# Introduction To NGS Data & Analytic Tools

Steve Pederson

Bioinformatics Hub
University of Adelaide

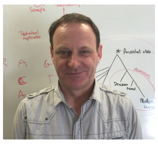
April 19, 2017

# Licensing

# Contents

# Workshop Information

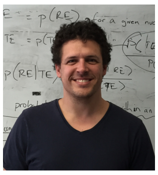
# The Trainers



**Mr. Steve Pederson**
Co-ordinator
Bioinformatics Hub
The University of Adelaide
South Australia
stephen.pederson@adelaide.edu.au

**Dr. Terry Bertozzi**
Bioinformatician
SA Museum
South Australia
Terry.Bertozzi@samuseum.sa.gov.au

**Dr. Hien To**
Bioinformatician
Bioinformatics Hub
The University of Adelaide
South Australia
hien.to@adelaide.edu.au



**Dr. Jimmy Breen**
Bioinformatician
Bioinformatics Hub & Robinson Research Institute
The University of Adelaide
South Australia
jimmy.breen@adelaide.edu.au

**Mr. Alastair Luddington**
Bioinformatician
Bioinformatics Hub
The University of Adelaide
South Australia
alastair.luddington@adelaide.edu.au

# Welcome

Thank you for your attendance & welcome to the Introduction to NGS Data & NGS Analytic Tools Workshop. This is a free offering by the University of Adelaide, Bioinformatics Hub which is a centrally funded initiative from the Department of Vice-Chancellor (Research), with the aim of assisting & enabling researchers in their work. Training workshops & seminars such as this one are an important part of this initiative.

The Bioinformatics Hub itself has a web-page on the University website at http://www.adelaide.edu.au/bioinformatics-hub/. To be kept up to date on upcoming events and workshops, please join the internal Bioinformatics mailing list on http://list.adelaide.edu.au/mailman/listinfo/bioinfo. Also, please consider following the Bioinformatics Hub on Twitter as many handy hints are posted on this feed https://twitter.com/UofABioinfoHub.

Today's workshop has been prepared with generous technical support & advice provided by Dr Nathan Watson-Haigh (*ACPFG*), Dr Dan Kortschak (*Adelaide University, Adelson Research Group*), Dr Zhipeng Qu (*Adelaide University, Adelson Research Group*), Dr Mark Corbett (*Robinson Institute*) & Dr John Toubia (*Adelaide Centre for Cancer Genomics*). The tutors today are Steve Pederson, Dr Hien To, Dr Jimmy Breen, Mr Alastair Luddington from the University of Adelaide, Bioinformatics Hub, & Dr Terry Bertozzi (*SA Museum & Adelaide University, Adelson Research Group*). We hope it will be useful in enabling you to continue and to advance your research.

# Course Summary

In today's workshop we will be introducing you to a small number of the basic tools required for NGS data handling, as well as giving you a basic familiarity with what the data actually looks like. Whilst we will not be able to cover all of the rich & diverse set of tools available, we hope to cover many of the key concepts & questions to ask of your data, as well as give you an understanding of what information is actually in the data.

The majority of data handling and analysis required in the field of bioinformatics uses the *command line*, alternatively known as the terminal or the *bash shell*, so some useful tips for the command line will be included amongst the material. Whilst most of the session will involve looking at individual files, in reality most of our analysis will be performed using some type of script to automate, & easily reproduce an analysis.

Today's session is also intended to explore several tools in actual detail, rather than rush across the whole field. There is large amount of information that we won't have time to discuss, but hopefully some important tools and thought processes will be covered &

enable you make better progress with your own datasets.

## Post Workshop

The VMs which we work on today will remain active for week or so after the workshop, so feel free to continue exploring any sections that you weren't able to make it through. We'd also encourage you to sign up for some of the high-traffic websites like *BioStars* or *SEQanswers* as these are a rich resource for your own problem solving.

## Document Structure

We have provided you with a printed and an electronic copy of the workshop's hands-on tutorial documents. We have done this for two reasons: 1) you will have something to take away with you at the end of the workshop, and 2) you can save time (mis)typing commands on the command line by using copy-and-paste.

*We advise you to use Acrobat Reader to view the PDF. This is because it properly supports some features we have implemented to ensure that copy-and-paste of commands works as expected. This includes the appropriate copy-and-paste of special characters like tilde and hyphens as well as skipping line numbers for easy copy-and-past of whole code blocks.*

While you could fly through the hands-on sessions doing copy-and-paste, you will learn more if you use the time saved from not having to type all those commands, to understand what each command is doing!

The commands to enter at a terminal look something like this:

```
tophat --solexa-quals -g 2 --library-type fr-unstranded -j \
    annotation/Danio_rerio.Zv9.66.spliceSites -o tophat/ZV9_2cells \
    genome/ZV9 data/2cells_1.fastq data/2cells_2.fastq
```

The following styled code is not to be entered at a terminal, it is simply to show you the syntax of the command. You must use your own judgement to substitute in the correct arguments, options, filenames etc

```
tophat [options]* <index_base> <reads_1> <reads_2>
```

The following icons are used in the margin, throughout the documentation to help you navigate around the document more easily:

- i Important
- 🗒 For reference
- 👣 Follow these steps
- Q Questions to answer
- ! Warning - STOP and read
- ✸ Bonus exercise for fast learners
- ✸ Advanced exercise for super-fast learners

# Understanding NGS Data & FASTQ Format

Primary Author(s):
Steve Pederson, Bioinformatics Hub, University of Adelaide
stephen.pederson@adelaide.edu.au

Contributor(s):
Jimmy Breen, Robinson Research Institute & Bioinformatics Hub, University of Adelaide
jimmy.breen@adelaide.edu.au
Dan Kortschak, Adelson Research Group, University of Adelaide
dan.kortschak@adelaide.edu.au
Terry Bertozzi, SA Museum Terry.Bertozzi@samuseum.sa.gov.au

