1 Introduction Linux

Linux is an open-source operating system (OS) developed based on the kernel created by Linus Torvalds. In the last two decades, Linux has gained so much popularity and now it is used on many platforms. Nowadays, most of the high-end servers to mobile phones (Android OS or iOS) run on different variants of Linux.

Generally, Linux computers/servers are installed for multi-user usage. However, in this bioinformatics course, we will be working on a local Linux machine. All the commands what you learn here can be used in any distribution (i.e. Ubuntu, CentOS, Debian, SuSE, etc) of Linux operating system.

1.1 The Terminal

We use terminal (AKA command line interface) to interact with the operating system. The terminal by default runs one of the "shells". Shell is a program that sits between the user and the kernel and translates user commands into machine code. The advantages of using command line are greater control and flexibility over the system or software. Moreover, multiple commands can be saved in a file and executed as a script.

The most common shells are:

- Bourne Shell
- Bourne Again Shell (variant is Z Shell)
- C Shell (variant is T Shell)
- K Shell

Among these Bourne Again Shell (BASH) is the most popular one. This is the default shell on the system and we will be using it throughout our course.

1.2 Connecting to Linux Server

If you are going to work on a remote server use "ssh" command to connect to it.

```
ssh user@ServerName
ssh user@ipaddress
```

There are other commands to connect to the server (telnet, rsh etc...) which are unsecure and outdated. However, in this course we will be using locally installed Linux system. Please use provided username and password to enter into your account and open "Terminal".

1.3 Linux command structure

When you open a terminal, you will see command prompt ready to take commands. The default location on the terminal is your "home directory". It is represented with ~ (tilde) symbol.

All Linux commands are single words (can be alpha-numeric), with optional parameters followed by arguments. For historical reasons, some of the early commands are only two letter long and case sensitive. Most of the command options (also called flags) are single letters. They should be specified after the command before giving any input.

```
ls -l ~/Training

drwxr-xr-x  5 vatt01s  staff  160 24 Feb 19:03 Data
drwxr-xr-x  5 vatt01s  staff  160 24 Feb 20:27 exFiles
```

Here "ls" is the command to list the contents of the directory, "-l" is the option for long listing and "~/Training" is the input, which is optional in this case. Without the input, "ls" shows the contents of the current directory.

Please remember:

- Linux commands are case sensitive
- Single words
- Options have to follow the command
- Options can start with a single hyphen and a character or a double hyphen and a word
- Single character options can be combined
- Argument can be one or multiple inputs (ls -l Documents Desktop)

- You can write more than one command separating with a semicolon;

You can use "tab" to auto-fill the command.

1.4 First Commands

`ls`
(You might see a different output than shown below...)

```
 Lists information about the files/directories. Default is current directory.
Sorts entries alphabetically.

Commonly used options:
-l long list
-a show all files (including hidden files)
-t sort based on last modified time

Ex:
ls -l

drwxrwxr-x 31 training training   4096 Dec 19 13:15 anaconda2
drwxrwxr-x  5 training training   4096 Jan 18 14:42 bamtools-master
-rw-rw-r--  1 training training 669825 Jan 18 13:02 bamtools-master.zip
drwxrwxr-x  3 training training   4096 Jan 29 14:22 bin
drwxr-xr-x  2 training training   4096 Dec 18 20:19 Desktop
drwxr-xr-x  2 training training   4096 Dec 18 20:19 Documents
```

Here Left to right columns are:

- File permissions
- Number of links
- Owner's name
- Group's name
- Number of bytes
- Last modified time
- File/Directory name

`pwd`

```
 Will return current directory's name

Ex:
pwd

/home/training
```

`cd`

```
 Change directory.
It is used for changing the working directory

Ex:
cd Documents
```

We are now in "Documents" directory. Command "cd .." takes you out from the current directory. Entering "cd" command will bring you to home directory.

`mkdir`

```
 make directory
This command creates a directory if no file/directory exists with that name.

Ex:
mkdir Practice
ls -l

drwxrwxr-x  2 training training   4096 Jan 30 13:46 Practice
```

```
drwxrwxr-x 10 training training   4096 Jan 29 16:37 Training
drwxrwxr-x  3 training training   4096 Jan 29 14:22 bin
drwxrwxr-x 13 training training   4096 Jan 29 14:22 Programs
drwxrwxr-x  7 training training   4096 Jan 25 13:35 glue_dependencies
```

rmdir

```
 remove directory
This command removed an empty directory.

Ex:
rmdir Practice
ls -l

drwxrwxr-x 10 training training   4096 Jan 29 16:37 Training
drwxrwxr-x  3 training training   4096 Jan 29 14:22 bin
drwxrwxr-x 13 training training   4096 Jan 29 14:22 Programs
drwxrwxr-x  7 training training   4096 Jan 25 13:35 glue_dependencies
drwxr-xr-x  3 training training   4096 Jan 21 09:53 Downloads
drwxrwxr-x  5 training training   4096 Jan 18 14:42 bamtools-master
drwxr-xr-x  7 training training   4096 Aug  3  2017 gluetools
```

touch

```
 It is file's timestamp changing command. However, it can be used for
creating an empty file. This command is generally used for checking
write permission for the user.

Ex:
touch temp-file
ls -l

drwxrwxr-x 31 training training   4096 Dec 19 13:15 anaconda2
drwxrwxr-x  5 training training   4096 Jan 18 14:42 bamtools-master
drwxr-xr-x  7 training training   4096 Aug  3  2017 gluetools
drwxrwxr-x  3 training training   4096 Jan  9 09:58 Library
lrwxrwxrwx  1 training training     37 Jan  7 17:19 phyloscanner
drwxrwxr-x  2 training training   4096 Jan 30 13:46 Practice
drwxrwxr-x 13 training training   4096 Jan 29 14:22 Programs
-rw-rw-r--  1 training training      0 Jan 30 14:01 temp-file
drwxrwxr-x 10 training training   4096 Jan 29 16:37 Training
```

rm

```
 remove
rm is used for removing files and directories.

Ex:
rm temp-file
ls -l

drwxrwxr-x 31 training training   4096 Dec 19 13:15 anaconda2
drwxrwxr-x  5 training training   4096 Jan 18 14:42 bamtools-master
-rw-rw-r--  1 training training 669825 Jan 18 13:02 bamtools-master.zip
drwxrwxr-x  3 training training   4096 Jan 29 14:22 bin
drwxr-xr-x  2 training training   4096 Dec 18 20:19 Desktop
drwxrwxr-x  3 training training   4096 Jan  9 09:58 Library
lrwxrwxrwx  1 training training     37 Jan  7 17:19 phyloscanner
drwxrwxr-x  2 training training   4096 Jan 30 13:46 Practice
drwxrwxr-x 13 training training   4096 Jan 29 14:22 Programs
drwxrwxr-x 11 training training   4096 Jan 30 15:23 Training
```

to remove directories use “-r” option. Please remember once a file or directory is deleted, it will not go to “Recycle bin” in Linux and there is no way you can recover it.

`cp`

```
 copy files or directories

Ex:
touch temp1
cp temp1 temp2
ls -l

drwxrwxr-x 31 training training   4096 Dec 19 13:15 anaconda2
drwxrwxr-x  5 training training   4096 Jan 18 14:42 bamtools-master
-rw-rw-r--  1 training training 669825 Jan 18 13:02 bamtools-master.zip
drwxrwxr-x  3 training training   4096 Jan 29 14:22 bin
drwxr-xr-x  7 training training   4096 Aug  3  2017 gluetools
drwxrwxr-x  3 training training   4096 Jan  9 09:58 Library
lrwxrwxrwx  1 training training     37 Jan  7 17:19 phyloscanner
drwxrwxr-x  2 training training   4096 Jan 30 13:46 Practice
drwxrwxr-x 13 training training   4096 Jan 29 14:22 Programs
-rw-rw-r--  1 training training      0 Jan 30 15:34 temp1
-rw-rw-r--  1 training training      0 Jan 30 15:34 temp2
drwxrwxr-x 11 training training   4096 Jan 30 15:23 Training
```

To copy directories, use “-r” option

mv

```
 to move file or directory
to rename

Ex:
mv temp1 Desktop/.
mv temp2 NewfileName
ls -l

drwxrwxr-x 31 training training   4096 Dec 19 13:15 anaconda2
drwxrwxr-x  5 training training   4096 Jan 18 14:42 bamtools-master
-rw-rw-r--  1 training training 669825 Jan 18 13:02 bamtools-master.zip
drwxrwxr-x  3 training training   4096 Jan  9 09:58 Library
-rw-rw-r--  1 training training      0 Jan 30 15:34 NewfileName
lrwxrwxrwx  1 training training     37 Jan  7 17:19 phyloscanner
drwxrwxr-x  2 training training   4096 Jan 30 13:46 Practice
drwxrwxr-x 13 training training   4096 Jan 29 14:22 Programs
drwxrwxr-x 11 training training   4096 Jan 30 15:23 Training
```

In the first example "temp1" is move to Desktop directory. The "." at the end will retain the file name (name is unchanged). In the second example temp2 is renamed to NewfileName.

ln

```
 link
to make links to files/directories.

Ex:
ln -s ~/Training/exFiles/Ebola-1.fa .
ls -l

drwxr-xr-x  2 training training   4096 Dec 18 20:19 Documents
drwxr-xr-x  3 training training   4096 Jan 21 09:53 Downloads
lrwxrwxrwx  1 training training     24 Jan 31 10:17 Ebola-1.fa -> ~/Training/exFiles/Ebola-1.fa
drwxrwxr-x  7 training training   4096 Jan 25 13:35 glue_dependencies
drwxr-xr-x  7 training training   4096 Aug  3  2017 gluetools
drwxrwxr-x  3 training training   4096 Jan  9 09:58 Library
-rw-rw-r--  1 training training      0 Jan 30 15:34 NewfileName
lrwxrwxrwx  1 training training     37 Jan  7 17:19 phyloscanner
drwxrwxr-x  2 training training   4096 Jan 30 13:46 Practice
drwxrwxr-x 13 training training   4096 Jan 29 14:22 Programs
drwxrwxr-x 10 training training   4096 Jan 30 15:55 Training
```

We encourage you to create links instead of copying data into various directories to save space.

