A tutorial on using the HPC to assemble and analyze bacterial genomes

tgallagh@uci.edu

Resources

- Google is your best friend.
- In person classes offered by UCI Data Science Initiative: http://datascience.uci.edu/education/short-courses/
- Forum and discussion boards: https://stackoverflow.com/
- Dr. Rob Edwards metagenomics workshop manual: https://edwards.sdsu.edu/SDSU2017/WorkshopManual.pdf
- Edwards lab blog: https://edwards.sdsu.edu/research/category/lab-blog/
- Beginners guide to comparative bacterial genome analysis: https://microbialinformaticsj.biomedcentral.com/articles/10.1186/2042-5783-3-2
- Dr. Kevin Thornton github for advanced informatics class: https://github.com/ThorntonLab/AdvancedInformatics2017

Introduction to Linux

Written by Rob Edwards, link to metagenomics workshop manual: https://edwards.sdsu.edu/SDSU2017/WorkshopManual.pdf

Most computational biologists use either Apple Mac's or Linux machines. There are a couple of reasons for this:

Much of the software is free

Many of the tools require a command line to use

Many of the tools require large compute resources, either a cluster or lots of memory   It probably the last reason that dominates many of the discussions: many of the software tools you will want to use require more resources than your laptop or desktop machine can provide. You can definitely run the BLAST suite of programs on your laptop, but for large data sets like a whole metagenome, you probably want to run that on a server so that you can go and do other things while the program is running.   Most servers that you will access run the operating system linux, or a similar variant of the Unix operating system. Linux was started in 1991 as a PhD project by Linus Torvalds, and the initial release was not very powerful, however Linus released his software using an open source license that allows anyone to use and modify the source code, either commercially or not. This meant that other developers could pick up on his work, and several groups and companies developed operating systems based on the linux kernel (the piece that does all the work). RedHat has had the most success in the so-called enterprise sphere (selling servers to companies), and because RedHat's Fedora software is built on open-source software, it also releases all of its changes and updates.

> There is a free version of RedHat's operating system, called CentOS, that many servers use (especially those in academic settings!). CentOS is designed with performance in mind, and is designed to be run on servers that are very rarely rebooted. Therefore, only essential system changes are made between releases of the operating system. Another flavor of Linux is called Debian (after founder Ian Murdock and his then-girlfriend Debra) developed some elegant mechanisms to keep the software up-to-date. Again, Debian was built on free software, and a company (Canonical) built a smooth graphical user interface, and easy, graphical installation system, and released the software as Ubuntu. There are many   connections between Ubuntu, Debian, CentOS, and Fedora, and most users can't really tell which operating system they are using – they are all canonically called Linux!

The reason that we are using Linux for the course is because it is open and free, there is lots of software available for it, and it will help you become familiar with the servers you will doubtless use for high- performance computing. In addition, we can provide you with a server that you can start and it already has everything installed for you!

Communicating with a linux server

MacOS/Linux: Open a terminal app

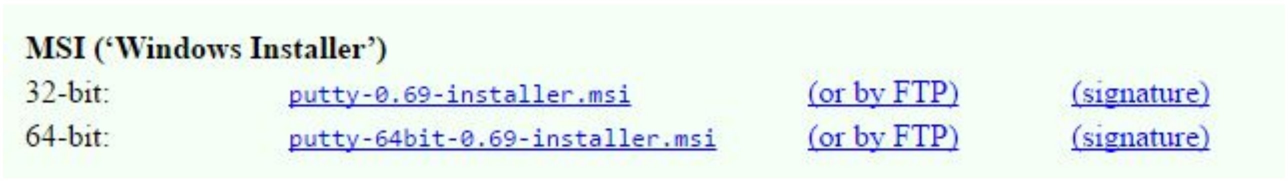
Windows

Written by Rob Edwards, link to metagenomics workshop manual: https://edwards.sdsu.edu/SDSU2017/WorkshopManual.pdf

Download PuTTY. **Note:** For PuTTY, there are some hacked versions of PuTTY floating around the internet and you don't want to inadvertently download one of those as it may give the bad guys access to your machine. There are also versions with advertisements, and other crap added on. I *always* start at https://www.putty.org/ and go from there. Do not Google for it, just go direct to the website and click the download link.