# Initial Goals

1. Understand the *Sequencing by Synthesis* Process & Data Generation

2. Understand how a NGS library is created and the difference between an index and a barcode

3. Become familiar with the `.fastq` file format

4. Understand what errors and artefacts lie within the data

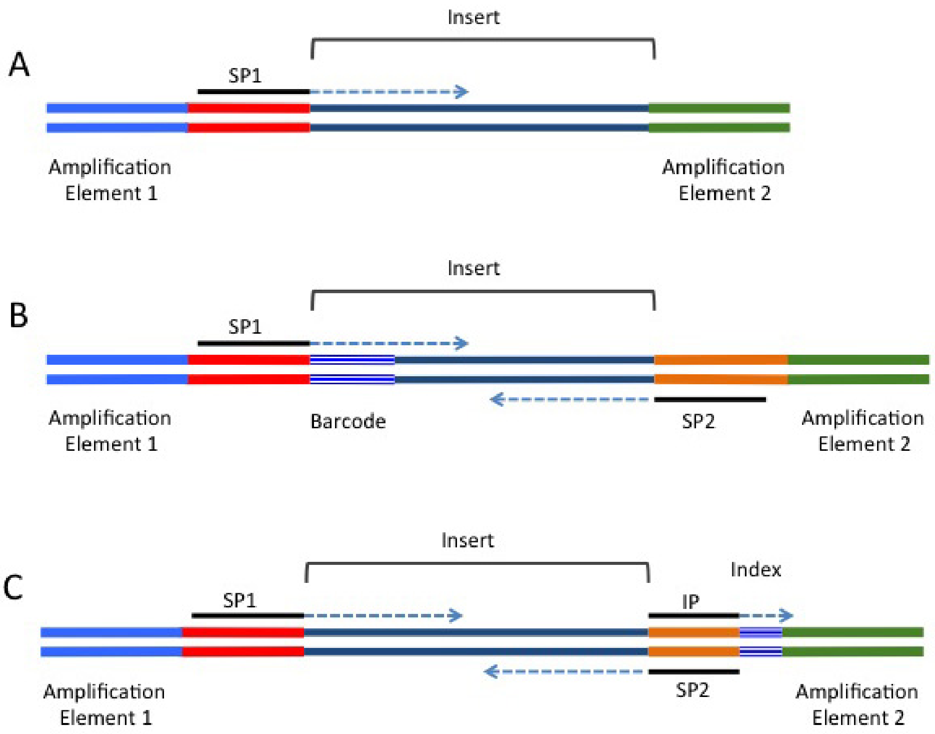
5. Learn how to assess the quality of data & make informed decisions

# NGS Data Generation

Before we can begin to analyse any data, it is helpful to understand how it was generated. Whilst there are numerous platforms for generation of NGS data, today we will look at the Illumina *Sequencing by Synthesis* method, which is one of the most common methods in use today. Many of you will be familiar with the process involved, but it may be worth looking at the following 5-minute video from Illumina: `http://youtu.be/womKfikWlxM` As setting up the sound with the VMs can be tricky, it will be easier to view this from your own regular browser. Briefly minimise the VM (see Section 1.4), open your regular browser & please use your headphones if you brought them.

This video refers to the process *tagmentation*. This is a relatively recent method for fragmenting & attaching adapters to DNA, with an alternative, more traditional methods being sonication, poly-adenylation & attachment of appropriate adaptors in separate steps. This step may vary depending on your experiment, but the important concept to note during sample preparation is that the DNA insert has multiple sequences ligated to either end. These include 1) the *sequencing primers*, 2) index & /or *barcode* sequences, and 3) the flow-cell binding oligos.

To demonstrate these concepts further, observe the following figure that shows the DNA construct needed to run an illumina sequencing run, and the amplification steps required:



Assuming each ’spot’ on the flowcell is generated from a unique DNA sequence, there are two important sequencing errors that will occur during this process. What do you think they might be?

Will these errors have a more significant effect if they occur during the sequence detection stage, or during generation of the DNA clones within each cluster?

The figure above that shows the setup of the sequenced DNA fragment contains two distinguishing sequences; an **index** and a **barcode**. What is the difference between the two?

What is likely to happen if your DNA insert is shorter than the machine’s read length? (e.g. My DNA sequence is 50bp, but the Illumina sequencing run is 75 base pairs)

# FASTQ File Format

As the sequences are extended during the sequencing reaction, an image is recorded which is effectively a movie or series of frames at which the addition of bases is recorded & detected. We mostly don't deal with these image files, but will handle data generated from these in *fastq* format, which can commonly have the file suffix *.fq* or *.fastq*. As these files are often very large, they will often be zipped using `gzip` or `bzip`. Whilst we would instinctively want to unzip these files using the command `gunzip`, most NGS tools are able to work with zipped fastq files, and this is usually unnecessary. This can save considerable hard drive space, which is an important consideration when handling NGS datasets as the quantity of data can easily push your storage capacity to it's limit.

We should still have a terminal open from the previous section &, if necessary, use the `cd` command to make sure you are in the home ( `~/`) directory. The command `zcat` unzips a file & prints the output to the terminal, or standard output (*stdout*). If we did this to these files, we would see a stream of data whizzing past in the terminal, but instead we can just pipe the output of `zcat` to the command head to view the first 10 lines of a file.

```
cd ~/rawData/WGS
zcat WGS_SRR065388_1.fastq.gz | head -n8
```

In the above command, we have used a trick commonly used in Linux systems where we have taken the output of one command (`zcat WGS_SRR065388_1.fastq.gz`) and sent it to another command (`head`) by using the *pipe symbol* (|). This is literally like sticking a pipe on the end of a process & redirecting the output to the input another process. If you think of things as being like a data factory you can almost visualise it. There are no limits to the number of commands that you can string together using this trick. Additionally, we gave the argument `-n8` to the command `head` to ensure that we only printed the first eight lines.

Don't Panic!!!

If at some stage today you find that the terminal has become unresponsive, or you are seeing an unexpected stream of data fly past, you can abort whichever process is currently running in the terminal by entering `Ctrl-c`. This is an instruction to the computer to 'kill the current process.' & you may be surprised at how often this comes in handy. Even experienced programmers rely on this trick from time to time.

In the output from the above terminal command, we have obtained the first 8 lines of the gzipped fastq file. This gives a clear view of the fastq file format, where each individual read spans four lines. These lines are:

1. The read identifier

2. The sequence read
3. An alternate line for the identifier (commonly left blank as just a + symbol acting as a placeholder)
4. The quality scores for each position along the read as a series of ascii text characters.

Let's have a brief look at each of these lines and what they mean.

### The read identifier

This line begins with an @ symbol and although there is some variability, it traditionally has several components. Today's data have been sourced from an EBI data repository with the identifier SRR065388. For the first sequence in this file, we have the full identifier @SRR065388.1 HWUSI-EAS687_61DAJ:6:1:1054:18644/1 which has the following components:

| | |
|---|---|
| SRR065388.1 | The aforementioned EBI identifier & the sequence ID within the file. As this is the first read, we have the number 1. NB: This identifier is **not** present when data is obtained directly from the machine or service provider. |
| HWUSI-EAS687_61DAJ | The unique machine ID |
| 6 | The flowcell lane |
| 1 | The tile within the flowcell lane |
| 1054 | The *x*-coordinate of the cluster within the tile |
| 18644 | The *y*-coordinate of the cluster within the tile |
| /1 | Indicates that this is the first read in a set of paired end reads |

As seen in the subsequent sections, these pieces of information can be helpful in identifying if any spatial effects have affected the quality of the reads. By and large you won't need to utilise most of this information, but it can be handy for times of serious data exploration.

While we are inspecting our data, have a look at the beginning of the second file.

```
zcat WGS_SRR065388_2.fastq.gz | head -n8
```

Here you will notice that the information in the identifier is identical to the first file we inspected, with the exception that there is a /2 at the end. This indicates that these reads are the second set in what are known as *paired-end* reads, as were introduced in the above video. The two files will have this identical structure where the order of the sequences in one is identical to the order of the sequences in the other. This way when they are read as a pair of files, they can be stepped through read-by-read & the integrity of the data will be kept intact.

