R on the Engaging HPC Cluster

Steven Finch, sfinch@mit.edu

Research Computing Specialist, Sloan Technology Services

This tutorial will grow with time. My purpose is to give simple examples, using only basic statistics and rudimentary SLURM (job scheduling) commands. As questions arise, I will update and add materials as needed.

Let us start with batch job submissions. SSH log into "eosloan1.mit.edu". Upload (using SecureFX, for example) the following data file

furnace.txt

to your Engaging directory and (optionally) read

furnace_desc.txt

for background. To the same directory, upload both the R program

furnace.R

and the shell script

furnace_R.sh

containing a number of commands starting with #SBATCH. We will explain these commands shortly; for now, our focus is on getting various statistical calculations to work. Type the command

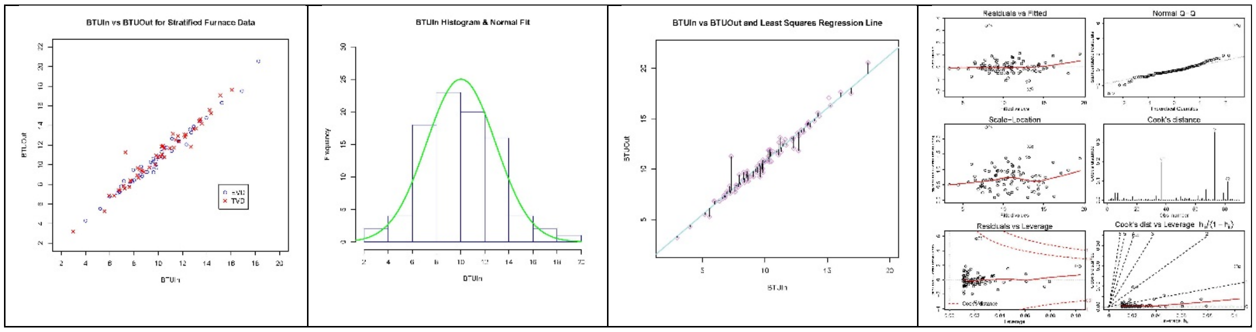
```
sbatch furnace_R.sh
```

in your remote terminal (SecureCRT or PuTTY, for example) and hit <Return>. You have now submitted a job to Engaging! Because this job is very short, three new files should readily appear in your directory:

furnace_R_err.txt, furnace_R_out.txt, furnace_R_out.pdf

The first of these should be empty since no errors ought to have occurred in the program execution. Download the second and third files to your local computer. The TXT output file contains all numerical tables and other texts from the job; the

PDF output file contains four plots (the latter consisting of six subplots). If you wish to explore other statistical aspects of the furnace dataset, simply edit my R code, overwrite the original program and resubmit the job.



Let us turn now to understanding the shell script:

```
#!/bin/bash

#SBATCH --job-name=furnace_R
#SBATCH --output=furnace_R_out.txt
#SBATCH --error=furnace_R_err.txt
#SBATCH -p sched_any_quicktest
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2000

module load engaging/R/3.2.5
module load sloan/R/CRAN

srun R --quiet --no-restore --no-save < furnace.R
```

The three options in the “srun” line are easily explained:

• --quiet	Don't print the R startup message
• --no-restore	Don't restore anything (previously saved objects or history file)
• --no-save	Don't save workspace at the end of the session

Note that, while the standard output and standard error filenames are prescribed in #SBATCH lines, the graphical PDF filename is given within the R code itself (first and last lines). The line

```
#SBATCH -p sched_any_quicktest
```

specifies the SLURM partition (or queue) under which the script will be run. The "sched_any_quicktest" partition is ideal for small, classroom examples like ours. It imposes a maximum runtime of 15 minutes on any job. Another partition, named "sched_mit_sloan", imposes a maximum runtime of 2 weeks. Since many pre-existing jobs may already be running on this partition, you might be forced to wait a long time for your task to begin.

The lines

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2000
```

require more explanation. For now, it suffices to say that we've requested 1 core on the cluster using 2 GB of RAM per core. Memory-intensive jobs will necessitate more cores. To be continued…

Our furnace data analysis used only base R commands. Here is another example – involving 93 cars (both domestic and foreign) – that employs several packages including *ggplot2* and *stargazer*. As before, upload the data file

93cars.txt

to your Engaging directory and (optionally) read

93cars_desc.html

for background. To the same directory, upload both the R program

93cars.R

and the shell script

93cars_R.sh

Type the command

sbatch 93cars_R.sh

and hit <Return>. The shell script line:

