Assembly: read QC and trimming

Conventions in this document

This is normal text

This is text describing a unix command, e.g. `grep` - the command will then be in italics.

```
This is a command you need to enter on the command line
```

```
This command has one word in bold that you need to change
```

For example, this might be the name of the folder that will contain the output of the command

Below is a table where you can fill in the answers to the question(s)

Question 1	your answer here
Question 2	your answer here

Working area

The area to do your 'work' and save your files is here:

```
/home/yourusername
```

Where is what

We will start the practical with two files we will use for the assembly module tomorrow:

```
/data/assembly/illumina/MiSeq_50x_R1.fastq
/data/assembly/illumina/MiSeq_50x_R2.fastq
```

They represent around 50 x coverage of E coli K12 (strain MG1655) randomly sampled from a MiSeq 2x150 bp PE (paired end) run

Part 1: Understanding reads, QC of sequence data

Learning points:

- Understand read file formats
- How to prepare and judge a QC report

A peak into the fastq files

Fastq files are very big. In order to be able to view them in a 'page-by-page' way, we will use the `less` command:

```
less /data/assembly/illumina/MiSeq_50x_R1.fastq
```

This file contains the 'read 1' dataset of a MiSeq run for the sample. Use the space bar to browse through the file. Use '`q`' to go out of the `less` program. Make sure you recognize the fastq format, if needed use the slides from the presentation.

Repeat this for the read 2 file:

```
less /data/assembly/illumina/MiSeq_50x_R2.fastq
```

Do you see whether the reads in the same order in both files?

Quality control of Illumina reads

We will be using a program called **FastQC**. The program is available with a graphical user interface, or as a command-line only version. We will use the latter one. It takes a single fastq file (the file can be compressed) as input, and produces a web page (html file) with the results of a number of analyses.

Program	**Options**	**Explanation**
fastqc		Quality control of sequence data
	-o foldername	tells the program to place the output in a folder called **foldername** instead of in the same folder as the input file
	fastq file	file to be analysed by the program

Before we run the program, let's create a new folder for the output. Do this in your home folder:

```
cd  ~
mkdir qc_filter
cd qc_filter
pwd
```

To run fastqc on the first MiSeq file, run the command below; **`yourusername`** should be the name you used for your folder in /work. Note that the command should be written on a *single line*.

```
fastqc -o /home/yourusername/qc_filter/
  /data/assembly/illumina/MiSeq_50x_R1.fastq
```

The program will tell you how far it has come, and should finish in a minute or so. Check that it finished without error messages.

In the folder you specified after `-o`, you should now see a new file called `MiSeq_50x_R1_fastqc.zip`, and a folder by the same name. The *folder* contains the following:

`Icons` → folder
`Images` → folder
`fastqc_data.txt` → file
`fastqc_report.html` → file
`summary.txt` → file

Now, open a webbrowser, and, using the menu option 'Open file', locate the html file, choose

`yourusername` → `qc_filter` → `MiSeq_50x_R1_fastqc` → `fastqc_report.html`

Open the file called `fastqc_report.html`. Alternatively, you could browse the file system and double-click on the file.

Study the results.
The plot called “Per base sequence quality” shows an overview of the range of quality scores across all based at each position in the fastq file. The y-axis shows quality scores and the x-axis shows the read position. For each read position, a boxplot is used to show the distribution of quality scores for all reads. The yellow boxes represent quality scores within the inter-quartile range (25% - 75%). The upper and lower whiskers represent 10% and 90% point. The central red line shows the median of the quality values and the blue line shows the mean of the quality values.

A rule of thumb is that a quality score of 30 indicates a 1 in 1000 probability of error and a quality score of 20 indicates a 1 in 100 probability of error (see the wikipedia page on the fastq format at http://en.wikipedia.org/wiki/Fastq). The higher the score the better the base call. You will see from the plots that the quality of the base calling deteriorates along the read (as is always the case with Illumina sequencing). A sometimes used minimum requirement for Per Base Sequence Quality is that the first 36 bases should have a median and mean quality score over 20.

Now, answer these questions:

Question	**Your answer**
How many reads were there in total in the `MiSeq_50x_R1.fastq` file ?	
How many bases were there in total in the file (trick question.)?	
Which part(s) of the reads would you say are of low quality - if any?	
Would the run have passed the minimum requirement for Per Base Sequence Quality?	

Repeat the fastqc analysis for the file

```
/data/assembly/illumina/MiSeq_50x_R2.fastq
```

Open the `fastqc_report.html` in your webbrowser.

Question	**Your answer**
How many reads were there in total in the `fastqc_report.html.fastq` file? Is this surprising?	
Are there part(s) of the reads that have a lower quality compared to the `MiSeq_50x_R1.fastq` file?	
Would the run have passed the minimum requirement for Per Base Sequence Quality?	