You want either the 32-bit or 64-bit MSI Windows Installer, depending on whether your computer has 32- or 64-bits.

Note sure how many bits? This is from the PuTTY FAQ:



MSI ('Windows Installer')

32-bit:	putty-0.69-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.69-installer.msi	(or by FTP)	(signature)

A.6.10 Should I run the 32-bit or the 64-bit version?

If you're not sure, the 32-bit version is generally the safe option. It will run perfectly well on all processors and on all versions of Windows that PuTTY supports. PuTTY doesn't require to run as a 64-bit application to work well, and having a 32-bit PuTTY on a 64-bit system isn't likely to cause you any trouble.

The 64-bit version (first released in 0.68) will only run if you have a 64-bit processor and a 64-bit edition of Windows (both of these things are likely to be true of any recent Windows PC). It will run somewhat faster (in particular, the cryptography will be faster, especially during link setup), but it will consume slightly more memory.

Using Linux

Written by Rob Edwards, link to metagenomics workshop manual: https://edwards.sdsu.edu/SDSU2017/WorkshopManual.pdf

Linux (especially the Ubuntu variety that we are using here) has many of the same tools that you already use in MacOS or Windows. For example, there is an office suite of applications including LibreOffice Writer (a replacement for MS Word), LibreOffice Calc (a replacement for MS Excel), and LibreOffice Impress (a replacement for MS PowerPoint). There are calculators and games, and the virtual box has both Google Chrome and Firefox installed as web browsers.

However, the real strength in using Linux comes from using the command line, and here we'll take a tour around the command line to get you started – or to act as a refresher.

To open the command line, click on the terminal application icon. This will open a blank screen and situate you in your home directory.

To move around in the command line, you type commands and press return. There is a cheat-sheet of common commands on the next page.

Here are some exercises that we will try:
- List the files in the directory.
- Copy one file to another file
- Look at the content of one file from the top
- Look at the content of one file from the bottom
- Check if two files are identical.
- Delete a file
- Make a test directory
- Move some files to your new directory
- Remove test directory (and the files)
- Search for a word in a file

Unix cheat sheet

cd *directory [.. - ~]*	`Change` *directory [to the parent directory, the last directory you were in,*

	or your home directory]
mkdir *directory*	Make *directory*
mv *filepath1 filepath2*	Move file from *filepath1* to location at *filepath2*
cp *filepath1 filepath2*	Copy file from *filepath1* to location at *filepath2*
pwd	Show current directory
ls *directory*	List contents of *directory* (leave *directory* blank for current)
ls -l *directory*	List contents of *directory* in long-listing format (leave directory blank for current)
cat *file*	*Print the entire contents of a file*
head *file*	Print out first 10 lines of *file*
tail *file*	Print out last 10 lines of *file*
less *file*	View file in Terminal. Use arrows to browse. Press ***q*** to quit.
df -h	Show disk space usage information
wc -l *file*	Display the number of lines in the *file*.
diff *file1 file2*	Display the differences between *file1* and *file2*
grep *word file*	Search for *word* in *file*
sort	*Sort the input*
uniq *file*	*Look for duplicate words in the file*
cut -f 1 *file*	*Split the contents of a file, e.g. if it is in columns*
xargs -n 1	*Use each entry as a parameter to pass to command*

command	

- Make five new directories one inside the other like level1/level2/level3/level4/level5 in one command.
- Move a file to the innermost directory.
- Copy a file to your home directory

Pipes

Written by Rob Edwards, link to metagenomics workshop manual: https://edwards.sdsu.edu/SDSU2017/WorkshopManual.pdf

One of the strengths of unix is the ability to pipe commands together. You can take the output of one command, and use it as the input to another command. This means that you can join arbitrary things together to generate new results. This is one of the reasons that unix is so popular among data analysts. The method for doing this is called piping, and we use the vertical line symbol to pipe the input "|".

