Trimming and quality control (2015-06-03)

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 of bad quality reads. In this tutorial we will use the prorgams FastQC and TrimGalore! to check the quality the sequenced libraries before and after trimming. We will also learn few more UNIX commands that extract important information from fastq files and that allow you to turn off your computer while the analysis continues to run on the remote server.

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 due to technical issues.

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

[*]University of Nordland, Norway

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.

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
```

To get a quick impression on 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 follwowing command allows to count the sequence lengths:

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

The line that follows makes 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
```

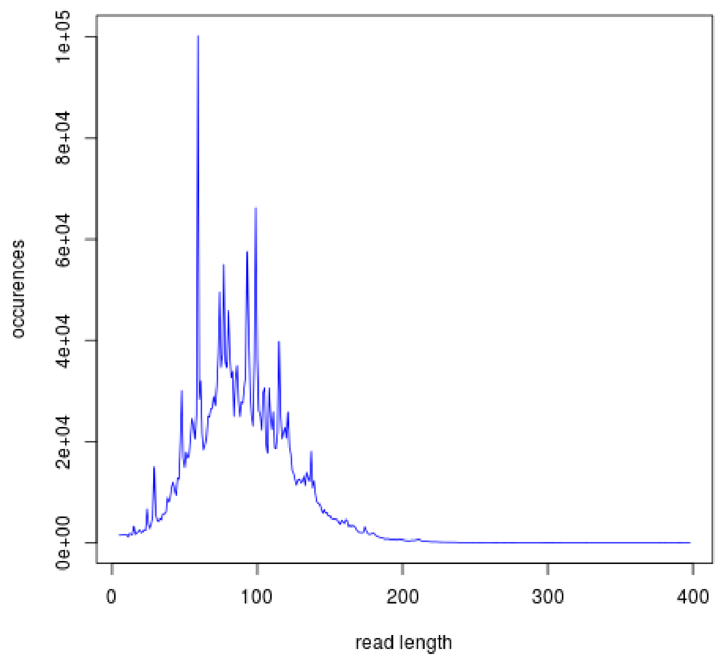


Figure 1: Example graphic on the length distribution in a fastq file

2 Quality control

To quality control 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 but feel free to play around yourself with the interactive version of FastQC.

So, to run FastQC on your file, simply type:

```
fastqc FILE.fastq
```

The output 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).

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

Get familiar with the output. Does the sequence length-distribution meet your expectations (400bp library)? You can find here guidance on how to how to interpret the output of each module.

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).

When the program is installed, it can be used with

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

You 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`).

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. Nevertheless, try to trim them off anyway in order to check if there are still adapters left in your library - they can have negative effects on your further analyses.

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

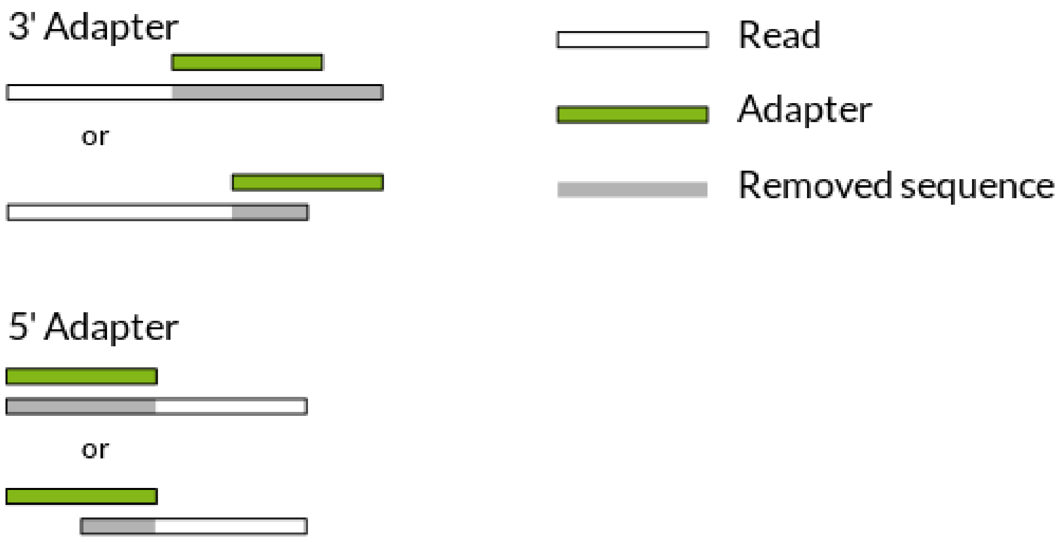


Figure 2: 3'- and 5'-adapter trimming (source)

The adapters used for Ion Torrent sequencing are shown in Fig. 3 and their orientation in the libraries is shown in Fig. 4.

```
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 3: Non-barcoded Ion-A adapter and -P1 adapter sequences. In each sequence, a "*" indicates a phosphorothioate bond, for protection from nucleases and to preserve the directionality of adapter ligation

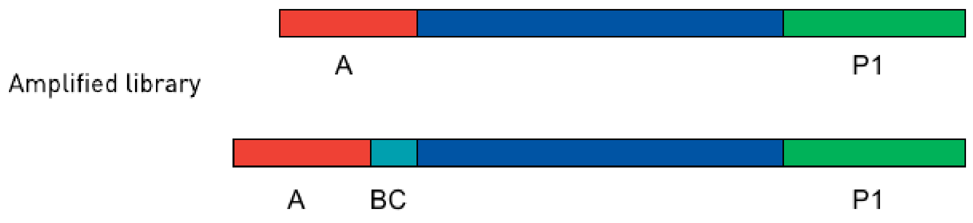


Figure 4: Ion adapters in the amplified library. BC is an optional barcode sequence.

.

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 also 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 now 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 FastWC outputs before and after trimming.

4 Fraction of duplicate reads

Duplicate reads (identical reads present more than once in the library) can skew genotype estimates and thus should be identified and removed before SNP calling. Duplicates can result from primer or PCR bias towards these reads and poor libraries can have levels of duplicates >50%.

At this step, we will calculate the fraction of duplicates but we will remove them only after *de novo* genome assembly and read mapping. The approach is based on the Simple fool's guide to population genomics via RNAseq and makes use of `fastx_collapser` from the FASTX-Toolkit and a python script (`fastqduplicatecounter.py`).

First, use `fastx_collapser` to combine and count all identical reads.

