         col=rainbow(length(classes))
+ )
> table(predictions$class, test.data$class)
```

	alt.atheism	comp.graphics	misc.forsale
alt.atheism	8	0	0
comp.graphics	0	15	1
misc.forsale	0	0	6

Note: currently cannot set `-max-ppsize` when using RStudio!!!

Building a Naïve Bayes Classifier

Raw Data

```
1 Path: cantaloupe.srv.cs.cmu.edu!crabapple.srv.cs.cmu.edu!fs7.ece.cmu.edu!europa.eng.gtefsd.com!gatech!asuvax!cs.utexas
2 From: joth@ersys.edmonton.ab.ca (Joe Tham)
3 Newsgroups: comp.graphics
4 Subject: Where can I find SIPP?
5 Message-ID: <yFXJ2B2w165w@ersys.edmonton.ab.ca>
6 Date: Mon, 05 Apr 93 14:58:21 MDT
7 Organization: Edmonton Remote Systems #2, Edmonton, AB, Canada
8 Lines: 11
9
10     I recently got a file describing a library of rendering routines
11 called SIPP (SImple Polygon Processor). Could anyone tell me where I can
12 FTP the source code and which is the newest version around?
13     Also, I've never used Renderman so I was wondering if Renderman
14 is like SIPP? ie. a library of rendering routines which one uses to make
15 a program that creates the image...
16
17                                     Thanks, Joe Tham
18
19 --
20 Joe Tham                               joth@ersys.edmonton.ab.ca
```

Naïve Bayes