### The Illumina Chastity Filter

It is also worth noting that the reads we've just glanced at come from a version of the Illumina *casava* pipeline which is <1.8, and which is a relatively common format. The casava version really just describes how old the software on the Illumina machine is & we don't choose this. For more recently generated reads where the casava software is >1.8 of the casava, there is an additional field in the identifier which indicates whether a read would have *failed* an initial QC check. An example of this format would be:

```
@D5B4KKQ1:554:C4YHPACXX:4:1101:1084:2100 1:Y:0:
```

Note the "Y" in the final fields, which indicates this sequence would have **failed** QC. These low-quality reads were automatically removed in earlier versions of the pipeline and were omitted from the fastq file. However, they are now included by some sequence providers with this additional field indicated in the read identifier. Inspection of this line will enable you to find out which version of the casava pipeline has been used, and whether you need to perform any additional filtering steps to remove low quality reads. The tool `fastq_illumina_filter` is designed to remove these reads for you & the tool, along with usage instructions can be found at http://cancan.cshl.edu/labmembers/gordon/fastq_illumina_filter/.

### Quality Scores

The only other line in the fastq format that really needs some introduction is the quality score information. These are presented as single *ascii* text characters for simple visual alignment with the sequence, and each character corresponds to a numeric value, which is the quality score. In the ascii text system, each character has a numeric value which we can interpret as an integer. Head to the website with a description of these at http://en.wikipedia.org/wiki/ASCII#ASCII_printable_code_chart.

The first 31 ASCII characters are non-printable & contain things like end-of-line marks and tab spacings, and note that the first printable character after the space (character 32) is '!' which corresponds to the value 33. In short, the values 33-47 are symbols like !, ; #, $ etc, whereas the values 48-57 are the characters 0-9. Next are some more symbols (including @ for the value 64), with the upper case characters representing the values 65-90 & the lower case letters representing the values 97-122.

In the sample you inspected above, try and determine the quality scores for the first 5 bases.

### The PHRED +33/64 Scoring System

Now that we understand how to turn the quality scores from an ascii character into a numeric value, we need to know what these numbers represent. The two main systems in common usage are PHRED +33 and PHRED +64 and for each of these coding systems we either subtract 33 or 64 from the numeric value associated with each ascii character to give us a PHRED score. As will be discussed later, this score ranges between 0 and about 41.

The PHRED system used is determined by the software installed on the sequencing machine, with early machines using PHRED + 64 (casava <1.5), and more recent machines tending to use PHRED + 33. For example, in PHRED +33, the @ symbol corresponds to Q = 64 - 33 = 31, whereas in PHRED +64 it corresponds to Q = 64 - 64 = 0.

The following table demonstrates the comparative coding scale for the different formats:

```
...............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII......................
.................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ......................
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL....................................................
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
|                         |    |        |                              |                     |
33                        59   64       73                            104                   126

I - Illumina 1.3+ Phred+64, raw reads typically (0, 40)
J - Illumina 1.5+ Phred+64, raw reads typically (3, 40)
L - Illumina 1.8+ Phred+33, raw reads typically (0, 41)
```

Have a look at the quality scores in our paired-end reads (WGS_SRR065388_1.fastq.gz and WGS_SRR065388_2.fastq.gz) Which coding system do you think has been used for the reads that we have?
In the PHRED +33 coding system, the character '@' is used. Can you think of any potential issues this would cause when searching within a fastq file? (Hint: Consider the sequence identifier rows)

### Interpretation of PHRED Scores

All NGS platforms have non-zero error rates during the sequencing process (http://www.molecularecologist.com/next-gen-table-3c-2014/), and the quality scores are related to the probability of calling an incorrect base. The PHRED scoring system predates NGS technologies & is based on whether the wavelength from one specific base is clearly dominant through the formula

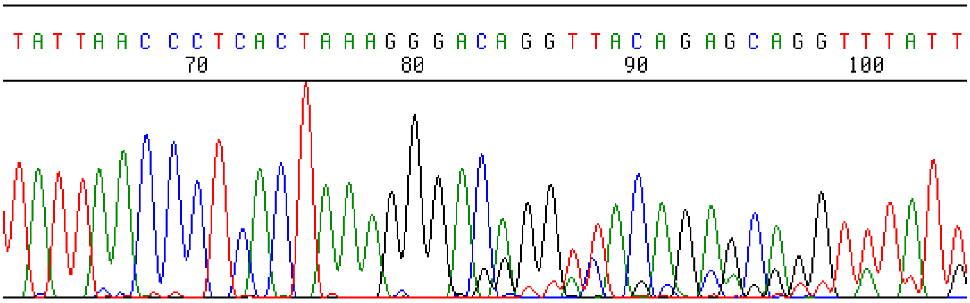
$$Q = -10log_{10}P \tag{1}$$

where $P$ is the probability of calling the incorrect base.

This is more easily seen in the following table:

| PHRED Score | Probability of Incorrect Base Call | Accuracy of Base Call |
|---|---|---|
| 0 | 1 in 1 | 0% |
| 10 | 1 in 10 | 90% |
| 20 | 1 in 100 | 99% |
| 30 | 1 in 1000 | 99.9% |
| 40 | 1 in 10000 | 99.99% |

As each base is added, the light wavelength associated with each base (i.e. the colour) is detected. In theory, there should only be one colour observed, but in reality there will always be a residual signal from the other bases. The following chromatogram based on the traditional Sanger sequencing methods demonstrates this well (Source: http://seqcore.brcf.med.umich.edu.



Note that early in this sequence, the colour peaks are clear but with small residual peaks of background signal. These peaks would receive high PHRED scores. Around position 83 however, the peaks become less obvious and any corresponding PHRED score would be lower. This basic principle is applied on a massively parallel scale during the generation of NGS data.

A common threshold for inclusion of a sequence is that all bases must have a Q score > 20. Considering the millions of sequences obtained from a flowcell, do you think that NGS data is likely to be highly accurate?

# FASTQC

Primary Author(s):
Steve Pederson, Bioinformatics Hub, University of Adelaide
stephen.pederson@adelaide.edu.au

Contributor(s):
Jimmy Breen, Robinson Research Institute & Bioinformatics Hub, University of Adelaide
jimmy.breen@adelaide.edu.au
Dan Kortschak, Adelson Research Group, University of Adelaide
dan.kortschak@adelaide.edu.au
Terry Bertozzi, SA Museum Terry.Bertozzi@samuseum.sa.gov.au

# Using fastqc

Removal of low-confidence base calls is an important part of any NGS analysis, and we can begin this process by checking the quality of libraries using the tool `fastqc`. As with all programs on the command line, we need to see how it works before we use it. The following command will open the help file in the `less` pager which we used earlier. To navigate through the file, use the <spacebar> to move forward a page, <b> to move back a page & <q> to exit the manual.

```
fastqc -h | less
```

Fastqc will create an html report on each file you submit, which can be opened from any web browser, such as `firefox` As seen in the help page, `fastqc` can be run from the command line or from a graphic user interface (GUI). Using a GUI is generally intuitive, so today we will look at the command line usage, as that will give you more flexibility & options going forward. Some important options for the command can be seen in the manual.

