Introduction To NGS Data & Analytic Tools

Steve Pederson

Bioinformatics Hub
University of Adelaide

March 30, 2016

Licensing

Contents

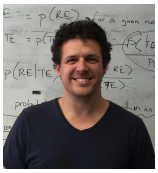
Workshop Information

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

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 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:

```
1 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:


Important


For reference


Follow these steps


Questions to answer


Warning - STOP and read


Bonus exercise for fast learners


Advanced exercise for super-fast learners

Computer Setup

We will all be working on our own computers today, and will be accessing Virtual Machines running the Ubuntu operating system on the Nectar Research Cloud (http://nectar.org.au/). The software client No Machine which you will have already installed, enables us to access these machines in a familiar Desktop style, even though the majority of our time will be spent within the terminal.

If you did not install NoMachine prior to today's workshop, the correct installers can be found at:

- http://tinyurl.com/lcn3ns8 (Windows)
- http://tinyurl.com/mnm23ew (Mac)

No Machine session files will be provided to each attendee which have been pre-configured to enable easy access to these machines. These machines are effectively dual-core machine with 8GB of RAM & 70GB of hard drive space. Whilst relatively small, this will be more than enough to become familiar with the important concepts for the day. To begin today's session, simply click (or double-click) on the No Machine session file that you have been given. The desktop from the Virtual Machine (VM) that you have connected to will appear on the No Machine client.

While this is connecting, select the button circled in blue above. This will resize NoMachine to be full screen to feel more like you are physically located at the VM. If any security warnings appear, ignore them & continue connecting. This may take several attempts and if you can't connect after 3 or 4 times, call a tutor over.

If you forgot to set the client to full screen, hover your mouse over the top-right corner and click where the 'page' folds over. This will enable you to resize the client again, and can also be used to disconnect from the VM.

The Ubuntu Desktop

Now that you are connected, you will notice we are in a standard graphical environment. The default Desktop in Ubuntu is Unity, but what we are seeing is known as the Gnome Desktop, as is the default for Nectar VMs. As many of us are used to seeing, there are click-able icons on the desktop, and drop-down menus.

The main interfaces we will be using today are the `terminal`, a text editor named `gedit` and the web browser `firefox` They will appear as icons on the desktop, but can also be accessed from the drop-down menus. A `Firefox` icon is also located on the desktop & this can be used for searching for answers as well as viewing the output of `fastqc`.

Basic Shell Comands

Today we will assume a basic familiarity with the bash terminal in Linux. Some key commands which we will be using are explained in the following table.

Command	Meaning
`cd`	Change directory
`ls`	List files in a directory
`man`	Call the manual for a given command
`head`	View the first 10 lines of a file

Open a terminal in the VM by either clicking on the icon, or using the drop-down menu. It will automatically open in your home directory (`/home/trainee`), which is indicated by the tilde before the dollar sign. The tilde (~) is a symbolic representation of the address `/home/trainee`, where the first slash is the root of the computerâĂŹs file system. To return to this directory from any other in the file system, you can simply enter the command

```
1 cd
```

Normally we specify the directory to change into after the command `cd`, but when no directory is given, this command will automatically change to your home directory.

Please note, the Linux operating system is case-sensitive. When entering the any commands, please make sure that you pay attention to this detail. Spaces are an important feature as well, so be careful not to add or omit them where they appear in any sample code.

Today's Data

Before we begin today's session, we need to download and run a script. This will make all the correct directories on your VM and download the data for today's session.

The script can be obtained from http://tinyurl.com/hkc77vf. Head to this address using `firefox` and save this on your VM as the script `getNGSData.sh`. It won't matter where, but `firefox` will probably place it in your `Downloads` directory.

Now open a terminal and head to the folder the file was saved in. The following assumes that the file was saved in your `Downloads` directory.

```
1 cd ~/Downloads
2 chmod +x getNGSData.sh
3 ./getNGSData.sh
```

This will start downloading all of the files to your VM and will create a directory `/home/trainee/NGSData` with today's data.

If you're curious open the script we've just run using gedit and see if you can understand what the script has done. You might be surprised at how much you understand.

