
FengLabWkshopMay2015

Release 1.0

May 03, 2017

Contents

1	Links	3
2	Contents	5
2.1	Getting started	5
2.2	Logging into your computer instance “in the cloud” (Windows version)	6
2.3	Logging into your computer instance “in the cloud” (NON-Windows version; for Mac and Linux)	11
2.4	Overview	12
2.5	Initial steps condensed for today	13
2.6	STEP# 1: Get proper resources for working on genome of interest	14
2.7	STEP #2: Acquire experimental data	15
2.8	Extract the Gal4 reads based on barcode	18
2.9	Mapping reads	19
2.10	Improving the mapping of the reads	21
2.11	Preliminary view of mappings	23
2.12	Peak Predictions with MACS	24
2.13	Peaks relative to known genomic features with CEAS	25
2.14	Optional: Look at reads and/or peaks in IGB or IGV or SeqMonk OR UCSC genome browser	26
2.15	Comparison to published data	26
2.16	Motif discovery with MEME	28
2.17	Sources	30
2.18	Going forward	31
2.19	Appendix: Help getting files on and off Amazon AWS EC2 instances	32
2.20	Appendix: Making Wiggle file Covering Genome	34
2.21	Appendix: Obtaining a BED of Telomere sequences	34
2.22	Appendix: Technical guide to the workshop	38
2.23	Appendix: Setting up a Mac for the workshop	39
2.24	Appendix: Setting up an Amzon EC2 instance for the workshop	46

Hands-on ChIP-Seq Analysis Workshop for Feng Lab Group meeting by Wayne Decatur
3:30am-5:20pm, May 14th, 2015.

This is the 3rd session in series; see [first](#) and [second session info here](#)

CHAPTER 1

Links

- [ChIP-Seq Analysis Workshop for Feng Lab Group meeting.](#)
- [Slides link.](#)
- [A Google Doc for sharing today](#)

Getting started

Using this Web site

Reload to get the latest links!

[A Google Doc for sharing today](#)

Preparation

You'll need to take a few minutes in advance and prepare the laptop you'll be bringing to the hands-on workshop.

For the day of the workshop, you will be provided with powerful virtual machines from Amazon Web Services with essentially all the special software installed, and so you'll need a means to communicate with your Amazon Web Services computer instance via SSH. For users of Windows operating systems, this means that you will need make sure you have installed the program PuTTY. (You can get PuTTY [here](#).) All other operating systems are usually capable out of the box using SSH in a program called Terminal.

A special key file is part of the way you'll identify yourself to the computer instance when you connect via SSH.

To help make the first few minutes of the workshop session go smoothly, complete these steps to prepare your system and yourself for connecting to Amazon Web Services.

1. Two key files will be provided by email. Download the two key files to the computer you will bring the day of the workshop.
2. Move both of the files to your Desktop. **OPTIONAL:** You'll actually only need one of these files. If you'd like, you can prepare for the workshop by identifying the appropriate key file you'll need for your system and deleting the one you will not need. Those with Windows operating systems, will need the `workshop.ppk` file; all others will need the `workshop.pem` file.
3. Before lab meeting take a few minutes to identify and review the login instructions specific to your computer in order to get familiar with the process of logging into Amazon Web Services with SSH on your operating system. The guides to connecting for each type of computer follow this page. You will not yet have the addresses of the

virtual machines, and so you will not be able to complete the steps until the day of the workshop. For Windows users, make sure you have installed the program PuTTY.

Be sure you have a modern, updated browser on your system. Preferably Chrome or Firefox.

Logging into your computer instance “in the cloud” (Windows version)

In an effort to make things go more smoothly today, I eliminated a step that users on Windows operating system usually have to perform. If you ever find yourself connecting to an Amazon EC2 instance on Windows and need to generate a ppk file from your pem file, see [here](#)

There are a few images here to guide you because I do not plan to demonstrate the Windows operating system route during the workshop.

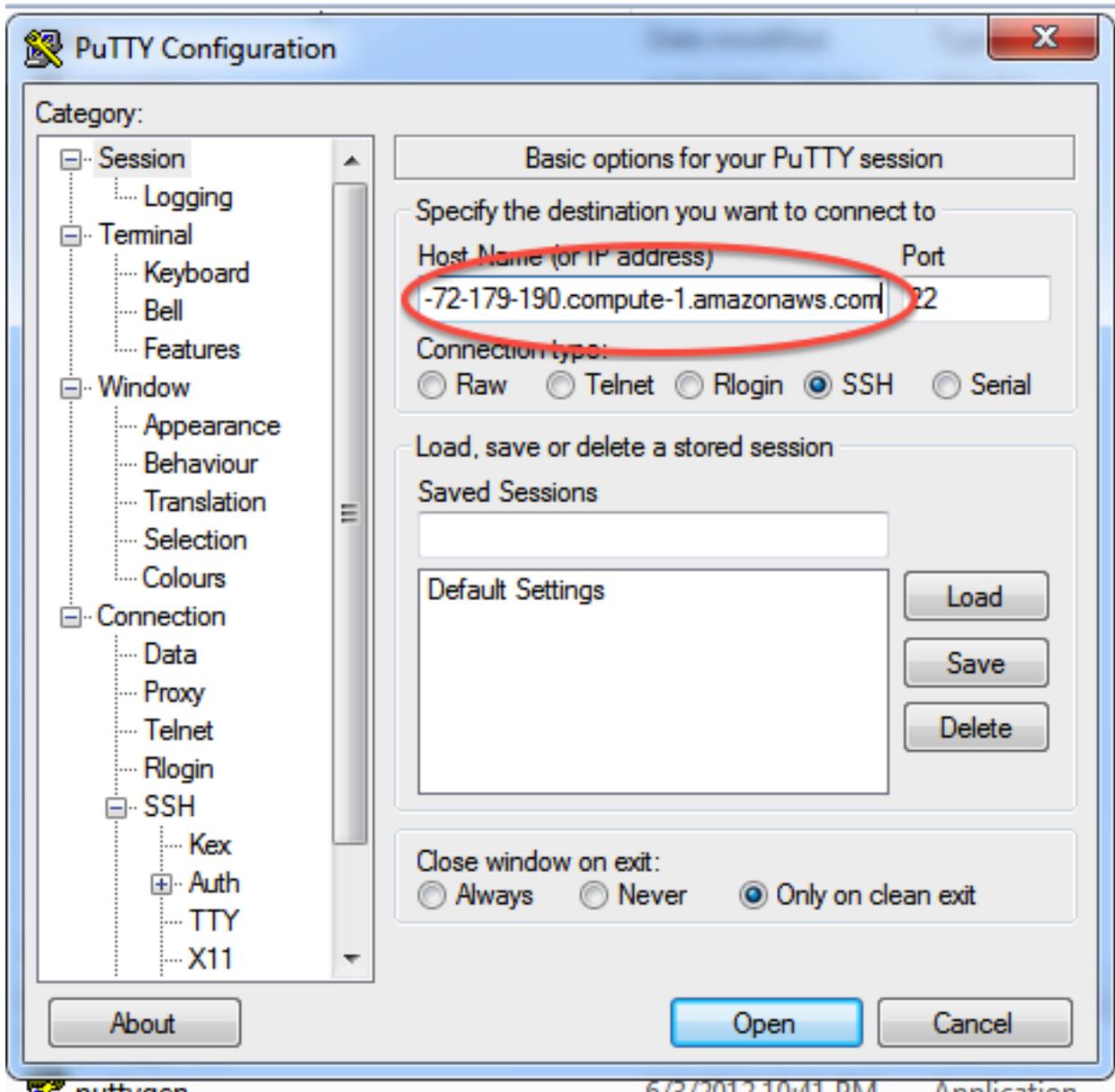
Logging into your EC2 instance with Putty

You’ll be provided with the network name, a.k.a. `public` DNS, of your virtual machine. This is the public name of your computer on the internet. It will look something like `ec2-174-129-122-189.compute-1.amazonaws.com`, for example.

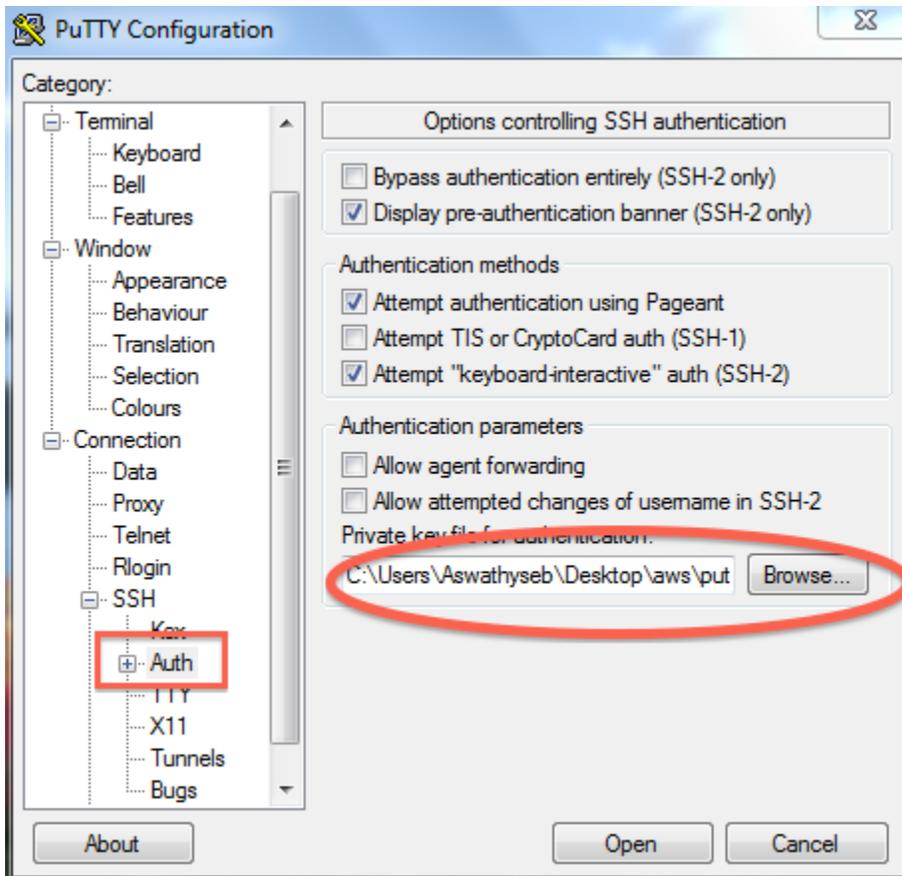
Connect to your particular virtual computer using PuTTY using the username `ubuntu`, as follows.

If you prepared in advance, the key provided for today should be on your Desktop already.

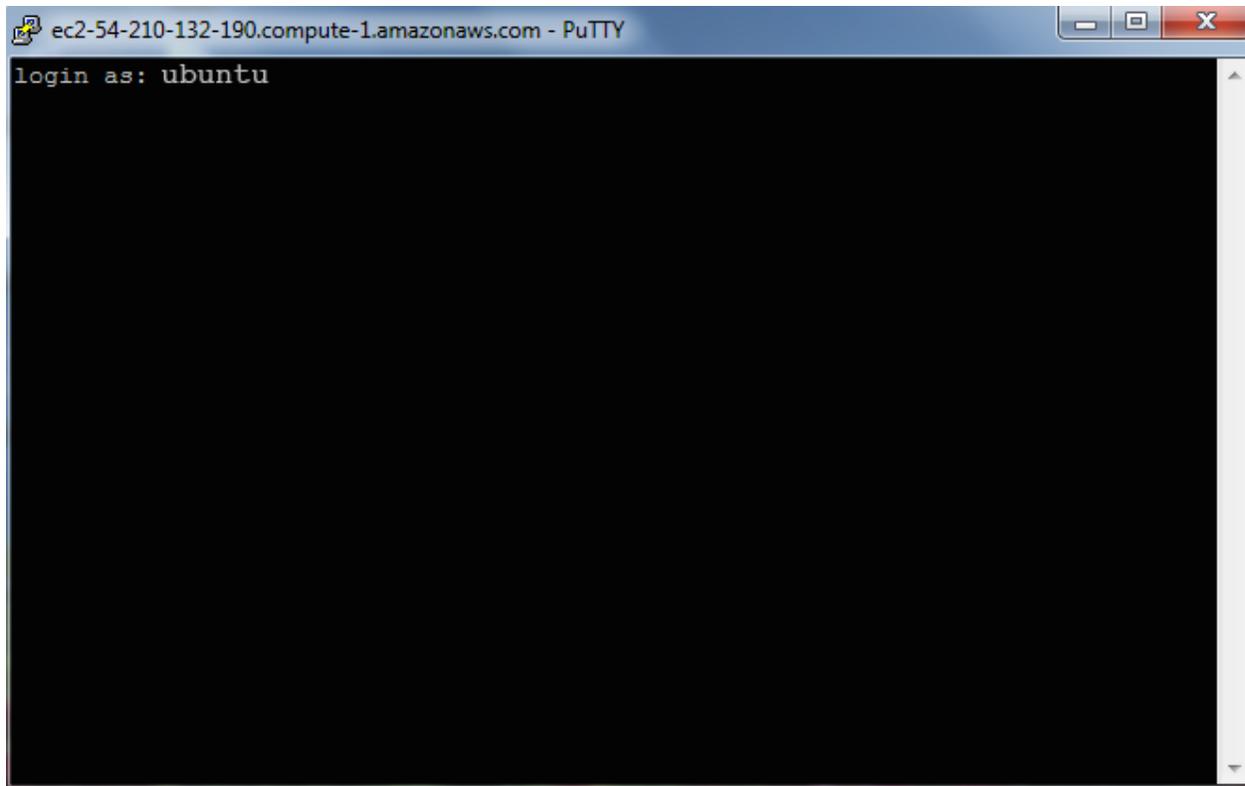
Open up PuTTY, and enter your virtual computer’s network name into the `Host Name` box. Just leave everything else in the window as it is.



