UND Genomics Core Bioinformatics Workshop
February 23, 2019

Section 1 – Sequence Quality and Alignment

In hands on activity, we're going to build on the basics we learned yesterday to set up our workspace and begin to work with sequencing files and applications. We're going to look at a program that will help us examine the quality of a raw sequence file. We will also align a sequence file to the genome and see what those results look like.

Notes:

1. All the code you need to type into the terminal will be in black boxes.
2. Mac users can simply copy paste the code into the terminal.
3. Windows users can copy code into the putty terminal by:
    a. Copy code using ctrl + C
    b. Paste into the putty terminal by right clicking with your mouse.
4. If you are typing the commands into the terminal, type the code as one continuous line.
5. Due to the large number of people working at the same time, for the more computationally extensive steps- alignment, and transcriptome assembly, we will have you start the command, then kill the process using ctrl + C. We will have output files already generated for you to examine.

Login to server

Login to your assigned server.

1.Make sure VPN is turned on.

2. Windows. You will nee the putty application to login.
    a. Username: workshop.2019
    b. Password: UNDworkshop!

2. Macs. Using the terminal app,
    a. Type: `ssh workshop.2019@buddy.med.und.edu`
    b. You will be prompted for a password- password is: UNDworkshop! As you're typing the password, the cursor won't move- that's fine, then hit enter.
    c. If login is successful the terminal will give a message:

```
last login: Tue Feb 19 15:26:49 2019 from 172.27.166.26
-bash: goto: command not found
```

Success! You're in!

Set up workspace

First we need to create some folders to keep ourselves organized:

```
mkdir FastQC Alignments Trimmed_fastq Assembly Counts
```

Next, we need to link some reference files that we'll be using later:

```
ln -s /local_storage/annotation_db/Mus_musculus/UCSC/mm10/Sequence/newHisat/*.* .
ln -s /local_storage/annotation_db/Mus_musculus/UCSC/mm10/Annotation/Genes/genes.gtf .
ln -s /local_storage/annotation_db/Mus_musculus/UCSC/mm10/Sequence/WholeGenomeFasta/*.fa
```

Today, we'll be using paired end fastq files that have been down sampled to 5 million reads. For a typical RNA-Seq experiment you would need approximately 40 million reads per sample.

Sequence Quality Assessment.

While it's possible to look at the first 10 lines or so of a fastq file using the head command, it's impossible to make sense of all the information in a fastq file just by looking at it. To extract and summarize some of the information found in the fastq file, we will be using the program FastQC:
http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

All of the options for the fastqc program can be found by typing:

```
fastqc -h
```

To run FastQC:

```
fastqc -o FastQC /home/workshop.2019/Files/example_1.fastq.gz
```

This command specifies two things:

- an output directory "FastQC", using to `-o` option
- a fastq file to be processed

Note that I didn't use the `-f` option to specify the file format. This is because the default value is fastq. It's important to be aware of the default values set for each program. Most of the time, the default values are reasonable, but sometimes they're not. FastQC produces a html report that can opened in a browser and a zipped folder with all of the raw data used to generate the report. The html report is the file you want to look at, but first it needs to be transferred to your local computer.

-

For windows users, use winscp to download the fastqc file to your local computer. For mac users, you can use the terminal to download the file. Open another terminal window (command + N). Then the secure copy command:

```
scp
workshop.2019@assigned_server.med.und.edu:/home/workshop.2019/Your_Folde
r/fastqc/* .
```

This will download the FastQC directory into your home directory.

Adapter Trimming

There are numerous programs for trimming adapters from sequencing data. We will be using trim_galore, which is a wrapper around the program cutadapt.

TrimGalore: https://www.bioinformatics.babraham.ac.uk/projects/trim_galore
Cutadapt: https://cutadapt.readthedocs.io/en/stable/guide.html

Again,we will look at the options available for trim galore and construct our command. There are a lot of options for trim galore, and we won't need to use them all, here are the ones we need use:

- Our data is Illumina 1.9+, use `--phred33` (its already default)
- We want to run FastQC after we're done : `--fastqc`

- This dataset was generated from illumine, so we'll use the `--illumina` flag to tell trimGalore to trim illumina adapters.
- Our data is paired-end, so we need the the `--paired` option
- Set the amount of overlap between sequence and adapter , at least 3: `--stringency 3`
- Specify our output directory `-o Trimmed_fastq`
- Trim low quality bases : `-q 20` (already default)

Let's put together our command:

```
trim_galore --paired --fastqc --illumina --stringency 3 -o Trimmed_fastq
/home/workshop.2019/Files/example_1.fastq.gz /home/workshop.2019/Files/example_2.fastq.gz
```

TrimGalore will produce a number of files:

- A trimming report for each input file with the command line parameters used and how many reads had adapters.
- Trimmed fastq files
- FastQC report for each trimmed file.

For paired end data like we have, TrimGalore will also produced two "validated" fastq files, which means that reads in the R1 and R2 files are the same order, which is important for downstream applications like alignment.

Alignment https://ccb.jhu.edu/software/hisat2/index.shtml
It's finally time to align our reads to the genome! We will be using hisat2 today. Hisat2 is a part of the newer 'Tuxedo Suite' developed by the Center for Computational Biology at John's Hopkins University This suite is an entire pipeline of tools for analyzing RNAseq data.