As you will see in the manual, setting the `-h` option as above will call the help page. Look up the following options to find what they mean.

| **Option** | **Usage** |
|---|---|
| -o | |
| -t | |
| --casava | |

Now we have two experiments, so firstly lets run the two paired-end whole-genome sequencing files. First we need to create the output directory, then we can run `fastqc` using 2 threads which will ensure the files are processed in parallel. This can be much quicker when dealing with large experiments.

```
cd ~/rawData/WGS
mkdir -p ~/QC/WGS/rawData
fastqc -o ~/QC/WGS/rawData -t 2 WGS_SRR065388_1.fastq.gz \
    WGS_SRR065388_2.fastq.gz
```

It's probably a good idea to scribble a note next to each line if you didn't understand what you did. If you haven't seen the command `mkdir` before, check the help page

```
man mkdir
```

The above command gave both files to fastqc, told it where to write the output (`-o ~/QC`) & requested two threads (`-t 2`). The reports are in the html files, with all of the plots stored in the zip files. To look at the QC report for each file, we can use `firefox`.

```
cd ~/QC/WGS/rawData
ls -lh
firefox WGS_SRR065388_1_fastqc.html WGS_SRR065388_2_fastqc.html &
```

The left hand menu contains a series of click-able links to navigate through the report, with a quick guideline about each section given as a tick, cross of exclamation mark.

Two hints which may make your inspection of these files easier are:

1. To zoom out in `firefox` use the shortcut Ctrl-. Reset using Ctrl0 and zoom in using Ctrl+
2. You can open these directly from a traditional directory view by double clicking on the .html file.

If your terminal seems busy after you close `firefox`, use the ’Ctrl C’ shortcut to stop whatever is keeping it busy

How many reads are there in the fastq file?
How long are the reads in both of the files?

## Interpreting the FASTQC Report

As we work through the QC reports we will develop a series of criteria for filtering and cleaning up our files. There is usually no perfect solution, we just have to make the best decisions we can based on the information we have. Some sections will prove more informative than others, and some will only be helpful if we are drilling deeply into our data. Firstly we'll just look at a selection of the plots.

### Per Base Sequence Quality

Both of the files should be open in `firefox` in separate tabs. Perform the following steps on both files. Click on the `Per base sequence quality` hyperlink on the left of the page & you will see a boxplot of the QC score distributions for every position in the read. This is the main plot that researchers will look at for making informed decisions about later stages of the analysis.

What do you notice about the QC scores as you progress through the read?

We will deal with trimming the reads in a later section, but start to think about what you should do to these reads to ensure the highest quality in your final alignment & analysis.

**Per Tile Sequence Quality** This section just gives a quick visualisation about any physical effects on sequence quality due to the tile within the each flowcell. Generally, this would only be of note if drilling deeply to remove data from tiles with notable problems. Most of the time we don't factor in spatial effects, unless alternative approaches fail to address the issues we are dealing with. It can occasionally be used to request a repeat sequencing run, although if this has been a factor, it is likely that other plots will be more compelling.

**Per Sequence Quality Scores** This is just the distribution of average quality scores for each sequence. There's not much of note for us to see here.

**Per Base Sequence Content** This will often show artefacts from barcode sequences or adapters early in the reads, before stabilising to show a relatively even distribution of the bases.

**Sequence Duplication Levels** This plot shows about what you'd expect from an whole genome experiment. There are a few duplicated sequences lots of unique sequences representing the diverse genome. While this FastQC report was calculated on two small files (subset from a much larger sequencing run), each FastQC analysis is calculated on a small sample of the library for computational efficiency. However this will give a rough guide if anything unusual stands out

**Kmer Content** Statistically over-represented sequences can be seen here & often they will overlap. In our first plot, the some sequences derive from the same motif, and are the extracts of the same longer sequence, just shifted along one base. No information is given as the source of these sequences, and you would expect to see barcode sequences or motifs that correspond to any digestion protocols here. This is a plot that can cause significant confusion, but can alert you to any unexpected sequence-based problems in your data.

## A Different Dataset

The whole-genome sequencing dataset is relatively ”well-behaved”, so let's look at another dataset which is a little different. Let's run `fastqc` & inspect an RNA-Seq dataset.

```
cd ~/rawData/small_RNA
mkdir -p ~/QC/small_RNA/rawData
fastqc -o ~/QC/small_RNA/rawData -t 2 SmallRNA_SRR2039540.fastq.gz
cd ~/QC/small_RNA/rawData
firefox SmallRNA_SRR2039540_fastqc.html &
```

The RNAseq dataset that we will run `fastqc` on is a small-RNA dataset, the library containing all the small RNA sequences expressed by the sample at one time-point. This can contain a number of different types of RNA molecules, such as:

- microRNA (miRNA)
- piwi-interacting RNA (piRNA)
- small interfering RNA (siRNA)
- small nucleolar RNA (snoRNAs)
- tRNA-derived small RNA (tsRNA)
- small rDNA-derived RNA (srRNA)
- small nuclear RNA, also commonly referred to as U-RNA

All these RNA types are less than 200bp, however the majority are likely to be microRNAs and piRNAs that are less than 30bp.

What is the read length of this sequencing run?
What sort of biases or contamination can we expect with this small RNA sequencing run, given that the DNA inserts are likely to be > 30bp?

**Overrepresented Sequences** Go to the *over-represented* sequences section for Small-RNA_SRR2039540.fastq.gz. There seem to be a large number of over-represented sequences which seem to contain Illumina Single End Adapter 1. These sequences look to be almost > 26% of all sequences! You will also notice that all the other over-represented sequences are similar to Illumina Single End Adapter 1 and only differ by a few SNPs. The top 5 over-represented sequences looks extremely similar:

```
TGAGATCGTTCAGTACGGCAATCGTATGCCGTCTTCTGCTTGAAAAAAA
TGAGATCGTTCAGTACGGCAATTCGTATGCCGTCTTCTGCTTGAAAAAA
TGACTAGAGACACATTCAGCTTCGTATGCCGTCTTCTGCTTGAAAAAAA
CACCCGTACATATGTTTCCGTGCTTCGTATGCCGTCTTCTGCTTGAAAA
```

Most of the sequences show a similar pattern, starting with ”TGAG” type sequence and finishing with a ”TCTTCTGCTTG” motif and a odd poly-adenine stretch of bases. Each of these sequences are also 49-50bp, exactly the same as our read length.