```
1 gedit getNgsData.sh &
```

Today's Notes

Today's notes are also available electronically from the site http://uofabioinformaticshub.github.io/Intro-NGS_2016-03-31/

Understanding NGS Data & FASTQ Format

Primary Author(s):
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

Initial Goals

1. Understand the *Sequencing by Synthesis* Process & Data Generation
2. Become familiar with the `.fastq` file format
3. Understand what errors and artefacts lie within the data
4. 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 adaptors 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.

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?

Will all of the individual ssDNA molecules within a spot be amplified perfectly in sync with each other?

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 `~/NGSData/RNASeq/rawData` 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 ~/NGSData/RNASeq/rawData
zcat reads1.fq.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 reads1.fq.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 `SRR031714`. For the first sequence in this file, we have the full identifier `@SRR031714.1 HWI-EAS299_130MNEAAXX:2:1:785:591/1` which has the following components:

`SRR031714.1`	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.
`WHI-EAS299_130MNEAAXX`	The unique machine ID
`2`	The flowcell lane
`1`	The tile within the flowcell lane
`785`	The x-coordinate of the cluster within the tile
`591`	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.

```
1 zcat reads2.fq.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 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/.

An alternate set of reads which we'll also explore today is in included in the directory `~/NGSData/iCLIP/rawData`. These are immunopreciptated mRNAs and were supplied by some collaborators who'd either forgotten the removal of low-quality reads or had deliberately neglected step to increase read numbers. Note that these files are not compressed and are essentially plain text files, so we can directly inspect them using the command `head` instead of decompressing first.

```
cd ~/NGSData/iCLIP/rawData
head -n12 iCLIP_Sample1.fastq
```

Note that the header lines follow the format described on the previous page with this additional field at the end of the line.

Of the first three reads which have been printed to the terminal above, which one(s) will have failed the chastity filter, and which ones will have passed the 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 characters are non-printable & contain things like end-of-line marks and tab spacings, and note that the first printable character after the space 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 line of quality scores for the first read in the iCLIP sample above, the first character is "=", so this base has a quality score of 61. The second symbol in this line is"D", so the second base has a quality score of 68.

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:

```
.............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.....................
.................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ.....................
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 the RNA-Seq data. Which coding system do you think has been used for the RNA-Seq 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

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

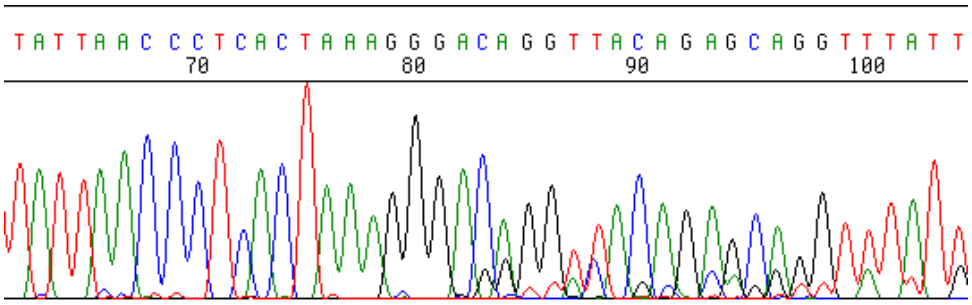
where P is the probability of calling the incorrect base.

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):
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.

```
1 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	

As we have two RNA-Seq files, we will first 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.

```
1 cd ~/NGSData/RNASeq/rawData
2 mkdir -p ~/QC/RNASeq/rawData
3 fastqc -o ~/QC/RNASeq/rawData -t 2 reads1.fq.gz reads2.fq.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

```
1 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/RNASeq/rawData
ls -lh
firefox reads1.fq_fastqc.html reads2.fq_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 both files?
How long are the reads in these 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. We'll investigate some of the others with some 'bad' data later.

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. For the first file, you will notice an even breakdown in the quality of sequences near the end of the reads across all tiles. In our second QC report, you will notice a poor quality around the 25th base in the 2nd (or 3rd) tile. 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.

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 RNA-Seq experiment. There are a few duplicated sequences (rRNA?, highly expressed genes? etc.) and lots of unique sequences represented the diverse transcriptome. This is only calculated on a small sample of the library for computational efficiency and is just to 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.

