ITGE2017

INTRODUCTION TO TUBERCULOSIS GENOMIC EPIDEMIOLOGY
RIO DE JANEIRO 2017

COURSE DOCUMENTATION

September 21-22nd, 2017

LISBOA
UNIVERSIDADE DE LISBOA

FACULDADE DE FARMÁCIA
Universidade de Lisboa

iMed. ULisboa Research Institute for Medicines

DESDE 1902
SANITATEM QUAERENS IN TROPICOS
INSTITUTO DE HIGIENE E MEDICINA TROPICAL
UNIVERSIDADE NOVA DE LISBOA

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

FACULDADE DE MEDICINA
U F R J

INTRODUCTION TO TUBERCULOSIS GENOMIC EPIDEMIOLOGY 2017

:: Course Overview ::

Teaching Staff:

- João Perdigão, PhD, Research Institute for Medicines (iMed.ULisboa), Faculty of Pharmacy of the University of Lisbon (PORTUGAL) :: E- mail: jperdigao@ff.ulisboa.pt
- Elis R. Dalla-Costa, PhD, Centro de Desenvolvimento Científico e Tecnológico (CDCT), Secretaria Estadual de Saúde (ES-RS) (BRAZIL) :: E-mail: dallacostaer@gmail.com

Course Contents

The course is structured in three distinct modules (Total Hours: 7) that covers basic Next Generation Sequencing analysis workflows:

1. **Module 1** – Mapping Sequence Data
   1.1. FASTQ format and Quality Control
   1.2. Mapping/Reference Assembly
   1.3. Data Visualization
   1.4. Variant Calling
   1.5. Variant Functional Annotation
2. **Module 2** – *De novo* Assembly
   2.1. *De novo* Assembly: concepts and parameter optimization
   2.2. Contig Ordering and Scaffolding
   2.3. Genomic Annotation
3. **Module 3** – Introduction to Phylogenetics
   3.1. Core-genome SNP Alignments: from reads to multiple genome-wide SNP alignment
   3.2. Distance-matrix methods: the Hamming distance
   3.3. Evolutionary Distance and Models of Molecular Evolution
   3.4. Phylogenetic Reconstruction

Course documentation includes an introduction the topics covered in the course as well as exercises/tutorials. Moreover, further documentation concerning the installation of the computing system, overview and basic Linux commands is also provided.

Course Schedule

Hour	21st September 2017	22nd September 2017
08:30	PVE EVALUATION	PVE EVALUATION
09:00		
09:30		
10:00		
10:30		
11:00		
11:30		
12:00		
12:30		
13:00		
13:30		
14:00	ITGE 2017 MODULE 1	ITGE 2017 MODULE 2
14:30		
15:00		
15:30		
16:00		ITGE 2017 MODULE 3
16:30		
17:00		
17:30		
18:00		

TABLE OF CONTENTS

:: Course Computing System Overview ::

Introduction

This course will make use of a Virtual Operating System based on CentOS 7. This is a free, enterprise-class, community-supported Linux distribution that is sourced from Red Hat Enterprise Linux (RHEL).

For this course we will use a Virtual CentOS image that will run on Oracle VirtualBox, a free virtualization platform that runs across multiple systems. Every computer made available for this course is already pre-installed with Virtual Box already configured with CentOS 7. You just need to find the shortcut on your desktop, double-click it, then select CentOS Virtual Machine (VM) from the VM list on the left and click Start. This will open a new window and CentOS with all course files and bioinformatic software necessary for the course will soon start.

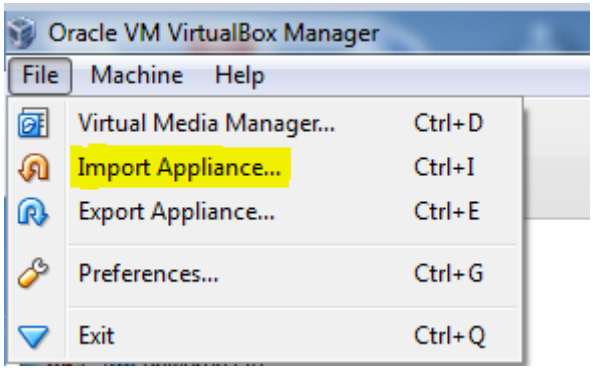
The image of this VM is also available (approx. size 40Gb) if you wish to take it and carry out some of the course analysis at home. Just ask the course instructors for the link to download this image. The following section explains how to import this image to VirtualBox on your computer.

Importing the OS Virtual disk into VirtualBox

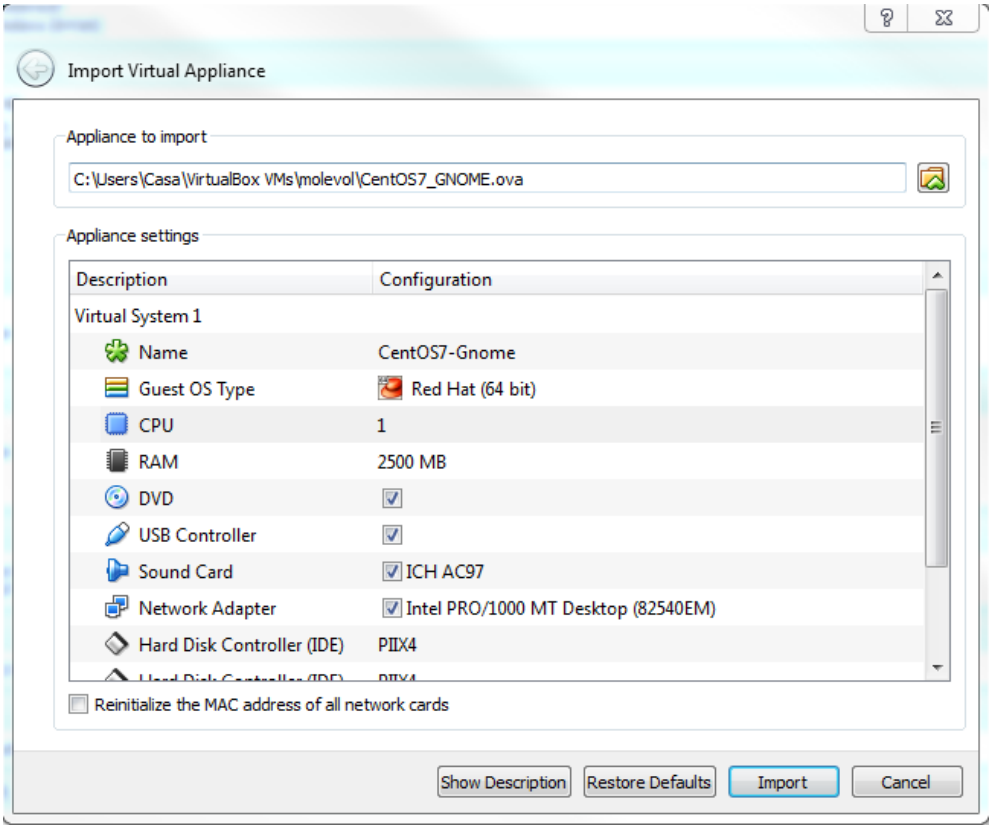
There are two main methods for importing the course VM to your VirtualBox installation: using an image disk file (vdi or vmdk file) or using an exported Open Virtualization Format Archive (OVF or OVA file). You will therefore which type of file was made available to you and proceed accordingly.

Import from a OVA/OVF Archive:

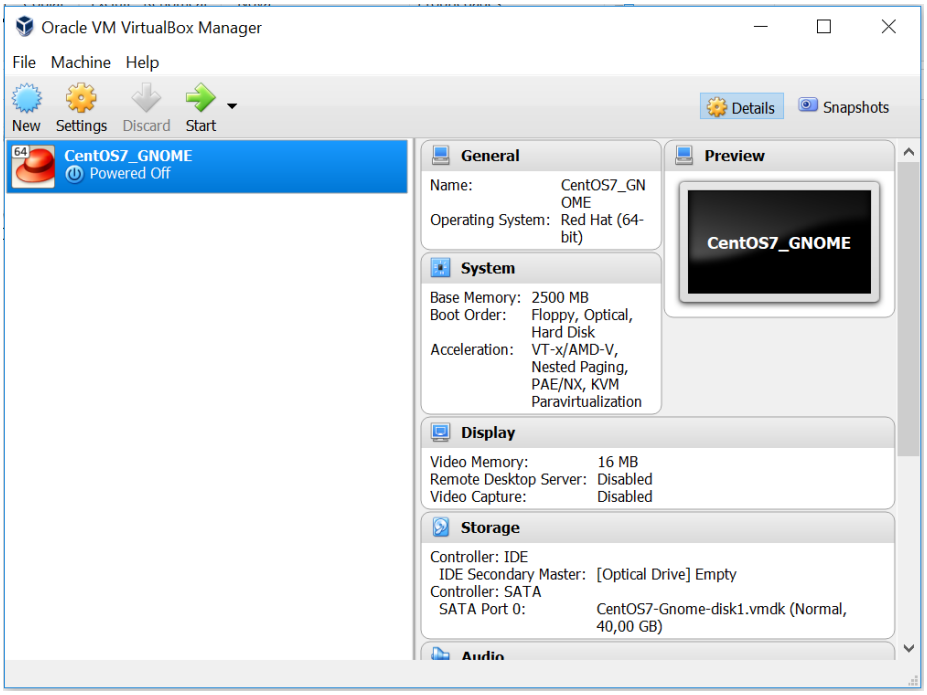
To import a VM in OVA/OVF format go to *File* > *Import Appliance*…



Then, select the OVA/OVF file from its location on your computer and click on *Import*.

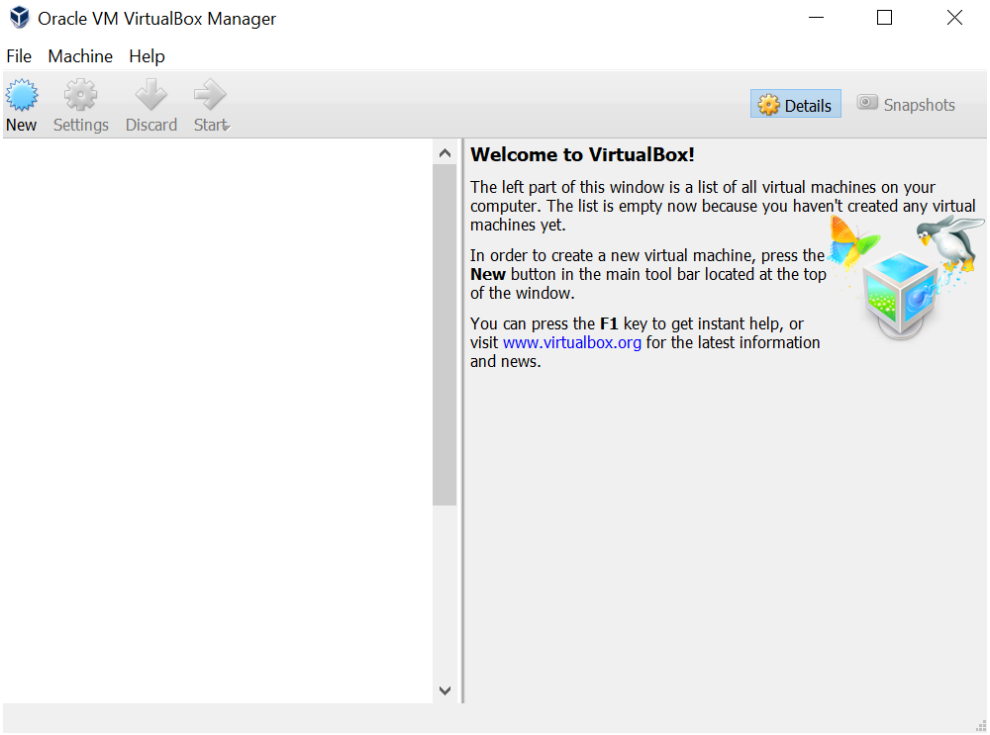


It may take some time to finalize the importation process.

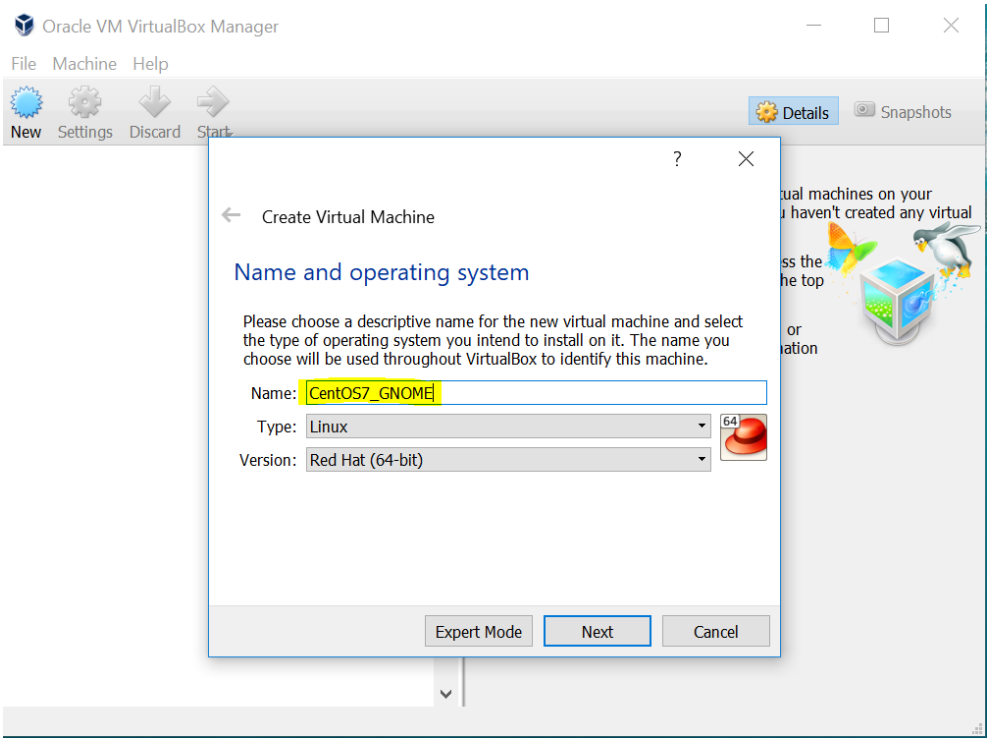


After the Importation process this VM will appear on VirtualBox VM list. To start it just select it and click *Start*.

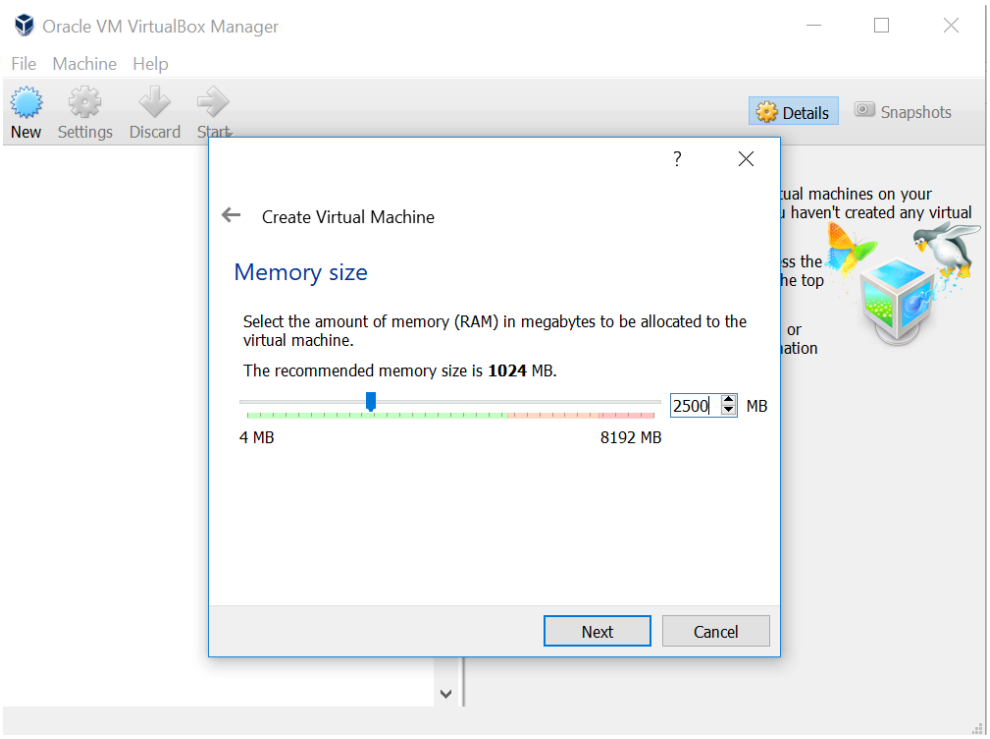
Import from a vdi or vmdk disk file:



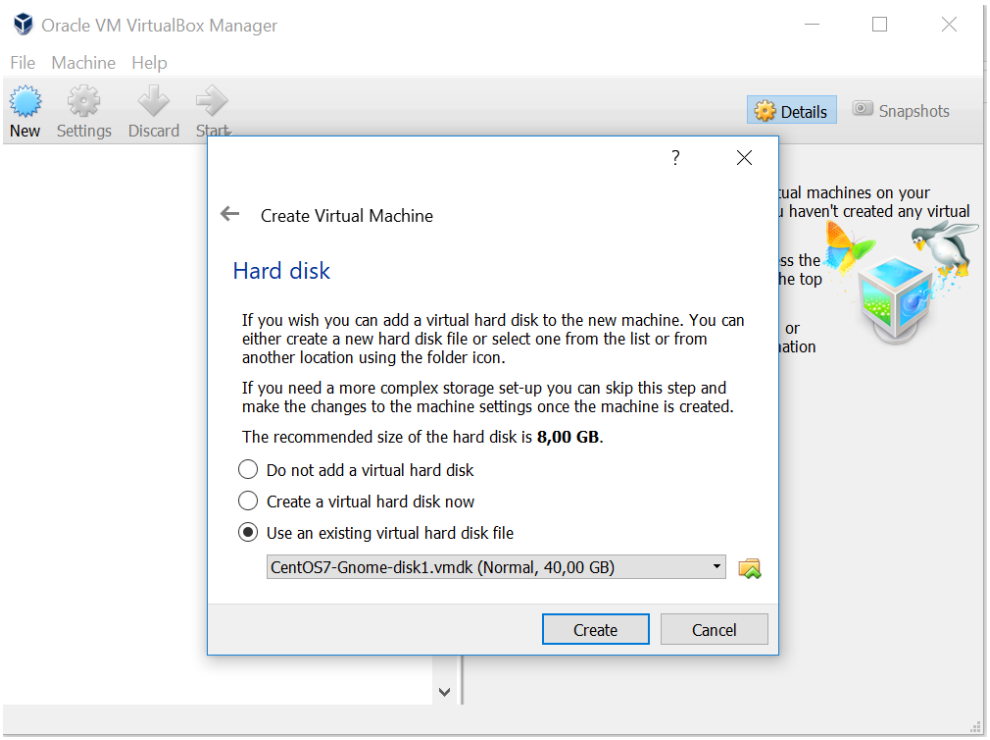
Start VirtualBox and click *New*.



