

Lecture 10: Supercomputing, Bash & Command Line

LING 1340/2340: Data Science for Linguists

Na-Rae Han, Daniel Zheng

Objectives

- ▶ Supercomputing at CRC
 - ◆ Server access through SSH
 - ◆ Running jobs
 - ◆ Code efficiency
- ▶ More command line, bash shell
 - ◆ Regex-based text search using grep
 - ◆ Batch processing through for loop
 - ◆ Shell scripting

Let us now supercompute.




3/21/2019

By Argonne National Laboratory's Flickr page - originally posted to Flickr as Blue Gene / P From Argonne National Laboratory Uploaded using F2ComButton, CC BY-SA 2.0, <https://commons.wikimedia.org/w/index.php?curid=6412306>

You got a supercomputing account.

- ▶ You received this mysterious email:

Center for Simulation and Modeling (SaM) Account

 **no-reply@core.sam.pitt.edu**
Thu 11/2/2017 10:42 AM
Han, Na-Rae ✓

Dear user,

Welcome to SaM!

An account has been created for you on Center for Simulation and Modeling (SAM) resources. Your username is "naraehan" (without the quotes). All authentication is through Pitt's Active Directory. Therefore, your password is the password associated with your Pitt account.

After logging in, you can update any of your account personal information by clicking on the "My account" link found near the top right of your browser window.

I got you all an account at Pitt's **Center for Research Computing** (CRC, formerly known as SaM)

CRC: Center for Research Computing

▶ <https://crc.pitt.edu>



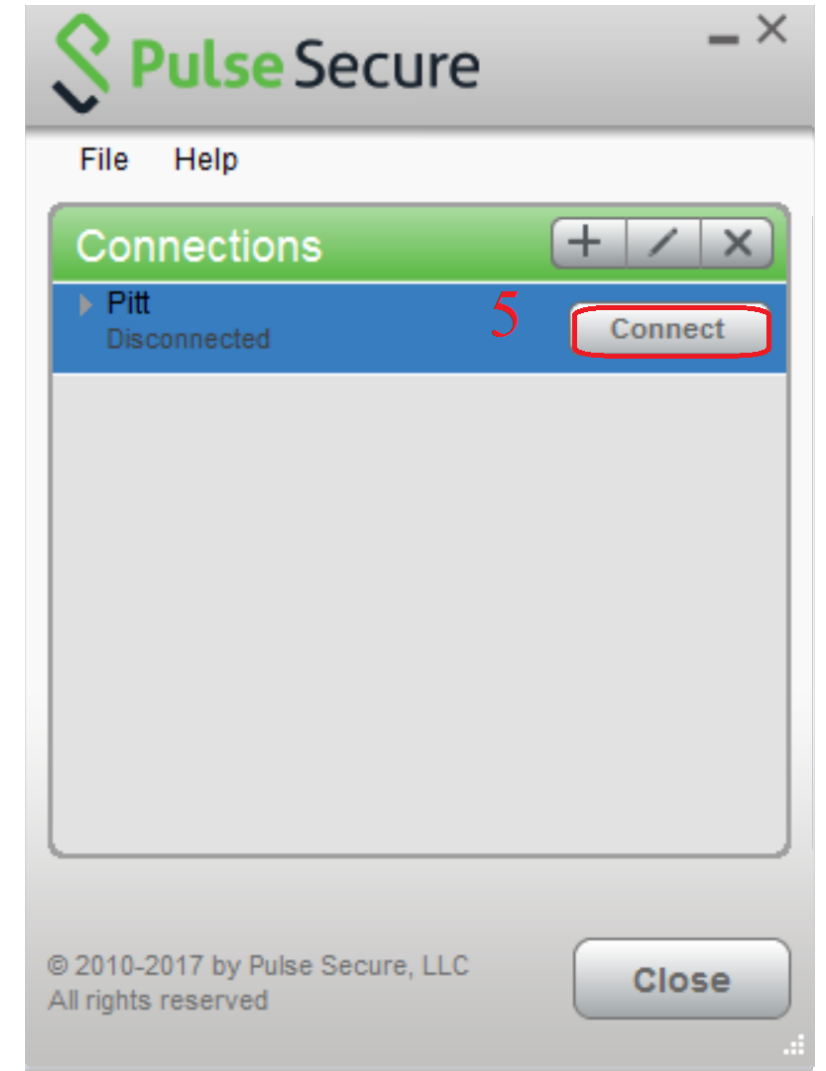
▶ Documentation here: <https://crc.pitt.edu/UserSupport>

◆ READ them!