```
fastx_collapser -Q 33 -v -i INPUTFILE.fq -o OUTPUTFILE.txt
```

The `INPUTFILE` is your trimmed fastq file. `-Q 33` specifies that quality scores are Phred33 encoded. The `OUTPUTFILE` is used in the next step with the python script 'fastqduplicatecounter.py'.

```
fastqduplicatecounter.py OUTPUTFILE.txt OUTPUTFILE_header.txt > OUTPUTFILE_duplicatecount.txt
```

This script calculates the fractions of duplicate and singleton reads. Open the outputfile with `less OUTPUTFILE_duplicatecount.txt` and check the percentage of duplicate reads.

5 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 learn to 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`.

5.1 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

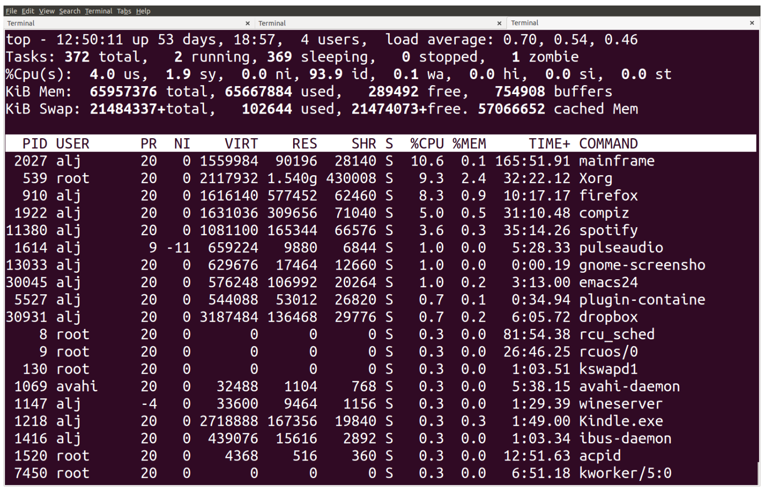
If the process would still run, you would get a line similar to:

```
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 5.

At the top of the screen, it lists the most CPU-intensive tasks on the system. 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 allows you also to cancel the process before it finishes. To cancel the process comes in handy when you figure out that you started it with wrong parameters or input files and



```
top - 12:50:11 up 53 days, 18:57,  4 users,  load average: 0.70, 0.54, 0.46
Tasks: 372 total,   2 running, 369 sleeping,   0 stopped,   1 zombie
%Cpu(s):  4.0 us,  1.9 sy,  0.0 ni, 93.9 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:  65957376 total, 65667884 used,   289492 free,   754908 buffers
KiB Swap: 21484337+total,   102644 used, 21474073+free. 57066652 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2027 alj       20   0 1559984  90196  28140 S  10.6  0.1 165:51.91 mainframe
  539 root      20   0 2117932 1.540g 430008 S   9.3  2.4  32:22.12 Xorg
  910 alj       20   0 1616140 577452  62460 S   8.3  0.9  10:17.17 firefox
 1922 alj       20   0 1631036 309656  71040 S   5.0  0.5  31:10.48 compiz
11380 alj       20   0 1081100 165344  66576 S   3.6  0.3  35:14.26 spotify
 1614 alj        9 -11  659224   9880   6844 S   1.0  0.0   5:28.33 pulseaudio
13033 alj       20   0  629676  17464  12660 S   1.0  0.0   0:00.19 gnome-screensho
30045 alj       20   0  576248 106992  20264 S   1.0  0.2   3:13.00 emacs24
 5527 alj       20   0  544088  53012  26820 S   0.7  0.1   0:34.94 plugin-containe
30931 alj       20   0 3187484 136468  29776 S   0.7  0.2   6:05.72 dropbox
    8 root      20   0       0      0      0 S   0.3  0.0  81:54.38 rcu_sched
    9 root      20   0       0      0      0 S   0.3  0.0  26:46.25 rcuos/0
  130 root      20   0       0      0      0 S   0.3  0.0   1:03.51 kswapd1
 1069 avahi     20   0   32488   1104    768 S   0.3  0.0   5:38.15 avahi-daemon
 1147 alj       -4   0   33600   9464   1156 S   0.3  0.0   1:29.39 wineserver
 1218 alj       20   0 2718888 167356  19840 S   0.3  0.3   1:49.00 Kindle.exe
 1416 alj       20   0  439076  15616   2892 S   0.3  0.0   1:03.34 ibus-daemon
 1520 root      20   0    4368    516    360 S   0.3  0.0  12:51.63 acpid
 7450 root      20   0       0      0      0 S   0.3  0.0   6:51.18 kworker/5:0
```

Figure 5: Screenshot of the `top` tool output

you want to re-start it with different settings. The `kill` command allows you to cancel a specific project.

```
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 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.

5.2 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 redirecting-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 &
```