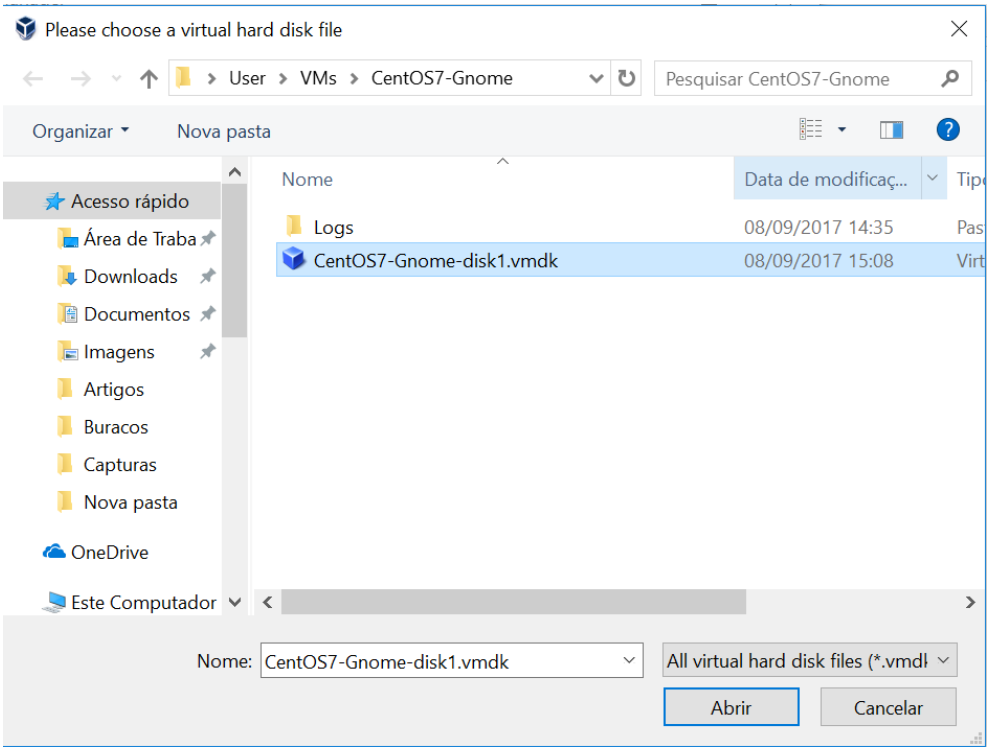
You can give it the name of your choice (e.g. CentOS7_GNOME), select type as Linux and for Version you can go with Red Hat (64-bit).

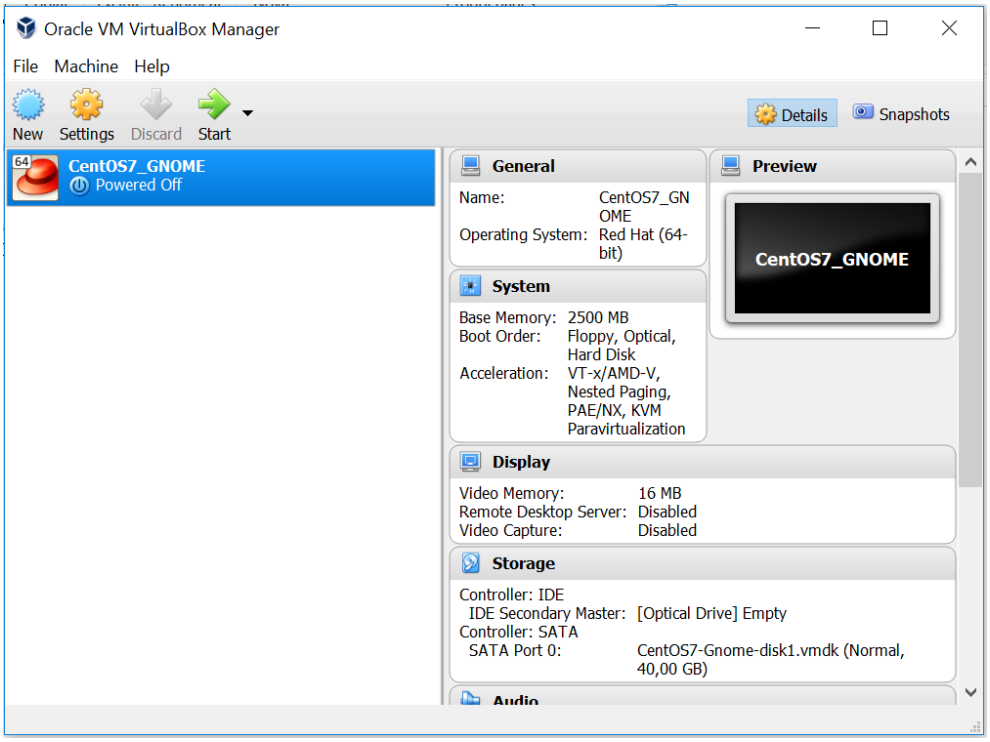


For memory size allocation it will depend on your system. For this course we recommend a minimum of 2500 Mb.



Start VirtualBox and click *New*. In this next window select *Use an existing virtual hard disk file* and select the disk image provided (.vmdk file) as shown below. Please note that before selection it is advisable to put this file in its final destination folder.





After clicking on *Create* this VM will appear on VirtualBox VM list. To start it just select it and click *Start*.

About the Course VM

As mentioned above the course VM runs on a CentOS7 Linux distribution. It comes with a GNOME Graphical User Interface (GUI), which is a free and open-source desktop environment that runs on Linux distributions.

This CentOS installation is configured to log in automatically with user centos. The user passwords are the following:

User: centos

Password: reverse

User: root

Password: reverse

root is the super-user and it won't be necessary to use this account during the course. It may eventually become necessary if you deploy this VM onto your own computer and wish to undertake some system changes.

After logging in you will see the following desktop or a similar one:

Notice the desktop is already populated with some of the software that you will use over the course. Also, there are direct shortcuts to:

- your *home* directory folder (centos user *home* directory in this case) (Path: /home/centos/).
- open up a terminal/console window. Every time you open a new window from this shortcut it will open on your home directory.

You will find the course files within your home directory in the following folders/directories:

- **Module1** – Module 1 working directory with some files already present;
- **Module2** - Module 2 working directory with some files already present;
- **Module3** - Module 3 working directory with some files already present;
- **course_files** – Contains the FASTQ files to be used over the course;
- **course_documentation** – Contains this document and other course documentation.

Under the menu *Applications* in the desktop top bar you will find links to additional software and tools already installed (*e.g.*, Libre Office [an open-source alternative to Microsoft Office] or System Monitor).

:: Linux Command-line Basics ::

What is Linux?

During the 70's programmers from the Bell laboratories (AT&T) developed a new Operating System (UNIX) to overcome the existing limitations of the Operating Systems and Programming Languages of that time. Although initially used only for internal use at AT&T, universities and research centers became increasingly interested in this new OS with AT&T making available its source code. This enabled this new OS to expand and new extensions and tools to de developed by the scientific community.

However, AT&T started to restrict the licensing of UNIX and in order to protect the free software a new foundation called Free Software Foundation was created along with a new licence: General Public Licence (GPL). And from this new foundation a new UNIX-based OS was born: GNU (which means GNU is Not UNIX). In the 90's, Linus Torvalds inspired by the GNU project developed the a new OS kernel: Linux, fully compatible with UNIX and GNU environments. Although quite rudimentary and incomplete, at its beginning Linux became increasingly popular, reaching one hundred new users in its first week. Over the last decades since its creation it has become a very robust and sophisticated OS and is currently at the same level of other commercial OS, if not beyond.

Why on earth should I use Linux and the command-line?

First, it's free! And, being an open-source OS, each user can be a contributor to Linux development which increases its reliability and stability since any error or bug can be readily corrected and incorporated in future releases.

Two of the most important Linux features are that this is a multitask and multiuser OS. Multiple users can be logged in simultaneously and can start multiple tasks/programs – called processes. Furthermore, each user is prevented from interfering with another user's work.

Presently, many Linux distributions already come bundled with graphical desktop environments (e.g. GNOME or KDE) but the command-line provides the user with a more flexible and enhanced control of the system. In the beginning it may be hard to know the commands but as the user gets familiar and learns how to use the command-line he can achieve a performance that is superior to the graphical interfaces.

Do I have to memorize all the commands?

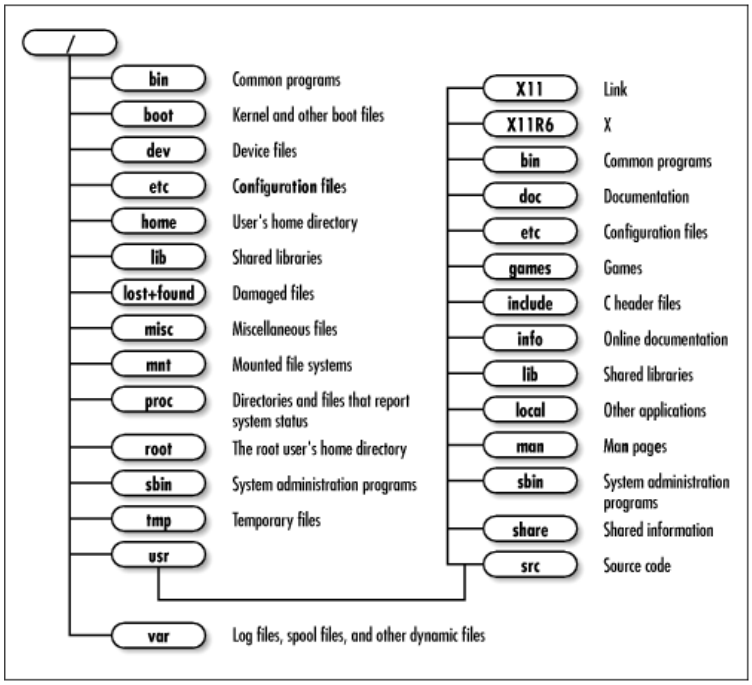
Actually no! Some of the commands are essential while others you will probably never use them. Linux comes with a plethora of basic commands and programs that you won't even know they are there. You'll end up by naturally assimilate this "new language" as you use these commands and if you don't recall any specific command just look it up on the textbooks, online on webpages or the extensive community support forums, or in this document – that's what those are for!

Filesystem

As in windows or MacOS, files are organized in folders or directories where each directory can have multiple sub-directories and so on. The main directory (/) is called **root** and contains all directories and folders in the system. Each user only has access to its home directory unless if granted access to other directories or has administrative privileges. Each user home directory is located on /home/[user login] (in this course VM: /home/centos).

There is a special user: the **super-user** called **root**, because it has access to the root directory.

Take a look at an example of a Filesystem hierarchy:



The command-line

You can access to the command-line by opening a terminal window using the shortcut on your desktop. It is called command-line because the user has to write commands in the form of text. Once you open the command-line interface you will see an initial message looking like this:

```
[centos@localhost ~]$
```

This is called the prompt and it tells you that the system is waiting for commands. It also tells you that you are logged in as user *centos* at (@) the *localhost* (the machine name you are using) and you are at your home directory (~). This is important because you can use the command-line to switch to another user and to connect to another machine.

Basic commands

You can start by obtaining a list of the files and sub-directories that exist in your current directory by typing:

```
$ ls
```

```
[centos@localhost ~]$ ls
commands.txt          Downloads   Music             Public       test.sh
course_documentation  files_copy  NC000962_3.fasta  R            Videos
course_files          Module1     NC000962_3.gbk    RAST_output
Desktop               Module2     other             Templates
Documents             Module3     Pictures          test.Rexec
[centos@localhost ~]$
```

Or if you want to obtain a more detailed list (long) list you can type:

```
$ ls -l
```

```
[centos@localhost ~]$ ls -l
total 17544
-rw-rw-r--. 1 centos centos     5429 Sep  1 18:51 commands.txt
drwxrwxr-x. 2 centos centos        6 Sep 11 10:22 course_documentation
drwx------. 2 centos centos     4096 Aug 19 18:45 course_files
drwxr-xr-x. 2 centos centos     4096 Sep  7 11:26 Desktop
drwxr-xr-x. 2 centos centos        6 Nov  1  2014 Documents
drwxr-xr-x. 5 centos centos     4096 Sep 11 10:26 Downloads
drwxrwxr-x. 2 centos centos     4096 Aug 19 18:29 files_copy
drwxrwxr-x. 3 centos centos       61 Sep  8 14:14 Module1
drwxrwxr-x. 3 centos centos     4096 Sep 11 10:23 Module2
drwxrwxr-x. 6 centos centos       66 Sep 14 16:39 Module3
drwxr-xr-x. 2 centos centos        6 Nov  1  2014 Music
-rw-r--r--. 1 centos centos  4474567 Jul  6 13:25 NC000962_3.fasta
-rw-r--r--. 1 centos centos 13443692 Jul 23  2013 NC000962_3.gbk
drwxrwxr-x. 5 centos centos       35 Sep 11 10:31 other
drwxr-xr-x. 2 centos centos     4096 Sep  7 11:36 Pictures
drwxr-xr-x. 2 centos centos        6 Nov  1  2014 Public
drwxrwxr-x. 3 centos centos       44 Aug 16 23:38 R
drwxrwxr-x. 2 centos centos        6 Sep  8 17:54 RAST_output
drwxr-xr-x. 2 centos centos        6 Nov  1  2014 Templates
-rwxrwxr-x. 1 centos centos       80 Aug 17 02:20 test.Rexec
-rwxrwxrwx. 1 centos centos       80 Aug 17 02:21 test.sh
drwxr-xr-x. 2 centos centos        6 Nov  1  2014 Videos
[centos@localhost ~]$
```

Important note: the commands are case-sensitive so Ls or LS will not work. Keep this always in mind!!

This more exhaustive list gives a number of details that we will not cover here but notice the modification date and time and size in bytes. Notice that NC000962_3.gbk file has a size of 13443692 bytes. It seems a lot but it is only 13.4 Mb!

Now, how do we find the full path of the directory where we are working? Just type:

```
$ pwd
```

It means print working directory outputs this:

```
[centos@localhost ~]$ pwd
/home/centos
```

What if I want to change directory:

```
$ cd Module1
```

cd means change directory and you can use the full path of a directory to change to a distant directory from where you are (e.g. cd /home/centos/Module1/vcfs). When you don't use the full path, cd assumes that the directory you are specifying exists in the present working directory. Now type pwd. What do you see?

```
$ pwd
```

```
[centos@localhost Module1]$ pwd
/home/centos/Module1
```

To go back type:

```
$ cd ..
```

On the command-line, two dots (..) means the directory above while one dot (.) means the current directory. That is why if you type `cd .` it will not get you anywhere but the present directory.

File manipulation

Let's say we wish to manipulate files and directories. To create a new directory type:

```
$ mkdir test_dir
```

To remove it:

```
$ rmdir test_dir
```

If the directory contains files rmdir will not work,you have to type:

```
$ rm -r test_dir
```

Which is more drastic. Try this and use the cd or ls command to go inside and see your directory.

And what about files?

You can copy (cp command) or move (mv command) files easily from the command-line:

```
$ cp NC000962_3.fasta Documents
```

This command copied NC000962_3.fasta file to your Documents directory. To avoid errors youl would write it in the following format:

```
$ cp ./NC000962_3.fasta ./Documents/
```

That way it is more intelligible that you are copying it to a folder. But here you are not specifying if you want the copied file to remain with the same name in the destination folder, the prompt assumes that it should stick with the same name. Otherwise:

```
$ cp ./NC000962_3.fasta ./Documents/test.fasta
```

The mv command works approximately in the same way, except that it deletes the original file.

To remove a file just type:

```
$ rm ./Documents/test.fasta
```

Can we visualize text files on the command-line? Sure, let's use the cat command (from your home directory, to go there from anywhere type cd ~) and type:

```
$ cat NC000962_3.fasta
```

Did you see it all? Probably not. By the way, cat takes multiple files and can concatenate those, hence the name cat. You can use CTRL+S and CTRL+Q to stop the scroll or, scroll up using SHIFT+PgUp or SHIFT+PgDn when the listing stops.

To see a file gradually with stoppings, type:

```
$ more NC000962_3.fasta
```

If you press ENTER or SPACE it will scroll down, to exit just press q.

Now let's stop for a moment and introduce three other characters:

- >, redirects the output to a file and erases a previously existing file if it has the same name;
- >>, redirects the output but appends it to the end of an existing file;
- |, named pipe character, does exactly that, pipes or channels the output of a command to another.

We can, for example, do:

```
$ ls -l > list.txt
```

This created a new file (list.txt). Let's read the content:

```
$ more list.txt
```

Is it familiar?

And if you do this:

```
$ ls -l >> list.txt
$ more list.txt
```

What now?

Wild-cards

To end this basic Linux tutorial. We'll introduce wild-cards and regular expressions. Wild-cards are special characters that enable you to call many files recursively. These are:

Character	Meaning
*	Any sequence of one or more characters
?	Any single character
[]	A sequence of one or more characters containing the characters within brackets

Let's use wild-cards to list files that only start with NC:

```
$ ls -l NC*
```

Or if we only want fast files:

```
$ ls -l *.fasta
```

You can use wild-cards on file operations as well:

```
$ cp ./NC* ./Documents/
```

What happened? Do you notice you have copied both files to the documents directories?

Let's delete those:

```
$ rm ./Documents/NC*
```

You just deleted two files with a single command. Be aware of the risks of using wild-cards as you can easily delete thousands of files with a single command. But, notice the power that comes with the command-line.

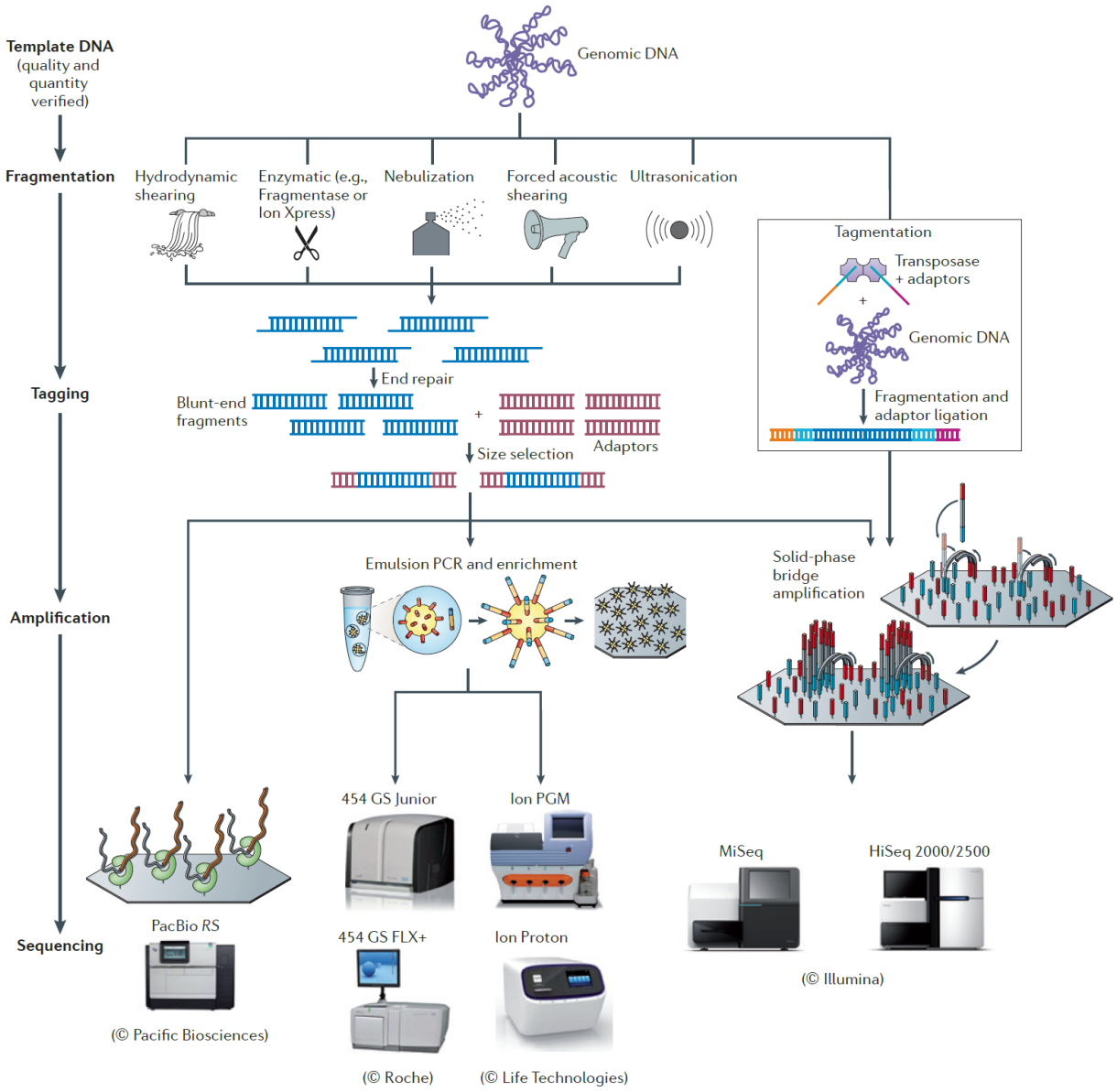
Hope this basic introduction is helpful for the following course Modules.