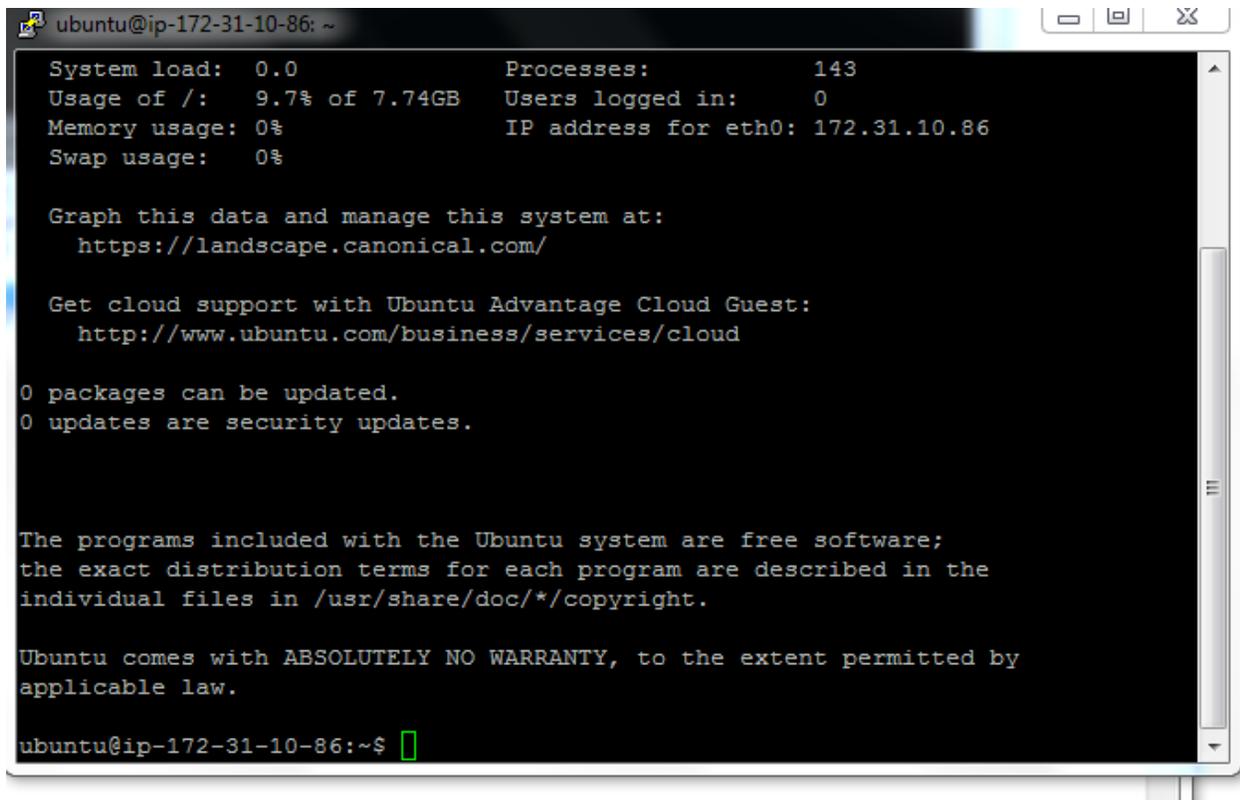
Now, you need to point PuTTY to the key file. In the left side of your PuTTY window, go down and find the SSH section, select Auth and in the bottom of the window on the right use the Browse button to locate the workshop.ppk file provided to you via email and sitting on your Desktop. Then select Open.



At the prompt log in as ubuntu.



Some information should scroll by and at the bottom you should now see a text line that starts with something like `ubuntu@ip-10-235-34-223:~$` as in the image below.

A terminal window titled 'ubuntu@ip-172-31-10-86: ~' with standard window controls. The terminal output shows system statistics: System load: 0.0, Processes: 143, Usage of /: 9.7% of 7.74GB, Users logged in: 0, Memory usage: 0%, IP address for eth0: 172.31.10.86, and Swap usage: 0%. It also provides links for system management and cloud support, and reports that 0 packages can be updated and 0 updates are security updates. A copyright notice and warranty disclaimer are also present.

```
ubuntu@ip-172-31-10-86: ~
System load: 0.0          Processes:          143
Usage of /:  9.7% of 7.74GB  Users logged in:  0
Memory usage: 0%          IP address for eth0: 172.31.10.86
Swap usage:  0%

Graph this data and manage this system at:
  https://landscape.canonical.com/

Get cloud support with Ubuntu Advantage Cloud Guest:
  http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@ip-172-31-10-86:~$
```

You're now officially connected to the machine at Amazon Web Services. Hooray! But you have have one last, IMPORTANT step to totally take control.

Type:

```
sudo bash
cd /usr/workshop
```

The first command restarts the bash shell with you using the machine as the super user and the second sets you in the directory where we will work today `usr/workshop`. You have to issue these commands each time you login; these settings don't persist.

This is where we will begin today and it after this it will be much like you are working on your own computer's command line.

OPTIONAL: To check out what you have you can type the command below to see

```
df -h
```

Exiting

(You probably won't need to do this today.)

To log out, type:

```
exit
logout
```

or just close the terminal or Putty window. (You cannot do this step wrong because ultimately you (or me, for today) have control of the instance in Amazon Web Services console.)

Logging into your computer instance “in the cloud” (NON-Windows version; for Mac and Linux)

These instructions are written for a Mac operating system, but they should work for Linux once you open your terminal program as well.

I will step through this process using a Mac at the beginning of the workshop.

Connecting using the key and SSH

You’ll be provided with the network name, a.k.a. public DNS, of your virtual machine. This is the public name of your computer on the internet. It will look something like `ec2-174-129-122-189.compute-1.amazonaws.com`, for example.

Connect to your particular virtual computer with `ssh` under the username `ubuntu`, as follows.

If you prepared in advance, the key provided for today should be on your Desktop already.

Next, start Terminal (,on a Mac it is in `Applications > Utilities`,) and type:

```
chmod og-rwx ~/Desktop/workshop.pem
```

to set the permissions on the private key file to “closed to all evildoers”. SSH will be protective and won’t allow you to connect if you skip this step. You only ever need to do this once for each key on your computer.

Next type:

```
ssh -i ~/Desktop/workshop.pem ubuntu@ec2-???-???-???-???.compute-1.amazonaws.com
```

(but you have to replace the stuff after the ‘@’ sign with the network name of the computer that is provided to you. You may wish to open your text editor and paste the template in the text editor for editing if it makes it easier for you because you can point and click there.)

With the above command, you’re using `ssh` to log in as user `ubuntu` to the machine `ec2-174-129-122-189.compute-1.amazonaws.com`, for an example name, using the authentication key located in `workshop.pem`, a.k.a. the `identity` file prefaced by the `-i` option in the command, on your Desktop.

The first time you try this on your own computer it will say that it cannot establish the authenticity of the machine you are trying to connect to and ask if you’d like to continue. Simply type `yes` at the prompt and the Amazon machine instance will be added to the list of known hosts on your computer.

```
$ yes
```

Some information should scroll by and at the bottom you should now see a text line that starts with something like `ubuntu@ip-10-235-34-223:~$`. You’re now officially connected to the machine at Amazon Web Services. Hooray! But you have have one last, IMPORTANT step to totally take control.

Type:

```
sudo bash
cd /usr/workshop
```

The first command restarts the bash shell with you using the machine as the super user and the second sets you in the directory where we will work today `usr/workshop`. You have to issue these commands each time you login; these settings don't persist.

This is where we will begin today and it after this it will be much like you are working on your own computer's command line.

OPTIONAL: To check out what you have you can type the command below to see

```
df -h
```

Exiting

(You probably won't need to do this today.)

To log out, type:

```
exit  
logout
```

or just close the terminal or Putty window. (You cannot do this step wrong because ultimately you (or me, for today) have control of the instance in Amazon Web Services console.)

Overview

General Overview of Workshop

```
Get computer with software and data up and running  
  |  
  V  
Map the appropriate reads  
  |  
  V  
Call peaks with MACS  
  |  
  V  
Examine peaks relative to known genomic features using CEAS  
  |  
  V  
Motif discovery with MEME
```

More Detailed Overview of Workshop Pipeline

```
Get proper resources for working on genome of interest  
  |  
  V  
Identify proper data on NCBI Short Read Archive  
  |  
  V  
Obtain in a format that will work for subsequent steps (ideally fastq)  
  |  
  V
```

```
Extract the Gal4 reads based on barcode
|
V
Map the reads
|
V
Improve the mapping of the reads
|
V
Preliminary view of mappings
|
V
Peak calling with MACS
|
V
Peaks relative to known genomic features with CEAS
|
V
Filter additional regions
|
V
Compare to published data
|
V
Motif discovery with MEME
```

Software Utilized

- Python
- BWA (Burrows-Wheeler Aligner)
- SAMtools OLD site: <http://samtools.sourceforge.net/>
- FASTX-Toolkit
- libgtextutils
- MACS2
- CEAS
- BEDtools
- MEME webservice

Suggested Other Software

- FastQC ← Not currently used here but good to have for initial assessment of reads. We will touch on quality control but not assess here ourselves.

Initial steps condensed for today

In the interest of time we are going to vastly condense the first two steps. I have done these steps on your instances already. Additionally, we'll get the third one running ahead of before we'll really need the results

So you now should have

```
genome.fa genome.fa.fai SRR346368.fastq
```

in your `/usr/workshop` directory.

You can just enter `ls` to see this if you are already in the `/usr/workshop` directory.

If you are unsure what is your current working directory, type `pwd`. If in doubt about any of that, just enter the commands below in your terminal

```
cd /usr/workshop
ls
```

- The `genome.fa` represents getting the genome data for the organism with which we'll be working.
- The `genome.fa.fai` file has summarized information about the lengths of each chromosome that we'll use for today's work. The details of obtaining these files is in the early part of the pipeline.
- The `SRR346368.fastq` is the experimental data. Because it is important to know what it is and where it came from for today's work, we'll look at that some.

Before that though, in the interest of time we are going to initiate a rate-limiting step first and then build up to you understanding what you did after.

Please enter in your terminal

```
git clone https://github.com/fomightez/sequencework.git
cp sequencework/SOLiD/split_SOLiD_reads.py .
python split_SOLiD_reads.py SRR346368.fastq TAGCGT 10
```

Now we'll talk about the two big parts of steps #1 and #2 we skipped. That should take us to the point of the commands we just ran and then we'll actually continue with the workshop steps as documented, picking up with the Mapping reads section.

Note that the instructions for steps #1 and #2 start off assuming you have a clean, new system (the set up of which is described [here](#) and [the pages after that](#)). This obviously isn't the case today and so we will not run those commands as we talk about those steps.

STEP# 1: Get proper resources for working on genome of interest

It is typical that you get a basic set of genome data for any organism with which you'll be working. By way of example, in preparation for today we'll get a file that includes the sequence of the entire budding yeast genome.

[Illumina's iGenomes site](#) is a collection of reference sequences and annotation files for commonly analyzed organisms. Importantly, many of these are in multiple forms and can save you conversions at times. You'll see today file conversions are very common, and obviating the need for more can be a welcome convenience.

We'll get the information that corresponds to the current version available from the [UCSC Genome Bioinformatics](#) for the budding yeast *Saccharomyces cerevisiae*. Download `sacCer3` from the [iGenome site](#).

```
wget ftp://igenome:G3nom3s4u@ussd-ftp.illumina.com/Saccharomyces_cerevisiae/UCSC/
↪sacCer3/Saccharomyces_cerevisiae_UCSC_sacCer3.tar.gz
```

And unpack it.

```
tar xzf Saccharomyces_cerevisiae_UCSC_sacCer3.tar.gz
```

(OPTIONAL) We'll clean up a bit by deleting the gzipped file we downloaded. Plus the IGenomes README.txt also because soon we'll have lots of files in our working directory.

```
rm Saccharomyces_cerevisiae_UCSC_sacCer3.tar.gz
rm README.txt
```

We'll need the sequence of the entire genome in fasta form and so we'll pull that out of the large collection of files for yeast, copying it to our current directory. The current directory is designated by the dot Unix shortcut at the end of the command below.

```
cp Saccharomyces_cerevisiae/UCSC/sacCer3/Sequence/WholeGenomeFasta/genome.fa .
```

We'll do likewise for another form of the genome information.

```
cp Saccharomyces_cerevisiae/UCSC/sacCer3/Sequence/WholeGenomeFasta/genome.fa.fai .
```

(OPTIONAL) We have what we need but let's just put all the yeast data somewhere out of the way so we have it if we need it but it isn't cluttering our working directory.

```
mv Saccharomyces_cerevisiae /usr
```

Verify when that is all set by checking directory.

```
ls
```

Alternatively, you can use [rsync](#) to go obtain these files directly from [UCSC Genome Bioinformatics](#).

In a browser you can download [here](#) or do [via FTP here](#). The fasta file from there seems to require additional conversion, see the about [twoBitToFa here](#)

STEP #2: Acquire experimental data

Identify proper experimental data

We are going to use data from this publication today.

```
Rhee HS and Pugh BF. 2011. Comprehensive genome-wide protein-DNA
interactions detected at single-nucleotide resolution.
Cell. 2011 Dec 9;147(6):1408-19. doi: 10.1016/j.cell.2011.11.013.
PMID: 22153082 http://www.ncbi.nlm.nih.gov/pubmed/22153082.
```

- Footnotes section of [paper](#) contains an ACCESSION NUMBERS section that provides us with the accession SRA044886.

That leads to [NCBI Sequence Read Archive\(SRA\)](#) where similar to the process of looking up Genbank sequence entry with an accession. So at the SRA you'd search the accession SRA044886 similarly.

Enter on the search box SRA044886.

Leads us [here](#)

Leads to 5 experiments with the last on the list, [SSFs 1](#) being, the one with Gal4 reads in which we are interested today.

Finally, you'll see two runs are listed on the bottom of that page, and if you were to parse the cryptic metadata at the top of the page, you'd see the specific run in which we are interested is SRR346368.

- Alternatively, the READ ME at http://downloads.yeastgenome.org/published_datasets/Rhee_2011_P MID_22153082/ leads to SRA044886 at SRA as well.

Obtain file of the experimental data in a workable format

When I first downloaded naively trying to click things in the browser, it downloaded as an SRA file.

NCBI regards the SRA file format as a basic format that saves space and allows conversion to several popular formats.

Going by my recent experience developing the pipeline for this workshop, I strongly suggest you always use the SRA-Toolkit, see [Converting SRA format data into FASTQ](#) to acquire and convert your data. The SRA handbook is [here](#). Presently, you can find the [SRA Toolkit Installation and Configuration Guide](#) [here](#). I strongly advise you to follow [those instructions](#) in step-by-step conjunction with the technical details of installation for installing on a Mac and EC2 instance the instructions exactly, trying the `./fastq-dump -X 5 -Z SRR390728 test` before doing any troubleshooting or configuring. The [NCBI SRA toolkit instructions](#) as they are written in April 2015 seem to skip a step and assume you really understand how to specify absolute paths on your system. Pointing the program file with the absolute path is critical to getting it working on your system. And the step they skip is placing the contents of the `bin` directory in a directory you set up; the instructions in the technical details of setting up the SRA toolkit on a Mac and EC2 instance include this VERY IMPORTANT STEP.

You will see other, older posts on dealing with getting SRA files and converting to fastq or going directly to fastq, such as [\[here\]\(https://www.biostars.org/p/128831/\)](https://www.biostars.org/p/128831/). Bear in mind the NCBI has been actively developing the SRA toolkit a while now and things have progressed making a lot of the information out there outdated. What worked in the past may not work now. The SRA toolkit is what NCBI has been emphasizing as the singular approach for obtaining data files. For example, if you can go through steps in [advice under post#7 here](#) to try and download select the fastq format using check boxes on a web form, you'll see you be presented with directions about using the SRA toolkit now. Similarly, despite old posts advising it, if you try to use curl or wget to download a link to reads on the command line, all you'll receive is a short note to use the SRA toolkit.

From experience I can tell you this is a very real issue and you might not know it when you are faced with massive files and determining absolute numbers and things isn't easy and troubleshoot when you don't necessarily know what you should have at the end when the process works or what form is best for going forward. This will become evident when we deal with the options needed to get today's data.

