RNA-seq processing: extracting counts Hands-on

Software used in this practical

- Bowtie2 : Burrows-Wheeler Aligner
- Tophat: RNA-seq aligner
- IGV
- HTSeq
- SAMtools, http://samtools.sourceforge.net/
- Shell commands: wc, cat, head, tail, cut, sort, uniq, grep, awk

Data used in this practical inside the folder "**rna-seq-counts/data**":

- A fasta file as reference genome (chromosome 21):
  - chr21_ref_genome_sequence.fa

- One feature file with annotations for the reference genome (chromosome 21)
  - chr21_genome_annotation.gtf

- We have 6 samples (3 cases and 3 controls) in parired-end mode:
  - case1_read1.fastq
  - case1_read2.fastq
  - case2_read1.fastq
  - case2_read2.fastq
  - case3_read1.fastq
  - case3_read2.fastq
  - control1_read1.fastq
  - control1_read2.fastq
  - control2_read1.fastq
  - control2_read2.fastq
  - control3_read1.fastq
  - control3_read2.fastq

Now, start tutorial:

```
cd rna-seq-counts
```

Explore the reference genome file

```
ls -lh data/chr21_ref_genome_sequence.fa
head data/chr21_ref_genome_sequence.fa
```

Display the chromosomes/sequences names inside the reference genome file:

```
grep "^>" data/chr21_ref_genome_sequence.fa
```

Explore the GTF annotation file

The GTF (General Trasnfer Format) format consists of one line per feature, each containing 9 columns of data. See more details at:
http://www.ensembl.org/info/website/upload/gff.html

```
head data/chr21_genome_annotation.gtf
```

How many features do the GTF file contain?

```
wc -l data/chr21_genome_annotation.gtf
```

Display the type of features.

```
cut -f3 data/chr21_genome_annotation.gtf
cut -f3 data/chr21_genome_annotation.gtf | sort | uniq
```

Explore the FastQ files

```
head data/case1_read1.fastq
head data/case1_read2.fastq

tail data/case1_read1.fastq
tail data/case1_read2.fastq
```

Each read consists of four lines:

- Line 1 begins with a '@' character and is followed by a sequence identifier and an optional description (like a FASTA title line).
- Line 2 is the raw sequence letters.
- Line 3 begins with a '+' character and is optionally followed by the same sequence identifier (and any description) again.
- Line 4 encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence.

How many lines do the FastQ files contain? And reads?

```
wc -l data/case1_read1.fastq
wc -l data/case1_read2.fastq
```

Map against the reference genome using bowtie2 and tophat2

First, we need to **build an index** for bowtie2. Check the bowtie2-build parameters.

```
bowtie2-build
```

And then **build an index** for bowtie2.

```
bowtie2-build data/chr21_ref_genome_sequence.fa bowtie_index
```

Check the tophat2 parameters.

```
tophat2
```

And now we can run the **alignments** for the **paired end** files using tophat2.

```
tophat2 -r 300 -o case1_tophat_out bowtie_index data/case1_read1.fastq data/case1_read2.fastq
tophat2 -r 300 -o case2_tophat_out bowtie_index data/case2_read1.fastq data/case2_read2.fastq
tophat2 -r 300 -o case3_tophat_out bowtie_index data/case3_read1.fastq data/case3_read2.fastq

tophat2 -r 300 -o control1_tophat_out bowtie_index data/control1_read1.fastq data/control_read2.fastq
tophat2 -r 300 -o control2_tophat_out bowtie_index data/control2_read1.fastq data/control_read2.fastq
tophat2 -r 300 -o control3_tophat_out bowtie_index data/control3_read1.fastq data/control_read2.fastq
```

REMARK: For **paired end** samples the left and the right files have to be separated using an **space**.

If separated by a **coma**, tophat understands the two files contain reads form the same sample sequenced in a **single end** protocol.

Check the results files from tophat2 and use the SAMtools to explore the generated BAM files.

```
ls -ltr case1_tophat_out/
cat case1_tophat_out/align_summary.txt

samtools view case1_tophat_out/accepted_hits.bam | head
```

Interpret some alignments. In order to decode the SAM flags you can use the utility at: https://broadinstitute.github.io/picard/explain-flags.html

Sort BAM files

Use the SAMtools to sort the BAM files **by read names.** This arrangement mode is required by the htseq-count program but it is not suitable to use other tools like IGV for instance. Thus sometimes we will need to use two different BAM files per sample, with a different arrangement each.

```
samtools sort -n case1_tophat_out/accepted_hits.bam case1_sorted_by_name
samtools sort -n case2_tophat_out/accepted_hits.bam case2_sorted_by_name
samtools sort -n case3_tophat_out/accepted_hits.bam case3_sorted_by_name

samtools sort -n control1_tophat_out/accepted_hits.bam control1_sorted_by_name
samtools sort -n control2_tophat_out/accepted_hits.bam control2_sorted_by_name
samtools sort -n control3_tophat_out/accepted_hits.bam control3_sorted_by_name
```