▶ h2p cluster user guide: <https://crc.pitt.edu/h2p>

CRC machines require secure access

- ▶ Unless we are accessing from a *wired* connection on Pitt's campus, our laptop should be running a **Secure Remote Access client**.
 - ▶ Details in the h2p cluster user guide:
 - ◆ <https://crc.pitt.edu/h2p>
- ← Windows users: no-need to download Xming or Putty. You got git Bash!



Accessing CRC's cluster

- ▶ Remote-access your account via SSH:
 - ◆ `ssh yourpittid@h2p.crc.pitt.edu`
- ▶ Getting your bearings:
 - ◆ Where are you? `pwd`
 - ◆ What is your user 'group'? `groups`
 - ◆ Is python installed on this machine? `which python`
 - ◆ What are your configuration files? `ls -a`
 - ◆ `.bash_profile`
 - ← Customize with your own aliases, etc.
 - ◆ `.bash_history`
 - ← Bash commands you typed in are logged here.

Shared data space for the group

- ▶ You are all assigned to our class **group**: [ling1340-2019s](#)

```
naraehan@login0:~  
[naraehan@login0 ~]$ groups  
nhan ling1340-2017f pyling ling1340-2019s
```

- ▶ Our shared data sets are stored in: [/zfs2/ling1340-2019s/shared_data/](#)

```
naraehan@login0:/zfs2/ling1340-2019s/shared_data/yelp_dataset_13  
[naraehan@login0 ~]$ cd /zfs2/ling1340-2019s/shared_data/  
[naraehan@login0 shared_data]$ ls  
ETS_Corpus_of_Non-Native_Written_English  nltk_data  word_vectors  yelp_dataset_13  
[naraehan@login0 shared_data]$ cd yelp_dataset_13/  
[naraehan@login0 yelp_dataset_13]$ ls  
business.json          review.json  
checkin.json           tip.json  
Dataset_Challenge_Dataset_Agreement.pdf  user.json  
photo.json             Yelp_Dataset_Challenge_Round_13  
[naraehan@login0 yelp_dataset_13]$
```

ETS corpus, Yelp dataset,
and nltk data!

Processing multiple files -- for loop!

- ▶ Your command line is actually running a programming environment: **bash shell**.
- ▶ You can *program* in command line, even **for loops**!

```
narae@T450s MINGW64 ~/Desktop/inaugural
$ for file in *.txt
> do
> iconv -f US-ASCII -t UTF-16 $file > try/$file
> echo $file complete
> done
1789-Washington.txt complete
1793-Washington.txt complete
1797-Adams.txt complete
1801-Jefferson.txt complete
1805-Jefferson.txt complete
1809-Madison.txt complete
1813-Madison.txt complete
1817-Monroe.txt complete
1821-Monroe.txt complete
1825-Adams.txt complete
```

Keywords **do** and **done** indicate the **for loop block**.

iconv is a handy encoding conversion tool.