:: Mapping Sequence Data ::

Introduction

A wide array of Next Generation Sequencing platforms are available nowadays and, depending on the platform, a single machine can output a total of 6000 Gbp in approximately 40h (the equivalent to approximately 937 human diploid genomes). Moreover, this data is produced in the form of short or long **sequencing reads**, also depending on the platform [1]. Given the ability of these machines to produce millions of reads from a single organism, the process of arranging these in an attempt to find overlaps and delineate contigs (genome assembly) is computationally demanding. Long read lengths are more adequate for genome assembly whereas the assembly of a genome using short read lengths may prove challenging particularly due to repetitive regions. This challenge can be partially overcomed using a reference genome for the organism being studied, in which the sequencing reads are aligned with this reference genome. This procedure is often called **reference assembly** or **mapping**. This designation contrasts with the procedure in which sequencing reads are assembly without a reference genome – ***de novo* assembly.** An important aspect for mapping that should be taken in account is that the reference genome should be a high quality well assembled genome [1].

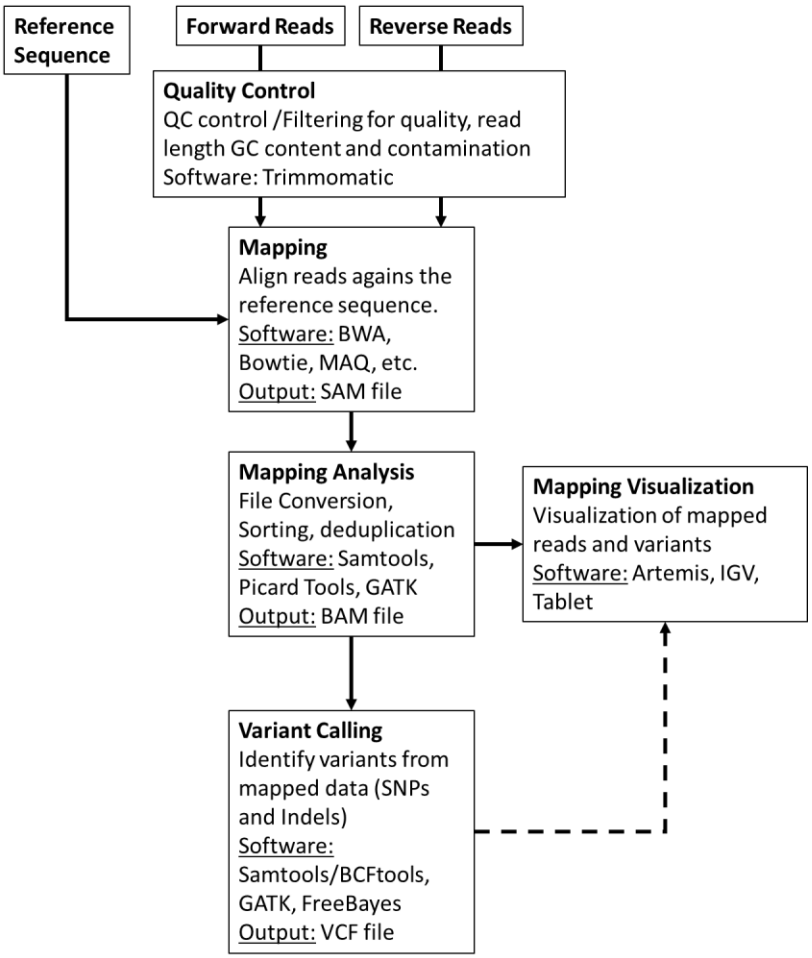
This module will address the basic bioinformatic analytical steps underlying mapping/reference assembly of NGS data. We will mainly focus on the analysis of sequence data obtained from Illumina platforms, that presently dominates 80-90% of the market, sequenced in paired-end mode. **What does "paired-end mode" mean?** It is important to first start by understanding the process of Illumina "sequencing-by-synthesis" methodology: starting with genomic DNA, it is randomly sheared and an appropriate length is selected after fractioning by agarose gel electrophoresis, followed by adapter ligation and solid phase PCR amplification (bridge amplification) on a sequencing flow cell [1]. Afertwards, sequencing by synthesis is carried out using reversible fluorescently-labelled terminator nucleotides where each fragment is sequenced on both ends producing two mate reads for each fragment (hence, **paired-end sequencing**). The size of the region that is not sequenced corresponds to the region between reads in this fragment and is often called **insert size** and therefore depends on the read length (which depends on the machine and number of cycles, currently up to 2x300bp on MiSeq).



(Loman *et al*, 2012)

The ensuing computational or in silico analysis picks up from the FASTQ files produced after adapter removal. The two main approaches at this point are as mentioned, **Mapping** and ***De novo* Assembly**. In this module we will focus on **Mapping** sequence reads obtained from *M. tuberculosis* clinical isolates collected in Portugal. By mapping sequence reads to a reference genome it will enable the downstream identification of Single Nucleotide Polymorphisms (SNPs), insertions and deletions (indels) and copy number variants (CNVs) that exist between the two organisms. Comparative Genomics underpinned on mapping analysis is also possible if the reference sequence remains the same.

The general workflow for this analysis consists in mapping the reads against a reference sequence using a mapper/aligner software to produce a mapping file (SAM/BAM) that can be further analysed and sorted using programs such as Picard Tools/Genome Analysis Toolkit or Samtools. Afterwards, it is possible to visualize the alignments using appropriate software with graphical interface and perform additional downstream analysis such as variant calling.



First, let's have a quick look at the FASTQ file format (Exercise 1).

Exercise 1 – Understanding the FASTQ file format

Let's have a look at the files made available for this course. Under the home directory there is a another directory called fastq_files.

Let´s access this folder: open up a terminal and type:

```
$ cd course_files
```

You can list its contents by typing ls (list command) which should give you the list of files and sub-directories within. You'll find a fasta file (NC000962_3.fasta) for the M. tuberculosis H37Rv reference genome (GenBank Acc. NC000962.3) and ten compressed fastq files for five different M. tuberculosis clinical isolates:

- PT000033: PT000033_1.fastq.gz and PT000033_2.fastq.gz
- PT000049: PT000049_1.fastq.gz and PT000049_2.fastq.gz
- PT000050: PT000050_1.fastq.gz and PT000050_2.fastq.gz
- PT000271: PT000271_1.fastq.gz and PT000271_2.fastq.gz
- PT000279: PT000279_1.fastq.gz and PT000279_2.fastq.gz

Some of the programs we will use ahead can deal directly with fastq compressed files but let us decompress the PT000033 files while keeping the compressed files unchanged:

```
$ gzip -dc PT000033_1.fastq.gz > PT000033_1.fastq
$ gzip -dc PT000033_2.fastq.gz > PT000033_2.fastq
```

The -d option indicates that it is a decompressing operation and -c option redirects the output to the file we indicate after >.

Let's use this strain for the following exercises. By the way, you can learn more about these strains on htttp://cplp-tb.ff.ulisboa.pt.

```
$ more NC000962_3.fasta
```

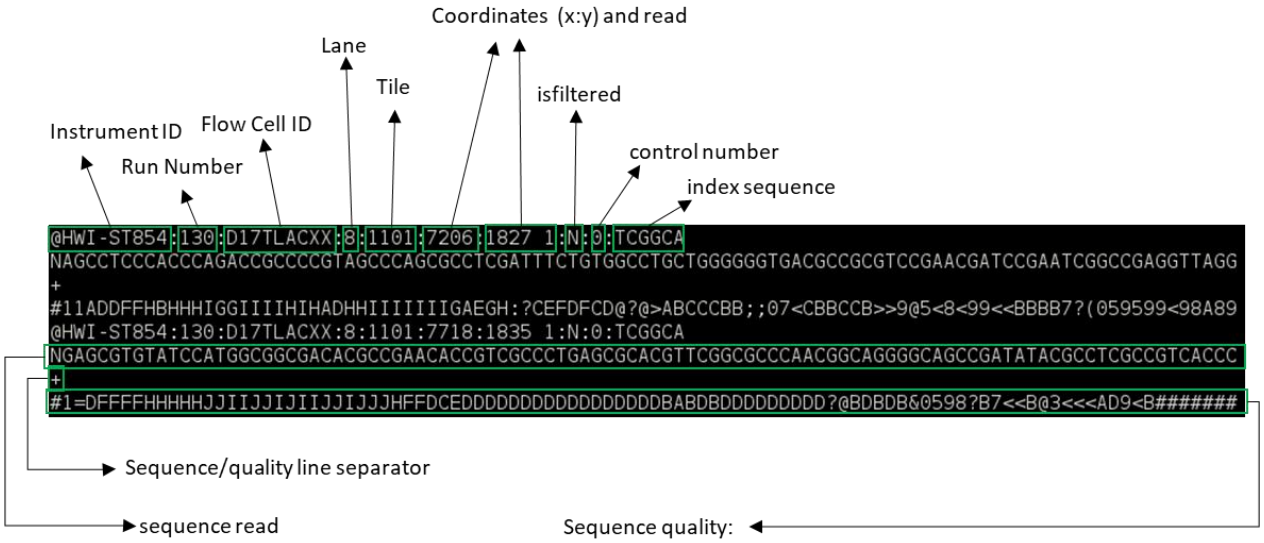
Now let's look at the file content and format. Let's start by looking at the reference fasta file.

Now let us compare the FASTA format with the FASTQ format:

```
$ more PT000033_1.fastq
```

What are the differences?

The FASTQ format:



ASCII encoded Phred-33 quality score for each nucleotide of the sequence read above. Example: the 3rd nucleotide in the bottom read has a quality (Phred33 Q) of 28. Check the table below. What about the 5th nucleotide of the same read? _____

ASCII_BASE=33 Illumina, Ion Torrent, PacBio and Sanger

Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	33 !	11	0.07943	44 ,	22	0.00631	55 7	33	0.00050	66 B
1	0.79433	34 "	12	0.06310	45 -	23	0.00501	56 8	34	0.00040	67 C
2	0.63096	35 #	13	0.05012	46 .	24	0.00398	57 9	35	0.00032	68 D
3	0.50119	36 $	14	0.03981	47 /	25	0.00316	58 :	36	0.00025	69 E
4	0.39811	37 %	15	0.03162	48 0	26	0.00251	59 ;	37	0.00020	70 F
5	0.31623	38 &	16	0.02512	49 1	27	0.00200	60 <	38	0.00016	71 G
6	0.25119	39 '	17	0.01995	50 2	28	0.00158	61 =	39	0.00013	72 H
7	0.19953	40 (	18	0.01585	51 3	29	0.00126	62 >	40	0.00010	73 I
8	0.15849	41)	19	0.01259	52 4	30	0.00100	63 ?	41	0.00008	74 J
9	0.12589	42 *	20	0.01000	53 5	31	0.00079	64 @	42	0.00006	75 K
10	0.10000	43 +	21	0.00794	54 6	32	0.00063	65 A			

Let's take a look at the beginning and end of the bottom sequence in the figure above. What are the Phred33 Q scores compared to the rest of the sequence?

How we deal with this problem will be topic of **the next exercise**: **Read Quality Control**!!

Exercise 2 – Read Quality Control

Before introducing sequencing reads into an analytical pipeline it is highly important to check the overall quality of the reads through the evaluation of the percentage of low quality bases, contamination or length. The impact of low quality or base calling errors can be minimised in downstream applications by applying some filters where low quality reads or bases are removed if these do not meet thresholds defined by the user [2, 3].

Let's start by analysing the reads from the previous exercise (PT000033). These have been put in the Module1 directory. To assess the quality of these reads we will use a Java written software named FastQC that provides a user-friendly way to carry out some quality control checks on raw sequence data.

To start, double-click on the FastQC shortcut available on the desktop. This will open the FastQC graphical interface. Alternatively, you can start the FastQC from the command prompt by typing:

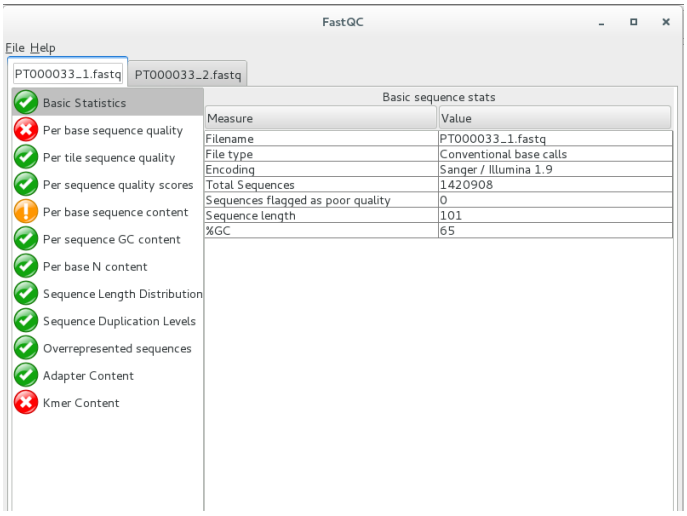
```
$ fastqc
```

Go to File>Open and then select the two reads for PT000033 strain present in the Module1 directory. This will start the analysis. After analysis is completed you can examine the data for each read file showing up on different tabs.

Go ahead and take a look at the different parameters that are evaluated. Normal QC tests will show up with a green tick, slightly abnormal with an exclamation mark in orange, and abnormal results with a cross highlighted in red. What problems seem to be affecting these reads?

Let's have a look at the QC parameters:

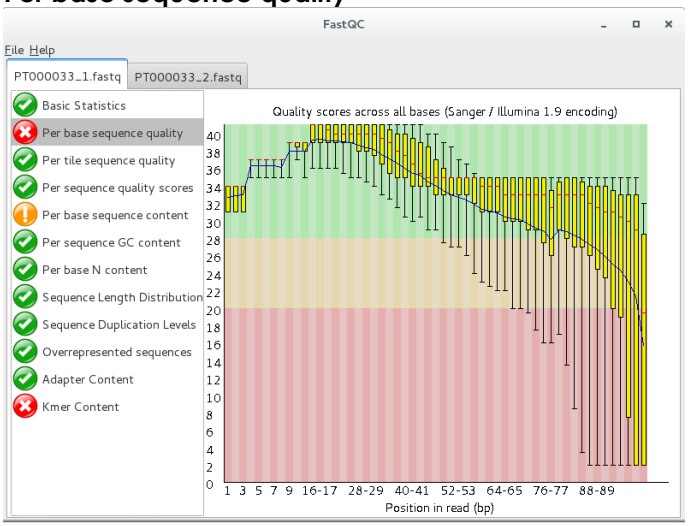
Basic Statistics



The Basic Statistics module generates some simple and self-explanatory composition statistics for the file analysed:

- Filename
- File type
- Encoding: Says which ASCII encoding of quality values was found in this file.
- Total Sequences
- Filtered Sequences
- Sequence Length
- %GC

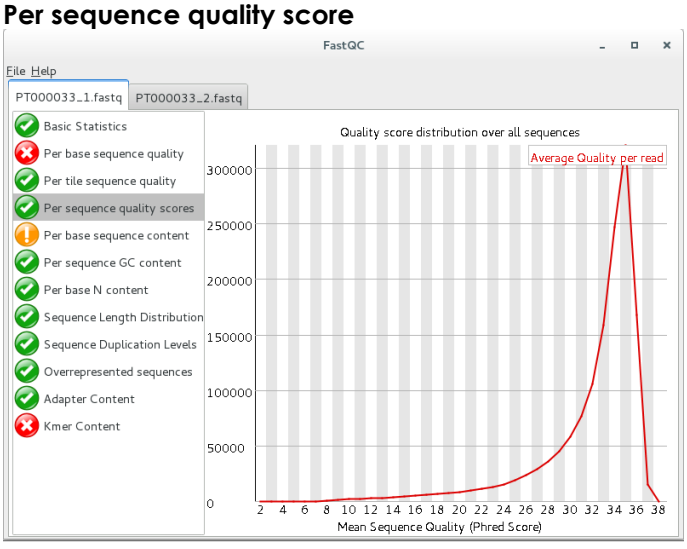
Per base sequence quality



Overview of the range of quality values across all bases at each position in the FastQ file. In general sequencing chemistry degrades with increasing read length and for long runs you may find that the general quality of the run falls to a level where a warning or error is triggered.

Warning: the lower quartile for any base <10, or median for any base < 25.Failure: lower quartile for any base is less than 5 or if the median for any base is less than 20. Phred33 scores below 20 are considered to be an indicator of poor quality.

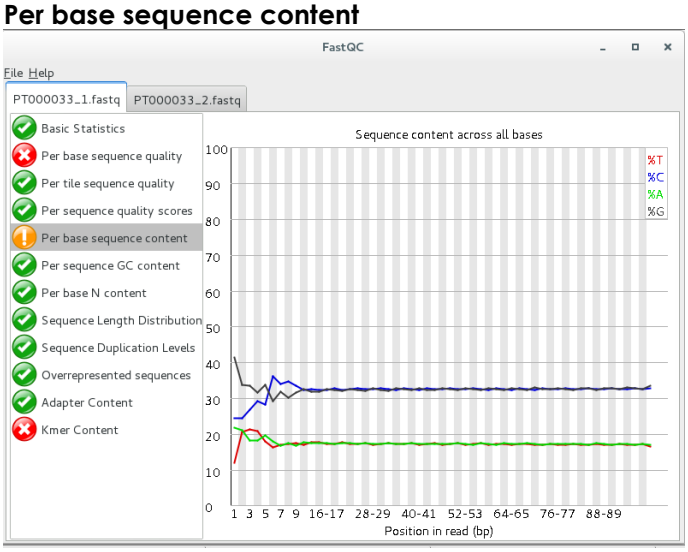
Per sequence quality score



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

A warning is raised if the most frequently observed mean quality is below 27 - this equates to a 0.2% error rate. An error is raised if the most frequently observed mean quality is below 20 - this equates to a 1% error rate.

