
MMG434 Documentation

Release 0.0.1

Benjamin K Johnson

March 16, 2015

1	Contents:	1
1.1	Install software and use FileZilla to transfer files	1
1.2	RNA-seq background information, basic Linux/Unix commands, Trimmomatic, and FastQC	9
1.3	Align sequences with Bowtie and count gene features with HTSeq	22
1.4	Analyzing <i>L. reuteri</i> data	25
1.5	Modification to the scripts from today (3-4-15)!	29
1.6	Differential gene expression analysis with edgeR	29
1.7	FastQC files	37
1.8	License	38
1.9	Questions?	39
2	Indices and tables	41

Contents:

1.1 Install software and use FileZilla to transfer files

Before moving forward to mapping reads to the genome, it is necessary to QC the reads and remove low quality reads and adapter sequences. Since most of you are familiar with Trimmomatic and FastQC on the HPCC from MMG 433, we will do all the analysis there instead of on your own local machine. Further we need the ability to perform differential gene expression between conditions. We have chosen to do this through several software packages. Below are some instructions on how to install them.

1.1.1 Software to install:

1. *R*
2. *RStudio*
3. *Qualimap*

1.1.2 Using FileZilla to transfer files:

1. *Transferring files from your machine to the HPCC using FileZilla*

1.1.3 R

In order to do differential gene expression to compare treatments and identify what is changing at the transcript level, we need to install the statistical programming language R and a really neat interface to work in R called RStudio.

1. R can be downloaded from [here](#).
2. Click on the appropriate link for your operating system (Linux, Mac OS X, or Windows).

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

3. Then, click on the latest version of the software. This will initiate the download. **Windows users** unless you know you already have R installed, click on the **install R for the first time** link (see below). **Mac users** you have two options based on what flavor of OS X you have. If you are not sure what version you have, look at the screen shots below.

Windows users:

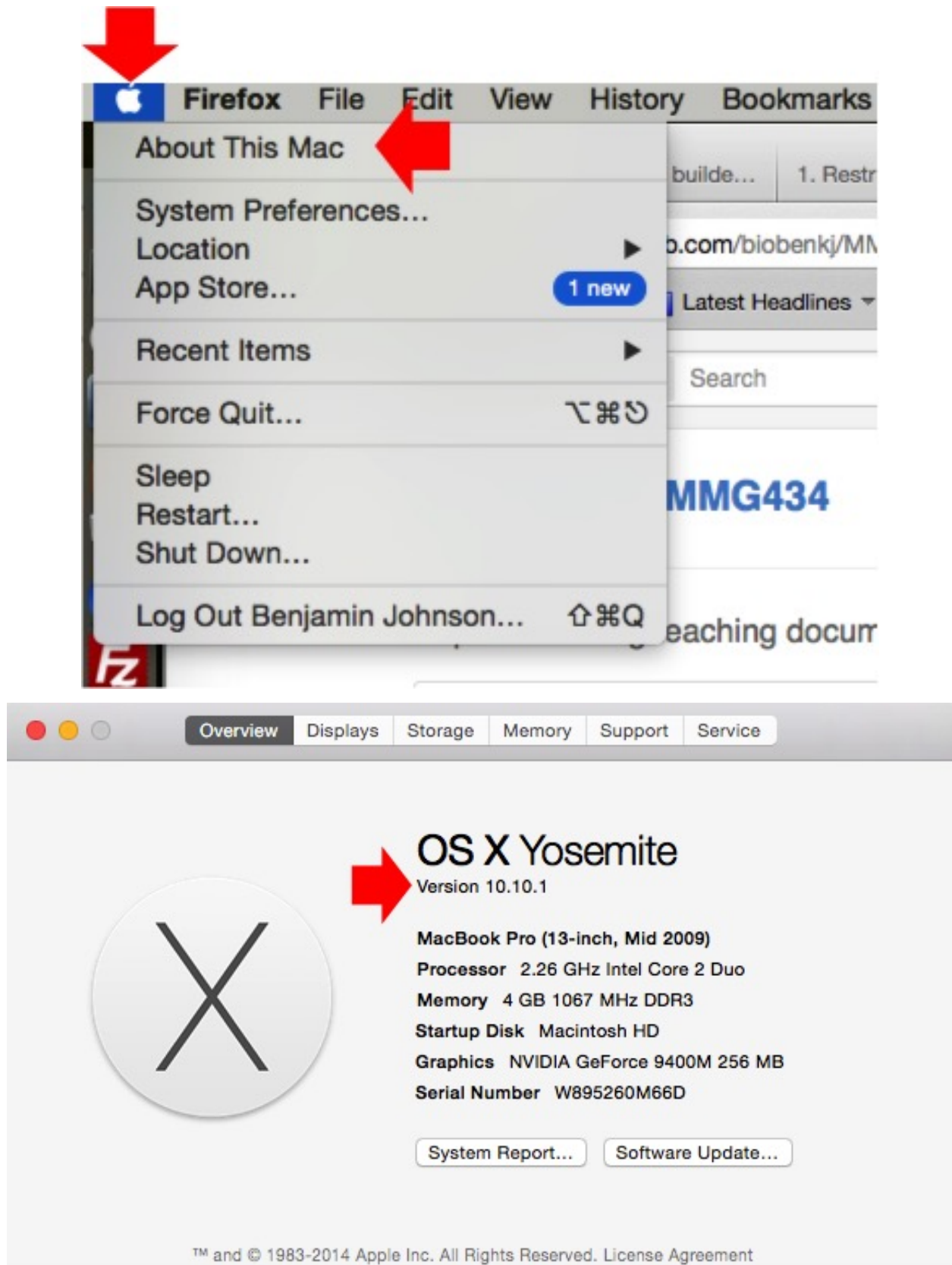
R for Windows

Subdirectories:

base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to install R for the first time .
contrib	Binaries of contributed packages (managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
Rtools	Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.



Mac users (to determine OS X version):



Mac users (version of R to download):

Files:

[R-3.1.2-snowleopard.pkg](#)

MD5-hash: 8af93200b567282932992decff5daf1d
 SHA1-hash: e8aee3cc4d3d97d8e5237fb50afaede38e1fb993
 (ca. 68MB)

R 3.1.2 binary for Mac OS X 10.6 (Snow Leopard) and higher, signed package. Contains R 3.1.2 framework, R.app GUI 1.65 in 64-bit for Intel Macs. The above file is an Installer package which can be installed by double-clicking. Depending on your browser, you may need to press the control key and click on this link to download the file.

This package contains the R framework, 64-bit GUI (R.app) and Tcl/Tk 8.6.0 X11 libraries. The latter component is optional and can be omitted when choosing "custom install", it is only needed if you want to use the `tcltk` R package. GNU Fortran is **NOT** included (needed if you want to compile packages from sources that contain FORTRAN code) please see [the tools directory](#).

[R-3.1.2-mavericks.pkg](#)

MD5-hash: d8fb6ea80357dd058aa1691c684e091
 SHA1-hash: 61c78cbb3024bf648032006fe19d8421c52ac8ba
 (ca. 55MB)

R 3.1.2 binary for Mac OS X 10.9 (Mavericks) and higher, signed package. It contains the same software versions as above, but this R build has been built with Xcode 5 to leverage new compilers and functionalities in Mavericks not available in earlier OS X versions.

Note: the use of X11 (including `tcltk`) requires [XQuartz](#) to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your OS X to a new major version.

4. After the download finishes, double-click on the file and follow the instructions to install the software.
5. Congratulations! You've installed yet another piece of software for this module.

1.1.4 RStudio

RStudio is a fantastic interface to work in R. R does have a graphical user interface (GUI) that you can download and use, however I find RStudio much more intuitive/easier to use.

1. RStudio can be downloaded [here](#).
2. We want to download and install the open-source version of RStudio for the desktop.

The screenshot shows the RStudio website with the following content:

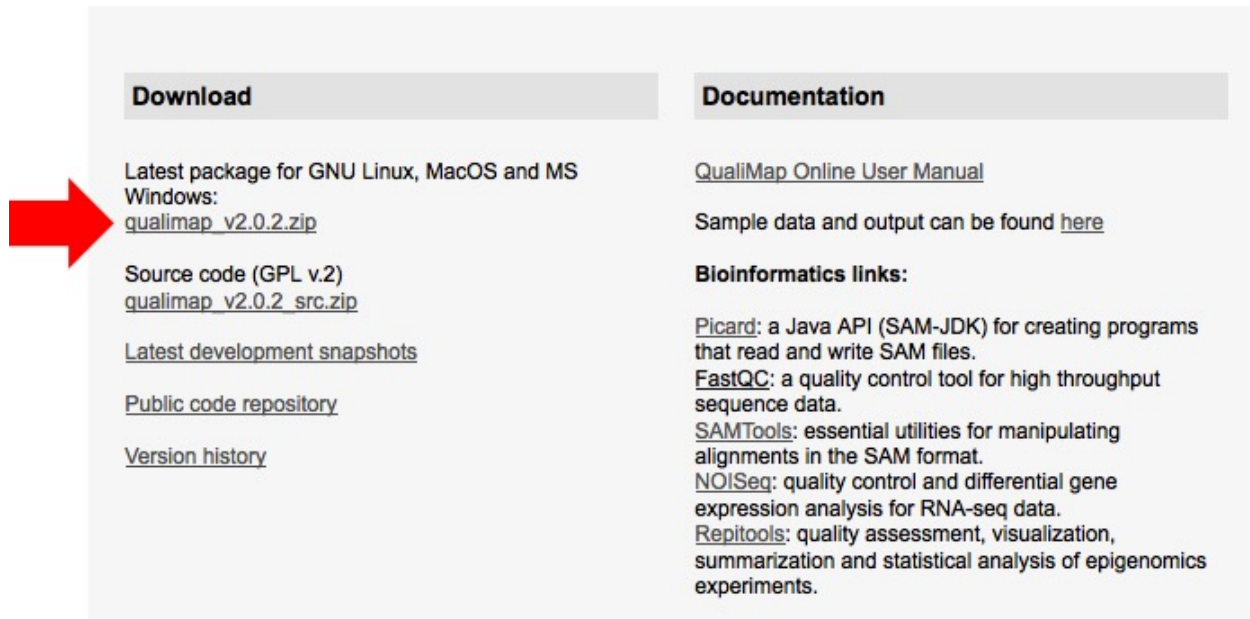
- Navigation:** Products, Resources, Pricing, About Us, Blog, Search icon.
- Overview:**
 - editor
 - Quickly jump to function definitions
 - Easily manage multiple working directories using projects
 - Integrated R help and documentation
 - Interactive debugger to diagnose and fix errors quickly
 - Extensive package development tools
- Support:**
 - Community forums only
 - Priority Email Support
 - 8 hour response during business hours (ET)
- License:** AGPL v3, [RStudio License Agreement](#)
- Pricing:** Free, \$995/year
- Buttons:** A red arrow points to a blue "DOWNLOAD RSTUDIO DESKTOP" button, and a "BUY NOW" button is also visible.

3. Double click the file after the software has finished downloading and follow the instructions to install the software.
4. That's it!

