labibi Documentation

Release 1.0

C. Titus Brown

August 02, 2015

Contents

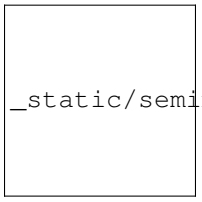
This is the material for two workshops given at MSU by Titus Brown and Matt Scholz. The first workshop (Dec 4/5) was on model organism mRNAseq, and the second workshop (Dec 10/11) will be on semi-model organism mRNAseq.

Both workshops were sponsored by iCER, http://icer.msu.edu/, and made use of the MSU High Performance Compute Center, http://hpcc.msu.edu/.

CHAPTER 1

2014 / December / mRNAseq on semi-model organisms

This workshop was given on December 10th and 11th, 2014, by C. Titus Brown and Matt Scholz.



What are semi-model organisms? One with bad transcriptomes.!

Tutorials:

1.1 Semi-model organisms and RNAseq: an overview of the options

When you're performing RNAseq on a non-model system *with* a genome but *without* a mature gene annotation or a mature functional annotation, you have a few options.

1. You can use the official gene annotation. Your analysis will then leave out any missing genes or unannotated genes, and your analysis will also be less sensitive to UTRs and may return incorrect results on incompletely annotated genes.
2. You can build your own gene models from your own (or others') data, merge them with the official gene models, and transfer annotations from the official gene models. This will give you a more robust analysis by extending UTRs and improving gene models, but you may end up with "anonymous" (unannotated) genes that are differentially expressed.
3. You can build your gene models, merge them with the official gene models, and both transfer annotations from the official gene models *and* build your own annotations. This is very time consuming but is probably the most sensitive!

Of these three options, the second is the one I most recommend for RNAseq from organisms with an evolutionarily close model system (e.g. all vertebrates). A few reasons –

- the official gene annotation for the model system should be pretty good, and those annotations will have been transferred effectively to your organism;
- in particular, you're unlikely to be missing entire genes (as long as they're in the genome sequence, they will probably be annotated!);

- but you *may* be missing UTRs or exons in the official annotation for *your* organism - RNAseq is cheaper and higher resolution than most previous methods for gene discovery.
- you may also be missing important genes for the process you're studying, but if they are not in the model reference, they will be unknown and therefore under- or un-annotated anyway; you'll see them with differential expression but won't have any information on what they do. That's OK and expected, but there's very little bioinformatics can do for you - you'll need to put in the time and effort to characterize the gene experimentally.

So, the upshot from this is that you can *increase sensitivity* by building your own gene models, but are probably ok without building your own annotations.

1.2 Short read quality and trimming

Note: Reminder: if you're on Windows, you should install mobaxterm.

Log into the HPC with SSH; use your MSU NetID and log into the machine 'hpc.msu.edu'. There copy/paste:

```
cd
module load powertools
getexample RNAseq-semimodel
```

This will put all of the example files for today in your home directory under the directory 'RNAseq-semimodel'.

1.2.1 0. Getting the data

http://genomebiology.com/content/14/3/R26

http://www.ebi.ac.uk/ena/data/view/SRA055442

http://www.ebi.ac.uk/ena/data/view/SAMN01096082 http://www.ebi.ac.uk/ena/data/view/SAMN01096083

http://www.ebi.ac.uk/ena/data/view/SAMN01096084 http://www.ebi.ac.uk/ena/data/view/SAMN01096085

Note that each sample has two replicates, and each replicate has two files.

Don't download them, but if you were downloading these yourself, you would want the "Fastq files (ftp)", both File 1 and File 2. (They take a few hours to download!)

We've already loaded the data onto the MSU HPC, and you've loaded it with 'module load powertools'.

To log into a compute node, type:

```
~/RNAseq-semimodel/login.sh
```

If this doesn't work, do:

```
ssh dev-intel14-phi
```

Now do:

```
ls -l ~/RNAseq-semimodel/data/
```

You'll see something like

```
-r--r--r-- 1 mscholz common-data 6781517200 Dec  9 09:46 SRR534005_1.fastq.gz
-r--r--r-- 1 mscholz common-data 7023515467 Dec  9 09:50 SRR534005_2.fastq.gz
-r--r--r-- 1 mscholz common-data 7285848617 Dec  9 09:41 SRR534006_1.fastq.gz
-r--r--r-- 1 mscholz common-data 7542383700 Dec  9 09:43 SRR534006_2.fastq.gz
```

```
-r--r--r-- 1 mscholz common-data 7219923066 Dec  9 09:47 SRR536786_1.fastq.gz
-r--r--r-- 1 mscholz common-data 7467116873 Dec  9 09:49 SRR536786_2.fastq.gz
-r--r--r-- 1 mscholz common-data 7694614208 Dec  9 09:40 SRR536787_1.fastq.gz
-r--r--r-- 1 mscholz common-data 7944043814 Dec  9 09:44 SRR536787_2.fastq.gz
```

These files are each approximately 7-8 GB in size!

1.2.2 1. Copying in some data to work with.

First, make a directory:

```
mkdir ~/rnaseq
cd ~/rnaseq
```

Copy in a subset of the data (100,000 reads):

```
gunzip -c ~/RNAseq-semimodel/data/SRR534005_1.fastq.gz | head -400000 | gzip > female_repl1_R1.fq.gz
gunzip -c ~/RNAseq-semimodel/data/SRR534005_2.fastq.gz | head -400000 | gzip > female_repl1_R2.fq.gz
```

These are FASTQ files – let's take a look:

```
less female_repl1_R1.fq.gz
```

(type 'q' to exit less)

Question:

- why are some files named SRR*?
- why are some files named female*?
- why are there R1 and R2 in the name?

Links:

- FASTQ Format

1.2.3 2. FastQC

We're going to use FastQC to summarize the data.

First, we need to load the FastQC software into our account:

```
module load FastQC/0.11.2
```

(You have to do this each time you log in and want to use FastQC.)

Now, run FastQC on both of the female files:

```
fastqc female_repl1_R1.fq.gz
fastqc female_repl1_R2.fq.gz
```

Now type 'ls':

```
ls
```

and you will see

```
female_repl1_R1_fastqc.html
female_repl1_R1_fastqc.zip
female_repl1_R2_fastqc.html
female_repl1_R2_fastqc.zip
```

Copy these to your laptop and open them in a browser. If you're on a Mac or Linux machine, you can type:

```
scp username@hpc.msu.edu:rnaseq/female*fastqc.* /tmp
```

and then open the html files in your browser. For Windows, if you're using mobaxterm, most of you should have a file transfer window on the left. Click 'refresh' (green circle icon fourth from the left) and then navigate into the 'rnaseq' folder; you should see the 'female_repl...' files there. Drag and drop those onto your Windows machine.