Obtaining a fastq file using the SRA toolkit

Go to the working directory; this will be the User system resources directory of our Amazon instances for no other reason except it will allow me to set the instances in a so-called `stop` mode that is cheaper to maintain. It is more like a `pause` mode because you can `restart` the instance. This way if you have questions at the end of the workshop, for the next few days we'll have the option to keep your instance exactly as you left it. (If we did it in the scratch disk area of the drive `/mnt`, which is otherwise a perfectly suitable place to work, all the contents would disappear if I `stop` the workshop instances. Then we'd be back at square one.) I don't plan to terminate them immediately. (Termination would destroy everything.)

We are going to make a subdirectory there to keep better organized. We'll call the subdirectory `workshop`.

```
cd /usr
mkdir /workshop
```

Enter

```
~/sratoolkit/bin/fastq-dump -B -I --split-spot --minReadLen 0 SRR346368
```

The download and conversion to fastq format triggered by the command should take a few minutes to run. Let it go.

Exploring the command

The path in front of `fastq-dump` is to point at a special location when I put this in program for convenience so I could stop and restart EC2 instance without needing to redownload the software. You generally could put this somewhere more useful and easily accessible.

What do the options mean? (You can type `./sratoolkit/bin/fastq-dump --help` to see the full list of other options.)

- The `-B` option is [covered here](#). It will yield you the fastq data as bases and not other representative symbols. Generally in newer data you do not need this option, but our data here is not from the latest Illumina pipelines and in fact is old and from a different (yet [still used, surprisingly to some](#)) technology, the Applied Biosystem, SOLiD.

Fastq files that have numbers instead of letters are in color space format. This format is a feature unique to the methodology used by the ABI SOLiD sequencer. ... In order to dump a “normal” base space fastq file, please run the `fastq-dump` command with the “-B” option.

- The `-I` option adds read id after spot id as `accession.spot.readid` on define of the read information. Just accept at this point it will make things easier to keep track of the mates in the read pairs. You may find this option convenient for other public data you access.
- Similar to the prior two options, `--split-spot` option generally will not be necessary for most modern datasets, but it is critical for the way this older form this older SOLiD data form is stored (as a single spot with a read for the genomic data and a read for the barcode) and for the next steps we are going to take with this data to filter it to only the reads for which we are concerned today. For modern data, the somewhat related `--split-files` option is commonly used when dealing with paired-end data and results in each pair of ends in a separate file, usually resulting in two files names similar to `SRRXXX_1.fastq` and `SRRXXX_2.fastq`. For data you are receiving directly from a facility it will probably already have been handled this way and hopefully they will disclose this information to you or have an FAQ describing how the indexed (or barcoded) and paired-end reads have been handled.
- `--minReadLen` of zero is designated so all reads will be kept. This is also to facilitate the next step dealing barcode by helping insure same number of lines for each and every set of reads. Again, this is not an option you’d typically use.

(As others have pointed out ([for example](#), splitting the paired-end reads into two files actually goes against the `tidy` data idiom of keeping associated data in a single file, but it is commonly done for convenience. A number of [processing and analysis programs expect the data in the separated form](#). The combined form is sometimes referred to `megred` or `interleaved` or `interwoven`. There are alternate ways to go back and forth between the two forms, see [split-pe.py here](#) and [interleave.py or here](#) or `SeqPrep` for examples, but as you can imagine, [best to handle this delicately as you have to hope there are no hiccups](#). Sorting out if there were or not can be a challenge with huge files with millions of reads involved. Plus as you’ll soon experience [files multiply](#).)

Ideally we should have just been able to type `./sratoolkit/bin/fastq-dump SRRXXXXXXXX` or maybe `./sratoolkit/bin/fastq-dump --split-files SRRXXXXXXXX`. With data you find publically it may take a bit of sleuthing to determine how it is stored and then decide how to best fit it into your analysis pipeline. The options used here today for the `fastq-dump` are a result of considering that.

Results of the command

The download and conversion should a good maybe 40 minutes or upwards to run. (This is why I set up the systems some in advance on the day of the actual workshop.)

When done, you should see

```
Read 46798614 spots for SRR346368
Written 46798614 spots for SRR346368
```

And you’ll have a file `SRR346368.fastq` in your directory.

If you type

```
ls -lah
```

you'll see the file is around 18 Gb.

Additional help for getting your data from the Short Read Archive

```
* [Several solutions from early 2015] (http://genomespot.blogspot.com/2015/01/sra-toolkit-tips-and-workarounds.html)

* Use ENA as described in a 2013 post [here] (http://seqanswers.com/forums/archive/index.php/t-19848.html)

    You go to the [European Nucleotide Archive] (http://www.ebi.ac.uk/ena) and search
    for your data. For example, `SRR346368` in `text` search.
    Then drill down by clicking on 'Run 1' and then [you'll see a table] (http://www.ebi.ac.uk/ena/data/view/SRR346368) with 'Fastq files (ftp)'
    The odd thing is that for this example the ENA approach didn't seem helpful.
    First, there are three files when by contrast there is only one for this run at
    NCBI SRA. Second, turns out these are colorspace format?!?! Puzzling is that in the
    middle of the three files there is 35 length sequences with many 10s in the middle
    and then more 35 bp length ones at end. Plus the reads seem out of order if you
    simply look at the first 10 lines of the first file. So just be aware you may need
    to look around at the possible resources to see what provides the most useful form
    of the dataset.
```

Two take away lessons: (1) when using public data, you may need to try several forms to get what is easiest to work with for you. (2) When working with millions of reads, it can be easy to miss small details that could have huge effects downstream. For example, depending on how subsequent steps are performed, such as demultiplexing by barcode, tossing out of the zero length reads could bring things out of sync and result in reads getting sorted completely wrong.

Extract the Gal4 reads based on barcode

Note: Generally, most times the sort of processing (demultiplexing based on barcodes or indexing) described in this section will already have been done in modern datasets you either receive from a facility or find publically.

According to metadata at [http://www.ncbi.nlm.nih.gov/sra/SRX098212\[{}accn{}\]](http://www.ncbi.nlm.nih.gov/sra/SRX098212[{}accn{}])

```
Experiment2 (SSFs 1). Barcoded paired-end run was performed in SRX098212 by AB SOLiD System 3.0. 35 bp F3 (mate 1) run contains genomic sequences. 10 bp R2 (mate 2) run contains 6 bp barcoded sequences (first 6 bp are barcode in 10 bp R3 paired-end sequences). SRX098212 contains the following 2 runs with indicated barcode sequences: Run1. SRR346368 (1.8 Gb) contains two samples: ChIP-exo Reb1 (biological replicate 1, GGGCTT) and ChIP-exo Gal4 (biological replicate 1, TAGCGT).
```

We only want to map the Gal4 reads in our subsequent analysis. Hence, we want to sort out and make a new file of the genomic sequences F3 (mate 1) that match the Gal4 barcode TAGCGT. Thus removing those that correspond to the Reb1 data.

The journal article was unclear how they did this step so I made a custom Python script to implement this approach. One thing to keep in mind about this program is that it is very stringent in that the barcode has to be an exact match at the start of mate 2.

We'll get the script from among a batch of sequence analysis scripts I have at Github.

```
git clone https://github.com/fomightez/sequencework.git
```

Then to make it easy we'll copy the particular script into our current working directory, which we'll specify via `.` in the line below.

```
cp sequencework/SOLiD/split_SOLiD_reads.py .
```

With the data and the script in the same directory, enter on the command line

```
python split_SOLiD_reads.py SRR346368.fastq TAGCGT 10
```

It will take about 15 to twenty minutes to run.

For the sake of time, we are not going to concern ourselves much with the details of the script today. If you want to continue your exploration of coding with Python that we started in the first session, I'd suggest you look over the implementation of the sorting described above [here](#). The main code is way at the bottom, and keep in mind this is a fairly advanced script only because it includes several bells and whistles, such as a built-in debugging option, a usage guide, and reading arguments from the command line. The link is actually for the code for a simpler version of the script we used today. The one we used has an extra check built in to make sure the sets of reads always matched up based on the annotation for each read. The documentation for both scripts can be found [here](#).

Upon completion of the script running and processing the data, you should see something very similar to below. Make sure your collected reads count is similar (hopefully, even exactly the same).

```
root@ip-10-169-123-185:/usr/workshop# python split_SOLiD_reads.py SRR346368.fastq_
↳TAGCGT 10
Reading in your FASTQ file...
Concluded.
46798614.0 read pairs analyzed. A grand total of 93597228 reads.
180054 reads collected based on barcode.
The file SRR346368_TAGCGT.fastq has been created in same directory as the input file.
```

Mapping reads

There are several steps involved in mapping our sequence reads and getting the output into a usable form. First we need to tell bwa to make an index of the reference genome. This is a way to set up the genome data in a way that will be computationally efficient for accessing. You'll see this is a common theme in bioinformatics pipelines.

```
bwa index genome.fa
```

Now we are ready to do the mapping. It takes two commands. The `bwa aln` command aligns the reads and produces a file of all the possible candidates. `bwa samse` takes that file and the knowledge it is single-end sequencing to process the alignment data to produce the resulting SAM file where repetitive hits will be randomly chosen.

```
bwa aln genome.fa SRR346368_TAGCGT.fastq > SRR346368_TAGCGT_Ys_genome.sai
bwa samse genome.fa SRR346368_TAGCGT_Ys_genome.sai SRR346368_TAGCGT.fastq > SRR346368_
↳TAGCGTvsYsGenome_bwa.sam
```

See [manual](#) and [Biostars question on bwa align and bwa samse](#) or [Biostars question on what is a .sai file](#) for a full and two quick guides, respectively, to what is going on with running the two commands.

Note if you are working with paired-ends you'd use `bwa sampe` to generate your SAM file, providing the information for both reads in your command. This would use the file created by `bwa aln` and the knowledge that it is paired-end sequencing to select the best hits according the orientations, distance between the reads, etc.. An unrelated example command for paired-ends is below as an EXAMPLE. NOT FOR RUNNING TODAY.

```
bwa sampe dmel-all-chromosome-r5.37.fasta RAL357_1.sai RAL357_2.sai RAL357_1.fastq_
↳RAL357_2.fastq > RAL357_bwa.sam
```

BWA is able to take advantage of multi-threading and so you'll get more speed if you have multiple cores on your computer which many have. For example, my iMac at home has two cores while my iMac at work has four and several of Amazon's Elastic Cloud computing instances have 4 or more cores as summarized [here](#). The example we chose for today has few reads to map and so it is fast without running on multiple threads. However, if you had a lot of reads it would be preferable to run the alignment on multiple threads at the same time using your machines multi-core processor. To do so you use the `-t` option to specify the number of threads. For example, to run in four simultaneous threads:

```
bwa aln -t 4 genome.fa SRR346368_TAGCGT.fastq > SRR346368_TAGCGT_Ys_genome.sai
```

(You can use a UNIX trick to let the computer determine how many cores it has and thus how many threads it can run. On Linux, replace 4 on the above line with `$(nproc --all)` as illustrated [here](#). On a Mac, you'd replace 4 with `sysctl -n hw.ncpu`.)

Now look at the SAM output file

```
head -100 SRR346368_TAGCGTvsYsGenome_bwa.sam
```

As summarized [here](#)

The first four columns, in order, are: identifier, flag, chromosome, position, map-quality.

One of the key pieces of information is found in the second column. See [here](#) for a nice summary of how to interpret it for unpaired reads. For paired reads interpreting is more complex as described there as well. For full details of the SAM format specification see [here](#). A nice primer can be found [here](#).

We see a lot of '4's indicating only a few mapped. Can we get some feedback easily? We can if we use the SAMtools software to look at the SAM file. Type:

```
samtools
```

You'll see a listing that includes:

```
flagstat    simple stats
```

That looks promising. As described in [post #5 here](#) it isn't documented in the manual. Let's try `flagstat` anyways.

```
samtools flagstat SRR346368_TAGCGTvsYsGenome_bwa.sam
```

We get:

```
EOF marker is absent. The input is probably truncated.
[bam_header_read] invalid BAM binary header (this is not a BAM file).
```

Plus a bunch of zero results.

Despite the name, it seems it needs a BAM file to do this. Well, a BAM file is just a compressed binary version of a SAM file according to the list of Data File Formats at UCSC Genome Bioinformatics.

The first example under Examples on SAMtools documentation page (WAY AT THE BOTTOM) illustrates what we'd like to do.

```
samtools view -bS SRR346368_TAGCGTvsYsGenome_bwa.sam > SRR346368_TAGCGTvsYsGenome_bwa.
↪bam
```

(According to [here](#) the `-S` tag for `samtools view` had in the past been used to be to specify that the input is in the SAM format. That flag looks to be unnecessary now according to the [documentation for the updated version](#). The way shown above should work no matter what version of SAMtools is on your system.)

Now since it is a binary and compressed version of a SAM file, unlike the SAM file we just examined, a BAM file is not human-readable in any way. If you need convincing, just type:

```
head SRR346368_TAGCGTvsYsGenome_bwa.bam
```

However, `samtools flagstat` hopefully will work with this now. Try:

```
samtools flagstat SRR346368_TAGCGTvsYsGenome_bwa.bam
```

Only 4.44% mapped?!?!?

Improving the mapping of the reads

Let's review the methods and see how it compares to what we did. (Fortunately, in this case they detailed parts of the computational analyses right in the supplemental information of the paper. You won't always be this lucky.)

From the supplemental methods information in paper:

Alignment to genome, peak calling, and data sharing. The *Saccharomyces* reference genome was obtained from www.yeastgenome.org (build: 19-Jan-2007). The entire length of the sequenced tags were aligned to the reference genome using Corona Lite software provided by the SOLiD system, allowing up to 3 mismatches. This process was repeated for the remaining tags, after removal of the 3' most 6 bp, which tend to have higher error rates.

As noted in the supplemental methods information they allowed mismatch to be 3, the [default for BWA listed as the first option](#) is 4% of read length which is much smaller than 3. So let's try what they did by setting that `-n` option we just examined.

```
bwa aln -n 3 genome.fa SRR346368_TAGCGT.fastq > SRR346368_TAGCGT_Ys_genomeN3.sai
bwa samse genome.fa SRR346368_TAGCGT_Ys_genomeN3.sai SRR346368_TAGCGT.fastq > \
↳SRR346368_TAGCGTvsYsGenome_bwaN3.sam
```

Let's convert that to BAM and look at the stats.

```
samtools view -bS SRR346368_TAGCGTvsYsGenome_bwaN3.sam > SRR346368_TAGCGTvsYsGenome_
↳bwaN3.bam

samtools flagstat SRR346368_TAGCGTvsYsGenome_bwaN3.bam
```

Indeed, improvement. Nearly 6% now.

The supplemental methods for the paper go on to describe that they took these unmapped reads and cut off the last six and then ran the mapping with those again. Obviously they are sharing some of the results if their quality assessment here and this is in part while we didn't address earlier. (We skipped running FastQC as the very first step in the interest of time and because they reported related information.) How would we do such cutting off of bases?

As you probably guessed, with **more software**.

First we need a way to get the unmapped reads. To get them, we'll again rely on SAMtools.

```
samtools view -f 4 SRR346368_TAGCGTvsYsGenome_bwaN3.bam > SRR346368_TAGCGTvsYsGenome_
↳bwaN3.unmapped.sam
```

That filters the reads that have 4 for the [flag value](#). (We won't need the `@SQ` header for this file so we left off the `-h` option.) You can look at the output using the `head` command.

