Trimming and quality control (May 2017)

Alexander Jueterbock, Martin Jakt*

PhD course: High throughput sequencing of non-model organisms

Contents

After a general introduction to the UNIX command line, it is time for you to analyze your own fastq files. The first important step for any kind of sequencing data is to get rid of adapter contamination and bad quality reads. In this tutorial we will use the programs FastQC and TrimGalore! to check the quality of the sequenced libraries before and after trimming.

IMPORTANT NOTE Before you get started: to compare characteristics of your libraries, please keep record of the resulting numbers, like the number of raw reads, reads after quality control, number of mapped reads etc. This helps to identify peculiarities/outliers in your libraries which may either be due to biological peculiarities of your species or unknown technical issues.

Log on (with `ssh`) to the remote computer with the `-X` option (like `ssh -X user@158...`) to be able to use graphical interfaces.

1 Overview of sequence lengths

Next Generation Sequencing data is generally stored in fastq files. Most of the time the data are compressed, either in .zip or in .gz format.

*Nord University, Norway

If your file is zip-compressed, you can use the following command to unzip it:

```
unzip FILE.fastq.zip
```

If your file iz gz-compressed, use the following command instead:

```
gunzip FILE.fastq.gz
```

NOTE: For paired-end sequencing from Illumina, you have two files. One file with the forward reads (`Sequence_R1.fq`) and one file with the reverse reads (`Sequence_R2.fq`).

To get a quick impression of the minimum and maximum read lengths in your fastq file, you can use the following commands (replace `FILE.fastq` with your own filename):

```
awk '{if(NR%4==2) print length($0)}' FILE.fastq| sort -n | head -n1
awk '{if(NR%4==2) print length($0)}' FILE.fastq| sort -n | tail -n1
```

It reads like this: measure the length of every second line in every group of 4 lines (the sequence line in a fastq file), `sort` it (numerically with `-n`) and print out either the first (smallest) value with `head` or the last (biggest) value with `tail`. `NR` represents the current line number and the `%` sign is the modulus operator, which divides the line number by 4 (`NR%4`) and returns only the remainder. This extracts all the sequences, which are on line 2,6,10,14...

The following command allows you to count the sequence lengths:

```
awk '{if(NR%4==2) print length($0)}' FILE.fastq | sort -n | uniq -c > read_length.txt
```

The lines that follow make use of the program R. If you copy and paste the code into the command line, you will get an overview graphic of the sequence length distribution

```
cat >> Rplot.r << 'EOF'
reads<-read.csv(file="read_length.txt", sep="", header=FALSE)

png(filename = "SequenceLengthDistribution.png",
         width = 480, height = 480, units = "px", pointsize = 12,
          bg = "white")
plot(reads$V2,reads$V1,type="l",xlab="read length",ylab="occurences",col="blue")
dev.off()

EOF


R CMD BATCH Rplot.r
```

You can open the created figure with the GNOME image viewer using the following command:

```
eog SequenceLengthDistribution.png
```

Does the sequence length-distribution meet your expectations?

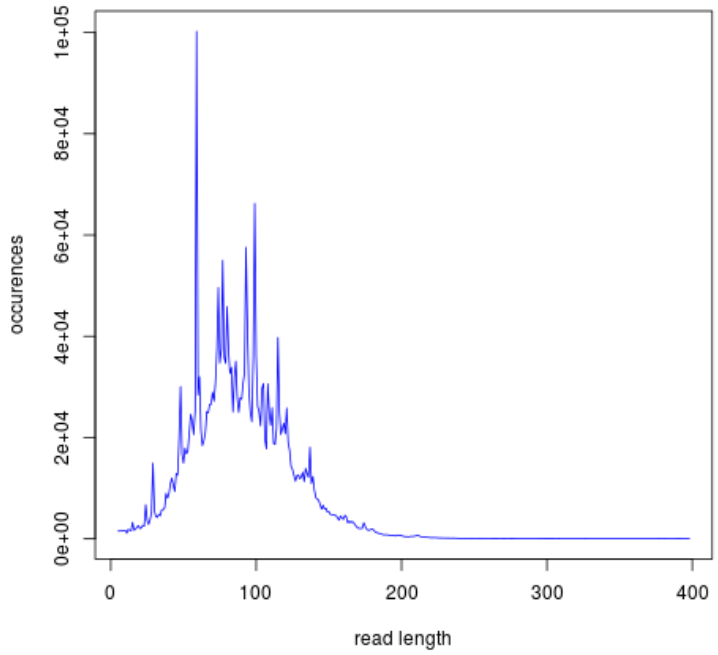

Figure 1: Example graphic of the length distribution in a fastq file

NOTE: While the Ion Torrent sequences will differ in length (as in Figure 1), the Illumina sequences will all have the same read length (301 bp).