You can also view my versions: female_repl1_R1_fastqc.html and female_repl1_R2_fastqc.html

Questions:

- What should you pay attention to in the FastQC report?
- Which is "better", R1 or R2?

Links:

- FastQC
- FastQC tutorial video

1.2.4 3. Trimmomatic

Now we're going to do some trimming! We'll be using Trimmomatic. For a discussion of optimal RNAseq trimming strategies, see MacManes, 2014.

First, load the Trimmomatic software:

```
module load Trimmomatic/0.32
```

Next, run Trimmomatic:

```
java -jar $TRIM/trimmomatic PE female_repl1_R1.fq.gz female_repl1_R2.fq.gz\
     female_repl1_R1.qc.fq.gz s1_se female_repl1_R2.qc.fq.gz s2_se \
     ILLUMINACLIP:$TRIM/adapters/TruSeq3-PE.fa:2:40:15 \
     LEADING:2 TRAILING:2 \
     SLIDINGWINDOW:4:2 \
     MINLEN:25
```

You should see output that looks like this:

```
...
Quality encoding detected as phred33
Input Read Pairs: 100000 Both Surviving: 95583 (95.58%) Forward Only Surviving: 4262 (4.26%) Reverse
...
```

Questions:

- How do you figure out what the parameters mean?
- How do you figure out what parameters to use?
- What adapters do you use?
- What version of Trimmomatic are we using here? (And FastQC?)
- Are parameters different for RNAseq and genomic?

- What's with these annoyingly long and complicated filenames?
- What do we do with the single-ended files (s1_se and s2_se?)

Links:

- Trimmomatic

1.2.5 4. FastQC again

Run FastQC again:

```
fastqc female_repl1_R1.qc.fq.gz
fastqc female_repl1_R2.qc.fq.gz
```

(Note that you don't need to load the module again.)

Copy them to your laptop and open them, OR you can view mine: female_repl1_R1.qc_fastqc.html and female_repl1_R2.qc_fastqc.html

Let's take a look at the output files:

```
less female_repl1_R1.qc.fq.gz
```

(again, use 'q' to exit less).

Questions:

- Why are some of the reads shorter than others?
- is the quality trimmed data "better" than before?
- Does it matter that you still have adapters!?

1.2.6 5. Subset and trim the rest of the sequences

Copy and paste all of the below at once:

```
gunzip -c ~/RNAseq-semimodel/data/SRR534006_1.fastq.gz | head -400000 | gzip > female_repl2_R1.fq.gz
gunzip -c ~/RNAseq-semimodel/data/SRR534006_2.fastq.gz | head -400000 | gzip > female_repl2_R2.fq.gz

gunzip -c ~/RNAseq-semimodel/data/SRR536786_1.fastq.gz | head -400000 | gzip > male_repl1_R1.fq.gz
gunzip -c ~/RNAseq-semimodel/data/SRR536786_2.fastq.gz | head -400000 | gzip > male_repl1_R2.fq.gz

gunzip -c ~/RNAseq-semimodel/data/SRR536787_1.fastq.gz | head -400000 | gzip > male_repl2_R1.fq.gz
gunzip -c ~/RNAseq-semimodel/data/SRR536787_2.fastq.gz | head -400000 | gzip > male_repl2_R2.fq.gz

java -jar $TRIM/trimmomatic PE female_repl2_R1.fq.gz female_repl2_R2.fq.gz\
     female_repl2_R1.qc.fq.gz s1_se female_repl2_R2.qc.fq.gz s2_se \
     ILLUMINACLIP:$TRIM/adapters/TruSeq3-PE.fa:2:40:15 \
     LEADING:2 TRAILING:2 \
     SLIDINGWINDOW:4:2 \
     MINLEN:25

java -jar $TRIM/trimmomatic PE male_repl1_R1.fq.gz male_repl1_R2.fq.gz\
     male_repl1_R1.qc.fq.gz s1_se male_repl1_R2.qc.fq.gz s2_se \
     ILLUMINACLIP:$TRIM/adapters/TruSeq3-PE.fa:2:40:15 \
     LEADING:2 TRAILING:2 \
     SLIDINGWINDOW:4:2 \
     MINLEN:25
```

```
java -jar $TRIM/trimmomatic PE male_repl2_R1.fq.gz male_repl2_R2.fq.gz\
     male_repl2_R1.qc.fq.gz s1_se male_repl2_R2.qc.fq.gz s2_se \
     ILLUMINACLIP:$TRIM/adapters/TruSeq3-PE.fa:2:40:15 \
     LEADING:2 TRAILING:2 \
     SLIDINGWINDOW:4:2 \
     MINLEN:25
```

Next: Building a new reference transcriptome

1.3 Building a new reference transcriptome

The distinguishing feature of (what I call) semi-model organisms is that while they may have a decent genome reference, their transcriptome annotation is poor. There can be several reasons for this, but generally it boils down to lack of resources and/or attention – it takes a *lot* of effort to build a high quality transcriptome!

For this purpose, we've already installed the chicken reference genome set on the HPC (as part of the data you loaded at the beginning). In this case we've loaded in the Illumina iGenomes project into the RNAseq-semimodel location.

See paper:

> http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3334321/

1.3.1 Map all the reads to the genome with TopHat

We'll be using the TopHat software.

Do:

```
module load TopHat2/2.0.12

tophat -p 4 \
    -G  $HOME/RNAseq-semimodel/reference/Gallus_gallus/UCSC/galGal3/Annotation/Genes/genes.gtf \
    --transcriptome-index=$HOME/RNAseq-semimodel/tophat/transcriptome \
    -o tophat_all \
    $HOME/RNAseq-semimodel/reference/Gallus_gallus/UCSC/galGal3/Sequence/Bowtie2Index/genome \
 female_repl1_R1.qc.fq.gz,male_repl1_R1.qc.fq.gz,female_repl2_R1.qc.fq.gz,male_repl2_R1.qc.fq.gz \
 female_repl1_R2.qc.fq.gz,male_repl1_R2.qc.fq.gz,female_repl2_R2.qc.fq.gz,male_repl2_R2.qc.fq.gz
```

Questions:

- What are all these parameters?!
- How do we pick the transcriptome/genome?
- Why is it so slow?
- What is different about mapping RNAseq reads vs mapping genomic reads?