The filtering created a SAM file with the unmapped, but we need to supply bwa with a fastq file for the mapping step. As described [here](#), you have options for doing this. We are going to take advantage of a [unix utility called Awk](#) to do this, adapting the solution [described here](#).

```
grep -v ^@ SRR346368_TAGCGTvsYsGenome_bwaN3.unmapped.sam | awk '{print "@$1"\n"$10\n\n"$11}' > SRR346368_TAGCGTvsYsGenome_bwaN3.unmapped.fastq
```

The description of the approach nicely breaks down the relevant steps. One thing it assumes is you know `|` stands for using pipes to pass the output from one command to another's input. The [Practical Computing for Biologists](#) book I recommended covers this and redirects if you need guidance about pipes with a biological bent beyond what the internet provides.

You can look at the output using the `head` command and see we have our fastq file of unmapped reads.

As for what software for trimming the reads... this is one of those processes where a lot of solutions exist. You could even build a customized one. For example, the Python program we used earlier to sort based on barcode could easily be copied and adapted to make a new program that takes every read and collects all except the last 6 bases and outputs it to a new file. One of the most popular set of tools for post-processing raw short reads for downstream approaches is the [FASTX-Toolkit from the Hannon lab](#). There are web-based versions within Galaxy and command line versions. It is a handy set of programs to be able to run and so we'll use a program from the collection today for trimming, `fastx_trimmer`, as having the tools installed will present additional post-processing abilities to you, including a [more full-featured approach to demultiplexing based on barcodes](#).

The Fastx toolkit requires you install `libgtextutils` first. Once you have those installed you are ready to trim. (I have installed them for you today.)

```
fastx_trimmer -l 29 -i SRR346368_TAGCGTvsYsGenome_bwaN3.unmapped.fastq -o unmapped.\n→trimmed.fastq
```

Now we have the unmapped reads in a trimmed form. To make the steps after easier, let's take the untrimmed reads that mapped previously and append them to the unmapped reads in a trimmed form.

Since we saw before that filtering on the 4 flag worked, [the solution to getting the mapped reads](#) is to instead keep all **WITH THE EXCEPTION OF** the unmapped.

```
samtools view -F 4 SRR346368_TAGCGTvsYsGenome_bwaN3.bam > bwaN3.mapped.sam
```

(Note that one solution you may have thought of is that you could separately collect those that mapped to the forward and the reverse using the `-f` option to filter on 0 and 16, respectively. **DON'T**. While it will work for filter on 16 to get those mapped to the reverse, it fails to filter out anything if you try to filter on 0. Perhaps counterintuitively, to get those forward mapping reads you can see in the SAM file as having a 0 value for the flag, you need to actually use `-F 20` as the option setting. The reason the filter on 0 approach fails and the `F -20` option works is because the flag is bitwise so that the values can be combined to express complex sets in a compact, 'computer science-wisely' way. [This tool here](#) is particularly helpful for novices deciphering this. In this tool's `flag` box, enter the flag value of 20 and hit `explain` you see that is a way of expressing those that are unmapped or map to the reverse strand. With `-F` option we exclude those. Compounding the complex nature of this encoding is the fact it is hexadecimal based. You may wish to seek additional help on this concept by looking at [slides 28-31 of this presentation](#) and information and links or [here](#) or [herehere](#) or [here](#) or [here](#) as mentioned earlier or [swbarnes' answer here](#).)

Again, we need to convert these to fastq as we did before.

```
grep -v ^@ bwaN3.mapped.sam | awk '{print "@$1"\n"$10\n\n"$11}' > bwaN3.mapped.\n→fastq
```

Now that we have a fastq file of just the mapped reads we are going to start a combined file and append those to the unmapped. We'll use the redirection symbol `>>` to designate appending.

```
cat unmapped.trimmed.fastq > combined.fastq
cat bwaN3.mapped.fastq >> combined.fastq
```

Finally, we should be able to emulate the mapping approach used in the paper.

```
bwa aln -n 3 genome.fa combined.fastq > combined_Ys_genomeN3.sai
bwa samse genome.fa combined_Ys_genomeN3.sai combined.fastq > combined_vs_YsGenome_
↪bwa.sam
```

The first line of that should take about three minutes.

And convert to a BAM file so we can use flagstat.

```
samtools view -bS combined_vs_YsGenome_bwa.sam > combined_vs_YsGenome_bwa.bam
```

So how did we do?

```
samtools flagstat combined_vs_YsGenome_bwa.bam
```

Alright, 13% mapped is a big improvement over the 6% we had. But normally you might expect better. I have communicated with Ho Sung Rhee and he says this data was from the “early days” and was concerned about a particularly high error rate. High throughput requires a different mindset than a lot of the traditional molecular biological data we collect. A lot of times you are fighting tooth and nail to get a signal, and are concerned what you want to see was lost along the way. We still have A LOT of data here even though we only around 10% of what we corralled up to this point is mapping. A lot of what Titus Brown and colleagues do does in processing of metagenome assembly data is data reduction approaches. Maybe the ChIP-exo processing steps combined with our stringent requirements of the barcode resulted in a reduction of data that benefits us ultimately? Subsequent analysis will let us evaluate the data in this case.

Preliminary view of mappings

So far we have just looked at the mappings in a list. SAMtools has a program called `tview` that can help us get a sense visually of what we have done so far.

What does `tview` need? Let’s check [the manual](#). I’d suggest searching `tview` on the page because oddly the documentation is not internally linked and indexed.

We see need to do yet some further file formatting to get the data in a usable form for `tview`.

In order to be able to use our most recent mapping results with `flagstat` we already converted it into a BAM file. Now we need a **sorted** BAM file.

```
samtools sort combined_vs_YsGenome_bwa.bam combined_vs_YsGenome_bwa.sorted
```

Okay, check your file listings.

```
ls
```

Now that we have that sorted BAM file, let’s TRY to use that to run `tview`. Recall from the [documentation](#) we need a fasta formatted reference sequence too.

```
samtools tview combined_vs_YsGenome_bwa.sorted.bam genome.fa
```

Okay, so this is going to cause SAMtools to give you feedback.

```
[bam_index_load] fail to load BAM index.  
Cannot read index for 'combined_vs_YsGenome_bwa.sorted.bam'.
```

This is telling us we need to format yet one more type of file concerning the data. We need to index the data. If you search on `index` in the [SAMtools documentation](#) you'll see there is a command to do this and that it is needed for commands when `region` arguments are used. Going back to the documentation for `tvview` you'll see `tvview` can use these. *Lesson: the documentation may not always explicitly say what you need where you might think it would be best placed and error/feedback messages will often guide you.*

Let's index our sorted bam file.

```
samtools index combined_vs_YsGenome_bwa.sorted.bam
```

You'll see this creates a file ending `combined_vs_YsGenome_bwa.sorted.bam.bai` with `.bai` at the end indicating SAM/BAM index file and if you look at the `samtools index` documentation you'll see the purpose is to enable fast access by `tvview` to the coordinate-sorted data. Creation of an index file (or lookup table, etc.) is a common computer science solution to speeding things up by otherwise eliminating a computationally-costly process.

Finally, now that we have everything, let's run `tvview`.

```
samtools tvview combined_vs_YsGenome_bwa.sorted.bam genome.fa
```

You'll be whisked away to a sequence map. [This tutorial](#) has a guide to navigating. Use `h` and `l` to move right and left; `j` and `k` for up and down. (Arrow keys seem to work too.)

Type `?` for help. You may need to expand your terminal window to see all the way to the bottom of the `help` menu.

Hit `q` to exit.

Let's go somewhere specific.

```
samtools tvview -p chrII:278586 combined_vs_YsGenome_bwa.sorted.bam genome.fa
```

You can also go to a location when `tvview` by typing `g` for `goto` and entering the location. Try with `chrXII:289738`. (You may need to expand your terminal window to see all the way to the bottom.)

Peak Predictions with MACS

We'll use popular program MACS for peak prediction. The current version is 2. and Tao Liu maintains it [here](#). We'll use the sub-command `callpeak`, this being the main function of MACS. See [here](#) for a guide to the options.

```
macs2 callpeak -t combined_vs_YsGenome_bwa.bam --name gal4 --gsize 1.21e7 --nomodel --  
↵shift -100 --extsize 200
```

`--gsize` is the genome size option and needs to be set or it will use human as default.

`name` is to give the output files identifying names.

Finally, `--nomodel --shift -100 --extsize 200` comes from [the documentation on Github concerning the shift option](#). These are suggested for DNase-Seq datasets and so used here since ChIP-exo shares some similarities.

I found same result with or without the use of the tag size option, `--tsize 35`, and so I left it out.

Note that MACS originally required the ChIP data in BED format so at least we dodged **ONE** conversion.

Additionally, `-outdir` is a great option to use to help you keep organized. We are purposefully not using it today only to make things easy for navigating on these transient machines. Obviously, this isn't good data management practice.

Peaks relative to known genomic features with CEAS

We now go onto visualizing the ChIP enrichment signals relative known genomic features. For that we'll use [CEAS](#), another Python package by Hyunjin Shin and Tao Liu from when they were in Xiaole Shirley Liu's Lab. (There is a [web-server version](#) but it is limited to mouse and human and seems no longer operating.)

Probably not surprisingly at this point, running CEAS requires some additional preparation. CEAS requires the peaks to be specified in the BED format, whereas the signal is presented in a WIG file. We have the peaks in the Bed format already. We need to convert the signal from a sorted Bam file to WIG.

Additionally, the CEAS program operates on a SQLITE database that contains the information on the annotated regions of the genome. We need to build that database ourselves since there is not a prebuilt one for yeast available from the author's site. If we provide as option `-d` the name of the genome assembly and as option `-g` the annotation table, both as they are referred to at [the UCSC Genome Bioinformatics site](#), a utility included in the CEAS installation package will contact [the UCSC Genome Bioinformatics site](#) and build the database for us. In addition to indicating the genome and annotation table, the annotation building process requires a wiggle file that describes the genomic locations that are valid for data selection. For example, a tiling array may only cover some parts of a genome. In our case, we theoretically should have covered all the genome and so we simply generate a wiggle file that covers the full length of each and every chromosome. In the interest of time, this has already been done for you; the file is `sacCer3.wig`. You can acquire it by running the command below on the command line of your instance or [get it at github](#) if you are running this pipeline locally; information about making this file is [here](#).

```
wget https://raw.githubusercontent.com/fomightez/may2015feng_gr_m/master/sacCer3.wig
```

To build the annotation database run

```
build_genomeBG -d sacCer3 -g sgdGene -w sacCer3.wig -o sc3.db
```

(NOTE for anyone running this not on Amazon EC2 instances: the above command needs to access an external MySQL database and I believe may be a problem on certain networks.)

We now have a custom database `sc3.db` that can be used to generate reports with CEAS.

What file formats do we need for running CEAS?

[CEAS manual](#)

We still need a WIG file with the Chip-Seq data. Following from the advice of Stew on [this page Bam2Wig](#) and [this page](#), we can use the sorted bam file, `combined_vs_YsGenome_bwa.sorted.bam`, to make such a file. (Yes, this file contains the same information as BAM file used for peak predictions with MACS2, but even related bioinformatics software will often require files in different formats.)

```
samtools mpileup combined_vs_YsGenome_bwa.sorted.bam | perl -ne 'BEGIN{print "track_
↪type=wiggle_0 name=combined_vs_YsGenome_bwa description=combined_vs_YsGenome_bwa\n"}
↪;($c, $start, undef, $depth) = split; if ($c ne $lastC) { print "variableStep chrom=
↪$c\n"; };$lastC=$c;next unless $. % 10 ==0;print "$start\t$depth\n" unless $depth<3;
↪' > combined_vs_YsGenome_bwa.wig
```

Finally, CEAS uses the peaks `gal4_summits.bed` predicted with MACS, the wiggle file `combined_vs_YsGenome_bwa.wig` with the ChIP-Seq signal and the custom built database `sgd.db`:

```
ceas -g sc3.db -b gal4_summits.bed -w combined_vs_YsGenome_bwa.wig
```

The reports are generated as the file `gal4_summits.pdf` and contains several graphs and plots.

How do you view the pdf? The issue here is that it is on a EC2 instance at Amazon and not your local computer. While it is fairly straightforward to put it on your computer (I highly recommend scp), in the interest of time I have placed it in the associated Github repository. Click [here](#) or paste the link below into your browser URL bar.

```
https://github.com/fomightez/may2015feng_gr_m/blob/master/gal4_summits.pdf
```

See the legends at the bottom of [here](#) for help in interpreting the graphs.

The first plot shows 5.9% mappable on genome background because we didn't provide a typical control sample here and it just calculates 5.9% without such input due to the way the calculation works having a cutoff of 5.9. See in the [CEAS paper](#)

where the background is the 9th order nucleotide Markov dependency estimated from the human genomic sequence. A score cutoff of Max ($5,0.9 \times \text{Motif Relative Entropy}$) is used to call a motif a hit.

Plus it is probably moot for the ChIP-exo data used here. Recall from Rhee and Pugh, 2011

Uncrosslinked nonspecific DNA is largely eliminated by exonuclease treatment, as evidenced by the repeated failure to generate a ChIP-exo library from a negative control BY4741 strain. Therefore, use of an exonuclease makes comparisons to input DNA or mock IP moot, in that such DNA is destroyed.

Optional: Look at reads and/or peaks in IGB or IGV or SeqMonk OR UCSC genome browser

In most viewers the MACS2 output can be directly loaded into according to documentation. Your mileage may vary. We will not be doing this in the interest of time today.

Comparison to published data

Preparation

Conveniently, the authors and SGD made the locations they deemed binding sites available as a BED file.

You can use your browser on your local computer to get it at

```
http://downloads.yeastgenome.org/published_datasets/Rhee_2011_PMIID_22153082/track_
↳files/Rhee_2011_Gal4_ChIP_exo_bound_locations_V64.bed
```

Alternatively if you wish to stay working on the command line...

Downloading on a Mac

```
curl -o Rhee_2011_Gal4_ChIP_exo_bound_locations_V64.bed "http://downloads.yeastgenome.
↳org/published_datasets/Rhee_2011_PMIID_22153082/track_files/Rhee_2011_Gal4_ChIP_exo_
↳bound_locations_V64.bed"
```

Downloading on a Linux instance:

```
wget -O Rhee_2011_Gal4_ChIP_exo_bound_locations_V64.bed "http://downloads.yeastgenome.
↳org/published_datasets/Rhee_2011_PMIID_22153082/track_files/Rhee_2011_Gal4_ChIP_exo_
↳bound_locations_V64.bed"
```

Now we could compare that to our peak summits.

Before we do that, however, let's do a little more filtering based on standards set in the paper.

From the supplemental information Data Analysis section for the paper Rhee and Pugh, 2011: From Set of Gal4 bound locations

Set of Gal4 bound locations. The final set of 15 Gal4 bound locations was determined to be those locations having all of the following properties: 1) peak-pair distances between -1 and 61 bp, 2) peak-pair found in 2 of 3 replicates, in which a replicate peak-pair was <28 bp away (midpoint to midpoint), 3) a median of 75 normalized tags (see note above) per peak-pair (5% of the average tag count of all 15 peak-pairs), and 4) not at telomeric, tRNA, and rDNA regions.