For example, to sort the lines in a file we can combine two commands, `cat` and `sort.` The first command prints the entire contents of a file and the second sorts the lines on the input. For example, if we have a file called poem.txt, we can sort the lines using this command:

```
cat poem.txt | sort
```

You can sort things either alphabetically or numerically by adding a -n to the command line. There is another command, `uniq`, that looks for duplicated lines, and has an option -c to count the duplicates, and a command cut that allows you to print one column of a file. We can tie all these commands together to sort the contents of a file, count recurrent lines, and sort them numerically to generate a list of lines in the file. For example, if we have a text file containing subsystems in metagenomes, we can split the list into a specific column using a tab as a separator (the shortcut for a tab is '\t'), sort those lines, identify and count duplicates, and then sort the output numerically.

```
cut -f 2 -d$'\t' subsystems.txt | sort | uniq -c | sort -nr | less
```

stdout and stderr

Linux machines have two "output streams", places where they print things. One is designed for normal output and is called "stdout" (standard output), and the other is designed to print error messages and is called "stderr".

Normally, both of these print to the screen, however we can redirect one or both of them to print to a file.

To redirect stdout, we use a greater than sign (">"). For example, we can take the output from the command above and write it to the file subsystem_counts.txt:

```
cut -f 2 -d$'\t' subsystems.txt | sort | uniq -c | sort -nr >
subsystem_counts.txt
```

Warning: This will overwrite any existing file called subsystem_counts.txt and you will loose all the content.

Requesting an HPC Account

Instructions here: http://hpc-trac.oit.uci.edu/wiki/HowTo/Signup

Make sure to indicate you are part of Dr. Katrine Whiteson's group and a biological sciences student.

HPC Tutorial

Read through and follow tutorial here: https://hpc.oit.uci.edu/HPC_USER_HOWTO.html

Working on the HPC

Read through: https://github.com/ThorntonLab/biocluster

After reading through all the dense stuff, check out Prof. Thornton's HPC "cheat sheet": https://github.com/ThorntonLab/AdvancedInformatics2017/blob/master/materials/HPC.md

Other things to do before getting started:

- Download and install filezilla: https://filezilla-project.org/
  - To connect with the HPC through the filezilla client:
    - Host = hpc.oit.uci.edu
    - username = your HPC username
    - pw = your HPC pw
    - port = 22

Starting the *P. aeruginosa* Bacterial Genomes Analysis

- ### Bioinformatics Tip 1: Keep your notes and data organized!

There are multiple electronic notebooks you can use to save notes and scripts you write for bioinformatics project. I like github. You don't need to worry about learning github for now, but if you continue doing bioinformatics analyses, I would recommend learning it. Here is a great tutorial on using github from Prof. Kevin Thornton:

https://github.com/ThorntonLab/intro2github

In addition to documenting what you are doing with an electronic notebook, you want to make sure you are well-organized for a number of reasons:

> (1) In this day and age, bioinformatics data is becoming mainstream in biology journals. If you publish a bioinformatics project, the journal will most likely ask for scripts or code you used for your project. Having a well-organized project directory will make it much easier to find and share these scripts.
>
> (2) To make redoing an analysis or reusing scripts from a project easier!
>
> (3) so that your collaborator won't secretly hate you if you end up doing a collaborative bioinformatics project

So before we even get started on the *P. aeruginosa* project, we are going to make a nice parent directory for your project filled with empty subdirectories.

- **Make an empty, well-organized project directory using a pre-written bash script**

From here on out, I am assuming you read through the linux tutorial and are somewhat familiar with unix commands. If you have no idea what is going on, reread the linux chapter.

