***Highly recommend that the list of sample names sent to the sequencing center contain names that have the same length as this will be used to label the files. This simplifies things later on.***

**Using the pipeline to process sequencing data.**

All files should be on Github (files may need to small amounts of editing for your particular sequences)

To get the most current versions type *git pull* .

To upload changes you have made, type *git commit –am “Describe my changes”*.

If you haven’t *git pulled* recently, *git commit* may not work.

If you have no files at all, then do *git clone* [*https://github.com/chapkinlab/sequencing-pipeline.git*](https://github.com/chapkinlab/sequencing-pipeline.git) .

1. Sequencing files should have been received on an external hard drive or downloaded from a server. Place these files on the NAS in a folder that is descriptive but does not use spaces (Linux has issues). Use - or \_ in the folder name. Be sure to check the checksums if available (this is key).

To get files, be in the directory you want them places then:

*wget –user=Chapkin –password=pswrd location.of.files.downloading*

Use *md5sum filename* or *sum filename* depending on the type of checksum. You may need to contact the core facility.

1. Place a **copy** onto the computer that you plan to use for the analysis (oracle – datab, sequencer – data1). The original should never be altered. If the files are in a .tar archive, do not untar on the NAS. Be sure to check that the files copied over okay using the md5sum or checksum.

Use cp: *cp location.and.file.copying.from location.and.file.copying.to*

1. You will need to be sure you have space on the computer hard drive. I usually use Sequencer’s *data1*. To check for hard drive space, *df -h*.
2. Once the files are on the computer for processing, untar them:

*tar –xvf file.tar*

Once complete, you can remove the .tar files to save on space.

There are several files that we do not need (the “FILTERED” and “ADAPTOR” and Undetermined). Remove these: *rm –r files.to.remove*

1. All programs are on GitHub in sequencing-pipeline. You will need these on your linux home space.
2. If your sequences were run on multiple lanes, you will first need to concatenate your files so each sample has only one file. Use *concat.sh*. ***If you edit a program and change the filename you will need to change the file permissions:*** *chmod u+x filename* or *chmod 777 filename* I find as each experiment uses different lengths for names, I change the program to grab the appropriate number of characters to capture the name.

Now you will be able to execute the new programs. Also, the section that writes the output has be starred (##) out. Once you are sure everything is working okay, remove the stars (##) and proceed.

*main-scripts/concat.sh location.of.files.to.concat location.to.place.concatenated.files*

If you do not need to concatenate your files, you will need to convert the .gz to .bz2 as follows:

*gunzip –c < file.gz | bzip2 –c > file.bz2*

The concat.sh program includes this step.

7) Some reads need to be trimmed, such as those libraries created using the NuGen Ovation RNA-Seq Single cell system (these need to have the first 8 nucleotides trimmed as they are part of the adaptor, not the sample). Trim after concatenating so all of a single sample is trimmed at the same time.

*Bunzip2 –kc filename | fastx\_timmer –f 9 | bzip2 > new name of file*

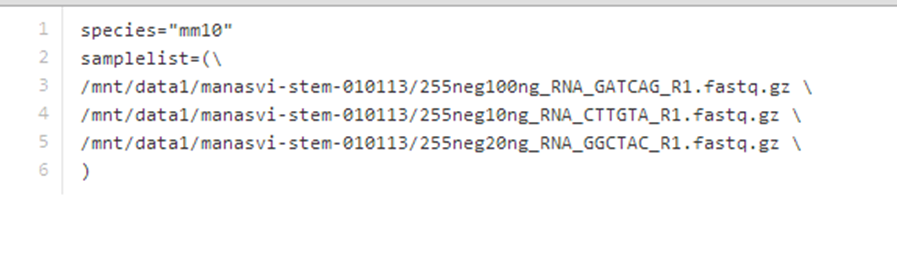
The –f indicates the first nucleotide to be kept and the default for the last is all so (-f 9) starts keeping the nucleotides from bp 9 to the end, effectively trimming the first 8.

8) Now a list needs to be made of all the samples. This can be facilitated by making a digital list of all of the files in your sequencing folder that we placed on the linux computer. This can be began by using the following command:

*ls /folder-where-my-sequences-are/\* > listname.txt*

You want to be in the sequencing-pipeline so that the list contains the correct map to the files. Now you can open the list file in your favorite linux writing program (vin, nano, others)

The key is to make a file that looks like the following:



Some key features:

* Species needs to be listed by the name of the files the sequences will be mapped to (with ensemble we would use “mouse”. Oracle is grch38-human, grcm38-mouse currently). Check the directory name. Location will be listed in config.sh.
* The sample list needs to list all the files with a space at the end before the \ . The backslash allow the next line to be read as if a single space was there. There cannot be a space after the \.
* This list will be used with many of the following programs so having this correct will save a lot of time.

9) Next, if the sequencing facility has not done so, you can run *preqc.sh* to check on the quality of the sequencing. This program is usually run from the sequencing pipeline directory. Below is an example:

*[jennifer@sequencer sequencing-pipeline]$ main-scripts/preqc.sh lists/FADS1*

10) Now that you know that the sequences have passed QC, it is time to map them against a reference genome. The genome that you will reference against should be the first line in your “list”. This program is also usually run from the sequencing pipeline directory. Below is an example:

[jennifer@sequencer sequencing-pipeline]$ main-scripts/map.sh lists/FADS1 *2 > err.log | tee out.log* Adding the text in italic will print the log and what is printed onto the screens into a written log that you can check later if there is a problem.

**NOTE!** If your samples are stranded, then you will need to edit the map.sh program to remove the -stranded no from the htseq-count portion of the program. The single cell NuGen kit produces stranded libraries.