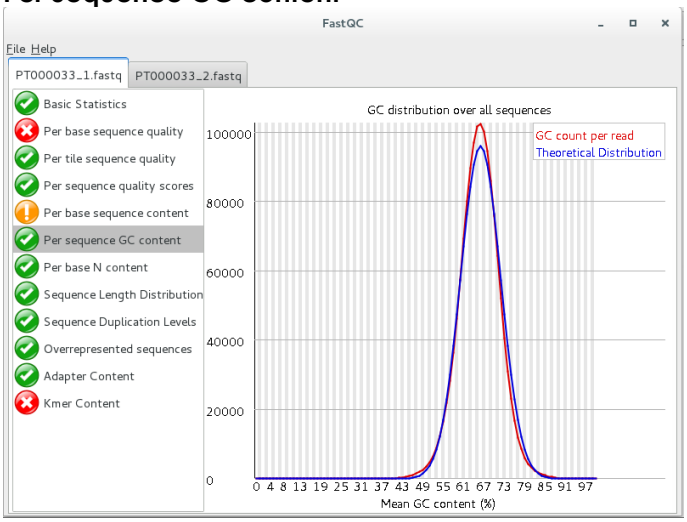
Per base sequence content



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

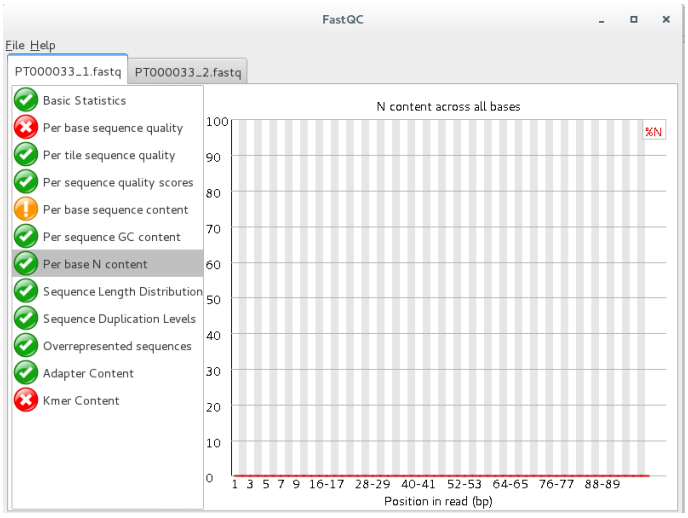
This module issues a warning if the difference between A and T, or G and C is greater than 10% in any position. This module will fail if the difference between A and T, or G and C is greater than 20% in any position.

Per Sequence GC content



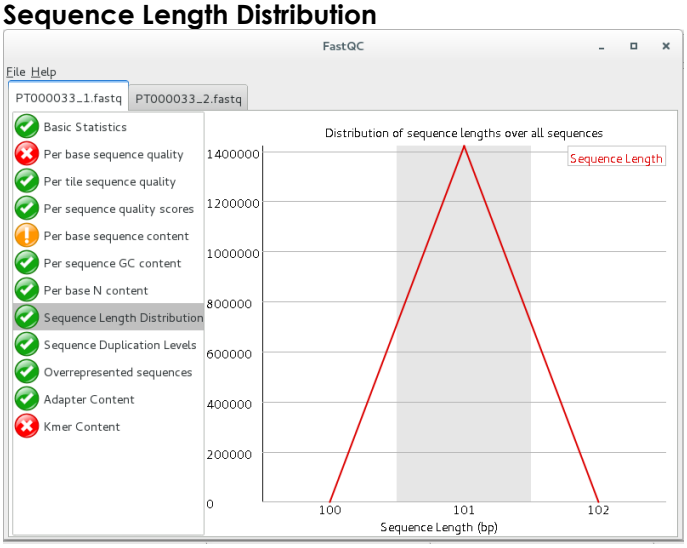
This module measures the GC content across the whole length of each sequence in a file and compares it to a modelled normal distribution of GC content.An unusually shaped distribution could indicate a contaminated library or some other kinds of biased subset. A normal distribution which is shifted indicates some systematic bias which is independent of base position. If there is a systematic bias which creates a shifted normal distribution then this won't be flagged as an error by the module since it doesn't know what your genome's GC content should be.
A warning is raised if the sum of the deviations from the normal distribution represents more than 15% of the reads. This module will indicate a failure if the sum of the deviations from the normal distribution represents more than 30% of the reads.

Per base N content



This module plots out the percentage of base calls at each position for which an N (uncalled base) was called. If the N proportion rises above a few percent it suggests that the analysis pipeline was unable to interpret the data well enough to make valid base calls. This module raises a warning if any position shows an N content of >5%. This module will raise an error if any position shows an N content of >20%.

Sequence Length Distribution

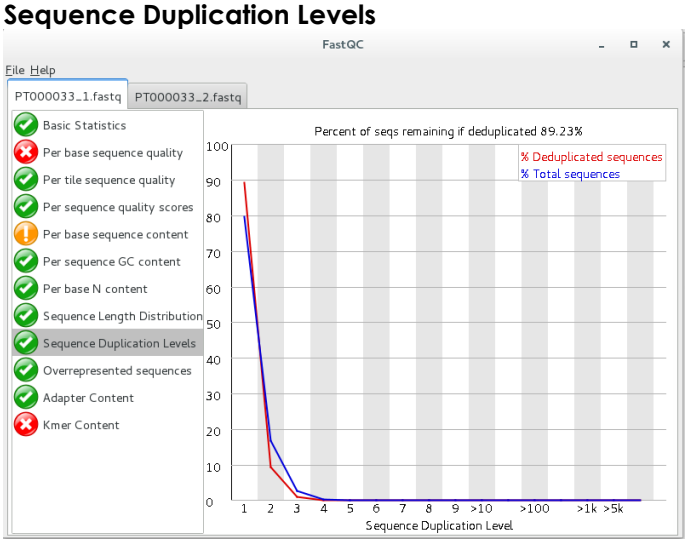


This module generates a graph showing the distribution of fragment sizes in the file which was analysed. In many cases this will produce a simple graph showing a peak only at one size, but for variable length FastQ files this will show the relative amounts of each different size of sequence fragment.

This module will raise a warning if all sequences are not the same length. This module will raise an error if any of the sequences have zero length.

For some sequencing platforms, it is entirely normal to have different read lengths so warnings here can be ignored.

Sequence Duplication Levels

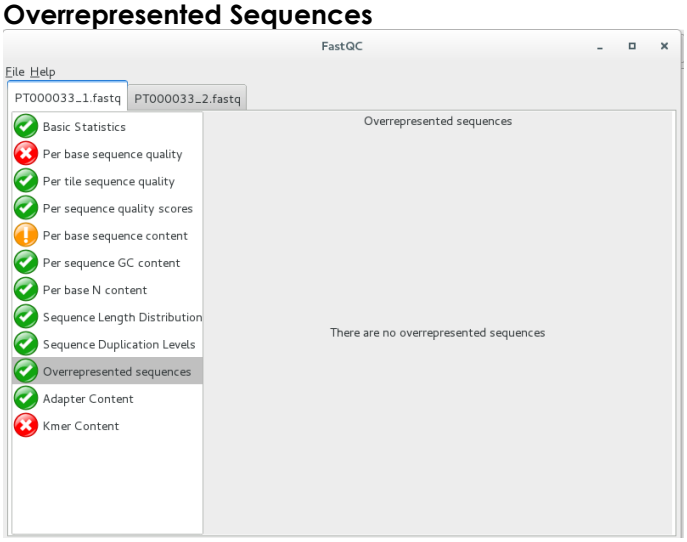


In a diverse library most sequences will occur only once in the final set. A low level of duplication may indicate a very high level of coverage of the target sequence, but a high level of duplication is more likely to indicate some kind of enrichment bias (eg PCR over amplification).

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

This module will issue a warning if non-unique sequences make up more than 20% of the total. This module will issue a error if non-unique sequences make up more than 50% of the total.

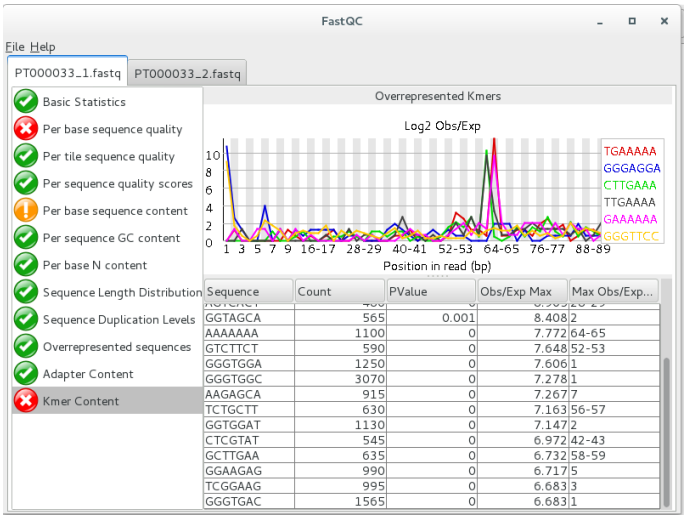
Overrepresented Sequences



A normal high-throughput library will contain a diverse set of sequences, with no individual sequence making up a tiny fraction of the whole. Finding that a single sequence is very overrepresented in the set either means that it is highly biologically significant, or indicates that the library is contaminated, or not as diverse as you expected. For each overrepresented sequence the program will look for matches in a database of common contaminants and will report the best hit it finds. It's also worth pointing out that many adapter sequences are very similar to each other so you may get a hit reported which isn't technically correct.

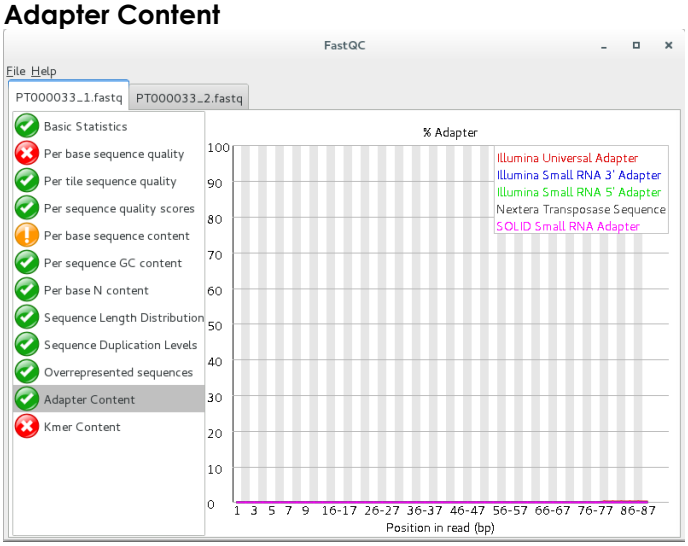
This module will issue a warning if any sequence is found to represent more than 0.1% of the total. This module will issue an error if any sequence is found to represent more than 1% of the total.

Kmer Content



The analysis of overrepresented sequences will spot an increase in any exactly duplicated sequences, but there are a different subset of problems where it will not work: if you have very long sequences with poor sequence quality then random sequencing; and if you have a partial sequence which is appearing at a variety of places within your sequence then this won't be seen either by the per base content plot or the duplicate sequence analysis. The Kmer module starts from the assumption that any small fragment of sequence should not have a positional bias in its apearance within a diverse library. Any Kmers with positionally biased enrichment are reported. The top 6 most biased Kmer are additionally plotted to show their distribution.

Adapter Content



One obvious class of sequences which you might want to analyse are adapter sequences. It is useful to know if your library contains a significant amount of adapter in order to be able to assess whether you need to adapter trim or not. Although the Kmer analysis can theoretically spot this kind of contamination it isn't always clear. This module therefore does a specific search for a set of separately defined Kmers and will give you a view of the total proportion of your library which contain these Kmers. A results trace will always be generated for all of the sequences present in the adapter config file so you can see the adapter content of your library, even if it's low. This module will issue a warning if any sequence is present in more than 5% of all reads. This module will issue a warning if any sequence is present in more than 10% of all reads.

For your reference: it is possible to save the report for each analysis. Just go to File>Save Report… and you will be able to save an html file containing the analysis displayed in FastQC.

Adapted from https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3 Analysis Modules/.

Exercise 3 – Sequence Read Trimming

Improving the quality of the read files is an essential step that prevents the emergence of errors in downstream steps such as erroneously mapped reads or incorrectly called variants [3]. We will be using a Java written command-line tool that is designed to trim and crop Illumina FASTQ files and can also be used to remove adapters: **Trimmomatic** [2]. This tool also has the advantage of parallelization across multiple cores thereby increasing its execution speed in multi-processor machines (multithreading). Trimmomatic can operate on Single-end and Paired-end mode. We will be using the paired-end mode where the software will keep the correspondence between read mates.

Learn more about Trimmomatic:

- Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. *Bioinformatics*, btu170.
- http://www.usadellab.org/cms/?page=trimmomatic

The Trimmomatic options are:

- ILLUMINACLIP: Cut adapter and other illumina-specific sequences from the read.
- SLIDINGWINDOW: Perform a sliding window trimming, cutting once the average quality within the window falls below a threshold.
- LEADING: Cut bases off the start of a read, if below a threshold quality
- TRAILING: Cut bases off the end of a read, if below a threshold quality
- CROP: Cut the read to a specified length
- HEADCROP: Cut the specified number of bases from the start of the read
- MINLEN: Drop the read if it is below a specified length
- TOPHRED33: Convert quality scores to Phred-33
- TOPHRED64: Convert quality scores to Phred-64

Let's get started, from your home directory go to the Module1 directory:

```
$ cd Module1

## Now let's start trimming PT000033 reads:

$ java -jar /opt/Trimmomatic-0.36/trimmomatic-0.36.jar PE -phred33
PT000033_1.fastq PT000033_2.fastq PT000033_1_trimmed_paired.fastq
PT000033_1_trimmed_unpaired.fastq PT000033_2_trimmed_paired.fastq
PT000033_2_trimmed_unpaired.fastq LEADING:3 TRAILING:3
SLIDINGWINDOW:4:20 MINLEN:36
```

What files did the software generated? Rerun the FastQC analysis for the paired-end files (*_paired.fastq) and take a look…

Exercise 4 – Mapping

Time to start mapping! To align sequence reads to a reference genome we must first choose one of the many short-read aligner software that are currently available. These softwares are based on alignment algorithms that can cope with the millions of reads produced for a single organism by NGS platforms and, are more efficient in doing this than traditional aligning algorithms. Different algorithms exist for this task as well as different software implementations [4, 5]. For a thorough review on mapping algorithms and sequence read alignment you can check the following articles:

- Mielczarek M, Szyda J. Review of alignment and SNP calling algorithms for next-generation sequencing data. *J Appl Genet*. 2016; **57**: 71-79.
- Li H, Homer N. A survey of sequence alignment algorithms for next-generation sequencing. *Brief Bioinform*. 2010; **11**: 473-483.

Most well-known aligners include MAQ, Bowtie, BWA, etc. The output of most of these aligners is a file containing the alignment in the SAM/BAM format which is the most widely used format to store NGS mapped reads [6]. We will look at this format ahead. In this course we will use one of the most popular aligners – Burrows-Wheeler Aligner (BWA) [7]. BWA comprehends different three alignment algorithms: BWA-backtrack (aln and sampe/samse); BWA-SV; and BWA-MEM. For 70bp or longer Illumina, 454, Ion Torrent and Sanger reads, assembly contigs and BAC sequences, BWA-MEM is usually the preferred algorithm. For short sequences, BWA-backtrack may be better. BWA-SW may have better sensitivity when alignment gaps are frequent.

In this exercise we will use the BWA-MEM algorithm. We will now map the reads from strain PT000033 to the *M.tuberculosis* H37Rv genome. Let's start by open a terminal window and from your home directory go to Module1 directory:

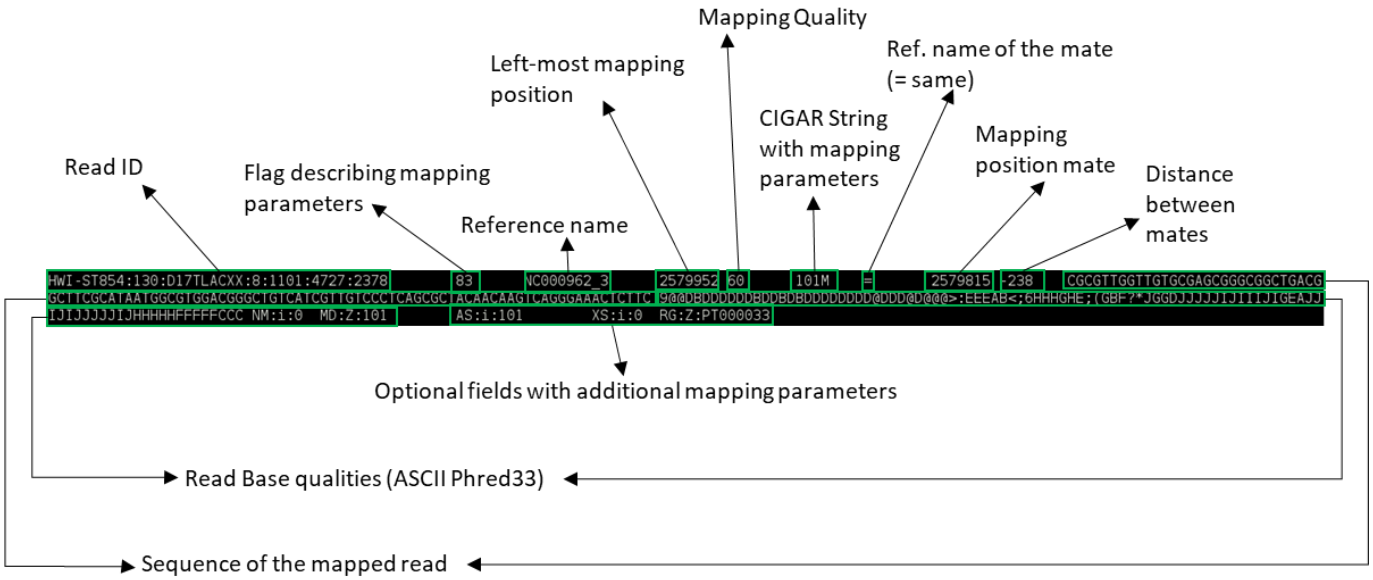
```
$ cd Module1

## First, we have to index the reference sequence (NC00962_3.fasta)
to allow efficient access by BWA upon mapping:

$ bwa index NC000962_3.fasta
```

```
## Next, we will map the trimmed paired reads obtained in the
previous exercise. We can also map the unpaired reads but we would
need to merge the resulting three SAM/BAM files. In this exercide
we will continue with the paired reads only:


$ bwa mem -R "@RG\tID:PT000033\tSM:PT000033\tPL:Illumina" -M
NC000962_3.fasta PT000033_1_trimmed_paired.fastq
PT000033_2_trimmed_paired.fastq > PT000033.sam


##The -R and -M option are included for compatibility with Picard
tools and GATK during variant calling. Although we will not use
these, the SAM file produced can be subsequently analysed by these
programs if necessary. The -R option specifies the read group
header line and the -M option is grants compatibility with Picard
tools.
```

The commands above should have generated a SAM file. Let's look at its content:

```
$ more PT000033.sam
```

This seems rather complex but it is in fact a flexible format to store the coordinates of mapped and unmappeded reads, associated Phred33 Q scores and chromosome. See the figure below to have an idea of what it represents (the image below represents a single line):



You can also use the built-in word-count program in Linux to check the number of lines in this SAM file:

```
$ wc -l PT000033.sam
```

Now that we have a SAM file we can convert it to its compressed binary format (which cannot be read as a text file but can be read with Samtools view). We also need to sort the BAM file by coordinates to allow efficient access and analysis and index this BAM file. The BAM index file (*.bai) is required by some visualization and downstream analysis software, as it allows rapid access to the BAM file.

```
## We will again index the reference sequence, this time with
Samtools:

$ samtools faidx NC000962_3.fasta

## We can then convert the SAM file to a BAM file:

$ samtools view -bt NC000962_3.fasta.fai PT000033.sam >
PT000033.bam

## Sorting:

$ samtools sort PT000033.bam -o PT000033.sorted.bam

## Finally, we will index the sorted bam file (if posteriorly you
change the name of the BAM file it is necessary to re-index this
same file):

$ samtools index PT000033.sorted.bam
```

You can produce some basic mapping statistics by using samtools flagstat command:

```
$ samtools flagstat PT000033.sorted.bam

## This will output the result to standard output (screen), or you
can alternatively save it in a new file (PT000033_stats.txt)

$ samtools flagstat PT000033.sorted.bam > PT000033_stats.txt
```

At this point we have checked the quality of reads (QC), trimmed low-quality reads and bases and mapped the sequence reads to a reference genome. We can now proceed to variant calling but let's first look visualize the mapped reads onto a reference chromosome.