1.1.5 Qualimap

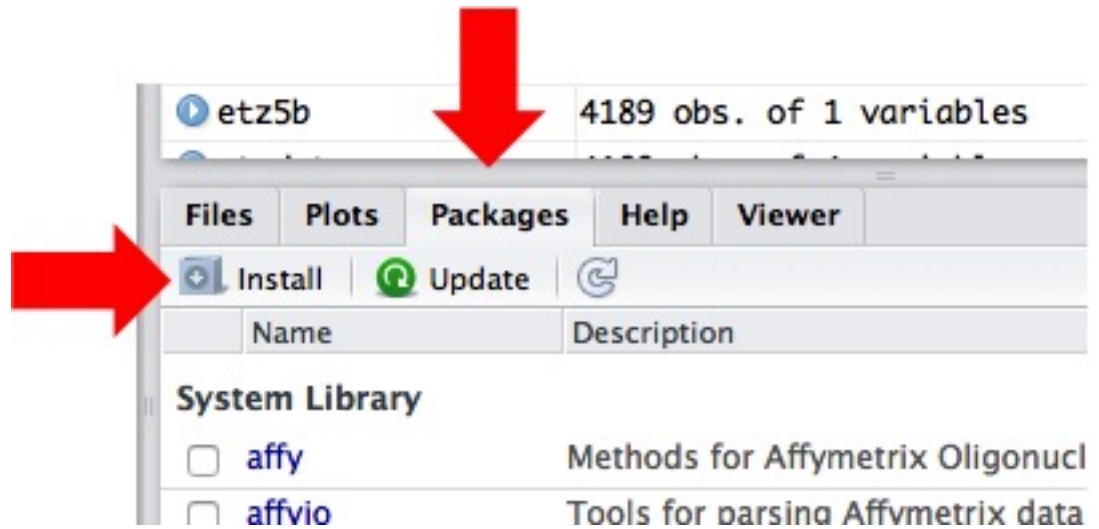
The last piece of software that is nice to have is a software suite that works with Java and R to generate PDF documents summarizing the data post-alignment to the genome.

1. Qualimap can be downloaded [here](#).
2. Click on the .zip file (as seen below)



Download	Documentation
<p>Latest package for GNU Linux, MacOS and MS Windows: qualimap_v2.0.2.zip</p> <p>Source code (GPL v.2) qualimap_v2.0.2_src.zip</p> <p>Latest development snapshots</p> <p>Public code repository</p> <p>Version history</p>	<p>QualiMap Online User Manual</p> <p>Sample data and output can be found here</p> <p>Bioinformatics links:</p> <p>Picard: a Java API (SAM-JDK) for creating programs that read and write SAM files. FastQC: a quality control tool for high throughput sequence data. SAMTools: essential utilities for manipulating alignments in the SAM format. NOISeq: quality control and differential gene expression analysis for RNA-seq data. Repitools: quality assessment, visualization, summarization and statistical analysis of epigenomics experiments.</p>

3. The file will end up in your downloads folder and will likely need to be unzipped as it is compressed. This can typically be accomplished just by double clicking on the file itself in your downloads folder.
4. Once this is done, move the folder to your desktop.
5. Before we can run this software suite, we will need to install a few packages in R: optparse (from CRAN), NOISeq, Repitools, Rsamtools, GenomicFeatures, rtracklayer (all available from Bioconductor).
6. To do this, open RStudio, and click on the **packages** tab and then the **Install** button.



7. Type **optparse** into the **Packages (separate multiple with space or comma):** field. Then click **Install**. RStudio should do the rest.
8. To install the packages from Bioconductor, click next to the **>** cursor. Copy and paste this command next to the **>** and hit Enter/Return:

```
source("http://bioconductor.org/biocLite.R")
```

```
Console ~/ ↵

R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

Loading required package: limma
Loading required package: edgeR
>
>
>
>
>
>
>
> source("http://bioconductor.org/biocLite.R")
```

9. Then click next to the > cursor and type `biocLite("NOISEq")` and hit Enter/Return. Repeat for Repitools, Rsamtools, GenomicFeatures, and rtracklayer.
10. That's all the software we need to install for now!

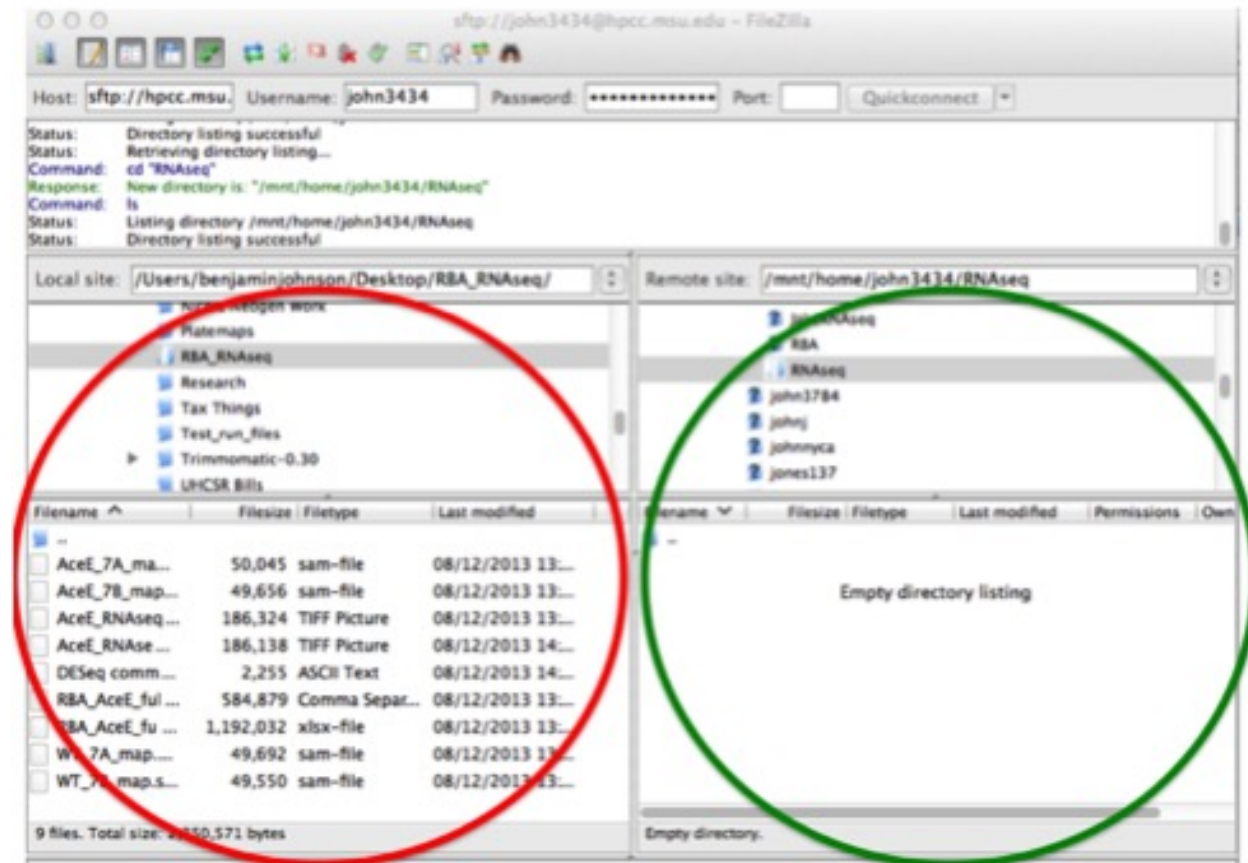
1.1.6 Transferring files from your machine to the HPCC using FileZilla

1. If you haven't already downloaded and installed [FileZilla](#), please do so. We want to download the **FileZilla Client** and *not* the server version.

2. Open the application and then we will need to input a few things to get connected to the MSU HPCC.

- Host: hpcc.msu.edu
- Username: Your MSU NetID
- Password: Your MSU NetID password
- Port: 22
- Click **Quickconnect**

3. Now that you are connected, you can move files from your computer (red circle) to the MSU HPCC (green circle) and vice versa, simply by double clicking the file. You can navigate through the folders on your machine and the HPCC just like you would when browsing for documents on your own computer. The connection closes if you exit out of the application.



Let's get to work!

1.2 RNA-seq background information, basic Linux/Unix commands, Trimmomatic, and FastQC

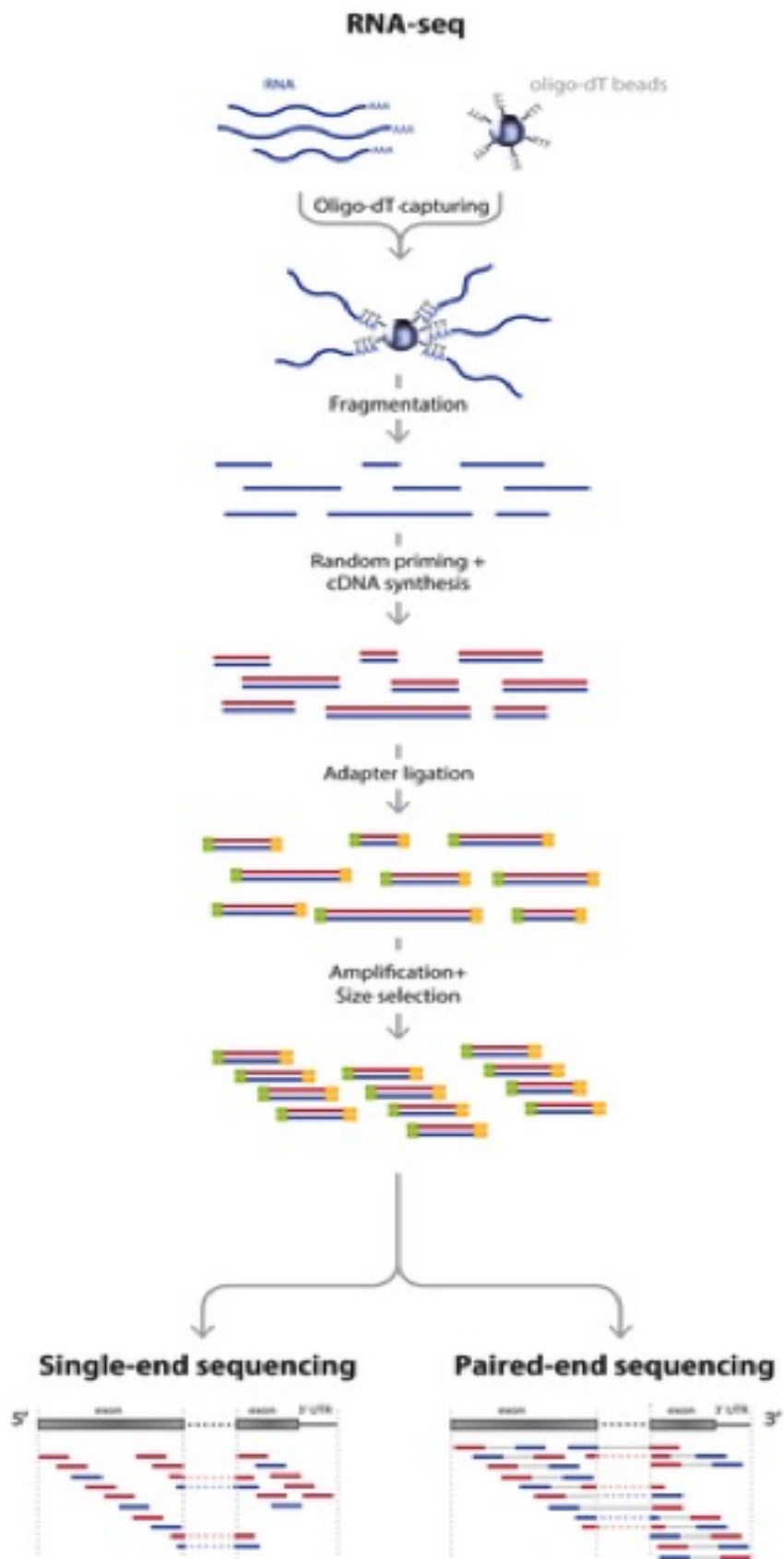
Before we dig into the data and begin trimming and aligning the reads to the genome, I think it is useful to understand what happens *after you submit your RNA to the sequencing facility*. This sort of knowledge can be very useful in understanding what could potentially provide bias and any number of issues to the end dataset. In this session we will cover several things including:

1. *RNA-seq background information*
2. *Basic Linux/Unix commands*
3. *Getting some RNA-seq data*
4. *Trimmomatic*
5. *FastQC*

1.2.1 RNA-seq background information

Before we begin, let's watch a video about how [Illumina sequencing works](#).