We will use one unique file type for alignment, the hisat2 index. You can think of the index as a compressed version of the genome. For more information on the computational details on these files, consult the hisat2 website https://ccb.jhu.edu/software/hisat2/manual.shtml#the-hisat2-build-indexer

The basic structure of the alignment command for paired end reads is:

```
hisat2 <options> -x <index> -1 <read1.fq> -2 <read2.fq>
```

I encourage you to look at all the options with: `hisat2 -h`
Here are some important options to be aware of:

- `--dta` This will add some information that will be used in the transcript assembly
- `-x` basename of the hisat2 index for the reference genome.
- `--rna-strandness` Specify if your library is strand-specific.

```
hisat2 --dta -x genome -1 /home/workshop.2019/Files/example_1_val_1.fq.gz -2
/home/workshop.2019/Files/example_2_val_2.fq.gz | samtools view -b - >
Alignments/example.bam
```

I've also used the pipe character (|), which allows us to string two commands together. Here, we're adding the `samtools view` after the alignment command to convert the SAM file into a more compact BAM file.

Kill the alignment process with:

```
Ctrl + C
```

Hisat2 will produce an alignment file, and some alignment statistics. Copy the files into your folder, and have a look!

```
cp /home/workshop.2019/Files/hisat_out.txt .
cp  /home/workshop.2019/Files/example.bam .
```

Inspecting Alignment files

Samtools http://samtools.sourceforge.net/ is an extremely useful collection of utilities for manipulating alignments in the SAM and BAM format including sorting, merging, and indexing. Today, we're going to use it to look at the alignment file produced by hisat2, and also to sort and index the alignment file.

The syntax for samtools commands has one extra piece. You not only have to call the tool, you will also have to tell it what utility you would like to use. We are going to be using the view, sort, and index utilities today:

```
samtools <utility> -options <file>
```

Samtools allows us to easily work with and inspect alignment files in BAM format. Take a look at the help page for the view utility (`samtools view –h`). There are many different things you can do with this utility, but the easiest way to think about it is just like a fancy `cat`. You are able to read binary input or output binary files from text.

Let's take a look at a couple of parts of the alignment file. The following command will print the first 25 lines of the alignment file:

```
samtools view example.bam | head -n 25
```

Finally, sort and index your bam file:

```
samtools sort -o Alignments/example_sorted.bam example.bam

samtools index Alignments/example_sorted.bam
```

There are two ways of sorting BAM files: by read coordinate, which is what we've done here, or by sequence name. Indexing your BAM files will allow certain programs to read your BAM file quicker. After you've sorted your bam file, you can remove the unsorted bam- they contain the same information only in a different order.

Section 3 – Assembly and Counting

This section of the exercise will finally put all of this data into a format that is digestible by a human. We will be summing up the hits with a couple different programs and generating data tables that can be mined for real results.

Counting with featureCounts http://bioinf.wehi.edu.au/featureCounts
We will be using the program featureCounts to count reads aligning each gene. How does featureCounts do this? First, print the first few lines of your gtf file.

```
head genes.gtf
```

featureCounts is going to take every line labeled exon in the third column of this gtf file (called the feature) and count the number of reads aligning within the interval defined by the 4th and 5th columns. It is then going to match the total of all the hits within those features to the gene_id (called the metafeature) pair listed in the 9th column. This will output a text file with a total number of hits matching to each gene.

The 'exon' feature and 'gene_id' attribute can be changed depending on the type of analysis. For instance, you might want to look at isoforms using the 'transcript_id' attribute.

Here's the command we'll be using:

```
featureCounts -p -a genes.gtf -o counts.txt \
Alignments/example_sorted.bam 2> featureCounts_stderr.txt
```

Important options:

- `-p` this will count each fragment, instead of counting counting both ends of the fragment separately
- `-s` strandness the default setting is 0, for unstranded data, which is what we'll be using today, but this option needs to be changed depending on the library prep kit used to generate your libraries
- `-a` the annotation file (genes.gtf)
- `-M` this option will count multimapping reads, the default is to not count them.
- `-T` This options species the number of threads to use.

This program should only take a few minutes to run. When it's finished, take a few minutes to look through the results. Can you use the command `grep` to identify your favorite gene? How about a housekeeping gene like ACTB?

Counting and Assembly with Cufflinks http://cole-trapnell-lab.github.io/cufflinks/
Cufflinks s a program that assembles a transcriptome from RNAseq data and quantifies their expression. Again, we'll look at the options for cufflinks using:

```
cufflinks -h
```

There a quite few options for cufflinks. Here are some that are important:

- `-g` to supply cufflinks with annotation
- `-u` tells cufflinks to "rescue" multimapping reads
- `--library-type`

```
cufflinks -o example \
          -g genes.gtf \
          -b genome.fa \
          -u \
          --library-type fr-unstranded \
          Alignments/example_sorted.bam 2> cufflinks_stderr.txt
```

After running the command, kill the process with:

```
Ctrl + C
```

Copy the output files in to your home directory:

```
cp -r /home/workshop.2019/Files/cufflinks_results/ .
```

Cufflinks output consists of two types of files, count files and an annotation file. The count files are labeled fpkm_tracking. There are two types, genes and isoforms. They are made using different feature ids pulled from the gtf, gene_id and transcript_id, respectively. These files are primarily for reporting the FPKM values of each gene or transcript to be used in differential analysis. Although there are several fields containing identification information, (http://www.broadinstitute.org/cancer/software/genepattern/file-formats-guide#FPKM_tracking) the column labeled FPKM reports the actual counted value. The FPKM_conf_lo and FPKM_conf_hi columns report the upper and lower bounds of the 95% confidence interval of the measurement.

The transcripts.gtf file contains all of the information in the counting files collected in gtf format. This represents a full annotation, complete with the quantitative fpkm information, of the transcriptome contained in the analyzed sample. Take a look at this file and the genes.gtf annotation file in the directory above. What's different?