Links:

- TopHat manual
- Illumina iGenomes project

1.3.2 Evaluating the mapping

Check out the details:

```
less tophat_all/align_summary.txt
```

1.3.3 Build a new transcriptome ("ab initio") from the combined reads using Cufflinks

Now that we've mapped the reads, let's put them all together into exons and gene models:

```
module load cufflinks/2.2.1

cufflinks -o cuff_all tophat_all/accepted_hits.bam
```

Questions:

- What exactly is Cufflinks doing?

1.3.4 Merge the new transcriptome with the existing reference transcriptome

We already have some decent gene models; let's merge our new and the old ones:

```
ls -1 cuff_all/transcripts.gtf > cuff_list.txt

cuffmerge -g $HOME/RNAseq-semimodel/reference/Gallus_gallus/UCSC/galGal3/Annotation/Genes/genes.gtf \
    -o cuffmerge_all \
    -s $HOME/RNAseq-semimodel/reference/Gallus_gallus/UCSC/galGal3/Sequence/WholeGenomeFasta/genome.f
    cuff_list.txt
```

Do some cleanup:

```
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/remove-nostrand.py
python remove-nostrand.py cuffmerge_all/merged.gtf > cuffmerge_all/nostrand.gtf
```

Questions:

- why do you want to merge?
- why would you have a list of more than one thing in list.txt?
- come to think of it, why aren't you (re)mapping all your reads every time?
- what's with the 'remove nostrand' script?

1.3.5 Extracting your new transcriptome sequences

To get a look at the actual DNA sequences, do:

```
gffread -w cuffmerge_all.fa \
        -g $HOME/RNAseq-semimodel/reference/Gallus_gallus/UCSC/galGal3/Sequence/WholeGenomeFasta/geno
        cuffmerge_all/nostrand.gtf
```

Questions:

- What's the difference between a GTF file and the FA file?

1.3.6 Checking out your new transcriptome

Take a look at the top of your FASTA file:

```
head -30 cuffmerge_all.fa
```

Head on over to the chicken genome browser and try BLATing the sequence!

Next: Mapping reads to the transcriptome with TopHat

1.4 Mapping reads to the transcriptome with TopHat

Now that we have some quality-controlled reads and a new reference transcriptome, we're going to map the reads to the reference genome, **using the new reference transcriptome**. We'll again be using the TopHat software

1.4.1 Mapping reads

Load the TopHat software, if you haven't already:

```
module load TopHat2/2.0.12
```

And now run TopHat:

```
cd ~/rnaseq
tophat -p 4 \
    -G cuffmerge_all/nostrand.gtf \
    --transcriptome-index=$HOME/RNAseq-semimodel/tophat/merged \
    -o tophat_female_repl1 \
    ~/RNAseq-semimodel/reference/Gallus_gallus/UCSC/galGal3/Sequence/Bowtie2Index/genome \
    female_repl1_R1.qc.fq.gz female_repl1_R2.qc.fq.gz
```

This will take about 15 minutes.

Questions:

- How do we pick the transcriptome/genome?

1.4.2 Viewing the mapped reads percentage

Let's look at these numbers specifically:

```
less tophat_female_repl1/align_summary.txt
```

1.4.3 Making gene counts

Now that we know which reads go with which gene, we'll use htseq-count.

First, load the PySAM and HTSeq software packages:

```
module load HTSeq/0.6.1
```

And next, run HTSeq:

```
htseq-count --format=bam --stranded=no --order=pos \
    tophat_female_repl1/accepted_hits.bam \
    cuffmerge_all/nostrand.gtf > female_repl1_counts.txt
```

When this is done, type:

```
less female_repl1_counts.txt
```

(again, use 'q' to exit). These are your gene counts.

Note, these are *raw* gene counts - the number of reads that map to each feature (gene, in this case). They are not normalized by length of gene. According to this post on seqanswers, both DEseq and edgeR want exactly this kind of information!

Questions:

- what are the 'TCONS...' names?
- what do these parameters mean?
- what parameters does HTSeq take?
- why are we using so many programs?

Links:

- HTSeq
- htseq-count documentation

Next: Processing another sample with TopHat and HTSeq

1.5 Processing another sample with TopHat and HTSeq

Let's script this!

Type:

```
cd ~/rnaseq/
cat > chick-tophat-2.sh <<EOF
```

and then paste this in:

```
module load TopHat2/2.0.12
module load PySAM/0.6
module load HTSeq/0.6.1

# go to the 'rnaseq' directory in my home directory
cd ~/rnaseq

# now run Tophat!
tophat \
    -G cuffmerge_all/nostrand.gtf \
    --transcriptome-index=\$HOME/RNAseq-semimodel/tophat/merged \
    -o tophat_female_repl2 \
    ~/RNAseq-semimodel/reference/Gallus_gallus/UCSC/galGal3/Sequence/Bowtie2Index/genome \
    female_repl2_R1.qc.fq.gz female_repl2_R2.qc.fq.gz

htseq-count --format=bam --stranded=no --order=pos \
    tophat_female_repl2/accepted_hits.bam \
    cuffmerge_all/nostrand.gtf > female_repl2_counts.txt
```

```
EOF
```

(Be sure to press the Enter or Return key after pasting this in!) This is called a 'heredoc' and it gives a way to write a shell script via copy-paste ;).

Next, type:

```
bash chick-tophat-2.sh
```

This will run all of the commands in the file 'chick-tophat-2.sh'.