You can see how the reads are sorted by name (ie. by pairs) in the BAM file.

```
samtools view case1_sorted_by_name.bam | head
```

or display only the read names:

```
samtools view case1_sorted_by_name.bam | cut -f1 | head
```

and find the original reads in the fastq file

```
grep -A 1 ENST00000270112_43591_43135_1_0_0_0_2:0:0_1:0:0_10 data/case1_read1.fastq
grep -A 1 ENST00000270112_43591_43135_1_0_0_0_2:0:0_1:0:0_10 data/case1_read2.fastq
```

What does the `-A 1` option does?
Where the two reads named exactly the same in the original files?

Now we can use htseq-count to compute read counts for each sample **at gene level**:

```
htseq-count --format=bam --stranded=no --type=gene case1_sorted_by_name.bam data/chr21_genome_annotation.gtf > case1.count

htseq-count --format=bam --stranded=no --type=gene case2_sorted_by_name.bam data/chr21_genome_annotation.gtf > case2.count

htseq-count --format=bam --stranded=no --type=gene case3_sorted_by_name.bam data/chr21_genome_annotation.gtf > case3.count


htseq-count --format=bam --stranded=no --type=gene control1_sorted_by_name.bam data/chr21_genome_annotation.gtf > control1.count

htseq-count --format=bam --stranded=no --type=gene control2_sorted_by_name.bam data/chr21_genome_annotation.gtf > control2.count

htseq-count --format=bam --stranded=no --type=gene control3_sorted_by_name.bam data/chr21_genome_annotation.gtf > control3.count
```

How do you have to change the previous command lines to get counts **at exon level**?

Observe the structure of the “count” files.

```
head case1.count
tail case1.count
```

Explore the gene “ENSG00000141959” in sample “case1” for instance. Which is the count number assigned to this gene by `htseq-count` at a *gene* level?

```
grep ENSG00000141959 case1.count
```

Explore regions and visualize BAM files (IGV viewer)

Sorting BAM files by names is needed to use htseq-count but is not suitable to use other tools like IGV for instance. Thus sometimes we will need to use two different BAM files per sample, with a different arrangement each.

Use the SAMtools to sort the BAM files **by chromosome position** (standard/default sorting):

```
samtools sort case1_tophat_out/accepted_hits.bam case1_sorted_by_pos
samtools sort case2_tophat_out/accepted_hits.bam case2_sorted_by_pos
samtools sort case3_tophat_out/accepted_hits.bam case3_sorted_by_pos
```

```
samtools sort control1_tophat_out/accepted_hits.bam control1_sorted_by_pos
samtools sort control2_tophat_out/accepted_hits.bam control2_sorted_by_pos
samtools sort control3_tophat_out/accepted_hits.bam control3_sorted_by_pos
```

And index them (this step is needed by the IGV)

```
samtools index case1_sorted_by_pos.bam
samtools index case2_sorted_by_pos.bam
samtools index case3_sorted_by_pos.bam

samtools index control1_sorted_by_pos.bam
samtools index control2_sorted_by_pos.bam
samtools index control3_sorted_by_pos.bam
```

Explore the alignments for the gene “ENSG00000141956”.

Get the feature gene “ENSG00000141956” from the annotation file.

```
grep ENSG00000141956 data/chr21_genome_annotation.gtf | awk '{ if ($3 == "gene") { printf("%s\n", $0); } }'
```

Get the start and end position of the feature gene “ENSG00000141956” from the annotation file.

```
grep ENSG00000141956 data/chr21_genome_annotation.gtf | awk '{ if ($3 == "gene") { printf("%s\n", $0); } }' | cut -f4,5
```

Using the SAMtools, get the alignments for the gene “ENSG00000141956”.

```
samtools view case1_sorted_by_pos.bam 21:43218385-43299591
```

Open the IGV viewer, load the genome “chr21_ref_genome_sequence.fa” and the BAM file case1_sorted_by_pos.bam.

Visualize the alignment “ENST00000398548_66784_67235_0_1_0_0_0:0:0_2:0:0_2” (a read paired whose mate is mapped). Check their mismatches and the distance between mates.

Visualize the alignment “ENST00000398548_77167_77653_0_1_0_0_1:0:0_4:0:0_c” (a read paired whose mate is unmapped).