**Adapter Contamination** Curiously however when we look at the *Adapter Content* section of the report we see no sign of significant adapter contamination. By the time you reach the 60th nucleotide in the reads from Sample1, this plot tells you that > 50% of reads contain matches to the Illumina Universal Adapter.

**Per base GC content** Looking at the "Per base GC content" and there seems to be a massive increase in sequences at a very specific GC content. While the peak number of reads from theoretical distribution of the dataset should be around 20,000 reads > 43%, the number of reads at that peak has more than doubled

**Per base sequence content** If you look at the "Per base sequence content" section, this plot looks all over the place! Across the X-axis (base-pair position on the read) we see massive spikes in each nucleotide content. Starting from base-pair 1, Thymine looks to be found in > 70% of the sequences, followed by Guanine in base 2 and Adenine for base 3. This is quite consistent with our over-represented sequences above.
Additionally, at the end of the read, we see that Adenine is the majority of all base's from 41-50bp, also consistent with our over-represented sequences.

What is the explaination of the spike in the number of reads at > 43% and the > 60% of each nucleotide across the read?
What possible library preparation issue has caused the pattern seen in this FastQC report?

Interpreting the various sections of the report can take time & experience. A description of each of the sections is available from the `fastqc` authors at http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/

Another interesting report is available at http://www.bioinformatics.babraham.ac.uk/projects/fastqc/RNA-Seq_fastqc.html Whilst the quality scores generally look pretty good for this one, see if you can find a point of interest in this data. This is a good example, of why just skimming the first plot may not be such a good idea.

In our dataset of two samples it is quite easy to think about the whole experiment & assess the overall quality. What about if we had 100 samples? Each .zip archive contains text files with the information which can easily be parsed into an overall summary.

Whilst this will require low-level scripting skills to perform on an experiment, we can quickly look at two of the important files today. The overall summary in terms of PASS/FAIL is contained in the 'summary.txt' file within the archive. Open this file in the `less` pager, and once you've had a look type `q` to quit, as we have become familiar with. First make sure you are in the correct directory, then inspect these files using `less`.

```
cd \~/QC/small\_RNA/rawData
unzip -oc SmallRNA_SRR2039540_fastqc.zip '*summary.txt' | less
```

The raw numbers for each of the sections are in the file fastqc_data.txt. Page through the file, until you lose interest then quit the pager.

```
unzip -oc SmallRNA_SRR2039540_fastqc.zip '*fastqc_data.txt' | less
```

We can also extract any specific image file for compiling into a pdf, or find whatever we need by using these ideas. This makes handling the data for a large experiment much simpler. There are plenty of hints online for how to write a *shell script*, or alternatively, attend one of our scripting workshops.

# Further Reading

An excellent article which deals with some of the issues around data quality is:

Zhou, X and Rokas, A. (2014). *Prevention, diagnosis and treatment of high-throughput sequencing data pathologies.* Molecular Ecology 23, 1679-1700.

This has been included on your VM as the file QC.pdf & contains many examples of good data and low quality data, as well as a detailed discussion. If you feel like you are running ahead of schedule, or if you finish early it may be a good opportunity to download & read through the article. The workflow given at the end may also be particularly useful.

# Trimming, Filtering and Pre-Processing NGS Data

Primary Author(s):
Jimmy Breen, Robinson Research Institute & Bioinformatics Hub, University of Adelaide
jimmy.breen@adelaide.edu.au
Steve Pederson, Bioinformatics Hub, University of Adelaide
stephen.pederson@adelaide.edu.au

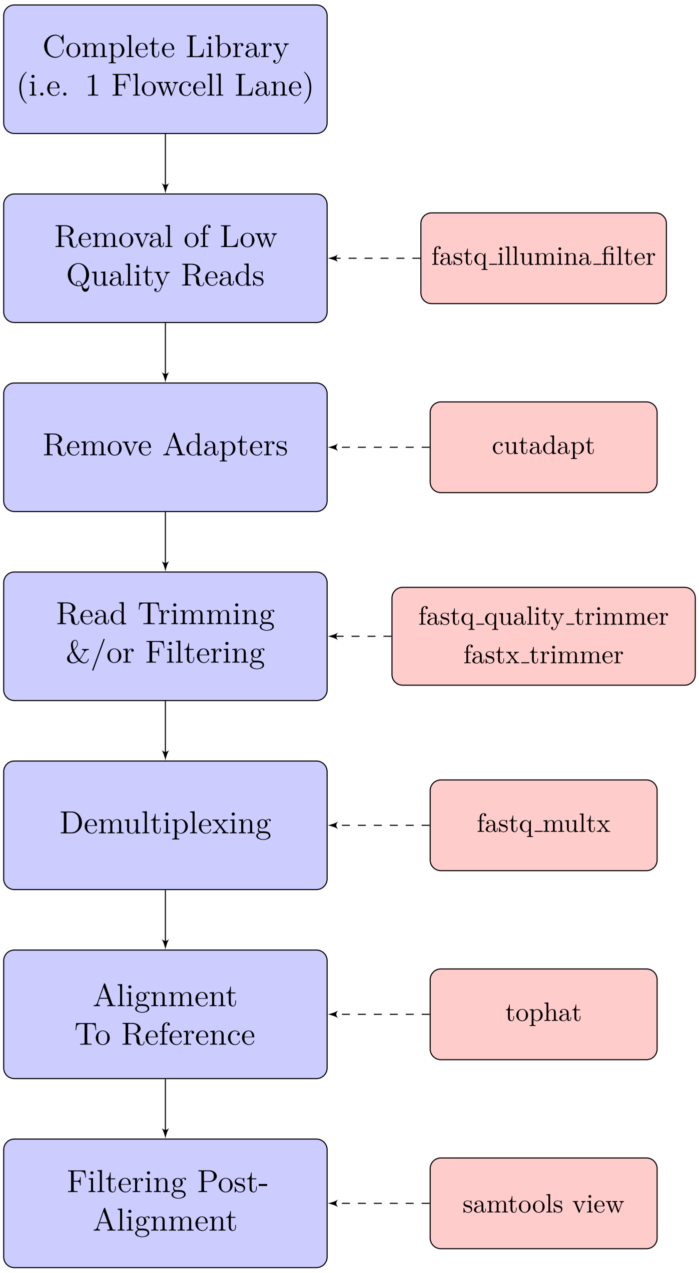
Contributor(s):
Dan Kortschak, Adelson Research Group, University of Adelaide
dan.kortschak@adelaide.edu.au
Terry Bertozzi, SA Museum Terry.Bertozzi@samuseum.sa.gov.au

Once we have inspected our data & have an idea of how accurate our reads are, as well as any other technical issues that may be within the data, we may need to trim or filter the reads to make sure we are aligning or analysing sequences that accurately represent our source material. As we've noticed, the quality of reads commonly drops off towards the end of the reads, and dealing with this behaviour is an important part of most processing pipelines. Sometimes we will require reads of identical lengths for our downstream analysis, whilst other times we can use reads of varying lengths. The data cleaning steps we choose for our own analysis will inevitably be influenced by our downstream requirements.