```
> sample.data
  .file .class .text      boy  girl fast  slow  ran  dog  want love
[1,] 1    "Run" "The boy ran fast." TRUE  FALSE TRUE  FALSE TRUE  FALSE FALSE FALSE
[2,] 2    "Run" "The girl ran slow." FALSE TRUE  FALSE TRUE  TRUE  FALSE FALSE FALSE
[3,] 3    "Run" "He is fast."      FALSE FALSE TRUE  FALSE FALSE FALSE FALSE FALSE
[4,] 4    "Pet" "I love my dog."    FALSE FALSE TRUE  FALSE FALSE TRUE  FALSE TRUE
[5,] 5    "Pet" "I want a dog."    FALSE FALSE TRUE  FALSE FALSE TRUE  TRUE  FALSE
[6,] 6    "Pet" "My dog is slow."  FALSE FALSE FALSE TRUE  FALSE TRUE  FALSE FALSE
```

Compute probabilities for each class:

$$P(\text{Pet} \mid \text{dog}, \text{slow}) = P(\text{dog}, \text{slow} \mid \text{Pet}) * P(\text{Pet}) / P(\text{dog}, \text{slow})$$

$$P(\text{dog}, \text{slow} \mid \text{Pet}) = P(\text{dog} \mid \text{Pet}) * P(\text{slow} \mid \text{Pet}) \text{ *Assumes Independence!!!*}$$

$$P(\text{dog}, \text{slow}) = P(\text{dog}) * P(\text{slow})$$

$$P(\text{Run} \mid \text{dog}, \text{slow}) = P(\text{dog}, \text{slow} \mid \text{Run}) * P(\text{Run}) / P(\text{dog}, \text{slow})$$

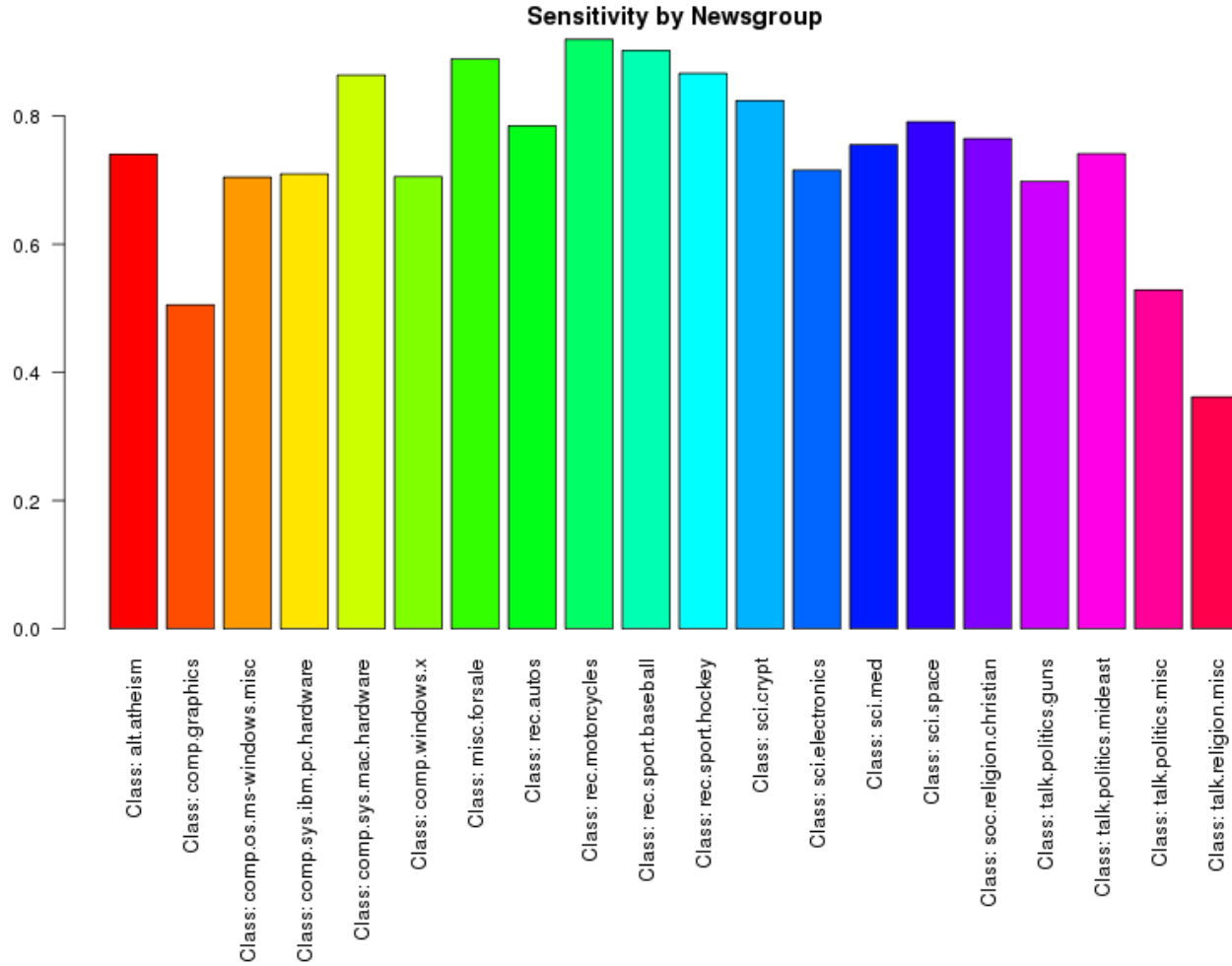
$$P(\text{dog}, \text{slow} \mid \text{Run}) = P(\text{dog} \mid \text{Run}) * P(\text{slow} \mid \text{Run})$$

$$P(\text{dog}, \text{slow}) = P(\text{dog}) * P(\text{slow}) \text{ *Same as above, we can ignore!!!*}$$

Assign class with highest probability:

$$P(\text{Pet} \mid \text{dog}, \text{slow}) > P(\text{Run} \mid \text{dog}, \text{slow})$$

Results



Code Walkthrough

- Create dataset from raw data files using basic R tools + *data.table/rbindlist* for speed
- Split dataset into train (90%) and test (10%) sets
- Add 'term' variables using *tm*
- Train our classifier using *klaR*
- Test our classifier and analyze results using *caret* and *ggplot2*
- Profile our code using *Rprof*
- Enhance using *parallel (mclapply)*

Performance Tips

- Avoid for loops! Use apply functions instead
- Only use data.frame when needed, i.e. have diff data types
- Use mclapply when inner code takes > 1min.
 - **Be careful of memory allocation!!!**
- Use optimized packages: data.table/fread/rbindlist, C implemented code, etc...
- Be aware of subtle diffs in similar functions: data.matrix vs as.matrix
- Search for existing code before writing your own
- Develop against subsets of data to get everything working
- Profile code at small/medium scale to quickly identify bottlenecks

Batch Script

rJob.slurm

```
1 #!/bin/bash
2 #
3 #-----
4 # An example script for launching an R job in batch mode
5 #-----
6 #
7 #SBATCH -J myRJob                # Job name
8 #SBATCH -o myRJob.out            # Name of stdout output file (%j expands to jobId)
9 #SBATCH -p normal                # Queue name
10 #SBATCH -N 1                    # Total number of nodes requested (16 cores/node)
11 #SBATCH -n 1                    # Total number of mpi tasks requested
12 #SBATCH -t 04:00:00             # Run time (hh:mm:ss) - 4 hours
13 #SBATCH -A TACC-DIC
14 #SBATCH --mail-user=walling@tacc.utexas.edu
15 #SBATCH --mail-type=begin       # email me when the job starts
16 #SBATCH --mail-type=end        # email me when the job finishes
17
18 module load R
19 Rscript myScript.R
```