An incomplete guide to high-performance computing on Oscar (largely taken from Brown CCV user manual – read that for more information!)

Brown CCV resources:

Overview of Oscar:

https://web1.ccv.brown.edu/technologies/computing

User manual:

https://web1.ccv.brown.edu/doc

(click "getting started" if you have trouble logging in)

Oscar help:

support@ccv.brown.edu

CCV walk-in office hours

bash man pages

`man <command>`; e.g. `man chmod`; press "q" to exit

Get yourself familiar with command line computing.

Linux/bash tutorials available online, e.g. https://ryanstutorials.net/linuxtutorial/

Practical computing for biologists by Haddock and Dunn

how are computing clusters organized? head/login nodes vs. compute nodes; SLURM job scheduler

never run jobs directly on head node: always submit using SLURM

Logging into the head nodes & orienting yourself:

```
ssh <username>@ssh.ccv.brown.edu
```

where am I? ("print working directory")

```
pwd
```

what is here? (list files/directories in list format with human-readable filesizes)

```
ls -lh
```

file/directory ownership and permissions

to change permissions (see man pages for use): `chmod`

Oscar file management: where to put scripts, data, etc

IMPORTANT: Every lab has its own conventions for data storage. This is a combination of my own strategy & general good practices. Make sure you know what your lab's policy is, especially for long-term storage of large datafiles, permissions, etc.

See how much space you're using and how much you have available:

```
myquota
```

1. home directory: /users/<username>

```
cd ~
cd /users/<username>
```

Limited space (10GB). I use this to store SLURM job submission scripts (batch scripts).
Helpful to maintain a directory structure that mirrors the data directory

2. data directory: /users/<username>/data
   ```
   cd ~/data
   cd /users/<username>/data
   ```
   Optimized for large files. Store big data files here.
   Helpful to maintain a directory structure here that mirrors home directory so it's easy to find submit scripts that go with your input/output files.
   Don't fill up with large temporary files (use scratch)!
   I also keep locally installed programs and scripts here in ~/data/bin (but see notes below about installing programs yourself)
3. scratch: /users/<username>/scratch
   ```
   cd ~/scratch
   cd /users/<username>/scratch
   ```
   Optimized for large files but NOT BACKED UP AND PURGED EVERY 30 DAYS
   Do not use for long-term storage of anything important! Scratch is good for large, temporary files (e.g. files produced during a job run); you can always move files from ~/scratch to ~/data when your job is finished

Oscar has 2 "transfer" nodes (transfer3, transfer4) that are more efficient than the login nodes for uploading large files. Transferring files from a Mac or Linux machine to Oscar (all on one line, make sure you *are not logged into Oscar*):

```
scp /path/to/source/file
<username>@transfer3.ccv.brown.edu:/path/to/destination/file
```

To transfer files from Oscar to your computer:

```
scp <username>@transfer3.ccv.brown.edu:/path/to/source/file
/path/to/destination/file
```

Modules

CCV staff install and maintain a lot of different software packages on Oscar and will install new software upon request. *Do not try installing software yourself without checking in with them first!* Exceptions for R and Python packages, see:
https://web1.ccv.brown.edu/doc7/python-installs
https://web1.ccv.brown.edu/r-installs

Most programs are not immediately available when you log in. They are accessible as "modules" that can be loaded and unloaded into your environment (e.g. the settings of your current Oscar session). More info here:
https://web1.ccv.brown.edu/doc7/software

To see all programs available:

```
module avail
```

To search for a particular program:

```
module avail <searchterm>
```

e.g. `module avail samtools`

To load a module to make a program available (if there are multiple versions available, it will load the default – usually the most recent version):

```
module load samtools
module load samtools/0.1.18
```

To unload a module (program will no longer be in your environment):

```
module unload <module>
```

Running jobs using SLURM (see https://web1.ccv.brown.edu/doc7/jobs)

Most of the time you will submit a "batch" job, where you execute a short script to tell the cluster how to run your program. Once submitted your job will go into the queue until resources are available on the compute nodes for it to start. You can log off of Oscar at any time after submitting without affecting your job status.

Batch scripts follow a straightforward formula where you

1) specify job parameters like number of nodes, memory, and runtime;
2) then specify the commands you want to execute on the compute node

A basic script that I use to run the program Stacks on RADseq data is below. Note that the first line must be `#!/bin/bash`; that SLURM job parameters are notated by `#SBATCH`; and that you must remember to load any necessary modules and navigate to the directories where your files are stored (or provide paths to files)!

```
#!/bin/bash

#SBATCH -n 16
#SBATCH -t 3:00:00
#SBATCH --mem=20G

module load stacks

cd /users/rkartzin/data/rkartzin/ambr_GBS/

ref_map.pl -m 4 -T 16 -S -b 1 -X "populations:-r .80" -X
"populations:--vcf"  -X "populations:--fstats" --samples
mapped_bams -O popMap.txt -o stacks_m4
```

See for online Oscar manual for a longer list of job parameters.

Make sure to tune your SBATCH parameters to the job you are running. This way you will avoid unnecessarily tying up resources that could be dedicated to other jobs. Your wait time in the queue will also be shorter if you request fewer resources or shorter runtimes; but be conservative in your estimates of how much you need! If you run out of memory or time, your job will be "killed" prematurely and you'll need to re-submit.

To submit a job to the queue:

```
sbatch <jobscript>
```

To check on your jobs in the queue:

```
squeue -u <username>
```

To cancel a running job:

```
scancel <jobid>
```

To see memory usage of completed jobs:

```
myjobinfo -j <jobid>
```

You can also run commands directly without submitting a batch job on compute nodes in an "interactive" session, which can (for example) be helpful to avoid waiting in the queue for troubleshooting scripts. You still specify resources that you need but you will be logged in directly to the compute node.

```
interact -n 1 -t 01:00:00 -m 1g
```

This requests 1 node for 1 hour and 1GB memory. Unlike batch jobs, anything you are running in an interactive job terminates when you log out of the compute node (`exit`).