1.5 File viewers

cat

```
 concatenate
combines files and prints on the screen

Ex:
cat ~/Training/exFiles/Ebola-1.fa

>gi|10313991|ref|NC_002549.1| Zaire ebolavirus isolate
CGGACACACAAAAAGAAAGAAGAATTTTTAGGATCTTTTGTGTGCGAATAACTATGAGGAAGATTAATAA
TTTTCCTCTCATTGAAATTTATATCGGAATTTAAATTGAAATTGTTACTGTAATCACACCTGGTTTGTTT
CAGAGCCACATCACAAAGATAGAGAACAACCTAGGTCTCCGAAGGGAGCAAGGGCATCAGTGTGCTCAGT
TGAAAATCCCTTGTCAACACCTAGGTCTTATCACATCACAAGTTCCACCTCAGACTCTGCAGGGTGATCC
AACAACCTTAATAGAAACATTATTGTTAAAGGACAGCATTAGTTCACAGTCAAACAAGCAAGATTGAGAA
TTAACCTTGGTTTTGAACTTGAACACTTAGGGGATTGAAGATTCAACAACCCTAAAGCTTGGGGTAAAAC
ATTGGAAATAGTTAAAAGACAAATTGCTCGGAATCACAAAATTCCGAGTATGGATTCTCGTCCTCAGAAA
ATCTGGATGGCGCCGAGTCTCACTGAATCTGACATGGATTACCACAAGATCTTGACAGCAGGTCTGTCCG
TTCAACAGGGGATTGTTCGGCAAAGAGTCATCCCAGTGTATCAAGTAAACAATCTTGAAGAAATTTGCCA
.............
```

pg/more/less

```
 these commands are used for viewing the files.

Ex:
pg ~/Training/exFiles/Ebola-1.fa

>gi|10313991|ref|NC_002549.1| Zaire ebolavirus isolate
CGGACACACAAAAAGAAAGAAGAATTTTTAGGATCTTTTGTGTGCGAATAACTATGAGGAAGATTAATAA
TTTTCCTCTCATTGAAATTTATATCGGAATTTAAATTGAAATTGTTACTGTAATCACACCTGGTTTGTTT
CAGAGCCACATCACAAAGATAGAGAACAACCTAGGTCTCCGAAGGGAGCAAGGGCATCAGTGTGCTCAGT
TGAAAATCCCTTGTCAACACCTAGGTCTTATCACATCACAAGTTCCACCTCAGACTCTGCAGGGTGATCC
AACAACCTTAATAGAAACATTATTGTTAAAGGACAGCATTAGTTCACAGTCAAACAAGCAAGATTGAGAA
TTAACCTTGGTTTTGAACTTGAACACTTAGGGGATTGAAGATTCAACAACCCTAAAGCTTGGGGTAAAAC
ATTGGAAATAGTTAAAAGACAAATTGCTCGGAATCACAAAATTCCGAGTATGGATTCTCGTCCTCAGAAA
ATCTGGATGGCGCCGAGTCTCACTGAATCTGACATGGATTACCACAAGATCTTGACAGCAGGTCTGTCCG
TTCAACAGGGGATTGTTCGGCAAAGAGTCATCCCAGTGTATCAAGTAAACAATCTTGAAGAAATTTGCCA
.............
```

Press “q” to come out from the program

head/tail

```
 these commands show first and last 10 lines respectively from a file

Ex:
head ~/Training/exFiles/Ebola-1.fa

>gi|10313991|ref|NC_002549.1| Zaire ebolavirus isolate
CGGACACACAAAAAGAAAGAAGAATTTTTAGGATCTTTTGTGTGCGAATAACTATGAGGAAGATTAATAA
TTTTCCTCTCATTGAAATTTATATCGGAATTTAAATTGAAATTGTTACTGTAATCACACCTGGTTTGTTT
CAGAGCCACATCACAAAGATAGAGAACAACCTAGGTCTCCGAAGGGAGCAAGGGCATCAGTGTGCTCAGT
TGAAAATCCCTTGTCAACACCTAGGTCTTATCACATCACAAGTTCCACCTCAGACTCTGCAGGGTGATCC
AACAACCTTAATAGAAACATTATTGTTAAAGGACAGCATTAGTTCACAGTCAAACAAGCAAGATTGAGAA
TTAACCTTGGTTTTGAACTTGAACACTTAGGGGATTGAAGATTCAACAACCCTAAAGCTTGGGGTAAAAC
ATTGGAAATAGTTAAAAGACAAATTGCTCGGAATCACAAAATTCCGAGTATGGATTCTCGTCCTCAGAAA
ATCTGGATGGCGCCGAGTCTCACTGAATCTGACATGGATTACCACAAGATCTTGACAGCAGGTCTGTCCG
TTCAACAGGGGATTGTTCGGCAAAGAGTCATCCCAGTGTATCAAGTAAACAATCTTGAAGAAATTTGCCA
..............
```

1.6 File editors

File viewers show the content of the file without making any changes. To change the file content you have to use file editors. There are many non-graphical text editors like ed, emacs, vi and nano are available on most of the Linux distributions. Some of them are very sophisticated (ex. vi) and for advanced users. Here we will be learning about a non-graphical file editor, "nano".

Nano (earlier called pico) is very much like any graphical editor without a mouse. All commands are executed through the use of the keyboard, using the <CTRL> key modifier. It can be used to edit virtually any kind of text file from the command line

Nano without a filename gives you a standard (blank) nano window.

At the bottom of the screen, there are commands with a symbol in front. The symbol tells that you need to hold down the Control (Ctrl) key, and then press the corresponding letter of the command you wish to use.

For Example:

Ctrl+X will exit nano and return you to the command line.

Nano Quick Reference

- Ctrl+X: Exit the editor. If you've edited text without saving, you'll be prompted as to whether you really want to exit.

- Ctrl+O: Write (output) the current contents of the text buffer to a file. A filename prompt will appear; press Ctrl+T to open the file navigator shown above.
- Ctrl+R: Read a text file into the current editing session. At the filename prompt, hit Ctrl+T: for the file navigator.
- Ctrl+K: Cut a line into the clipboard. You can press this repeatedly to copy multiple lines, which are then stored as one chunk.
- Ctrl+J: Justify (fill out) a paragraph of text. By default, this reflows text to match the width of the editing window.
- Ctrl+U: Uncut text, or rather, paste it from the clipboard. Note that after a Justify operation, this turns into unjustify.
- Ctrl+T: Check spelling.
- Ctrl+W: Find a word or phrase. At the prompt, use the cursor keys to go through previous search terms, or hit Ctrl+R to move into replace mode. Alternatively you can hit Ctrl+T to go to a specific line.
- Ctrl+C: Show current line number and file information.
- Ctrl+G: Get help; this provides information on navigating through files and common keyboard commands

Getting help in Linux
Most of the Linux commands have manual pages. To access them, use "man" or "info" command. The manual page gives a detailed explanation of the command, all available options and sometimes, also provides examples. For example to get the manual page for ls command type "man ls"

```
 LS(1)               User Commands                                LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the  FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor
       --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
```

```
               do not list implied . and ..

       --author
               with -l, print the author of each file
```

Please explore manual pages of all the above commands for available options.

1.7 Commands for text processing

`cut`

```
 The cut command is a command line utility to cut a section from a file.
Please see "man cut" for available option.

To cut a section of file use "-c" (characters)

Ex:
cut -c 1-10  ~/Training/exFiles/Ebola-1.fa

 >gi|103139
 CGGACACACA
 TTTTCCTCTC
 CAGAGCCACA
 TGAAAATCCC
 AACAACCTTA
 TTAACCTTGG
 ATTGGAAATA
 ATCTGGATGG
 TTCAACAGGG
```

The option "-c 1-10" will give you 1-10 characters from the input file.

Here are some of the useful options:

- -c: cut based on character position
- -d: cut based on delimiter
- -f: field number

We have a file named “viruses.txt” with all the virus names, genbank ids and genome length etc. These fields are separated by “|” symbol.

```
 head ~/Training/exFiles/viruses.txt
gi|167006425|ref|NC_010314.1| Abaca bunchy top virus DNA-N, complete genome|1090
gi|167006427|ref|NC_010315.1| Abaca bunchy top virus segment 2, complete sequence|1057
gi|167006428|ref|NC_010316.1| Abaca bunchy top virus DNA-S, complete genome|1087
gi|167006430|ref|NC_010317.1| Abaca bunchy top virus DNA-M, complete genome|1074
gi|167006432|ref|NC_010318.1| Abaca bunchy top virus DNA-C, complete sequence|1015
gi|167006434|ref|NC_010319.1| Abaca bunchy top virus DNA-R, complete genome|1099
```

To get only genbank id

```
 cut -d "|" -f2 ~/Training/exFiles/viruses.txt

167006425
167006427
167006428
167006430
167006432
```

Here “|” is a delimiter and second field is our choice of interest. Next try printing virus name (5th field) and it’s length (6th field)

`sort`

It is used for sorting input content

Some options are:

- -t: field separator
- -n: numeric sort
- -k: sort with a key (field)
- -r: reverse sort
- -u: print unique entries

to sort viruses.txt based on their length.

```
sort -t "|" -nk6 ~/Training/exFiles/viruses.txt

gi|10518489|ref|NC_002566.1| Trichoplusia ni cytoplasmic, complete sequence|200
gi|18450262|ref|NC_003380.1| Rice yellow mottle virus satellite, complete genome|220
gi|21450051|ref|NC_004033.1| Turnip crinkle virus satellite RNA, complete genome|230
gi|9627203|ref|NC_001546.1| Arabis mosaic virus small satellite RNA  genome|300
gi|401715661|ref|NC_018451.1| Arabis mosaic virus small satellite genome|301
gi|14329174|ref|NC_002802.1| Discula destructiva virus 1 RNA 4, complete genome|308
gi|20087066|ref|NC_003533.1| Cereal yellow dwarf virus-RPV satellite RNA |322
```

grep

---

searches input for a given pattern

Some options are:

- -A: after context
- -B: before context
- -C: before and after context
- -c: count
- -l: file with match
- -i: ignore case
- -o: only match
- -v: invert match
- -w: word match

to get all Hepatitis viruses from virus.txt

```
grep Hepatitis ~/Training/exFiles/viruses.txt

gi|9629718|ref|NC_001837.1| Hepatitis GB virus A, complete genome|9550
gi|9628705|ref|NC_001710.1| GB virus C/Hepatitis G virus, complete genome|9392
gi|9626732|ref|NC_001489.1| Hepatitis A virus, complete genome|7478
gi|21326584|ref|NC_003977.1| Hepatitis B virus, complete genome|3215
```

Other text processing commands worth looking at are: tr, rev, sed, uniq, wc and paste.

