RNA-seq Data Analysis

Guillermo Parada < *gp7@sanger.ac.uk* >
Ashley Sawle < *Ashley.Sawle@cruk.cam.ac.uk* >
(Designed by Luigi Grassi < *lg490@medschl.cam.ac.uk* >)

Introduction to RNA-seq data analysis
September, 2018

General information

The following standard icons are used in the hands-on exercises to help you locating:

- Important Information
- General information / notes
- Follow the following steps
- Questions to be answered
- Warning – Please take care and read carefully
- Optional Bonus exercise
- Optional Bonus exercise for a champion

Resources used

FastQC

HISAT2

stringtie

gffcompare

samtools

IGV genome browser

STAR RNA-seq aligner

Salmon

RNA-seq Practical

Using the fastq reads, publicly released with the article (https://doi.org/10.1093/nar/gkt1359) we are going to reconstruct the transcriptome of one zebrafish sample. We will then quantify the expression level of genes and transcripts and finally identify genes and transcripts present in our sample and non reported in the reference transcriptome.

Prepare the environment

Open the Terminal.

First, go to the folder, where the data are stored.

```
cd ~/Desktop/RNA-seq
```

Note that all commands in this tutorial are supposed to be run within the main folder `RNA-seq` so when you run them as indicated in the handout please be sure you are in the `RNA-seq` folder.

Understand the quality encoding of your data

To check the quality of our sequenced reads we are going to use the FastQC tool (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/). The final result of FastQC is a report that we are going to inspect.

A FASTQ file reports four lines per sequenced read. The fourth line is made of ASCII 33-126 symbols, representing the quality of the sequence reported in the second line. The range of the quality depends by the technology and by the chemistry of the sequencing. (https://en.wikipedia.org/wiki/FASTQ_format)

Question

1. Can you tell which quality encoding have the reads in the SRR1048063_1.fastq.gz file?

   ____

Hint: Look at the first read of the file SRR1048063_1.fastq.gz by typing:

```
gzip -cd data/SRR1048063_1.fastq.gz | head -n 4
```

And compare the quality strings (last line) with the table found at http://en.wikipedia.org/wiki/FASTQ_format#Encoding

Running FastQC

FastQC is installed on your computer, check the possible options you have to run it by typing:

```
fastqc --help
```

Which version of the program is currently installed on your system?

Create a directory where store the FASTQC results.

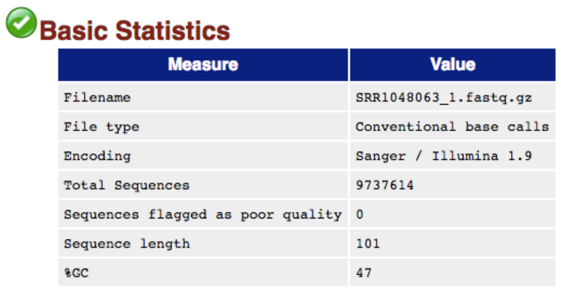
```
mkdir -p fastqc
```

Then run FastQC on each of the FASTQ files contained in the directory data (SRR1048063_1.fastq.gz and SRR1048063_2.fastq.gz) and save the relative results in the fastqc directory.

Quality visualisation

FastQC will generate a QC report, containing several items.

For example, the report file will have a **Basic Statistics** table and various graphs and tables for different quality statistics.



Basic Statistics

Measure	Value
Filename	SRR1048063_1.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	9737614
Sequences flagged as poor quality	0
Sequence length	101
%GC	47

Figure 1: FastQC Basic Statistics

In addition, FastQC reports information about the quality scores of the reads.

Q-scores

A quality score (or Q-score) expresses an error probability. In particular, it is a convenient and compact way to communicate very small error probabilities. Given an assertion, A, the probability that A is not true, $P(\neg A)$, is expressed by a quality score, $Q(A)$, according to the relationship:

$$Q(A) = -10\log_{10}(P(\neg A)) \tag{1}$$

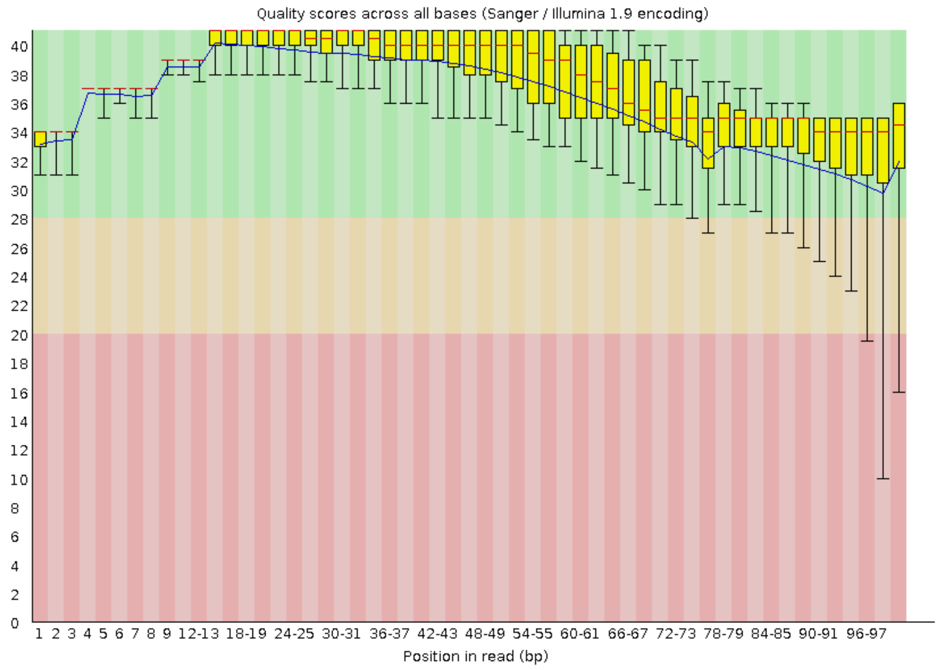


Figure 2: Per base sequence quality plot: visual output from FastQC. Base positions in the reads are shown on x-axis and quality score (Q Score) are shown on the Y-axis.

where $P(\neg A)$ is the estimated probability of an assertion A being wrong. The relationship between the quality score and error probability is reported in the following table:

Quality score, Q(A)	Error probability, $P(\neg A)$
10	0.1
20	0.01
30	0.001
40	0.0001

Questions

1. How many sequences were there in the `SRR1048063_1` file? What is the read length? ______
2. Does the quality score value vary throughout the read length? ______
3. What is the quality score range you see? ______
4. Why does the quality deteriorate at the end of the read? ______
5. Sequencing errors can complicate the downstream analysis. Sequence reads containing errors may lead to ambiguous paths in the assembly or improper gaps. Based on your evaluation of the FastQC report do you think is it worth to filter or trim some of the reads in our dataset? ______

Index preparation

There are numerous tools performing short read alignment and the choice of aligner should be carefully made according to the analysis goals/requirements. Here we will use *HISAT2*, a fast aligner with low memory requirements that performs spliced alignments. It is the program suggested for the alignment in the *new Tuxedo* protocol (https://doi.org/10.1038/nprot.2016.095) and it requires an indexed genome to keep its memory footprint small and the running time short. The program creates a genome index simply by using the FASTA file of the sequence we want to use as reference. A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol at the beginning. It is recommended, but not mandatory, that all lines of text be shorter than 80 characters in length.

Due to time constraints we will build the index only for one chromosome of the zebrafish genome. For this we need the chromosome sequence in fasta format. This can be downloaded from one of the reference databases (ENSEMBL, NCBI, UCSC).

We are going to download it from ENSEMBL

```
cd ~/Desktop/RNA-seq/genome

wget ftp://ftp.ensembl.org/pub/release-90/\
fasta/\danio_rerio/dna/\
Danio_rerio.GRCz10.dna.chromosome.4.fa.gz

gunzip Danio_rerio.GRCz10.dna.chromosome.4.fa.gz

cd ..

less -S genome/Danio_rerio.GRCz10.dna.chromosome.4.fa

mkdir -p genome/GRCz10_chr4

hisat2-build genome/Danio_rerio.GRCz10.dna.\
chromosome.4.fa genome/GRCz10_chr4/GRCz10chr4

ls -ltrh genome/GRCz10_chr4
```

Questions

In the directory genome is also present a subdirectory GRCz10 containing the index of all the zebrafish genome. Which are the similarities and differences you notice by comparing the content of the two index directories (the chr4 you generated and the complete genome one)?