This video does a pretty good job explaining how, in generalities the sequencing process works for DNA. So for sequencing RNA, the process is as follows:



Adapted from: Zhernakova et al., PLoS Genetics 2013

So actually, we aren't sequencing RNA at all! We are sequencing the cDNA made from the RNA. RNA-seq is a high resolution next generation sequencing (NGS) method to assess the transcriptome of an organism and compare transcriptional changes between organisms/treatments to ascertain specific pathways/genes that are moving in response. But now, let's talk about what can add bias to the data and what we do with the data to make sure that it is reasonable to proceed to further analysis steps.

But first, let's brainstorm a little bit. Look back at the RNA-seq workflow figure above and let's suggest a few places where things could potentially affect the output dataset.

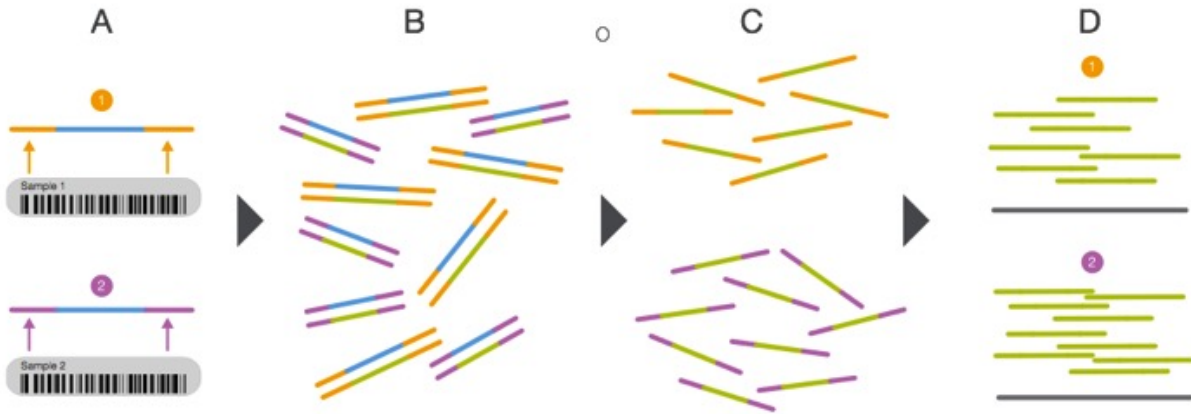
Here are a few thoughts...

- How could the random priming step affect downstream results?
- How could RNA secondary structures affect the library preparation process?
- Would GC content be a problem?
- Could gene length cause issues?
- What might happen if you have genes with substantially different expression levels?
- During the cluster generation on the Illumina flow cell, what might happen if you have too few clusters? Too many?
- How is it possible to sequence many samples at one time?
- What if you run out of reagents from one kit and have to open another kit to finish the library preparation process?
- Could sequencing depth be an issue?

So now that you may be questioning the validity of any RNA-seq dataset, take heart! Many very smart people have thought about these issues and come up with ways to assess technical artifacts and correct for them. So again, let's brainstorm some potential solutions to these problems. Which problems can be addressed through better chemistries/processes vs. mathematical/computational correction?

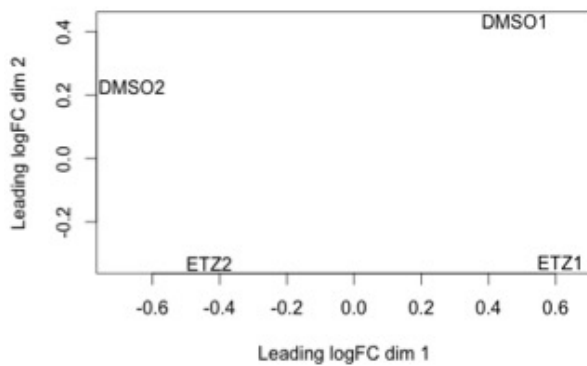
These sorts of issues should always be considered, but recognize that RNA-seq is becoming fairly commonplace and solutions to many of these questions exist. Be critical of your data and *always* look at the raw data.

Multiplexing the sequencing process by pooling several samples together is not only cheaper, it can overcome what are known as *batch effects*. Batch effects are when you have samples that correlate with one another based on batch/time/etc. instead of biological replication. This is a very real phenomenon and can be caused by using different lots of the same kit/flow cells when preparing samples! You can correct for this, but we will get there later... For now, have a look at the diagram showing how multiplexing is achieved.



From: http://www.illumina.com/content/dam/illumina-marketing/documents/products/sequencing_introduction_microbiology.pdf

This is an example of what a *batch effect* looks like. Note how DMSO1 and ETZ1 group together and DMSO2 and ETZ2 group together (e.g. by batch).



	Batch 1 Time:ETZ	Batch 2 Time:ETZ
Batch 1 Time:ETZ	1.0	0.62
Batch 2 Time:ETZ	0.62	1.0

We can determine what is considered a “good” base call from a “bad” one through using what is known as the Phred scoring system or Q-score.

Where Q is defined as a property that is logarithmically related to the base call error probability:

$$Q = -10 \log_{10} P \mid \text{error probability} = P^2$$

So this means:

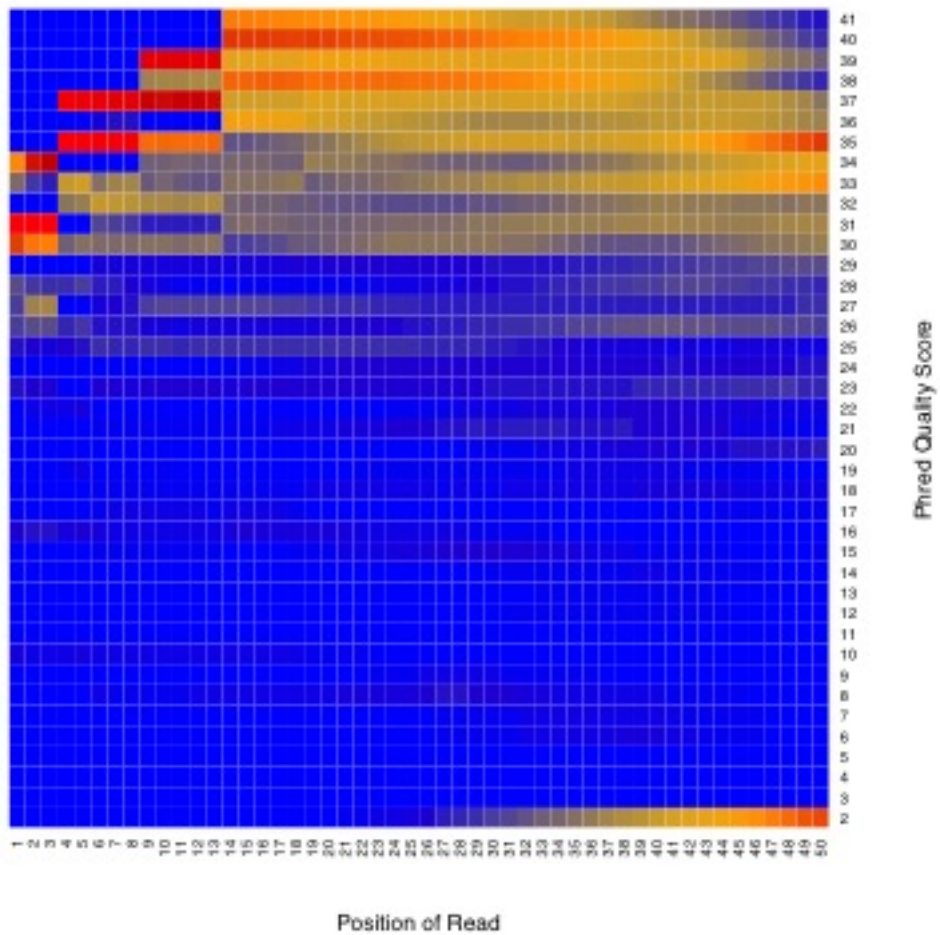
Table 1: Quality Scores and Base Calling Accuracy

Phred Quality Score	Probability of Incorrect Base Call	Base Call Accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1,000	99.9%
40	1 in 10,000	99.99%
50	1 in 100,000	99.999%

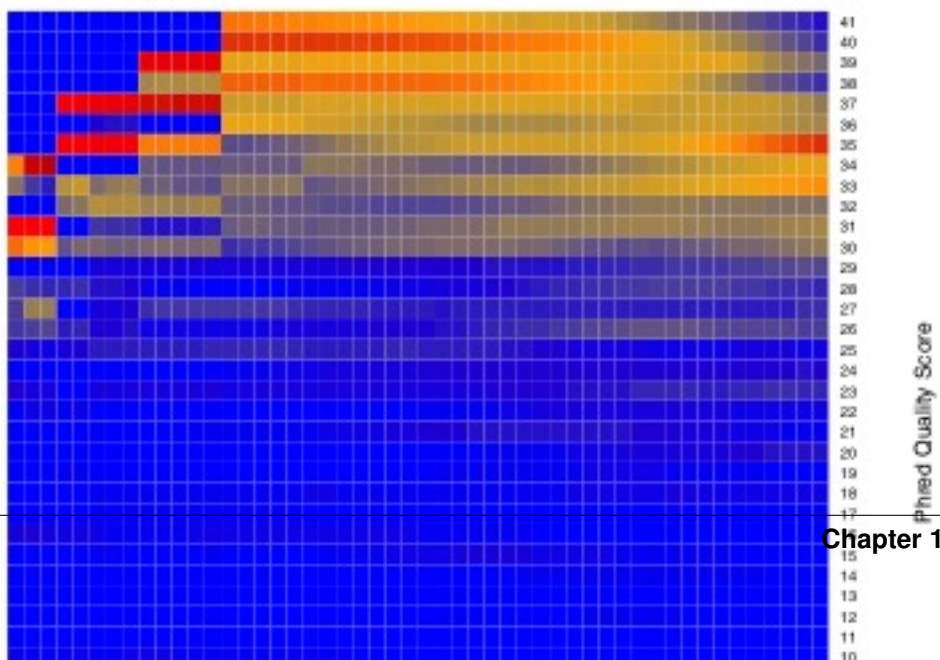
From: http://res.illumina.com/documents/products/technotes/technote_q-scores.pdf

Illumina tends to output sequence results with a Q > 30. So let's have a look at what some raw data looks like in terms of Q-scores before and after trimming adapters and low quality reads.

Untrimmed alignment



Trimmed alignment



This is why we do the trimming before attempting to align the reads to the reference genome. Since we are using FastQC, let's have a look at some sample data of what [good Illumina data looks like](#).