We'll discuss some of this shortly, but for now let's focus on property four. The main issue I saw in my development of this workshop was summits in the telomeric DNA regions and so let's at least get rid of those. Using the lengths of the chromosomes that is at the beginning of several of our SAM file, it would be fairly easy to make a Python program to pick a distance from the ends of the chromosomes and remove any summits in those. However, there is a way to filter out regions that is more general and we'll use that next.

Filter additional regions

Ideally we would filter on all the regions, the authors of the study also filtered on. In the interest of time though, today we'll just focus on one. The telomeric sites.

The Rhee and Pugh, 2011 paper used genomic features to filter out results from peak-calling. We want to do this as well.

A good, general approach seems:

- get list or lists as Bed file from YeastMine
- Uses Bedtools with `intersectBed -v` to filter out those summits that are in those regions. Our example here will be telomeres.

The steps for getting the telomere Bed file are spelled out [here](#).

I'll demonstrate the process of acquiring the file fairly fast here, and then we'll just download the result file in the interest of time. The page at the link above spells the approaches if you are doing this yourself.

```
wget https://raw.githubusercontent.com/fomightez/may2015feng_gr_m/master/telomere_
↪regions.bed
```

Now to filter with that file. We'll use `bedtools's intersect` function with the `-v` option to exclude those that have any overlap with the telomere regions.

```
bedtools intersect -v -a gal4_summits.bed -b telomere_regions.bed > filtered_gal4_
↪summits.bed
```

Now we are ready to compare.

Compare

Compare your `filtered_gal4_summits.bed` to `Rhee_2011_Gal4_ChIP_exo_bound_locations_V64.bed`.

I am going to write the command line commands but feel free to use whatever method you want to examine and compare the files.

```
head -30 filtered_gal4_summits.bed
tail -15 Rhee_2011_Gal4_ChIP_exo_bound_locations_V64.bed
```

Interesting. While not perfect our summits do overlap with many of theirs.

Two parts from the supplemental information Data Analysis section for the paper Rhee and Pugh, 2011 deserve highlighting in the light of this:

From Alignment to genome, peak calling, and data sharing

The resulting sequence read distribution was used to identify peaks on the forward (W) and reverse (C) strand separately using the peak calling algorithm....

From Set of Gal4 bound locations

Set of Gal4 bound locations. The final set of 15 Gal4 bound locations was determined to be those locations having all of the following properties: 1) peak-pair distances between -1 and 61 bp, 2) peak-pair found in 2 of 3 replicates, in which a replicate peak-pair was <28 bp away (midpoint to midpoint), 3) a median of 75 normalized tags (see note above) per peak-pair (5% of the average tag count of all 15 peak-pairs), and 4) not at telomeric, tRNA, and rDNA regions.

Note two huge differences probably contribute largely to the differences in our results:

```
* Largely in the interest of time, I didn't have us do the peak calling with the
↳ forward and reverse mapped reads separately and then process those peaks further as
↳ the authors describe. Since the authors went on to use the peak-pair midpoint to
↳ locate the binding site, as described in Figure S3, I figured the peak calling
↳ algorithm would largely average this out to locate the midpoint on its own and that
↳ should be good enough for our purposes today.

* As best I can tell, we were only supplied data from 1 replicate. Their properties
↳ also mention things about 3 replicates but they only posted one replicate for Gal4
↳ data it seems. I checked all 5 sets of data linked and only see the one replicate
↳ for Gal4 mentioned. This will obviously factor into differences with the results we
↳ get in workshop as compared to the paper.

* Additional differences could arise from the different software used in the pipeline
↳ and the more limited filtering we performed. For example, the MACS 2012 guide
↳ suggests using more specialized tools for data types like DNase-seq and ChIP-exo is
↳ similar to that.
```

Motif discovery with MEME

Preparation

To look for motifs, let's try to take double the read length (35 * 2) of flanking sequence on each side of the peaks to account for reads going either direction from the peak summits and allowing ample space for a reasonable size DNA binding site. We'll use `bedtools slop` from the Bedtools collection to expand the ranges upstream and downstream to the locations of our peaks.

```
bedtools slop -i filtered_gal4_summits.bed -g genome.fa.fai -b 70 > gal4_summits_more.
↳ bed
```

If you'd like to use the Rhee and Pugh 2011 data, the command is

```
bedtools slop -i Rhee_2011_Gal4_ChIP_exo_bound_locations_V64.bed -g genome.fa.fai -b
↳ 29 > rhee_more.bed
```

Note: the `genome.fa.fai` file has summarized information about the lengths of each chromosome that `bedtools slop` can use to limit the ranges added to intervals to not extend beyond the actual length of the chromosome sequences. (The support for `.fai` files for `bedtools slop` was only noted among the comments at the bottom of the page [here](#).)

(The 29 base selection for the Rhee and Pugh Bed file was simply an arbitrary value based around the midpoint of the peak-pair distances identified in the paper. Obviously if we were doing this for actual discovery we may have wanted to try a few different values here, or better yet given the method, worked out for ourselves the peak-pair values for our data.)

The Bed file just listed start and end locations for the peaks and we simply expanded this window with `bedtools slop`. We actually need the sequences specified by our expanded ranges in order to provide the MEME analysis tool with the sequences as input. We'll use another tool from the Bedtools collection to accomplish this.

I am going to illustrate the commands for both our data and the published data here for the next few commands. You can see how our data comes out today using the MEME tool and then check back and see how it would come out with there data if you'd like.

```
bedtools getfasta -fi genome.fa -bed gal4_summits_more.bed -fo gal4_summits_more.fa
```

If you'd like to use the Rhee and Pugh 2011 data, the command is

```
bedtools getfasta -fi genome.fa -bed rhee_more.bed -fo rhee_more.fa
```

Expand your window in which you are working as tall as you can and then enter on command line

```
less gal4_summits_more.fa
```

If you'd like to use the Rhee and Pugh 2011 data, the command is

```
less rhee_more.fa
```

Now highlight in your terminal and copy the highlighted text to the clipboard.

Open your text editor.

Paste the text you copied into a text editor program and save the file as `gal4_summits_more.fa` or `rhee_more.fa`, respectively.

Now using the browser on your computer go to [here](#). Use the above file to upload it where it asks for Input the primary sequences. (Supposedly [here](#) is the most up-to-date version of the site. However, the Upstate network said it was unavailable or it violated policy and has been blocked when I submitted jobs there.)

Under `Select the number of motifs`, adjust the number of motifs to find to one.

Provide an e-mail address.

Click on 'Start Search' to submit your task. It will be a short wait. Your results will be available in your browser soon. No need to check email most likely.

Click on `MEME html output` and view the results by hitting blue arrow below the `More` column on the next screen.

How do your results match up with [the figure 3 MEME output logo from Rhee and Pugh, 2011](#)?

How do these results match up with [the structural basis of DNA binding by Gal4p](#)?

[The slides](#) will be used to discuss the above two questions in more depth.

Why not PRECISELY same (even when using their file)? We have to imagine we were coming at this naively and trying to identify a motif. (Or least be performing a proof of concept experiment as they were here.) This was obviously just an initial first pass to see if any motif is identified. One is. This would obvious warrant further examination. In this case, we'd also have to look and see that because of the amount of flanking sequence added to each site region, some of the sites identified in the ChIP-exo data overlap with neighboring sites in the sequences submitted to MEME or are left out because the peak identification in MACS2 we used seems to squelch out neighboring sites. Thus we have probably biased this rough first pass to the *best* ones when faced with overlap. Today was just meant to give you a flavor for the steps involved if you were trying to use this to search for a novel motif. In a thorough investigation of the motif there'd be subsequent steps and the results seen on the left side of [Figure 3 of Rhee and Pugh, 2011](#) are likely

the result of such further analysis. The authors may have restricted the sites they submitted down to say the best match within the distances they had identified with the peak pairs. Additionally, judging by the indication to the orientation of the gene, I think they may have used the advanced option settings in MEME to limit the search to given strand and provided the upstream sequence from the gene which is not what we did here. Additionally, as touched in numerous places, their approach to identify peak pairs and sites was more detailed than what we did here, and had additional replicates and filtering based on features (tRNA and rDNA) to confirm or rule out peaks, and so the upstream results before MEME were obviously different.

(Actually, I have used the data available in the [Rhee and Pugh Gal4 Bed file available from SGD](#) to eliminate the overlaps and found I still don't get an exact match to the [Figure 3 MEME output logo](#). It is *very, very* close. However, there is a further complication. [Figure 3](#) shows the sequence of 15 binding sites. If you count the lines of the actual data in the Gal4 Bed file, you'll see there is only 14. Likewise, the Gal4 data in the supplemental Excel file with the publication has a heading of *15 Gal4 binding sites*, but there are only 14 there. One of the Gal7 sites shown in [Figure 3](#) is absent in the Excel spreadsheet, and hence the Bed file because the Bed file header comments says the SGD curators used supplemental data to make the Bed file. Even adjusting for that and the orientation, plus setting the motif to have to be exactly 19 bps long, I didn't obtain exactly their motif logo, and so maybe due to a difference in other setting(s) or different software version.)

Sources

The sources for the information used today came from those linked throughout the content.

However, certain sources deserve special highlighting as they were particularly useful in developing this presentation, contain a wealth of related resources, or are especially pertinent at this stage.

- [README for MACS \(2.1.0\)](#)
- [Feng et al. 2012. Identifying ChIP-seq enrichment using MACS. Nat Protoc. 2012 Sep; 7\(9\): 10.1038/nprot.2012.101.](#)
- [Using MACS to Identify Peaks from ChIP-Seq Data. Curr Protoc Bioinformatics. 2011 Jun; CHAPTER: Unit2.14.](#)
- – [ChIP-Seq: technical considerations for obtaining high-quality data. Kidder BL, Hu G, Zhao K. Nat Immunol. 2011 Sep 20;12\(10\):918-22. doi: 10.1038/ni.2117. PMID: 21934668](#)
- [Titus Brown and Colleague's Next-Gen Sequence Analysis Workshops, most recent is \[Next-Gen Sequence Analysis Workshop \(2014\)\(<http://angus.readthedocs.org/en/2014/>\)](#) Particularly pertinent are:
 - the sections [Istvan Albert's 2012 ChIP-Seq lecture](#)
 - [Day 7: ChIP-seq: Peak Predictions and Cis-regulatory Element Annotations* Using MEME to identify TF binding motif from ChIP-seq data * here.](#)
 - [For the mapping of the reads using bwa I relied heavily on Mapping reads with bwa and bowtie tutorial](#)
 - [Software packages to install](#)
 - [Another recent guide to software installation](#)
- [ChIP- and DNase-seq data analysis workshop 2014](#)
- [UCSC Genome Bioinformatics- Data File Formats FAQ section](#)
- [Rhee HS and Pugh BF \(2011\) Comprehensive Genome-wide Protein-DNA Interactions Detected at Single-Nucleotide Resolution. Cell 147\(6\):1408-19](#)

Going forward

Look into

- Titus Brown and Colleague's Next-Gen Sequence Analysis Workshops, most recent are Next-Gen Sequence Analysis Workshop (2014) and Next-Gen Sequence Analysis Workshop (2015). Particularly pertinent to today's efforts are the sections Istvan Albert's 2012 ChIP-Seq lecture, Day 7: ChIP-seq: Peak Predictions and Cis-regulatory Element Annotations, Using MEME to identify TF binding motif from ChIP-seq data and here. However, there's lots of stuff to be found there for those trying to get up and running in bioinformatics in general. And for those ready for more, see Titus's coverage (chock full of links) of his week of classes offered first in 2015 for workshop alumni and advanced folks here.
- ChIP- and DNase-seq data analysis workshop 2014
- Cistrome - a web-based ChIP-seq analysis portal based on Galaxy open source framework run by the labs of Myles Brown and Xiaole Shirley Liu. It provides a complete workflow for ChIP-seq and downstream analysis without the need for local installation and configuration.
- A History of Bioinformatics (in the Year 2039), a presentation by Titus Brown encouraging good data practices in his unique way

Data Acquisition

- NCBI's Sequence Read Archive (SRA) hosts a great deal of raw data.
- As described here, NCBI's Gene Expression Omnibus (GEO) hosts processed forms of data and refers to the SRA for the raw data.
- European Nucleotide Archive (ENA) also hosts raw data but I found some select datasets slightly different from forms stored at SRA, i.e., split reads vs interleaved, etc..
- The Yeast Genome Database hosts a selection of curated datasets for *S. cerevisiae* here.

File Formats and Conversion help

- UCSC Genome Bioinformatics- Data File Formats FAQ section
- Data File format on Abecasis Group Wiki, for example SAM
- Ensembl's documentation on file formats
- Bam2Wig example
- File Formats Listing associated with the Broad Institute's Integrated Genomics Viewer

ChIP-seq data analysis

- README for MACS (2.1.0)
- Feng et al. 2012. Identifying ChIP-seq enrichment using MACS. Nat Protoc. 2012 Sep; 7(9): 10.1038/nprot.2012.101.
- Using MACS to Identify Peaks from ChIP-Seq Data. Curr Protoc Bioinformatics. 2011 Jun; CHAPTER: Unit2.14.
- ChIP-Seq: technical considerations for obtaining high-quality data. Kidder BL, Hu G, Zhao K. Nat Immunol. 2011 Sep 20;12(10):918-22. doi: 10.1038/ni.2117. PMID: 21934668

- Titus Brown and Colleague's Next-Gen Sequence Analysis Workshops, most recent is [Next-Gen Sequence Analysis Workshop (2014)](<http://angus.readthedocs.org/en/2014/>) Particularly pertinent are the sections Istvan Albert's 2012 ChIP-Seq lecture, Day 7: ChIP-seq: Peak Predictions and Cis-regulatory Element Annotations, Using MEME to identify TF binding motif from ChIP-seq data and here.
- ChIP-seq analysis using GALAXY: Part 1 - Introduction -uses MACS, CEAS, and WEEDER. The latter being an alternative to MEME for motif discovery.
- The ChIP-seq pipeline "Cistrome" integrated into UGENE uses MACS and CEAS and additional characterization of binding sites. You can read about it and the UGENE NGS platform here. The UGENE pipeline is originally based on the General ChIP-seq pipeline from the public Cistrome installation on the Galaxy workflow platform
- ChIP- and DNase-seq data analysis workshop 2014
- Cis-regulatory Element Annotation System by Hyunjin Shin and Tao Liu from Xiaole Shirley Liu's Lab
- ab initio motif finder MEME and the related MEME suite
- MEME-LaB wraps the popular ab initio motif finder in a web tool
- Motif enrichment tool. Blatti C, Sinha S. Nucleic Acids Res. 2014 Jul;42(Web Server issue):W20-5. doi: 10.1093/nar/gku456. Epub 2014 May 23. PMID: 24860165
- Motif-based analysis of large nucleotide data sets using MEME-ChIP

ChIP-exo methods

- Rhee HS and Pugh BF (2011) Comprehensive Genome-wide Protein-DNA Interactions Detected at Single-Nucleotide Resolution. Cell 147(6):1408-19
 - Several groups report good results and Illumina and other companies have kits and protocols
 - Development of an Illumina-based ChIP-exonuclease method provides insight into FoxA1-DNA binding properties
 - Webinar: "Precision Mapping"- High Resolution Mapping using ChIP-exo"
 - No formaldehyde, no problem for new genome-wide ChIP method
- ORGANIC detected more Reb1 binding sites than either X-ChIP-chip or ChIP-exo.