When you initially log into the HPC, you are connected to a "login node". The function of these nodes are to let people into the HPC, so avoid doing any computationally intensive work in them. Request a "compute" node with the command:

> *qrsh*

Then, do the following:

1. *cd* into my directory: /bio/tgallagh/GenomicsTutorial/code/scripts
2. *ls* to see what files are there
3. copy (*scp)* the "create_project.sh" file into your /bio/<USERNAME> directory

*Note, anytime you see "< >" this signifies user input

Next, we are going to run the "create_project.sh" script. This script was written by Prof. JJ Emerson, it basically sets up an empty shell of nicely named directories for the user.

After *cd*-ing into your /bio/<USERNAME> directory, if you enter "bash create_project.sh" into the commandline, the usage of this script should appear:

```
[tgallagh@compute-1-13 scripts]$ bash create_project.sh
```

```
usage: create_project.sh name path
```

We need to give the bash script a directory name and path to where you want the directory to be located. For example, if I want to make a new project directory named “Cheetos” in my /bio/tgallagh folder:

```
[tgallagh@compute-1-13 scripts]$ bash create_project.sh Cheetos
/bio/tgallagh

Project Cheetos created at /bio/tgallagh.
```

As you look throughout your new project directory and subdirectories, you’ll notice each subdirectory has an empty “README.md” files. These files are great for putting notes for yourself or potentially other users.

There are a couple of different linux text editors you can use to make edits to text files. I like using “vim” but it takes some getting used to… definitely not as intuitive as microsoft word, and you can only use your keyboard to edit and save! http://www.vim.org/about.php.

- **Copy the raw sequencing data into your directory**

Background information on the sequencing data:

This sequencing data you will be working with in this tutorial consists of DNA from 4 *Pseudomonas aeruginosa* strains. Specifically, the names of the strains are *P. aeruginosa* FLR01 (sometimes nicknamed “P1” in our lab) and 3 “substrains” of *P. aeruginosa* PA14 (all derived from different labs: Dorrestein, Siryaporn, Hochbaum). FLR01 was isolated from cystic fibrosis sputum. The other 3 substrains are basically the same lab strain of *P. aeruginosa* that has been passaged in other labs, and we want to see if there are any noticeable genome changes as these 3 substrains adapted to different lab environments.

To see the data, first “cd” into my directory:

cd /bio/tgallagh/GenomicsTutorial/data/raw

And then ls to see what is in there:

```
[tgallagh@compute-1-13 raw]$ ls

PA14_Dorrestein.read1.fastq.gz
PA14_Hochbaum.read2.fastq.gz
PAnmFLR01_S10_L001_R1_001.fastq
PA14_Dorrestein.read2.fastq.gz
PA14_Siryaporn.read1.fastq.gz
PAnmFLR01_S10_L001_R2_001.fastq
PA14_Hochbaum.read1.fastq.gz
PA14_Siryaporn.read2.fastq.gz
```

README.md

The most common sequencing file formats you will probably encounter are ".fastq" or ".fastq.gz" (compressed fastq) and ".fasta." Sometimes this file extensions are shortened to ".fq", ".fa", ".fna", etc. To read about fastq files: https://en.wikipedia.org/wiki/FASTQ_format