2 Quality control

To inspect the quality of the sequencing data, we use FastQC. In the installation and setup instructions of the program (link), you will find that FastQC can run in an interactive mode or in a command line mode. This tutorial uses the command-line version.

FastQC knows a number of standard adapter sequences used for HTS, including adapters used on Illumina platforms; To get an overview of the contaminants (overrepresented sequences) that FastQC will look for by default, type

```
less /usr/share/fastqc/Contaminants/contaminant_list.txt
## or possibly:
less /usr/local/fastqc/FastQC_v0.11.3/Configuration/contaminant_list.txt
```

However, FastQC is not aware of the sequences used by the IonTorrent platform. To inform FastQC of the Ion Torrent adapter sequences call FastQC with the `--contaminants` option to specify a file containing the adapter sequences we have used.

So, to run FastQC on your file, type:

```
fastqc --contaminants adapters.txt FILE.fastq
```

The adapters.txt file contains the adapter sequences in a name[tab]sequence format, like

```
IonTorrentAAdapter      CCATCTCATCCCTGCGTGTCTCCGACTCAG
IontTorrentP1Adapter    CCACTACGCCTCCGCTTTCCTCTCTATGGGCAGTCGGTGAT
```

You can create this file on the adapters.txt file on the remote computer with:

```
touch adapters.txt
```

Open the text file with the program nano:

```
nano adapters.txt
```

Then copy and paste the name-sequence combinations of the Ion Torrent adapters (see above) into the file and close the file by pressing Ctrl+O, then Ctrl+X on your keyboard.

The output of the FastQC program will be saved in a folder that has the name of your fastq file and ends with fastqc, like `FILE_fastqc`. Use the `cd` command to move into the folder and open the produced `fastqc_report.html` either with `firefox` or `chromium-browser` (one of the two should work).

```
cd FILE_fastqc
firefox fastqc_report.html
chromium-browser fastqc_report.html
```

Scrolling through this html file on the remote computer will be quite slow. it may be more convenient to copy the output folder to your computer with FileZilla or 'rsync' (see in the 'Unix tools' session) . Get familiar with the output of each module.

For example, it is normal that the the per base sequence quality drops towards the end of the read, as seen in Figure 2. In the next section we will see how to trim away these low-quality reads.

The figure on duplication levels (Figure 3) informs you about the percentage of duplicate reads in your sequenced library. Duplicates result from primer or PCR bias towards these reads. As they can skew genotype estimates, we will remove duplicate reads later in the week before SNP calling.

You can find guidance on how to interpret the output of each module here

3 Trimming low quality reads and adapters

TrimGalore! is a wrapper script to automate quality and adapter trimming as well as quality control (User Guide).

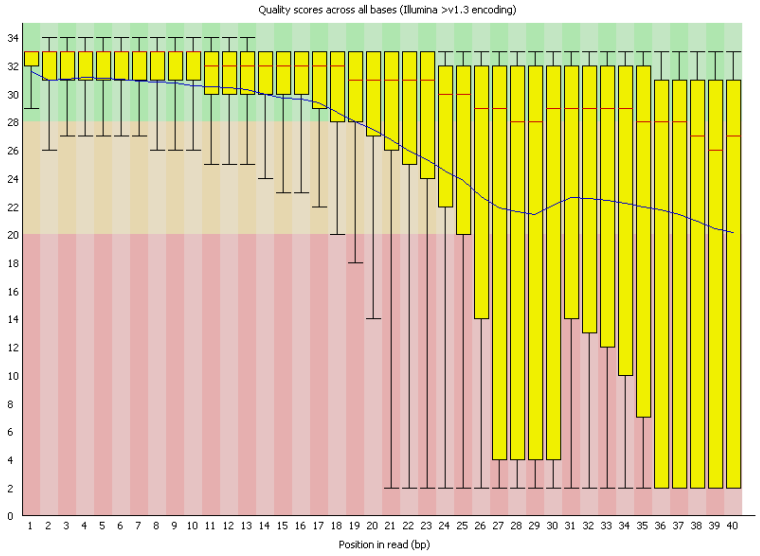

Figure 2: Per base sequence quality (from link)

When the program is installed, it can be used with

```
trim_galore [options] <filename(s)>
```

You can get an overview of the options with the `--help` option:

```
trim_galore --help
```

With the default settings, TrimGalore! trims low-quality ends with a Phred quality score threshold of 20 (can be changed with `-q`) and discards reads that become shorter than 20 bp (can be changed with `--length`).

