Quality assessment of NGS data

Ines de Santiago

July 23, 2015

Contents

1 Introduction

This training gives an introduction to quality assessment of NGS data through various workflows using command line tools. We will use a software called FASTQC to assess the quality of sequenced reads and lead you through the different preprocessing steps, from removing reads of low quality to removing adapter contamination.

2 Checking read quality with FASTQC

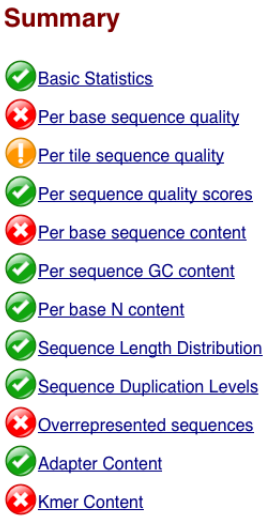
FASTQC checks whether a set of sequence reads in a .fastq file exhibit any unusual qualities. There are two ways in which FASTQC can be run: in "command line" mode, or as a GUI (graphical user interface). This exercise addresses the command line version of FASTQC.

Use Case: Run FASTQC to check read quality.

The data we will use in this session consists of a fastq file with example sequences. The file is called "sample.fastq" and is located in the" located in the "/qc" directory:

```
fastqc sample.fastqc
```

FASTQC generates a html report with a nice graphical summary output about the quality of sequencing reads. Two files are generated, one compressed, and one not. To view the report, open the file "sample_fastqc.html" in a browser. Your output should have a menu on the left-hand side that looks like this:



Make sure you understand the results. Do they look OK? What do the warnings and errors mean?

3 Preprocessing with FASTX-Toolkit

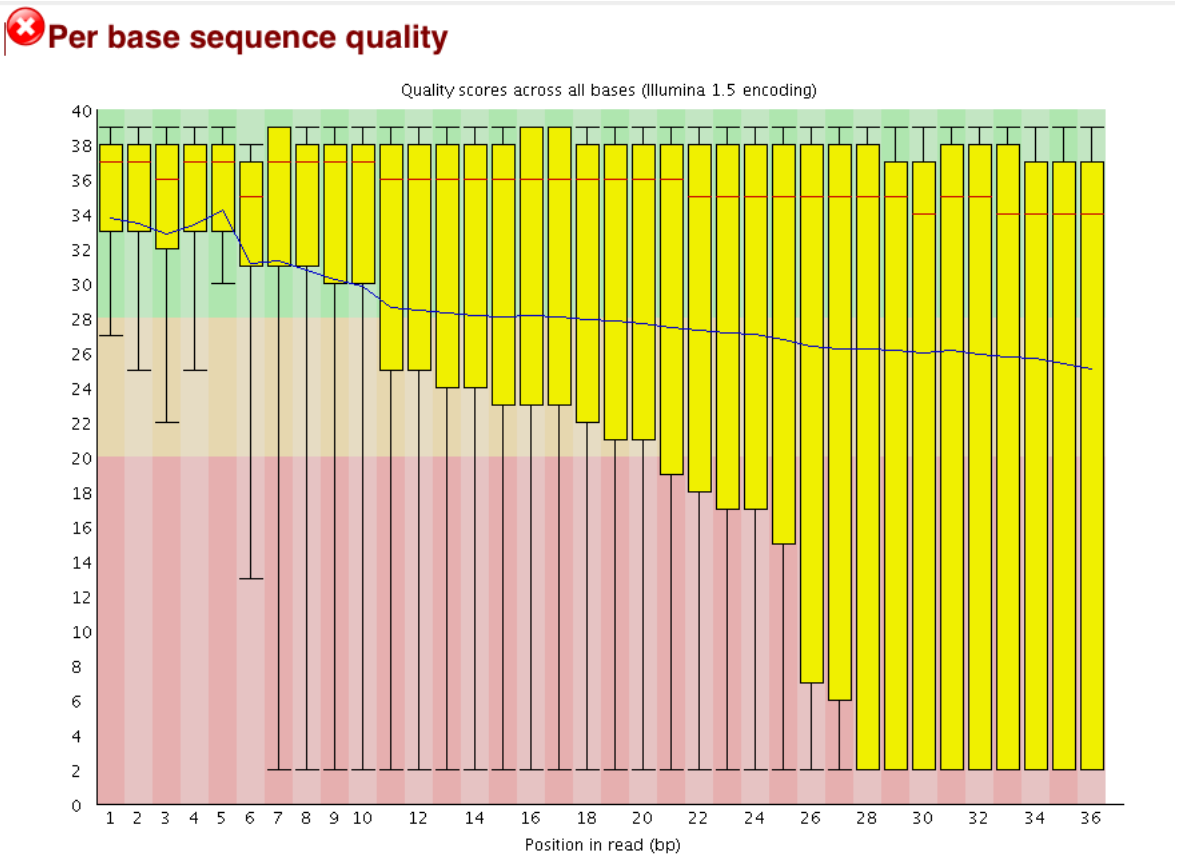
In this section of the tutorial we will be using FASTX-Toolkit (http://hannonlab.cshl.edu/fastx_toolkit/) to filter and trim sequences based on quality. The FASTX-Toolkit is a collection of command line tools for preprocessing of FASTA/FASTQ files. There are many tools available within FASTX-toolkit, we will be using two of those tools:

- fastq_quality_filter: Filters sequences based on quality
- fastx_trimmer: Shortening reads in a FASTQ or FASTQ files (removing barcodes or noise).

For most programs and scripts, you can see their instructions by typing their name in the terminal followed by the flag -h. There are many options available, and we will use only a few of those.

3.1 Preprocessing with FASTX-Toolkit: fastq_quality_filter

The 'Per-base sequence quality' plot from sample.fastq file shows the average and range of the sequence quality values across the read. It looks like there are many reads in the file with very low quality. We will remove low quality reads using `fastq_quality_filter` tool:



Use Case: Use fastq_quality_filter to remove reads with lower quality, keeping only reads that have at least 75% of bases with a quality score of 20 or more.

```
fastq_quality_filter -v -Q 64 -q 20 -p 75 -i sample.fastq -o sample_filtered.fastq
```

the following options were used:

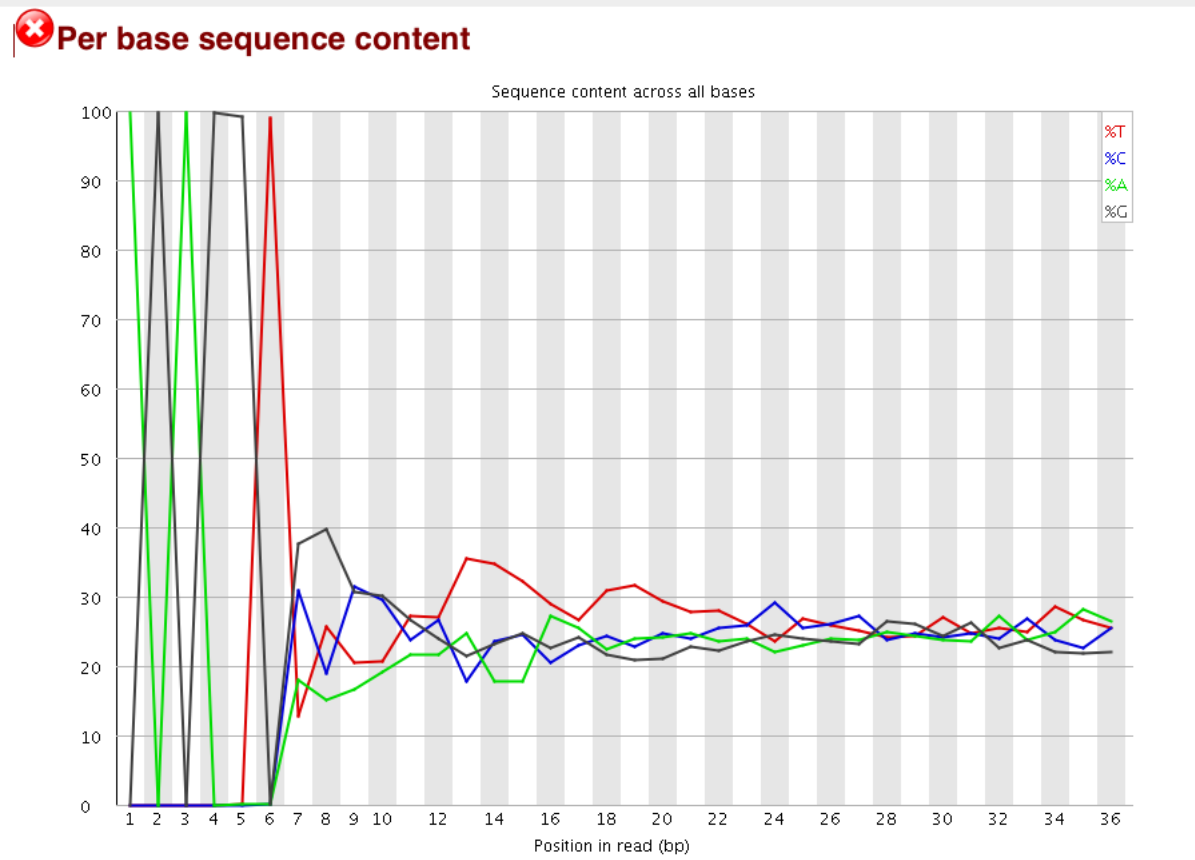
```
-i:  input file
-o:  output file
-v:  report number of sequences
-Q 64: determines the input quality ASCII offset (in this case 64). You will need version 0.0.10 or later
-q 20: the quality value required.
-p 75: the percentage of bases that have to have that quality value
```

fastq_quality_filter should give you the following output:

```
Quality cut-off: 20
Minimum percentage: 75
Input: 9053 reads.
Output: 6629 reads.