← Converting all files from ASCII to UTF-16

```
[naraehan@login0 ~]$ cd /zfs2/ling1340-2019s/shared_data/ETS_Corpus_of_Non-Native_Written_English/
```

```
[naraehan@login0 ETS_Corpus_of_Non-Native_Written_English]$ ls
```

```
data docs index.html tools
```

```
[naraehan@login0 ETS_Corpus_of_Non-Native_Written_English]$ cd data/text/prompts/
```

```
[naraehan@login0 prompts]$ ls
```

```
P1.txt P2.txt P3.txt P4.txt P5.txt P6.txt P7.txt P8.txt
```

```
[naraehan@login0 prompts]$ cat P1.txt
```

```
Do you agree or disagree with the following statement?
```

```
It is better to have broad knowledge of many academic subjects than to specialize in one specific subject.
```

```
Use specific reasons and examples to support your answer.
```

```
[naraehan@login0 prompts]$ head -3 P1.txt | tail -1
```

```
It is better to have broad knowledge of many academic subjects than to specialize in one specific subject.
```

```
[naraehan@login0 prompts]$
```

```
[naraehan@login0 prompts]$ for x in *txt
```

```
> do
```

```
> echo $x
```

```
> head -3 $x | tail -1
```

```
> done
```

```
P1.txt
```

```
It is better to have broad knowledge of many academic subjects than to specialize in one specific subject.
```

```
P2.txt
```

```
Young people enjoy life more than older people do.
```

```
P3.txt
```

```
Young people nowadays do not give enough time to helping their communities.
```

```
P4.txt
```

```
Most advertisements make products seem much better than they really are.
```

```
P5.txt
```

```
In twenty years, there will be fewer cars in use than there are today.
```

```
P6.txt
```

```
The best way to travel is in a group led by a tour guide.
```

```
P7.txt
```

```
It is more important for students to understand ideas and concepts than it is for them to learn facts.
```

```
P8.txt
```

```
Successful people try new things and take risks rather than only doing what they already know how to do we
```

```
ll.
```

Grepping the inaugural on CRC

Activity
15 minutes



- ▶ Download inaugural.zip from NLTK's data page. How?
- ▶ Unzip the .zip archive. How?
- ▶ Grep for 'prosperity'. Hmm lines are too long...
- ▶ Use fold to fold long lines.
 - ◆ Line breaks in the middle of words! How to break along space? Use man page to find out.
 - ◆ Create another version inaugural2 with folded lines. Use a for loop.
- ▶ Which presidents talked about 'Russia'? 'war'? 'unity'?
- ▶ How about 'God bless'?
- ▶ Which presidents used split infinitives?
 - ◆ How to print out more context: 2 lines before and after?

Grepping the inaugural

Activity
15 minutes



- ▶ Download inaugural.zip from NLTK's data page. How?

```
wget https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/packages/corpora/inaugural.zip
```

- ▶ Unzip the .zip archive. How?

- ◆ `unzip inaugural.zip`

- ▶ Grep for 'prosperity'. Hmm lines are too long...

- ▶ Use `fold` to fold long lines.

- ◆ Line breaks in the middle of words! How to break along space? Use man page to find out.
- ◆ Create another version inaugural2 with folded lines. Use a for loop.

- ▶ Which presidents talked about 'Russia'? 'war'? 'unity'?

- ▶ How about 'God bless'?

```
mkdir inaugural2
```

```
cd inaugural
```

```
for x in *.txt; do fold -s $x > ../inaugural2/$x; done
```

- ▶ Which presidents used split infinitives?

- ◆ How to print out more context: 2 lines before and after?
 - ◆ `grep -PC 2 "\bto \w+ly " *.txt`

Before you get carried away



- ▶ Do NOT yet run any jobs that may be resource-intensive.
- ▶ This is a powerful super-computer, shared by many research groups at Pitt.
 - ◆ Our class as a group has a limited, shared allocation.
 - ◆ You do not want to accidentally initiate a run-away process and hog resources.
- ▶ There are PROPER ways to run jobs.
 - ◆ Dan will show you now.