TrimGalore! uses the program Cutadapt to find and remove adapters from the 3' end of the reads (see Fig. **??**). The program Cutadapt itself gives you more options for adapter trimming and allows you to remove adapters also from the 5'-end of the sequence (see `http://cutadapt.readthedocs.org/en/latest/guide.html`)

3.1 Trimming Ion-Torrent adapters

The Ion-P1- and Ion-A-adapters are supposed to be automatically trimmed off on the Ion Server. So, the fastq files with the raw reads should not contain these adapters anymore. Still, it is good to check if there are any adapters left in your library - they can have negative effects on further analyses.

The adapters used for Ion Torrent sequencing are shown in Fig. **??** and their orientation in the libraries is shown in Fig. **??**.

.

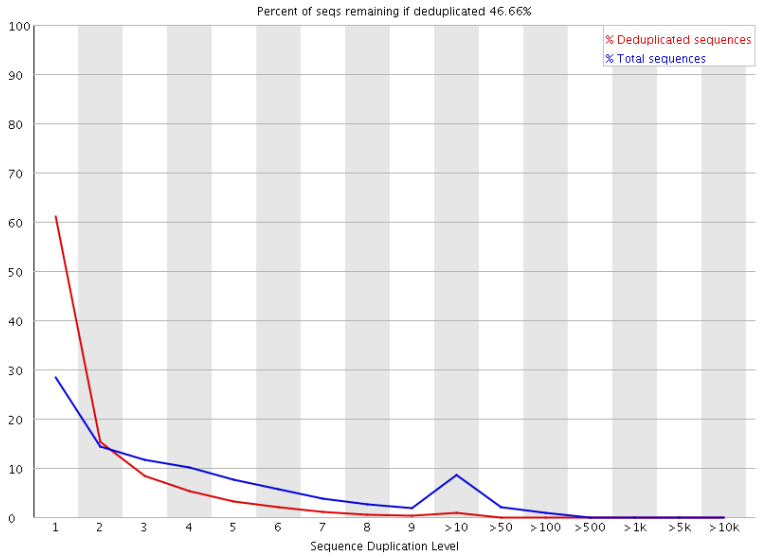

Figure 3: Per base sequence quality (from link)

To trim off the A-adapter, use TrimGalore! with the command:

```
trim_galore \
-a CCATCTCATCCCTGCGTGTCTCCGACTCAG \
--stringency 3 \
FILE.fastq
```

The \ sign just means that the command continues on the next line. You could type the entire command on a single line.

The option `--stringency 3` means that a >3bp overlap with the adapter sequence will be trimmed off the 3' end. The program writes a file that ends with `trimming_report.txt`, which reports the number of reads that have been trimmed and/or removed.

The output file has the ending `trimmed.fq`. Use this file as input to TrimGalore! to trim off the P1-adapter:

```
trim_galore \
-a CCACTACGCCTCCGCTTTCCTCTCTATGGGCAGTCGGTGAT \
--stringency 3 \
--fastqc FILE_trimmed.fq
```

The `--fastqc` option will automatically run FastQC in the default mode. Compare the FastQC outputs before and after trimming.

Figure 4: 3'- and 5'-adapter trimming (source)

```
Ion A Adapter (non-barcoded)
5'-    CCATCTCATCCCTGCGTGTCTCCGACTCAG-3'
3'-T*T*GGTAGAGTAGGGACGCACAGAGGCTGAGTC-5'
```

```
Ion P1 Adapter
5'-    CCACTACGCCTCCGCTTTCCTCTCTATGGGCAGTCGGTGAT-3'
3'-T*T*GGTGATGCGGAGGCGAAAGGAGAGATACCCGTCAGCCACTA-5'
```

Figure 5: Non-barcoded Ion-A and -P1 adapter sequences. In each sequence, a "*" indicates a phosphorothioate bond, for protection from nucleases and to preserve the directionality of adapter ligation. This is not relevant for adapter trimming.

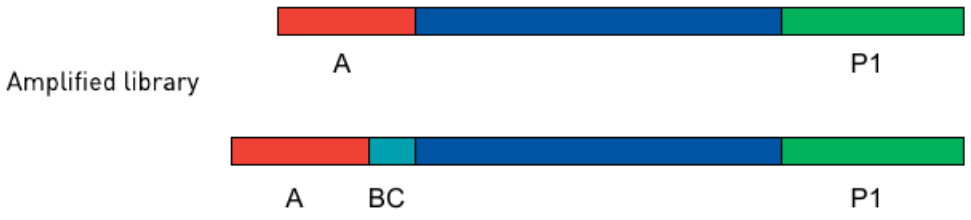

Figure 6: Ion adapters in the amplified library. BC is an optional barcode sequence.