So, we have come to the end of the background section. Even with all of the great tools and chemistries that have been developed to handle RNA-seq datasets, the old mantra still applies: *garbage in; garbage out* and *with great power comes great responsibility*. Take care in analyzing these sorts of data as they typically influence many downstream experiments.

Questions!

1.2.2 Basic Linux/Unix commands

To refresh your memory on some basic Linux/Unix commands, we will cover the basic commands necessary to:

1. Move through folders
2. List the contents of a folder
3. Make new folders
4. Rename files/folders
5. Delete files/folders
6. Load modules on the MSU HPCC

	Com- mand	What it does...	Examples
1.	cd	Change directory/folder	> cd ~ (this changes to your home directory); > cd .. (this goes back one folder)
2.	ls	List the contents of a folder	> ls
3.	mkdir	Make a new directory/folder	> mkdir NewFolder (this will make a new folder called 'NewFolder' in your current directory)
4.	mv	Rename or move a file from one name to another	> mv file1 file2 (this will rename/move file1 to file2)
5.	rm	Remove a file (add the -r flag to remove a folder)	> rm file1 (remove file1); > rm -r folder1 (remove folder1)
6.	module load	Load a module on the MSU HPCC	> module load Bowtie (loads the most recent version of Bowtie on the HPCC)

Command reference sheet

Unix/Linux Command Reference

FOSSwire.com

File Commands	System Info
ls - directory listing	date - show the current date and time
ls -al - formatted listing with hidden files	cal - show this month's calendar
cd dir - change directory to <i>dir</i>	uptime - show current uptime
cd - change to home	w - display who is online
pwd - show current directory	whoami - who you are logged in as
mkdir dir - create a directory <i>dir</i>	finger user - display information about <i>user</i>
rm file - delete <i>file</i>	uname -a - show kernel information
rm -r dir - delete directory <i>dir</i>	cat /proc/cpuinfo - cpu information
rm -f file - force remove <i>file</i>	cat /proc/meminfo - memory information
rm -rf dir - force remove directory <i>dir</i> *	man command - show the manual for <i>command</i>
cp file1 file2 - copy <i>file1</i> to <i>file2</i>	df - show disk usage
cp -r dir1 dir2 - copy <i>dir1</i> to <i>dir2</i> ; create <i>dir2</i> if it doesn't exist	du - show directory space usage
mv file1 file2 - rename or move <i>file1</i> to <i>file2</i> if <i>file2</i> is an existing directory, moves <i>file1</i> into directory <i>file2</i>	free - show memory and swap usage
ln -s file link - create symbolic link <i>link</i> to <i>file</i>	whereis app - show possible locations of <i>app</i>
touch file - create or update <i>file</i>	which app - show which <i>app</i> will be run by default
cat > file - places standard input into <i>file</i>	
more file - output the contents of <i>file</i>	
head file - output the first 10 lines of <i>file</i>	
tail file - output the last 10 lines of <i>file</i>	
tail -f file - output the contents of <i>file</i> as it grows, starting with the last 10 lines	
Process Management	Compression
ps - display your currently active processes	tar cf file.tar files - create a tar named <i>file.tar</i> containing <i>files</i>
top - display all running processes	tar xf file.tar - extract the files from <i>file.tar</i>
kill pid - kill process id <i>pid</i>	tar czf file.tar.gz files - create a tar with Gzip compression
killall proc - kill all processes named <i>proc</i> *	tar xzf file.tar.gz - extract a tar using Gzip
bg - lists stopped or background jobs; resume a stopped job in the background	tar cjf file.tar.bz2 - create a tar with Bzip2 compression
fg - brings the most recent job to foreground	tar xjf file.tar.bz2 - extract a tar using Bzip2
fg n - brings job <i>n</i> to the foreground	gzip file - compresses <i>file</i> and renames it to <i>file.gz</i>
	gzip -d file.gz - decompresses <i>file.gz</i> back to <i>file</i>
File Permissions	Network
chmod octal file - change the permissions of <i>file</i> to <i>octal</i> , which can be found separately for user, group, and world by adding:	ping host - ping <i>host</i> and output results
<ul style="list-style-type: none"> • 4 - read (r) • 2 - write (w) • 1 - execute (x) 	whois domain - get whois information for <i>domain</i>
Examples:	dig domain - get DNS information for <i>domain</i>
chmod 777 - read, write, execute for all	dig -x host - reverse lookup <i>host</i>
chmod 755 - rwx for owner, rx for group and world	wget file - download <i>file</i>
For more options, see man chmod .	wget -c file - continue a stopped download
SSH	Installation
ssh user@host - connect to <i>host</i> as <i>user</i>	Install from source:
ssh -p port user@host - connect to <i>host</i> on port <i>port</i> as <i>user</i>	./configure
ssh-copy-id user@host - add your key to <i>host</i> for <i>user</i> to enable a keyed or passwordless login	make
	make install
	dpkg -i pkg.deb - install a package (Debian)
	rpm -Uvh pkg.rpm - install a package (RPM)
Searching	Shortcuts
grep pattern files - search for <i>pattern</i> in <i>files</i>	Ctrl+C - halts the current command
grep -r pattern dir - search recursively for <i>pattern</i> in <i>dir</i>	Ctrl+Z - stops the current command, resume with fg in the foreground or bg in the background
command grep pattern - search for <i>pattern</i> in the output of <i>command</i>	Ctrl+D - log out of current session, similar to exit
locate file - find all instances of <i>file</i>	Ctrl+W - erases one word in the current line
	Ctrl+U - erases the whole line
	Ctrl+R - type to bring up a recent command
	!! - repeats the last command
	exit - log out of current session
	* use with extreme caution.



Ref. sheet from: <http://files.fooswire.com/2007/08/fwunixref.pdf>

1.2.3 Getting some RNA-seq data

Before we begin, we need to collect some RNA-seq data to work with! To do this, we are going to take the first 100,000 sequences from a study looking at how *M. tuberculosis* changes its metabolism in response to different carbon sources at neutral and acidic pH (Baker et al.).

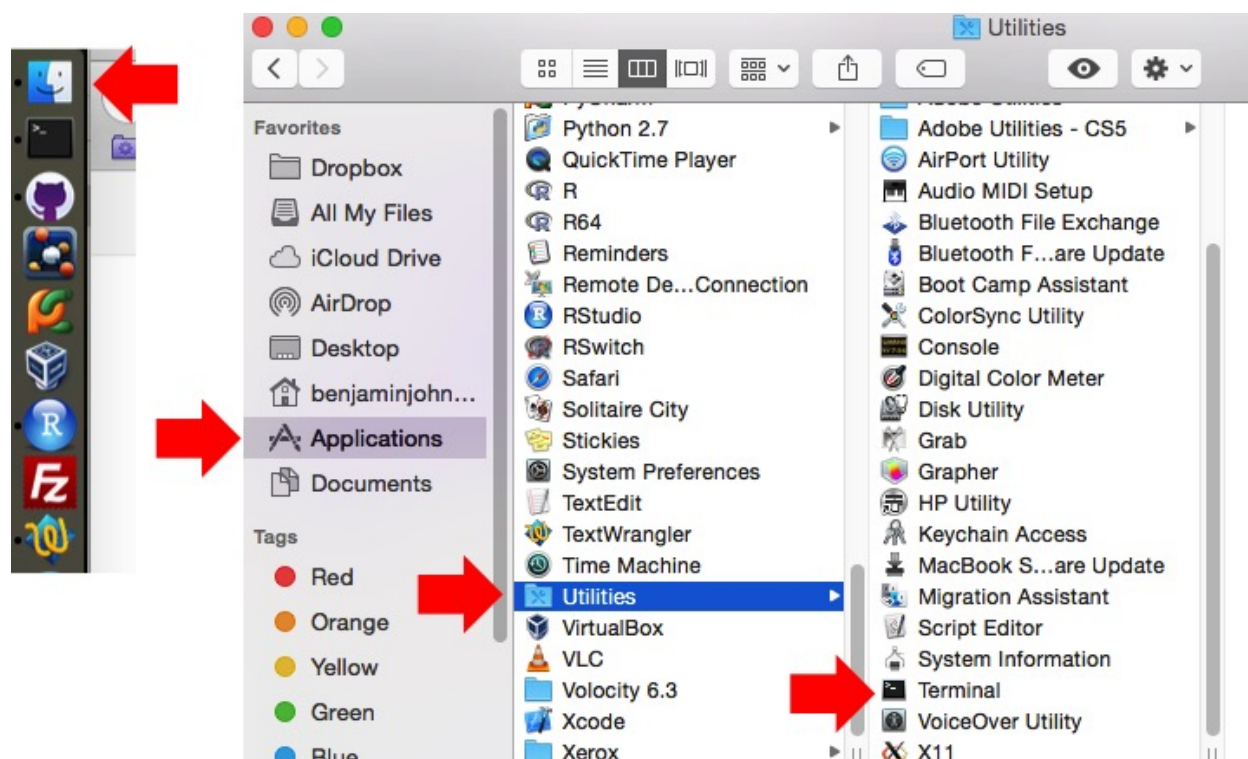
To do this, let's log onto the HPCC.

Logging into the MSU HPCC

There are multiple ways with which you can access the HPCC and transfer files to the iCER machines. This overview will be from a GUI standpoint. If you would like to get fancy and learn the Linux/Unix commands, you can access the examples on the HPCC wiki [here](#). Any other questions you may have not detailed here (which is a lot...) on how to use the HPCC more effectively, you can check out the user manual [here](#). There are even videos of examples on how to use various software packages.

Mac users:

1. Open the terminal by going to Finder -> Applications -> Utilities -> Terminal (might just be worth dragging it onto your dock).



2. Type: `ssh YourMSUNetID@hpcc.msu.edu`

3. You will then be prompted for your MSU NetID password. As you begin to type, the cursor will not show that you are entering characters, but you are. Hit the Enter/Return key at the end and you will be logged in. If this is the first time accessing the HPCC, it will send you a warning about not recognizing the RSA fingerprint. Type `yes` or `y` or whatever it needs to continue. It is okay, and necessary, to say you trust iCER to use the HPCC at MSU. If you are uncomfortable with any of this, utilize a lab or MSU computer.

4. As an example of **Step 3.**, for me it would be `> ssh john3434@hpcc.msu.edu`.

5. Once you are logged in, it should look something like this:

```

benjaminjohnson — john3434@gateway-00:~ — ssh — 86x33
~ benjaminjohnson$ ssh john3434@hpcc.msu.edu
Password:
Last login: Wed Sep 25 20:32:58 2013
Load Warning: Did not find: use.cus
Try: "module spider use.cus"

  _____; _____
 /  ___  /  /  ___  /  /  ___  /
|  /___/  |  /___/  |  /___/  |
 \  ___/  \  ___/  \  ___/  \
  _____  _____

Welcome to Michigan State's High Performance Computing Center
** Unauthorized access is prohibited **

We recommend using dev-amd09 (or nodes with low usage).
For GPU development please use green nodes.
For MIC development please use underlined nodes

Development Nodes (usage)                Filesystem Information
-----
dev-intel07 (low)   dev-amd09 (low)           $(HOME) at 35% usage
dev-intel10 (low)  dev-gfx10 (low)          (used ~86G of 250G)
dev-gfx13 (low)    dev-ph113 (low)

Cluster Load (utilization)
-----
short jobs (< 4 hrs) (97%)   general jobs (82%)
large memory jobs (100%)     gpu jobs (33%)

[john3434@gateway-00 ~]$

```

6. Congratulations! You've logged in. Let's make a new folder here in anticipation of putting the data into it. Let's call it **RNAseq**. Please don't add any spaces. If you aren't sure how to make a new folder, scroll up a bit to the [Basic Linux/Unix commands](#).