1.8 I/O control in Linux

When you run a command the output will be sent to standard out (stdout) ie. terminal. However, we can send the stdout to a file using ">" redirection. This will redirect the standard output to a file.

```
 ls > list
cat list

anaconda2
bamtools-master
bamtools-master.zip
bin
Desktop
Documents
Downloads
```

This will create a new file called list with all the file names in the directory. Remember, if there is a file exists with a name "list", its content will be overwritten by the stdout. Instead, we can append to a file using ">>" redirection.

Another kind of output that is generated by programs is standard error. We have to use "2>" to redirect it.

```
 ls /foo 2> error
```

To redirect stdout and stderr to a file use "&>"

1.9 Pipes

Piping in Linux is very powerful and efficient way to combine commands. Pipes (|) in Linux act as connecting links between commands. Pipe makes a previous commands output as next commands input. We can nest as many commands as we want using pipes. They play an important role in smooth running of the command flow and reducing the execution time.

To print 10 smallest viruses

```
 sort -t "|" -nk6 ~/Training/exFiles/viruses.txt  |head

gi|10518489|ref|NC_002566.1| Trichoplusia ni cytoplasmic polyhedrosis |200
gi|18450262|ref|NC_003380.1| Rice yellow mottle virus satellite, complete genome|220
```

```
gi|21450051|ref|NC_004033.1| Turnip crinkle virus satellite RNA, complete genome|230
gi|9627203|ref|NC_001546.1| Arabis mosaic virus small satellite RNA|300
gi|401715661|ref|NC_018451.1| Arabis mosaic virus small satellite complete genome|301
gi|14329174|ref|NC_002802.1| Discula destructiva virus 1 RNA 4, complete genome|308
gi|20087066|ref|NC_003533.1| Cereal yellow dwarf virus-RPV satellite RNA genome|322
gi|20522156|ref|NC_003798.1| Lucerne transient streak virus satellite RNA genome|324
gi|11119632|ref|NC_002602.2| Cucumber mosaic virus satellite RNA, complete genome|336
gi|55831391|ref|NC_006451.1| Turnip crinkle virus virulent satellite RNA C genome|355
```

We will be working on other examples during the course, where we use pipes to combine more than two commands.

1.10 Process control

Some commands take time to complete the job. For example if you want to compress a huge file with gzip command it might take few minutes to finish the job. Till it finishes you cannot run any command. In such situations, it is better to run this command in background by appending with "&" (we can also suspend a running job by Ctr-z and type bg).

```
gzip Training/Sreenu/temp-file &
```

Once it completes the task we get a message saying "Done" . We can get list of currently jobs in the terminal by "jobs" command. This will give you all the background jobs running in the current terminal. If you want to see all the running processes in the system, use "top". You can get user specific details in top using "-u" option.

```
top -u $USER

Tasks: 177 total,   1 running, 176 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.0 sy,  0.0 ni, 99.6 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  8173000 total,  3493840 free,   631640 used,  4047520 buff/cache
KiB Swap:  8386556 total,  8385224 free,     1332 used.  7152328 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 5770 training  20   0  292648   5704   4608 S   0.0  0.1   0:00.01 gvfs-udisks2-vo
 5780 training  20   0  264652   3552   3180 S   0.0  0.0   0:00.00 gvfs-goa-volume
 5784 training  20   0  266596   3404   2960 S   0.0  0.0   0:00.00 gvfs-mtp-volume
 5788 training  20   0  411104   9492   6416 S   0.0  0.1   0:00.00 gvfs-afc-volume
 5793 training  20   0  278796   4024   3496 S   0.0  0.0   0:00.00 gvfs-gphoto2-vo
 6414 training  20   0   96076   4844   3452 S   0.0  0.1   0:00.09 sshd
 6415 training  20   0   29860   5468   3388 S   0.0  0.1   0:00.24 bash
 7250 training  20   0   49124   3824   3136 R   0.0  0.0   0:00.00 top
 8670 training  20   0  183584   5424   5424 S   0.0  0.1   0:33.31 dirmngr
 9605 training  20   0  173820   2112   2020 S   0.0  0.0   0:27.25 gpg-agent
13141 training  20   0   68492   4076   3616 S   0.0  0.0   0:00.10 gconfd-2
13142 training  20   0  338012   3668   3212 S   0.0  0.0   0:00.00 at-spi-bus-laun
18980 training  20   0   62868   6544   5488 S   0.0  0.1   0:00.11 systemd
18985 training  20   0  171204   2324     36 S   0.0  0.0   0:00.00 (sd-pam)
18997 training  20   0   43212   4040   3492 S   0.0  0.0   0:00.21 dbus-daemon
19022 training  20   0  290600   6640   5944 S   0.0  0.1   0:00.01 gvfsd
19027 training  20   0  428872   9688   6784 S   0.0  0.1   0:00.00 gvfsd-fuse
19038 training  20   0  187556   4908   4296 S   0.0  0.1   0:00.00 dconf-servic
```

“top” command prints 12 column dynamic output. These columns are...

- PID: Process Id, this is a unique number used to identify the process.
- User : The username of whoever launched the process.
- PR : Priority: The priority of the process. Processes with higher priority will be favored by the kernel and given more CPU time than processes with lower priority. Oddly enough, the lower this value, the higher the actual priority; the highest priority on *nix is -20 and the lowest is 20.
- NI: nice value : nice is a way of setting your process’ priority.
- VIRT: Virtual Memory Size (KiB) : The total amount of virtual memory used by the process.
- RES: Resident Memory Size (KiB) : The non-swapped physical memory a task has used.
- SHR: Shared Memory Size (KiB) : The amount of shared memory available to a task, not all of which is typically resident. It simply reflects memory that could be potentially shared with other processes.

- S: Process Status : The status of the task which can be one of:
  - D = uninterruptible sleep
  - R = running
  - S = sleeping
  - T = traced or stopped
  - Z = zombie
- %CPU: CPU Usage : The percentage of your CPU that is being used by the process. By default, top displays this as a percentage of a single CPU. On multi-core systems, you can have percentages that are greater than 100%. For example, if 3 cores are at 60% use, top will show a CPU use of 180.
- %MEM: Memory Usage (RES) : A task's currently used share of available physical memory (RAM).
- TIME+: CPU Time, hundredths : Total CPU time the task has used since it started.
- COMMAND: Command Name or Command Line : To see the full command line that launched the process, start top with the -c flag

If you want to stop a running background job use "kill" command.

```
kill 1234
```

This will kill the job with process id 1234. As a user you can kill only your jobs. You do not have permission to run this command on other users' process ids.

Command line shortcuts

- Up/Down arrows: Previous commands
- !!: Reruns previous command
- Tab: Auto complete
- Tab+Tab: All available options
- Ctr+a: Move cursor to start of line
- Ctr+e: Move cursor to end of line
- Ctr+ : Alternates between terminals
- Ctr+l: Clear screen ((or Command+k on Mac)
- Ctr+c: Terminates the running program
- Ctr+z: Suspends the running program
- Ctr+w: Removes a previous word
- Ctr+d: Logout
- Ctr+d(in a command): Removes a character
- Ctr+u: Removes till the beginning

2 NGS file formats

The information of biological sequences (nucleotides and amino acids) is stored in various file formats for easy access and analysis. Some of these files are in human-readable (text) format and others are in machine-readable (binary) format.

2.1 AB1 files

First-generation DNA sequencers (i.e. Applied Biosystems) store sequencing information in three files: .ab1, .seq and .phd.1. AB1 files are raw (binary) output files. They are also called as "Chromatogram" or "electropherogram" files. .ab1 file stores the DNA sequence chromatogram and raw data along with some other information. .phd.1 file (Phred file) also called a qual file is a text file containing quality values for each base. ".seq" file is a text file stores sequence in FASTA format.

2.2 PHD.1 files

Phred file or quality file is a simple text file containing each base's Phred quality value (more about quality values in a later section) A typical qual files look like below

```
 >fileName
27 29 31 34 35 36 37 37 38 37 38 38 39 38 36 32 36 36
36 34 35 36 35 36 34 35 34 35 36 36 36 37 36 37 37 34
27 29 31 34 35 36 37 37 38 37 38 38 39 38 36 32 36 36
36 34 35 36 35 36 34 35 34 35 36 36 36 37 36 37 37 34
```

2.3 Fasta files

Fasta file is a text file, consists of a single or multiple sequences. Though it is not mandatory, fasta files generally have an extension of .fa, .fas, fsa, .fna or .fasta. Each sequence has a one-line header, starting with the '>', followed by the nucleotide (or amino acid) sequence. This sequence can be in a single line or span across multiple lines.

```
 >HQ613402.1
TTGAATTTATATCGATCG
ATCGCATCGCGAGCCGAT
CGAATCGCGCGCATCGAT
TAGCCCCGATCGATCGAT
```

2.4 Standard Flowgram Format: SFF

The Standard Flowgram Format(SFF) file is a binary file used to store pyrosequencing (454) data. It is equivalent to the ABI chromatogram files. It contains information about:

- flowgram
- called sequence
- qualities
- recommended quality
- adaptor clipping.

Like the AB1 files, these are binary files and require specialized programs to view or convert them to other formats.

2.5 Fastq files

As the technologies evolved to sequence millions of "short reads", storing the information in multiple files became difficult to manage and also consumes more disk space. To address these difficulties, a new file format, fastq (fast Q) was introduced in 2009. Now, fastq format is the de facto standard of high-throughput sequence file format for next-generation sequencers.
Fastq is a text file (human-readable). It has 4 lines for each sequence entry.

- Header: starts with"@"
- Sequence
- Optional field: starts with "+"
- Qualities

As the sequence and quality correspond to each other, the length of 2nd and 4th lines are always equal.

Depending on the kind of sequencing method, fastq files can be either single-end or paired-end. Single-end reads are sequenced from one direction of the template only, whereas paired-end reads are sequenced from both 5' and 3' ends of a template, giving rise to forward and reverse reads. Paired-end sequences may also have sequence identifiers in the headers ending with "/1" and "/2" for forward and reverse directional reads respectively.

Example fastq file

```
head ~/Training/Data/goodData-1.fq

@SRR1564803.1 1 length=101
CTGTCTGACTCCCAAGCATCAATACTCTGGCATTCTGAGTTTGATAAGCTTCGCTAACCTCTCGGCCTG...
+SRR1564803.1 1 length=101
CCCFFFFFHHHHHJJJJJJJJJJJJJJJJJJJJJJJJJJIIJJJJJJJJJJJJJJJJJJJJJJJIJIHH...
@SRR1564803.2 2 length=101
CCGGTGAGAAACACGCGACGCTGCACCTAAATGCATTTCGGGGAGAACCAGCTATCACGGAGTTTGAGC...
+SRR1564803.2 2 length=101
B@CFDFFFHHHHHJJJJJJJJJJIJJJJJJJJJJJIJJJJJJHFDDEEDDDDDDDDDDDDDDDDDDDD...
```

Most often, paired-end data stored in two files and they have "R1" and "R2" in their file names. The order of the reads in paired-end files corresponds to their pairing order, i.e. entry 1 in R1 file is sequenced from the same template as entry 1 in R2 file, and so on so forth. Therefore, the total number of sequences should always be equal in R1 and R2 files. If for any reason, any entry is removed from R1, its corresponding entry from R2 should also be removed.