Exercise 5 – Visualization of Mapped Data

There are several programs to visualize genomic data, including annotated genes and mapped reads: Artemis, Tablet, IGB, etc [8-10]. Artemis and Tablet are available in the Linux Operating System Image made available for this course. However, we will be using Artemis in this exercise. Artemis is a powerful visualization and annotation tool with multiple functionality, having the advantage of being a Java written platform and can therefore be used on Windows, MacOS or GNU/Linux platforms [9].

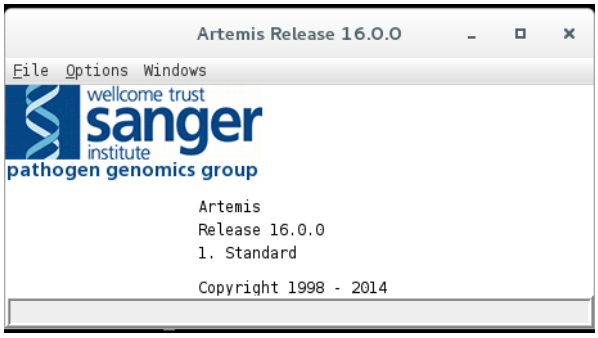
To use artemis we must first open a genome, this can be either in the format of a fasta file, such as the file containing the reference genome we used in the previous exercise or for example a GenBank file (*gb or *gbk). The latter has the advantage of containing the sequence data along with the annotation for genes and open reading frames (ORFs). It is also possible to open a FASTA file and subsequently read in entries from a GFF3 file (annotation). All these file types can be easily retrieved from GenBank (see send to file on: https://www.ncbi.nlm.nih.gov/nuccore/NC_000962; don't forget to check the Show sequence on the Display options).

On the Module1 directory the NC000962_3.gbk file is available. Please open it with Artemis.

To start Artemis either double-click on the shortcut on the desktop or open up a terminal window and type:

```
$ art
```

This will open an Artemis window:



Go to File>Open… > then select the NC000962_3.gbk and click OPEN. You can skip the warnings.

A new window will open:

The Artemis window has three main views. The first contains the plus and minus DNA strands along with their respective three reading frames. You can see genes annotated on these reading frames. The middle view is similar but zooming at the nucleotide level along with respective amino acids at each reading frame. The bottom view shows each

annotated feature, starting and ending genomic positions and, additional information on the gene function or similarity, etc., if available.

Before reading more data in, navigate around, you can use the horizontal scroll bar on the top view to move along the genome, and you can use the vertical scroll bar to zoom in and out. To move to regions more efficiently, click on Goto>Navigator… to open the navigator window. Try to search for a gene of your interest (e.g. *rpsL*). After selecting, note that you can view the gene in FASTA format by right-clicking on it and then View>Bases>Bases of Selection as FASTA. Similarly by doing the same thing but by starting with Create you can save the FASTA directly to a file of your choice.

You can also add additional plots by, e.g., going to Graphs>GC Content (%).

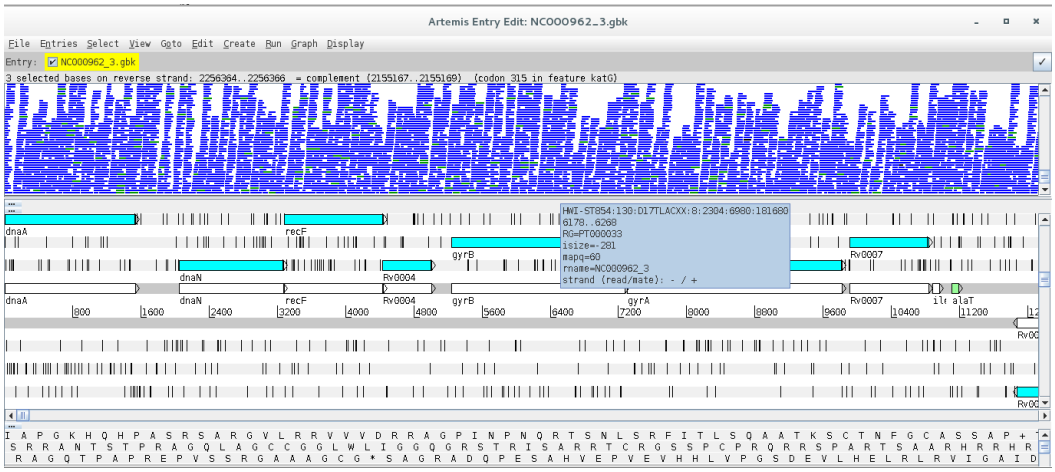
These are just some of the functionalities.

Tip: To select bases or amino acids inside a feature click the region of interest while pressing Alt.

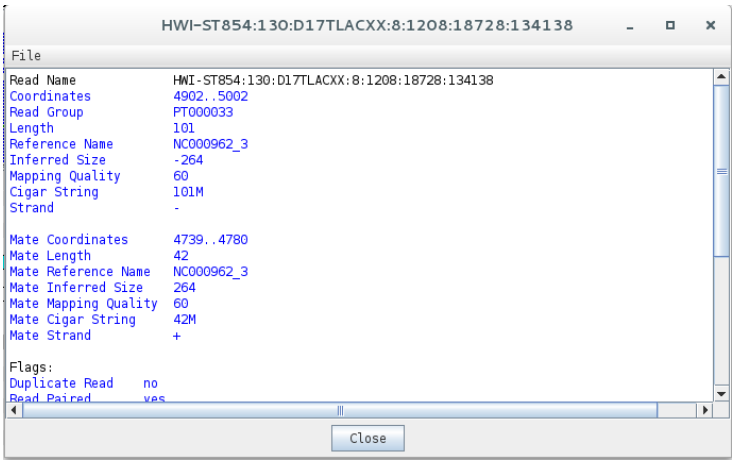
Next, it is time to look at your alignment file. Go to File>Read BAM/VCF and open the sorted BAM file you created in the previous exercise for the PT000033 strain.

(**Attention:** if the BAM file is not indexed and the bai file is not in same directory Artemis won't be able to open it)

Something like this should appear:



Now, there is a new view representing the mapping location of every successfully mapped read. Moreover, you can for example select any read right-click and then click the Show Details of the read. A new window will appear with data on the sequence read:

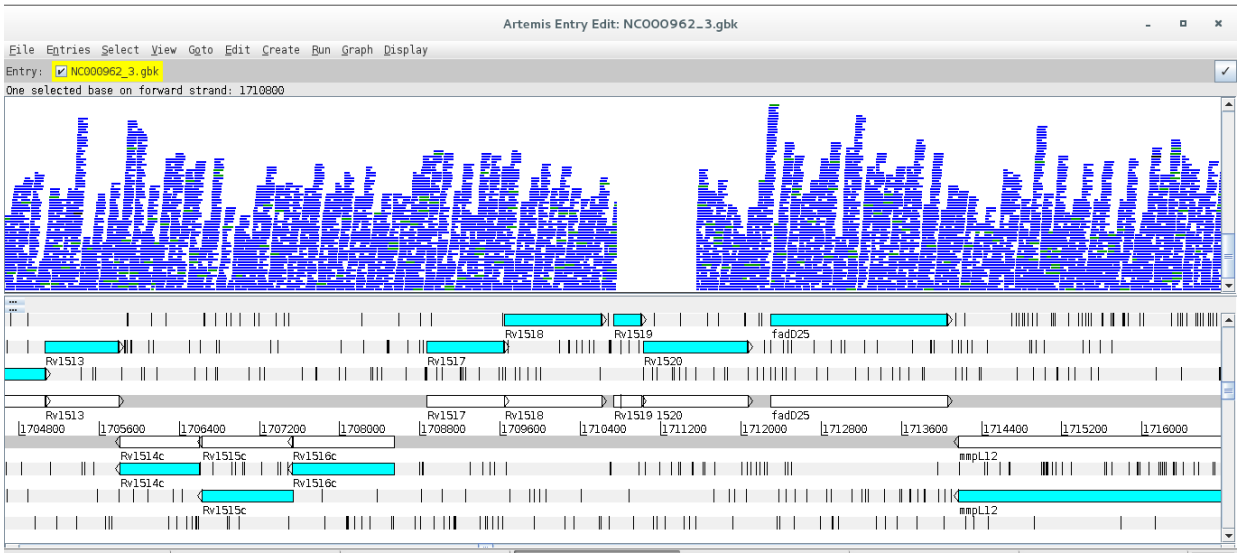


Does this information remind you of anything? Perhaps the data contained in the SAM file…

Another useful option is to look at coverage plots. Right click on the stacked view and select Views>Coverage. What happened?

Differences in coverage:

Let's again select the Stack view from the Views>Coverage. Next, using the Navigator window let's go the region around nucleotide 1710800. What is happening here?

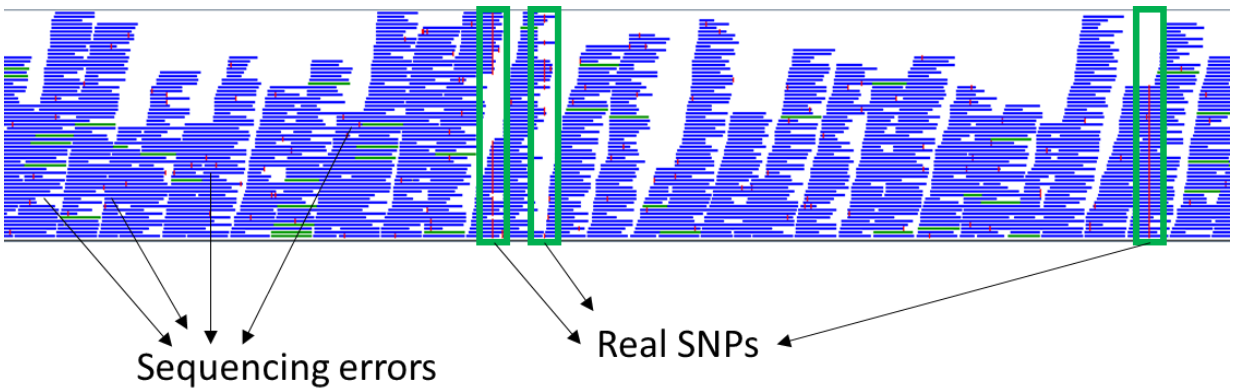


Now go to the article by Gagneux et al 2006 entitled "Variable Host-Pathogen Compatibility" and check Supplementary Table 4?

What can you conclude from this?

What about Single Nucleotide Polymorphisms?

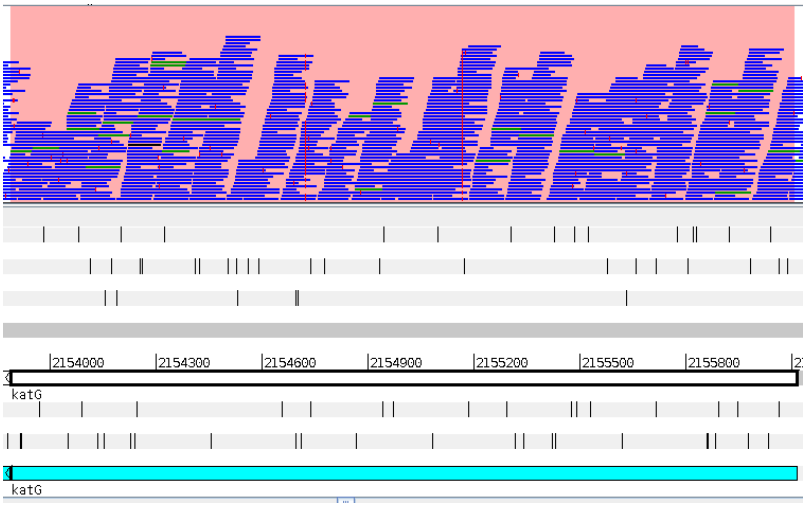
Another important aspect is to look for SNPs. Return to the stack view and to the beginning of the genome. Now on the stack view righ-click Show>SNP Marks. Each SNP is highlighted by a red marking on the respective read. Take a look at it. Why are there so many red markings? How to distinguish those from the real SNPs?



Note that real SNPs consistently show up in almost all reads, whereas sequencing errors appear dispersed.

The strain we have been analysing is available at CPLP-TB (http://cplp-tb.ff.ulisboa.pt). Go to CPLP-TB and find this specific strain. What is its resistance pattern?

If you already went to CPLP-TB you can confirm that this strain is monorresistant to Isoniazid (INH). To uncover the molecular basis of resistance in this isolate we can start by looking at the main gene associated with INH resistance: *katG*. Using the Navigator go to the katG gene. How many real SNPs can you detect and which one is associated with resistance?



Now, isn't there an easier way to detect and visualize SNP variants? Yes: **Variant Calling**.

Exercise 6 – Variant Calling

Different software packages are also available for variant calling. Three main programs are available: samtools mpileup, GATK and FreeBayes [6, 11]. In this practical we will use samtools mpileup in combination with Bcftools for filtering variants. A good approach is also to use multiple callers and combining the data to produce more robust datasets. At this point, the parameters used by variant calling software play a crucial role in correct variant identification [3].

Open a terminal and type:

```
$ samtools mpileup
```

The complete parameter listing for samtools mpilup should appear:

```
Usage: samtools mpileup [options] in1.bam [in2.bam [...]]

Input options:
  -6, --illumina1.3+      quality is in the Illumina-1.3+ encoding
  -A, --count-orphans     do not discard anomalous read pairs
  -b, --bam-list FILE     list of input BAM filenames, one per line
  -B, --no-BAQ            disable BAQ (per-Base Alignment Quality)
  -C, --adjust-MQ INT     adjust mapping quality; recommended:50, disable:0 [0]
  -d, --max-depth INT     max per-BAM depth; avoids excessive memory usage [250]
  -E, --redo-BAQ          recalculate BAQ on the fly, ignore existing BQs
  -f, --fasta-ref FILE    faidx indexed reference sequence file
  -G, --exclude-RG FILE   exclude read groups listed in FILE
  -l, --positions FILE    skip unlisted positions (chr pos) or regions (BED)
  -q, --min-MQ INT        skip alignments with mapQ smaller than INT [0]
  -Q, --min-BQ INT        skip bases with baseQ/BAQ smaller than INT [13]
  -r, --region REG        region in which pileup is generated
  -R, --ignore-RG         ignore RG tags (one BAM = one sample)
  --rf, --incl-flags STR|INT  required flags: skip reads with mask bits unset []
  --ff, --excl-flags STR|INT  filter flags: skip reads with mask bits set
                                [UNMAP,SECONDARY,QCFAIL,DUP]
  -x, --ignore-overlaps   disable read-pair overlap detection

Output options:
  -o, --output FILE       write output to FILE [standard output]
  -g, --BCF               generate genotype likelihoods in BCF format
  -v, --VCF               generate genotype likelihoods in VCF format

Output options for mpileup format (without -g/-v):
  -O, --output-BP         output base positions on reads
  -s, --output-MQ         output mapping quality

Output options for genotype likelihoods (when -g/-v is used):
  -t, --output-tags LIST  optional tags to output:
            DP,AD,ADF,ADR,SP,INFO/AD,INFO/ADF,INFO/ADR []
  -u, --uncompressed      generate uncompressed VCF/BCF output

SNP/INDEL genotype likelihoods options (effective with -g/-v):
  -e, --ext-prob INT      Phred-scaled gap extension seq error probability [20]
  -F, --gap-frac FLOAT    minimum fraction of gapped reads [0.002]
  -h, --tandem-qual INT   coefficient for homopolymer errors [100]
  -I, --skip-indels       do not perform indel calling
  -L, --max-idepth INT    maximum per-sample depth for INDEL calling [250]
  -m, --min-ireads INT    minimum number gapped reads for indel candidates [1]
  -o, --open-prob INT     Phred-scaled gap open seq error probability [40]
  -p, --per-sample-mF     apply -m and -F per-sample for increased sensitivity
  -P, --platforms STR     comma separated list of platforms for indels [all]
     --input-fmt-option OPT[=VAL]
            Specify a single input file format option in the form
            of OPTION or OPTION=VALUE
     --reference FILE
            Reference sequence FASTA FILE [null]

Notes: Assuming diploid individuals.
```

The samtools mpileup command generates VCF, BCF or pileup for one or multiple BAM files and the calling process is completed by the Bcftools [12]. Therefore, we have to pipe (' | ') the output from samtools mpileup to bcftools call. On a terminal window, type:

```
$ bcftools call
```

… to list bcftools call parameters and options:

```
Usage:  bcftools call [options] <in.vcf.gz>

File format options:
      --no-version              do not append version and command line to the header
  -o, --output <file>           write output to a file [standard output]
  -O, --output-type <b|u|z|v>   output type: 'b' compressed BCF; 'u' uncompressed BCF; 'z' compressed VCF; 'v' uncompressed VCF [v]
      --ploidy <assembly>[?]    predefined ploidy, 'list' to print available settings, append '?' for details
      --ploidy-file <file>      space/tab-delimited list of CHROM,FROM,TO,SEX,PLOIDY
  -r, --regions <region>        restrict to comma-separated list of regions
  -R, --regions-file <file>     restrict to regions listed in a file
  -s, --samples <list>          list of samples to include [all samples]
  -S, --samples-file <file>     PED file or a file with an optional column with sex (see man page for details) [all samples]
  -t, --targets <region>        similar to -r but streams rather than index-jumps
  -T, --targets-file <file>     similar to -R but streams rather than index-jumps
      --threads <int>           number of extra output compression threads [0]

Input/output options:
  -A, --keep-alts               keep all possible alternate alleles at variant sites
  -f, --format-fields <list>    output format fields: GQ,GP (lowercase allowed) []
  -F, --prior-freqs <AN,AC>     use prior allele frequencies
  -g, --gvcf <int>,[...]        group non-variant sites into gVCF blocks by minimum per-sample DP
  -i, --insert-missed           output also sites missed by mpileup but present in -T
  -M, --keep-masked-ref         keep sites with masked reference allele (REF=N)
  -V, --skip-variants <type>    skip indels/snps
  -v, --variants-only           output variant sites only

Consensus/variant calling options:
  -c, --consensus-caller        the original calling method (conflicts with -m)
  -C, --constrain <str>         one of: alleles, trio (see manual)
  -m, --multiallelic-caller     alternative model for multiallelic and rare-variant calling (conflicts with -c)
  -n, --novel-rate <float>,[...] likelihood of novel mutation for constrained trio calling, see man page for details [1e-8,1e-9,1e-9]
  -p, --pval-threshold <float>  variant if P(ref|D)<FLOAT with -c [0.5]
  -P, --prior <float>           mutation rate (use bigger for greater sensitivity), use with -m [1.1e-3]
```

Let's start the variant calling process, type the following commands in the Module1 directory:

```
$ samtools mpileup -Q 23 -d 2000 -C 50 -ugf NC000962_3.fasta
PT000033.sorted.bam | bcftools call -O v -vm -o PT000033.raw.vcf

## Notice that you have set the -Q option to skip aligned bases
with quality below 23 and set the maximum read depth to 2000.

$ vcfutils.pl varFilter -d 10 -D 2000 PT000033.raw.vcf >
PT000033.filt.vcf

## Here, we used the vcfutils.pl perl script from BCFtools to
filter the raw vcf file with the -d option to set the minimum read
depth of 10 to call a variant and the -D option to again set the
maximum read depth to 2000.
```

The PT000033.filt.vcf file contains the called SNPs and small INDELs. Large structural variants that are longer than the read length must be identified using other approaches and software.