```
module load sloan/R/CRAN
```

has conveniently made all packages available to us; note that we need only invoke "library()" to bring *ggplot2* and *stargazer* into our R session. The latter gives nice HTML output tables that are easily imported into MS Word:

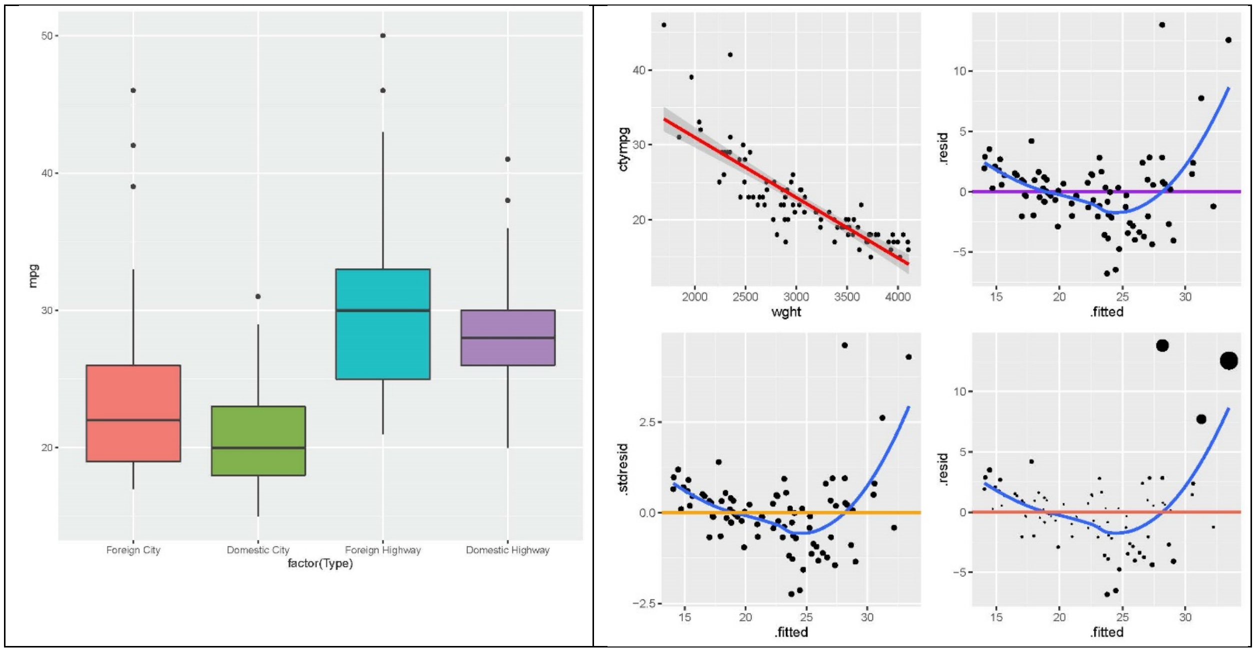
Domestic Car Price, City MPG, Highway MPG & Weight

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Median	Pctl(75)	Max
price	48	18.573	7.817	7.400	13.475	16.300	20.725	40.100
ctympg	48	20.958	3.994	15	18	20	23	31
hwympg	48	28.146	4.151	20	26	28	30	41
wght	48	3,195.312	565.228	1,845	2,705	3,282.5	3,638.8	4,105

Foreign Car Price, City MPG, Highway MPG & Weight

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Median	Pctl(75)	Max
price	45	20.509	11.307	8.000	11.600	19.100	26.700	61.900
ctympg	45	23.867	6.673	17	19	22	26	46
hwympg	45	30.089	6.248	21	25	30	33	50
wght	45	2,942.333	593.753	1,695	2,475	2,950	3,405	4,100

(a considerable improvement on base R tables!) The former gives extraordinary graphics:



that clearly convey the power of visualization in statistics.

We finally discuss briefly how to open an interactive RStudio session on Engaging. SecureCRT no longer suffices; you will need to use an X forwarding client such as MobaXterm (for Windows) or XQuartz (for Mac). With this in place, the SLURM commands

```
module load engaging/rstudio-desktop/0.98.1103

srun --x11 --cpus-per-task=2 --mem=8000 --pty -p sched_mit_sloan rstudio
```

open a new GUI window (resembling the familiar RStudio console) into which you can type commands and perform analysis. Here we've requested 2 cores on the cluster using a total of 8 GB of RAM (hence 4 GB per core). This contrasts with our earlier batch R jobs, which did not use multiple cores.

Please feel free to write to me (sfinch@mit.edu) with questions and suggestions on how to improve this tutorial.

Addendum: Rscript rather than R CMD BATCH

Another way of performing the same analysis of 93 cars is to replace “R” by “Rscript” in the shell and to remove the redirect:

```
srun Rscript --quiet --no-restore --no-save 93cars.R
```

It’s curious that a similar replacement for the furnace data:

```
srun Rscript --quiet --no-restore --no-save furnace.R
```

does *not* work! Here we need to insert two lines

```
library(methods)
library(grDevices)
```

at the top of the furnace.R program, as explained at

http://stackoverflow.com/questions/19468506/rscript-could-not-find-function

(Rscript doesn't load the packages by default because it's time-consuming).

Addendum: Replacing DOS line breaks

If you receive a "Batch script contains DOS line breaks instead of expected UNIX line breaks" error message, you need to replace all DOS carriage returns by UNIX carriage returns. This is easily done on the UNIX command line within Engaging:

```
dos2unix furnace_R.sh
```

and eliminates the need to do further editing using VI or other UNIX text editor.