Using CRC clusters

► Job submissions

- ◆ On a computing cluster, many people are using the same resources so we have a "job queue" that accepts job submissions
- ◆ CRC and many other clusters use [Slurm](#) for managing and scheduling these jobs.
- ◆ The following slides are mostly based on intro slides by CRC's Barry Moore III:
 - ◆ <https://pitt.app.box.com/v/crc-cluster-naraehan>
- ◆ Basic shell commands from Barry:

Command	Description
<code>cd <directory></code>	Change to directory
<code>echo <thing></code>	Print thing to command line
<code>ls [<directory>]</code>	List contents of directory
<code>cp <file> <location></code>	Copy file to a location
<code>man <command></code>	Manual on command
<code>mkdir <directory></code>	Make directory
<code>pwd</code>	Print working directory
<code>export [<VAR>=<thing>]</code>	Make or view environment variables



Modules

- ▶ Before we start submitting jobs, we need to learn how to use modules.
- ▶ CRC uses something called [Lmod](#) to manage modules, which are basically environment configurations for different types of projects.
- ▶ We just need python, so type `module spider python` to view what's available, and load an appropriate version, like `module load python/3.7.0`
- ▶ This sets things up so now you have the python 3.7.0 installation provided by CRC, complete with a bunch of useful packages.

Command	Description
<code>module avail</code>	What's available in my Lmod Path?
<code>module spider</code>	What's available?
<code>module load <module>...</code>	Load a module
<code>module purge</code>	Unload all modules
<code>module list</code>	List loaded modules

Slurm Jobs

- ▶ To make a slurm job script, you basically need to write a bash script of what you would do to run your program on the command line. This is just a text file, usually with a **.sh** ending.
- ▶ Also need some slurm configs
- ▶ Example (let's call this **hello.sh**)

```
#!/usr/bin/env bash

#SBATCH --job-name=hello
#SBATCH --output=hello.out
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --partition=smp
#SBATCH --cluster=smp

echo "hello world"
```

<-- Copy this into a file
and name it something
like **hello.sh**

Below are some other Slurm config options (prefix with #SBATCH) as in hello.sh. EVEN MORE at <https://slurm.schedmd.com/sbatch.html>

Option	Environment Variables
--output	-
--time	(Format: DAYS-HOURS:MINUTES:SECONDS)
--job-name	SLURM_JOB_NAME
--nodes	SLURM_NNODES
--ntasks	SLURM_NTASKS
--cpus-per-task	SLURM_CPUS_PER_TASK
--ntasks-per-node	SLURM_NTASKS_PER_NODE
--partition	SLURM_JOB_PARTITION
--mem	SLURM_MEM_PER_NODE
--account	SLURM_JOB_ACCOUNT

Job management commands

So from the directory with our [hello.sh](#) script, we can submit it with `sbatch hello.sh`

This should run pretty much instantly and we can check our [hello.out](#) output file.

Try tacking on [crc-job-stats.py](#) at the end.

Command	Description
<code>sinfo</code>	Quick view of partitions
<code>sbatch <job></code>	Submit your job ^a
<code>squeue</code>	View all jobs
<code>squeue -u <user></code>	Look at your jobs
<code>scancel <jobid></code>	Cancel your job
<code>crc-sinfo.py</code>	<code>sinfo</code> wrapper
<code>crc-squeue.py</code>	<code>squeue</code> wrapper
<code>crc-scancel.py <jobid></code>	<code>scancel</code> wrapper
<code>crc-usage.pl</code>	View your group's usage

To-do #12 redux

Activity
20 minutes



- ▶ This time on h2p!
- ▶ Copy files *from local computer* to remote servers with `scp`
 - ◆ Local to remote:
`scp foobar.txt your_username@remotehost.edu:/some/remote/directory`
 - ◆ Remote back to local:
`scp your_username@remotehost.edu:foobar.txt /some/local/directory`
 - ◆ Let's copy over your `process_reviews.py` file (part of To-do12) to your CRC home directory
`scp process_reviews.py daz53@h2p.crc.pitt.edu:~`
- ▶ Alternatively you can use `sftp`
 - ◆ `scp/sftp` are handy for moving large files and directories when you can't just copy paste

To-do #12 redux: setting up

(1) New location of yelp data:

```
/zfs2/ling1340-2019s/shared_data/yelp_dataset_13/review.json
```

(2) We'll sample 2 million lines from this so this doesn't take forever:

```
shuf /zfs2/ling1340-2019s/shared_data/yelp_dataset_13/review.json -n 2000000 >
~/review_2mil.json
```