You can use the 'nano' editor to modify this file – type:

```
nano chick-tophat-2.sh
```

Next: Using the HPC to run jobs in parallel

1.6 Using the HPC to run jobs in parallel

Type:

```
cat > chick-tophat-3.sh <<EOF
```

and then paste this in:

```
#PBS -l walltime=24:00:00,nodes=1:ppn=2,mem=10gb
module load TopHat2/2.0.12
module load HTSeq/0.6.1

# go to the 'rnaseq' directory in my home directory
cd ~/rnaseq

# now run Tophat!
tophat \
    -G cuffmerge_all/nostrand.gtf \
    --transcriptome-index=\$HOME/RNAseq-semimodel/tophat/merged \
    -o tophat_male_repl1 \
    ~/RNAseq-semimodel/reference/Gallus_gallus/UCSC/galGal3/Sequence/Bowtie2Index/genome \
    male_repl1_R1.qc.fq.gz male_repl1_R2.qc.fq.gz

htseq-count --format=bam --stranded=no --order=pos \
    tophat_male_repl1/accepted_hits.bam \
    cuffmerge_all/nostrand.gtf > male_repl1_counts.txt

EOF
```

Next, do it for male_repl2:

```
cat > chick-tophat-4.sh <<EOF
```

and then paste this in:

```
#PBS -l walltime=24:00:00,nodes=1:ppn=2,mem=10gb
module load TopHat2/2.0.12
module load HTSeq/0.6.1

# go to the 'rnaseq' directory in my home directory
cd ~/rnaseq
```

```
# now run Tophat!
tophat \
    -G cuffmerge_all/nostrand.gtf \
    --transcriptome-index=\$HOME/RNAseq-semimodel/tophat/merged \
    -o tophat_male_repl2 \
    ~/RNAseq-semimodel/reference/Gallus_gallus/UCSC/galGal3/Sequence/Bowtie2Index/genome \
    male_repl2_R1.qc.fq.gz male_repl2_R2.qc.fq.gz

htseq-count --format=bam --stranded=no --order=pos \
    tophat_male_repl2/accepted_hits.bam \
    cuffmerge_all/nostrand.gtf > male_repl2_counts.txt

EOF
```

To submit these to the queue, do:

```
qsub chick-tophat-3.sh
qsub chick-tophat-4.sh
```

And now wait until they're done; you can figure that out by doing:

```
qstat -u $USER
```

to monitor the status of the job; once it goes away, it will be done.

Next: Data analysis & differential expression

1.7 Data analysis & differential expression

Note: if you want to start from here, you can do:

```
mkdir ~/rnaseq
cd ~/rnaseq
```

and then run the 'curl' commands below.

At this point, you should have four files:

```
female_repl1_counts.txt
female_repl2_counts.txt
male_repl1_counts.txt
male_repl2_counts.txt
```

If you look at the file content with 'head',

```
head female_repl1_counts.txt
```

you'll see something like this:

```
XLOC_000001     1
XLOC_000002     0
XLOC_000003     2
XLOC_000004     0
XLOC_000005     0
XLOC_000006     3
XLOC_000007     2
XLOC_000008     0
```

```
XLOC_000009     0
XLOC_000010     14
```

These are the *unique gene identifiers* from cuffmerge_all.fa, together with their counts as measured by TopHat and htseq-count.

What we'll do next is compare gene counts across all four files and do differential expression analysis.

We'll be using edgeR to do the basic differential expression analysis of our counts.

To run edgeR, you need to write a data loading and manipulation script in R. In this case, I've provided one – chick.R. This script will load in two samples with two replicates, execute an MA plot, do an MDS analysis/plot, and provide a spreadsheet with differential expression information in it. To download it, click here.

Links:

- False Discovery Rate
- Learn R with Swirl
- Data Carpentry

1.7.1 Running edgeR on a data subset

To run the script on the HPC, download the script and data files for the 100k read subsets. At the command line, do:

```
cd ~/rnaseq
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/chick.R
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/chick-subset/female_repl1_counts.txt
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/chick-subset/female_repl2_counts.txt
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/chick-subset/male_repl1_counts.txt
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/chick-subset/male_repl2_counts.txt
```

These various .txt files are produced by Mapping reads to the transcriptome with TopHat, Processing another sample with TopHat and HTSeq, and Using the HPC to run jobs in parallel.

Next, to run the R script, do:

```
module load R
Rscript chick.R
```

This will produce three files, chick-edgeR-MA-plot.pdf, chick-edgeR-MDS.pdf, and chick-edgeR.csv; they will be in your `rnaseq` folder in your home directory on the HPC. The CSV file can be opened directly in Excel; you can also look at it here. It consists of five columns: gene name, log fold change, P-value, and FDR-adjusted P-value.

If you look closely at the MA plot, you'll see that there are three red dots. These are the genes with a False Discovery Rate of 0.2 or less (see chick.R), line 28. Note that the axes on the MA plot are counts per million (CPM, X axis) and fold change (Y axis).

Next, let's take a look at chick-edgeR.csv. This is a comma-separated value file that you can download and open in Excel; go ahead and do so.

As you can see, it's got the two columns with fold change and counts per million; it's also got a P value and a FDR (false discovery rate) value for each gene. And, if you look at the first three rows, you'll see that these are three rows that yield the red dots on the MA plot! Hey, I wonder what those genes are...

Well, here is where the XLOC gene names are perhaps not that useful :). Let's go back and see if we can get any more information out of our transcriptome...

Links:

- edgeR tutorial from UT Austin

Questions:

- Why does the MA plot have the shape that it does?

1.7.2 Transferring "official" gene names from the official transcriptome

If you look at Building a new reference transcriptome, we used TopHat and Cufflinks to build new gene models from our RNAseq, and then merged the gene models with the already existing gene models from the official annotation. This gave us a file 'cuffmerge_all/nostrand.gtf' which contained gene annotaions and the gene coordinates for exons; from this, we extracted 'cuffmerge_all.fa', which contains a bunch of FASTA sequences. If you look at the top of *this* file, you'll see that the FASTA sequence names look like this:

>TCONS_00000001 gene=17.5

These 'TCONS' names are unique transcript identifiers; what we really want are the gene names, though. Unfortunately, we don't have TCONS, we have XLOC, which are unique *gene* identifiers. How do we turn those into gene names!?

If you look at cuffmerge_all/nostrand.gtf,

```
head -1 cuffmerge_all/nostrand.gtf
```

you'll see lines that contain information like this:

```
"XLOC_000001"; transcript_id "TCONS_00000002"; exon_number "1"; gene_name "17.5"; oId "NM_205429"; ne
```

There's the XLOC number, along with a bunch of other info! We want (at least!) two pieces of information from this - the gene name (here '17.5') and the nearest reference gene (here 'NM_205429'). How do we get those into the same spreadsheet as the differentially expressed genes?

As with the R script above, this is a situation where a little bit of scripting comes in handy - I've written a small Python script to do this, add-gene-name-to-diffexpr-csv.py.

To download and run it, do:

```
curl -O https://raw.githubusercontent.com/ngs-docs/2014-msu-rnaseq/master/files/add-gene-name-to-diff
python add-gene-name-to-diffexpr-csv.py cuffmerge_all/nostrand.gtf chick-edgeR.csv > chick-edgeR-name
```

You can download my copy of this file and open it in Excel, or you can just look at it online. And hey, look, gene names!