discarded 2424 (26%) low-quality reads.
```

3.2 Preprocessing with FASTX-Toolkit: fastx_trimmer

The 'Per-base sequence content' plot from sample.fastq file shows the average percentage of A, G, C and T across the read length. The pattern we see here at the beginning of the reads show some sort bias for the first 6 bases. We will use `fastx_trimmer` to deal with this issue:



Use Case: Use fastx_trimmer to trim the first 6 bases from reads.

```
fastx_trimmer -v -f 7 -l 36 -i sample_filtered.fastq -o sample_filtered_and_trimmed.fastq
```

the following options were used:

```
-f: first base to keep
-l: last base to keep
-i: input file
-o: output file
-v: report number of sequences
```

fastx_trimmer should give you the following output:

```
Trimming: base 7 to 36
Input: 6629 reads.
Output: 6629 reads.
```

4 Cleaning adapter containing reads from FASTQ data using cutadapt

In this section we will demonstrate the effect of adapter contamination.

We will use another example dataset, the file is called 'sample2.fastq' and is located in the '/qc' directory. First, let's start by running FASTQC with this file.

4.1 Checking read quality with FASTQC

Use Case: Run FASTQC to check read quality

```
fastqc sample2.fastqc
```

Now open the output file "sample2_fastqc.html" in a browser and inspect the results.

Many of the issues present in the report can be linked to adapter contamination. Do you have adapter sequences in your reads?