REFERENCE TRANSCRIPTOME

In the analysis we are going to do we need a reference transcriptome. It can notably improve the alignment and the transcriptome assembly results. It is generally saved as a gtf file, a tab separated text file made at least of nine columns (for more information see https://www.ensembl.org/info/website/upload/gff.html. We are using the ENSEMBL release 90 based on the genome version GRCz10. You can inspect the relative gtf in the annotation directory.

```
less -S annotation/Danio_rerio.GRCz10.90.chr.gtf
```

Finally we want to prepare the splice sites from the reference transcriptome and save this information for the alignment we are going to do with *HISAT2*.

```
hisat2_extract_splice_sites.py \
annotation/Danio_rerio.GRCz10.90.chr.gtf >\
 annotation/ens90z10_splicesites.txt
```

Alignment

There are numerous tools performing short read alignment and the choice of the aligner typically depends by the analysis goals and requirements. We will use *HISAT2*, the one indicated in the *Tuxedo2* pipeline. The index files of *Danio rerio* generated from the genome version GRCz10 are in the directory genome/GRCz10. All of them have a file name starting with the Drerioz10 prefix.

There are several parameters we might want to specify in order to align our reads with *HISAT2*. To view them all type

```
hisat2 --help
```

The general hisat2 command is:

```
hisat2 [options]* -x <hisat2-idx>
{-1 <m1> -2 <m2> | -U <r> [-S <hit>]
```

Now we will proceed with the alignment of the paired-end read files from the sample SRR1048063.

Questions

The fastq files we are going to align are in the data directory. You can verify it by listing the directory content:

```
ls data
```

The reference splice junction file you generated by using the ensembl V90 zebrafish transcriptome is in the annotation directory and its file name is ens90z10_splicesites.txt.

1. Create a directory, named hisat2, where you will save the alignment results
2. Run *HISAT2* by using the options listed below.

```
--phred33
    It specifies that qualities of the fastq files
    are encoded as Phred quality plus 33.

--known-splicesite-infile SPLICE_SITE_FILE
   SPLICE_SITE_FILE is the reference splice
   junction file you have generated in the
   previous session.

--downstream-transcriptome-assembly
    To report alignments tailored for transcript
    assemblers, StringTie in our case.

-p 8
    To execute the program launching
    8 parallel search threads.

-x INDEX_BASENAME
    INDEX_BASENAME is the base-name of the index
    for the reference genome you are going to use.

--summary-file SUMMARY_FILE
    SUMMARY_FILE is the file were you are going
    to save the alignment summary file.

-1 m1 -2 m2
    m1 and m2 arguments are the fastq
    files of the paired-end reads.

 -S SAM_FILE
    SAM_FILE is the file were you will
    save the alignment results,
    in your case hisat2/SRR1048063.sam
```

Each time you are specifying a file please remember to include also the directory where the file is located (E.g. use annotation/ens90z10_splicesites.txt to refer to the ens90z10_splicesites.txt if you run hisat2 from the directory ~/Desktop/RNA-seq).

If you have any problem in launching the command you might have a look at the *HISAT2* help

```
hisat2 --help
```

Exploring alignment results

The alignment will take few minutes. In the mean time let's describe the file format of main output result. *HISAT2* reports the alignments in a Sequence Alignment Map (SAM) file. It is a text file designed to store biological sequences aligned to a reference sequence. It is made of two parts: a header section, with all the lines preceded by a @ symbol, and an alignment section, made of 11 mandatory tab separated fields and extra fields, depending by the alignment program. Have a look at the format specification manual for a more detailed description of the file format (https://samtools.github.io/hts-specs/SAMv1.pdf).

Questions

Once the alignment is completed have a look at the files created in the output folder.

1. How many files have been created by *HISAT2*?
2. Can you visualise them by using the *less* command? How do they look like?
3. How many extra fields are contained in the SAM file generated by *HISAT2*?
4. Looking at the summary file do you think the alignment step has been correctly made?
5. How many unaligned reads there are? Why do you have them?

The SAM format is human readable, sou you can visually inspect it or extract its information with a script. The BAM format provides the same information of the SAM file in a binary compressed form. For this reason BAM files use less storage space than SAM counterpart and are faster to manipulate. BAM files can be sorted by coordinates, or by read name and the majority of downstream programs only take sorted BAM files as input.

SAMTools is made of various tools designed for manipulating alignments in the SAM/BAM format. We are going to use it to convert the SAM file of our alignment into a sorted (by coordinates) BAM.

```
samtools view -u hisat2/SRR1048063.sam | samtools \
sort -@ 7 - -o hisat2/SRR1048063.sorted.bam
```

Indexing a BAM file aims to achieve fast retrieval of alignments overlapping a specified region without going through the whole alignments. It is a mandatory step to to easy access and visualise the alignment in genome browser programs.