Questions

- Try Google, probably will lead you to one of my listed resources or..
- Biostars
- Stackoverflow for general scripting and computing
- SEQanswers - a high throughput sequencing community
- Try Twitter - for example this

Appendix: Help getting files on and off Amazon AWS EC2 instances

SCP

Limited to linux/Unix/Mac systems, but very straightforward.

- <http://angus.readthedocs.org/en/2014/amazon/transfer-files-between-instance.html>

Example that worked in the terminal on my Mac to download from my Amazon instance (back when public DNS/hostname formatted like `ec2-50-19-16-34.compute-1.amazonaws.com`).

```
scp -i workshop.pem ubuntu@ec2-50-19-16-34.compute-1.amazonaws.com:/usr/workshop/sc3.db .
```

The key is `workshop.pem`. And `ec2-50-19-16-34.compute-1.amazonaws.com` was my current instance hostname. The `.` specified to download the file `sc3.db` to the current working directory.

For newer Amazon EC2 instances (Summer 2015 forward->), that command will look more like

```
scp -i workshop.pem ubuntu@54.86.18.122:/usr/workshop/sc3.db .
```

Where `54.86.18.122` is the public IP address of the instance.

WARNING: You may observe errors like `scp: /root/my_directory/genome.fa: Permission denied` if you try to download/upload to places without proper permissions. Check the permissions (command `ls -l`) for learning which directories you can read and write to on your instance for using `scp` to upload/download, see [here](#) for interpreting the codes. (Basically you want `r` for downloading and `w` for uploading to be among the six characters on the right). For example, you cannot read or write direct to `root`. Aside from `root`, most directories created by the instance at creation, even those below `root` allow reading which makes downloading from most directories to a local drive possible using `scp`. Tip: for Amazon EC2 instances there is a `tmp` directory that allows both reading and writing. For uploading to your instance you can use that `tmp` as a target to go from your local machine to your instance. Then once uploaded, move the file or files to a more logical place within your instance.

See also:

- <http://stackoverflow.com/questions/6558080/scp-secure-copy-to-ec2-instance-without-password>
- <http://stackoverflow.com/questions/10364950/uploading-files-on-amazon-ec2>

SFTP

Most likely best choice if you are on Windows.

Pay attention to the user name. You want `ubuntu` for Ubuntu instances. Those links below should also provide information if you happen to be using other types of AWS EC2 instances. Be sure to read the comments too. They often give you nice pointers when you are troubleshooting.

with FileZilla

- [Example on Windows PC](#)
- [Example on Mac with more information at Yasitha Chinthaka's post here](#)
- [Another take on this is here](#)

with cyberduck

- [Example on Mac](#)

Dropbox

Looks like it still works since in 2014 course info

<http://angus.readthedocs.org/en/2014/amazon/installing-dropbox.html>

BUT BE WARY!!!

IMPORTANT: Dropbox will sync everything you have to your EC2 machine, so if you are already using Dropbox for a lot of stuff, you might want to create a separate Dropbox account just for the course.

Appendix: Making Wiggle file Covering Genome

One of the required files for the `build_genomeBG` utility included with the CEAS package is a wiggle file that describes the genomic locations that are valid for data selection. For example, a tiling array may only cover some parts of a genome. In our case, we theoretically should have covered all the *Saccharomyces cerevisiae* genome and so we simply generate a wiggle file that covers the full length of each and every chromosome.

For example a tiling array may only cover some parts of the genome. In our case we simply generate a wiggle file that covers the full chromosomes. Start will be one for each and every chromosome and the `step` will simply be the length of the chromosome. A convenient place to mine that information is to look at the header for the SAM file we generated earlier in our pipeline. The lines beginning `@SQ` have all this information. So you'd build of the lines below in your text editor using the length of the chromosomes to make your own file. Looking at the file you are making, the first few lines will be like so and continue on to cover all the chromosomes, included the mitochondria.

```
track type=wiggle_0
fixedStep chrom=chrI start=1 step=230218
1
fixedStep chrom=chrII start=1 step=813184
1
fixedStep chrom=chrIII start=1 step=316620
1
fixedStep chrom=chrIV start=1 step=1531933
1
... etc ...
```

This can actually be done right in the command-line using utilities like `nano` and `vim`.

For the sake of time today, the result is made available for you already [here](#).

Appendix: Obtaining a BED of Telomere sequences

Rationale

The Rhee and Pugh, 2011 paper used genomic features to filter out results from peak-calling. We want to do this as well.

A good approach seems:

- get list or lists as Bed file from YeastMine
- Uses Bedtools with `intersectBed -v` to filter out those summits that are in telomeres.

Obtain Telomeric sequences as a BED file from YeastMine

Go to YeastMine

Scroll down to where you see Lists.

The screenshot shows the YeastMine website interface. At the top, there are three main buttons: **SEARCH**, **ANALYSE** (with a text input field containing gene names like `msh1; msh5; mre11; xrs2; ndt80; tid1; sch1; pra2; apr1; doa2; rad54; cef1` and a link to `advanced`), and **TAKE A TOUR**.

Below these buttons is a navigation menu with tabs for: **GENOME**, **PROTEINS**, **FUNCTION**, **PHENOTYPES**, **INTERACTIONS**, **REGULATION**, **HOMOLOGY**, **EXPRESSION**, and **LITERATURE**.

Under the **GENOME** tab, there is a section titled "Query for genome:" with a list of options:

- Gene → Flanking features within a specific distance
- Chromosomal Region → All genes
- Feature Type → Features of a selected feature Type
- Gene → Protein Sequence
- Gene → Chromosomal location
- All genes of a selected Feature Type → Genes with introns
- Gene → Genomic DNA
- Gene → Transcripts

There is a link for `More queries` and a diagonal banner that says "popular templates".

Below this is a section titled **Lists**, indicated by a red arrow. It contains a list of genomic features:

- [ALL Verified Uncharacterized Dubious ORFs \(6604 Genes\)](#)
This List includes ALL ORFs
- [Centromeres \(16 Centromeres\)](#)
- [Long Terminal Repeat \(383 LongTerminalRepeats\)](#)
- [NotPhysicallyMapped \(6 NotPhysicallyMappeds\)](#)
- [RetroTransposons \(50 Retrotransposons\)](#)
- [Telomeres \(142 Telomeres\)](#)
- [Uncharacterized Verified ORFs \(5820 Genes\)](#)
This List excludes Dubious ORFs

There is a link for `More lists` at the bottom of this section.

At the very bottom of the page, there are two sections: **Perl, Python, Ruby and & Java API** (with a sub-link "Access our YeastMine data via our") and **News & Updates** (with a sub-link "Sharing the Health").

Among the Lists listed there should be Telomeres. If not, click on More lists and examine the list of lists.

Click on the Telomeres list link.

Click on the Download button.

SGD **YeastMine** Search and retrieve *S. cerevisiae* data with YeastMine, populated by SGD and powered by InterMine.
 Data Updated on: Apr-20-2015 Contact Us Video Tutorials Help Log in

Home Templates Lists QueryBuilder Tools Regions Data Sources API MyMine

Upload | View Search: GO

List Analysis for Telomeres (142 Telomeres)

Columns 1 Filters

List Code **Download**

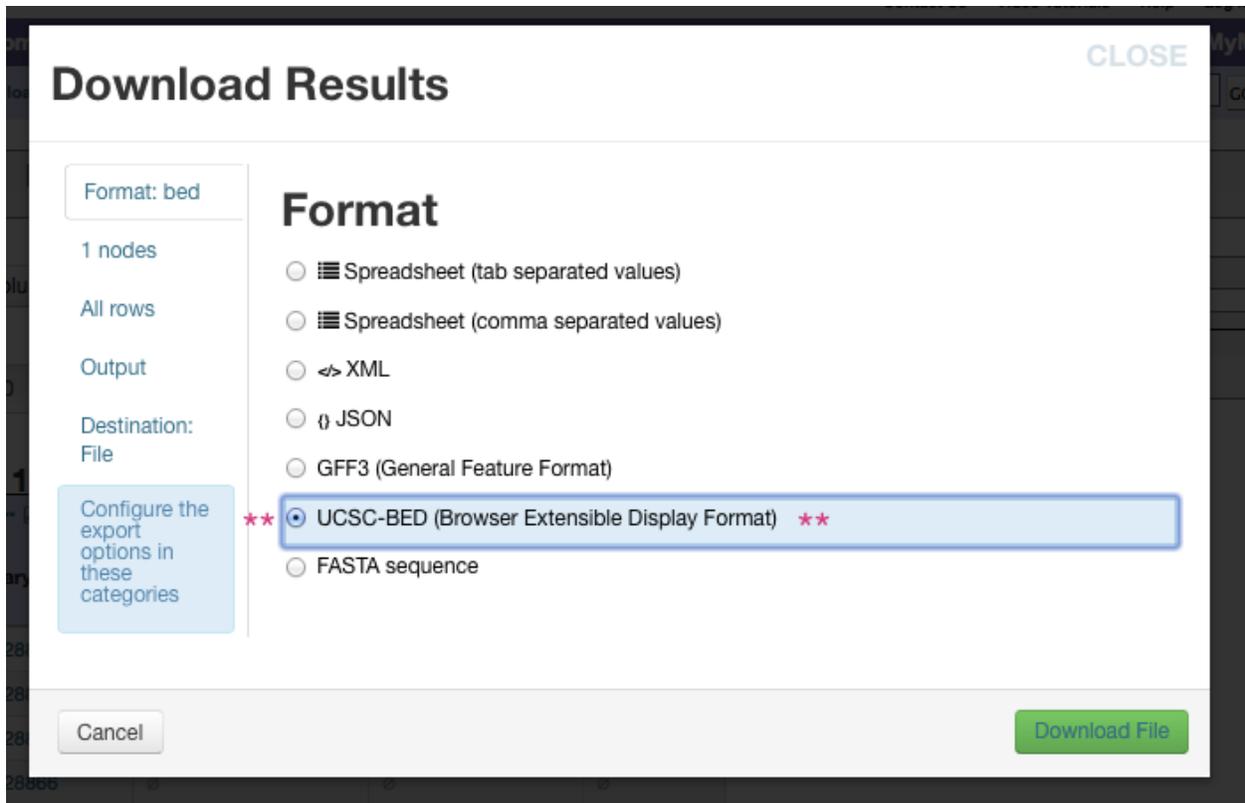
10 p. 1

External Links
No external links.

1 to 10 of 142 rows

Primary DBID	Systematic Name	Standard Name	Name
S000028862	TEL01L	∅	∅
S000028864	∅	∅	∅
S000028865	∅	∅	∅
S000028866	∅	∅	∅
S000028867	TEL02L	∅	∅
S000028868	∅	∅	∅
S000028869	∅	∅	∅

Choose UCSC-BED (Browser Extensible Display Format) from the choices.



Now either click green `Download File` button at the bottom to download to your local computer. You could then use `scp` to upload as illustrated [here](#) it to an Amazon EC2 instance, for working on the worksop analysis example.

Alternatively, click on the `Destination File` tab in the toolbar on the left side of the `Download Results` window.

Note that there is a URL that makes uploading to an EC2 instance on Amazon Web Services even easier. In fact you can click copy to clipboard and then paste after the following text once you ssh into your working directory on your instance. For example,

```
wget http://yeastmine.yeastgenome.org/yeastmine/service/query/results?format=tab&
↪start=0&token=ilr4ud70h133R7ned6x2&columnheaders=1&query=%3Cquery+model%3D%22genomic
↪%22+view%3D%22Telomere.primaryIdentifier+Telomere.secondaryIdentifier+Telomere.
↪symbol+Telomere.name%22+%3E%3Cconstraint+path%3D%22Telomere%22+op%3D%22IN%22+value
↪%3D%22Telomeres%22+code%3D%22A%22+%3D%2F%3E%3C%2Fquery%3E
```

Note that the above command is just an example for constructing your command. The API token in the URL is temporary and expired long ago.

Appendix: Technical guide to the workshop

A lot of software and supporting libraries are necessary on a machine to run the analysis used in the workshop. Many of these resources are useable for general analysis of modern high-throughput sequencing data.

In the hopes it will be useful for workshop participants looking to expand their knowledge and experience beyond what we could fit in today, the following sections spell out the details of setting up such a system on a Linux (Ubuntu Amazon Webs Services instance) and Mac. These were the two systems on which this workshop was developed and therefore the only ones for which I have first-hand knowledge to share. I apologize to those on Windows systems, but you may find these guides useful as general sign posts although you will have to look elsewhere for the details of installing.

Software Utilized

- Python
- BWA (Burrows-Wheeler Aligner)
- SAMtools OLD site: <http://samtools.sourceforge.net/>
- FASTX-Toolkit
- libgtextutils
- MACS2
- CEAS
- BEDtools
- MEME webserver

Suggested Other Software

- FastQC ← Not currently used here but good to have for initial assessment of reads. We will cover quality control throughout but not assess here ourselves.

Appendix: Setting up a Mac for the workshop

This was current Spring 2015 and done on a Mac running Mavericks. Keep in mind things may change subsequently.

Caveat: I found two steps not to work on my work iMac and I strongly suspect at least one of those cases is blocked by the Upstate's network.

These are my notes concerning installing many of the bioinformatic software packages on a Mac. This is basically my own, Mac-based counterpart to the [ANGUS listing of software packages to install on Linux machines from 2013](#). Some parts are more detailed than others. Generally those that were more difficult than a normal Mac installation where you download in your browser and click on it in the finder are described.

General software from previous installations prior to 2015

Many of these adapted from http://ged.msu.edu/angus/tutorials-2012/bwa_tutorial.html, updating links as I went.

- BWA (Burrows-Wheeler Aligner)
- Bowtie2
- SAMtools OLD site: <http://samtools.sourceforge.net/>
- FastQC
- IGB
- IGV

Added specifically for developing ChIP-seq workshop pipeline April 2015:

- FASTX-Toolkit
- SRA Toolkit
- libgtextutils
- MACS2 and here
- CEAS
- BEDtools

Installing notes for prior installations

Mainly the prior ones were from yeast analyses on work iMac

BWA

Waynes-iMac:~ Wayne\$ mkdir bioinf Wayne-iMac:~ Wayne\$ cd bioinf

next line updated from <http://sourceforge.net/projects/bio-bwa/files/>

```
curl -O -L http://sourceforge.net/projects/bio-bwa/files/bwa-0.7.5a.tar.bz2
tar xvfj bwa-0.7.5a.tar.bz2
cd bwa-0.7.5a
make
sudo cp bwa /usr/local/bin
```

IT WILL ASK FOR PASSWORD HERE

```
cd ../
```

FastQC

According to <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/INSTALL.txt>, I wanted the zipped file of FastQC to be able to run it on command line, EVEN for Mac OS. Note that the Mac OS GUI version (from '.dmg' download) does load even gzipped fastq files and the report can be saved to give the same thing the command line does and so you can do it via a more typical installation and run it not on the command line if you'd prefer for a Mac; I don't know about Linux GUI options for this program for installing and running on local machines. So I downloaded it, unzipped, and now I need to give it permissions to run as executable from command line, following http://ged.msu.edu/angus/tutorials-2012/fastqc_tutorial.html:

```
cd ../
cd Downloads/
cd FastQC/
chmod +x fastqc
```

Note that the GUI version (from '.dmg' download) does load even gzipped fastq files and the report can be saved to give the same thing the command line does.