3.2 Trimming Illumina adapters

Depending on the settings for Illumina sequencing, the adapters can be automatically removed from the fastq files that you get from the sequencing machine. This, however, has to be defined before sequencing. If you are not sure whether adapters have been trimmed off or not, it is safe to trim the adapters before using the sequences for any further analyses.

The Illumina adapters are as follows:

```
TruSeq Universal Adapter:
5' AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT 3'

TruSeq Indexed Adapter
5' P*GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATCTCGTATGCCGTCTTCTGCTTG 3'
```

Here, `NNNNNN` represents a barcode of six nucleotides in the indexed adapter.

TrimGalore! can be run with the option `--illumina`. This trims the first 13bp of the Illumina universal adapter `AGATCGGAAGAGC`. This option removes illumina adapters from most standard libraries, including TruSeq adapters.

The location of this sequence in the TruSeq adapter is shown here:

```
TruSeq Universal Adapter:
5' AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT 3'
   Reverse                                        CGAGAAGGCTAGA

TruSeq Indexed Adapter
5' P*GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATCTCGTATGCCGTCTTCTGCTTG 3'
    AGATCGGAAGAGC
```

The A on the 5'-end of the TruSeq indexed adapter is added during A-tailing of your DNA library fragments. The orientation of the adapters in the illumina library are shown in Fig. **??**.

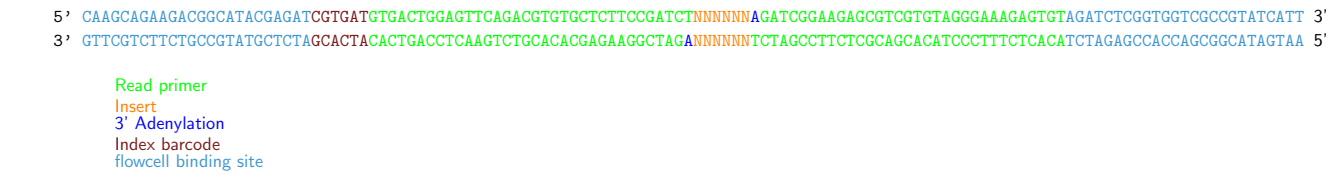

Figure 7: Orientation of the illumina adapters around the DNA inserts

TrimGalore! also performs trimming of paired-end libraries, like the Illumina libraries that were prepared in this course. This allows us to discard read pairs without disturbing the sequence order of FastQ files which is required by many aligners. With the option `--paired` TrimGalore! expects two paired fastq input files, like `file1_1.fq` and `file1_2.fq`. Here, both sequences of a sequence pair must have a certain minimum length (specified by the `--length` option) in order to be kept. If only one of the two paired end reads becomes too short, the option `--retain_unpaired` can be applied to write the long-enough unpaired read to either `unpaired_1.fq` or `unpaired_2.fq`. The length cutoff for unpaired single end reads is governed by the parameters `--length_1` and `--length_2`

To trim our illumina paired-end libraries we can use:

```
trim_galore \
--illumina \
--stringency 3 \
--paired \
--retain_unpaired \
--length_1 21 \
--length_2 21 \
--fastqc \
Illumina_R1.fastq \
Illumina_R2.fastq
```

This command runs fastqc automatically on the trimmed libraries. In addition to the `fastqc.html` files, you will find fastq files with the validated sequences (`Illumina_R1_val_1.fq`, `Illumina_R2_val_2.fq`) and with the unpaired sequences (`Illumina_R1_unpaired_1.fq`, `Illumina_R2_unpaired_2.fq`). The files ending with `trimming_report.txt` provide information on the number of reads that have been trimmed and/or removed.

You can now compare the quality of your raw libraries and your quality-trimmed libraries. What did improve? Are there still any problems with your libraries after trimming?

4 BONUS Running programs in the background with `nohup`

What if your data analysis on a remote server takes several hours, days, or even weeks, to finish? No worries, you don't need to be connected to the remote server while the data are being analysed. Here, you will learn the tools that allow you to start an analysis, disconnect from the server, and then look at the progress or the results at a later time point.

The `nohup` tool allows you to run a process in the background; which means that, while the analysis is running, you can do other tasks in parallel or log off from the remote server.