(3) Running our script on this data will look like:

```
python process_reviews.py ~/review_2mil.json
```

(4) But we need to load the appropriate python environment first with Lmod, so we need:

```
module load python/3.7.0
```

(5) Now we can toss all this into a bash script. Let's call it `todo12.sh`

(6) Start with `hello.sh` (make a copy of it)

(7) Change the bash commands at the bottom to run our script for todo 12, and change the name and output file to something like `todo12` and `todo12.out`

```
#!/usr/bin/env bash
```

```
#SBATCH --job-name=todo12  
#SBATCH --output=todo12.out  
#SBATCH --nodes=1  
#SBATCH --ntasks=1  
#SBATCH --partition=smp  
#SBATCH --cluster=smp
```

```
module load python/3.7.0  
python process_reviews.py review_2mil.json
```

todo12.sh

```
import pandas as pd  
import sys  
from collections import Counter  
  
filename = sys.argv[1]  
  
df = pd.read_json(filename, lines=True, encoding='utf-8')  
print(df.head(5))  
  
wtoks = ' '.join(df['text']).split()  
wfreq = Counter(wtoks)  
print(wfreq.most_common(20))
```

process_reviews.py
(What we used for
To-do #12)

To-do #12 redux: running a job

- ▶ Submit your job with `sbatch todo12.sh` and check it out with `squeue -u daz53` (or your username). Add `-i 5` to refresh every 5 seconds automatically
- ▶ Should be done shortly.
- ▶ `cat todo12.out`
 - ◆ Memory error, nice!
 - ◆ Check out how much it used with `seff <job id>`, view past jobs with `sacct`
- ▶ What to do...

```
[naraehan@login0 ~]$ cat todo12.out
slurmstepd: error: Job 2083029 exceeded memory limit (10397452 > 8192000), be
slurmstepd: error: Exceeded job memory limit
slurmstepd: error: *** JOB 2083029 ON smp-n42 CANCELLED AT 2019-03-21T19:14:5
```

Quick aside

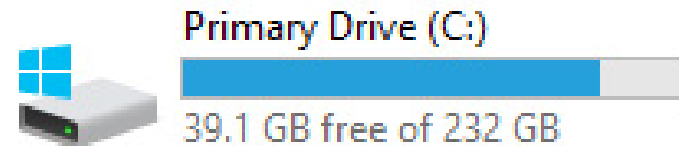
▶ Memory here refers to Random Access Memory (RAM)

- ◆ You probably have 4 or 8 GB on your laptop
- ◆ Running programs uses RAM to store temporary data (in our case opened files, variables, etc) that they use or produce
- ◆ Stuff stored in RAM is removed when a program terminates, or if your computer shuts off.
- ◆ Opening a file in a program loads it into RAM
- ◆ Running out of RAM on your laptop will probably cause your computer to freeze/crash
- ◆ Expensive per GB

▶ NOT Disk space aka your hard drive -->

- ◆ Disk space stores files long-term
- ◆ Cheap per GB, 256+GB is pretty standard.

Devices and drives (3)



To-do #12 redux: managing RAM use

- ▶ Let's tweak our Python code to use less RAM...
 - ◆ Copy the script and rename it, maybe `process_reviews_eff.py`, like so:

```
cp process_reviews.py process_reviews_eff.py
```
 - ◆ Edit the code (→ see next slide):
 - ◆ Try using `chunksize` parameter, which will give us an **iterator** that goes over a certain # of lines at a time, so we don't read the whole file at once...
 - ◆ Update Counter each iteration
 - ◆ Create a new job script and submit!
- ◆ We *could* also add `#SBATCH --mem 20000` to our slurm script, which will allocate 20k MB, or 20 GB of RAM instead of the default 8GB. Kind of cheating though... Find details on what is available at <https://crc.pitt.edu/h2p#nodeconfig>
 - ◆ Up to 190 GB RAM per node!