Sometimes paired-end data is saved in interleaved fastq format, storing forward and reverse reads sequentially in a single file. A typical interleaved fastq file looks like below.

```
@B10941:62:000000000-A8CVL:1:2302:15112:1132 1:N:0:10
ATCCTAGGTAGCGACATGATGGCCCCAGGAGATGGGAAGAACCCCCGATAGAGATA
+
CCBCCCBBBGBGFGGGGGGGGG@;FFGGGEG@FF<EE<@FFC,CEGCCGGFF<FGF
@B10941:62:000000000-A8CVL:1:2302:15112:1132 2:N:0:10
TCGGCATGACGCTATCGCATCATTTTACTACGCTAGCATATCGCTTTTAGA
+
DDDCGCGFGGGGGGGGG@;FFGGGEG@FF<EE<@FFC,CEGCCGGFF<FGF
@B10941:63:000000000-A8CVL:1:2302:15112:1132 1:N:0:10
GCGAGGTCGGCATGATGGGGGAATACGAAGCATGGGAAGAAAAGGAGAGATACGAA
+
CCGGGGGGGGAAAACCCGGGGG@;FFGGGEG@FF<EE<@FFC,CEGCCGGGGFFFG
@B10941:63:000000000-A8CVL:1:2302:15112:1132 2:N:0:10
CGCGACGACTCGACGACATACGACACGCGATCGATCGGAAATGAATCGCATCGAC
+
CGGGGGFEGGCGCGAAGGGGG@;FFGGGEG@FF<EE<@FFC,CEAGGGACGGAGA
```

Generally a fastq file is stored with the suffix .fq or .fastq (if compressed, .fq.gz or .fastq.gz)

2.6 Hierarchical Data Formats 5: HDF5

The Hierarchical Data Format Version 5, (HDF5), file formats is a complex format developed to support large, complex and heterogeneous data. HDF5 uses a hierarchical structure to organize

data within the file in many different structured ways. It also allows for the embedding of metadata making it self-describing. Moreover, HDF5 is a compressed format. Oxford Nanopore Technologies raw output format FAST5 is a type of HDF5 file format.

2.7 Sequence Alignment/Map (SAM) format

The sequence alignment/map (SAM) format is a tab-delimited text file format developed to store read mapping. Every SAM file has two sections:

1) Header section (optional)
2) Alignment section.

Entries in the header section always start with "@" and come before alignment section. Each line in the header is tab-delimited and has a two letter header code called TAG. They follow "TAG:VALUE" format. These TAG are:

- HD – The Header Line – 1st line
- SO – Sorting order of alignments (unknown (default), unsorted, queryname and coordinate)
- SQ - Reference sequence dictionary.
- SN - Reference sequence name.
- LN - Reference sequence length.
- PG – Program
- ID – The program ID
- PN – The program name
- VN – Program version number
- CL – The Command actually used to create the SAM file
- RG – Read Group– "a set of reads that were all the product of a single sequencing run on one lane"

In the alignment section, there are 11 mandatory fields. These are

- QNAME: Read Name
- FLAG: Info on if the read is mapped, part of a pair, strand etc
- RNAME: Reference Sequence Name that the read aligns to
- POS: Leftmost position of where this alignment maps to the reference

- MAPQ: Mapping quality of read to reference (phred scale P that mapping is wrong)
- CIGAR: Compact Idiosyncratic Gapped Alignment Report
- RNEXT: Paired Mate Read Name
- PNEXT: Paired Mate Position
- TLEN: Template length/Insert Size (difference in outer co-ordinates of paired reads)
- SEQ: The actual read DNA sequence
- QUAL: ASCII Phred quality scores (+33)
- TAGS: Optional data – Lots of options e.g. MD=String for mismatches

Let us take a close look at FLAG and CIGAR values. FLAG is a sum of alignment bit flags. Below is the table showing what each bit corresponds to.

Bit	Description
0x1	template having multiple segments in sequencing
0x2	each segment properly aligned according to the aligner
0x4	segment unmapped
0x8	next segment in the template unmapped
0x10	SEQ being reverse complemented
0x20	SEQ of the next segment in the template being reversed
0x40	the first segment in the template
0x80	the last segment in the template
0x100	secondary alignment
0x200	not passing quality controls
0x400	PCR or optical duplicate

For example, Flag 99 codes for
64: first in the segment
16: read reverse complemented
8: second read is unmapped
2: each segment is properly aligned
1: paired end reads

This is the only combination that gives the total of 99

Concise Idiosyncratic Gapped Alignment Report: CIGAR
CIGAR is a string of alpha-numerics representing the alignment result. For example, take a look at below mapping results

```
Coor     12345678901234  5678901234567890123456789012345
ref      AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1        TTAGATAAAGGATA*CTG
+r002         aaaAGATAA*GGATA
+r003       gcctaAGCTAA
+r004                     ATAGCT..............TCAGC
-r003                           ttagctTAGGC
-r001/2                                      CAGCGCCAT
```

Corresponding SAM file will be

```
@HD VN:1.3 SO:coordinate
@SQ SN:ref LN:45
r001 163 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002   0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003   0 ref  9 30 5H6M       *  0   0 AGCTAA   *   NM:i:1
r004   0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003  16 ref 29 30 6H5M       *  0   0 TAGGC    *   NM:i:0
r001  83 ref 37 30 9M         =  7 -39 CAGCGCCAT         *
```

Where

```
Ref: AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT
R001/1:    TTAGATAAAGGATA*CTG
```

CIGAR values: 8M2I4M1D3M tells us there are

- 8 matches
- 2 insertions

- 4 matches
- 1 deletion
- 3 matches

Below is the list of allowed CIGAR letters and their descriptions

Op	BAM	Description
`M`	0	alignment match (can be a sequence match or mismatch)
`I`	1	insertion to the reference
`D`	2	deletion from the reference
`N`	3	skipped region from the reference
`S`	4	soft clipping (clipped sequences present in SEQ)
`H`	5	hard clipping (clipped sequences NOT present in SEQ)
`P`	6	padding (silent deletion from padded reference)
`=`	7	sequence match
`X`	8	sequence mismatch

2.8 BAM files

Binary Alignment Map (BAM) is a compressed SAM file. It is compressed using BGZF compression method..

2.9 CRAM files

CRAM files are also compressed SAM file, designed by the EBI to reduce the storage space. CRAM files compression is based on the reference the data is aligned to.

Data is compressed using one of the general purpose compressors (gzip, bzip2). CRAM records are compressed using several different encoding strategies. For example, bases are reference compressed by encoding base differences rather than storing the bases themselves. External reference sequences introduce the only external dependency into the CRAM format. When external reference sequences cannot be conveniently used the reference sequences also can be embedded within the CRAM files. However, when embedded reference sequences are used then only those reference sequence regions are preserved in CRAM that has reads aligned against them.

Further reading:
http://samtools.github.io/hts-specs/SAMv1.pdf
https://www.ebi.ac.uk/sites/ebi.ac.uk/files/groups/ena/documents/cram_format_1.0.1.pdf

3 Phred quality scores

The Phred quality score is a nucleotide identification accuracy measure. It is a probability of the base call is incorrect. The quality score calculation takes into account the ambiguity of the signal for the respective base as well as the quality of neighbouring bases. It is developed to help automate DNA sequencing and quality control of the data. In a fastq file Phred qualities are converted to ascii characters (q+33, printable ascii characters start from ascii 33) and stored.

It is calculated as
Phred-Q = -10 $\log_{10}$(P)

If the probability of an error is 0.01 (1 in 100 error), its phred score will be

= -10 X $\log_{10}$(0.01)
= -10 X -2 = 20

ascii character for 53 (20+33) is “5”

Here is the list of phred qualities and their error rates

Phred quality	Q33 code	Error rate	Phred quality	Q33 code	Error rate
0	!	1.00000	21	6	0.00794
1	”	0.79433	22	7	0.00631
2	#	0.63096	23	8	0.00501
3	$	0.50119	24	9	0.00398
4	%	0.39811	25	:	0.00316
5	&	0.31623	26	;	0.00251
6	’	0.25119	27	<	0.00200
7	(	0.19953	28	=	0.00158
8	)	0.15849	29	>	0.00126
9	*	0.12589	30	?	0.00100
10	+	0.10000	31	@	0.00079
11	,	0.07943	32	A	0.00063
12	-	0.06310	33	B	0.00050
13	.	0.05012	34	C	0.00040
14	/	0.03981	35	D	0.00032
15	0	0.03162	36	E	0.00025
16	1	0.02512	37	F	0.00020
17	2	0.01995	38	G	0.00016
18	3	0.01585	39	H	0.00013
19	4	0.01259	40	I	0.00010
20	5	0.01000			

4 NGS data quality control and read cleaning

Quality check of NGS data is a standard practice to do before analysing the data. NGS data is cleaned based on the reads' base qualities, length, and low complexity. Poor quality reads are untrustworthy and can lead to poor alignments, erroneous variant callings and incorrect interpretation of the data.

