Practical 1

Quality control and alignment

Concepts and main aspects of RNA-Seq
WTCHG, 25 April 2016
Irina Pulyakhina

Today we will cover such steps of RNA-Seq data analysis as alignment and quality control.

1) Pre-alignment quality control

In order to estimate whether the sequencing experiment went well at all, we will analyze quality of the raw sequencing data. We will look at such characteristics of NGS data as the actual reported sequencing quality for each base (Phred quality score), sequence content, quantity of N nucleotides.

Look at the content of a *fastq* file using the following command:

```
> cd Step01_QC/
> zcat read1.fastq.gz | head -4
```

In general, `cat` or `zcat` for gzipped files is a way to view files in a terminal. However, it will print the whole file for you, which is not practical when a file contains hundreds of lines. Therefore we limit the number of lits that we want to see to four top lines using `head -4`. To see N bottom lines of a file you can use `tail -4`.

The fourth line (starting with `<,AF<`) contains per-base sequencing quality, also known as Phred quality score. To know more about Phred quality score, you can visit: https://en.wikipedia.org/wiki/Phred_quality_score.

We will use a tool called `fastqc` to assess the quality of the sample. Run the following command to let your Linux environment know the location of the tool:

```
> fastqc=`which fastqc`
```
Now you do not have to specify the whole path to the program and can call it like this:

```
> $fastqc --help
```
Create a new folder to store the output of `fastqc`:

```
> mkdir RNAcourse_read1_fastqc_output
```

Now we are ready to run `fastqc` and generate a quality report:

```
> $fastqc -f fastq \
  -o RNAcourse_read1_fastqc_output \
  read1.fastq.gz
```

Note that "\" is used here to continue the command on the next line. This is a common practice for programmers, as we find super-long lines less convenient to work with than shorter lines. You can either type your commands the way they are printed here (using "\") or as one long line, without "\" (in this case, the previous command becomes "`$fastqc -f fastq -o RNAcourse_read1_fastqc_output read1.fastq.gz`").

Open the *html* report generated by `fastqc`. Which quality filters failed? Can you give an explanation to each failure? Does it indicate that the data is low quality and cannot be used?

Exercise:
- look at the first generate quality plot – "Per base sequence quality". From the X axis you can see that quality values are grouped for all the positions starting from position 10. How can you remove this grouping and plot values per position for all the nucleotides?

2) Alignment

After we analyze the quality of the experiment and make a conclusion that it is high and the data does not need any pre-processing (such as adapter trimming or removing low quality bases), we can proceed to the next step – alignment.

`Tophat2` is one of the most popular aligners for RNA-Seq data available at the moment. To align your data to the reference sequence with `tophat2` (just like with any other aligner), you first have to index the reference sequence. This is a very time-consuming procedure, and for the sake of space and time today we will work with only one chromosome of the human genome – chromosome 22. Note that almost every aligner requires its one specific reference sequence index, unless the core of two aligners is the same. This is the case for `tophat2` and `bowtie2`, therefore, we can use index built with `bowtie2` for `tophat2` alignment.

Go to the folder containing the reference sequence:

```
> cd alignment_Step02
```

There you will find the *fastq* sequence of chromosome 22. We will build the index using `bowtie2-build`:

```
> bowtie2-build chr22.index.fa chr22.index
> mkdir Chr22.index
> mv chr22.index.* Chr22.index/
```

Now, we are ready to run the alignment:

```
> tophat2=`which tophat2`
> $tophat2 \
  Chr22.index/chr22.index \
  read1.to_map.fq \
  read2.to_map.fq
```

You have just run RNA-Seq alignment with default options.

Exercise:
– Make `tophat2` report only one alignment per read;
– Limit maximum allowed big insertion to 5,000;
– Make `tophat2` write each new alignment results in a new folder.

3) Post-alignment quality control

Let's look at the results:

```
> cd tophat_out/
```

By default `tophat2` provides output in a binary BAM format. We have to use `samtools` to see the actual alignment in a human-readable format:

```
> samtools=`which samtools`
> $samtools view accepted_hits.bam | head
```

Look at the CIGAR string. All alignments you are looking at right now are perfect matches, hence "100M" as the CIGAR string. Find the alignments covering exon-exon junctions:

```
> $samtools view accepted_hits.bam | awk '$6 ~ /N/{print $0}'
```

How many alignment are there in total?

```
> $samtools view accepted_hits.bam | wc -l
```

Are all reads mapped uniquely or multiple times? You can get this information from one of the columns in a sam file, namely "NH:i:20":

```
> $samtools view accepted_hits.bam | grep 'NH:i:20'
```

Look at the other files in the results folder. One of the files contains only reads that were not mapped to the reference at all. How many unmapped reads are there?

```
> $samtools view unmapped.bam | wc -l
```

Exercise:

– When you look at the unmapped reads, which columns of the *sam* file still provide any information about the reads?

– Do you have any suggestions on what information is stored in “deletions.bed” (in each column)?

– Look at the file “junctions.bed”. It has the same file extention as “deletions.bed” which suggests the same file format, however, the files look somewhat different. Which extra column does it have compared to “deletions.bed”?

– Look at the SAM file specification (Step02_alignment/SAMv1.pdf) if you need/want to know more about SAM files and the information contained in each column.