You can look up the NM_ stuff in genbank (actually, googling "genbank NM_204286" will bring you right to a birdbase link), and the gene names can be fed direclty into services like DAVID.

One quick note before we move on – it's important to realize that we didn't do any clever analysis to get the gene name and nearest reference gene information into this file. It was simply transferred from the official gene annotation for chick when we ran cuffmerge. We'll talk a little bit about how to generate your own annotations later.

1.7.3 Working with DAVID

When you're interested in looking at enrichment of functional gene categories, the DAVID tool for gene enrichment analysis is a common recommendation. The essential idea is to look at some selection of genes (ones that are differentially expressed, usually!) in the background context of a much larger set of genes (all expressed genes that are not differentially expressed).

The simplest way to do this is to pick an FDR, and select all gene accessions above that FDR. For example:

- go to DAVID;

- Select 'upload', and paste in the first 1,000 accessions from chick-edgeR-named;
- Under "Select identifier", choose "GENBANK_ACCESSION";
- Select "Gene List" for List Type;
- and then "Submit List".

  DAVID will now tell you that less than 80% of the list has mapped; that's expected, since there are a number of blank spots in the list. Select "Continue to submit the IDs that DAVID could map."
- You should now be on Step 2. Select "Functional annotation tool." Go to that link.
- Now, each of the three views (Clustering, Chart, and Table) will give you more information.

For me, under Clustering, Annotation Cluster 6 shows an enrichment of sex-related genes, so I guess that's good, since we're comparing male and female blastoderm gene expression from chick! But this also highlights the problems with this kind of analysis – we can see what we want! Bear in mind that we are really looking more at the background of *what genes are expressed* than what genes are *differentially* expressed; to do the latter, we'd need to do a larger analysis.

Next: Data analysis 2: more sequence fu

1.8 Data analysis 2: more sequence fu

In Data analysis & differential expression, we talked about using existing annotations, transferred to your new gene models from the primary gene models. What if you want to annotate any new genes (genes that weren't in the original annotation)?

This is much more technically challenging, and the pipeline doesn't work entirely on the HPC just yet. Here are some tips to get started.

1.8.1 Running TransDecoder to turn transcripts into ORFs

cufflinks produces transcripts, but many programs want protein sequences. TransDecoder will turn transcripts into (predicted) peptide sequences:

```
module load TransDecoder
TransDecoder.LongOrfs -t cuffmerge_all.fa

curl -O https://raw.githubusercontent.com/ngs-docs/2014-msu-rnaseq/master/files/remove-stop-codon.py
python remove-stop-codon.py cuffmerge_all.fa.transdecoder_dir/longest_orfs.pep > cuffmerge_orfs.pep
```

The file 'cuffmerge_orfs.pep' now contains entries that look like this:

```
>TCONS_00000001|m.1 TCONS_00000001|g.1 type:complete len:173 gc:universal TCONS_00000001:696-1214(+)
MCVPAWGLDQLNPLPQREVLGVQQSRQLRPLKEGDKSAVTDPIPGSSCQRCWALRAISVCWSPPISSTATLLFLVWTLPA
VLLAVSQETQLLHCALKGGPLGVLTDQIPMVRLGVQLTITAADPWSCQDLEQRSQQCEHSTEQCCVPRGSAAIWCSRFPL
DSAQFSPLTLLP
```

– which is to say, protein sequences rather than DNA transcripts as in cuffmerge_all.fa.

These can now be fed into InterProScan.

1.8.2 Running InterProScan (iprscan)

InterProScan will go through and integrate information from a number of databases into an annotation of your sequences.

```
module load iprscan
# interproscan.sh -i cuffmerge_orfs.pep -f tsv
```

(The last command doesn't work yet on the HPC!)

This will give you output in a tab-delimited format, cuffmerge_orfs.pep.tsv.

This can then potentially be used to annotate any new genes with GO terms and other putative functional annotation.

Next: Miscellaneous advice

1.9 Miscellaneous advice

1.9.1 Sequencing depth and number of samples

Hart et al. (2013) provides a nice description and a set of tools for estimating your needed sequencing depth and number of samples. They provide an Excel based calculator for calculating number of samples. Their numbers are surprisingly large to me ;).

In a proposal for an exploratory effort to discover differentially expressed genes, I would suggest 3-5 biological replicates with 30-50 million reads each. More reads is usually cheaper than more replicates, so 50-100m reads may give you more power to resolve smaller fold changes.

1.9.2 Downloading your data

If you do your sequencing at the MSU Core Facility, you'll get an e-mail from them when you're samples are ready. The e-mail will give you an FTP site, a username, and a password, as well as a URL. You can use these to download your data. For example, if you get:

```
hostname:       titan.bch.msu.edu
username:       rnaseqmodel
password:       QecheJa6

URI:            ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu
```

you can go to ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu in your Web browser; that is, it lets you combine your username and password to open that link.

In this case, you will see a 'testdata' directory. If you click on that, you'll see a bunch of fastq.gz files. These are the files that you want to get onto the HPC.

To download these files onto the HPC, log into the HPC, go to the directory on the HPC you want to put the files in, and run a 'wget' – for example, on the HPC:

```
mkdir ~/testdata
cd ~/testdata

wget -r -np -nH ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu/testdata/
```

This will download _all_ of the files in that directory. You can also do them one at a time, e.g. to get 'Ath_Mut_1_R1.fastq.gz', you would do

wget ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu/testdata/Ath_Mut_1_R1.fastq.gz

Tada!

1.9.3 Developing your own pipeline

Even if all you plan to do is change the filenames you're operating on, you'll need to develop your own analysis pipeline. Here are some tips.

1. Start with someone else's approach; don't design your own. There are lots of partly done examples that you can find on the Web, including in this tutorial.
2. Generate a data subset (the first few 100k reads, for example).
2. Run commands interactively on an HPC dev node until you get all of the commands basically working; track all of your commands in a Word document or some such.
3. Once you have a set of commands that seems to work on small data, write a script. Run the script on the small data again; make sure that works.
4. Turn it into a qsub script (making sure you're in the right directory, have the modules loaded, etc.)
5. Make sure the qsub script works on your same small data.
6. Scale up to a big test data set.
7. Once that's all working, SAVE THE SCRIPT SOMEWHERE. Then, edit it to work on all your data sets (you may want to make subsets again, as much as possible).
8. Provide your scripts and raw counts files as part of any publication or thesis, perhaps via figshare.

