Analysis of Next Generation Sequencing Data
Introduction to Unix & Getting to know our systems

Luce Skrabanek

7 January, 2020

1 Network Access

If your laptop is not tagged, you can use the following credentials to access the Wifi network:

```
WiFi SSID: WCMEVENT
UserID:    HPCCLASS
Password:  HPCCLASS2020
```

2 Accessing SCU resources

2.1 Login nodes

The servers that we will be using for this class are hosted and maintained by the Scientific Computing Unit (`https://scu.med.cornell.edu`).

We first log in to a gateway node using `ssh`. These login nodes are monitored by an intrusion prevention software, and users can be temporarily locked out of their accounts if they fail to use the correct password, or permanently if they use an invalid username during `ssh` login.

To avoid the permanent ban, the following can be added to your `~/.ssh/config` file:

```
Host *.med.cornell.edu
user USERNAME
ServerAliveInterval 60
```

If you used the above config, you can login using:

```
ssh aristotle.med.cornell.edu
```

otherwise use:

```
ssh USERNAME@aristotle.med.cornell.edu
```

There are three login nodes available; you may use any of them.

`aristotle.med.cornell.edu`
`pascal.med.cornell.edu`
`aphrodite.med.cornell.edu`

You should never run any jobs on any of the login nodes.

If there are any issues with your password, go to `https://scu.med.cornell.edu/sspr` to reset it.

3 Introduction to Unix

UNIX is the dominant operating system for high-performance, scientific computing. UNIX was originally developed at AT&T Bell Labs in 1969. There are now many flavors of UNIX. Some common aliases are: IRIX, Solaris, AIX, HP-UX, BSD, Linux (Red Hat, SUSE, Debian, Mandrake, etc), Mac OS X. There are a couple of different UNIX shells. They all fall into two families, the Bourne shell family and the C shell family. Whenever possible, use `bash` or `tcsh`. These are the latest and greatest members of the two families; others are lesser forms of these. The `tcsh` shell has historically been preferred by scientists. The `bash` shell is preferred by computer geeks. Traditionally, scripts are written in `sh` or `bash`. We will be using bash in this class.

3.1 Looking at files (`ls, cat, more, less, head, tail`)

In Unix, (almost) everything is a file. Use the `ls` command to see what files you have.

```
ls
```

Most common UNIX commands are usually a just few letters long. This savz kystrks. Although this makes UNIX appear cryptic and a little harder to learn, you will find you'll get to know these commands quickly. In the long run, you will be much more efficient.

The `cat` command will show you the contents of a file.

```
cat slurm/hello_slurm.bash
```

Note that case matters in UNIX.

When you're typing commands, you can use the following:

1. The left and right arrow keys move the cursor left and right within a command.
2. The up and down arrow keys scroll forward and backward in your history of commands. This is useful if you need to type a command that is similar to one you previously ran.
3. [TAB] autocompletes a (partial) unambiguous filename.
4. Double [TAB] lists the files that share an ambiguous prefix.
5. CTRL-A moves to the beginning of the line.
6. CTRL-E moves to the end of the line.
7. CTRL-D deletes the character that the cursor is on.
8. [Delete] works as expected.
9. CTRL-C abandons the whole affair and lets you try again.

© *Copyright 2020 Luce Skrabanek, Weill Cornell Medicine*

You can look at other people's files.

```
cat /home/luce/angsd/quotation
cat ~luce/angsd/quotation
```

But this is not useful when viewing very big files.

```
cat ~luce/angsd/demo-data/demo.fastq
```

For large files, we can use `more`. With `more`, to advance a line at a time, press `[Return]`. To advance a screenful, press the space bar. To quit the pager, press `q`. Some UNIX systems have a better pager called `less` (because "less is more") which lets you scroll up too, but it is not on all UNIX systems.

```
more ~luce/angsd/demo-data/demo.fastq
less ~luce/angsd/demo-data/demo.fastq
```

We can also use the commands `head` and `tail` to show just a few lines at the beginning or end of the file.

```
head -n 10 ~luce/angsd/demo.fastq
tail -n 25 ~luce/angsd/demo.fastq
```

3.2 Directories (`pwd`, `cd`, relative and absolute pathnames)

Directories are hierarchical, similar to Mac or PC files and folders. In UNIX, the `/` separates the folder names from the file name. There are no drive letters of volume names. The root directory is just `/` and the current directory is `.` and the parent directory is `..` (two periods).