7. To **log out**, type: **exit**.

Windows users:

1. I am going to take the easy way out and [here](#) is a video on how to install an ssh client on Windows.

2. Congratulations! You've logged in. Let's make a new folder here in anticipation of putting the data into it. Let's call it **RNAseq**. Please don't add any spaces. If you aren't sure how to make a new folder, scroll up a bit to the [Basic Linux/Unix commands](#).

3. To **log out**, type: **exit**.

Great! Now that we are logged onto the HPC and created the RNAseq folder, let's grab the data. To do this, copy and paste these commands into your terminal and then hit Enter/Return (Mac users can use Command+C to copy and Command+V to paste; Windows (PuTTY) users can use Ctrl+C to copy and then right mouse click to paste into the terminal:

```

module load powertools
intel14
cd ~/RNAseq
mkdir Data QC Bowtie HTSeq
cd Data
cp /mnt/research/mmg434/ExampleRNAseqData/*.fq.gz .
ls

```

So we have grabbed data from 4 different conditions (done in biological duplicate):

1. Glycerol pH 7.0
2. Glycerol pH 5.7
3. Pyruvate pH 7.0
4. Pyruvate pH 5.7

Thus, in total, we have 8 different libraries. However, before we move on, let's have a look at the commands we just entered. This is important to think about as the next time we will be working with RNA-seq data, it will be with the samples that you submitted for *L. reuteri*. Data management during the analysis is critical! Please be organized when doing analysis with large datasets. This will make your life much, MUCH easier when you have to come back later and sort out what is where and at what point in the analysis you are. So, let's break down these commands:

1. `module load powertools` - This command loads a series of commands that we can utilize to work on the HPCC in a fashion that makes our lives much easier.
2. `intel14` - This is a powertools command that logs us into the intel14 compute node on the HPCC.
3. `cd ~/RNAseq` - This command changes directory/folder to our RNAseq directory/folder we made earlier.
4. `mkdir Data QC Bowtie HTSeq` - This command creates some directories/folders within the RNAseq directory/folder with which to put all of our data. This allows us to keep each portion of the analysis in reasonably defined places.
5. `cd Data` - Change directory/folder to Data.
6. `cp /mnt/research/mmg434/ExampleRNAseqData/*.fq.gz .` - This command copies the data from the `/mnt/research/mmg434/ExampleRNAseqData` directory/folder to the directory/folder that you are currently in.
7. `ls` - This command lists the contents of the directory/folder you are currently in.

Now we can move on to trimming the data!

1.2.4 Trimmomatic

Trimmomatic is a lightweight java application that can remove Illumina adapter sequences and low quality reads. It uses a sliding window to analyze chunks of each read, examining the quality score, minimum read length, if it corresponds to an adapter sequence, etc. Let's have a look at the [documentation](#) to see what each option does.

We are going to trim the first file: `gly7a.fq.gz`. Later we will automate all of the trimming with a script, which we will look at the script below, but for now, let's trim the first file! Copy and paste these commands into your terminal and hit Enter/Return:

```

module load trimmomatic
java -jar $TRIM/trimmomatic SE -threads 4 ~/RNAseq/Data/gly7a.fq.gz ~/RNAseq/QC/trimmedgly7a.fq.gz I

```

You will likely see some output that looks like this:

```

TrimmomaticSE: Started with arguments: -threads 4 /mnt/home/john3434/RNAseq/Data/gly7a.fq.gz /mnt/home/john3434/RNAseq/QC/trimmedgly7a.fq.gz
Using Long Clipping Sequence: 'AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT'
Using Long Clipping Sequence: 'AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC'
ILLUMINACLIP: Using 0 prefix pairs, 2 forward/reverse sequences, 0 forward only sequences, 0 reverse

```

```
Quality encoding detected as phred33
Input Reads: 100000 Surviving: 96867 (96.87%) Dropped: 3133 (3.13%)
TrimmomaticSE: Completed successfully
```

Note: Copy this output into a text file somewhere and save it. You might want this for a report when you're finished.

Congratulations! You've just trimmed the first file of RNA-seq data! But, let's remind ourselves what each command and parameter is doing. Look through the command and discuss with a neighbor what is going on there. If you don't remember what each parameter does, have another look at the [documentation](#).

Now, beyond what Trimmomatic requires for input, we are doing one fancy thing here when we specify where the software file is and where the adapter file is with the **\$TRIM**. This is specific for the MSU HPCC and is simply a shortcut that represents a file path.

Go ahead and work with a neighbor and see if you can trim the second data file: gly7b.fq.gz. Take 5 minutes and give it a shot!

Have fun! Let me know if you have questions by placing a red sticky note on your computer.

1.2.5 FastQC

FastQC is a piece of software that allows us to analyze the quality of our data before proceeding to aligning the reads to the reference genome. Let's have a look again at what [good Illumina data](#) and [bad Illumina data](#) look like. This will help us determine the quality of our own sequence based on their examples.

Let's analyze the first file (trimmedgly7a.fq.gz) that we trimmed! To run FastQC on the HPCC, copy and paste the following commands into your terminal:

```
module load fastqc
cd ~/RNAseq/QC
fastqc trimmedgly7a.fq.gz
```

You will see some output that looks like this:

```
Started analysis of trimmedgly7a.fq.gz
  Approx 5% complete for trimmedgly7a.fq.gz
  Approx 10% complete for trimmedgly7a.fq.gz
  Approx 15% complete for trimmedgly7a.fq.gz
  Approx 20% complete for trimmedgly7a.fq.gz
  Approx 25% complete for trimmedgly7a.fq.gz
  Approx 30% complete for trimmedgly7a.fq.gz
  Approx 35% complete for trimmedgly7a.fq.gz
  Approx 40% complete for trimmedgly7a.fq.gz
  Approx 45% complete for trimmedgly7a.fq.gz
  Approx 50% complete for trimmedgly7a.fq.gz
  Approx 55% complete for trimmedgly7a.fq.gz
  Approx 60% complete for trimmedgly7a.fq.gz
  Approx 65% complete for trimmedgly7a.fq.gz
  Approx 70% complete for trimmedgly7a.fq.gz
  Approx 75% complete for trimmedgly7a.fq.gz
  Approx 80% complete for trimmedgly7a.fq.gz
  Approx 85% complete for trimmedgly7a.fq.gz
  Approx 90% complete for trimmedgly7a.fq.gz
  Approx 95% complete for trimmedgly7a.fq.gz
Analysis complete for trimmedgly7a.fq.gz
```


Now, to view the report, please use either FileZilla or WinSCP to transfer the file: **trimmedgly7a_fastqc.html** onto your desktop. If you don't remember how to use FileZilla, check back to *Install software and use FileZilla to transfer files*.

Let's open the file in a browser like FireFox. What do we think? Good or bad data?

Please work with a neighbor and perform FastQC analysis on the next file (trimmedgly7b.fq.gz). Give it a shot and put a green sticky note on your computer once you have done this and viewed the result in a browser.

1.2.6 Automating all of this

This was fine to just do two samples, but it sure does take a lot of time to do each one individually! Let's have a look at a script that will do all the trimming and FastQC analysis for us in one shot.

Have a look at the script below:

```
#!/bin/bash

#Load the trimmomatic and fastQC module
module load trimmomatic
module load fastqc

#Change to the QC directory
cd ~/RNAseq/QC

#Do the trimming!
java -jar $TRIM/trimmomatic SE -threads 4 ~/RNAseq/Data/gly7a.fq.gz ~/RNAseq/QC/trimmedgly7a.fq.gz I
java -jar $TRIM/trimmomatic SE -threads 4 ~/RNAseq/Data/gly7b.fq.gz ~/RNAseq/QC/trimmedgly7b.fq.gz I
java -jar $TRIM/trimmomatic SE -threads 4 ~/RNAseq/Data/gly5a.fq.gz ~/RNAseq/QC/trimmedgly5a.fq.gz I
java -jar $TRIM/trimmomatic SE -threads 4 ~/RNAseq/Data/gly5b.fq.gz ~/RNAseq/QC/trimmedgly5b.fq.gz I
java -jar $TRIM/trimmomatic SE -threads 4 ~/RNAseq/Data/pyr7a.fq.gz ~/RNAseq/QC/trimmedpyr7a.fq.gz I
java -jar $TRIM/trimmomatic SE -threads 4 ~/RNAseq/Data/pyr7b.fq.gz ~/RNAseq/QC/trimmedpyr7b.fq.gz I
java -jar $TRIM/trimmomatic SE -threads 4 ~/RNAseq/Data/pyr5a.fq.gz ~/RNAseq/QC/trimmedpyr5a.fq.gz I
java -jar $TRIM/trimmomatic SE -threads 4 ~/RNAseq/Data/pyr5b.fq.gz ~/RNAseq/QC/trimmedpyr5b.fq.gz I

#Analyze the reads pre-alignment with FastQC
fastqc trimmedgly7a.fq.gz
fastqc trimmedgly7b.fq.gz
fastqc trimmedgly5a.fq.gz
fastqc trimmedgly5b.fq.gz
fastqc trimmedpyr7a.fq.gz
fastqc trimmedpyr7b.fq.gz
fastqc trimmedpyr5a.fq.gz
fastqc trimmedpyr5b.fq.gz
```

Let's run it! Copy and paste the command into your terminal and hit Enter/Return:

```
bash /mnt/research/mmg434/RNAseqScripts/qcexempladata.sh
```

That's it! It has done everything for you at one time! This is also nice if we want to go back and re-run this analysis again, which will produce the same results each time.

Presentation time!

Please have one person from each treatment group come and present a *representative* report from each treatment, assessing the results.

Note: Save your report so that we can compile them at the end of the module.

1.3 Align sequences with Bowtie and count gene features with HTSeq

Today we will be covering two things:

1. *Bowtie*
2. *HTSeq*

1.3.1 Bowtie

Before we can align the trimmed sequences that we did yesterday, we need to first decompress the trimmed files in the QC folder on the HPCC. Please log onto the HPCC and copy and paste these commands into your terminal and hit Enter/Return (if you don't remember how to copy and paste, refer back to *RNA-seq background information, basic Linux/Unix commands, Trimmomatic, and FastQC*):

```
module load powertools
intel14
bash /mnt/research/mmg434/RNAseqScripts/decompressfastq.sh
ls ~/RNAseq/QC
```

These commands decompress the .gz trimmed files to .fastq files. Bowtie expects .fastq files for input.

What is Bowtie?

“Bowtie is an ultrafast, memory-efficient short read aligner geared toward quickly aligning large sets of short DNA sequences (reads) to large genomes... Bowtie indexes the genome with a [Burrows-Wheeler](#) index to keep its memory footprint small...”

What isn't Bowtie?

“Bowtie is not a general-purpose alignment tool like MUMer, BLAST, or Vmatch. Bowtie works best when aligning short reads to large genomes, though it supports arbitrarily small reference sequences (e.g. amplicons) and reads as long as 1024 bases. Bowtie is designed to be extremely fast for sets of short reads where (a) many of the reads have at least one good, valid alignment, (b) many of the reads are relatively high-quality, and (c) the number of alignments reported per read is small (close to 1).”