Next: More resources

1.10 More resources

1.10.1 Informational resources

UT (Austin) Sequencing Core prices - costs and yields for sequencing.

ANGUS - summer NGS course - lots of resources and materials and book reference

Data Carpentry - intro to R, etc.

Software Carpentry - more scripting, Python, etc.

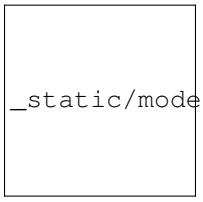
1.10.2 Places to share data, scripts, and results files

Figshare.

CHAPTER 2

2014 / December / mRNAseq on model organisms

This workshop was given on December 4th and 5th, 2014, by C. Titus Brown and Matt Scholz.



Tutorials:

2.1 Short read quality and trimming

Log into the HPC with SSH; use your MSU NetID and log into the machine 'gateway.hpcc.msu.edu'. There copy/paste:

```
module load powertools
getexample RNAseq-model
```

and you should see something about linking.

2.1.1 0. Getting the data

We'll be using a few RNAseq data sets from Fagerberg et al., Analysis of the Human Tissue-specific Expression by Genome-wide Integration of Transcriptomics and Antibody-based Proteomics.

You can get this data from the European Nucleotide Archive under ERP003613 – go to http://www.ebi.ac.uk/ena/data/view/ERP003613 to see all the samples.

I picked two samples: salivary gland and lung. Note that each sample has two replicates, and each replicate has two files.

Don't download them, but if you were downloading these yourself, you would want the "Fastq files (ftp)", both File 1 and File 2. (They take a few hours to download!)

We've already loaded the data onto the MSU HPC, and you've loaded it with 'module load powertools'.

To log into a compute node, type:

```
~/RNAseq-model/login.sh
```

If this doesn't work, do:

```
ssh dev-intel14-phi
```

Now do:

```
ls -la ~/RNAseq-model/data/
```

You'll see something like

```
-r--r--r-- 1 mscholz common-data 3714262571 Dec  4 08:44 ERR315325_1.fastq
-r--r--r-- 1 mscholz common-data 3714262571 Dec  4 08:44 ERR315325_2.fastq
-r--r--r-- 1 mscholz common-data 2365633645 Dec  4 08:44 ERR315326_1.fastq
-r--r--r-- 1 mscholz common-data 2365633645 Dec  4 08:44 ERR315326_2.fastq
```

which tells you that this file is 900,000,000 bytes or about 900 MB. Quite large!

2.1.2 1. Copying in some data to work with.

First, make a directory:

```
mkdir ~/rnaseq
cd ~/rnaseq
```

Copy in a subset of the data (100,000 reads):

```
head -400000 ~/RNAseq-model/data/ERR315325_1.fastq | gzip > salivary_repl1_R1.fq.gz
head -400000 ~/RNAseq-model/data/ERR315325_2.fastq | gzip > salivary_repl1_R2.fq.gz
```

These are FASTQ files – let's take a look:

```
less salivary_repl1_R1.fq.gz
```

(type 'q' to exit less)

Question:

- why are some files named ERR*?
- why are some files named salivary*?
- why is there R1 and R2 in the name?

Links:

- FASTQ Format

2.1.3 2. FastQC

We're going to use FastQC to summarize the data.

First, we need to load the FastQC software into our account:

```
module load FastQC/0.11.2
```

(You have to do this each time you log in and want to use FastQC.)

Now, run FastQC on both of the salivary gland files:

```
fastqc salivary_repl1_R1.fq.gz
fastqc salivary_repl1_R2.fq.gz
```

Now type 'ls':

```
ls
```

and you will see

```
salivary_repl1_R1_fastqc.html
salivary_repl1_R1_fastqc.zip
salivary_repl1_R2_fastqc.html
salivary_repl1_R2_fastqc.zip
```

Copy these to your laptop and open them in a browser. If you're on a Mac or Linux machine, you can type:

```
scp username@hpc.msu.edu:salivary*fastqc.* /tmp
```

and then open the html files in your browser. For Windows, we'll figure it out ;).

You can also view my versions: salivary_repl1_R1_fastqc.html and salivary_repl1_R2_fastqc.html

Questions:

- What should you pay attention to in the FastQC report?
- Which is "better", R1 or R2?

Links:

- FastQC
- FastQC tutorial video

2.1.4 3. Trimmomatic

Now we're going to do some trimming! We'll be using Trimmomatic. For a discussion of optimal RNAseq trimming strategies, see MacManes, 2014.

First, load the Trimmomatic software:

```
module load Trimmomatic/0.32
```

Next, run Trimmomatic:

```
java -jar $TRIM/trimmomatic PE salivary_repl1_R1.fq.gz salivary_repl1_R2.fq.gz\
     salivary_repl1_R1.qc.fq.gz s1_se salivary_repl1_R2.qc.fq.gz s2_se \
     ILLUMINACLIP:$TRIM/adapters/TruSeq3-PE.fa:2:40:15 \
     LEADING:2 TRAILING:2 \
     SLIDINGWINDOW:4:2 \
     MINLEN:25
```

You should see output that looks like this:

```
...
Input Read Pairs: 100000 Both Surviving: 95236 (95.24%) Forward Only Surviving: 4764 (4.76%) Reverse
TrimmomaticPE: Completed successfully
```

Questions:

- How do you figure out what the parameters mean?
- How do you figure out what parameters to use?

- What adapters do you use?
- What version of Trimmomatic are we using here? (And FastQC?)
- Are parameters different for RNAseq and genomic?
- What's with these annoyingly long and complicated filenames?
- What do we do with the single-ended files (s1_se and s2_se?)

Links:

- Trimmomatic

2.1.5 4. FastQC again

Run FastQC again:

```
fastqc salivary_repl1_R1.qc.fq.gz
fastqc salivary_repl1_R2.qc.fq.gz
```

(Note that you don't need to load the module again.)

Copy them to your laptop and open them, OR you can view mine: salivary_repl1_R1.qc_fastqc.html and salivary_repl1_R2.qc_fastqc.html

Let's take a look at the output files:

```
less salivary_repl1_R1.qc.fq.gz
```

(again, use 'q' to exit less).

Questions:

- Why are some of the reads shorter than others?
- is the quality trimmed data "better" than before?
- Does it matter that you still have adapters!?