NB. You can get more information about the use of the fastqc program by writing

```
fastqc -h
```

Part 2: Filtering and trimming

We want to do the following with the raw reads:

- reads may contain adaptor sequences left from library preparation, these we want to remove
- remove bad-quality parts of reads
- remove the end of reads that are given the quality score 2 (encoded in the fastq file with a 'B'), indicating these bases should not be used for further analysis. See http://en.wikipedia.org/wiki/Fastq#Encoding

To see these trailing ‘B’s, do the following.

Type

```
less /data/assembly/illumina/MiSeq_50x_R1.fastq
```

Now, type the forward slash ‘/’.This allows you to do text search in the file you are viewing. Type

```
BBBBB$
```

That is five B’s followed by a dollar sign ($), which means ‘look at the end of a line only’. The first string of five (or more) B’s is indicated. Type ‘N’ to see more of these. Type q to get out of less.

For the actual filtering, we are going to use the **cutadapt** program for adaptor removal and filtering. The following commands will be used, with the options we will give to them:

Program	**Options**	**Explanation**
cutadapt		Reads a FASTA or FASTQ file, finds and removes adapters, and writes the changed sequence to standard output.
	-a	the sequence following this option is the adaptor to remove
	-q 20	quality value cutoff
	-m 20	minimum read length after trimming, otherwise discard the read
	-O 10	Minimum overlap length. If the overlap between the read and the adapter is shorter than 10 bases, the read is not modified

NOTE that we will need to give the adaptor sequence twice: once for the forward, one for the reverse complement sequence

In addition, we will use a script written by your teacher:

Program	Options	Explanation
fastq_trim_trailing_Bs.pl	none	Removal of sequences at the end with quality score 2 ('B')

Piping, saving to file

Those who are familiar with unix can skip this section.

One could run each of these programs individually and save the results in intermediate files, to be used as input for the next step. However, it is much more practical to have the output of one command used directly by the next (without saving to disk). This is possible using the unix 'pipe'. The pipe symbol '`|`' allows for sending the output from one command to the next, thereby allowing for stringing commands together. For example, using the `cat` command that list the contents of the file(s) mentioned after it, we can send (pipe) it's output to the `less` command, which allows for page-by-page viewing:

```
cat file.txt | less
```

Note that a shorter way to do this would be

```
less file.txt
```

Using the pipe symbol, we can be creating very long command lines, involving many commands and pipes between them to do the filtering:

```
command1 -option1 -option2 | command2 -option1 |
    command3 -option1 -option2 | command4 -option1
    -option2 | command5 -option1 -option2
```

Another trick we are going to us is to split up this long command over several lines to make it much more readable. This can be done in unix by typing a space, the backslash symbol '`\`' and pressing the return key:

```
command1 -option1 -option2 | \
command2 -option1 -option2 | \
command3 -option1 -option2 | \
command4 -option1 -option2 | \
command5 -option1 -option2
```

One final thing to know is that many programs send their output to the screen (technically, to 'standard output') for you to read. You can save this output to a file instead by writing the '>' symbol (called 'redirect') and a filename, e.g.

```
command 1 | command 2 >output.txt
```

Tab completion

One of the most useful short-cuts to know when using command-line is *tab completion*. When typing in the name of a command or file, you can hit the tab key part-way through to try and get the system to finish it off for you. When it is unambiguous what you are trying to do, the system will auto complete for you. Otherwise it will partially complete. You can hit tab twice to see a list of all the available options. Another useful trick is the up-arrow, which lets you find commands you have previously entered and re-run them. This goes in hand with the `history` command which shows you all of the commands you have entered recently.

Trimming and filtering - the command

This is the command we will use:

```
cutadapt -a GATCGGAAGAGCGGTTCAGCAGGAATGCCGAG \
     -a CTCGGCATTCCTGCTGAACCGCTCTTCCGATC \
     -q 20 -m 20 -O 10 \
     /data/assembly/illumina/MiSeq_50x_R1.fastq \
     | fastq_trim_trailing_Bs.pl
>MiSeq_50x_R1_trimmed.fastq
```

This process will take around a minute.

The cutadapt command will report a few metrics.

Question	Your answer
How many reads were removed from `MiSeq_50x_R1.fastq` during filtering/trimming? What percentage of the total was this?	
And how many bases and % of the total?	

Now, run the same filtering on the file `MiSeq_50x_R2.fastq`, saving the results to `MiSeq_50x_R2_trimmed.fastq`

Question	Your answer
How many reads were removed from `MiSeq_50x_R2.fastq` during filtering/trimming? What percentage of the total was this?	
And how many bases and % of the total?	

Again, to learn more about the cutadapt program, run it as follows:

```
cutadapt -h
```

Redoing the fastqc on the filtered reads

Run fastqc on the filtered files (as before), and compare the results with the those from the unfiltered reads.

Question	Your answer
When comparing the “Per base sequence quality” graphs, what is the biggest difference between the filtered and unfiltered data sets?	
Explain the difference in the “Sequence length distribution” plots between the filtered and unfiltered data sets?	
What did we achieve by filtering this way?	

You could consider running fastQC on other fastq files from the course (in the /data folder).