From: <http://bowtie-bio.sourceforge.net/manual.shtml#what-is-bowtie>

In order for Bowtie to work, we need to provide it with trimmed reads files and the reference genome in a FASTA format file. This type of file typically ends in .fa or .fasta.

We can acquire our favorite reference genome and feature file (GTF) from the [Ensembl website](#).

Once we get our data from the RTSF, we will download the *L. reuteri* JCM1112 genome file and feature file. The feature file contains data to inform HTSeq where the start and end of a gene is. This is important as HTSeq produces the number of transcripts per gene identified in a given sample. For now, we will just grab these two files from the /mnt/research/mmg434/ExampleRNAseqData directory.

To run Bowtie, we have to do a couple things. First, we need to build the index from the reference genome (trimmedMtbCDC1551.fa). To do that, copy and paste these commands into your terminal and hit Enter/Return:

```
cd ~/RNAseq/Bowtie
module load bowtie
bowtie-build /mnt/research/mmg434/ExampleRNAseqData/trimmedMtbCDC1551.fa trimmedCDC1551
```

So let's break down what we just did:

1. `cd ~/RNAseq/Bowtie` - Changing to the Bowtie subfolder inside of the RNAseq folder.
2. `bowtie-build /mnt/research/mmg434/ExampleRNAseqData/trimmedMtbCDC1551.fa trimmedCDC1551` - Calling Bowtie to build an index called `trimmedCDC1551` from the reference (`trimmedMtbCDC1551.fa`) found in the `/mnt/research/mmg434/ExampleRNAseqData` folder.

Now that we have built the index, we can ask Bowtie to do the alignment on the first file for us! Copy and paste these commands into your terminal and hit Enter/Return:

```
cd ~/RNAseq/Bowtie
bowtie -S trimmedCDC1551 ~/RNAseq/QC/trimmedgly7a.fastq > aligngly7a.sam
```

Let's break down what we just did:

1. `cd ~/RNAseq/Bowtie` - Changing to the Bowtie subfolder inside of the RNAseq folder.
2. `bowtie -S` - This calls Bowtie and asks the program to produce files with `.sam` output (this allows us to be able to interpret the alignment *somewhat*... It's still a lot to take in).
3. `trimmedCDC1551 ~/RNAseq/QC/trimmedgly7a.fastq > aligngly7a.sam` - Use the index we made called `trimmedCDC1551` and take the reads from `trimmedgly7a.fastq` in the QC subfolder within the RNAseq folder, and write the alignment (using the `>` character) to the file `aligngly7a.sam`.

Note: It is **very** important that you end the output file with `.sam`.

Now, work with a neighbor and see if you can change the above commands to align the `trimmedgly7b.fastq` file.

If you have problems, please put a red stick on your computer. If all is well, place a green sticky on your computer.

Bear in mind though, that we will have to do something *very* different when we go to analyze the full read set from the *L. reuteri* samples. We will do something like this (below).

The document will look something like this (take a minute and read through it):

```
#!/bin/bash -login
#PBS -l walltime=00:20:00,nodes=1:ppn=8,mem=8gb
#PBS -j oe

cd $PBS_O_WORKDIR
module load bowtie

mkdir $PBS_JOBID
cp trimmed* ./ $PBS_JOBID
cd $PBS_JOBID

bowtie-build trimmedlreuterijcm1112.fa trimmedlreuterijcm1112

#ppn above should be 1 larger than -p below (thus if ppn=8 up top, the number after '-p' below should
#the first file name is the name of the trimmed RNAseq read data that comes from Trimmomatic
#in order for this script to work you need to make sure that whatever the file is named, it starts w
#the second file is the file name that will contain the alignment of the reads to the genome. THIS HA
#every sample = 20 minutes of walltime
#thus, if you have four samples, you will need to change the walltime to 00:80:00 instead of 00:20:00
#this just makes sure that the job will complete
#simply copy and paste each one of these commands below for each sample you want to align with bowtie
#it is as easy as that

time bowtie -S -p 7 trimmedlreuterijcm1112 trimmedLRWT1.fastq > alignLRWT1.sam

#Do not do anything below this line-leave it as is
rm *.fastq
```

Questions!

1.3.2 HTSeq

This step will take the longest time, computationally, out of the entire workflow.

HTSeq is a powerful Python package for analyzing NGS data. For our purposes, we will be using the counting feature of HTSeq. Let's have a look at the way HTSeq can **count whether a read maps to a gene**.

We need to supply htseq-count with a couple things:

1. A genome feature file (GTF) so that HTSeq “knows” where the start and end of a gene is
2. The *.sam* file that was output from Bowtie

To do the counting, copy and paste these commands into your terminal and hit Enter/Return:

```
module load htseq
cd ~/RNAseq/HTSeq
htseq-count -m intersection-nonempty --stranded=yes ~/RNAseq/Bowtie/aligngly7a.sam /mnt/research/mmg
```

You likely have an idea by now what the first two commands are doing, but let's talk briefly about the htseq-count command:

1. htseq-count - This is calling the counting functionality of HTSeq.
2. -m intersection-nonempty - This is telling HTSeq how we want to determine what is or is not a gene feature.
3. --stranded=yes - This is telling HTSeq whether or not our data has strand specific information.
4. ~/RNAseq/Bowtie/aligngly7a.sam /mnt/research/mmg434/ExampleRNAseqData/alignMtbCDC1551.gtf > gly7amap.sam - This is saying where the aligned sequence file is, where the genome feature file (.gtf) and write the counts to the file gly7amap.sam

Note: It is **very** important that you end the output file with *.sam*.

Work with a neighbor and see if you can perform the counting on the second file (aligngly7b.sam). If you have trouble, put up a red sticky. If all is fantastic, put up a green sticky.

To use htseq-count on the HPC once we have the full read set from the *L. reuteri* samples, we will have to edit a job submission script, which will look something like this:

```
#!/bin/bash -login
#PBS -l walltime=01:20:00,nodes=1:ppn=8,mem=8gb
#PBS -j oe

#Load Numpy module
module load NumPy

#Set the number of threads to match ppn
export MKL_NUM_THREADS=8

#Load HTSeq module
module load HTSeq

#This is changing directories to the folder that you renamed that contains the aligned reads
#In this case I have a folder called Bowtie within the RNAseq folder
#These can be whatever you have chosen to name or rename your folders to be as long as it ends with t

cd ~/RNAseq/Bowtie/
```

```
mkdir $PBS_JOBID
cp align* ./ $PBS_JOBID
cd $PBS_JOBID
```

```
#in order for this script to work you need to make sure that whatever the file is named, it starts with
#the second file is the file name that will contain the counts of the aligned reads to the genome. This
#every sample = 1 hour and 10 minutes of walltime
#thus, if you have four samples, you will need to change the walltime to 04:40:00 instead of 01:10:00
#this just makes sure that the job will complete
#simply copy and paste each one of these commands below for each sample you want to count with HTSeq
#it is as easy as that
```

```
htseq-count -m intersection-nonempty --stranded=reverse alignLRWT1.sam alignlreuterijcm1112.gtf > LRWT1
```

Questions! Otherwise, that is all for now!

1.4 Analyzing *L. reuteri* data

We have our data! Hooray! Now, we can go through and analyze all of it, from Trimmomatic to HTSeq. We are only going to take it through HTSeq and then do the differential gene expression interactively on our own machines. This is simply because it's nice to work with everything on our own machines because we will be generating plots and may have to iteratively do the differential gene expression analysis based on the results/plots.

To this point, we have worked with a small subset of example RNA-seq data (100,000 reads per sample). Now, we will be working with full datasets (~15 million reads per sample). Thus, we are going to submit the jobs to the MSU HPCC to run for the next 20 hours or so.

Let's have a look at the raw data that we received from the MSU RTSF. We sequenced 24 samples but got 48 files back! Why did we get twice the data back? This is due to the limitations of the sequencer itself. The particular flowcell for our data has 2 lanes. Each lane generates around 150 million reads total and we want around 15 million reads per sample. Thus, 24 samples * 15 million reads = 360 million total reads. This is more than twice what a single lane can produce for us. Thus, we pooled all of our samples together and ran them on both lanes to achieve the desired read number. So in order to work with this data, we need to combine the reads from both lanes together into a single file. However, to do this effectively, we need to decompress the files and then merge them.

If you are interested in what I did to get to this point, the commands are as follows in pseudocode.

Decompress files: `gunzip -c file_I_want_to_decompress.fastq.gz > decompressed_file.fastq`

Merge files: `cat sample1_read_data_from_lane2.fastq >> sample1_read_data_from_lane1.fastq`

Recompress files: `gzip -c file_I_want_to_compress.fastq > compressed_file.fastq.gz`

I have already done all of this for you and grouped the merged and recompressed data into folders corresponding to different treatment comparisons:

LB vs Indole treated *L. reuteri*

LB vs Commensal *E. coli* conditioned medium treated *L. reuteri*

LB vs EHEC conditioned medium treated *L. reuteri*

These folders are located in `/mnt/research/mmg434/LreuteriRNAseqData/RawData`

Dr. Viswanathan has put together a list of who is doing which comparisons.

LAST_NAME	FIRST_NAME	Lbcontrol/ Sample #	Indole/ Sample #	E.coli conditioned medium/Sample #	EHEC conditioned medium/ Sample #
Asopa	Mrinal	A1	B1		
Babiker	Leena	A2	B2		
Biernat	Emily Rachel	A3	B3		
Cermak	Megan Molloy	A4	B4		
Collins	Sasha Lauren	A1	B5		
Freiberger	Katharina Marianne	A2	B6		
Ginzburg	Daniel	A3		C1	
Griffin	Caleigh Ellen	A4		C2	
Jones	Ashley Sade	A1		C3	
Kaminski	Theresa Marie	A2		C4	
Li	Irene	A3		C5	
Macko	Haley Marie	A4		C6	
Rahn	Christopher Louis Von				
Reske	Jake Jordan	A1			D1
Stawkey	Danielle Marie	A2			D2
Struble	Nicole Danielle	A3			D3
Tencer	Joel Ira	A4			D4
Vanalst	Andrew John	A1			D5
Wolfson	Abigail Esther	A2			D6