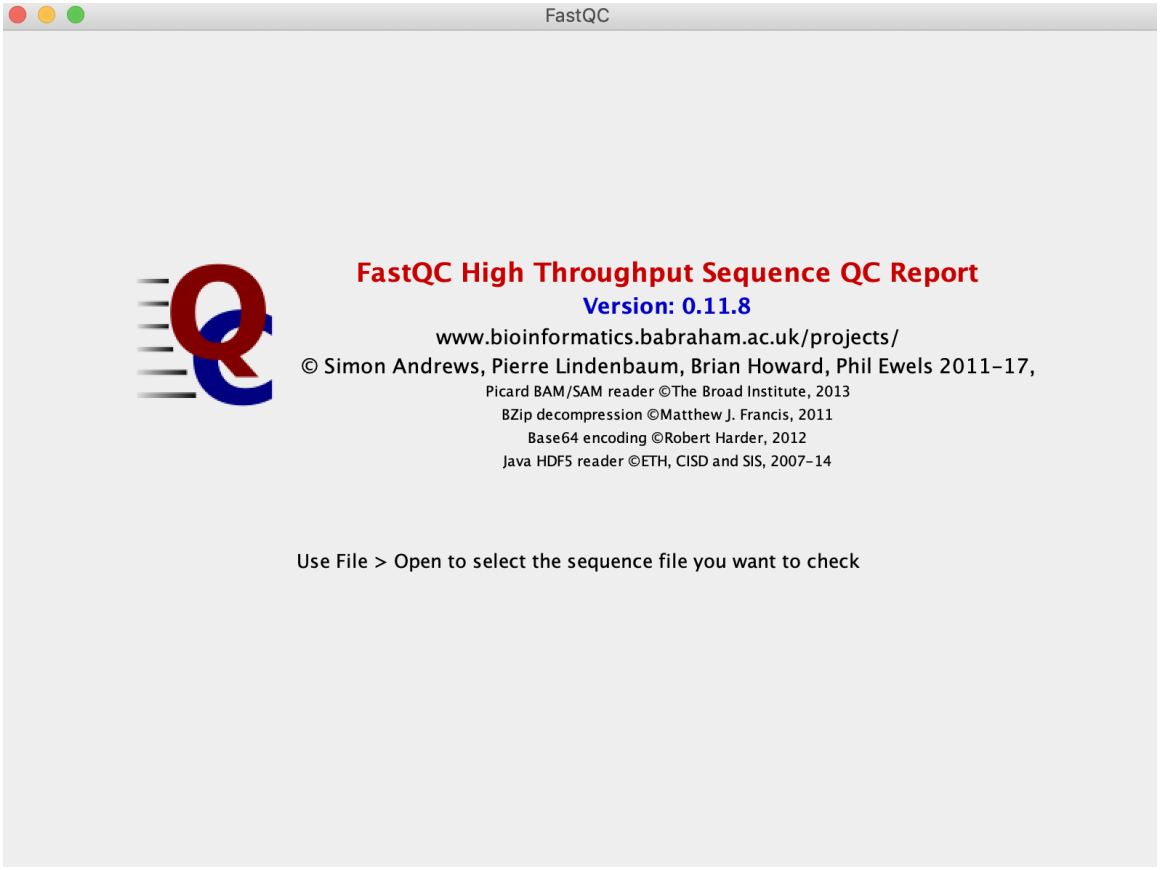
Here are some of the parameters for NGS data quality control.

- Check for primers and adaptors
- Trim low quality ends
- Remove low quality reads
- Remove short sequences
- Remove reads with ambiguous bases (“N”)
- Remove duplicates

In this training, you will be using a program called "fastqc" to visually inspect the NGS data quality. To run fqstqc program, open the terminal and enter

```
fastqc &
```

This will open FastQC program in the background.

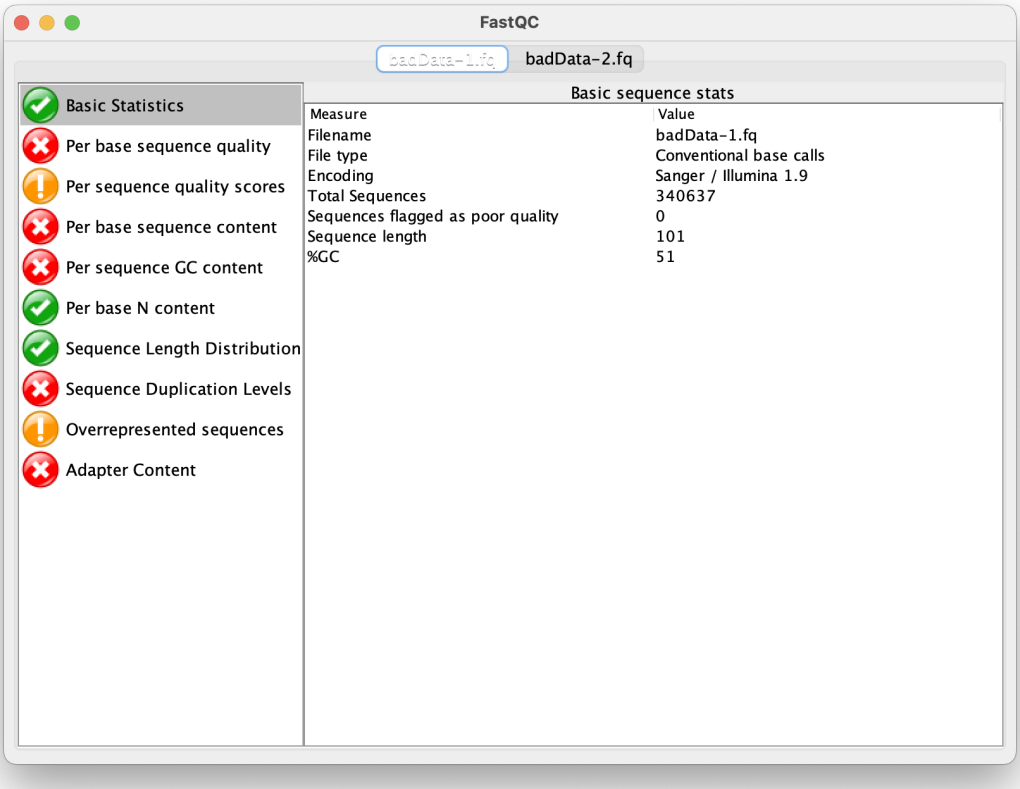


go to File -> Open and navigate to "Training/Data/" folder. Holding the control key select

```
badData-1.fq
badData-2.fq
```

It will generate a QC report after checking all the parameters. Here is the summary of all the statistics accroding to developer's page
(https://www.bioinformatics.babraham.ac.uk/projects/fastqc/)

4.1 Basic Statistics



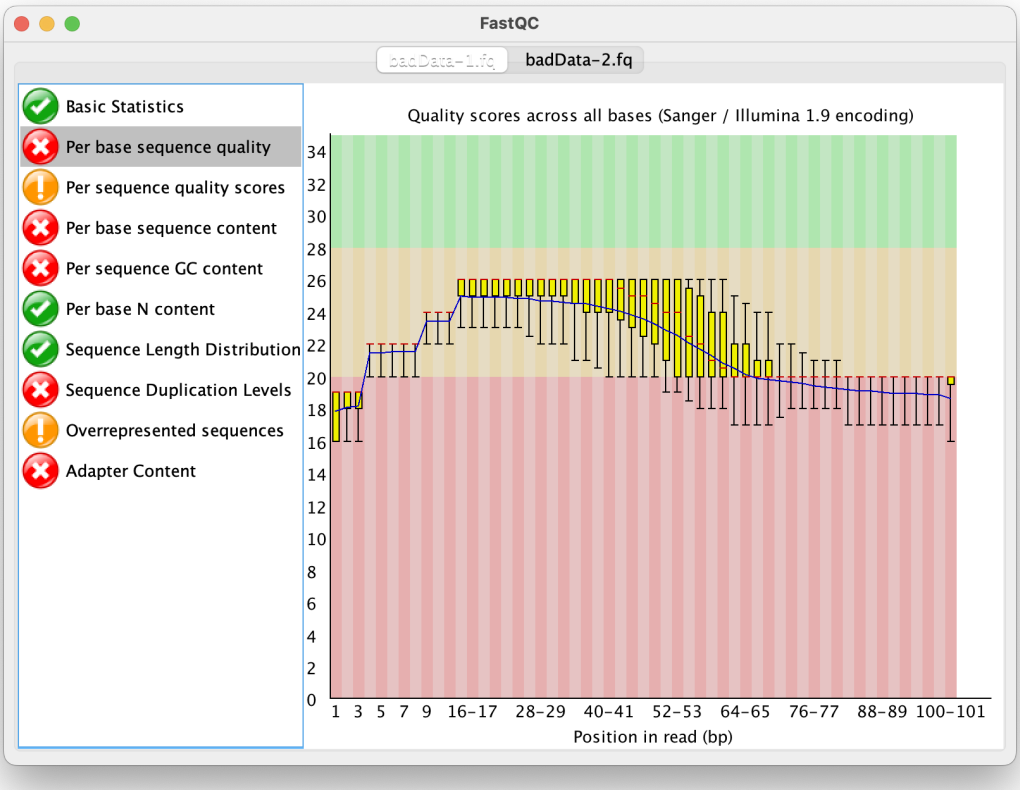
The Basic Statistics module generates some simple composition statistics for the file analysed.

- Filename: The original filename of the file which was analysed
- File type: Says whether the file appeared to contain actual base calls or colorspace data which had to be converted to base calls
- Encoding: Says which ASCII encoding of quality values was found in this file.
- Total Sequences: A count of the total number of sequences processed. There are two values reported, actual and estimated. At the moment these will always be the same. In the future it may be possible to analyse just a subset of sequences and estimate the total number, to speed up the analysis, but since we have found that problematic sequences are not evenly distributed through a file we have disabled this for now.
- Filtered Sequences: If running in Casava mode sequences flagged to be filtered will be removed from all analyses. The number of such sequences removed will be reported here. The total

sequences count above will not include these filtered sequences and will the number of sequences actually used for the rest of the analysis.

- Sequence Length: Provides the length of the shortest and longest sequence in the set. If all sequences are the same length only one value is reported.
- %GC: The overall %GC of all bases in all sequences

4.2 Per base sequence quality



This view shows an overview of the range of quality values across all bases at each position in the FastQ file. For each position a BoxWhisker type plot is drawn. The elements of the plot are as follows:

- The central red line is the median value
- The yellow box represents the inter-quartile range (25-75