We are going to use *SAMTools* for this purpose.

```
samtools index hisat2/SRR1048063.sorted.bam
```

SAMTools is a versatile tool. You can use it to convert a BAM file to a SAM file, to filter out unmapped reads, to extract entries mapping to a specific region and for many other purposes. For a more detailed description have a look at the manual http://www.htslib.org/doc/samtools.html.

Questions

1. Describe differences between the produced SAM and BAM files.
2. Which is the index file and where is located?

Integrative Genomics Viewer (IGV) is a visualisation tool for High Throughput Sequencing (HTS) data. We are going to use it to inspect our alignment results. In this hands-on exercise we will explore some of its simplest functionalities, if you want to deepen it please have a look at this tutorial completely dedicated to IGV https://github.com/griffithlab/rnaseq_tutorial/wiki/IGV-Tutorial.

In order to launch *IGV* type on the terminal

```
igv &
```

As first thing you have to load the genome of interest.

On the top left of your screen choose from the drop down menu *Zebrafish (GRCz10/danRer10)*. Then in order to load the desire files go to:

File –> Load from File

On the pop up window navigate to **home –> participant –> Desktop –> RNA-seq –> hisat2** folder and select the file `SRR1048063.sorted.bam`. Once the file is loaded you might want to rename the track. In order to do it right-click on the name of the track on the left and choose Rename Track.

Finally following the same process load the Ensembl annotation `Danio_rerio.GRCz10.90.chr.bed` present in the annotation directory.

On the top middle box you can specify the region you want your browser to zoom. Type `chr12:19,064,687-19,071,805`.

Right-click on the name of the Ensembl track and choose **Expanded**.

Questions

1. Can you identify the splice junctions from the BAM file? ______________________
   ____

2. Are the junctions annotated for CBY1 consistent with the annotation? ____________

   ____

3. Are all annotated transcripts (both from RefSeq and Ensembl) expressed? ________

   ____

   ____

Transcriptome assembly

There are several tools designed for the transcriptome reconstruction, for this hands-on session we are going to use *StringTie*, as indicated in the *Tuxedo2* pipeline. It uses a network flow algorithm and an optional de novo assembly step to assemble full-length transcripts derived by the multiple splice variants of each expressed gene locus. It also quantifies the isoform expression.

The general stringtie command is:

```
stringtie <aligned_reads.bam> [options]*
```

The program takes as input a BAM file reporting RNA-Seq read mappings sorted by their genomic location.

StringTie has a number of options you might want to specify to perform transcriptome assembly. To view them all type

```
stringtie --help
```

Questions

1. Run *StringTie* by using the options listed below.

```
-p 8
   To execute the program launching
   8 parallel search threads.

 -o GTF_FILE
   Replace GTF_FILE with the file were
   you will save the transcriptome
   reconstruction results,
   in your case stringtie/SRR1048063.gtf
 -G ANN_REF_FILE
   Replace ANN_REF_FILE with the reference
   ensembl gtf file contained in the
   annotation directory
```

Each time you are specifying a file please remember to include also the directory where the file is located (E.g. use annotation/Danio_rerio.GRCz10.90.chr.gtf to refer to the ensembl gtf if you run stringtie from the directory ~/Desktop/RNA-seq).

Exploring gtf results

At the completion of transcriptome assembly visualise the created gtf file by using the *less* command.

Question

You might have noticed that in the stringtie directory there are also other assembled transcriptomes. They are the stringtie results of the other samples from the article https://doi.org/10.1093/nar/gkt1359. How many transcripts have been assembled in each sample and how many are reported in the transcriptome you used as reference?

Merging of all the reconstructed transcriptomes

You have already noticed that in the stringtie directory there are the 6 assembled transcriptomes. One was reconstructed by you in the previous session and the other 5 are the stringtie results of the other 5 samples studied the article. You can verify it by listing the directory content:

```
ls stringtie
```

We need to merge all them together with the reference one in one merged transcriptome. This extended reference will consist in a unified non-redundant set of transcripts (isoforms) across multiple RNA-Seq samples. It will be used for the requantification step and for the differential expression analysis. *StringTie* has a "transcript merge" mode, distinct from the assembly usage mode used in the previous session. To know more about it have a look at the stringtie manual

https://ccb.jhu.edu/software/stringtie/index.shtml?t=manual

or visualise the program help.

```
stringtie --help
```

Questions

1. Run *StringTie* in the "transcript merge" mode by using the options listed below and providing the file stringtie/assemblylist.txt with the list of transcriptomes to assembly.

```
-p 8
    To execute the program launching
    8 parallel search threads.

 --merge
    To specify to run stringtie in the
    merge mode

 -o GTF_FILE
    Replace GTF_FILE with the file were
    you will save the transcriptome
    reconstruction results,
    in your case stringtie/merged.gtf

 -G ANN_REF_FILE
    Replace ANN_REF_FILE with the reference
    ensembl gtf file contained in the
    annotation directory
```