Some Flawed Data

This RNA-Seq dataset is relatively "well-behaved", so let's look at another dataset which is a bit more problematic. Let's run `fastqc` & inspect the iCLIP dataset, which is a version of RNA-Seq data where the RNA has been pulled down via IP after cross-linking to an RNA-associated protein (Argonaute).

```
mkdir -p ~/QC/iCLIP/rawData
cd ~/NGSData/iCLIP/rawData
fastqc -o ~/QC/iCLIP/rawData -t 2 iCLIP_Sample1.fastq iCLIP_Sample2.fastq
cd ~/QC/iCLIP/rawData
firefox iCLIP_Sample1_fastqc.html iCLIP_Sample2_fastqc.html &
```

There are several things to note about these reports. Firstly, we know from inspection that these fastq libraries contain reads which have failed the Illumina Chastity filter. We would expect these to show up in the *Basic Statistics* table, but they don't by default.

How would we know that there are low quality reads here if the Illumina flag is ignored?

To see how these reads look without the flagged reads, we can turn on the `--casava` option when running `fastqc`. You can ignore any warning messages you receive about not looking like part of a CASAVA group.

```
cd ~/NGSData/iCLIP/rawData
fastqc --casava -o ~/QC/iCLIP/rawData -t 2 iCLIP_Sample1.fastq \
    iCLIP_Sample2.fastq
cd ~/QC/iCLIP/rawData
firefox iCLIP_Sample1_fastqc.html iCLIP_Sample2_fastqc.html &
```

Note that the above code will have over-written the original QC reports. As this second analysis is the best approach, this is actually OK at this point. However. this can be a common pitfall when running an analysis, which is why we will often record our analysis as a script. That way we have a record of what we have done and know exactly what options we have set for a process.

Adapter Contamination Go to the over-represented sequences section for Sample1. There seem to be a large number of over-represented sequences here which are not listed as matching any known sequences. Scanning through these manually is near impossible and solving this may take some leg work. Shift over to the same section in Sample2 and you will notice about 23% of sequences seem to contain an Illumina Paired End PCR Primer. The next line also contains matches to this but with some indels.

Now head to the *Adapter Content* section of each report & consider what we are seeing here. 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.

Some More Example Reports

We'll actually try to clean the iCLIP dataset up in the next chapter, but for now let's just head to another sample plot at http://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad_sequence_fastqc.html

Per Base Sequence Quality Looking at the first plot, we can clearly see this data is not as high quality as the one we have been exploring ourselves.

Consider that the minimum sequence length required for confidently obtaining unique alignments is ¿20bp. Two approaches to this data might be to only include high quality sequences, or to trim the low quality bases from the ends and use shorter reads for downstream analysis. What would be the consequences of either approach?

Per Tile Sequence Quality Some physical artefacts are visible & some tiles seem to be consistently lower quality. Whichever approach we take to cleaning the data will more than likely account for any of these artefacts. Sometimes it's just helpful to know where a problem has arisen.

Overrepresented Sequences Head to this section of the report & scan down the list. Unlike our sample data, there seem to be a lot of enriched sequences of unknown origin. There is one hit to an Illumina adaptor sequence, so we know at least one of the contaminants in the data. Note that some of these sequences are the same as others on the list, just shifted one or two base pairs. A possible source of this may have been non random fragmentation.

Kmer Content

Do you notice anything unusual about this plot?

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/RNASeq/rawData
unzip -oc reads1.fq_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 reads1.fq_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):
Steve Pederson, Bioinformatics Hub, University of Adelaide
stephen.pederson@adelaide.edu.au

