BIOL*3300 Lab9 F21

Reference-based RNA-seq data analysis

Introduction to RNA-seq data

In recent years, RNA sequencing (in short RNA-Seq) has become a very widely used technology to analyze the continuously changing cellular transcriptome, i.e. the set of all RNA molecules in one cell or a population of cells. One of the most common aims of RNA-Seq is the profiling of gene expression by identifying genes or molecular pathways that are differentially expressed (DE) between two or more biological conditions.

The dataset we are using for this lab is part of a larger study described in Kenny PJ et al., Cell Rep 2014. The authors are investigating interactions between various genes involved in Fragile X syndrome, a disease of aberrant protein production, which results in cognitive impairment and autistic-like features. **The authors sought to show that RNA helicase MOV10 regulates the translation of RNAs involved in Fragile X syndrome.**

From this study we are using the RNA-seq data which is publicly available in the Sequence Read Archive (SRA). For this lab we will be looking at a small subset on chr1 (~300,000 reads/sample) and for each group we have three replicates as described in the figure below.

Step1 Planning and obtaining data

We will start by creating a directory that we can use for the rest of the lab as we did for SNP analysis. Within your scratch/Biol3300 directory, create the RNAseq_lab directory, so it has 5 subdirectories within it. The purpose of these directories will become clear.
The structure of RNAseq_lab directory:

```
RNAseq_lab/
        genome/
        genome_index/
        raw_data/
        results/
```

I have created this SNP_lab directory and you can copy that to your scratch/Biol3300 folder directly with:

```
      cp –r /scratch/hchang02/Biol3300/RNAseq_lab/ ~/scratch/Biol3300
```

Step2 Quality control using FASTQC

The first step in the RNA-Seq workflow is to take the FASTQ files received from the sequencing facility and assess the quality of the sequence reads. As we introduced what information is stored in a FASTQ file, the next step is to examine quality metrics for our data.

FastQC provides a simple way to do some quality checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses, which you can use to obtain an impression of whether your data has any problems that you should be aware of before moving on to the next analysis.

Run FastQC

Change directories to raw_data and let's create a directory to store the output of FastQC:

```
    cd ~/scratch/Biol3300/RNAseq_lab/raw_data
    mkdir ../../RNAseq_lab/results/fastqc
```

FastQC will accept multiple file names as input, so we can use the *.fq wildcard.

```
    module load fastqc
    fastqc –o ../../RNAseq_lab/results/fastqc *.fq
```

Step3 Read alignment with STAR

The alignment process consists of choosing an appropriate reference genome to map our reads against, and performing the read alignment using one of several splice-aware alignment tools such as STAR or HISAT2 (HISAT2 is a successor to both HISAT and TopHat2). The

choice of aligner is a personal preference and also dependent on the computational resources that are available to you.

For this lab we will be using STAR (Spliced Transcripts Alignment to a Reference), an aligner designed to specifically address many of the challenges of RNAseq read mapping.

Aligning reads using STAR is a two-step process:

1. Create a genome index
2. Map reads to the genome

Now let's get started by loading up some of the modules for tools we need for this section to perform alignment and assess the alignment:

```
module load star
```

Creating a genome index

For this lab we are using reads that originate from a small subsection of chromosome 1 (~300,000 reads) and so we are using only chr1 as the reference genome.

For this lab, **I have already indexed the reference genome for you as this can take a while**. The code was provided below that you would use to index the genome for your future reference, but **please do not run the code below as you have copied the genome_index directory**. For indexing the reference genome, a reference genome (FASTA) is required and an annotation file (GTF or GFF3) is suggested a more accurate alignment of the reads.

The basic options to generate genome indices using STAR are as follows:

--runThreadN: number of threads
--runMode: genomeGenerate mode
--genomeDir: /path/to/store/genome_indices
--genomeFastaFiles: /path/to/FASTA_file (reference genome)
--sjdbGTFfile: /path/to/GTF_file (gene annotation)
--sjdbOverhang: readlength -1

And the command to run indexing is:

```
** DO NOT RUN **
    STAR --runThreadN 6 \
    --runMode genomeGenerate \
    --genomeDir ./genome_index/ \
    --sjdbGTFfile ./genome/chr1-hg19_genes.gtf \
    --genomeFastaFiles ./genome/chr1.fa \
    --sjdbOverhang 99
```