2. How many transcripts have been included in the merged transcriptome ?

Each time you are specifying a file please remember to include also the directory where the file is located (E.g. use annotation/Danio_rerio.GRCz10.90.chr.gtf to refer to the ensembl gtf if you run stringtie from the directory ~/Desktop/RNA-seq).

Comparing the transcriptomes

Once we have merged the transcriptomes we want to compare the final transcripts with the ones in reference transcriptome. The program gffcompare can be used to compare, merge, annotate and estimate accuracy of one or more GTF/GFF3 files (the "query" files), when compared with a reference annotation (also provided as GTF/GFF3).

You can visualise all the options you have by using the program help:

```
gffcompare --help
```

Questions

1. Create a directory named gffcompare
2. Run gff compare with the options below listed and using as input the merged transcriptome

```
-r ANN_REF_FILE
   Replace ANN_REF_FILE with the reference
   ensembl gtf file contained in the
   annotation directory

 -s GENOME_FASTA
   Replace GENOME_FASTA with the fasta
   of your reference genome, the one you
   used to generate the genome index
   (genome/Danio_rerio.GRCz10.dna.toplevel.fa)

 -o PREFIX_FILE
   Replace PREFIX_FILE with the prefix
   you want to use for the files generated
   by gffcompare. Please include also
   the directory (gffcompare/PREFIX_FILE).
```

Inspecting the results

Gffcompare produces different output files, among them visualise the ".tracking" file. Each row of this file contains a transcript structure that is present in one or more input GTF files. The 4th column contains a "class code" value with the relationship between a transfrag and the closest reference transcript (where applicable). For a detailed description of the different class codes have a look at the online documentation: http://ccb.jhu.edu/software/stringtie/gffcompare.shtml#transfrag-class-codes.

1. Which is the occurrence of each class code in your transcriptome? (There are are multiple ways to extract this information, in the command-line you could use cut together with sort and uniq)

Requantifying transcript expression

We finally have to requantify the expression of the transcripts from the merged transcriptome. In order to do this we are going to use again stringtie, this time we will limit the processing of read alignments to only estimate and output the assembled transcripts included in the merged transcriptome.

Question

1. Create a new directory *stringtie_quant* and run *StringTie* with the options listed below and using as input the sorted bam file.

```
-p 8
    To execute the program launching
    8 parallel search threads.

 -B
    To enable the output of Ballgown input
    table files (*.ctab) containing coverage
    data for the reference transcripts.

 -e
    To limit the processing of read alignments
    to requantify transcripts, without
    assembling new ones.

 -o GTF_FILE
    Replace GTF_FILE with the file were
    you will save the transcriptome
    requantification results,
    in your case specify also
    the directory stringtie_quant

 -G ANN_REF_FILE
    Replace ANN_REF_FILE with the
    merged gtf file
```

DESeq2 and *edgeR* are two popular Bioconductor packages for analysing differential expression. They both take as input a matrix of read counts mapped to particular genomic features (e.g., genes). The *StringTie* developers also provide a Python script (prepDE.py), currently in your scripts directory, designed to extract the read count information directly from the files generated by StringTie. You can find further information in the *StringTie* manual http://ccb.jhu.edu/software/stringtie/index.shtml?t=manual.

Create in the directory *stringtie_quant* a file named sample_list.txt. On the first line of this file write the name of the sample you have analysed so far and, space separated, the relative gtf requantification result. Finally run the prepDE.py script.

```
python2 scripts/prepDE.py -i stringtie_quant/sample_list.txt
```

The script will produce two result files: gene_count_matrix.csv and transcript_count_matrix.csv. These are the input files for the differential expression analysis or the principal component analysis (PCA).

Questions

1. In the transcriptcountmatrix.csv file are reported transcript counts for all the transcripts annotated in the merged transcriptome. Integrate these results with the gff-

compare results to identify intergenic (no overlap with other reference transcript) transcripts. Visualise on IGV the top 10 with highest count, loading the SRR1048063 alignment, the merged transcriptome and the ensembl reference bed file and saving the screenshots.