IGV and IGB

Downloaded IGB (intergrated genome browser) Downloaded IGV (intergrated genome viewer)

SAMtools and Bowtie2

So I started reviewing how to download: <http://sourceforge.net/projects/samtools/files/samtools/0.1.19/samtools-0.1.19.tar.bz2> (I note on <http://ged.msu.edu/angus/2013-04-assembly-workshop/installing-software.html> that in April 2013, Titus was getting Samtools from github.) Looking ahead, I did the same for bowtie. Seems bowtie is at <http://bowtie-bio.sourceforge.net/index.shtml> and bowtie2 is out that is supposed to be better for long reads, i.e., those over 50 bp. Since the ones from the mitochondrial landscape paper are over 50, maybe I should try and get bowtie2 working. See → <http://bowtie-bio.sourceforge.net/bowtie2/index.shtml>. Download from http://downloads.sourceforge.net/project/bowtie-bio/bowtie2/2.1.0/bowtie2-2.1.0-macos-x86_64.zip. In april 2013, Titus was installing both <http://ged.msu.edu/angus/2013-04-assembly-workshop/installing-software.html>. Installed these two:

```
cd samtools-0.1.19
make
sudo cp samtools /usr/local/bin
cd ../
cd bowtie2-2.1.0
```

[MAKE FAILED BUT LOOKS LIKE SINCE I DOWNLOADED MAC SPECIFIC ONE, THAT IT IS ALL SET. JUST NEED TO COPY EXECUTABLE TO PATH]

```
sudo cp bowtie2 /usr/local/bin
```

However, executing it gave an error

```
Error: Expected bowtie2 to be in same directory with bowtie2-align:
```

Searching that error, I found <http://seqanswers.com/forums/showthread.php?t=29727> which lead to <http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml#adding-to-path> that said to add all the executables to your path "make sure that you copy all the executables, including bowtie2, bowtie2-align, bowtie2-build and bowtie2-inspect." So finished by,

```
sudo cp bowtie2-align /usr/local/bin
sudo cp bowtie2-build /usr/local/bin
sudo cp bowtie2-inspect /usr/local/bin
```

NOW IT WORKS WHEN I TYPE 'bowtie2'

More recent installations

Added specifically for ChIP-seq pipeline May 2015

FastX Toolkit

Download FASTX-Toolkit from http://hannonlab.cshl.edu/fastx_toolkit/download.html Also downloaded libgtextutils-0.7.tar.gz from there as well. THIS ONE NEEDS TO BE INSTALLED FIRST. (See [Software packages to install](#) and [Installing FastX on Mac](#) for help on Linux and Macs, respectively. Note on the Mac, I always seem to need to change the from the ./configure && make && make install Titus uses in Unix/Linux to ./configure && make && sudo make install for the Mac. I did the first four lines of the Mac approaches by hand and I was having trouble getting the Linux commands working on the Mac because I kept getting errors installing libgtextutils, which lead to errors installing pkg-config, but finally got both to work (see detailed notes and steps below).)

Unzipped and moved to my bioinf folder.

Items to note about the next steps:

- libgtextutils NEEDS TO BE INSTALLED FIRST!! The FASTX-Toolkit relies on this and seems to look for related items during installation.
- You have to install `pkg-config` and edit your `PATH` using the text on lines 6 and 7 in part II of the instructions [Installing FastX on Mac](#) involving setting the path (see page 87 of Haddock and Dunne's Practical Computing for Biologists; although doing it a different way here!!!) AS ONE LINE ON COMMAND LINE or you'll get this error below:

```
checking for GTEXTUTILS... configure: error: in '/Users/Wayne/bioinf/fastx_toolkit-0.0.14': configure: error:
The pkg-config script could not be found or is too old. Make sure it is in your PATH or set the PKG_CONFIG
environment variable to the full path to pkg-config.
```

Alternatively, you may set the environment variables `GTEXTUTILS_CFLAGS` and `GTEXTUTILS_LIBS` to avoid the need to call `pkg-config`. See the `pkg-config` man page for more details.

To get `pkg-config`, see <http://pkg-config.freedesktop.org/>. See 'config.log' for more details
Waynes-iMac:fastx_toolkit-0.0.14 Wayne\$ make make: *** No targets specified and no makefile found. Stop. Wayne\$
Waynes-iMac:fastx_toolkit-0.0.14 Wayne\$

In part 2 of [Installing FastX on Mac](#), lines six and seven which need to be combined into one command to read:

```
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH
```

- According to [here](#) and [here](#) `MacPorts` is supposed to make installing `pkg-config` easy. However, after going through the long version of installing `MacPorts`, which needs the Apple developer tools (to [install Xcode and the developers tools](#) or [install developers tools on your Mac](#)). I couldn't get it to work even carefully following [these instructions](#). But I noticed there was an 'install' document in the `pkg-config` folder I downloaded and unzipped from [here](#) and it mentioned the usual `./configure && make && make install` steps so I tried `./configure && make && sudo make install`. Seemed to work but hit an error about 'glib' but in the error report it said I could tell it to use internal glib with an option and that worked. The exact text was or pass `--with-internal-glib` to `configure` to use the bundled copy.

The actual, worked out additional steps for installing the FASTX-Toolkit on Mac OS X (Mavericks, presently).

```
Download the latest version of pkg-config from http://pkgconfig.freedesktop.org/
↳ releases/
Unzip it.
Open terminal, navigate to the folder you unpacked, and issue the command:
```

```
./configure --with-internal-glib && make && sudo make install
```

```
Now you are finally ready to finish the steps for libgtextutils and fastx_toolkit_
↳ installing
```

```
Navigate to the directory with `libgtextutils`. Here is the record of what I did. You_
↳ want to run everything after the $ in your own terminal.
```

```
Waynes-iMac:bioinf Wayne$ cd libgtextutils-0.7/
Waynes-iMac:libgtextutils-0.7 Wayne$ ./configure && make && sudo make install
Waynes-iMac:libgtextutils-0.7 Wayne$ cd ../
Waynes-iMac:bioinf Wayne$ cd fastx_toolkit-0.0.14/
Waynes-iMac:fastx_toolkit-0.0.14 Wayne$ export PKG_CONFIG_PATH=/usr/local/lib/
↳ pkgconfig:$PKG_CONFIG_PATH
Waynes-iMac:fastx_toolkit-0.0.14 Wayne$ ./configure && make && sudo make install
```

SRA Toolkit

SRA toolkit downloading

Downloaded Mac version from [here](#) and unpacked in finder.

Then followed [SRA Toolkit Installation and Configuration Guide](#) exactly.

Specifically, I made a folder in my schmitlabwayne user folder that I named sra-toolkit.

I dragged all the files in the /bin folder in the unpacked sratoolkit.2.4.5-2-mac64 folder into that folder I jsut named sra-toolkit. Ran initial command to see if I was on righ track

```
~/sra-toolkit/fastq-dump
```

It yielded the fastq-dump manual page. Sofar it is looking promising. (But I got that far on work computer.)

Next, as a full test I issued the command below in the terminal

```
~/sra-toolkit/fastq-dump -X 5 -Z SRR390728
```

This worked. Output:

```
Read 5 spots for SRR390728
Written 5 spots for SRR390728
@SRR390728.1 1 length=72
CATTCTTCACGTAGTTCTCGAGCCTTGGTTTTTCAGCGATGGAGAATGACTTTGACAAGCTGAGAGAAGNTNC
+SRR390728.1 1 length=72
;;;;;;;;;;;;;9;;665142;;;;;;;;;;;;;96&&&& (
@SRR390728.2 2 length=72
AAGTAGGTCTCGTCTGTGTTTTCTACGAGCTTGTTCCAGCTGACCCACTCCCTGGGTGGGGGGACTGGGT
+SRR390728.2 2 length=72
;;;;;;;;;;;;;4;;3;393.1+4&&5&&;;;;;;;;;;;;;;<9;<;464262
@SRR390728.3 3 length=72
CCAGCCTGGCCAACAGAGTGTACCCCGTTTTTACTTATTATTATTATTATTGAGACAGAGCATTGGTC
+SRR390728.3 3 length=72
-;;8;;;;;;;;;*;;'-4,44;,:&,1,4'./&19;;;;;;;;;669;;99;;;;;;;;;-3;2;0;+;7442&2/
@SRR390728.4 4 length=72
ATAAAATCAGGGGTGTTGGAGATGGGATGCCTATTCTGCACACCTTGGCCTCCCAAATTGCTGGGATTACA
+SRR390728.4 4 length=72
1;;;;;;;;;4;3;38;8%&,;)*;1;;)/%4+;,1;;);;;;;;;;;;4;(1;;;24;;;41-444//0
@SRR390728.5 5 length=72
TTAAGAAATTTTGCTCAAACCATGCCCTAAAGGGTCTGTAATAAATAGGGCTGGGAAAACCTGGCAAGCCA
+SRR390728.5 5 length=72
;;;;;;;;;;;;;9445552;;;;;;;;;;;;;446662
```

Success! See terminal console accompanying facile installation of sra toolkit on home Mac for the actual few commands run on terminal and output.

So far installation on computer at work has not resulted in an SRA toolkit able to output a fasta formatted file. It gets hung up with the SRA file in the cache.

MACS2

When I search macs2 I found it at <https://pypi.python.org/pypi/MACS2>. The site being pypi.python.org indicated to me that I should be able to use the package manager pip on my Mac to easily download and install.

That was the case, I simply typed in my Mac's terminal:

```
pip install macs2
```

Sanity check for verifying installation

```
macs2 -h
```

Display usage information so all is good.

CEAS

Downloaded [here](#).

Unpacked in Finder and moved contents of unpacked folder to a newly created folder called CEAS that I made within my standard bioinformatics working directory.

In terminal navigated to that folder where unpacked and ran

```
sudo python setup.py install
```

It unpacked. Writing /Users/Wayne/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-packages/CEAS_Package-1.0.2-py2.7.egg-info

Sanity check. Typed in terminal

```
ceas
```

Reported back about program and options and so it works.

CEAS's build_genomeBG utility needs to access external databases so in an attempt to get it to work on work iMac, I tried `pip install MySQL-Python` after reading a few places how to install it.

No luck so far. When trying to do `pip install MySQL-Python` keep getting

```
EnvironmentError: mysql_config not found
```

AHA!! I at least got past the `EnvironmentError: mysql_config not found` problem, but I had to follow ALL of the first part of user3429036's answer (at the top where he assumed you tried all that) at <http://stackoverflow.com/questions/25459386/mac-os-x-environmenterror-mysql-config-not-found>. Below are those specific steps.

[I didn't touch Python because already installed.] So I had to first install homebrew (so homebrew used to install mysql) in his approach.

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/↵install)"
```

Then I installed mysql which is where I think I was having problems. I had not not done that before this.

```
brew install mysql
```

Then I issued the command in the terminal

```
export PATH=$PATH:/usr/local/mysql/bin
```

Finally I did

```
pip install MySQL-Python
```

Progress. No more `EnvironmentError: mysql_config not found`

But when I run `build_genomeBG` I get

```
`_mysql_exceptions.OperationalError: (2003, "Can't connect to MySQL server on 'genome-
↳mysql.cse.ucsc.edu' (60)")`
```

This is progress because I was getting

```
/Users/Wayne/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-packages/CEAS/
↳inout.py:64: UserWarning: sqlite3 is used instead of MySQLdb because MySQLdb is not_
↳installed
  warnings.warn("sqlite3 is used instead of MySQLdb because MySQLdb is not installed")
CRITICAL @ Mon, 27 Apr 2015 11:15:55: MySQLdb package needs to be installed to use_
↳UCSC or a local sqlite3 db file must exist.
CRITICAL @ Mon, 27 Apr 2015 11:15:55: Check -g (--gdb). No such file or species as
↳'sgdGene'
```

But the `build_genomeBG` doesn't function. I don't think that is the installation. ***** Is it possible it is blocked by network at work????? ***** (Like I suspect SRA was?) Following from <http://genome.ucsc.edu/goldenpath/help/mysql.html> lead me to the Google groups which I searched. Lead me to ...

from [https://groups.google.com/a/soe.ucsc.edu/forum/#!searchin/genome/an\protect\T1\textdollar27\protect\T1\textdollar20connect\protect\T1\textdollar20to\protect\T1\textdollar20MySQL\protect\T1\textdollar20server\protect\T1\textdollar20on\protect\T1\textdollar20\protect\T1\textdollar27genome-mysql.cse.ucsc.edu\protect\T1\textdollar27\genome/QISWjaGaZq4/hLdo8hbcqHEJ](https://groups.google.com/a/soe.ucsc.edu/forum/#!searchin/genome/an%5Cprotect%5C%27%5Cprotect%5C%20connect%5C%20to%5Cprotect%5C%20MySQL%5C%20server%5C%20on%5Cprotect%5C%20genome-mysql.cse.ucsc.edu%5C%27%5Cgenome/QISWjaGaZq4/hLdo8hbcqHEJ) > If you're attempting to connect from an institution, it's possible that your IT staff have external mysql connections blocked by default for security reasons.

For trouble shooting I tried on command line

```
mysql --user=genome --host=genome-mysql.soe.ucsc.edu -A
```

Ughh!

```
Waynes-iMac:bioinf Wayne$ mysql --user=genome --host=genome-mysql.soe.ucsc.edu -A
ERROR 2003 (HY000): Can't connect to MySQL server on 'genome-mysql.soe.ucsc.edu' (60)
Waynes-iMac:bioinf Wayne$
```

Seems we are blocked at Upstate????

Seems we are blocked at Upstate???? Consistent with my concern this is the case and why this (and probably SRA toolkit) fail is that that running `build_genomeBG` on a Ubuntu instance on AWS worked great.

```
build_genomeBG -d sacCer3 -g sgdGene -w sacCer3.wig -o sc3.db
```

Command above on Amazon AWS resulted in generating `sc3.db`. I downloaded it from there to have on local Mac.

BEDtools

Following [here](#) at first it looked like this would be an easy installation. I recently installed Homebrew on the Mac and it looks like I can just use that to manage the installation.