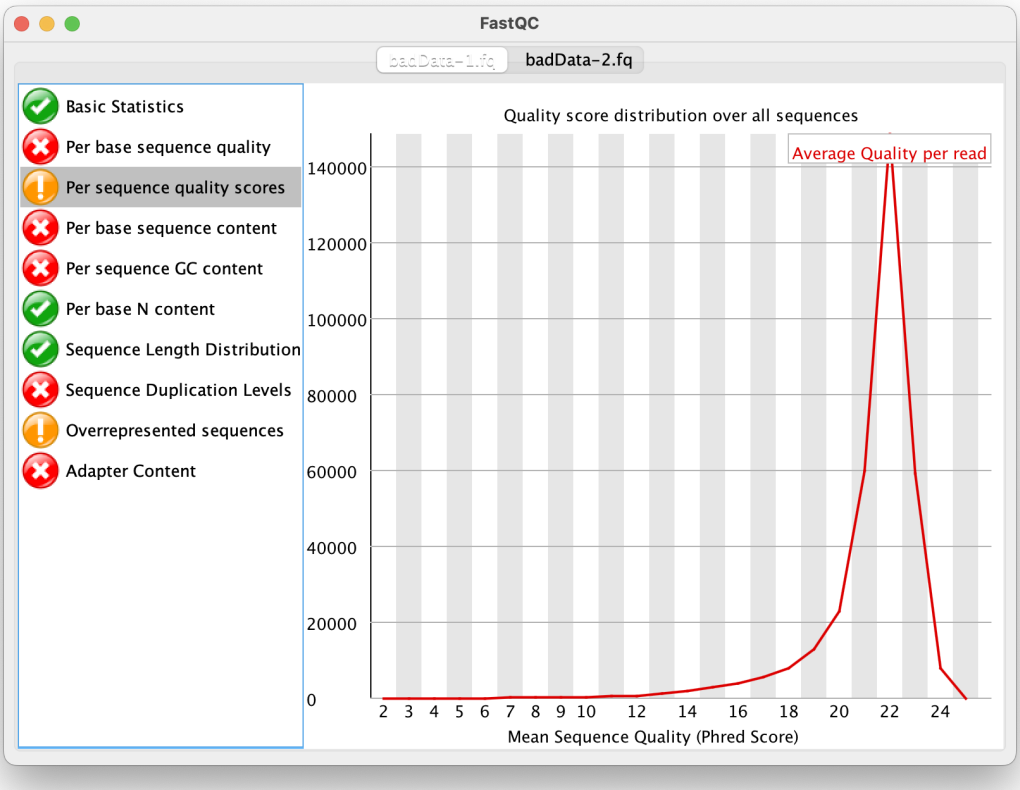
- The upper and lower whiskers represent the 10
- The blue line represents the mean quality

The y-axis on the graph shows the quality scores. The higher the score the better the base call. The background of the graph divides the y axis into very good quality calls (green), calls of reasonable quality (orange), and calls of poor quality (red). The quality of calls on most platforms will degrade as the run progresses, so it is common to see base calls falling into the orange area towards the end of a read.

It should be mentioned that there are number of different ways to encode a quality score in a FastQ file. FastQC attempts to automatically determine which encoding method was used, but in some very limited datasets it is possible that it will guess this incorrectly (ironically only when your data is universally very good!). The title of the graph will describe the encoding FastQC thinks your file used.

Results from this module will not be displayed if your input is a BAM/SAM file in which quality scores have not been recorded.

4.3 Per Sequence Quality Scores

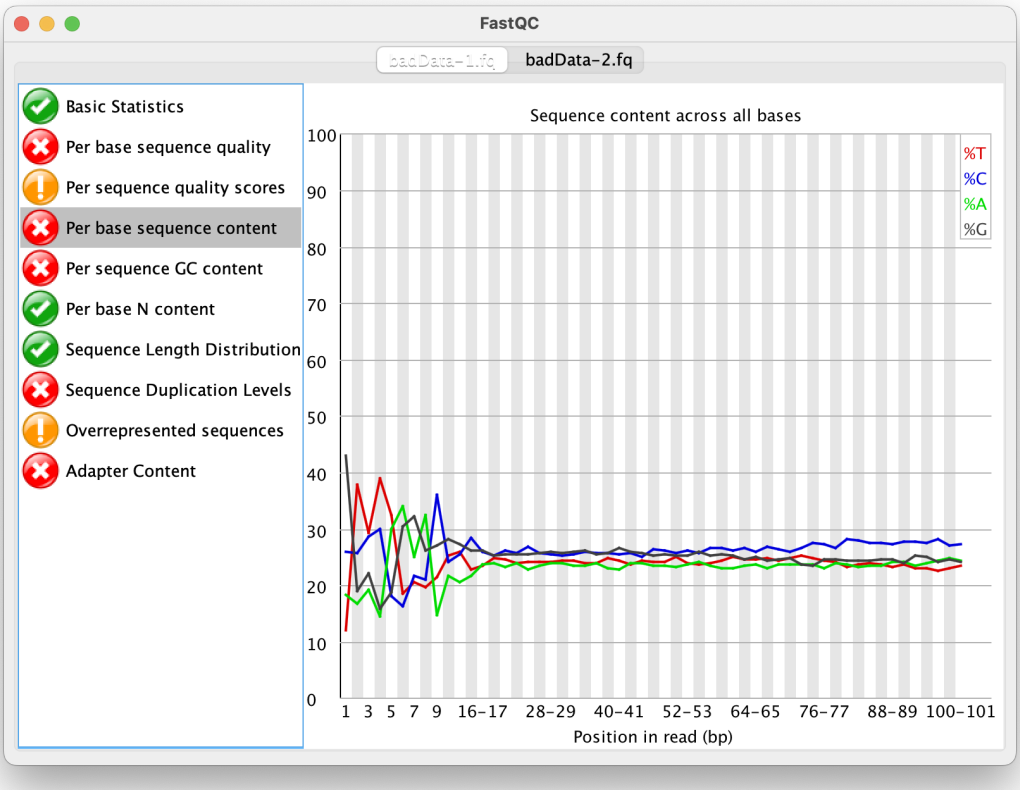


The per sequence quality score report allows you to see if a subset of your sequences have universally low quality values. It is often the case that a subset of sequences will have universally poor quality, often because they are poorly imaged (on the edge of the field of view etc), however these should represent only a small percentage of the total sequences.

If a significant proportion of the sequences in a run have overall low quality then this could indicate some kind of systematic problem - possibly with just part of the run (for example one end of a flowcell).

Results from this module will not be displayed if your input is a BAM/SAM file in which quality scores have not been recorded.

4.4 Per Base Sequence Content

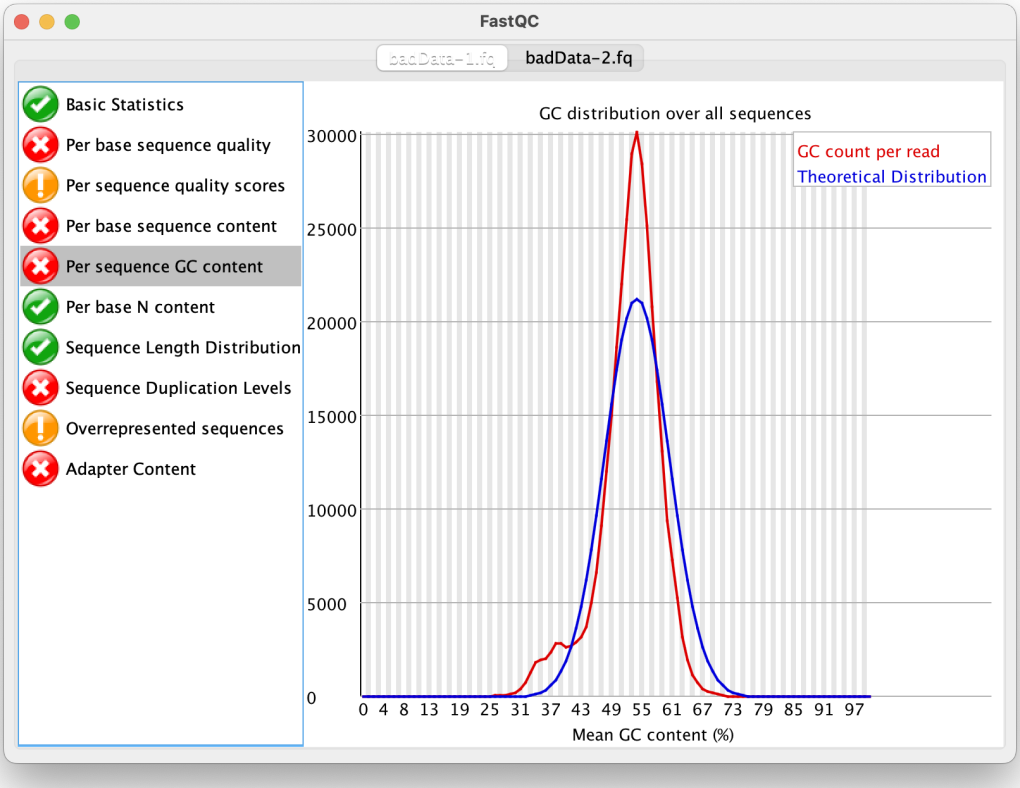


Per Base Sequence Content plots out the proportion of each base position in a file for which each of the four normal DNA bases has been called.

In a random library you would expect that there would be little to no difference between the different bases of a sequence run, so the lines in this plot should run parallel with each other. The relative amount of each base should reflect the overall amount of these bases in your genome, but in any case they should not be hugely imbalanced from each other.

It's worth noting that some types of library will always produce biased sequence composition, normally at the start of the read. Libraries produced by priming using random hexamers (including nearly all RNA-Seq libraries) and those which were fragmented using transposases inherit an intrinsic bias in the positions at which reads start. This bias does not concern an absolute sequence, but instead provides enrichement of a number of different K-mers at the 5' end of the reads. Whilst this is a true technical bias, it isn't something which can be corrected by trimming and in most cases doesn't seem to adversely affect the downstream analysis. It will however produce a warning or error in this module.

4.5 Per Sequence GC Content

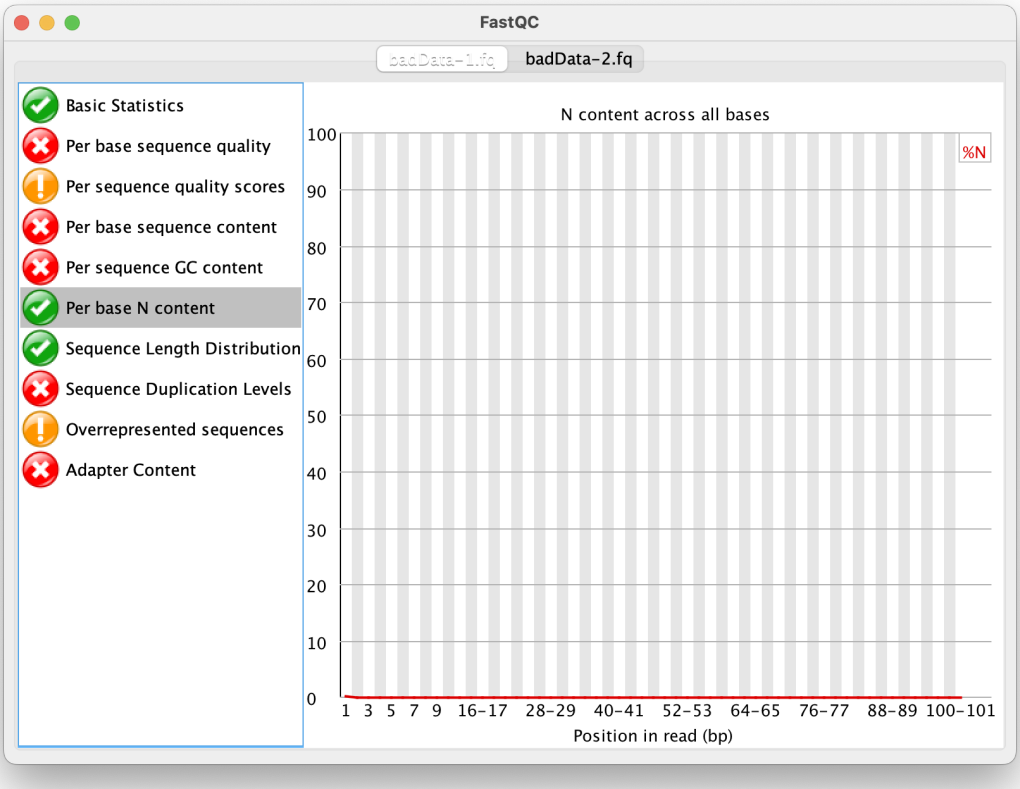


This module measures the GC content across the whole length of each sequence in a file and compares it to a modelled normal distribution of GC content.