To find out what your current directory is, use the `pwd` command.

To change your current working directory, use the `cd` command. `cd` by itself is shorthand for going directly to your home directory.

```
# by itself, cd is shorthand for going directly to your home directory.
cd
# go to my home directory
cd /home/luce
# from there, go to my ANGSD directory
cd angsd
# shorthand to go back to your home directory
cd ~
# go to my ANGSD directory from anywhere
cd ~luce/angsd
```

3.3 Manipulating Files and Directories (`cp`, `mkdir`, `mv`)

We can now see what files we have and look at their contents. Next we will learn how to move files around in the directory structure. Moving files around is equivalent to dragging and dropping files

© *Copyright 2020 Luce Skrabanek, Weill Cornell Medicine*

on your desktop computer. However, since we can't use a mouse with UNIX, we need some new commands.

To copy a file, use `cp`.

```
# To copy the demo.fastq file from my account
cp ~luce/angsd/demo-data/demo.fastq .
```

Make sure you're in your home directory before you do this. The period indicates that the file will be copied with the same name to the current directory.

Let's create a directory to put our newly acquired file into. We do this with the `mkdir` and `mv` commands.

```
mkdir data
mv demo.fastq data
```

We can also rename files using the `mv` command.

```
mv demo.fastq human_expt.fastq
```

What do you need to do before you can execute this command? [HINT: what's your pwd; where's your file?] Can you think of a way in which you could have renamed the file without `cd`'ing into the directory?

```
mv data/demo.fastq data/human_expt.fastq
mv data/demo.fastq human_expt.fastq
```

Note that the second command would have renamed the file and moved it up into the current directory.

```
# To copy the demo.fastq file from my account, and re-name it
cp ~luce/angsd/demo.fastq data/new_demo.fastq
```

You can also move multiple files at the same time, but note that when you are moving files this way, the destination must be a directory, and you can't rename files while you're moving them.

You can also move and rename directories with the same commands.

3.4 Deleting Files and Permissions (`rm`, `rmdir`, `chmod`)

You can remove files using the `rm` command. There is NO UNDO. If you remove a file, you're not getting it back.

To delete a directory, use the `rmdir` command. Note that the directory must first be empty.

We saw before that UNIX is a multi-user operating system. There needs to be a mechanism in place to ensure that you don't corrupt (or delete) other people's files (even if you can see and read them), and that other people don't corrupt (or delete) yours. To do this, each file and directory has a series of permissions associated with it. This tells the system which people can and can't read, write and execute those files.

© *Copyright 2020 Luce Skrabanek, Weill Cornell Medicine*

We've already learnt about the `ls` command. The `ls` command has a -l option which shows you a lot of information about the files and directories, including the permissions associated with them. Let's try to understand the output from `ls -l`:

```
drwxr-xr-x  4 luce abc   117 Feb  5  2019 demo-data/
-rw-r--r--  1 luce abc 11476 Jan 15  2019 gene_counts.txt
drwxr-xr-x 10 luce abc  4096 Feb  8  2019 gierlinski/
-rw-r--r--  1 luce abc  2019 Jan 22  2019 markdown_quick_reference
-rw-r--r--  1 luce abc   132 Jan 11  2019 quotation
-rw-r--r--  1 luce abc    81 Jan 11  2019 Quotation
drwxr-xr-x  4 luce abc   147 Feb  4  2019 referenceGenomes/
drwxr-xr-x  2 luce abc   106 Jan  6 13:20 slurm/
-rw-r--r--  1 luce abc   780 Jan  6 12:58 spack.config
```

The last column is the filename or directory name, which is what appears when you do an ordinary `ls` command. Preceding that, is the date and time at which that file or directory was last modified. Continuing to work backwards, we see the size of the file or directory in bytes. The next column is the group owner (we'll say more about this in a moment). Then we have the username of the owner of the file. If you own the file, this will be your username.