But let's look at the VCF (Variant Call Format) format, type:

```
$ more PT000033.filt.vcf
```

The VCF file is composed of a number of header lines starting with the hash symbol ('#') containing metadata and other information, followed by 10 columns of data on called variants:

```
##fileformat=VCFv4.2
##FILTER=<ID=PASS,Description="All filters passed">
##samtoolsVersion=1.3+htslib-1.3
##samtoolsCommand=samtools mpileup -Q 23 -d 2000 -C 50 -ugf NC000962_3.fasta PT000033.sorted.bam
##reference=file://NC000962_3.fasta
##contig=<ID=NC000962_3,length=4411532>
##ALT=<ID=*,Description="Represents allele(s) other than observed.">
##INFO=<ID=INDEL,Number=0,Type=Flag,Description="Indicates that the variant is an INDEL.">
##INFO=<ID=IDV,Number=1,Type=Integer,Description="Maximum number of reads supporting an indel">
##INFO=<ID=IMF,Number=1,Type=Float,Description="Maximum fraction of reads supporting an indel">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Raw read depth">
##INFO=<ID=VDB,Number=1,Type=Float,Description="Variant Distance Bias for filtering splice-site artefacts in RNA-seq data (bigger is better)",Version="3">
##INFO=<ID=RPB,Number=1,Type=Float,Description="Mann-Whitney U test of Read Position Bias (bigger is better)">
##INFO=<ID=MQB,Number=1,Type=Float,Description="Mann-Whitney U test of Mapping Quality Bias (bigger is better)">
##INFO=<ID=BQB,Number=1,Type=Float,Description="Mann-Whitney U test of Base Quality Bias (bigger is better)">
##INFO=<ID=MQSB,Number=1,Type=Float,Description="Mann-Whitney U test of Mapping Quality vs Strand Bias (bigger is better)">
##INFO=<ID=SGB,Number=1,Type=Float,Description="Segregation based metric.">
##INFO=<ID=MQOF,Number=1,Type=Float,Description="Fraction of MQ0 reads (smaller is better)">
##FORMAT=<ID=PL,Number=G,Type=Integer,Description="List of Phred-scaled genotype likelihoods">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##INFO=<ID=ICB,Number=1,Type=Float,Description="Inbreeding Coefficient Binomial test (bigger is better)">
##INFO=<ID=HOB,Number=1,Type=Float,Description="Bias in the number of HOMs number (smaller is better)">
##INFO=<ID=AC,Number=A,Type=Integer,Description="Allele count in genotypes for each ALT allele, in the same order as listed">
##INFO=<ID=AN,Number=1,Type=Integer,Description="Total number of alleles in called genotypes">
##INFO=<ID=DP4,Number=4,Type=Integer,Description="Number of high-quality ref-forward , ref-reverse, alt-forward and alt-reverse bases">
##INFO=<ID=MQ,Number=1,Type=Integer,Description="Average mapping quality">
##bcftools_callVersion=1.5-26-g81c6dd3+htslib-1.5-14-ge1380c8
##bcftools_callCommand=call -O v -vm -o PT000033.raw.vcf; Date=Tue Aug 22 13:51:27 2017
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  PT000033
NC000962_3      1849    .       C       A       142     .       DP=12;VDB=0.00174423;SGB=-0.670168;MQOF=0;AC=2;AN=2;DP4=0,0,0,10;MQ=45  GT:PL   1/1:172,30,0
NC000962_3      1977    .       A       G       154     .       DP=10;VDB=0.198441;SGB=-0.651104;MQSB=1;MQOF=0;AC=2;AN=2;DP4=0,0,7,1;MQ=44      GT:PL   1/1:184,24,0
NC000962_3      3446    .       C       T       225     .       DP=54;VDB=0.0158223;SGB=-0.693147;MQSB=0.97197;MQOF=0;AC=2;AN=2;DP4=0,0,28,19;MQ=45     GT:PL   1/1:255,141,0
NC000962_3      4013    .       T       C       225     .       DP=49;VDB=0.0969683;SGB=-0.693146;MQSB=0.0153297;MQOF=0;AC=2;AN=2;DP4=0,0,25,19;MQ=45   GT:PL   1/1:255,132,0
NC000962_3      7362    .       G       C       225     .       DP=62;VDB=0.827911;SGB=-0.693147;MQSB=0.971079;MQOF=0;AC=2;AN=2;DP4=0,0,30,23;MQ=44     GT:PL   1/1:255,160,0
NC000962_3      7585    .       G       C       225     .       DP=46;VDB=0.761957;SGB=-0.693146;MQSB=0.202323;MQOF=0;AC=2;AN=2;DP4=0,0,14,28;MQ=44     GT:PL   1/1:255,126,0
NC000962_3      8405    .       C       A       225     .       DP=31;VDB=0.121957;SGB=-0.693079;MQSB=0.960561;MQOF=0;AC=2;AN=2;DP4=0,0,14,15;MQ=45     GT:PL   1/1:255,87,0
NC000962_3      9304    .       G       A       225     .       DP=33;VDB=0.502902;SGB=-0.693054;MQSB=0.267776;MQOF=0;AC=2;AN=2;DP4=0,0,19,9;MQ=45      GT:PL   1/1:255,84,0
NC000962_3      11879   .       A       G       225     .       DP=31;VDB=0.00639057;SGB=-0.693079;MQSB=0.840657;MQOF=0;AC=2;AN=2;DP4=0,0,15,14;MQ=44   GT:PL   1/1:255,87,0
NC000962_3      12204   .       G       A       225     .       DP=22;VDB=0.246837;SGB=-0.692067;MQSB=0.984877;MQOF=0;AC=2;AN=2;DP4=0,0,5,15;MQ=44     GT:PL   1/1:255,60,0
NC000962_3      14785   .       T       C       225     .       DP=37;VDB=0.400878;SGB=-0.693127;MQSB=0.976914;MQOF=0;AC=2;AN=2;DP4=0,0,16,17;MQ=44    GT:PL   1/1:255,99,0
NC000962_3      15117   .       C       G       225     .       DP=36;VDB=0.22674;SGB=-0.693132;MQSB=0.997491;MQOF=0;AC=2;AN=2;DP4=0,0,13,21;MQ=44     GT:PL   1/1:255,102,0
NC000962_3      21795   .       G       A       225     .       DP=28;VDB=0.40874;SGB=-0.692717;MQSB=0.95306;MQOF=0;AC=2;AN=2;DP4=0,0,10,13;MQ=44     GT:PL   1/1:255,69,0
NC000962_3      24007   .       G       T       225     .       DP=36;VDB=0.921407;SGB=-0.693136;MQSB=0.99869;MQOF=0;AC=2;AN=2;DP4=0,0,22,13;MQ=44     GT:PL   1/1:255,105,0
```

Column	Field	Description
1	CHROM	Chromosome name
2	POS	Left-most position of the variant
3	ID	Unique variant identifier
4	REF	Reference allele
5	ALT	Alternate allele
6	QUAL	Variant/Reference Quality
7	FILTER	Filters applied
8	INFO	Information related to the variant, separated by semi-colon
9	FORMAT	Format of the genotype fields, separted by colon (optional)
10	SAMPLE	Sample Genotypes and per-sample information (optional)

You can open the vcf file with any text editor or even import it into an Excel spreadsheet. You can right-click on the file and select Open with LibreOffice Calc and import it as a file containing fields separated by tabs.

We can also view the vcf file in Artemis, but first it is necessary to compress the vcf file and index with tabix (from BCFtools):

```
$ bgzip -c PT000033.filt.vcf > PT000033.filt.vcf.gz
$ tabix -p vcf PT000033.filt.vcf.gz
```

Some software only read compressed vcf files indexed with tabix. Indexation with tabix produces an index file (*.tbi) that should be in the same directory as the compressed vcf file (*.vcf.gz).

To open the vcf file in Artemis repeat the steps from Exercise 5 and load the NC000962.gbk files and read-in the PT000033.sorted.bam file. Now, go again to File>Read BAM/VCF and select the compressed vcf file (PT000033.filt.vcf.gz). Another view should appear with vertical lines highlighting the variants on the VCF file. It is generally recommended not to consider variants with QUAL score below 30. Recall the Phred33 table in Exercise 1. What is the error probability of QUAL 30 score?

You can apply different sorts of filters by right-clicking on the VCF view and selecting Filters.

Inside Module1 directory, there is a sub-directory called vcfs. This directory contains additional VCF files for you to use. You can add these vcf files for comparison purposes. Right-click on the VCF view and select Add VCF …

[Alternatively, you can generate these VCF files yourself by repeating the steps from the different exercises in this module using the FASTQ files in the course_files directory, see Exercise 1].

Exercise 7 – Variant Functional Annotation

The last, and optional, exercise of this module pertains the functional annotation of the VCF file. This last step is extremely useful as it will annotate each variant with the respective genetic impact. Not only we will be able to directly evaluate if each variant is synonymous or non-synonymous but it will be also possible to directly see the genetic mutation and affected gene.

We will use another Java written program – SnpEff - that requires some configuration that is ouside the scope of this module. However, the online documentation available at http://snpeff.sourceforge.net/SnpEff_manual.html is very comprehensive and contains detailed steps on how to download or build its database. SnpEff takes the uncompressed VCF file as input and generates another VCF file annotated with additional information in the INFO field (ANN) [13]. We can do this by typing:

```
$ java -Xmx4g -jar /opt/snpEff/snpEff.jar -no-downstream -no-upstream -
v -c /opt/snpEff/snpEff.config NC000962_3 PT000033.filt.vcf >
PT000033.ann.vcf

## With this command, we are specifying the path to SnpEff configuration
file using the -c option; the -v option turns on the verbose mode; and
the -no-downstream and -no-upstream options disable the annotation of
downstream and upstream variants. We selected this option as otherwise
the VCF file would be overpopulated with upstream and downstream modifier
variants as genes in bacteria are very close to each other.
```


This VCF file can also be loaded in Artemis although it has to be compressed with bgzip and indexed with tabix.

Let us look at this new VCF file by typing:

```
$ more PT000033.ann.vcf
```

Can you spot the difference?

What about the INH-associated mutation that we detected visually? Can you find it using grep (sort of a Linux/GNU built-in Find):

```
$ grep “katG” PT000033.ann.vcf
```

Tip: Try to open this newly annotated VCF file on Excel/LibreOffice Calc. Down the line, parsing this format in R will allow you to carry out more advanced statistical analysis.

:: *De novo* Assembly ::

Introduction

This second module explores an alternative approach to reconstruct genomic data from NGS reads – ***de novo* Assembly**. Rather than using a mapping strategy to guide the assembly of sequence reads, de novo Assembly attempts to reconstruct genomes by exploring read overlapping and contiguity. This approach is more informative when we are dealing with a new species for which there is no reference genome or even a new strain of a well-known pathogen [14, 15].

However, the process of assembling reads into contigs can be challenging due to the huge number of reads produced from random sampling of the genome being studied. Additionally, high sequencing error rate, repeat regions/patterns and uneven sequencing depth are some of the problems that can influence the correct assembly of a genome. There are three major approaches for assembling reads: i) **overlap-and-extend**; ii) **string graph**; and iii) **de Bruijn graphs**. Overlap-and-extend methods iteratively attempt to first find read overlaps (where the suffix of a read is equal to the prefix of another read with alength that meets a defined threshold) and then extend the first read constructing a longer read (SSAKE, VCAKE and SHARCGS). The String Graph based assemblers constructs a string graph for every read in which each read is a vertex and there is an edge from a vertex to another if there is read overlap (Edena and BOA). These first two approaches are more susceptible to sequencing errors and can lead to more memory consumption [15, 16].

De Bruijn Graph-based algorithms are presently the most widely used approaches and are used in several software packages for genome assembly from NGS reads (e.g. Velvet, SOAPdenovo, SPAdes, etc) [17, 18]. Graphs are mathematical structures used to model pairwise relations between objects. In essence, each vertex represents a length-*k* substring (*k*-mer) in a read and there is a directed edge from vertex *u* to vertex *v* if *u* and *v* are consecutive *k*-mers in a read, i.e., the last *k*–1 nucleotides of the *k*-mer represented by *u* is the same as the first *k*–1 nucleotides of the *k*-mer represented by *v* [16].

Exercise 1 – De Bruijn Graphs

Before going to the computational part we can do a simple and quick hands-on exercise that covers the basics of de Bruijn Graph based assemblies. A concept that will be important to retain is the **concept of *k*-mer**. As explained above the k-mer is a substring

of a read and, a read can thus have multiple k-mers depending on the k-mer length (sometimes this is also referred as hash length or word length). For example, if you have a 50bp read it can only yield one 50bp k-mer, but it can yield two 49bp k-mers [17]. The caveat of this approach is that the k-mer length you choose must always be below the read-length of your data.

Let's consider the following reads:

ATGCG
GGCGTT
GCGGGC
TGATC
GTTGAT

We can use a simple overlap and extend approach but let's use the de Bruijn Graph approach to find *k*-mers. List all 3 bp k-mers from the reads above:

First, link the k-mers that only differ by k-1 nucleotides. Then, you can try to establish links by creating a path between the nodes to find your contig – each k-mer is a node. The two main rules are: **visit all nodes at least once** and **use the minimal path length.**

Write the final contig here:

You can perform a BLAST search and try to identify to where this region belongs to!

Exercise 2 – *De novo* Assembly

Now that you have attempted to do de novo assembly by hand it is time to use real data. We'll continue with the PT000033 strain from Module 1 but, instead of mapping, we will do de novo assembly. Let's go to Module2 directory by typing (from your home directory):

```
$ cd Module2
```

Inside this sub-directory (you can see its content using the ls command) you will find the fastq files used in Module 1, including the files produced by Trimmomatic. In this exercise we will use Velvet to assemble the PT000033 strain from sequence reads without using a reference genome [19].

You can also find in this sub-directory the reference genome used in Module 1. Although we will not use it in the assembly process, this genome file will be necessary in Exercise 3 for ordering the contigs.

This will be a simple approach to computational assembly using **Velvet**, which relies on two main programs: velveth and velvetg. velveth takes in a number of sequence files, produces a hash table, then outputs two files in an output directory (creating it if necessary), Sequences and Roadmaps, which are necessary to velvetg. The file formats supported by velveth include FASTA, FASTQ, compressed (gunzip) FASTA/FASTQ files, SAM/BAM, etc. Moreover, there are different read categories that can be specified depending on the data: -short (default), -shortPaired (for paired-end short read data), -short2 (to specify a second library), -shortPaired2 (idem), -long (for Sanger, 454 or even reference sequences) and -longPaired.

The second program, velvetg, is the core of Velvet where the de Bruijn graph is built and then manipulated.

Perhaps the most important parameter you must specify is the hash length or *k*-mer length. In fact, it cannot be inferred from the data such as other parameters that we are setting to auto or using default values. The two main rules for choosing the *k*-mer length are: it must be an odd number, to avoid palindromes (if you specify an even number Velvet will automatically decrease it); and, it must be inferior to the read length. Longer *k*-mers generally provide higher specificity but decrease coverage (and therefore sensitivity).

But, let's start assembling. Inside Module2 directory type:

```
$ velveth PT000033_41 41 -fastq -shortPaired
PT000033_1_trimmed_paired.fastq PT000033_2_trimmed_paired.fastq -
fastq -short PT000033_1_trimmed_unpaired.fastq
PT000033_2_trimmed_unpaired.fastq

# This first command creates the hash table in a new directory
PT000033_21 (it will contain the assembly using a k-mer length of
21)

$ velvetg PT000033_41 -exp_cov auto -cov_cutoff auto

# This second command produces the graph and assembles the genome.
You can find your files in the newly created PT000033_21 sub-
directory.
```


Notice that in the first command you specified two distinct libraries: a paired library present across two files using the -shortPaired option and two unpaired libraries using the -short option. We could have just used the paired library but in this way we do not loose good quality data.

Upon velvetg completion look at the last line summarising the results. Also, you can run the following command to find out the number of contigs and some additional statistics:

```
$ assemblathon_stats.pl ./PT000033_41/contigs.fa
```

This command will output additional statistics. The assemblathon_stats.pl (Keith Bradnam, UC Davis) is a Perl written script that calculates some statistics for assemblies and scaffolds (we will talk about this ahead). It was written for the Assemblathon contests to assess state-of-the-art methods in the field of genome assembly [14]. You can look at some of these metrics and take note in the table below. The output will be similar to the one below and as we are only working with assembled contigs we can focus on the third section only:

Number of contigs	268	
Number of contigs in scaffolds	0	
Number of contigs not in scaffolds	268	
Total size of contigs	4343956	
Longest contig	2764650	
Shortest contig	61	
Number of contigs > 1K nt	53	19.8%
Number of contigs > 10K nt	3	1.1%
Number of contigs > 100K nt	2	0.7%
Number of contigs > 1M nt	2	0.7%
Number of contigs > 10M nt	0	0.0%
Mean contig size	16209	
Median contig size	127	
N50 contig length	2764650	
L50 contig count	1	
contig %A	17.25	
contig %C	32.44	
contig %G	32.91	
contig %T	17.33	
contig %N	0.06	
contig %non-ACGTN	0.00	
Number of contig non-ACGTN nt	0	

What do these parameters mean?

- Contigs: number of contigs? Which is better, more or less contigs?
- N50: Length of the contig that contains the middle nucleotide when the contigs are ordered by size.
- Longest contig: length of the longest contig;
- Total size: Sum of all contigs lengths.

How many contigs are larger than 10 Kb? _____

Statistic	***K*-mer**		
	41	**49**	**55**
Contigs			
Total Size			
Longest Contig			
Mean Contig Size			
N50			

Repeat this exercise using a *k*-mer length of 49 and 55 (**don't forget to change the directory name: PT000033_41 to PT000033_49 or PT000033_55**).

Which is the best *k*-mer length? _____

Upon completion of these commands you should have obtained your contigs. Each of the three assemblies are located in the respective folders. Let's look at the assembly:

```
$ cd PT000033_91

# This will get you inside the PT000033_49 subdirectory, assuming
you are inside the Module2 directory. Recall that this is the
assembly carried out using a k-mer length of 31bp.

# Then:

$ more contigs.fa
```

In which format are the contigs' sequences? _____

Besides the contigs file you will find other files containing the sequences, graphs and other files necessary if you want to perform a faster re-assembly of your data using different parameters.

But now that you have the contigs we can order these and/or join them in **Exercise 3**.