Imagine the `nohup` tool as a bracket which encloses the command that you want to run in the background:

```
nohup ... &
```

Always, `nohup` precedes and `&` follows the command that you want to run in the background (here shown as `...`). Let's say you want to run the command `ls -lhcrt` (which lists all files and subdirectories in your current directory) in the background.

```
nohup ls -lhcrt &
```

When you hit ENTER, the terminal prints out some information:

```
[1] 21118
nohup: ignoring input and appending output to 'nohup.out'
```

The number `21118` (which will differ in your case) in the first line is the process-ID of your background-process. The second line informs you that all 'results', that would be normally

printed in the terminal window, are now redirected to the file `nohup.out`.

4.1 Using the process-ID

If you have started a process that takes several hours to finish, then you can use the process-ID to see if the process is still running. For this, you can use the `ps` command with the `-p` option, which reports the status of a process with a certain process ID. To see the status of the process I have started above, I would use:

```
ps -p 21118
```

The output is

```
PID TTY          TIME CMD
```

Since this is only the header line of the process specifications, the process must have finished. Here:

- `PID` indicates the process-ID
- `TTY` indicates the controlling terminal
- `TIME` shows the time that the process is running already
- `CMD` shows the command name

If the process would still run, you would get a line similar to:

```
PID  TTY          TIME CMD
21118 ?        00:00:04 ls
```

The `top` tool provides an ongoing look at processor activity in real time, similar to Figure **??**.

At the top of the screen, it lists processes ordered by their CPU usage system (with the most intensive on top). Besides other information, it shows which user is running which process, as well as the process-ID. You can quit the program by hitting `q`.

The process-ID also allows you to cancel the process before it finishes. Cancelling processes comes in handy when you figure out that you started them with the wrong parameters or input files and you want to re-start with different settings. The `kill` command allows you to cancel a specific process.

```
kill 21118
```

This would cancel the process that we started before in the background. If you can't remember the process-ID but want to cancel all `ls` processes, then you could use the `pkill` command in

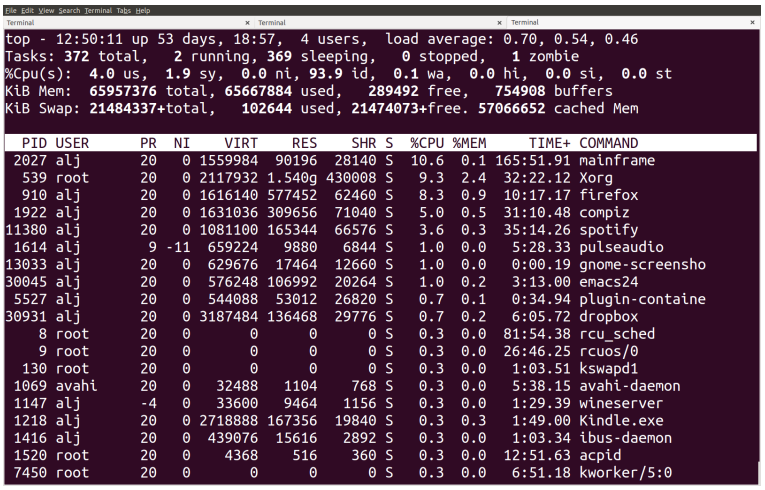

Figure 8: Screenshot of the `top` tool output

the following way:

```
pkill ls
```

Compared to the `kill` command, the `pkill` command allows you to specify the command-name instead of the process-ID of the running process that you want to cancel.

If you don't have a record of the PID you can find out the id of processes being run by a specific user by combining `ps` and `grep`:

```
ps aux | grep user_name
```

where `user_name` is your own user name. That will list all processes started by you (well using your id in any case). The `aux` option specifies all processes and the manner in which these are printed out. If you read the man file (`man ps`) you will see that there are ways in which you can get ps to only list processes started by the current user (`ps -eu`) in a similar manner and that `ps aux` is BSD syntax. So `grep` isn't really needed here, but I tend to like the way this formats the output.

4.2 Redirecting output

By default, the `nohup` command redirects all information from the terminal window to the `nohup.out` file. If the file exists already, it will not be overwritten. All new information will be appended to the end of the file. With the `>` operator, you can redirect the output to a different file. For example, to redirect the output of the `ls` command to the file `Directory-Listing.txt`, you can use the command

```
nohup ls -lhcrt > Directory-Listing.txt &
```

So, the redirection-operator (`>`) is followed by the name of the target file and precedes the closing `&` operator of the `nohup` command. If you want to save the output to a file in a different directory, just specify the entire file-path that precedes your target file, like:

```
nohup ls -lhcrt > /home/alj/Documents/DirectoryListing.txt &
```

Emacs 24.3.1 (Org mode 8.3.4)