At the beginning of the line are a series of 10 letters or hyphens which show the filetype and permissions. The first character gives the file type: directories are shown with a `d` and regular files with a hyphen. The remaining letters are `r`, `w`, and `x` and represent 'read', 'write' and 'execute' permissions, respectively. There are three sets of these letters:

1. The first set shows the permissions that the owner himself has for this file.
2. The second set indicates the permissions for the group of users that the user belongs to.
3. The third set shows the permissions for all other users who have accounts on the system.

You can specify who can read, write or execute your files by using the `chmod` command. The user is referred to by `u`, the group by `g` and all other users by `o`. You can add, or take away, permissions using `+` and `-` so long as you are the owner of the file.

```
# Make your slurm batch file executable by you only
chmod go-x hello_slurm.bash
# Make the same file readable and writable by everybody.
chmod go+rw hello_slurm.bash
```

Note that the system administrator can read anybody's file so this is not a solution to true privacy.

3.5 Piping commands

The `wc` command (word count) counts lines, words, and characters in a file.

© *Copyright 2020 Luce Skrabanek, Weill Cornell Medicine*

```
cat ~luce/angsd/quotation

THERE IS NOTHING NOBLE IN BEING SUPERIOR TO SOME OTHER
MAN. TRUE NOBILITY IS BEING SUPERIOR TO YOUR FORMER SELF.
  -- HINDU PROVERB

wc quotation
3 23  132 quotation
```

This reports 3 lines, 23 "words", and 132 characters in this proverb. Count the words yourself. Did you get 23? Why do you think the `wc` command reports 23 words? What is the definition of a word that the `wc` command uses?

To learn more about a command, use the `man` command.

```
man wc
```

This tells you everything you wanted to know about a command, and a lot more besides. The 'man pages' can be a bit dry, but are usually very precise.

When the output of the man command is longer than a screenful, it is "piped" into the `less` command, so you can view it one page at a time. This idea of "piping" the output of one command into the input of another is a key concept in getting the most out of UNIX.

The `man` command pipes its output to the `less` command silently, but usually we need to specify this behavior explicitly. This is done with the `|` operator, called the "pipe" character.

We can use pipes to figure out how many words there are in the first 25 lines of the markdown quick reference guide.

```
# Step 1: What are the first 25 lines of the reference guide?
head -n 25 ~luce/angsd/markdown_quick_reference
# Step 2: How many words are in the result of the previous command?
wc -w
# Steps 1 & 2 together!!!
head -n 25 ~luce/angsd/markdown_quick_reference | wc -w
```

The more commands you know, the better off you are. Your capabilities grow exponentially as does your ability to combine the commands you know.

3.6 Manipulating Data (`cut`, `paste`, `join`, `sort`, `uniq`, `wget`)

The `cut` command is used to extract selected columns or fields from a file.

When processing by field, lines that do not contain any field delimiters will be passed though to `stdout` untouched. This behavior can be overridden by using the `-s` option, which suppresses those lines.

```
# Example: Extract the Geneid and gene counts from gene_counts_long.txt
# (found in /home/luce/angsd/demo-data/gene-counts-demo)
cat gene_counts_long.txt | cut -f 1,7-12 > gene_counts.txt
cat gene_counts_long.txt | cut --complement -f 2-6 | head
```

© *Copyright 2020 Luce Skrabanek, Weill Cornell Medicine*

The `paste` command merges sequentially corresponding lines from different files. Note that this should not be used to merge files on a common field. The `join` command can be used for this purpose.

```
# Example: print the gene counts from Sample_7 beside the counts from
    Samples 1-6
paste gene_counts.txt new_sample.counts | head
join gene_counts.txt new_sample.counts | head
```

The `sort` command sorts files. Some of the commonly used options for `sort` include:

1. To specify which field to sort by, use the `-k` option.
2. `sort` assumes that fields are separated by whitespace. You can change the field delimiter with the `-t` option.
3. To sort in reverse order, use the `-r` option.
4. To sort in numerical order, use the `-n` option.

```
# Example: sort all the files in your directory by size:
ls -s | sort -n

# Example: sort the gene counts by the number of reads in Sample_1,
    highest to lowest
sort -rn -k 2 gene_counts.txt | head
```

Another interesting command to know about is the `uniq` command. It helps you find unique lines of fields. You might use the `uniq` command, for example, to list out the different kinds of keywords that a PDB file can begin with, or to list the types of amino acids that appear in such a protein structure file.

Commonly, you will have to download data from the web and manipulate it. To download web content, use the `wget` command.

```
# Example: download the p53 protein from Unigene
wget https://www.uniprot.org/uniprot/P04637.fasta
```

3.7 SSH keys

To effectively use the SCU's infrastructure, you will need to ensure that all the compute nodes have the ability to authenticate you, without the need to send your password over the network.

Setting up your ssh keys only needs to be done once, so first check if you don't already have ssh keys set up:

```
ls ~/.ssh
```

If the output shows the files `id_rsa` and `id_rsa.pub`, you already have keys in place (your private and public keys, respectively). Skip the following command and continue to authorizing your key.

If the output did not show those files, generate them with the following command:

© *Copyright 2020 Luce Skrabanek, Weill Cornell Medicine*

```
ssh-keygen -t rsa
```

Follow the instructions on screen, and accept the default location.

To authorize your key, add your public key to the list of public keys that can authenticate you.

```
cd ~/.ssh
cat id_rsa.pub >> authorized_keys
```

4 Storage space

You have several disk storage locations available to you:

1. Your home directory. This is 100 GB of backed-up storage space. In general, this should be used to store your scripts, any tools or packages that you install yourself, and anything else that you would be very sad to lose. Note that this space is limited, so use it wisely!
2. Scratch space on buddy/farina. We have 2.4 TB of local storage space here, shared between all users. This is accessible via `/scratchLocal`. This should be used for I/O-intensive operations, especially when running batch jobs.
3. Scratch space on athena. We have available a 3TB fileset on `/athena/angsd/scratch`, also shared between all users. This should be used to store datasets and other result files.

4.1 Batch jobs

You should never run any jobs on any of the login nodes.

There are two ways to run jobs on this infrastructure:

1. Request an interactive session.
2. Use the batch queuing system. This should always be the primary choice when running jobs that will take a while.

There are two compute nodes that have been reserved for this class:

`buddy.pbtech`
`farina.pbtech`

Slurm is used as the batch queuing system on the SCU infrastructure. To access the compute nodes reserved for this class, either via the batch queuing system, or via an interactive session, we must first log in to the submit host: `curie.pbtech`.

```
ssh curie.pbtech
```

There is a dedicated queue for this class, called `angsd_class`. To see basic information about this queue:

```
sinfo -p angsd_class
```

4.2 Interactive sessions

This is especially useful if you are debugging and testing code, or if you have a relatively simple work flow. In these cases, we can launch an interactive session.

```
srun -n1 --pty --partition=angsd_class --mem=8G bash -i
```

Explanations for the options in the above example:

- `-n 1, --ntasks=1` The number of concurrently running tasks.
- `--pty` Runs the job in a pseudo-terminal.
- `--partition=angsd_class` Specify the cluster partition you want to access. For this class, we have a dedicated partition called `angsd_class`.
- `--mem=8G` Requests 8G of memory for the job. If you use more memory than what you requested, the job will fail.
- `bash -i` The command to be run. In this case, we are running bash with an interactive terminal.

Since we can log directly to buddy and farina, we do not have to use this approach in this class, but this will be how to start an interactive session on most systems.

Make sure that you log out of an interactive session once you are done with it. Otherwise, you are tieing up valuable resources that could be used by somebody else.