Optional: Graph Visualization:

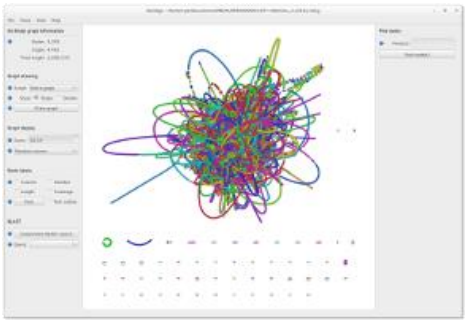
Under the assembly directories you can find the file LastGraph. It contains the final Graph of your assembly. It is possible to visualize and navigate this graph using the software Bandage. This is a program that can be installed on Windows, MacOS or Linux machines and it creates interactive visualizations of assembly graphs [20].

To open up Bandage double-click on its shortcut at the desktop or, from any terminal window just type:

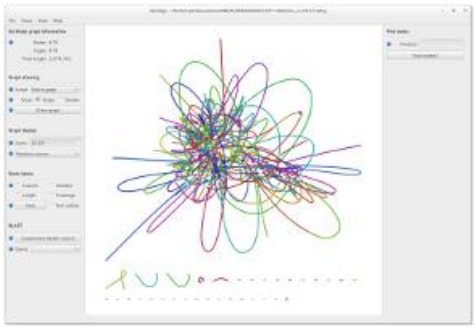
```
$ Bandage
```

This will start the Bandage window, go to *File* > *Load Graph* and select the LastGraph file from an assembly at your choice to open the Graph of your assembly. To visualize this Graph click on *Draw Graph*. Since this is a bacterial genome you can draw the entire graph. You can compare the different assemblies using this approach.

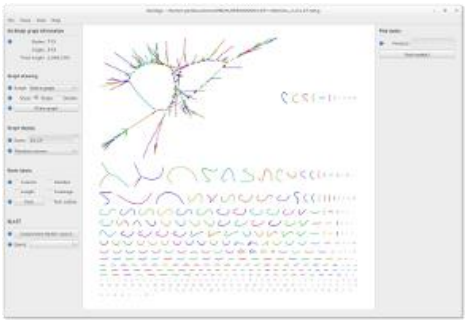
The graph for your assembly will show the different nodes and be similar to this:



K-mer of 21. This value is too small, resulting in short contigs and many connections, giving a dense tangled graph.



K-mer of 77. This is a good balance, giving long contigs that are well connected.



K-mer of 127. This value is too large, resulting in the graph breaking into many discontinuous pieces.

Additional notes:

The assembly exercise carried out in this exercise is a simplistic approach. In a real situation it is necessary to carefully estimate several parameters such as the expected coverage cut-off, expected coverage, k-mer length, insert size, etc. VelvetOptimiser script (https://github.com/tseemann/VelvetOptimiser) deals with these problems and carries out several Velvet assemblies while simultaneously adjusting for several of these parameters. You can input a range of K-mer lengths .

A different approach is implemented in SPAdes by using multi k-mer assemblies where you specify several *k*-mer lengths [18]. SPAdes takes as input paired-end reads, mate-pairs and single (unpaired) reads in FASTA and FASTQ. Besides SPAdes other assemblers implement a multi K-mer strategy (IDBA, Megahit, GATB-Pipeline) which always performs better than single K-mer strategies. The downside, and the main reason to why we do not used this strategy in this module, are the longer run times and the fact that these are more resource-demanding softwares.

Exercise 3 –Contig Ordering and Scaffolding

In this next exercise we will pick up from where we left in the previous exercise. Now that you have your contigs assembled from raw sequence reads we can order the contigs obtained using a suitable reference genome for this process (usually the closest high-quality finished genome). In this exercise we will use the reference genome used in Module 1 (M. tuberculosis H37Rv, GenBank Acc: NC000962.3). This reference-aided contig ordering becomes important as we progress towards discovery and comparative

genomics processes. Besides contig ordering, the process of joining the contigs is called scaffolding. Contigs within a scaffold are separated by N letters which represent gaps in the sequence. A scaffold is therefore chains of ordered contigs created using information about their relative position against a reference genome.

The software we will be using for this task at hand is called ABACAS [21]. This is a stand-alone program intended to rapidly contiguate (align, order, orientate), visualize and design primers to close gaps on shotgun assembled contigs based on a reference sequence. ABACAS is implemented in Perl and requires MUMmer and (optionally) BLAST installed on the local machine.

Let's order the contigs by simply typing the following:

```
$ cd PT000033_49

$ cp ../NC000962.3.fasta ./

$ abacas.pl -r ../NC000962_3.fasta -q contigs.fa -p promer -b
-d -a

# Abacas will align the contigs present in the contigs file
(contigs.fa) against the reference genome and will do so based on
the promer algorithm
```

You can choose to either do an alignment based on NUCmer (option: -p nucmer) or PROmer (option: -p promer) algorithms. NUCmer is a Perl script pipeline for the alignment of multiple *closely related* nucleotide sequences. PROmer (PROtein MUMmer) is a close relative to the NUCmer script. It follows the exact same steps as NUCmer and even uses most of the same programs in its pipeline, with one exception - all matching and alignment routines are performed on the six-frame amino acid translation of the DNA input sequence. This provides promer with a much higher sensitivity than nucmer because protein sequences tend to diverge much slower than their underlying DNA sequence. Therefore, on the same input sequences, promer may find many conserved regions that nucmer will not, simply because the DNA sequence is not as highly conserved as the amino acid translation.

By the end of this ordering and scaffolding process, ABACAS will show a screen like the one below:

```
*******************************************************************************************
* ABACAS: Algorithm Based Automatic Contiguation of Assembled Sequences                   *
*                                                                                         *
*                                                                                         *
*   Copyright (C) 2008-11 The Wellcome Trust Sanger Institute, Cambridge, UK.             *
*   All Rights Reserved.                                                                  *
*                                                                                         *
*******************************************************************************************

#  Checking user options:
#       -r Reference=../NC000962_3.fasta
#       -q Query=contigs.fa
#       -p promer
#       -d use default setting i.e. --mumreference in promer
#       -b  print multifasta file of contigs in bin to file
#       Input checking done!!
#       PREPARING DATA FOR   promer
#       Ordered contigs = 374
#       Bin contigs = 68
#       FINISHED CONTIG ORDERING
#
#       To view your results in ACT
#                Sequence file 1: NC000962_3.fasta
#                Comparison file 1: contigs.fa_NC000962_3.fasta.crunch
#                Sequence file 2: contigs.fa_NC000962_3.fasta.fasta
#
#               ACT feature file is: contigs.fa_NC000962_3.fasta.tab
#               Contigs bin file is: contigs.fa_NC000962_3.fasta.bin
#               Gaps in pseudomolecule are in: contigs.fa_NC000962_3.fasta.gaps
#-----------------------------------------------
```

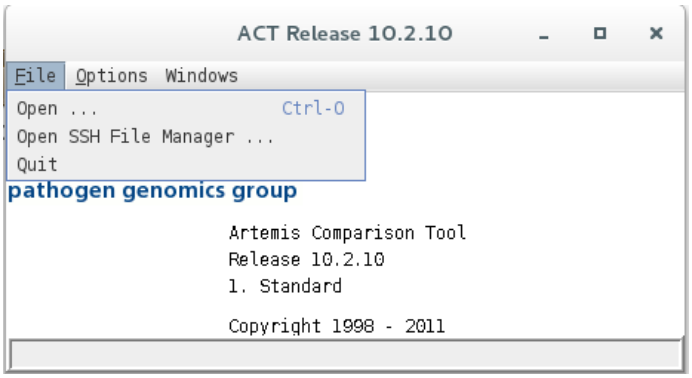
Now, what is ACT? ACT stands for Artemis Comparison Tool, a Java-written application for displaying pairwise comparisons between two or more DNA sequences (basically ACT has several Artemis windows) [22]. It can be used to identify and analyze regions of similarity and difference between genomes and to explore conservation of synteny, in the context of the entire sequences and their annotation. It can read complete EMBL, GENBANK and GFF entries or sequences in FASTA or raw format.

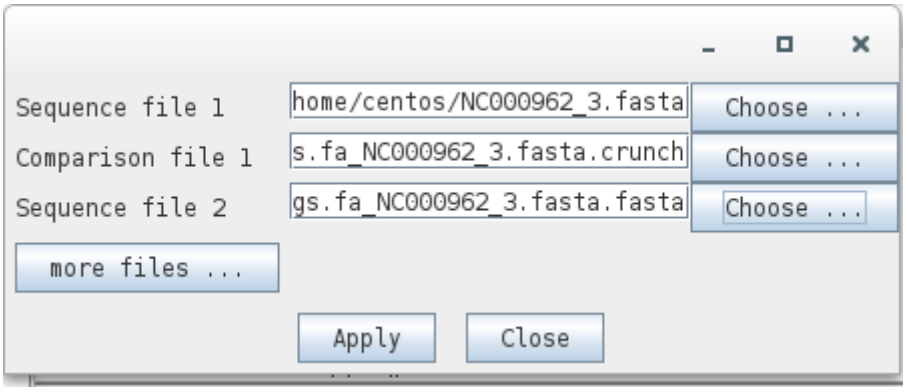
We can go ahead and have a look at the comparison that ABACAS already and present on the comparison file (*.crunch).

ACT can be started from any terminal window by typing:

```
$ act
```

ACT will start its main window, go to File > Open and select the files that ABACAS indicates from comparison:



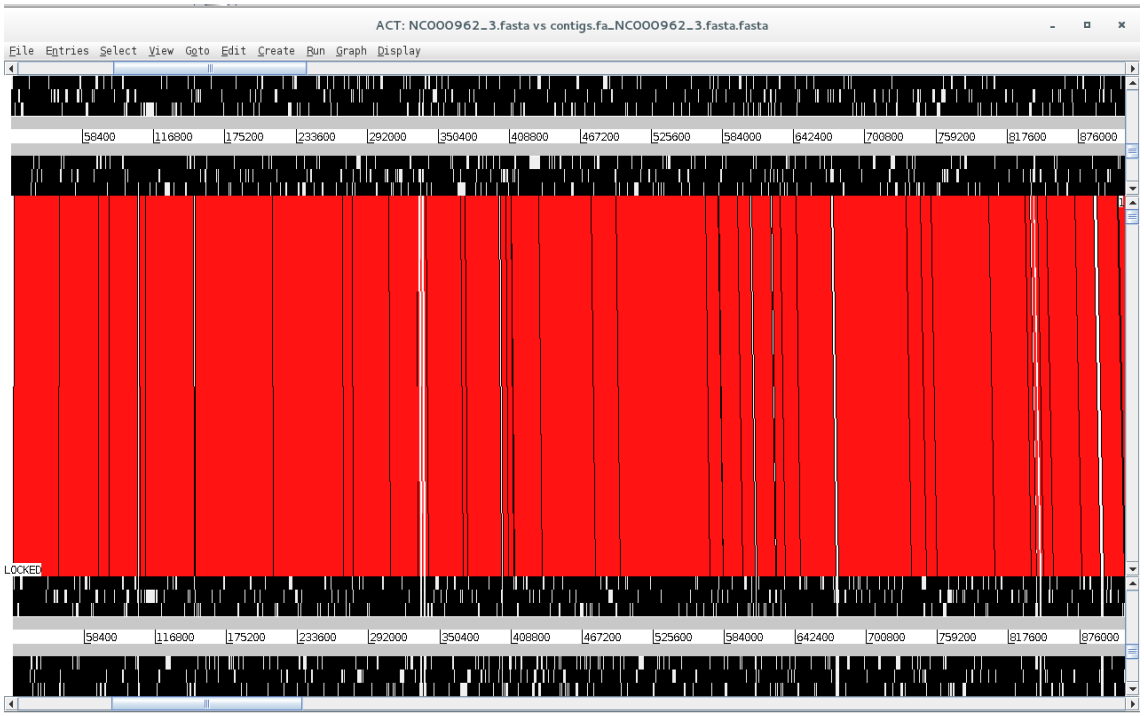


Sequence 1: NC000962_3.fasta

Comparison file 1: contigs.fa_NC00096_3.fasta.crunch

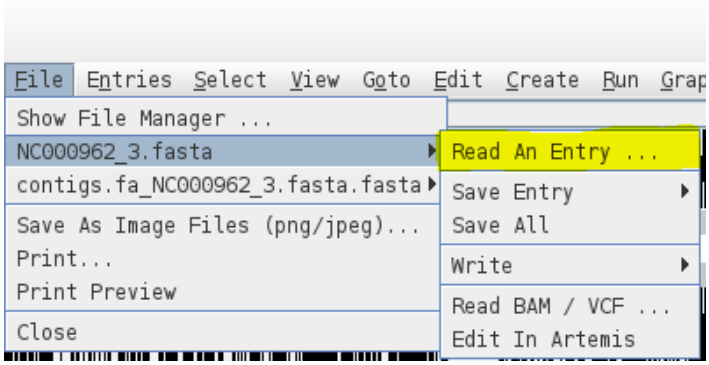
Sequence 2: contigs.fa_NC00096_3.fasta.fasta

Sequence 2 file now contains your contigs stitched to one another and separated by N's (scaffold) highlighting possible gaps in the assembly or regions not present in the reference genome. Look at the ACT comparison.

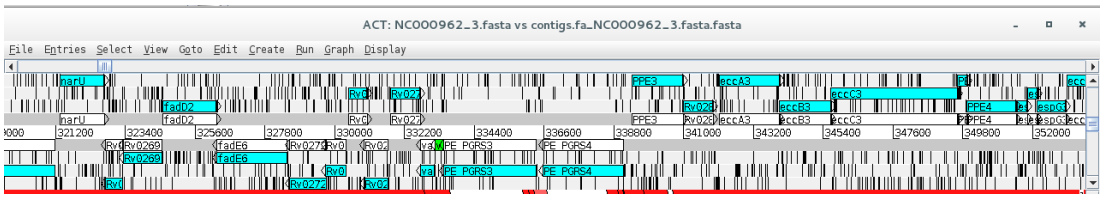


Is synteny conserved across these two genomes? What did you expect from M. tuberculosis?

Besides looking at the synteny of these large regions we can look at the conservation of gene synteny and distribution across this comparison. To do this we can read an entry onto of the sequences by going to *File > NC000962_3.fasta > Read an Entry…*



You can open the NC000962_3.gbk file used in the previous file. Remember this file is in the Genbank format, it contains the sequence along with genome annotation. Here, we are reading it onto the NC000962_3.fasta sequence and since the sequence is already present ACT will read the annotation only. You can see the annotation appearing on the top part of the comparison window.

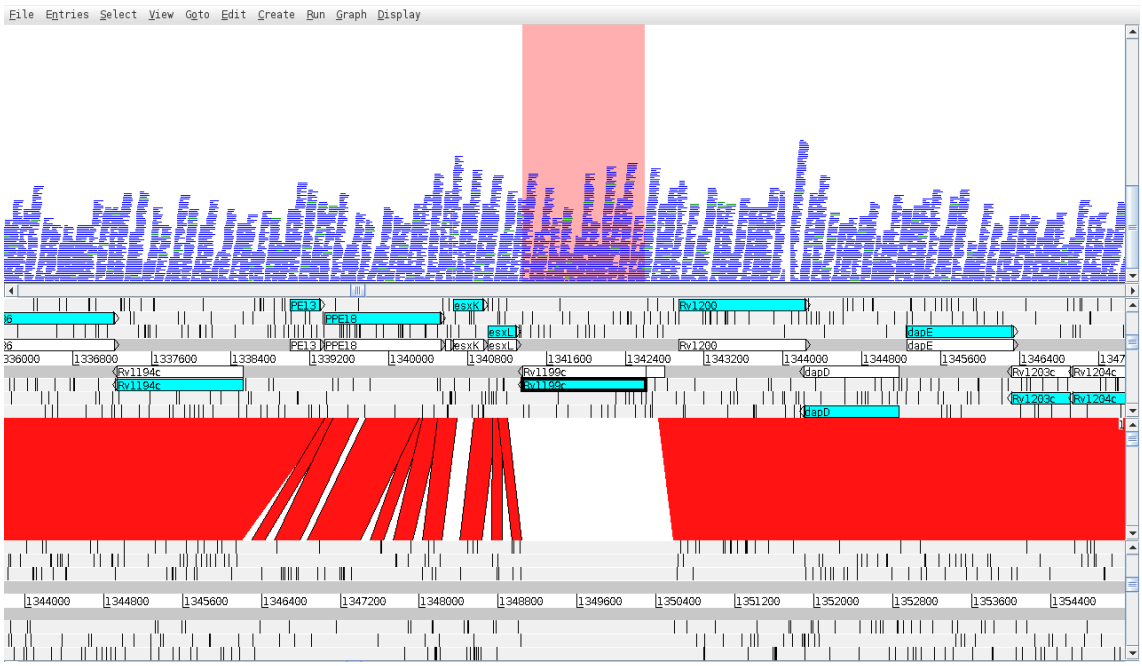


Take a look at the gaps in your assembly, where do the majority of these occur? Why is that?

We will not cover in this module how we can close these gaps but there are different strategies (see Additional notes below). We can also look at the mapped data from the previous module. Go to *File > NC000962_3.fasta > Read BAM/VCF…* and select the BAM file **PT000033.sorted.bam** in the **Module1** directory (remember that the correspondent .bai file must be in the same directory).

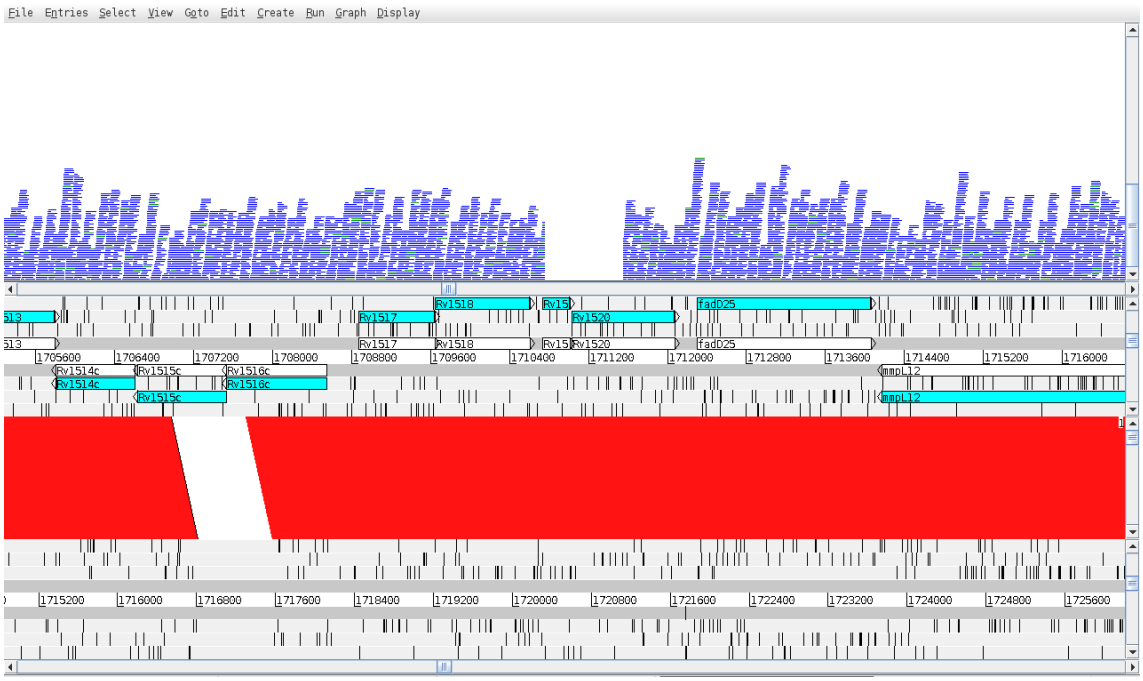
What do you see? Are most of these gapped regions real?_____

Look at the following example (around nucleotide 1341600) and locate it on your assembly:



What is happening here?_____

And here (nt 17111200):



Additional notes:

Contig ordering can also be carried out in Mauve, a genome aligner tool implemented with a Graphical User Interface (GUI), and therefore, more user friendly. The advantage of ABACAS over Mauve is that you can more easily implement contig ordering and scaffolding on a script that you run through the command-line.