Next: Mapping reads to the transcriptome with TopHat

2.2 Mapping reads to the transcriptome with TopHat

Now that we have some quality-controlled reads, we're going to *map* the reads to the reference gene set, for the purpose of counting how many reads have come from each gene. We'll be using the TopHat software

For this purpose, we've already installed the human reference gene set on the HPC (as part of the data you loaded at the beginning). In this case we've loaded in the Illumina iGenomes project into the RNAseq-model data set.

2.2.1 Mapping reads

Load the TopHat software:

```
module load TopHat2/2.0.12
```

And now run TopHat:

```
cd ~/rnaseq
tophat -p 4 \
    -G ~/RNAseq-model/Homo_sapiens/Ensembl/GRCh37/Annotation/Genes/genes.gtf \
    --transcriptome-index=$HOME/RNAseq-model/transcriptome \
    -o tophat_salivary_repl1 \
    ~/RNAseq-model/Homo_sapiens/Ensembl/GRCh37/Sequence/Bowtie2Index/genome \
    salivary_repl1_R1.qc.fq.gz salivary_repl1_R2.qc.fq.gz
```

This will take about 15 minutes.

Questions:

- What are all these parameters?!
- How do we pick the transcriptome/genome?
- Why is it so slow?
- What is different about mapping RNAseq reads vs mapping genomic reads.

Links:

- TopHat manual
- Illumina iGenomes project

2.2.2 Counting mapped reads percentage

Let's ask samtools for the total number of reads that mapped:

```
samtools view -c -F 4 tophat_salivary_repl1/accepted_hits.bam
```

You should get around 179,312.

If we look at the FastQC report, we can see (at the top) that the total number of reads in the R1 file is 95,236 (which is the same number of reads as in the R2 file, because we only looked at the paired reads that came out of Trimmomatic).

In total, that's 190,472 reads in your trimmed (QC) data, and 179,312 mapped – about 94%. That's pretty good!

Next: Counting the reads that map to each gene

2.3 Counting the reads that map to each gene

Now that we know which reads go with which gene, we'll use htseq-count.

First, load the PySAM and HTSeq software packages:

```
module load PySAM/0.6
module load HTSeq/0.6.1
```

And next, run HTSeq:

```
htseq-count --format=bam --stranded=no --order=pos \
    tophat_salivary_repl1/accepted_hits.bam \
    ~/RNAseq-model/Homo_sapiens/Ensembl/GRCh37/Annotation/Genes/genes.gtf > salivary_repl1_counts.txt
```

When this is done, type:

```
less salivary_repl1_counts.txt
```

(again, use 'q' to exit). These are your gene counts.

Note, these are *raw* gene counts - the number of reads that map to each feature (gene, in this case). They are not normalized by length of gene. According to this post on seqanswers, both DEseq and edgeR want exactly this kind of information!

Questions:

- what are the 'ENS*...' names?
- what do these parameters mean?
- what parameters does HTSeq take?
- why are we using so many programs?

Links:

- HTSeq
- htseq-count documentation

Next: Processing another sample with TopHat and HTSeq

2.4 Processing another sample with TopHat and HTSeq

Let's script this!

Type:

```
cd ~/rnaseq/
module load Trimmomatic/0.32
cat > lung-tophat.sh <<EOF
```

and then paste this in:

```
module load Trimmomatic/0.32
module load TopHat2/2.0.8b
module load PySAM/0.6
module load HTSeq/0.6.1

# go to the 'rnaseq' directory in my home directory
cd ~/rnaseq

# subset the data sets (100,000 reads) - you don't want to do this
# on real data :)
head -400000 ~/RNAseq-model/data/ERR315326_1.fastq | gzip > lung_repl1_R1.fq.gz
head -400000 ~/RNAseq-model/data/ERR315326_2.fastq | gzip > lung_repl1_R2.fq.gz

# run Trimmomatic.  Here, the inputs are 'lung_repl1_R1.fq.gz' and
# 'lung_repl1_R2.fq.gz', and the outputs are 'lung_repl1_R1.qc.fq.gz'
# and 'lung_repl1_R2.qc.fq.gz'.
java -jar \$TRIM/trimmomatic PE lung_repl1_R1.fq.gz lung_repl1_R2.fq.gz \
  lung_repl1_R1.qc.fq.gz s1_se lung_repl1_R2.qc.fq.gz s2_se \
  ILLUMINACLIP:\$TRIM/adapters/TruSeq2-PE.fa:2:40:15 \
  LEADING:2 TRAILING:2 \
  SLIDINGWINDOW:4:2 \
  MINLEN:25
```

```
# now run Tophat!
# The inputs are the outputs of the previous Trimmomatic step.
# The outputs are going to be under the 'tophat_lung_repl1' directory.
tophat -p 4 \
    -G ~/RNAseq-model/Homo_sapiens/Ensembl/GRCh37/Annotation/Genes/genes.gtf \
    --transcriptome-index=$HOME/RNAseq-model/transcriptome \
    -o tophat_lung_repl1 \
    ~/RNAseq-model/Homo_sapiens/Ensembl/GRCh37/Sequence/Bowtie2Index/genome \
    lung_repl1_R1.qc.fq.gz lung_repl1_R2.qc.fq.gz

# count the hits by gene -- 'tophat_lung_repl1' is the main output,
# from Tophat.
htseq-count --format=bam --stranded=no --order=pos tophat_lung_repl1/accepted_hits.bam \
    ~/RNAseq-model/Homo_sapiens/Ensembl/GRCh37/Annotation/Genes/genes.gtf > lung_repl1_counts.txt
EOF
```

(Be sure to press the Enter or Return key after pasting this in!) This is called a 'heredoc' and it gives a way to write a shell script via copy-paste ;).

Next, type:

```
bash lung-tophat.sh
```

This will run all of the commands in the file 'lung-tophat.sh'.

You can use the 'nano' editor to modify this file – type:

```
nano lung-tophat.sh
```

Next: Data analysis & differential expression

2.5 Data analysis & differential expression

We'll be using edgeR to do the basic differential expression analysis of our counts.

To run edgeR, you need to write a data loading and manipulation script in R. In this case, I've provided one – lung_saliva.R. This script will load in two samples with two replicates, execute an MA plot, and provide a spreadsheet with differential expression information in it. To download it, click here.