4.3 Batch queuing

To submit jobs to the cluster, you also need to be logged in `curie.pbtech`.

hello_slurm.bash

```
#! /bin/bash -l

#SBATCH --partition=angsd_class
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --job-name=hi_slurm
#SBATCH --time=00:05:00   # HH/MM/SS
#SBATCH --mem=1G          # memory requested, units available: K,M,G,T

echo "Starting at:" `date` >> hello_slurm_output.txt
echo "This is job #:" $SLURM_JOB_ID >> hello_slurm_output.txt
echo "Running on node:" `hostname` >> hello_slurm_output.txt
echo "Running on cluster:" $SLURM_CLUSTER_NAME >> hello_slurm_output.
   txt
echo "This job was assigned the temporary (local) directory:" $TMPDIR
   >> hello_slurm_output.txt
touch ${TMPDIR}/test_file_$SLURM_JOB_ID

sleep 30
exit
```

© *Copyright 2020 Luce Skrabanek, Weill Cornell Medicine*

The shebang line (the first line in the script) indicates the shell that we want to use to run the job. This line is absolutely required. In almost all cases, you will want to use a bash shell, and in most cases, that shell should be a login shell. This will ensure that your environment is consistent. The login shell option is specified with the `-l` flag at the end of the shebang line.

Between the shebang line and the main body of the script (which looks like any other bash script), are a set of lines that begin with `#$`. These are the slurm options that will be passed on to the scheduler. Some of them we have already seen in the interactive command.

- `--nodes=1` The number of nodes requested.
- `--job-name=hi_slurm` The name of the job. This will be used when listing the job in the queue. The name should be kept relatively short, as only the first 10 characters will be shown in many cases. Note that the name of the job is not the name of the script.
- `--time=00:05:00` The maximum amount of time requested for this job to run.
- `--mem=1G` Requests 1G of memory for the job. If you use more memory than what you requested, the job will fail.

Additionally, we are using some slurm-specific environmental variables:

- `$SLURM_JOB_ID` The job's ID number, assigned by slurm.
- `$SLURM_CLUSTER_NAME` The name of the cluster used.
- `$TMPDIR` The local temporary directory your job has access to. It's very important for your job performance, as well as cluster stability, that intense I/O (e.g. creating many temporary files) is performed in this temporary directory

To submit this job to the queue:

```
sbatch hello_slurm.bash
```

To monitor job status:

```
squeue -u USERNAME
```

More information about slurm can be found on the SCU wiki at
`https://wcmscu.atlassian.net/wiki/spaces/WIKI/pages/327731/Using+Slurm`
and at
`https://slurm.schedmd.com/pdfs/summary.pdf`

5 Installed tools

Many tools that we will be using in this class have already been installed for you by the SCU. These are accessed via `spack`. To see a list of all packages that are available on our systems, use:

```
spack find
```

Note that this command will not work on a gateway node, as those nodes are not meant to run any tools.

If the `spack find` command does not work, you will need to append a spack-specific config to your `~/.bash_profile`.

```
cat ~luce/angsd/spack.config >> ~/.bash_profile
```

Make sure that you can access and use some of the tools that we will be using more frequently. Note that some tools have multiple versions available, and you will have to be explicit about which version to use. Also note that some tools need a specific environment in which to work (e.g., R, Python, Java). Use the `-r` option to ensure that the appropriate environment is also loaded.

```
spack load -r fastqc
  fastqc --help
spack load subread
  featureCounts --help
spack load -r py-rseqc
  read_duplication.py --help
spack load samtools@1.9%gcc@6.3.0
  samtools --help
spack load star@2.7.0e
  STAR --help
spack load bedtools2@2.28.0
  bedtools --help
spack load -r py-deeptools@3.2.1%gcc@6.3.0
  deeptools --help
spack load bedops
  bedops --help
spack load bamtools
    bamtools --help
spack load -r py-macs2
  macs2 --help
spack load -r r@3.6.0
  R --version
```

© Copyright 2020 Luce Skrabanek, Weill Cornell Medicine