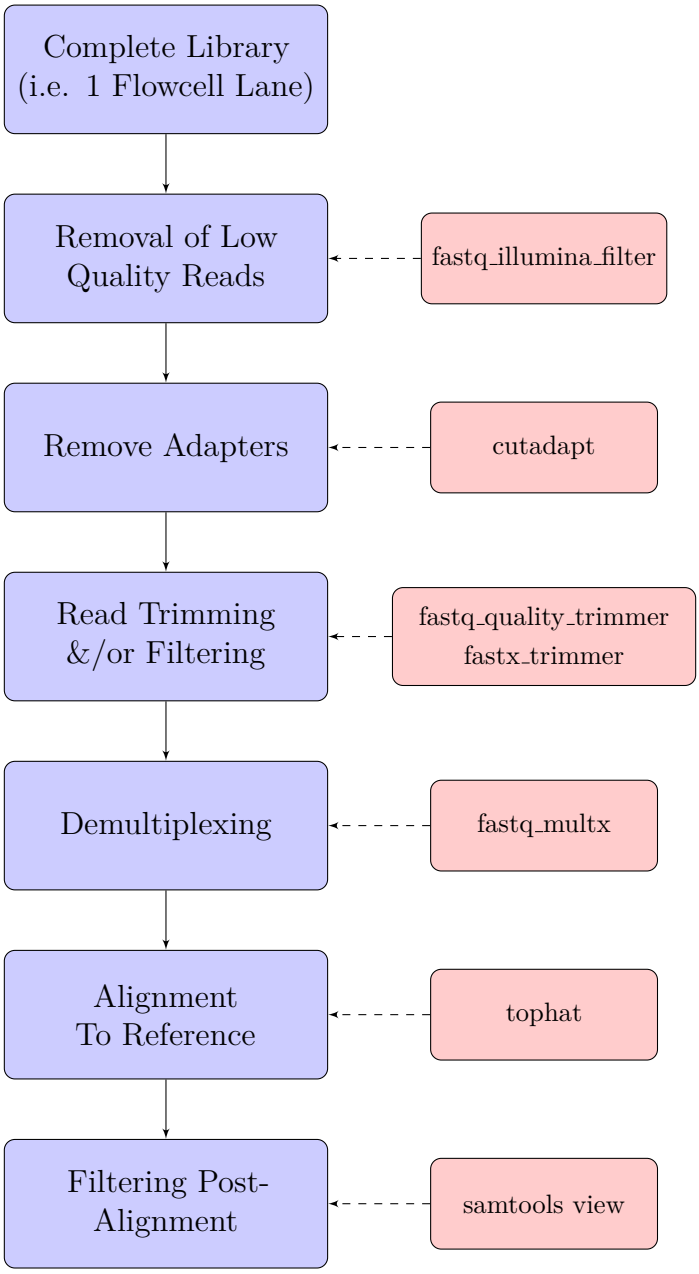
Contributor(s):
Dan Kortschak, Adelson Research Group, University of Adelaide
dan.kortschak@adelaide.edu.au
Terry Bertozzi, SA Museum Terry.Bertozzi@samuseum.sa.gov.au

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.

Unfortunately, for the purposes of keeping today's datasets manageable, we will be working on data that has already been demultiplexed. 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

Removal of Low Quality Reads

In earlier versions of the *CASAVA* pipeline, low quality reads were automatically filtered out. However, on some rare occasions researchers wanted those reads to try & extract additional information from their samples. Thus data generated using later versions of the software retains these reads and they must be removed by researchers instead of the data providers.

As discussed earlier, reads that fail will have the flag `1:Y:0` set as part of the identifier, and by switching the `--casava` option on during `fastqc`, we were able to see what the samples would look like after removal of these reads. Note, that they were not removed during this stage, but rather they were ignored whilst the QC steps were being run.

To remove these reads, we can use the tool `fastq_illumina_filter`, which simply creates a new set of `fastq` files with only the reads we wish to keep, i.e. those with the `1:N:0` flag set.

```
cd ~/NGSData/iCLIP
mkdir filteredData
cd rawData
fastq_illumina_filter -N -o ../filteredData/iCLIP_Sample1.fastq \
    iCLIP_Sample1.fastq
fastq_illumina_filter -N -o ../filteredData/iCLIP_Sample2.fastq \
    iCLIP_Sample2.fastq
```

Notice that in the above code, we used the setting `-o ../filteredData/iCLIP_Sample1.fastq`. What did this mean?

Now that we have the filtered fastq files, we should run `fastqc` on them. A sensible idea for organising the QC files is to mirror the directory structure of the fastq files. You'll also notice that these fastqc report are near identical to those produced using the `--casava` flag earlier. However, this time we have run the command on files where the low quality reads have been removed so we don't need the `--casava` option set.

```
cd ../filteredData
mkdir -p ~/QC/iCLIP/filteredData
fastqc -o ~/QC/iCLIP/filteredData -t 2 iCLIP_Sample1.fastq \
    iCLIP_Sample2.fastq
```

Adapter Removal

Looking through the fastqc files generated after removal of the chastity-filtered reads, reveals that we have some adapter sequences present in our read. These are caused by inserts that are shorter than our read lengths, and if your data has been size selected, this may not be a problem you have to worry about. However, it is very common for RNA seq experiments.