```
brew install bedtools
```

Unfortunately, despite what the nice looking installation instructions seemed to indicate, that failed

```
Waynes-iMac:bioinf Wayne$ brew install bedtools
Error: No available formula for bedtools
Searching formulae...
Searching taps...
homebrew/science/bedtools
```

```
Waynes-iMac:bioinf Wayne$ bedtools
-bash: bedtools: command not found
Waynes-iMac:bioinf Wayne$ port install bedtools
Error: Insufficient privileges to write to MacPorts install prefix.
Waynes-iMac:bioinf Wayne$ sudo port install bedtools
Password:
Error: Port bedtools not found
Waynes-iMac:bioinf Wayne$ sudo brew install bedtools
Error: Cowardly refusing to `sudo brew install`
You can use brew with sudo, but only if the brew executable is owned by root.
However, this is both not recommended and completely unsupported so do so at
your own risk.
Waynes-iMac:bioinf Wayne$
```

Looking under the section [Compiling from source via Google Code](#), lead me to the [Google code site](#) to look up the version number. That site said it was no longer to be hosted there as of end of 2103 and has [moved to Github](#). Looking back, in fact the Github directions for installing are at the bottom, although it is disfavored by saying it is the development version.

Downloaded the zipped file from [here](#) and unzipped it using finder.

Then ran make in terminal.

```
cd bedtools2-master
make
```

Then finishing installation following method under [Compiling from source via Google Code](#) copied files to `/usr/local/bin`.

```
sudo cp ./bin/* /usr/local/bin
```

Sanity check. Typing

```
bedtools
```

on command line gave usage information, and so it seems successfully installed.

Appendix: Setting up an Amzon EC2 instance for the workshop

This was current Spring 2015. Keep in mind things may change subsequently.

Building the Amazon EC2 instance

You'll find how I set up these systems for the workshop below, which was adapted from information mainly [here](#), [here](#), and [here](#).

Keep in mind in an effort to make things go more smoothly today, I eliminated a step that users on Windows operating system usually have to perform. If you ever find yourself connecting to an Amazon EC2 instance on Windows and need to generate a ppk file from your pem file, see [here](#)

record of software installation for NGS work on Amazon AWS

These are my notes concerning installing many of the bioinformatic software packages on a Linux AWS machines. This is basically my own counterpart to the [ANGUS](#) listing of software packages to install on Linux machines from

2013 and Technical guide to the 2014 ANGUS course. Developed during preparation of machines for May 2015 workshop.

List of installed software

- basics like screen, git, curl, gcc compiler, python, and r, etc.
- BWA (Burrows-Wheeler Aligner)
- Bowtie2
- SAMtools OLD site: <http://samtools.sourceforge.net/>
- FastQC <- NOT IN ACTUAL LIST HERE YET **EDIT OUT FOR POSTING??**
- FASTX-Toolkit
- SRA Toolkit
- libgtextutils
- MACS2 and here
- BEDtools
- CEAS

Machine preparation for April-May 2015

Largely adapted from [Starting up a custom operating system guide for ANGUS course](#) and [Technical guide to the 2014 ANGUS course](<http://angus.readthedocs.org/en/2014/amazon/technical-guide.html>)

I think between 2013 and 2014, Titus moved the ANGUS course from using the Starcluster AMIs to setting up the base Ubuntu system offered on Amazon AWS, mainly relying on the `apt-get` package manager to facilitate this process.

Registering an instance

- Logged into my Amazon AWS account.
- In console, picked 'Key Pairs' from under `Network & Security`.
 - Created a new key pair so I can only use that for workshop. Called it `workshop`.
 - Saved the `.pem` file to my `Downloads` directory on my Mac when prompted.* Copied it to my folder for the workshop in Dropbox.
- Followed the directions [here](#) to set up `Ubuntu Server 14.04 LTS (PV)` using the `workshop` key I had just made.
 - Note I tried to set up `ami-7606d01e` but searching it in `community AMIs` failed to yield anything.
 - It is important to set up the (PV) version that is way down on the list of `Quick Start` choices and not the (HVM) version that is near the top of the list. Otherwise you won't have the option to select `m1.xlarge`.
 - I decided to go with `m3.xlarge` and not the ANGUS-recommended `m1.xlarge` because since this will be a small dataset once demultiplexed I didn't have that many Gb to store on the machine and `m3.xlarge` had more compute units and is a little cheaper per hour. See [here](#) for the details of each.

After (1) noticing I lost data in /mnt when I stopped and restarted (I thought /mnt was where we worked in Titus's class in the ANGUS-recommended m1.xlarge); and (2) reading this-> <https://4sysops.com/archives/how-to-check-if-your-ec2-instance-uses-ssd/>, I started thinking going with m3.xlarge was a bad idea because the drives are SSD and therefore only ephemeral. PLUS IT SEEMS I HAD TO SPECIFY I WANTED THEM ACCESSIBLE IN 'ADD STORAGE' SECTION?!?! I DO ONLY SEEM TO SEE ONE BECAUSE I SKIPPED THAT?? Given I skipped that in add storage and this Jason McCreary's answer at <http://serverfault.com/questions/490597/wheres-my-ephemeral-storage-for-ec2-instance>, I cannot understand why I even have access to one of them. (Maybe since those two things, Amazon fixed it so at least one of them there be default. YES!! When I making new ones I noticed the instance store 0 one is listed by default but you have to specifically add the next ones even if the price guide says you have them. You still have to actively enable any beyond the first one here.) Maybe I should try next with m1.xlarge and see if my data in /mnt survives stopping and restarting and if I can see all I was told I should have. I MADE A NEW INSTANCE. Under 'add storage' I added all four 'instance storage' 0 thru 3 I had listed as options. And then when I ran `sudo fdisk -l` in my instance I could see them all unlike before when I didn't add. However, when I do `df -h` I still only see one of the four. It seems one always defaults to /mnt even if I don't mount it and that is all I get unless I mount them?

In light of this, I launched a brand new instance, with m3.xlarge, but I set the root to 30 Gb on the step # Step 4: Add Storage step. On this one I plan to install the software and the experimental data and verify it has enough space and that it keeps the data when it is stopped and restarted.

- Choose us-east-1d for availability zone. (I meant to do this for the newest one but I missed it and defaulted to 1a.)
- selected my quicklaunch-0 security group rules which has SSH, HTTP, and HTTPS included.
- Used the workshop key I had just made.

Logging in.

Follow [here](#) if you are on a Mac or Linux local machine. If your computer is a Windows PC, see [here](#). Note you have already been provided the key in .ppk form, and so you want to skip to the Logging into your EC2 instance with Putty section. BELOW IS MY MAC INFO.

```
* Protected my key on my computer from other users on this machine.

    chmod og-rwx workshop.pem

* Used `ssh` to log on, replacing the `ec2-??-???-???-?.compute-1.amazonaws.com`
↳network name part with the similar information from when you initiated your
↳instance on the AWS console.

    ssh -i workshop.pem ubuntu@ec2-??-???-???-?.compute-1.amazonaws.com
    ssh -i workshop.pem ubuntu@ec2-50-19-16-34.compute-1.amazonaws.com

* Now, once connected, to log on as super user, I issued following two commands.

    sudo bash
    cd /root
    (use cd /usr/workshop when working on workshop analysis steps)

The [first command](http://askubuntu.com/questions/57040/what-is-the-difference-between-su-sudo-bash-and-sudo-sh) restarts the bash shell with you using as the
↳super user and the second sets you in the home directory of the super user.

* To check out what we have you can type the command below to see
```

```
df -h
```

About half the `/dev/xvda1`` is filled with the system and installed software. We'll ↪ soon add more and our data there. The `/mnt`` directory and is essentially the ↪ scratch space for our AWS EC2 instance. It will go away if the instance is stopped ↪ so we'll stay in `/dev/xvda1`` so we don't have to keep adding our data in case we ↪ need to put the instance in `stop/pause`` mode.

- Exiting

To log out, type:

```
exit
logout
```

or just close the terminal or Putty window. (You cannot do this step wrong because ultimately you (or me, for today) have control of the instance in Amazon Web Services console.)

Preparing the instance for use

Followed [here](#) and MAINLY [here](#) to get started by putting on a lot of the basic software and some special bioinformatics ones.

```
apt-get update
apt-get -y install screen git curl gcc make g++ python-dev unzip \
    default-jre pkg-config libncurses5-dev r-base-core \
    r-cran-gplots python-matplotlib sysstat python-pip \
    ipython-notebook
```

(Oddly, second time I did this when setting up an instance with 30 Gb storage in root, I had trouble [triggered an error about holding broken packages at one time when pasting the above command all at once. I had to do line by line of the `apt-get -y install` command above. Then it worked fine. I recall the ANGUS course documentation had warned about this command can be tricky to paste right. I had edited my version some myself and maybe I disrupted something about it?)

As described [here](#) the first command resynchronize the package index files from their sources. The `y` option on the second line, the `install` command, says to assume answering `yes` to any prompts and helps speed things up but not needing the user to do anything.

Installed more specific software. Most is easy to install so I issued

```
apt-get -y install samtools bedtools bwa fastx-toolkit python-mysqldb
pip install macs2
```

The details of building this list is found below.

Installation notes for NGS software

Many of these adapted from http://ged.msu.edu/angus/tutorials-2012/bwa_tutorial.html, <http://ged.msu.edu/angus/2013-04-assembly-workshop/installing-software.html>, and <http://angus.readthedocs.org/en/2014/amazon/technical-guide.html>, updating as needed for May 2015 workshop.

Looks like Titus has moved from older method of installations that involved a lot of `configure` and `make` and `make install` commands or `make` followed by copying the contents of `/bin` directories to `/usr/local/bin` (see [here](#) for example) to using a package manager on the Ubuntu systems. Since I am trying to set up machines for April-May 2015 now, I

am going try to change things over to that. (I may leave some old notes I worked out.) See the links above for guidance along the lines the older methods.

SAMtools

For Ubuntu

```
apt-get install samtools
```

Bedtools

For Ubuntu

```
apt-get install bedtools
```

BWA

Looks like according to [here](#) maybe apt-get can install it.

```
sudo apt-get install bwa
```

Worked.

Other information I found, besides the Mac installation info, is [here](#) and [here](#) and [here](#)

FastQC

NOT DONE YET ON UBUNTU!!!

According to <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/INSTALL.txt>, I wanted the zipped file of FastQC to be able to run it on command line, EVEN for Mac OS. Note that the Mac OS GUI version (from '.dmg' download) does load even gzipped fastq files and the report can be saved to give the same thing the command line does and so you can do it via a more typical installation and run it not on the command line if you'd prefer for a Mac; I don't know about Linux GUI options for this program for installing and running on local machines. So I downloaded it, unzipped, and now I need to give it permissions to run as executable from command line, following http://ged.msu.edu/angus/tutorials-2012/fastqc_tutorial.html:

```
cd ../
cd Downloads/
cd FastQC/
chmod +x fastqc
```

Note that the GUI version (from '.dmg' download) does load even gzipped fastq files and the report can be saved to give the same thing the command line does.

Bowtie2

FastX Toolkit

Items to note about the next steps:

- libgtextutils NEEDS TO BE INSTALLED FIRST!! The FASTX-Toolkit relies on this and seems to look for related items during installation.
- FastX Toolkit also needs pkg-config but it looks like that is installed already in ami-7606d01e, and so that should be all set

Note for UBUNTU system, preferable way is to let package manager handle this and so looks like I can just use apt-get. See [here](#)

```
sudo apt-get install fastx-toolkit
```

WORKED and seemed to install the dependencies at the same time automatically.

In fact, the Hannon lab site has a link to [the installation instructions for Ubuntu and Debian](#) right on the [download and installation page](#) and the first suggestion is to use APT to get the pre-requisites and then lists commands to install libgtextutils first and then FastX Toolkit.

Alternatively for other Unix systems, someone nicely posted a link the full manual installation for CentOS [here](#) in response to [someone posting about the same errors I was seeing when trying to complete installation on my Mac](#) and this was helpful as a guide to the Mac installation as well.

SRA Toolkit

SRA toolkit downloading

Ubuntu Linux version

Best - get up to date version

First go to '~' directory in your instance. /mnt is the scratch disk space for Amazon machines but we are going to unpack the software in the root directory so it remains there when instance stopped. This will allow us to stop the instance to save money when not actively in use.

```
cd ~
```

Follow [here](#)

While in home directory (cd ~/), start with step #2 Download the Toolkit from the SRA website. You can get the link to use in the wget command by using a computer that had a browser and browsing to <http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current>. I saw in the list that one began with u so I clicked on that to verify it was ubuntu and copied the last part to combine with example in step 2 to replace Centos version with Ubuntu version download.

```
wget "http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.  
→tar.gz"
```

Unzip download (step #1 under Unpack the Toolkit)

```
tar -xzf sratoolkit.current-ubuntu64.tar.gz
```

Deleted download to clean up. (Optional)

```
rm sratoolkit.current-ubuntu64.tar.gz
```

Renamed directory to make building commands easier. (Optional but subsequent commands have paths assuming you did it. Change to match your directory hierarchy.)

```
mv sratoolkit.2.4.5-2-ubuntu64/ sratoolkit
```

Ran command

```
./sratoolkit/bin/fastq-dump
```

Gave me usage information. Looked promising.

Tried test recommended at [SRA Toolkit Installation and Configuration Guide](#) page.

```
./sratoolkit/bin/fastq-dump -X 5 -Z SRR390728
```

They say:

the test should connect to NCBI, download a small amount of data from SRR390728 and the reference sequence needed to extract the data, and stream the first 5 spots of the file (“-X 5” option) to the screen (“-Z” option).

If successful you should see a bit of data as they describe. It will also create an `ncbi` directory within my directory and that had `SRR390728.sra.cache` under the directory `~/ncbi/public/sra`.

`apt-get`

I STRONGLY ADVISE NOT USING THIS APPROACH!!! (directions only placed here to document what was tried and in hope eventually it is this easy.) I TRIED AND FOUND THIS DOWNLOADED AN OLD VERSION (fastq-dump was version 2.1.7 and there was no `prefetch` in `/bin`) I COULDN'T SEEM TO GET TO WORK. Can use `apt-get` according to [here](#) and [here](#), but [here](#) says not to do it this way as it will be old. I am going to try `apt-get` route and see if works for what I need. (IT INDEED DID NOT WORK FOR ME AS THE GENOMESPOT BLOG ADVISED.)

```
apt-get install sra-toolkit
```

I STRONGLY ADVISE NOT USING THIS APPROACH!!! SEE ABOVE.

MACS2

When I search `macs2` I found it at <https://pypi.python.org/pypi/MACS2>. The site being `pypi.python.org` indicated to me that I should be able to use the package manager `pip` once installed on Ubuntu to easily download and install.

```
pip install macs2
```

CEAS

Acquiring from [here](#)

```
wget http://liulab.dfci.harvard.edu/CEAS/src/CEAS-Package-1.0.2.tar.gz
```

Unpacking and installing, following [here](#)

```
tar xvf CEAS-Package-1.0.2.tar.gz
rm CEAS-Package-1.0.2.tar.gz
cd CEAS-Package-1.0.2/
```

```
python setup.py install
```

Sanity check.

```
ceas
```

Listed usage and so it worked.

CEAS's `build_genomeBG` utility needs to access external databases so I added `python-mysqldb` to the `apt-get` installation commands, similar to advised [here](#). (Actually, when I did that command after having instance already running but having not run it before it said it was already installed. Maybe something else I had already listed was dependent on it.)

MEME

Not available via `apt-get`.

Use on webserver [here](#). (Supposedly [here](#) is the most up-to-date version of the site. However, the Upstate network said it was unavailable or it violated policy and has been blocked when I submitted jobs there.)