This will give you the following output:

```
   STAR --runThreadN 6 --runMode genomeGenerate --genomeDir ./genome_index/ --sjdbGTFfile ./genome/chr1-hg
        STAR version: 2.7.9a   compiled: 2021-07-08T15:06:22+00:00 build-node.computecanada.ca:/tmp/ebuser
Nov 15 18:53:05 ..... started STAR run
Nov 15 18:53:05 ... starting to generate Genome files
Nov 15 18:53:11 ..... processing annotations GTF
!!!!! WARNING: --genomeSAindexNbases 14 is too large for the genome size=249250621, which may cause seg-fa
Nov 15 18:53:13 ... starting to sort Suffix Array. This may take a long time...
Nov 15 18:53:15 ... sorting Suffix Array chunks and saving them to disk...
Nov 15 18:54:18 ... loading chunks from disk, packing SA...
Nov 15 18:54:24 ... finished generating suffix array
Nov 15 18:54:24 ... generating Suffix Array index
Nov 15 18:55:15 ... completed Suffix Array index
Nov 15 18:55:15 ..... inserting junctions into the genome indices
Nov 15 18:55:36 ... writing Genome to disk ...
Nov 15 18:55:37 ... writing Suffix Array to disk ...
Nov 15 18:55:42 ... writing SAindex to disk
Nov 15 18:55:43 ..... finished successfully
```

Mapping reads

The basic options for mapping reads to the genome using STAR are as follows:

--runThreadN: number of threads
--readFilesIn: /path/to/FASTQ_file
--genomeDir: /path/to/genome_indices
--outFileNamePrefix: prefix for all output files

We will also be using some advanced options:

--outSAMtype: output filetype (SAM default)
--outSAMUnmapped: what to do with unmapped reads
--outSAMattributes: SAM attributes

More details on STAR and its functionality can be found in the user manual, we encourage you to peruse through to get familiar with all available options.

Now let's put it all together! The full STAR alignment command is provided below. And we will use the first replicate in the Mov10 overexpression group, Mov10_oe_1_subset.fq.

```
    STAR --runThreadN 6 \
    --genomeDir genome_index/ \
    --readFilesIn raw_data/Mov10_oe_1.subset.fq \
    --outFileNamePrefix results/Mov10_oe_1_ \
    --outSAMtype BAM SortedByCoordinate \
    --outSAMunmapped Within \
    --outSAMattributes Standard
```

This will give you the following output:

```
        STAR --runThreadN 6 --genomeDir genome_index/ --readFilesIn raw_data/Mov10_oe_1.subset.fq --outFi
        STAR version: 2.7.9a   compiled: 2021-07-08T15:06:22+00:00 build-node.computecanada.ca:/tmp/ebuse
Nov 15 19:13:02 ..... started STAR run
Nov 15 19:13:02 ..... loading genome
Nov 15 19:13:14 ..... started mapping
Nov 15 19:13:41 ..... finished mapping
Nov 15 19:13:41 ..... started sorting BAM
Nov 15 19:13:54 ..... finished successfully
```

Alignment Outputs (SAM/BAM)

The output we requested from STAR is a BAM file, and by default returns a file in SAM format. Let's take a quick look at our alignment. To do so we first convert our BAM file into SAM format using samtools and then pipe it to the less command. This allows us to look at the contents without having to write it to file.

```
    module load samtools
    samtools view -h results/Mov10_oe_1_Aligned.sortedByCoord.out.bam | less
```

We can also checj some simple stats for alignment file:

```
    samtools flagstats results/Mov10_oe_1_Aligned.sortedByCoord.out.bam
```

This will give you the following output:

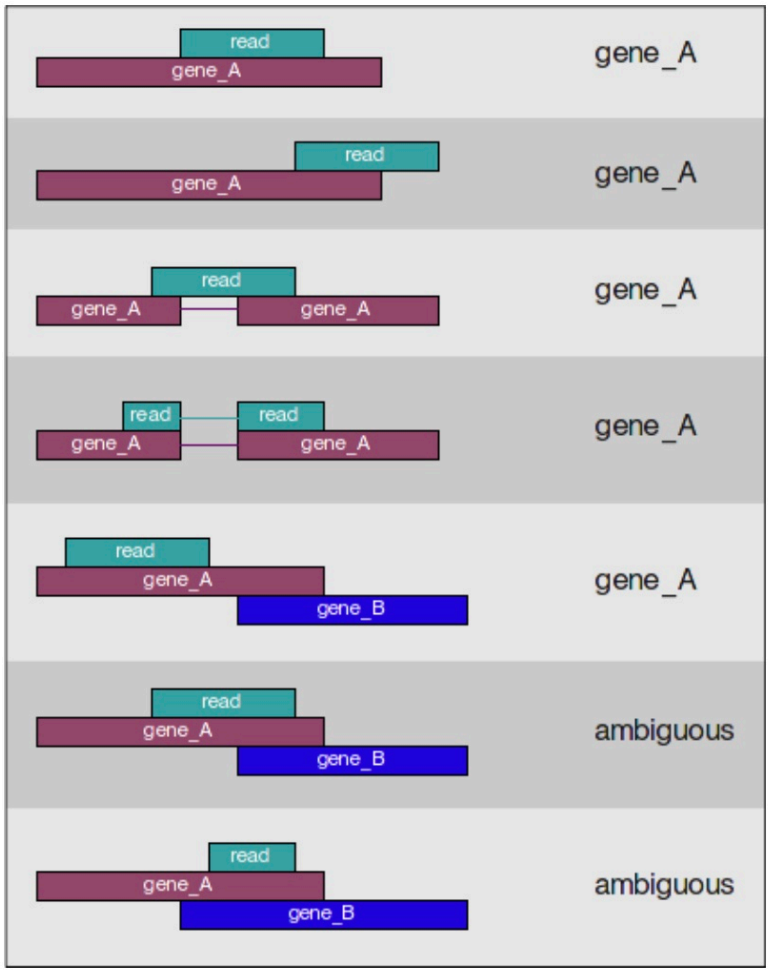
```
314463 + 0 in total (QC-passed reads + QC-failed reads)
8563 + 0 secondary
0 + 0 supplementary
0 + 0 duplicates
290665 + 0 mapped (92.43% : N/A)
0 + 0 paired in sequencing
0 + 0 read1
0 + 0 read2
0 + 0 properly paired (N/A : N/A)
0 + 0 with itself and mate mapped
0 + 0 singletons (N/A : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

Step4 Counting reads

Once we have our reads aligned to the genome, the next step is to count how many reads have been mapped to each gene. The input files required for counting include the BAM file and an associated gene annotation file in GTF format. Today, we will be using featureCounts program from subread software to get the gene counts. We picked this tool because it is accurate, fast and is relatively easy to use.

featureCounts works by taking the alignment coordinates for each read and cross-referencing that to the coordinates for features described in the GTF. Most commonly a feature is considered to be a gene, which is the union of all exons (which is also a feature type) that make up that gene. Please note that this tool is best used for counting reads associated with genes, and not for splice isoforms or transcripts.

featureCounts only includes and counts those reads that map to a single location (uniquely mapping) and follows the scheme in the figure below for assigning reads to a gene/exon.



To use the featureCounts, we need to load the subread software:

```
module load subread
```

```
featureCounts -s 2 \
-a ./genome/chr1-hg19_genes.gtf \
-o ./results/Mov10_featurecounts.txt \
./results/Mov10_oe_1_Aligned.sortedByCoord.out.bam
```

This will give you the following output:

```
//================================ Running =================================\\
||                                                                         ||
|| Load annotation file chr1-hg19_genes.gtf ...                            ||
||    Features : 37213                                                     ||
||    Meta-features : 2330                                                 ||
||    Chromosomes/contigs : 1                                              ||
||                                                                         ||
|| Process BAM file Mov10_oe_1_Aligned.sortedByCoord.out.bam...            ||
||    Strand specific : reversely stranded                                 ||
||    Single-end reads are included.                                       ||
||    Total alignments : 314463                                            ||
||    Successfully assigned alignments : 224925 (71.5%)                    ||
||    Running time : 0.01 minutes                                          ||
||                                                                         ||
|| Write the final count table.                                            ||
|| Write the read assignment summary.                                      ||
||                                                                         ||
|| Summary of counting results can be found in file "./results/Mov10_feature  ||
|| counts.txt.summary"                                                     ||
||                                                                         ||
\\=========================================================================//
```

FeatureCounts output

The output of this tool is 2 files, a count matrix and a summary file that tabulates how many the reads were "assigned" or counted and the reason they remained "unassigned". Let's take a look at the summary file:

```
    less results/Mov10_featurecounts.txt.summary
```

Now let's look at the count matrix:

```
    less results/Mov10_featurecounts.txt
```

Since the featureCounts output has additional columns with information about genomic coordinates, gene length etc., we can use the cut command to select only those columns that you are interested in.

```
    cut -f1,7,8,9,10,11,12 results/Mov10_featurecounts.txt > results/Mov10_featurecounts.Rmatrix.txt
    less results/Mov10_featurecounts.Rmatrix.txt
```

Assignment for Lab 9

Please follow the tutorial to perform the RNA-seq analysis with Irrel_kd_1.subset.fq and upload your clear screen short of featureCounts output of this sample with your terminal username.

Reference

These lab materials are from the following papers and website:

1. Rochette, N.C. and Catchen, J.M., 2017. Deriving genotypes from RAD-seq short-read data using Stacks. Nature Protocols, 12(12), pp.2640-2659.
2. The population structure and recent colonization history of Oregon threespine stickleback determined using restriction-site associated DNA-sequencing. Mol. Ecol. 22, 2864–2883 (2013).
3. https://datacarpentry.org/wrangling-genomics/04-variant_calling/index.html