# The Basic Workflow

Data cleaning & pre-processing can involve many steps, and today we will use the basic work-flow as outlined in the following flow chart. Every analysis is slightly different so some steps may or may not be required for your own data. Some steps do have a little overlap, and some pipelines (e.g. *Stacks*) may perform some of these steps for you.

Using today's datasets, we will take one sequencing through demultiplexing and adapter removal, and then use our C. elegans WGS to run genome mapping and alignment filtering. We will perform most steps on files at this stage, rather than on a complete library, but the principle is essentially the same.


Complete Library (i.e. 1 Flowcell Lane)
Removal of Low Quality Reads
fastq_illumina_filter
Remove Adapters
cutadapt
Read Trimming &/or Filtering
fastq_quality_trimmer
fastx_trimmer
Demultiplexing
fastq_multx
Alignment To Reference
tophat
Filtering Post-Alignment
samtools view

# Demultiplexing

In the previous section ”Understanding NGS Data & FASTQ Format” we discussed the difference between an *index* and a *barcode*. If you use an indexed adapter to distinguish samples on an Illumina sequencing run, the demultiplexing is *usually* done on the sequencing machine. However, sometimes it makes sense to use a barcode (or sometimes called ”inline barcode”), to further multiplex samples onto one sequencing run.

While barcode can be incredibly useful, it is important to note that Illumina cycle calibration and cluster calling is done in the first 4 cycles (first four base-pairs of read 1). It also is used to establish other metrics (e.g., signal thresholds) for base-calling. Therefore it is essential that the first four base pairs are ”diverse” (i.e. no particular nucleotide is over-represented in the first four base-pairs). Designing the right barcodes to add to the start of your reads extremely important!
Additionally, the Illumina NextSeq machine’s have a slighly different sequence setup to the other sequencing machines (MiSeq, HiSeq and GAII’s), using two-channel sequencing, which requires only two images to encode the data for four DNA bases, one red channel and one green channel. Guanine is an absence of colour and therefore at least one base other than G most be present in the first two cycles (http://blog.kokocinski.net/index.php/barcode-balancing-for-illumina-sequencing?blog=2).
For more information on Illumina cluster density and other technical aspects of cycles and imaging, read the following Illumina support material https://support.illumina.com/content/dam/illumina-marketing/documents/products/other/miseq-overclustering-primer-7
pdf.

To demonstrate demultiplexing we will use the a sequencing run with two samples that have a 7bp barcode. Our barcode sequences should be ”GCGTAGT” (for sample1) and ”CCTCGTA” (for sample2). Lets first see what possible barcodes are available in the first 7bp of our dataset and see if it matches what we expect:

```
cd ~/rawData/Multiplexed
zcat Run1_R1.fastq.gz | sed -n '2~4p' | cut -c 1-7 | sort | uniq -c | \
    sort -nr | head -n10
```

What top 5 barcodes are found in our data? Do the top two reflect our the barcodes we should have?
The command above is quite long and contains multiple unix commands that are separated by a pipe. What does each command do?

Our barcodes are actually in a file called barcodes

```
cd ~/rawData/Multiplexed/
gunzip Run1_R1.fastq.gz
gunzip Run1_R2.fastq.gz
sabre pe -m 1 -f Run1_R1.fastq -r Run1_R2.fastq -b barcodes_R1.txt \
-u unknown_barcode1.fastq -w unknown_barcode2.fastq
```

How many read pairs were extracted in each sample?
Run the command again without the one mismatch. How many read are now in each?

## Removal of Low Quality Reads and Adapters

Adapter removal is an important step in many sequencing projects, mainly projects associated with small DNA/RNA inserts. For example, a common RNAseq experiment is to sequence small non-coding RNAs that are generated by an indvidual to regulate other coding sequences. These small RNAs (namely miRNAs, snoRNAs and/or siRNAs) are generally between 19-35bp, which is significantly smaller than the shortest, standard read length of most Illumina MiSeq/NextSeq/HiSeq machines (usually have read length settings such as 50, 75, 100, 125, 250 or 300bp). Therefore it is important to trim adapters accurately to ensure that the genome mapping and other downstream analyses are accurate.

In previous workshops we would run multiple steps to remove both low-quality reads, but today's trimming algorithms have become better at removing low-quality data and the same time as removing adapters.

The tool we'll use today is `cutadapt` & it's one of the few bioinformatics tools to have a helpful webpage, so head to the site http://cutadapt.readthedocs.org/.

## Paired-End Data

In the above process, we removed the adapters from each file separately. What did the setting `-m 20` specify?
Could this cause any problems for paired-end reads, where each fastq file must contain exactly matching reads?

The more recent versions of `cutadapt` allow for trimming sets of paired reads. Check the website for the correct code at http://cutadapt.readthedocs.org/en/stable/guide.html#trimming-paired-end-reads.

Now we can trim the data our demultiplexed data using the Illumina Nextera paired-end adapters. This are commonly used in Illumina projects. The index in the 3' adapter of the first read, which is used to demultiplex on the sequencing machine, we need block the sequence otherwise it can cause problems in the adapter matching. To do this, we use a sequence of "N" nucleotides.

```
cd ~/rawData/Multiplexed
mkdir -p ~/rawData/Multiplexed/trimmedData
cutadapt -a \
    AGATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATCTCGTATGCCGTCTTCTGCTTG \
    -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATT \
    -o ~/rawData/Multiplexed/trimmedData/Sample1_r1_trim1.fastq \
    -p ~/rawData/Multiplexed/trimmedData/Sample1_r2_trim2.fastq \
    Sample1_r1.fastq Sample1_r2.fastq
```

Now lets do this with Sample2.

When running the adapter trimming we needed to add blocking ”N” nucleotides to the adapter sequence. Why do we do this?
How many final adapter trimmed reads were found in Sample1 and Sample2?
The cutadapt program produces a large amount of information about the trimming process, especially regarding the sequence length distribution of the output reads. To the nearest 10bp (i.e. 10-20 or 110-120), what would be the modal peak of each Sample (the 10bp containing the most number of sequences)?

# Aligning Reads To a Reference Sequence

Primary Author(s):
Jimmy Breen, Robinson Research Institute & Bioinformatics Hub, University of Adelaide
jimmy.breen@adelaide.edu.au
Steve Pederson, Bioinformatics Hub, University of Adelaide
stephen.pederson@adelaide.edu.au

Contributor(s):
Dan Kortschak, Adelson Research Group, University of Adelaide
dan.kortschak@adelaide.edu.au
Terry Bertozzi, SA Museum Terry.Bertozzi@samuseum.sa.gov.au

Once we have cleaned our data of any contaminating sequences, and removed the bases which are more likely to contain errors we can more confidently align our reads to a reference. Different experiments may have different reference sequences depending on the context. For example, if we have a sub-sample of the genome associated with restriction sites like RAD-Seq, we would probably align to a reference genome, or if we have RNA-Seq we might choose to align to the transcriptome instead of the whole genome. Alternatively, we might be interested in *de novo* genome assembly where we have no reference genome to compare our data to.

