Python 2.7 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.csv

to your Engaging directory and (optionally) read

furnace_desc.txt

for background. To the same directory, upload both the Python 2.7 program

furnace_Python.py

and the shell script

furnace_Python.sh

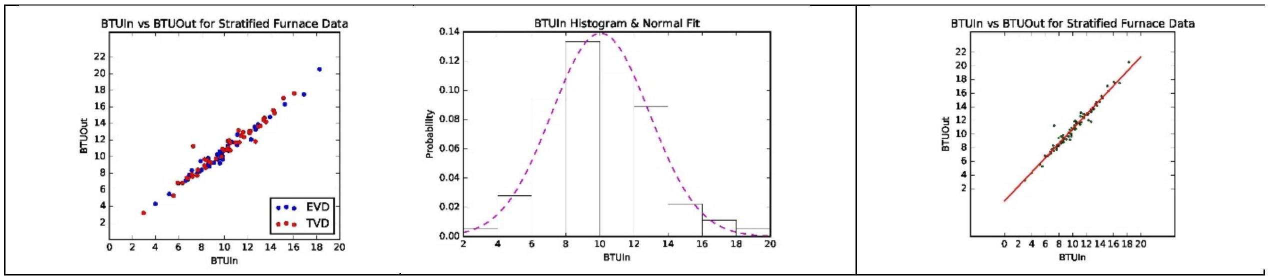
containing a number of commands starting with #SBATCH. We will explain these commands shortly; for now, our focus is on getting various statistical calculations to work. Type the command

```
sbatch furnace_Python.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_Python_err.txt, furnace_Python_out.txt, furnace_Python_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 three plots. If you wish to explore other statistical aspects of the furnace dataset, simply edit my Python code, overwrite the original program and resubmit the job.



Let us turn now to understanding the shell script:

```
#!/bin/bash

#SBATCH --job-name=furnace_Python
#SBATCH --output=furnace_Python_out.txt
#SBATCH --error=furnace_Python_err.txt
#SBATCH -p sched_any_quicktest
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2000

module load engaging/python/2.7.10
module load sloan/python/2.7-modules

srun xvfb-run python < furnace_Python.py
```

Note that, while the standard output and standard error filenames are prescribed in #SBATCH lines, the graphical PDF filename is given within the Python code itself (seventh 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…

We finally discuss briefly how to open an interactive Python 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/python/2.7.10

module load sloan/python/2.7-modules

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

python
```

open a rudimentary Python interface. If, within this, you type

```
import idlelib.PyShell

idlelib.PyShell.main()
```

then an Idle IDE (integrated development environment) arises, 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 Python job, which did not use multiple cores.

A more elaborate IDE known as Spyder (scientific python development environment) is available by typing just one line in SLURM:

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

At a later point, the Python modules "scrapy" and "textmining" deserve our attention, given their obvious application to web crawling and data mining.

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

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_Python.sh
```

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