This sequencing data is illumina paired end sequencing data, so we have two reads per sequenced DNA fragment. Let's look at the first ten lines of one of the uncompressed files:

```
[tgallagh@compute-1-13 raw]$ head -10 PAnmFLR01_S10_L001_R1_001.fastq

@M00285:18:000000000-A611E:1:1101:9210:1117 1:N:0:10

CCACGAAGAACAGCATCGAGCCNAAGGTCTTGGCCAGGATGAATATATTGATGGAGCTGATGTGGAG
+
CCCCCGGGGGGFGGGGGCGGGG#:CFGFGGGGGGGGGGGGGGGAFGFGGGGGGGGGGGGGGGGG
@M00285:18:000000000-A611E:1:1101:19487:1117 1:N:0:10
GTACAAGGCGGTGCCCGACGCGCTGGTCTACATGCACCCGGAGGATGCGCGCCAGCTCAAGCTGCGCCGCGGCAGCGAG
GTCAAGGNGNNNNNNNNNNNNNNNNGANNNNCGCGCGCGGGTCGAGACCCGCGGGCGCAACAAGCCGCCCCAGGGGCTGG
TGTTCGTGCCGTTCTTCGACGCCAACAAGCTGATCACCAAGTACACCCTGGACGCCACCGACCCGATTTCCAAGCAGAC
CGACTACAAGAAGTTCGCCGTTCGCCACGAACTGCTCAACCTGGCCTGAAGAG
+
CCCCCGDGGGC@FEDFD@F@FCFEEGDGGFGGGDGCDFGGCFECCGGG7FFGGGGGFFEGGGGFG@FGGGGGEGGGG>F
FGGGGGF#:###############++#####::@<7C@B1><FGGDGCCFFFECCB88>EE*;65CEE58?E8E58C*A>
FEC5@EGCEGDCGEGGGF*;=8CEG58CCFFGGC9C*AFGF++0<?C*:*7CFDGG<DFDDDECD)05.?:4>*<5>?D
7>>D5BF6=*7<6**1.>D5471).)(,)40,(.6)).5(4(.).3581).60

@M00285:18:000000000-A611E:1:1101:9237:1117 1:N:0:10
GCCGAGGCCCAGGCCGAAGGTTNGGAGGCGCTGCACCGGCAACTCGCCGAGGTCGACCCGGAATCGGCTGCGCGTATCC
ACCNCAANNNNNNNNNNNNNNNNNNGGCNNNCGCTCGAGGTGTATCGCCTCGGCGGGGTGTCGATGAGCGACCTGCGTCG
CCGGCAAAGCGCTGAAAAGGCGGATTTTGCTGCGTCAGGCGGGAATCAATTGCCGTATACTGTCGCGCAGCTGGCGATT
GGTCCCGAGCGGCGCCAAGATTTGCAGACGCGGATTGCGCAGCGTTTTCGCCACGTGCCGGAA
```

Use the *scp* command to copy all 8 fastq files from my directory into your new project directory. I would recommend copying these new files into your "raw" data directory, e.g. /bio/<USER>/<PROJECTNAME>/data/raw

Some bioinformatics programs do not allow you to use compressed (.gz or .zip) files as input, so unzip the 6 "PA14" files:

> *gunzip PA14**
>
> Note: that asterisk is a wildcard. It basically tells linux to replace the asterisk with any character, number, whitespace, etc. So it lets us unzip all 6 files that begin with "PA14" at once. Be careful when you are using wildcards! You can easily make mistakes, like unintentionally delete certain files.

- **FASTQC to check quality of raw sequencing files**
- **A5 Assembly to make assembled genomes from the raw**

sequencing data

https://sourceforge.net/p/ngopt/wiki/A5PipelineREADME/

- **PATRIC to annotate genomes and compare genomes**

URL to PATRIC :https://www.patricbrc.org/
Make an account and upload your A5 assembled genomes

Tutorials:

(1) Annotating a genome:
https://www.patricbrc.org/public/pdfs/Workshop-Genome-Annotation.pdf
(2) Making genome groups:
https://www.patricbrc.org/public/pdfs/Workshop-Genome-Groups.pdf *add additional *Pseudomonas aeruginosa* strains from the PATRIC database (such as PAO1)
(3) Comparing genes in genomes from a group:
https://www.patricbrc.org/public/pdfs/Workshop-Proteome-Comparison.pdf

Appendix

- **Downloading files (such as sequencing data) from a http path**

(1) make a new directory in your computer that you want to save new files to
(2) cd into that directory
(3) use wget to copy files to that directory: *wget* <URL>