Links:

- False Discovery Rate
- Learn R with Swirl
- Data Carpentry

2.5.1 Running edgeR on a data subset

To run the script on the HPC, download the script and data files for the 100k read subsets. At the command line, do:

```
cd ~/rnaseq
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/lung_saliva.R
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/subset/lung_repl1_counts.txt
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/subset/lung_repl2_counts.txt
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/subset/salivary_repl1_counts.txt
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/subset/salivary_repl2_counts.txt
```

Note: the `saliva_repl1_counts.txt` and `lung_repl1_counts.txt` are the files produced by Mapping reads to the transcriptome with TopHat and Processing another sample with TopHat and HTSeq, respectively. I've also run them on the replicate data sets, which produced the *repl2_counts.txt files.

Next, to run the R script, do:

```
module load R
Rscript lung_saliva.R
```

This will produce two files, edgeR-MA-plot.pdf and edgeR-lung-vs-salivary.csv; they will be in your `rnaseq` folder in your home directory on the HPC. The CSV file can be opened directly in Excel; you can also look at it here. It consists of five columns: gene name, log fold change, P-value, and FDR-adjusted P-value.

Links:

- edgeR tutorial from UT Austin

2.5.2 Functional and network analysis on differentially expressed genes

There are a number of sites that let you analyze gene lists, including DAVID. I have gotten a number of recommendations for PANTHER. PANTHER lets you upload gene lists and explore their functional categories interactively or in a gene list/annotation format.

More specifically, you can explore your RNAseq data with

- functional classifications, in pie chart or in list;
- tests for statistical overrepresentation;
- tests for statistical enrichment based on associated fold change.

You need to crop and transform the data a little bit before using the functional classification. The steps are:

1. Download the CSV file. If you've produced your own, copy it over from the HPC.
2. Open it in Excel.
3. Choose an FDR cutoff (suggest FDR < 0.05 or lower) and delete all the rows after that (or, copy the rows into a new spreadsheet – might be quicker).
4. Save as a "Tab-delimited text." (Note, on Mac OS X, you may need to save this as "Windows formatted text" instead.)

This is now a file you can upload to PANTHER.

(You can download my copy of this here)

To do the analysis, go to http://www.pantherdb.org/. In box 1, select "Choose file" and find the CSV file you want to upload to PANTHER. Nothing else in box one should be changed.

In box 2, select "Homo sapiens."

In box 3, select either "Functional analysis classification viewed in gene list" or "Functional analysis classification viewed in pie chart."

Click submit.

Now you can explore the results!

Links:

- PANTHER database

Next: Submitting jobs to the MSU HPC queue

2.6 Submitting jobs to the MSU HPC queue

In Processing another sample with TopHat and HTSeq, we showed you a *shell script*, which was a way of telling the computer to do multiple things in a row. We discussed several advantages to scripting –

1. It automates long-running processes;
2. It tracks what you did, and you can edit it (to modify analyses) and also copy it (to do a family of analyses);
3. You can also provide it as part of your Methods in your paper;

There is also a fourth advantage, or really a necessity, to scripting: it's how you use the MSU HPC to run your computation. Briefly, to run a job on the HPC, you create a shell script and then run 'qsub'.

Here are four scripts (plus a common library script) that you could use to run some analyses on the HPC. To run, do something like this:

```
module load powertools
getexample RNAseq-model

mkdir ~/rnaseq
mkdir ~/rnaseq/script
cd ~/rnaseq/script
ln -fs ~/RNAseq-model/data/*.fastq .

curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/subset/env.sh
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/subset/process-1.sh
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/subset/process-2.sh
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/subset/process-3.sh
curl -O http://2014-msu-rnaseq.readthedocs.org/en/latest/_static/subset/process-4.sh

qsub process-1.sh
qsub process-2.sh
qsub process-3.sh
qsub process-4.sh
```

You can use 'qstat | grep <username>' to check on your jobs' status.

Next: Miscellaneous advice

2.7 Miscellaneous advice

2.7.1 Sequencing depth and number of samples

Hart et al. (2013) provides a nice description and a set of tools for estimating your needed sequencing depth and number of samples. They provide an Excel based calculator for calculating number of samples. Their numbers are surprisingly large to me ;).

In a proposal for an exploratory effort to discover differentially expressed genes, I would suggest 3-5 biological replicates with 30-50 million reads each. More reads is usually cheaper than more replicates, so 50-100m reads may give you more power to resolve smaller fold changes.

2.7.2 Downloading your data

If you do your sequencing at the MSU Core Facility, you'll get an e-mail from them when you're samples are ready. The e-mail will give you an FTP site, a username, and a password, as well as a URL. You can use these to download

your data. For example, if you get:

```
hostname:       titan.bch.msu.edu
username:       rnaseqmodel
password:       QecheJa6

URI:            ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu
```

you can go to ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu in your Web browser; that is, it lets you combine your username and password to open that link.

In this case, you will see a 'testdata' directory. If you click on that, you'll see a bunch of fastq.gz files. These are the files that you want to get onto the HPC.

To download these files onto the HPC, log into the HPC, go to the directory on the HPC you want to put the files in, and run a 'wget' – for example, on the HPC:

```
mkdir ~/testdata
cd ~/testdata

wget -r -np -nH ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu/testdata/
```

This will download _all_ of the files in that directory. You can also do them one at a time, e.g. to get 'Ath_Mut_1_R1.fastq.gz', you would do

wget ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu/testdata/Ath_Mut_1_R1.fastq.gz

Tada!

2.7.3 Developing your own pipeline

Even if all you plan to do is change the filenames you're operating on, you'll need to develop your own analysis pipeline. Here are some tips.

1. Start with someone else's approach; don't design your own. There are lots of partly done examples that you can find on the Web, including in this tutorial.
2. Generate a data subset (the first few 100k reads, for example).
2. Run commands interactively on an HPC dev node until you get all of the commands basically working; track all of your commands in a Word document or some such.
3. Once you have a set of commands that seems to work on small data, write a script. Run the script on the small data again; make sure that works.
4. Turn it into a qsub script (making sure you're in the right directory, have the modules loaded, etc.)
5. Make sure the qsub script works on your same small data.
6. Scale up to a big test data set.
7. Once that's all working, SAVE THE SCRIPT SOMEWHERE. Then, edit it to work on all your data sets (you may want to make subsets again, as much as possible).
8. Provide your scripts and raw counts files as part of any publication or thesis, perhaps via figshare.

Next: More resources

2.8 More resources

2.8.1 Informational resources

UT (Austin) Sequencing Core prices - costs and yields for sequencing.

ANGUS - summer NGS course - lots of resources and materials and book reference

Data Carpentry - intro to R, etc.

Software Carpentry - more scripting, Python, etc.

2.8.2 Places to share data, scripts, and results files

Figshare.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search