In a normal random library you would expect to see a roughly normal distribution of GC content where the central peak corresponds to the overall GC content of the underlying genome. Since we don't know the the GC content of the genome the modal GC content is calculated from the observed data and used to build a reference distribution.

An unusually shaped distribution could indicate a contaminated library or some other kinds of biased subset. A normal distribution which is shifted indicates some systematic bias which is independent of base position. If there is a systematic bias which creates a shifted normal distribution then this won't be flagged as an error by the module since it doesn't know what your genome's GC content should be.

4.6 Per Base N Content

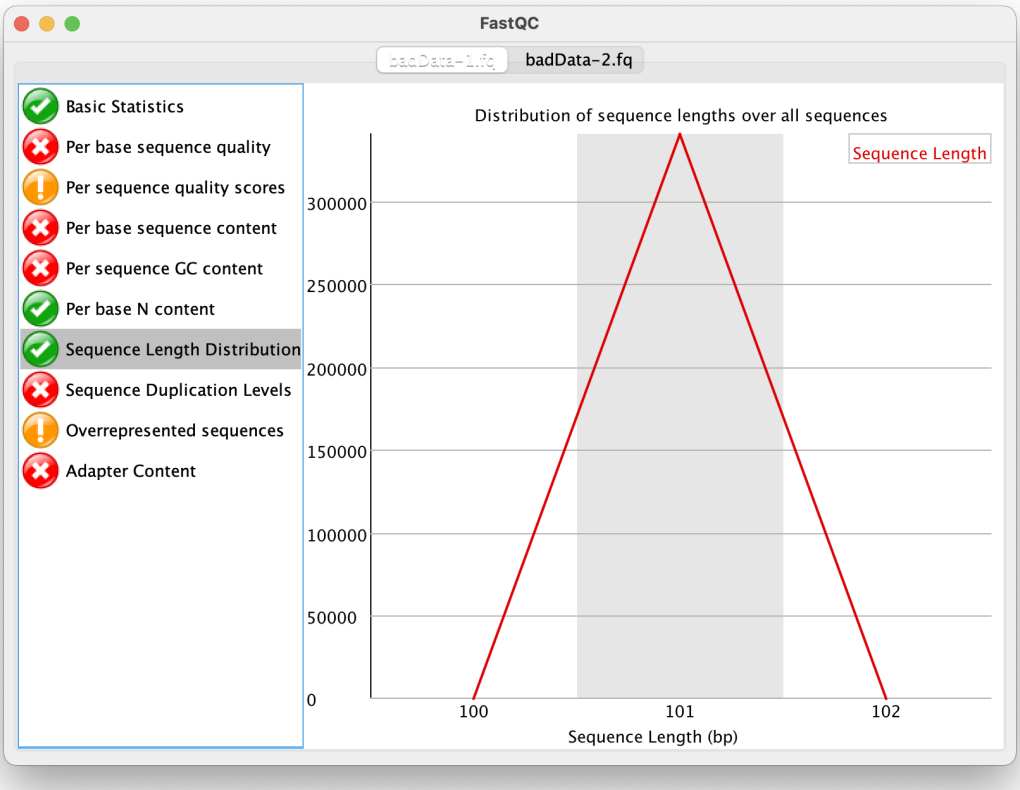


If a sequencer is unable to make a base call with sufficient confidence then it will normally substitute an N rather than a conventional base] call

This module plots out the percentage of base calls at each position for which an N was called.

It's not unusual to see a very low proportion of Ns appearing in a sequence, especially nearer the end of a sequence. However, if this proportion rises above a few percent it suggests that the analysis pipeline was unable to interpret the data well enough to make valid base calls.

4.7 Sequence Length Distribution

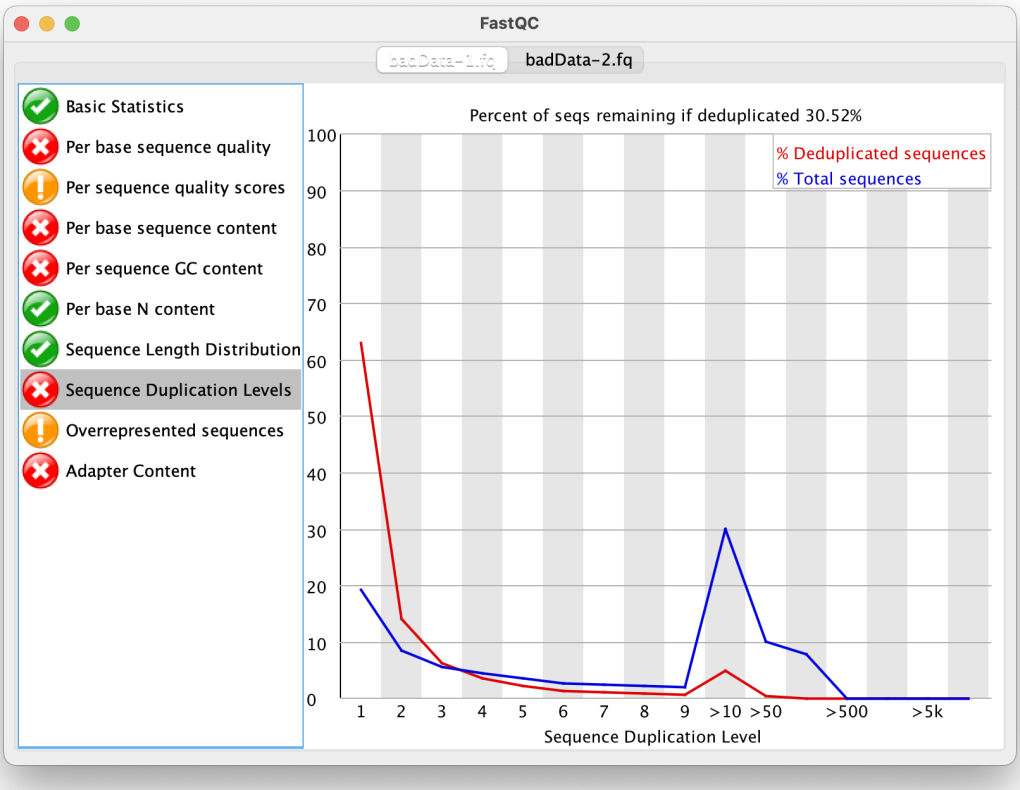


Some high throughput sequencers generate sequence fragments of uniform length, but others can contain reads of wildly varying lengths. Even within uniform length libraries some pipelines will trim sequences to remove poor quality base calls from the end.

This module generates a graph showing the distribution of fragment sizes in the file which was analysed.

In many cases this will produce a simple graph showing a peak only at one size, but for variable length FastQ files this will show the relative amounts of each different size of sequence fragment.

4.8 Sequence Duplication Levels



In a diverse library most sequences will occur only once in the final set. A low level of duplication may indicate a very high level of coverage of the target sequence, but a high level of duplication is more likely to indicate some kind of enrichment bias (eg PCR over amplification).

This module counts the degree of duplication for every sequence in a library and creates a plot showing the relative number of sequences with different degrees of duplication.

To cut down on the memory requirements for this module only sequences which first appear in the first 100,000 sequences in each file are analysed, but this should be enough to get a good impression for the duplication levels in the whole file. Each sequence is tracked to the end of the file to give a representative count of the overall duplication level. To cut down on the amount of information in the final plot any sequences with more than 10 duplicates are placed into grouped bins to give a clear impression of the overall duplication level without having to show each individual duplication value.

Because the duplication detection requires an exact sequence match over the whole length of the

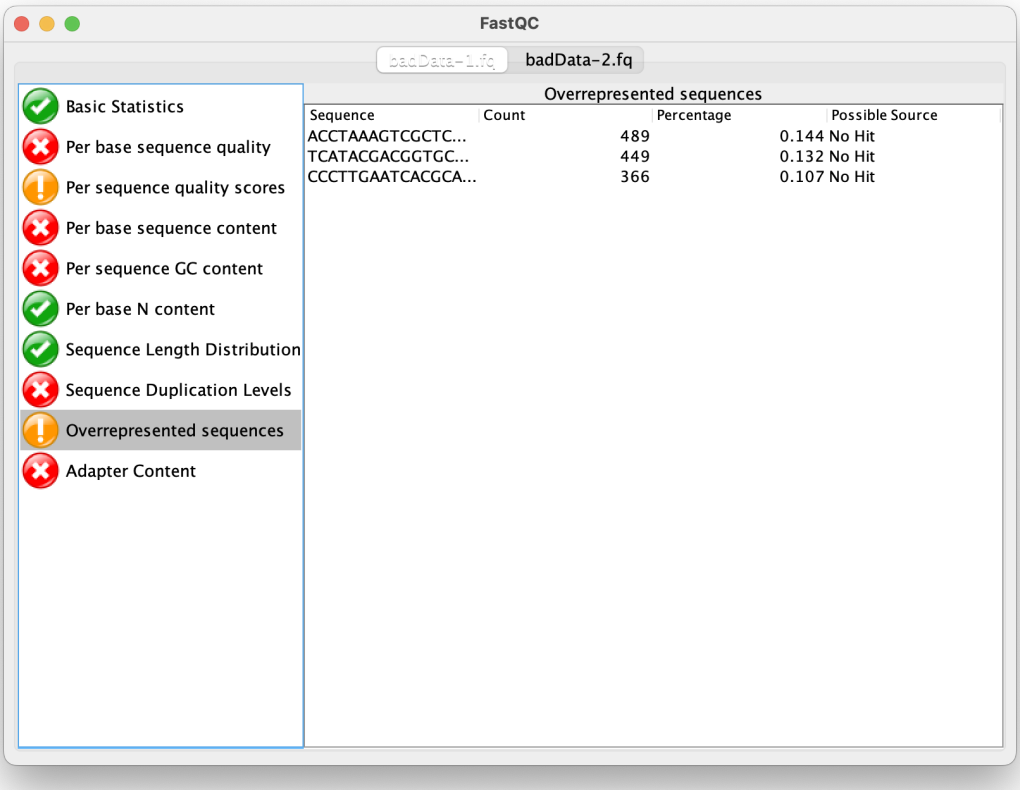
sequence, any reads over 75bp in length are truncated to 50bp for the purposes of this analysis. Even so, longer reads are more likely to contain sequencing errors which will artificially increase the observed diversity and will tend to underrepresent highly duplicated sequences.