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/.

On the `cutadapt` webpage, navigate to the User Guide, then the section marked Trimming Reads. Which type of adapter contamination would give the plots seen the fastqc output files we have seen for the iCLIP data?

Single End Data

The first question to ask is, how do we find what the adapter sequence is? The fastqc reports suggested the Illumina Universal Adapter in the plots, whilst the reports for the second sample suggested it was a paired end primer. Fortunately the tool `fastqc` comes with some sequences which make a good starting point. This is contained in a sub-directory which the software was downloaded & unzipped into, which in our installation is `/opt/FastQC/Configuration`.

Change into the directory & see what files are there:

```
cd /opt/FastQC/Configuration
ls -lh
```

The two files of interest are called `adapter_list.txt` and `contaminant_list.txt`. To find the sequences associated with our potential adapters, we can look in these files.

```
egrep 'Universal' adapter_list.txt
egrep 'Paired End PCR' contaminant_list.txt
```

Now we have some potential adapter sequences we can look in our reads & see if they're prevalent. First we'll look for the Universal Adapter & then the first few nuecleotides from the Paired End PCR Primer. In the code below the option `-m10` is telling `egrep` to only give us the first 10 matches.

```
cd ~/NGSData/iCLIP/filteredData
egrep -m10 'AGATCGGAAGA' iCLIP_Sample1.fastq
egrep -m10 'CAAGCAGAAGA' iCLIP_Sample1.fastq
```

Using `egrep -c`, find how many reads match these sequences in each of the samples. (NB: You won't need to set the option -m10 for this step)

Inspect the matches carefully. Have we found our contaminant?

Fortunately for this dataset the protocols were available, and inspecting the P3 primer reveals this as the potential contaminant. Now we have identified the culprit we can set about removing this sequence from our reads. Before we decide to remove this contamination, we also need to consider whether we throw away any reads which are too short, or which have low quality scores.

In the following code, what is the minimum length that we have selected and what is the minimum quality score?

Don't forget that the appearance of the \ symbol at the end of the line is just to let you know that the printer has manually broken a single line of code.

```
1 cd ~/NGSData/iCLIP
2 mkdir trimmedData
3 cutadapt -q 20 -m 20 -a 'AGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCG' -o \
    trimmedData/iCLIP_Sample1.fastq filteredData/iCLIP_Sample1.fastq > \
    cutadapt.Sample1.log
```

Repeat this process for the second sample, and run fastqc on them both. To do this just use the arrow up in the terminal to get the previous command, then change all of the occurrences of `Sample1` to `Sample2`.

Did this solve our adapter contamination problem?

Inspect the log files that we have created using the commands `cat, head` or `less`. These can be very helpful and alert us to if we haven't removed all the adapter sequence. For example, if you see a message telling you that 80% of adapters had a 'T' beforehand, that may be a sign that we haven't entered the complete adapter sequence.

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.

Read Trimming &/or Filtering

Now that we have removed the worst of the reads, we need to sort through the remaining reads and decide on what is the most appropriate for our particular experiment. Read trimming can be done in a variety of different ways, so choose a method which best suits your data. Some analytic protocols require fixed length reads, whilst others can accept varying lengths and this can affect your approach as well. Here we will give examples of fixed-length trimming and quality-based trimming.

Fixed Length Trimming

One method of improving the quality of the base calls in your dataset is to remove the low quality bases at the end using fixed-length trimming, via a tool such as `fastx_trimmer` from the FASTX-Toolkit. It is worth noting that reads shorter than 20bp can be increasingly difficult to map uniquely to the reference genome, so keeping any trimmed reads longer than this is usually a more ideal strategy. We will write the trimmed data as a new file so we should create a new directory to write this modified data to first. We should also check the help page for the `fastx_trimmer` command

```
fastx_trimmer -h
```

Now we can trim the data to just the first 32 bases. Also note from the help page above, that if no input file is specified the command looks to the standard input (*stdin*) for the reads. As this tool cannot process gzipped (i.e. compressed) files, we can thus decompress it using `zcat` as beforehand, and feed this to the command via the pipe (—).

```
cd ~/NGSData/RNASeq/rawData
mkdir -p ../trimmedData
zcat reads1.fq.gz | fastx_trimmer -Q 33 -f 1 -l 32 -z -o \
    ../trimmedData/reads1_len_trimmed.fq.gz
zcat reads2.fq.gz | fastx_trimmer -Q 33 -f 1 -l 32 -z -o \
    ../trimmedData/reads2_len_trimmed.fq.gz
```