Now, since time is of the essence, I have written a script to do the analysis for us! Let's have a look at it:

```
#PBS -l walltime=20:00:00,nodes=1:ppn=4,mem=24gb
```

```
-----|
|This script was written *specifically* for the MMG434 course to analyze the |
|L. reuteri RNA-seq data generated by the MSU RTSF. This series of processes |
|will generate a folder called 'RNAseq_(currentdate)' in your home (~) direc-|
|tory. Within that folder a series of sub-folders are made to organize the |
|data and analysis results. The general workflow is Trimmomatic -> FastQC -> |
|Bowtie -> HTSeq. After this has been run, you can download the map.sam files|
|and analyze them in your favorite differential gene expression software pac-|
|lage (e.g. edgeR). To use this script, you will need to change the data loca-|
|tion and then save it (e.g. use nano or another text editor to make the cha-|
|nges). To run this script, follow the tutorial at mmg434.readthedocs.org. |
|-----|
```

```
#change to home directory on HPC
cd ~
```

```
#check if the folder RNAseq_(current date) exists and if not, make the folder
NEWFOLDNAME=RNAseq
DATE=`date +%d-%m-%y`
NEWFOLD=${NEWFOLDNAME}_${DATE}
```

```
#if the folder already exists, make a new one with a _number added
n=1 #start with the number as 1
```

```
while [ -d ~/${NEWFOLD} ] #while the folder exists, do the commands below
do
```

```
    NEWFOLD=${NEWFOLDNAME}_${DATE}_${n} #add the number to the folder name
    n=$((n+1)) #increment n by 1
```

```

done

mkdir $NEWFOLD
cd ~/$NEWFOLD

#make several folders within this RNAseq_(current date) folder to organize data
mkdir TrimmedData FastQCreports Bowtie HTSeq edgeR

#load the appropriate modules for all of the analysis steps
module load Trimmomatic/0.32
module load FastQC/0.11.2
module load bowtie/1.0.0
module load HTSeq/0.6.1

#Copy the data into the TrimmedData folder
#DEPENDING ON YOUR COMPARISON, YOU NEED TO CHANGE THE LBvsCOMM FOLDER LOCATION TO EITHER
#LBvsEHEC OR LBvsIndole
#IF YOU DO NOT CHANGE IT, YOU WILL RUN THE ANALYSIS FOR THE LBvsComm COMPARISON

cp /mnt/research/mmg434/LreuteriRNAseqData/RawData/LBvsComm/*.fastq.gz ~/$NEWFOLD/TrimmedData

#Do the trimming with Trimmomatic

cd ~/$NEWFOLD/TrimmedData
for untrimmedreads in ~/$NEWFOLD/TrimmedData/*
do
    FILENAME='basename ${untrimmedreads%.*.*}'
    PREFIX=trimmed
    NEWTRIMFILE=${PREFIX}${FILENAME}
    java -jar $TRIM/trimmomatic SE -threads 4 $untrimmedreads $NEWTRIMFILE.fastq.gz ILLUMINACLIP
done

#Generate the FastQC reports

cp ~/$NEWFOLD/TrimmedData/trimmed*.fastq.gz ~/$NEWFOLD/FastQCreports
cd ~/$NEWFOLD/FastQCreports
for trimmedreads in ~/$NEWFOLD/FastQCreports/trimmed*.fastq.gz
do
    fastqc $trimmedreads
done
rm trimmed*.fastq.gz

#Decompress the reads files and move them to the Bowtie directory

cd ~/$NEWFOLD/TrimmedData
cp trimmed*.fastq.gz ~/$NEWFOLD/Bowtie

for compfiles in ~/$NEWFOLD/Bowtie/*.fastq.gz
do
    FILENAME=${compfiles%.*.*}
    gunzip -c $FILENAME.fastq.gz > $FILENAME.fastq
done

cd ~/$NEWFOLD/Bowtie
rm *.fastq.gz      #clean the .gz files

#Align the reads with Bowtie to the reference genome

```

```
cd ~/$NEWFOLD/Bowtie

bowtie-build /mnt/research/mmg434/LreuteriRNAseqData/RefGenomeFiles/trimmedlreuterijcm1112.fa trimmedlreuterijcm1112

for renametrिम in ~/$NEWFOLD/Bowtie/*.fastq
do
    LESSPREFIX=`basename ${renametrिम//trimmed/}`
    mv $renametrिम ~/$NEWFOLD/Bowtie/$LESSPREFIX
done

for alignment in ~/$NEWFOLD/Bowtie/*.fastq
do
    FILENAME=`basename ${alignment%.*}`
    PREFIX=align
    NEWALIGNFILE=${PREFIX}${FILENAME}
    bowtie -S -p 3 trimmedlreuterijcm1112 $FILENAME.fastq > $NEWALIGNFILE.sam
done

rm ~/$NEWFOLD/Bowtie/*.fastq #clean the .fastq files

#Count gene features with HTSeq

cp ~/$NEWFOLD/Bowtie/align*.sam ~/$NEWFOLD/HTSeq
cp /mnt/research/mmg434/LreuteriRNAseqData/RefGenomeFiles/alignlreuterijcm1112.gtf ~/$NEWFOLD/HTSeq
cd ~/$NEWFOLD/HTSeq

for renamealign in ~/$NEWFOLD/HTSeq/*.sam
do
    LESSPREFIX=`basename ${renamealign//align/}`
    mv $renamealign ~/$NEWFOLD/HTSeq/$LESSPREFIX
done

for counts in ~/$NEWFOLD/HTSeq/*.sam
do
    FILENAME=`basename ${counts%.*}`
    PREFIX=map
    NEWMAPFILE=${PREFIX}${FILENAME}
    htseq-count -m intersection-nonempty --stranded=reverse $FILENAME.sam alignlreuterijcm1112.gtf
done

rm ~/$NEWFOLD/HTSeq/LR*.sam #clean the alignment files
```

This is a lot to take in! All this script is doing is linking all of the steps together and analyzing the data in one fell swoop for us. But! We **NEED** to change one thing in the script, depending on the comparison we are making.

Let's grab the script file:

```
cp /mnt/research/mmg434/LreuteriRNAseqData/LreuteriAnalysisScript.sh ~
```

Now, let's open it in the **nano** text editor:

```
nano LreuteriAnalysisScript.sh
```

It will look something like the script above. You can't use your mouse here and **must** use your arrow keys to get around. Let's read through the beginning of the script and see what we need to change.

Now, we will move down through the script and change the name of the folder containing the appropriate comparison (**unless you happen to be in the LB vs Commensal E. coli conditioned medium treated group!**).

To save the changes, hit Ctrl + o, followed by the Enter/Return key.

To exit nano, hit Ctrl + x

To run the script:

```
module load powertools
intel14-phi
module load powertools
cd ~
qsub LreuteriAnalysisScript.sh
```

Hooray! The script has been submitted to the queue to run. Write down your job ID.

1.5 Modification to the scripts from today (3-4-15)!

Please copy and paste the appropriate commands into your terminal on the HPCC to fix my mistake. Sorry everyone!

If you are in the LB vs Indole group:

```
cp /mnt/research/mmg434/LreuteriRNAseqData/LreuteriAnalysisScriptLBvsIndole.sh ~
module load powertools
intel14-phi
qsub LreuteriAnalysisScriptLBvsIndole.sh
```

If you are in the LB vs Commensal group:

```
cp /mnt/research/mmg434/LreuteriRNAseqData/LreuteriAnalysisScriptLBvsComm.sh ~
module load powertools
intel14-phi
qsub LreuteriAnalysisScriptLBvsComm.sh
```

If you are in the LB vs EHEC group:

```
cp /mnt/research/mmg434/LreuteriRNAseqData/LreuteriAnalysisScriptLBvsEHEC.sh ~
module load powertools
intel14-phi
qsub LreuteriAnalysisScriptLBvsEHEC.sh
```

These will run now. Something weird happened when I uploaded the files to the HPCC.

1.6 Differential gene expression analysis with edgeR

Up to this point we have done several things: trimmed, QC'd, aligned, and counted reads that mapped to each gene. Now, we will finally move to the step where we will analyze the differential gene expression between the untreated and treated *L. reuteri* samples!

To do this, we have chosen to utilize an analysis package written in the R programming language called `edgeR`. `edgeR` stands for differential expression analysis of digital gene expression data in R. This is a fantastic tool that is actively maintained (as seen by the date of the most recent user guide update) and fairly easy to use. Several diagnostic plots are produced throughout the analysis that provide meaningful information as to whether we can even perform differential gene expression between samples and if there are batch effects we have to deal with.

RNA-seq data does not typically assume a normal (Gaussian) distribution, so to glean which genes are changing in a statistically significant manner, we have to model the data slightly differently. `EdgeR` implements what is called a [negative binomial distribution](#), sometimes referred to as a gamma-Poisson model. If you *really* enjoy statistics and would like to dig into the mathematical underpinnings of this software, see the references at the bottom of this page.

If you are less interested in understanding the math behind all of this, here is the short summary: we need to examine the data to make sure they separate enough between treatments to determine differential gene expression and we *always* use a false-discovery rate correction to determine significance (even then, it's worth looking at the fold-change differences to decide if it is "real"; though this is slightly more arbitrary).

Let's get going!

We are going to do things in several steps today, with the goal being to be able to copy and paste as much of the code as we can.

1. *Download and install edgeR*
2. *Read in the data to RStudio*
3. *Group the data together*
4. *Filter out low counts*
5. *Regroup and normalize the libraries*
6. *See how samples separate by treatment*
7. *Differential expression calculations*
8. *Plot the results*

1.6.1 Download and install edgeR

The first step is to open RStudio and download/install edgeR from the Bioconductor repository.

1. To install packages from Bioconductor, click next to the `>` cursor. Type `source("http://bioconductor.org/biocLite.R")` and hit Enter/Return.

```
Console ~/ ↵

R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

Loading required package: limma
Loading required package: edgeR
>
>
>
>
>
>
>
> source("http://bioconductor.org/biocLite.R")
```

2. Then type `biocLite("edgeR")` and hit Enter/Return. RStudio should also install all the necessary dependencies as well.

Note: If RStudio asks “Update all/some/none? [a/s/n]:”, type **a** and then hit Enter/Return to update **all** of the outdated packages. It’s best to work with the most recent version of everything.

1.6.2 Read in the data to RStudio

The next step is to read in the data to RStudio.

IMPORTANT If you did *not* get your job to complete by now, please copy and paste the appropriate commands into your terminal on the HPCC to copy the necessary files into a folder called `LreuteriData` in your home directory

If you are in the LB vs Commensal group:

```
cd ~
mkdir LreuteriData
cd ~/LreuteriData
cp /mnt/research/mmg434/LreuteriRNAseqData/AnalyzedLBvsComm/*.sam ~/LreuteriData
```

If you are in the LB vs EHEC group:

```
cd ~
mkdir LreuteriData
cd ~/LreuteriData
cp /mnt/research/mmg434/LreuteriRNAseqData/AnalyzedLBvsEHEC/*.sam ~/LreuteriData
```

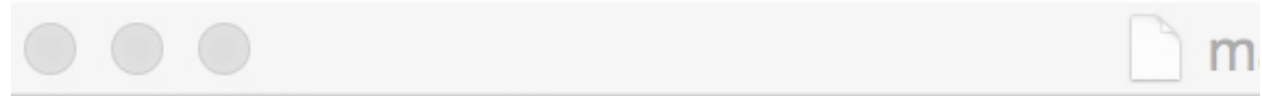
If you are in the LB vs Indole group:

```
cd ~
mkdir LreuteriData
cd ~/LreuteriData
cp /mnt/research/mmg434/LreuteriRNAseqData/AnalyzedLBvsIndole/*.sam ~/LreuteriData
```

1. We are going to download the data from the HPCC. Please download the `map.sam` files from the HTSeq directory to your desktop using FileZilla or WinSCP.

Note: We are going to need to edit each of these text files to remove the last five lines that will otherwise mess up the differential gene expression analysis.

- 1a. To do this, open the file in a text editor like TextEdit on Mac or Notepad on Windows.
- 1b. Scroll to the bottom of the file
- 1c. Remove the lines seen below in the screenshot starting at “`__no_feature`”.



LAR_tRNA40	40
LAR_tRNA41	0
LAR_tRNA42	102
LAR_tRNA43	22
LAR_tRNA44	165
LAR_tRNA45	38
LAR_tRNA46	503
LAR_tRNA47	206
LAR_tRNA48	128
LAR_tRNA49	58
LAR_tRNA50	160
LAR_tRNA51	3
LAR_tRNA52	1
LAR_tRNA53	9
LAR_tRNA54	19
LAR_tRNA55	41
LAR_tRNA56	48
LAR_tRNA57	2
LAR_tRNA58	7
LAR_tRNA59	42
LAR_tRNA60	29
LAR_tRNA61	3
LAR_tRNA62	1
LAR_tRNA63	0
__no_feature	1831425
__ambiguous	39
__too_low_aQual	0
__not_aligned	614984
__alignment_not_unique	0

1d. We also need to remove all the lines in the file that correspond to tRNA and rRNA.