For the next section, we are going to align our C. elegans WGS data to our genome without trimming. In the context of the FastQC reports, is this adviseable?

## How Aligning Works

Most fast aligners in widespread public use are based on a technique called the *Burrows-Wheeler Transform*, which is essentially a way of restructuring, or indexing, the genome to allow very rapid searching. This technique comes from computer science & is really beyond the scope of what most of us need to know. The essence of it is that we have a very fast searching method, and most aligners use a seed sequence within each read to begin the searching. These seeds are then expanded outwards to give the best mapping to a sequence. There are many different alignment tools available today & each one will have a particular strength. For example, `bowtie` is very good for mapping short reads, whilst `bowtie2` or `bwa` is more suited to reads longer than 50bp.

### What's the difference

Some key differences between aligners is in the way they index the genome, and in the way they are equipped to handle mismatches & indels. Choosing an aligner can be a difficult decision with the differences often being quite subtle. Sometimes there is a best choice, other times there really isn't. Make sure you've researched relatively thoroughly before deciding which to use.

# Aligning our WGS reads

### Downloading A Reference Genome

To align any reads, we first need to download the appropriate (i.e. latest) genome & then we can build the index to enable fast searching via the Burrows-Wheeler Transform. Like we've seen in the previous sections, our reads today come from the nematode or Roundworm *Caenorhabditis elegans*. We have a copy of the genome read for you in the rawData directory, however you can always redownload the genome (like you can do with all model genomes) by opening Firefox & head to `ftp://ftp.ensembl.org/pub/release-88/fasta/caenorhabditis_elegans/`. (Here you can find the transcriptome (*cdna*), genome (*dna*), proteome (*pep*) & non-coding RNA (*ncrna*) in separate folders).

With our C. elegans genome fasta file, we should have a quick look at the file. It will contain all chromosomes & the mitochondrial sequences.

We can print out the first 10 lines using the `head` command.

```
cd ~/rawData/
gunzip cel1.fa.gz
head cel1.fa
```

Note that the first line describes the following sequence & begins with a > symbol. We can use this to search within the file using *regular expressions* & print all of these description lines.

```
grep '^>' cel1.fa
```

Alternatively could simply count them using the `-c` option.

```
grep -c '^>' cel1.fa
```

How many chromosomes does C. elegans have?

### Building an Index

We will align using the tool `bwa` which is one of the original Burrows-Wheeler transformation mappers. `bwa` was developed in 2009 by Heng Li (Harvard/Broad Institute, USA). From the `bwa` manual, it details the three specific `bwa` algorithms:
*BWA-backtrack*, *BWA-SW* and *BWA-MEM*. The first algorithm is designed for Illumina sequence reads up to 100bp, while the rest two for longer sequences ranged from 70bp to 1Mbp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM, which is the latest, is generally recommended for high-quality queries as it is faster and more accurate."

Today we will be using *bwa-mem* to align our C. elegans WGS reads.
Once again, we need to check the help pages. Fortunately the `bwa` page is actually pretty friendly on the screen and appears without the usual `-h` option.

```
bwa
```

We should also inspect the help page for `bwa index` which we will use to build the index.

```
bwa index
```

Using this particular process you can usually just run the command on the fasta file and the index will be called by the same filename. However in this case, we will name the index "cel1" (thats cel for *C. elegans*) and "1" for the version, by using the -p flag/parameter

Now that we've had a look, type to following command which will take a few minutes to run.

```
bwa index -p Cel1 cel1.fa
```

Let's look at what files have been created.

```
ls
```

You should be able to open a few of the files with the "less" command, however the main files (*.sa, *.bwt and *.pac) are the BWT transformed files that are in binary, so we can't really see what they look like, but these are required by the aligner *bwa*.

## Aligning the reads

Because we only have a small subset of the actual sequencing run, we should be able to run this alignment in a reasonable period of time

```
bwa mem Cel1 WGS_SRR2003569_1.fastq.gz WGS_SRR2003569_2.fastq.gz \
        | samtools view -bhS -> WGS_SRR2003569.bam
```

Let's break down this command a little. The first part of the command:

```
bwa mem Cel1 WGS/WGS_SRR2003569_1.fastq.gz WGS/WGS_SRR2003569_2.fastq.gz
```

will align our compressed sequenced reads to the Cel1 bwa index that we made. Usually you can create a SAM file (see next section) to store all the alignment data. SAM files however are text files which can take up a significant amount of disk space, so its much more efficient to pipe it to the *samtools* command and create a compressed binary SAM file (called BAM). To do this, we run the program *samtools*:

```
samtools view -bhS - > WGS_SRR2003569.bam
```

In this context, *samtools view* is the general command that allows the conversion of the SAM to BAM. There is another more compressed version of the SAM file, called CRAM, which you can also create using *samtools view*. However, we will not use that today.

Although we haven't covered the deeper technicalities of aligning reads to the genome today, consider the following scenario:

During data generation, an insertion has occurred which makes a read align with an altogether different genomic region. How can this be corrected so that it aligns to it's original sequence?

# Working With Alignments

Primary Author(s):
Jimmy Breen, Robinson Research Institute & Bioinformatics Hub, University of Adelaide
jimmy.breen@adelaide.edu.au
Steve Pederson, Bioinformatics Hub, University of Adelaide
stephen.pederson@adelaide.edu.au

Contributor(s):
Dan Kortschak, Adelson Research Group, University of Adelaide
dan.kortschak@adelaide.edu.au
Terry Bertozzi, SA Museum Terry.Bertozzi@samuseum.sa.gov.au

# The SAM/BAM file format

Reads that have been aligned to a reference are no longer stored in fastq format but are stored in either SAM or BAM format. These two formats are virtually identical, however the SAM format is a text file which is easily readable to human eyes, whilst a BAM file is the same information converted to binary. This conversion means that file sizes are smaller, and that computational processes can be performed more efficiently. Typically, we work with BAM files as these provide considerable gains in storage space & analytic speed. The tools we use to inspect these files are provided in the package *samtools*, which has been installed on your VM. For this section we will only look at BAM files, removing the need to convert all our files between the two formats.

The reads from the previous dataset which mapped to *C. elegans* should be in the folder `~/rawData/WGS`. Let's look at the first 10 lines of the file.

```
cd ~/rawData/WGS
samtools view -h WGS_SRR065388.bam | head -n10
```

Notice that when we want to look at the BAM file, we always need to call *samtools view* first. The *samtools view* command is essentially the zcat/cat command that we used in previous sections. We use it to look at the BAM file in much the same way as cat/less/more with text files, except the file remains in binary and a compressed format.