We used the following options in the command above:

`-Q 33`	Indicates the quality scores are Phred+33 encoded
`-f 1`	First base to be retained in the output
`-l 32`	Last base to be retained in the output
`-z`	Compress (i.e gzip) the output
`-o`	Output file name

Now we can run `fastqc` on the trimmed files and visualise the quality scores.

```
cd ~/NGSData/RNASeq/trimmedData/
mkdir -p ~/QC/RNASeq/trimmedData
fastqc -o ~/QC/RNASeq/trimmedData -t 2 reads1_len_trimmed.fq.gz \
    reads2_len_trimmed.fq.gz
```

Open the new QC reports in `firefox` & compare with the original reports.

Quality Based Trimming

Instead of just removing all of the sequence data from the end of our reads, the base call quality scores can also be used to dynamically determine the trim points for each read. A quality score threshold and minimum read length following trimming can be used to remove low quality data.

First we'll have a look at the help page, then we run the following command to quality trim your data. This may take a minute or two so don't panic if it looks like nothing is happening.

```
fastq_quality_trimmer -h
```

```
cd ~/NGSData/RNASeq/rawData
zcat reads1.fq.gz | fastq_quality_trimmer -Q 33 -t 20 -l 32 -z -o \
    ../trimmedData/reads1_qual_trimmed.fq.gz
```

`-Q 33` Indicates the input quality scores are Phred+33 encoded

`-t 20` quality score cut-off of 20

`-l 32` minimum length of reads to output is 32

`-z` Compress (gzip) the output

`-o` Output file name

Run the same process on the 2nd pair in the set of reads, then inspect the QC report from `fastqc` using `firefox`.

```
cd ~/NGSData/RNASeq/trimmedData/
fastqc -o ~/QC/RNASeq/trimmedData -t 2 reads1_qual_trimmed.fq.gz \
    reads2_qual_trimmed.fq.gz
```

Was there a difference in the range of quality scores based on the two types of trimming?

Did you observe any unexpected behaviour?

Did the number of reads change depending on the trimming method?

Paired end samples need to keep the the paired structure of the reads intact between files. Would this structure be potentially disrupted by either of these methods?

Read Filtering

Another alternative to trimming reads would be to filter out reads which are low quality. In this RNA-Seq dataset, there were no reads which fail the Illumina Chastity Filter, but sometimes we can still be left with some low quality reads in our data. The tool `fastq_quality_filter` is available for this approach, which we will utilise for the RNA-Seq dataset.

```
1 fastq_quality_filter -h
```

By now we are becoming familiar with the commands, so once again let's create new gzipped fastq files with the filtered data. Once these processes are complete, inspect the reports obtained by `fastqc`.

```
cd
mkdir -p NGSData/RNASeq/filteredData
cd ~/NGSData/RNASeq/rawData
zcat reads1.fq.gz | fastq_quality_filter -Q 33 -q 20 -p 90 -z -o \
    ../filteredData/reads1_qual_filtered.fq.gz
zcat reads2.fq.gz | fastq_quality_filter -Q 33 -q 20 -p 90 -z -o \
    ../filteredData/reads2_qual_filtered.fq.gz
cd ../filteredData
mkdir ~/QC/RNASeq/filteredData
fastqc -o ~/QC/RNASeq/filteredData -t 2 reads1_qual_filtered.fq.gz \
    reads2_qual_filtered.fq.gz
```

`-Q 33` Indicates the input quality scores are Phred+33 encoded

`-q 20` quality score threshold of 20

`-p 90` 90 percent of bases in a read must be greater than the threshold

`-z` gzip the output

`-o` Output file name

Did we lose a significant number of reads by filtering the data?

Did this do a better job of improving the quality of the reads that any of the above methods?

Did you notice in the above code that we specified the use of PHRED+33 by setting the option `-Q 33`? Did you spot this in the help page for `fastx_trimmer` or

`fastq_quality_filter`? Neither did we. Welcome to the joys of bioinformatics! A large amount of our time is spent wondering why something didn't work that should've worked. Reading & understanding error messages can be very helpful, and sites like `www.biostars.org` and `seqanswers.com` can be life savers. Many bioinformaticians feel out of their depth a good amount of the time. Poor help pages & poorly written software is everywhere. If you're having trouble, you probably won't be the first to have that specific problem. Knowing where to look for the answers is one of the best skills you can have.