The plot shows the proportion of the library which is made up of sequences in each of the different duplication level bins. There are two lines on the plot. The blue line takes the full sequence set and shows how its duplication levels are distributed. In the red plot the sequences are de-duplicated and the proportions shown are the proportions of the deduplicated set which come from different duplication levels in the original data.

In a properly diverse library most sequences should fall into the far left of the plot in both the red and blue lines. A general level of enrichment, indicating broad oversequencing in the library will tend to flatten the lines, lowering the low end and generally raising other categories. More specific enrichments of subsets, or the presence of low complexity contaminants will tend to produce spikes towards the right of the plot. These high duplication peaks will most often appear in the blue trace as they make up a high proportion of the original library, but usually disappear in the red trace as they make up an insignificant proportion of the deduplicated set. If peaks persist in the blue trace then this suggests that there are a large number of different highly duplicated sequences which might indicate either a contaminant set or a very severe technical duplication.

The module also calculates an expected overall loss of sequence were the library to be deduplicated. This headline figure is shown at the top of the plot and gives a reasonable impression of the potential overall level of loss.

4.9 Overrepresented Sequences



A normal high-throughput library will contain a diverse set of sequences, with no individual sequence making up a tiny fraction of the whole. Finding that a single sequence is very overrepresented in the set either means that it is highly biologically significant, or indicates that the library is contaminated, or not as diverse as you expected.

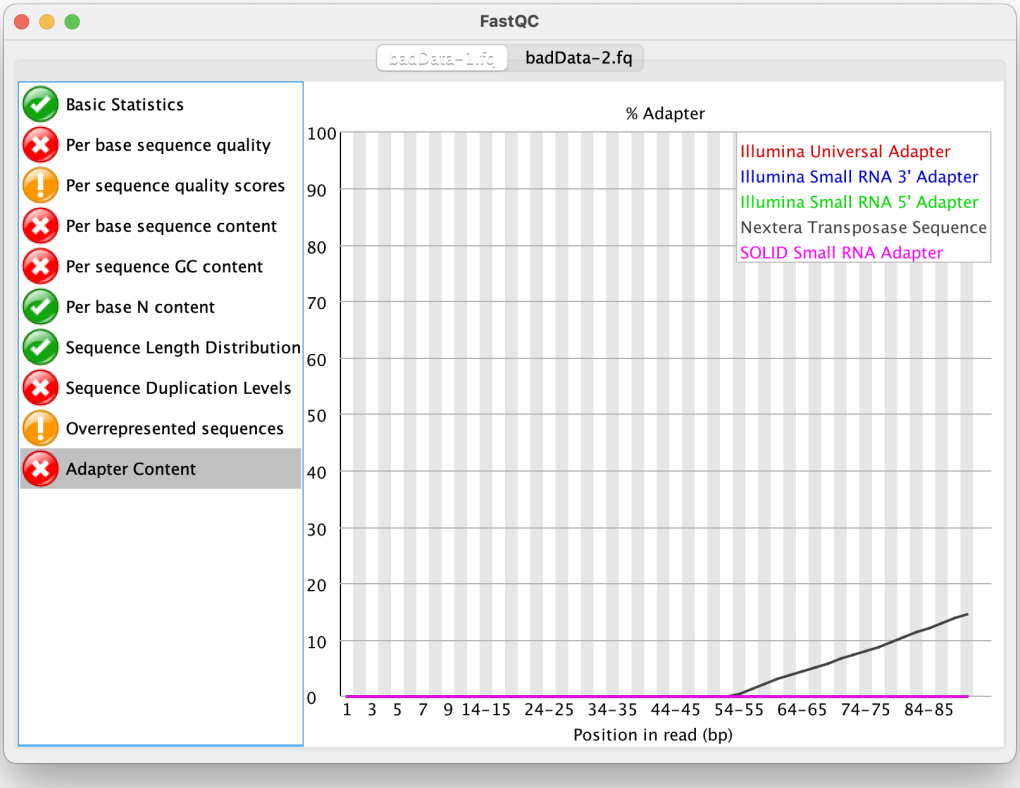
This module lists all of the sequence which make up more than 0.1

For each overrepresented sequence the program will look for matches in a database of common contaminants and will report the best hit it finds. Hits must be at least 20bp in length and have no more than 1 mismatch. Finding a hit doesn't necessarily mean that this is the source of the contamination, but may point you in the right direction. It's also worth pointing out that many adapter sequences are very similar to each other so you may get a hit reported which isn't technically correct, but which has very similar sequence to the actual match.

Because the duplication detection requires an exact sequence match over the whole length of the sequence any reads over 75bp in length are truncated to 50bp for the purposes of this analysis.

Even so, longer reads are more likely to contain sequencing errors which will artificially increase the observed diversity and will tend to underrepresent highly duplicated sequences.

4.10 Adapter Content



The Kmer Content module will do a generic analysis of all of the Kmers in your library to find those which do not have even coverage through the length of your reads. This can find a number of different sources of bias in the library which can include the presence of read-through adapter sequences building up on the end of your sequences.

You can however find that the presence of any overrepresented sequences in your library (such as adapter dimers) will cause the Kmer plot to be dominated by the Kmers these sequences contain, and that it's not always easy to see if there are other biases present in which you might be interested.

One obvious class of sequences which you might want to analyse are adapter sequences. It is useful to know if your library contains a significant amount of adapter in order to be able to assess whether

you need to adapter trim or not. Although the Kmer analysis can theoretically spot this kind of contamination it isn't always clear. This module therefore does a specific search for a set of separately defined Kmers and will give you a view of the total proportion of your library which contain these Kmers. A results trace will always be generated for all of the sequences present in the adapter config file so you can see the adapter content of your library, even if it's low.

The plot itself shows a cumulative percentage count of the proportion of your library which has seen each of the adapter sequences at each position. Once a sequence has been seen in a read it is counted as being present right through to the end of the read so the percentages you see will only increase as the read length goes on.

Now open the goodData-1.fq and goodData-2.fq and compare the results.

As we can see, in the bad data the quality of the reads drops at 3' ends and needs trimming of these regions.

We use "trim_galore" to remove adaptors, to trim low quality reads and to remove short sequences. Internally it uses another program called cutadapt. It will work on compressed and uncompressed fastq files. To run it

```
 cd ~/Training/Data/

trim_galore -q 15 --length 60 --paired badData-1.fq badData-2.fq
```

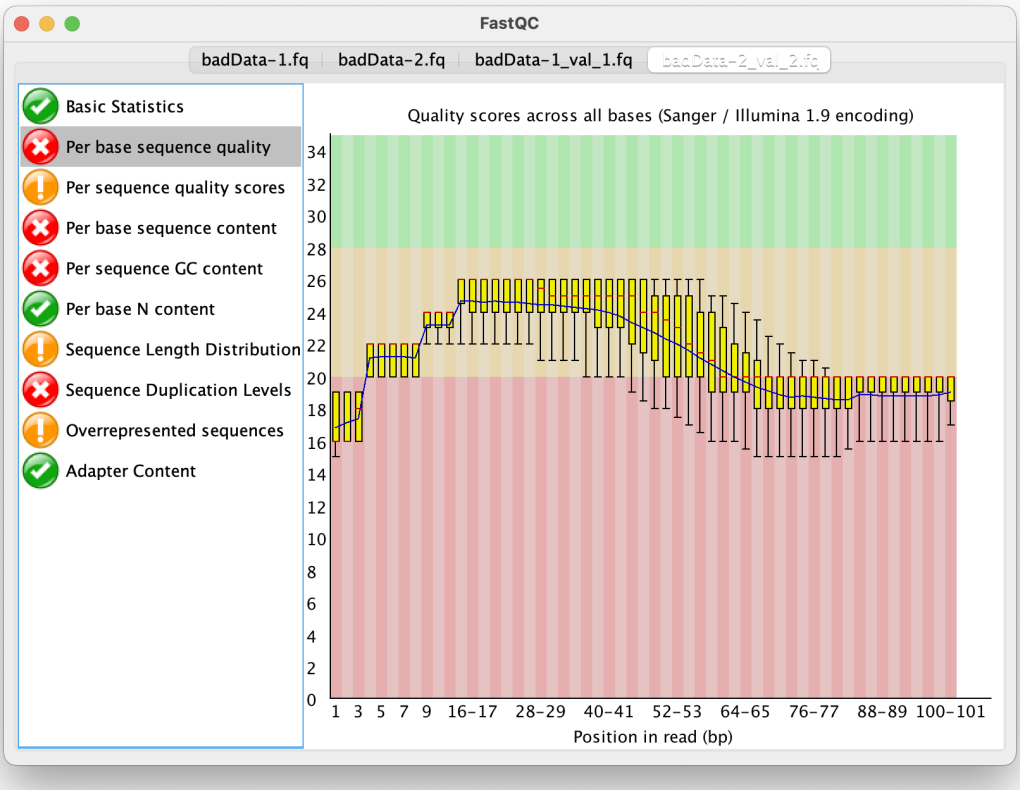
This trims reads with phred quality less than 15 and length shorter than 60bp. By default, it also detects commonly used adaptors and removes them if present. If special adaptor sequences are used in sequencing, provide the adaptor's sequence with "-a" option to remove.

After trimming, for each fastq file it generates summary files (badData-1.fq_trimming_report.txt and badData-2.fq_trimming_report.txt) and validated reads files (badData-1_val_1.fq and badData-2_val_2.fq)*.

(* These file names may vary based on the program version)

Recheck the cleaned data using fastqc. Did you see any change in the quality after trimming (do you see any difference at all?)

Here is the per base sequence quality



There is some improvement on the per base quality, though it is not a through cleaning. It is a good practice to clean the data before we use it for any downstream analysis.