If you are checking to see if your files are being produced, be sure to use:

du –sh \*

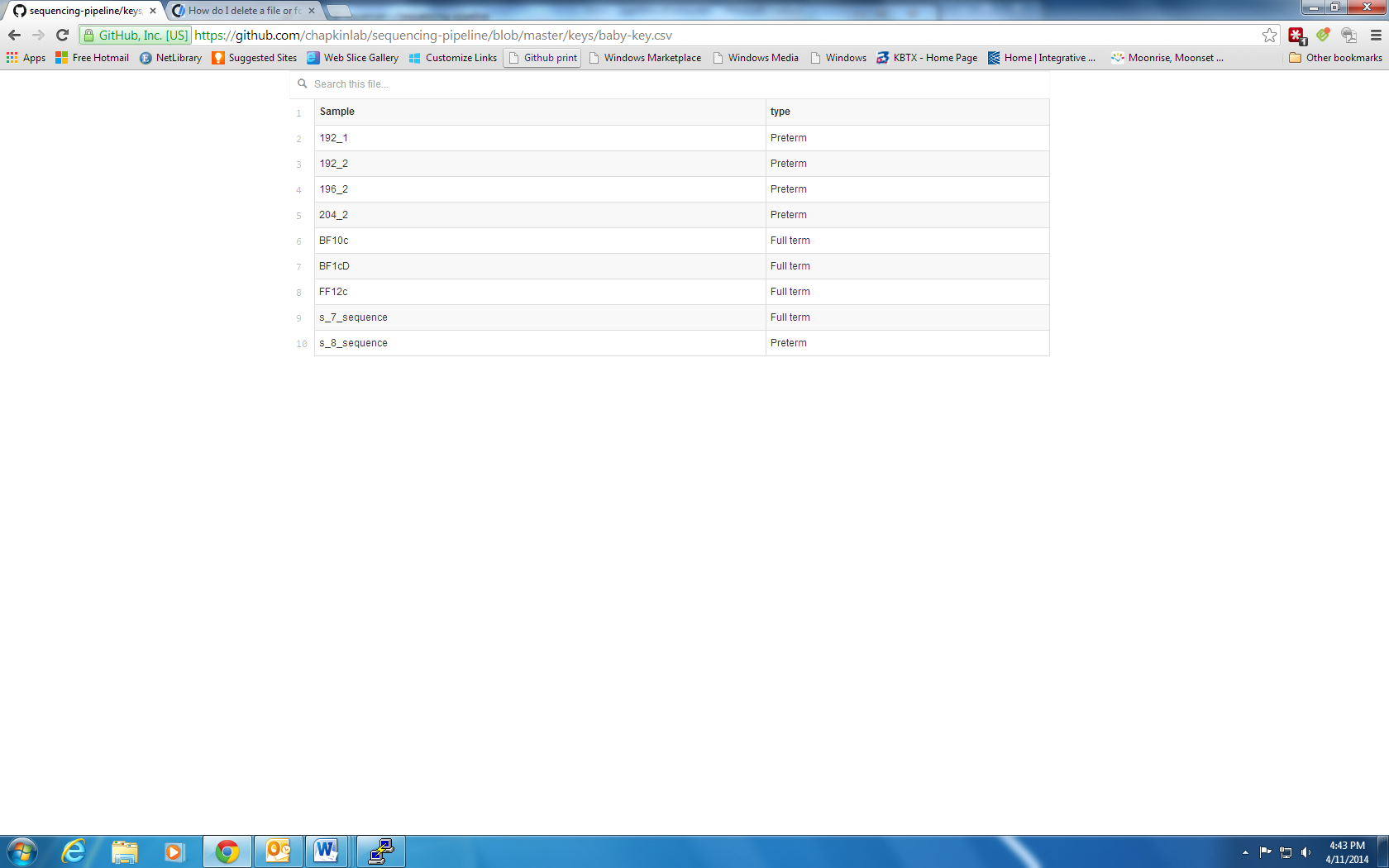
not

ls –lh

to get accurate results. Often ls –lh will show 0 in the files which is incorrect.

If the run gets killed for some reason (such as the ssh connection was dropped – it happens), be sure to delete the last samples output folder as it most likely didn’t finish and when you restart, it will be skipped as the program will think it has finished that sample.

11) In order to tell how the mapping went, we would like to see an overview (summary). You may need to create a “key” file as a comma delimited filetype (it is not necessary with the current set up). You have to have the first column be your sample name with a title of sample. The other columns may indicate the different treatment groups. This “key” will also be used when running cuffdiff (we don’t use cuffdiff anymore – use EdgeR) for comparisons between groups/differential expression.



Run *summary.py* in order to create a .csv file with that info.

[jennifer@sequencer sequencing-pipeline]$ main-scripts/summary.py lists/FADS1 FADS1-key.csv

The *summary.py* program also creates the list of genes with counts for each sample.

12) (Most will do EdgeR on their own computers rather run it on a Linux based computer) Now you are ready to find the differentially expressed genes. You will need to edit *run-comparisons.py* to set up the comparisons you want to run. Run *run-comparisons.py*.

[jennifer@sequencer sequencing-pipeline]$ main-scripts/run-comparisons.py lists/FADS1 FADS1-key.csv

Be sure that the sb.check\_call line (should be the last line) is uncommented (no #)

\*\*\*\*Need to update with EdgeR usage\*\*\*

Links to Kumaran/Radi videos

<https://www.youtube.com/watch?v=kXnN28aSUMA>

<https://www.youtube.com/watch?v=FtpHgvSFLIk>