De-multiplexing Data

One further data pre-processing stage that we may need to undertake is *de-multiplexing* data. As mentioned earlier, we can often run multiple samples per lane using unique '*barcode*' sequences to identify which sample a read can be assigned to. This approach is very advantageous for researchers, especially in terms of cost and minimising technical variability, but it adds an additional layer of pre-processing that is not as trivial as one would think, given the average 0.1-1% sequencing error rate that introduces a lot of multiplicity in the actual barcodes. On many occasions the data is de-multiplexed immediately after sequencing, and only FASTQ files that are ready for analyses are distributed. However, you might encounter the necessity to perform the de-multiplexing step yourself, or, be given de-multiplexed FASTQ files, to remove the adaptors manually; thus, it is important to learn how to deal with such data.

Change to the folder `~/rawData/multiplexed` & inspect the directory contents.

```
cd ~/NGSData/RNASeq/multiplexedData
ls -lh
```

Here you can see a fastq file which contains multiplexed data `barcoded.fq`, and the file `barcodes.txt` which contains information about which barcode belongs to which sample. Let's inspect the barcodes using the `less` pager, and once again quit by hitting the <q> key.

```
less barcodes.txt
```

In order to separate the reads in 4 different fastq files (one for each barcode/sample) we will use `fastq-multx`. As we have already learned, before we use a tool, we first need to check the help page.

```
fastq-multx -h | less
```

Note that the abbreviations `-bol` & `-eol` are used to indicate *beginning of line* and *end of line* respectively.

We will supply the reads as the file `barcoded.fq`, and specify the output files as %.barcoded.fq. For reads which match the barcode sequence `TTGCGA`, going by the help file, what do you think the filename will be that they are output to?
In the help file, you will see the `-m N` option, which allows specifying mismatches to the barcode to allow for sequencing errors within the barcode. Can you think of any potential pitfalls with this approach?
Why might we sometimes use the `-x` option?

Let's try de-multiplexing the data, using automatic guessing for the barcode location with the following command.

```
cd ~/NGSData/RNASeq/multiplexedData
mkdir ../demultiplexedData
fastq-multx barcodes.txt barcoded.fq -o ../demultiplexedData/%.fq
```

After executing the command above you should have five new fastq files: one corresponding to each sample and one for the reads that did not match any of the barcodes.

Try generating a QC report for the initial file, and for any one of the samples. How does the sample compare to the report for the initial multiplexed fastq file?
What happened to the read length?

Aligning Reads To a Reference Sequence

Primary Author(s):
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 our RNA-Seq data, we could align to the genome or the transcriptome. What might be some of the disadvantages of either approach?

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 like the ones we have in our RNASeq dataset, whilst `bowtie2` 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 RNASeq 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. The RNASeq reads we have today come from *Drosophila melanogaster*, so to find the genome, open Firefox & head to ftp://ftp.ensembl.org/pub/release-62/fasta/drosophila_melanogaster. Here you can find the transcriptome (*cdna*), genome (*dna*), proteome (*pep*) & non-coding RNA (*ncrna*) in separate folders. Let's align to the genome, so first create the directory to save it to then download using the linux command `wget`. Instead of typing that enormous last command, navigate to the /dna folder on the ftp server then right-click on the file Drosophila_melanogaster.BDGP5.25.62.dna_rm.toplevel.fa.gz & select `Copy Link Location`. After typing `wget`, right-click in the terminal & paste the contents of the clipboard.

```
cd ~
mkdir reference
cd reference
wget \
   ftp://ftp.ensembl.org/pub/release-62/fasta/drosophila_melanogaster/dna/Drosophila_mel
gunzip Drosophila_melanogaster.BDGP5.25.62.dna_rm.toplevel.fa.gz
```

Now we have unzipped the Drosophila genome, 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.

```
head Drosophila_melanogaster.BDGP5.25.62.dna_rm.toplevel.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 '^>' Drosophila_melanogaster.BDGP5.25.62.dna_rm.toplevel.fa
```

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

```
grep -c '^>' Drosophila_melanogaster.BDGP5.25.62.dna_rm.toplevel.fa
```

What does the "rm" stand for in the filename?