FYI: If you want to convert between SAM and BAM formats you can use these commands:

```
# From BAM to SAM
samtools view -h WGS_SRR065388.bam > WGS_SRR065388.sam

# From SAM to BAM
samtools view -bhS -o WGS_SRR065388.bam WGS_SRR065388.sam
```

Can you distinguish between the header of the SAM format and the actual alignments? What kind of information does the header provide?

# Handling BAM files

As BAM files are in binary format they will look like gibberish if we try to read them directly. Instead we can inspect them by using `samtools view` again. To view the header we use:

```
samtools view -H WGS_SRR065388.bam
```

Whereas to look through the first few mapped reads, we can use:

```
samtools view WGS_SRR065388.bam | head
```

As this data can easily spill across lines, it might be helpful to maximise your terminal to see the complete line structure.

Why didn't we need to set the `-b` or `-S` option when running `samtools view` in the previous two lines?

## The SAM/BAM data structure

If we understand what information is contained within a file, we can know what decisions to make as we progress with our analysis, so let's have a look at what the data structure is for a SAM/BAM file. A SAM file is *tab-delimited*, which means that each field is separated by a tab, giving a data structure effectively consisting of columns (or fields). In order, these are:

| | | |
|---|---|---|
| 1 | QNAME | Query template/pair NAME |
| 2 | FLAG | bitwise FLAG |
| 3 | RNAME | Reference sequence NAME |
| 4 | POS | 1-based leftmost POSition/coordinate of clipped sequence |
| 5 | MAPQ | MAPping Quality (Phred-scaled) |
| 6 | CIGAR | extended CIGAR string |
| 7 | MRNM | Mate Reference sequence NaMe ('=' if same as RNAME) |
| 8 | MPOS | 1-based Mate POSistion |
| 9 | TLEN | inferred Template LENgth (insert size) |
| 10 | SEQ | query SEQuence on the same strand as the reference |
| 11 | QUAL | query QUALity (ASCII-33 gives the Phred base quality) |
| 12+ | OPT | variable OPTional fields in the format TAG:VTYPE:VALUE |

Several of these fields contain useful information, so looking the the first few lines which we displayed above, you can see that these reads are mapped in pairs as consecutive entries in the QNAME field are often (but not always) identical. Most of these fields are self-explanatory, but some require exploration in more detail.

## SAM Flags

These are quite useful pieces of information, but can be difficult at first look. Head to `http://broadinstitute.github.io/picard/explain-flags.html` to see a helpful description. The simplest way to understand these is that it is a bitwise system so that each description heading down the page increases in a binary fashion. The first has value 1, the second has value 2, the third has value 4 & so on until you reach the final value of 2048. The integer value contained in this file is the unique sum of whichever attributes the mapping has. For example, if the read is paired & mapped in a proper pair, but no other attributes are set, the flag field would contain the value 3.

What value could a flag take if it had the following set of properties?

1. the read was paired;
2. the read was mapped in a proper pair
3. the read was was the first in the pair; &
4. the alignment was a supplementary alignment

Some common values in the bam file are 99, 147 & 145. Look up the meanings of these values.

Things can easily begin to confuse people once you start searching for specific flags, but if you remember that each attribute is like an individual flag that is either on or off (i.e. it is binary). If you searched for flags with the value 1, you wouldn't obtain the alignments with the exact value 1, rather you would obtain the alignments for which the first flag is set & these can take a range of values.

Let's try this using the command `samtools view` with the option `-f N` to include reads with a flag set and the option `F N` to exclude reads with a specific flag set. Let's get the first few reads which are mapped in a proper pair, so the flag '2' will be set.

```
samtools view -f 2 WGS_SRR065388.bam | head
```

Note that none of the flags actually have the value 2, but if you typed the values 99, 147 or 163 into the web page, you'll see that this flag is set for all of these values. Similarly if we wanted to extract only the reads which are NOT mapped in a proper pair we would change the option to a upper-case F.

```
samtools view -F 2 WGS_SRR065388.bam | head
```

Again, try entering a few of these sample values into the web page and you will see that this flag is not set for any of these values.

This can be a very helpful tool for extract subsets of your aligned reads. For example, we can create a new BAM file with only the reads which were aligned in a proper pair by entering the following command.

```
samtools view -f 2 -bo WGS_SRR065388.paired.bam WGS_SRR065388.bam
ls -lh
```

You can pull out highly specific combinations of alignments should you so choose

The program *samtools* also has some handy commands that give you information about the data presented in the BAM file. We can actually get information about the number of reads that have particular flags with the *samtools flagstat* command

```
samtools flagstat WGS_SRR065388.bam
```

Newer version of *samtools* also have a stats command, which gives you a tonne of information about reads within the BAM file

## CIGAR strings

These give useful information about the type of alignment that has been performed on the read. In the first few reads we called up earlier, most had the value `35M`, which stands for 37 **M**atches. The other abbreviations in common use are I (insertion) & D (deletion), however these are not found in this bam file.

A CIGAR string contained in this file is ‘88M12S’. Can you think of an interpretation of this string?
How many reads were mapped and paired in this dataset?

# Viewing the Alignments

In this final section we will use the package IGV Viewer to inspect the alignments in this bam file. We are going to use the IGV genome browser, which has been installed on your VM. This is just a simple tool for manually inspecting your alignments, and getting a feel for what levels of coverage you have in your data, or if there are any strange artefacts.

## Sorting and Indexing Our BAM Files

To visualise our alignments, we first need to sort our bam file so that all the alignments are in order based on each chromosome, instead of the order they appeared in the source fastq file. This allows much faster searching of the files, and also allows for better compression. The second step will be to index the sorted file to enable faster file access by the browser.

```
cd ~/rawData/WGS
samtools sort WGS_SRR2003569.bam -o WGS_SRR2003569.sorted.bam
samtools index WGS_SRR2003569.sorted.bam
```

The command `samtools index` then created a file with the exact same name as the input file, but with the extra suffix `.bai`, which indicates this is an index file.

Now that we've got our files ready to view we can load the IGV browser. This may take a minute or two so please be patient.

```
/opt/local/IGV_2.3.90/igv.sh
```

Once the browser has opened, go to the drop-down menu & load the genome for *Caenorhabditis elegans* (cel1). (This will create a .genome file in your `~/igv` folder.) Now we can load our sorted .bam file using *File > Load from File* in the top menu, and make sure you load the sorted .bam file. This will automatically check for the indexed file we created a minute ago.

In the second "button" go to the region chrI, then enter the word *del-1* in the window next to this. This will take us to the gene *del-1* and we will be able to see the reads in our .bam file that have been aligned to this gene. Unfortunately with these alignments, the further out we zoom we will begin to lose our alignment information.

Have a look at the gene *elf-1* as well, and have a look around by clicking the navigation at the very top of the graphical display.

# Bonus For Those Well Ahead of the Pack

For those who still feel like they want more, try trimming this dataset to see how it differs from the unclipped analysis. Record your commands in the `nano` text editor so you have a record of what you did.

# Space for Personal Notes or Feedback