1e. Teaching time: Didn't we remove all of the rRNA *before* we made the libraries for sequencing?! Guess we got most but not all...

Note: It is absolutely essential that we get rid of any line spaces at the bottom and between lines (e.g. after we get rid of the 16s rRNA lines). When you load it into Rstudio (next step), make sure you have exactly "1820 obs. of 1 variable".

2. Now, we need to read the files into RStudio. To do this we need to create a variable for each file. I will give an example for each treatment that you should be able to copy and paste into RStudio.

Mac users:

- **For LB control:** `wt1 = read.table("~/Desktop/mapLRLB1.sam", row.names=1)`
 - **For indole treated:** `in1 = read.table("~/Desktop/mapLRindole1.sam", row.names=1)`
 - **For E. coli commensal medium treated:** `co1 = read.table("~/Desktop/mapLRcomm1.sam", row.names=1)`
 - **For EHEC medium treated:** `eh1 = read.table("~/Desktop/mapLRehec1.sam", row.names=1)`
-

Note: Check to make sure you have "1820 obs. of 1 variable" by looking in the upper righthand corner of Rstudio (e.g. the Environment/Global Environment window).

Windows users:

This is only slightly more complicated for you. It's the same idea and naming convention, but we are going to use the Tab autocomplete function to help us determine the file path to the Desktop. To do this we are going to break the steps down using the LB control as an example:

1. Start typing in the command and place the cursor between the quotes so it looks like this (don't just copy and paste this in, type it out): `wt1 = read.table("")`
 2. Next, inside the quotes, type: `/Users/` and then hit the Tab key once. This will present you with a list of potential paths forward. The one you want will usually resemble your user name for the account on your computer (typically the first or second one, NOT "All users"). Click on the one that resembles your user name.
 3. Your command should now look something like this: `wt1 = read.table("/Users/yourusername/")`
 4. Next, you can complete the file path and will look something like this: `wt1 = read.table("/Users/yourusername/Desktop/mapLRLB1.sam")`
 5. Finally, we can complete the command: `wt1 = read.table("/Users/yourusername/Desktop/mapLRLB1.sam", row.names=1)`
-

Note: Check to make sure you have "1820 obs. of 1 variable" by looking in the upper righthand corner of Rstudio (e.g. the Environment/Global Environment window).

3. Repeat each of these commands for the respective treatment, making sure to change the variable name (e.g. `wt1`, `in1`, `co1`, `eh1`) each time (e.g. `wt2` for `LRWT2map.sam`).
 4. Now we need to rename the sample column names. To do this, I will give an example for each treatment that you should be able to copy and paste into RStudio.
 - **For LB control:** `colnames(wt1) <- 'wt1'`
 - **For indole treated:** `colnames(in1) <- 'in1'`
 - **For E. coli commensal medium treated:** `colnames(co1) <- 'co1'`
-

- **For EHEC medium treated:** `colnames(eh1) <- 'eh1'`

5. Repeat each of these commands for the respective treatment, making sure to change the variable name (e.g. `wt1`, `in1`, `co1`, `eh1`) each time (e.g. `wt2` for `LRWT2map.sam`) and the new name (e.g. `wt2`).

1.6.3 Group the data together

So that we don't have to work on each sample individually, we will put them all into a single variable.

Note: It is important that you put all of the same treatment type together in consecutive order (e.g. 1-7).

Note: It is important that you put the control treatment (LB) as the first set of samples for our purposes. Otherwise, it's simply important that you know what are your controls and treatments.

1. To do this, copy and paste one of the following depending on which treatments you are comparing.
 - **For LB vs Indole:** `wtvin <- cbind(wt1, wt2, wt3, wt4, in1, in2, in3, in4, in5, in6)`
 - **For LB vs Commensal conditioned medium:** `wtvco <- cbind(wt1, wt2, wt3, wt4, co1, co2, co3, co4, co5, co6, co7)`
 - **For LB vs EHEC conditioned medium:** `wtveh <- cbind(wt1, wt2, wt3, wt4, eh1, eh2, eh3, eh4, eh5, eh6, eh7)`
2. After you've copy and pasted this in, hit Enter/Return.
3. Now, let's group them by factor. Copy and paste the appropriate command based on what you are comparing.
 - **For LB vs Indole:** `group <- factor(c(1,1,1,1,2,2,2,2,2,2))`
 - **For LB vs Commensal conditioned medium:** `group <- factor(c(1,1,1,1,2,2,2,2,2,2))`
 - **For LB vs EHEC conditioned medium:** `group <- factor(c(1,1,1,1,2,2,2,2,2,2))`

Note: This step simply allows us to separate each treatment from each other (i.e. control vs. treatment).

1.6.4 Filter out low counts

It is important to filter out genes that have low read counts associated with them. This is because they will ultimately lead to a skewing of the data in subsequent steps of the analysis.

1. We will generate an edgeR data structure called a `DGEList`. This will create the scaffold with which edgeR can access the data and do differential gene expression. Copy and paste the appropriate command for what you are comparing and then hit Enter/Return.
 - **For LB vs Indole:** `y <- DGEList(counts=wtvin, group=group)`
 - **For LB vs Commensal conditioned medium:** `y <- DGEList(counts=wtvco, group=group)`
 - **For LB vs EHEC conditioned medium:** `y <- DGEList(counts=wtveh, group=group)`
2. Now we will filter out genes that have counts greater than 2 CPM (counts per million reads mapped) in at least four samples.
 - **Type:** `keep <- rowSums(cpm(y)>2) >= 4`

Note: We may come back and refine this number when we have a look at our final scatter plot of the data.

3. We will apply this filter we just made to our data set.

- **Type:** `y <- y[keep,]`

1.6.5 Regroup and normalize the libraries

For this next set of steps, we will regroup our data now that we have applied this filter and normalize everything based on effective library size to prevent sequencing depth and library size from skewing the data.

1. To regroup the data:

- **Type:** `y$samples$lib.size <- colSums(y$counts)`

2. Now let's normalize the data:

- **Type:** `y <- calcNormFactors(y)`

3. We can view what the scaling factor is by typing:

- **Type:** `y$samples`

1.6.6 See how samples separate by treatment

This is a *very* critical step. The results of this plot will let us know if we can proceed with differential gene expression. It will tell us whether our controls are separate enough from the treatment and if we have to deal with a batch effect.

Building the plot is easy at this point:

- **Type:** `plotMDS(y)`

1.6.7 Differential expression calculations

Now, assuming everything has passed the MDS plot. Let's move on to the differentially expressed genes.

1. Let's estimate the dispersion (variance):

- **Type:** `y <- estimateCommonDisp(y, verbose=TRUE)`
- **Type:** `y <- estimateTagwiseDisp(y)`

Note: An average BCV (biological coefficient of variation) for isogenic organisms in a lab setting (like what we are doing here) should be about 10-15%

2. We can plot the dispersion:

- **Type:** `plotBCV(y)`

Note: The results of this plot will give us an idea about the variances across all genes that are lowly expressed all the way to highly expressed. Normally, the more lowly expressed genes will have larger variation compared to the more highly expressed genes.

3. Now we can do the actual differential gene expression statistical test. In this case, we are going to use the exact test:

- **Type:** `res <- exactTest(y)`

4. To perform the FDR (false discovery rate) p-value correction:

- **Type:** `fdr <- p.adjust(res$table$PValue, method="BH")`

5. To extend our observations and compare consistency across samples within treatment, let's grab the CPM values per gene:
 - **Type:** `cpmres <- cpm(y)[rownames(res),]`
6. To quickly view how many genes are moving up and down:
 - **Type:** `summary(de <- decideTestsDGE(res))`
7. Let's export everything to the desktop:
 - **Type:** `write.csv(cpmres, file='~/Desktop/cpmresults.csv')`
 - **Type:** `write.csv(res$table, file='~/Desktop/DEresults.csv')`
 - **Type:** `write.csv(fdr, file='~/Desktop/fdrcorrection.csv')`

Note: Change the 'cpmresults.csv' file name to something more meaningful related to your sample comparison.

Note: Change the 'DEresults.csv' file name to something more meaningful related to your sample comparison.

Note: Change the 'fdrcorrection.csv' file name to something more meaningful related to your sample comparison.

1.6.8 Plot the results

Finally, let's look at a scatter plot where the red dots correspond to differentially expressed genes. The blue lines will indicate two-fold differential expression.

To generate the plot:

- **Type:** `detags <- rownames(y)[as.logical(de)]`
- **Type:** `plotSmear(res, de.tags=detags)`
- **Type:** `abline(h=c(-1,1), col='blue')`

Now marvel at your beautiful plot! Show your neighbor and be proud, you've navigated RNA-seq analysis successfully!

The final steps will be to take the three files you exported and put them together into a single Excel file, filter for genes with an adjusted p-value(FDR) < 0.05, and then filter genes that have two-fold differential expression (the logFC stands for logFoldChange, where it is log base 2; up two-fold is logFC=1, down two-fold is logFC=-1).

1.7 FastQC files

I hope you all had an excellent Spring Break! Here are all of the FastQC files to download for each sample.

LB Controls

- `trimmedLRLB1_fastqc.html`
- `trimmedLRLB2_fastqc.html`
- `trimmedLRLB3_fastqc.html`
- `trimmedLRLB4_fastqc.html`

Indole Treated

- trimmedLRindole1_fastqc.html
- trimmedLRindole2_fastqc.html
- trimmedLRindole3_fastqc.html
- trimmedLRindole4_fastqc.html
- trimmedLRindole5_fastqc.html
- trimmedLRindole6_fastqc.html

Commensal Conditioned Medium Treated

- trimmedLRcomm1_fastqc.html
- trimmedLRcomm2_fastqc.html
- trimmedLRcomm3_fastqc.html
- trimmedLRcomm4_fastqc.html
- trimmedLRcomm5_fastqc.html
- trimmedLRcomm6_fastqc.html
- trimmedLRcomm7_fastqc.html

EHEC Conditioned Medium Treated

- trimmedLRehec1_fastqc.html
- trimmedLRehec2_fastqc.html
- trimmedLRehec3_fastqc.html
- trimmedLRehec4_fastqc.html
- trimmedLRehec5_fastqc.html
- trimmedLRehec6_fastqc.html
- trimmedLRehec7_fastqc.html

Table 1.1: RNA-seq module outline

Day	Module
Fri 02/27	<i>Install software and use FileZilla to transfer files</i>
Fri 02/27	<i>RNA-seq background information, basic Linux/Unix commands, Trimmomatic, and FastQC</i>
Mon 03/02	<i>Align sequences with Bowtie and count gene features with HTSeq</i>
Wed 03/04	<i>Analyzing L. reuteri data</i>
Wed 03/04	<i>Modification to the scripts from today (3-4-15)!</i>
Fri 03/05	<i>Differential gene expression analysis with edgeR</i>
Mon 03/16	<i>FastQC files</i>

1.8 License

Creative commons 4.0 license

Full bacterial RNA-seq tutorial can be found [here](#)

1.9 Questions?

If you have additional questions, please e-mail Ben at john3434@msu.edu

Indices and tables

- *genindex*
- *modindex*
- *search*