To-do #12 redux, done better

- ▶ Revised `process_reviews_eff.py`
- ▶ Works and uses much less RAM! Check `seff <jobid>`

```
import pandas as pd
import sys
from collections import Counter

filename = sys.argv[1]

df_chunks = pd.read_json(filename, chunksize=10000, lines=True, encoding='utf-8')

wfreq = Counter()

for chunk in df_chunks:
    for text in chunk['text']:
        wfreq.update(text.split())

print(wfreq.most_common(20))
```


Efficient code matters

```
naraehan@login1:~  
[naraehan@login1 ~]$ sbatch todo12.sh  
Submitted batch job 2083026 on cluster smp  
[naraehan@login1 ~]$ queue -u naraehan  
      JOBID PARTITION      NAME      USER ST  
      2083026      smp      todo12 naraehan R  
[naraehan@login1 ~]$ seff 2083026  
Job ID: 2083026  
Cluster: smp  
User/Group: naraehan/nhan  
State: CANCELLED (exit code 0)  
Cores: 1  
CPU Utilized: 00:00:26  
CPU Efficiency: 86.67% of 00:00:30 core-walltime  
Job Wall-clock time: 00:00:30  
Memory Utilized: 11.26 GB  
Memory Efficiency: 144.18% of 7.81 GB
```

process_reviews.py
(Original code used for
To-do #12)

- ▶ Tries to use a LOT of memory (11+ GB), gets the job killed...

```
naraehan@login1:~  
[naraehan@login1 ~]$ sbatch todo12.sh  
Submitted batch job 2082992 on cluster smp  
[naraehan@login1 ~]$ queue -u naraehan  
      JOBID PARTITION      NAME      USER ST  
      2082992      smp      todo12_t naraehan R  
[naraehan@login1 ~]$ seff 2082992  
Job ID: 2082992  
Cluster: smp  
User/Group: naraehan/nhan  
State: COMPLETED (exit code 0)  
Cores: 1  
CPU Utilized: 00:01:07  
CPU Efficiency: 100.00% of 00:01:07 core-walltime  
Job Wall-clock time: 00:01:07  
Memory Utilized: 291.15 MB  
Memory Efficiency: 3.64% of 7.81 GB
```

process_reviews_eff.py
(Previous slide, updated to
use chunksize)

- ▶ Job actually finishes in about a minute, uses a fraction of memory (only 290 MB)!

Why use CRC?

- ▶ Load and compute with data that you can't handle on your laptop
- ▶ Remote computation, can submit job and close your laptop
- ▶ Preconfigured environments that can be tailored to suit your needs
 - ◆ Avoid hassle of setting up the right programming environment on your laptop

But mainly for the computing power.

CRC Hub

▶ Run Jupyter notebooks remotely on CRC:

- ◆ <https://hub.crc.pitt.edu/>
- ◆ Make sure to choose "**SMP -1 core, 3 hours**" for spawner! →
- ◆ Like Colab, but integrated with CRC filesystem
- ◆ Can write and run notebooks on here, then export to Python scripts to submit as Slurm job
 - ◆ `jupyter nbconvert --to script your_notebook.ipynb`
--> produces **your_notebook.py**

▶ Colab:

- ◆ Cloud service, accessible anywhere
- ◆ Good for external collaboration, Google account and GitHub integration

▶ CRC Hub:

- ◆ Accessible with Pitt VPN
- ◆ More customizable, tailored to Pitt research needs

Select a job profile:

Host Process

Host Process

SMP - 1 core, 3 hours

GTX 1080 - 1 gpu, 3 hours

Wrapping up

▶ To-do #13

- ◆ Due on Tue, another "visit your classmates" day

▶ Homework 4

- ◆ Fun with CRC!
- ◆ Due on Thursday, but you want to start early.
- ◆ Make sure to check out the list of tips at the bottom of the page...
- ◆ As a class we have 10000 Service Units, which is 10k hours of computing time (not that much).
 - ◆ Don't let your jobs run infinitely! Test on small samples of data and make sure they work before you run large jobs.

▶ Next class

- ◆ Word embeddings, clustering