Building an Index

We will align using the tool `tophat` which is based on the aligner `bowtie` so requires an index to be built using `bowtie-build`. This tool is specially designed to handle splice junctions & is highly suitable for aligning RNASeq reads to a reference genome, such as we have

Once again, we need to check the help pages. Unfortunately the `tophat` page isn't very friendly on the screen so after entering the following you might have to scroll up a little. There are a wealth of options available for you to look through.

```
1 tophat -h
```

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

```
1 bowtie-build -h
```

Note that in the output from this process we are not going to specify a file, but rather a `basename` which will be included in all files.

Now that we've had a look, type to following command which will take a few minutes to run, so don't worry if it looks like nothing is happening.

```
1 bowtie-build Drosophila_melanogaster.BDGP5.25.62.dna_rm.toplevel.fa \
    D_melanogaster.BDGP5.25.62.dna_rm.toplevel
```

Note that we could have specified anything as the basename, but keeping the key information which relates to the original file is often good practice, despite it making for long & cumbersome commands. This is where using a text editor, with cut & paste can make life a little easier.

Let's look at what files have been created.

```
1 ls
```

Unfortunately the BWT transformed files are in binary, so we can't really see what they

look like, but this is what is required by the aligner *tophat*, as well as *bowtie*

Aligning the reads

Unfortunately, this will take hours on our VMs so we can't really perform it in the time we have for the workshop. Instead, a set of aligned files should be on your VM in the folder `~/mapped`, which we will explore in the next section. **Please do not run this**, but if we wanted to align the trimmed reads, the command would look like the following.

```
tophat D_melanogaster.BDGP5.25.62.dna_rm.toplevel \
 ~/NGSData/RNASeq/trimmedData/reads1_trimmed.fq.gz \
  ~/NGSData/RNASeq/trimmedData/reads2_trimmed.fq.gz
```

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?

If you are running well ahead of time, try downloading the repeat masked version of the sequence for the chromosome 2L from the `ftp` site. Align the reads to to this sequence only.

Working With Alignments

Primary Author(s):
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.

The reads from the previous dataset which mapped to *chr2L* of *D. melanogaster* are in the folder `~/NGSData/RNASeq/alignedData`. Let's look at the first 5 lines of the file.

```
cd ~/NGSData/RNASeq/alignedData
head -n5 chr2L.sam
```

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

Conversion to BAM format

The BAM format is much more convenient computationally, so we can convert this file into the BAM format using `samtools view`. Whilst this line of code may look counter-intuitive, remember that the first section is all the information about how to process the input, & specify the output, whilst the final section is the file that the command will operate on. First look through the help page to understand what you are asking the tool to perform.

```
samtools view
samtools view -bS -o chr2L.bam chr2L.sam
```

Have a look at the file sizes to see the difference. Note that the BAM file is about 12% of the size, which can be an important consideration when managing large datasets.

```
ls -lh
```

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 chr2L.bam
```

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

```
samtools view chr2L.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 chr2L.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 chr2L.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 chr2L.paired.bam chr2L.bam
ls -lh
```

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

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 `37M`, 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 '10M240N27M'. Can you think of an interpretation of this string?

Knowing this pattern as splice junctions, you could write a *regular expression* search to find only the alignments that contain potential splice junctions.

What would be a pattern to search for, and how would you create this as a new BAM file.

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 ~/NGSData/RNASeq/alignedData
samtools sort chr2L.bam chr2L.sorted
samtools index chr2L.sorted.bam
```

Note that in keeping with the weirdness of bioinformatics *the input file is not specified last in the tool* `samtools sort`. This would normally be the convention but for some reason the programmers have chosen not to follow this convention. The final setting given to the command is the prefix that the output file will be given.

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.

```
igv
```

Once the browser has opened, go to the drop-down menu & load the genome for *Drosophila melanogaster* (dm3). (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 chr2L, then enter the word *poe* in the window next to this. This will take us to the gene *poe* 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 *drongo* 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 running an entire pipeline on the iCLIP data, including adapter removal and read trimming. Record your commands in the `gedit` text editor so you have a record of what you did. Instead of aligning to the entire human genome, try just aligning to chromosome 17.

Space for Personal Notes or Feedback