4.2 Using cutadapt

Under 'overrepresented sequences' option you will see the following table:

Overrepresented sequences

Sequence	Count	Percentage	Possible Source
GATCGGAAGAGCACACGTCTGAACTCCAGTCACACA	28971	28.971000000000004	TruSeq Adapter, Index 5 (100% over 36bp)
GCTAACAAATACCCGACTAAATCAGTCAAGTAAATA	392	0.392	No Hit
GTTAGCTATTTACTTGACTGATTTAGTCGGGTATTT	356	0.356	No Hit
GATCGGAAGAGCACACGTCTGAACTCCAGTCACACC	108	0.108	TruSeq Adapter, Index 1 (97% over 36bp)
GATCGGAAGAGCACACGTCTGAACTCCAGTCACACG	107	0.107	TruSeq Adapter, Index 15 (97% over 36bp)

As seen here, one sequence is present in more than 28.97% of the reads. FASTQC identifies the primer as "TruSeq Adapter, Index 5", possibly a left over from the library construction process or from a PCR amplification issue.

We now use the cutadapt utility to remove reads containing the TruSeq Adapter "GATCGGAAGAGCACACGTCT-GAACTCCAGTCACACA":

Use Case: Use cutadapt to remove the unwanted adapter sequences from sample2.fastq.

```
cutadapt -m 20 -e 0.1 -a GATCGGAAGAGCACACGTCTGAACTCCAGTCACACA sample2.fastq \
-o sample2--cutadapt.fastq
```

The following options were used:

```
-a: The sequence of the adaptor is given with the -a option
-m 20: throws away processed reads shorter than 20 bases.
-e 0.1: The level of error tolerance is adjusted by specifying
a maximum 10% error rate. Allowed errors are mismatches,
insertions and deletions.
-o: output file
```

cutadapt should give you the following output:

```
cutadapt version 1.1
Command line parameters: -m 20 -e 0.1 -a GATCGGAAGAGCACACGTCTGAACTCCAGTCACACA
sample2.fastq -o sample2--cutadapt.fastq
Maximum error rate: 10.00%
   Processed reads: 100000
     Trimmed reads: 32792 ( 32.8%)
   Total basepairs:       3600000 (3.6 Mbp)
 Trimmed basepairs:       1124674 (1.1 Mbp) (31.24% of total)
   Too short reads: 31113 ( 31.1% of processed reads)
    Too long reads: 0 (  0.0% of processed reads)
        Total time:       5.13 s
     Time per read:       0.05 ms

=== Adapter 1 ===

Adapter 'GATCGGAAGAGCACACGTCTGAACTCCAGTCACACA', length 36,
was trimmed 32792 times.

Lengths of removed sequences
length  count   expected
3       1267    1562.5
4       277     390.6
5       52      97.7
6       22      24.4
7       1       6.1
8       2       1.5
9       2       0.4
10      7       0.1
11      2       0.0
12      1       0.0
16      46      0.0
17      19      0.0
18      9       0.0
20      1       0.0
21      38      0.0
23      35      0.0
24      1       0.0
36      31010   0.0
```

Use Case: Inspect the results after cleaning the data (use FASTQC again).

```
fastqc sample2--cutadapt.fastq
```

Now many of the issues in the report seem to have been resolved. Only a low quality issue with the 3' of reads and some kmer noise seems to remain. Most mappers should not be affected by contaminating adapters or kmers because those reads should not map to the reference genome and are discarded during the alignment step. However, when a known contaminating sequence is detected or known to be present it can be good practice to remove and trim reads prior to alignment.

5 Conclusions

When you get your sequences back from a sequencing facility, its important to check that they are of high quality. FASTQC can tell us that there are unusual features in your fastq file, but it cant necessarily explain why. If you want more information on what each piece of FASTQCs output means, the documentation is available online at http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/Help/

Many of the issues observed are related to poor sequencing quality or bad base calling, adapter sequences or library contamination. When known adapter/linker sequences are detected, a simple removal of these contaminating sequences can make a very big difference in the quality of your fastq reads. There are many options that you can use with cutadapt, for instance it is possible to specify more than one adapter sequence, trim more than one adapter from each read, etc. We recommend reading cutadapt documentation for more details and examples (http://cutadapt.readthedocs.org/en/latest/guide.html).