For comparison purposes Mauve is also a great tool for multi-genome alignment (in particular using its Progressive Mauve algorithm). You can align your contigs against the contigs of other assembled strains, a reference strain and so on. Furthermore, during the alignment process, Mauve identifies conserved segments that appear to be internally free from genome rearrangements. Such regions are referred to as Locally Collinear Blocks (LCBs).

Also, ABACAS is part of a software bundle developed at the Wellcome Trust Sanger Institute called PAGIT. PAGIT is designed to finish high quality draft genome and, includes: ABACAS; IMAGE, an iterative approach for closing gaps in assembled genomes using mate pair information; iCORN, that enables errors in the consensus sequence to be corrected by iteratively mapping reads to the current assembly; and RATT, a tool to

transfer the annotation from a reference genome, or an earlier assembly, onto the latest assembly.

Exercise 4 - Genome Annotation

In the last exercise of this module we will attempt to find features along the scaffold created earlier and add some information concerning it to genome. This is a process called annotation and, it is necessary if we want to take full advantage of having the genome sequence and move towards functional genomics [15]. This annotation process is made with biologically relevant information that can range from gene models, functional data (including gene ontology or "Kyoto Encyclopedia of Genes and Genomes", KEGG, pathways) to epigenetic or microRNA modifications. Generally, this procedure is limited to the annotation of protein coding sequences and may include the annotation of ribosomal and transfer RNAs.

We can take three different approaches to annotation: web-based tools (RAST, NCBI annotation); command-line tools that perform *de novo* gene discovery (Prokka and DIYA); and, transfer of annotation data from a reference genome such as the one we used in the previous exercise (RATT, BG-7) [21, 23, 24].

Moreover, annotation will imply a different format other than FASTA. While the latter only stores the sequence, it is possible to have a second file in the GFF (General Feature File) format containing the annotation. Alternatively, we can combine these data on a GenBank file. You can look up online the structure of both formats.

Prokka Annotation:

In this exercise we will take the first two approaches and we will start with annotation with Prokka, a software tool to annotate bacterial, archaeal and viral genomes quickly and produce standards-compliant output files [23]. Prokka uses a variety of databases when trying to assign function to the predicted CDS features. It takes a hierarchial approach to make it fast. The initial core databases are derived from UniProtKB.

To start the annotation let's type:

```
$cd /home/centos/Module2/PT000033_49

$ prokka --outdir ./PT000033_prokka --prefix PT000033
contigs.fa_NC000962_3.fasta.fasta
```

Here, you have carried out a simple annotation procedure using Prokka's core database. With the --outdir option you specified the creation of a new output directory and the --prefix option allow you to set the prefix name of your files. You can look at other Prokka options by simply typing:

```
$ prokka
```

Meanwhile the annotation process will take about 16 minutes, but when finished you can find the new PT000033_prokka directory. Let's go inside:

```
$cd PT000033_prokka
```

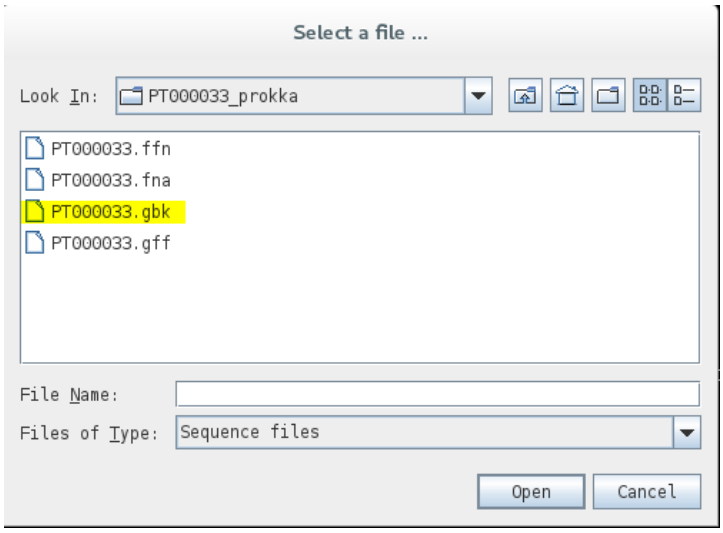
Prokka will have outputted the following files:

Extension	Description
.gff	This is the master annotation in GFF3 format, containing both sequences and annotations. It can be viewed directly in Artemis or IGV.
.gbk	This is a standard Genbank file derived from the master .gff. If the input to prokka was a multi-FASTA, then this will be a multi-Genbank, with one record for each sequence.
.fna	Nucleotide FASTA file of the input contig sequences.
.faa	Protein FASTA file of the translated CDS sequences.
.ffn	Nucleotide FASTA file of all the prediction transcripts (CDS, rRNA, tRNA, tmRNA, misc_RNA)
.sqn	An ASN1 format "Sequin" file for submission to Genbank. It needs to be edited to set the correct taxonomy, authors, related publication etc.
.fsa	Nucleotide FASTA file of the input contig sequences, used by "tbl2asn" to create the .sqn file. It is mostly the same as the .fna file, but with extra Sequin tags in the sequence description lines.
.tbl	Feature Table file, used by "tbl2asn" to create the .sqn file.
.err	Unacceptable annotations - the NCBI discrepancy report.
.log	Contains all the output that Prokka produced during its run. This is a record of what settings you used, even if the --quiet option was enabled.
.txt	Statistics relating to the annotated features found.
.tsv	Tab-separated file of all features: locus_tag,ftype,gene,EC_number,product

Next, open up Artemis:

```
$ art
```

And go to File > Open and open the PT000033.gbk file:



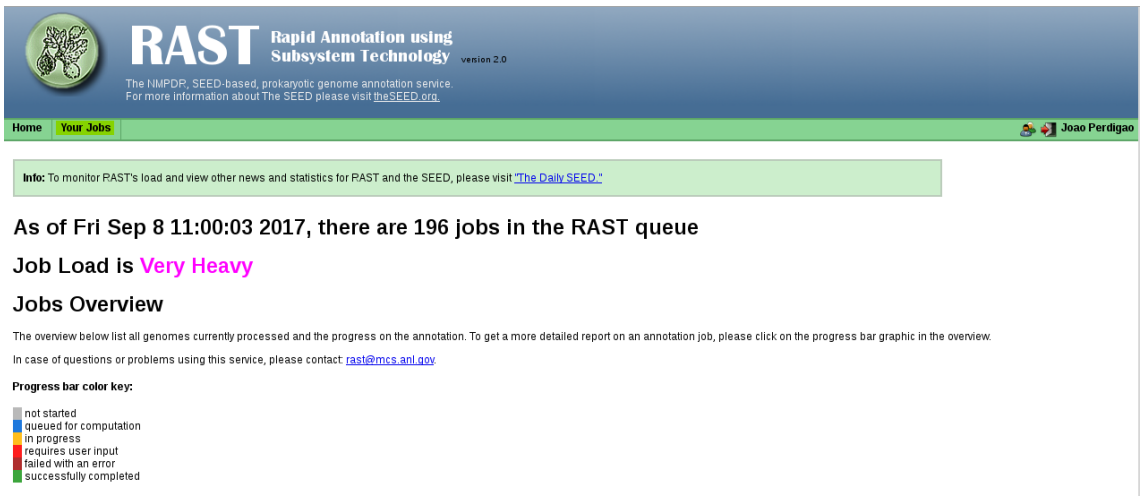
You can see that your scaffold from the previous exercise is now annotated:

As a side note, to improve Prokka annotations it is possible to use user-define databases from closely related genomes.

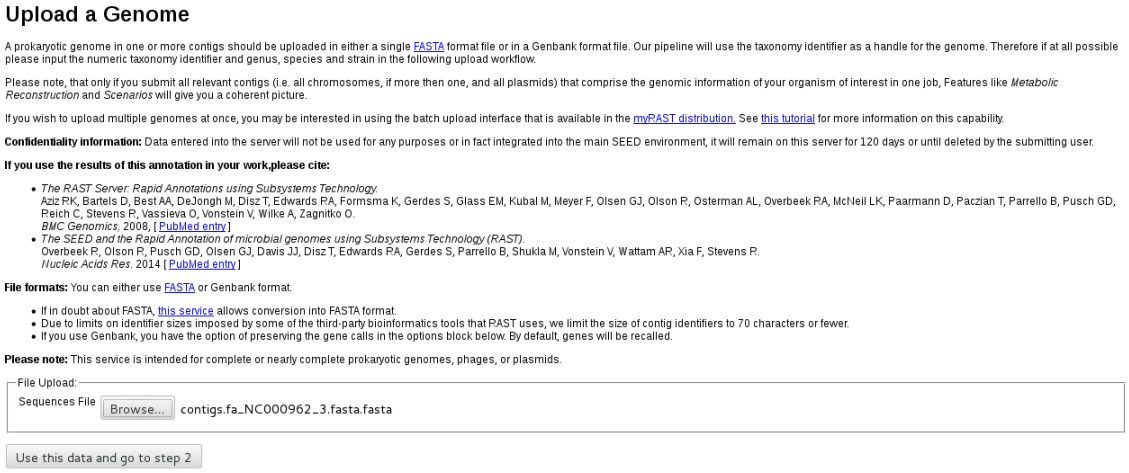
RAST Annotation:

RAST (Rapid Annotation using Subsystem Technology) is a fully-automated service for annotating complete or nearly complete bacterial and archaeal genomes available at http://rast.nmpdr.org/. RAST is designed to rapidly call and annotate the genes of a complete or essentially complete prokaryotic genome [24]. RAST, Rapid Annotations based on Subsystem Technology, uses a "Highest Confidence First" assignment propagation strategy based on a more sophisticated database based on manually curated subsystems and subsystem-based protein families that automatically guarantees a high degree of assignment consistency. RAST returns an analysis of the genes and subsystems in your genome, as supported by comparative and other forms of evidence. Despite being an on-line tool, it is possible to use RAST on the command-line through the myRAST toolkit, therefore allowing the integration of RAST in scripts and pipelines [24].

To use RAST you need to set up a free account.



Once you have registered to RAST and your account is active, log in to your RAST homepage and go to *Your Jobs* > *Upload New Job*.



Upload a Genome

A prokaryotic genome in one or more contigs should be uploaded in either a single FASTA format file or in a Genbank format file. Our pipeline will use the taxonomy identifier as a handle for the genome. Therefore if at all possible please input the numeric taxonomy identifier and genus, species and strain in the following upload workflow.

Please note, that only if you submit all relevant contigs (i.e. all chromosomes, if more then one, and all plasmids) that comprise the genomic information of your organism of interest in one job, Features like *Metabolic Reconstruction* and *Scenarios* will give you a coherent picture.

If you wish to upload multiple genomes at once, you may be interested in using the batch upload interface that is available in the myRAST distribution. See this tutorial for more information on this capability.

Confidentiality information: Data entered into the server will not be used for any purposes or in fact integrated into the main SEED environment, it will remain on this server for 120 days or until deleted by the submitting user.

If you use the results of this annotation in your work,please cite:

- *The RAST Server: Rapid Annotations using Subsystems Technology.*
  Aziz RK, Bartels D, Best AA, DeJongh M, Disz T, Edwards RA, Formsma K, Gerdes S, Glass EM, Kubal M, Meyer F, Olsen GJ, Olson R, Osterman AL, Overbeek RA, McNeil LK, Paarmann D, Paczian T, Parrello B, Pusch GD, Reich C, Stevens R, Vassieva O, Vonstein V, Wilke A, Zagnitko O.
  BMC Genomics, 2008, [PubMed entry]
- *The SEED and the Rapid Annotation of microbial genomes using Subsystems Technology (RAST).*
  Overbeek R, Olson R, Pusch GD, Olsen GJ, Davis JJ, Disz T, Edwards RA, Gerdes S, Parrello B, Shukla M, Vonstein V, Wattam AR, Xia F, Stevens R.
  Nucleic Acids Res. 2014 [PubMed entry]

File formats: You can either use FASTA or Genbank format.

- If in doubt about FASTA, this service allows conversion into FASTA format.
- Due to limits on identifier sizes imposed by some of the third-party bioinformatics tools that RAST uses, we limit the size of contig identifiers to 70 characters or fewer.
- If you use Genbank, you have the option of preserving the gene calls in the options block below. By default, genes will be recalled.

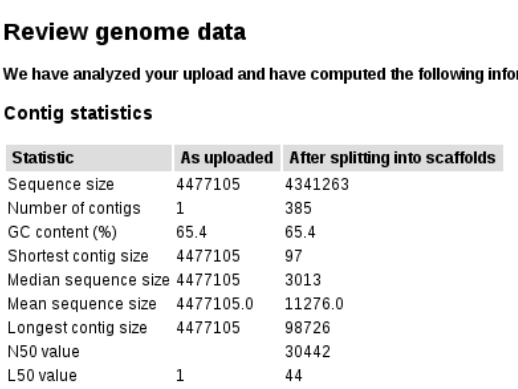
Please note: This service is intended for complete or nearly complete prokaryotic genomes, phages, or plasmids.

File Upload:

Sequences File Browse... contigs.fa_NC000962_3.fasta.fasta

Use this data and go to step 2

On this next screen select the scaffold file(contigs.fa_NC000962_3.fasta.fasta). On the next screen you can review some sequence statistics before proceeding:



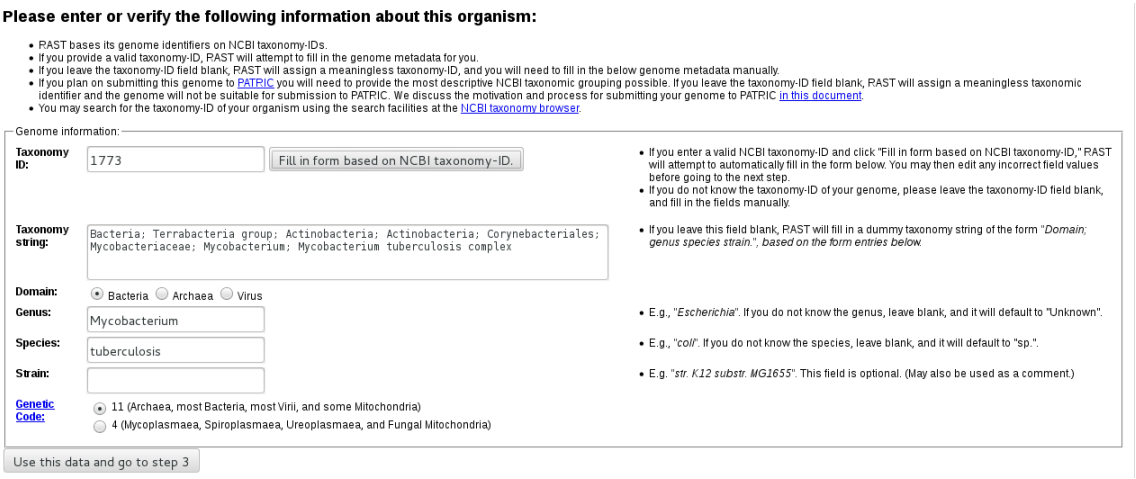
Review genome data

We have analyzed your upload and have computed the following information.

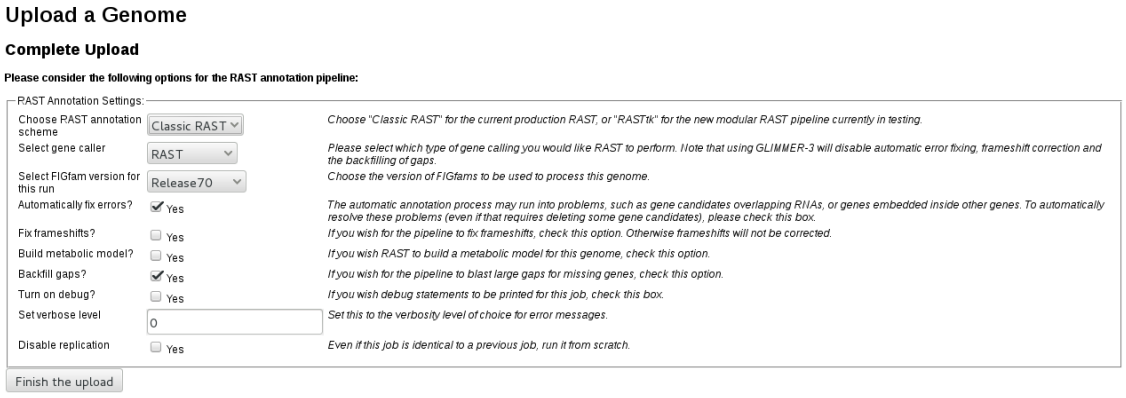
Contig statistics

Statistic	As uploaded	After splitting into scaffolds
Sequence size	4477105	4341263
Number of contigs	1	385
GC content (%)	65.4	65.4
Shortest contig size	4477105	97
Median sequence size	4477105	3013
Mean sequence size	4477105.0	11276.0
Longest contig size	4477105	98726
N50 value		30442
L50 value	1	44

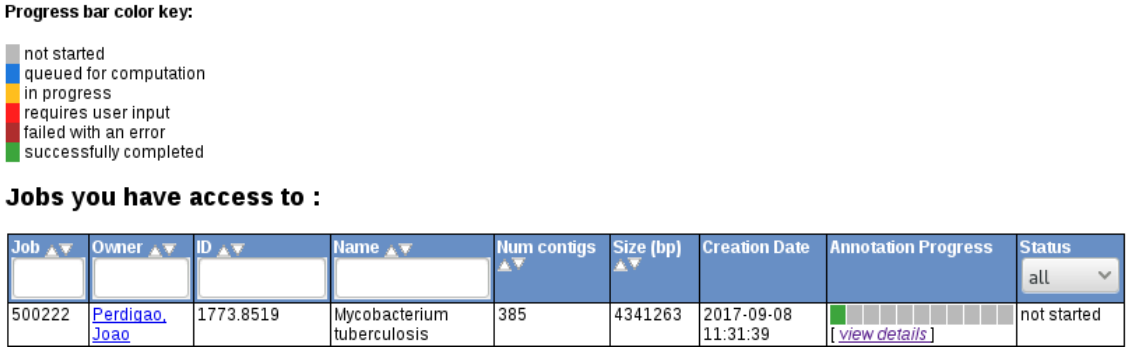
Below, in this same page you will be asked to introduce some information concerning the organism. Since we are working with Mycobacterium tuberculosis you can enter NCBI Taxonomy Id 1773 and click on the "Fill in form based on NCBI Taxonomy-ID" button as it will fill the form automatically:



In the last step you can chose some RAST annotation parameters. We will go with the default parameters:



Genome annotation using RAST may take half a day, an entire day or longer depending on server availability. Upon completion you can go to the Jobs Overview page where you can find your submitted jobs:



If your job is already completed you can click on *view details*, which will give you access to RAST output files. These files include, GenBank, GFF, FASTA, EMBL and even spradsheets

(Excel and tsv formats) containing the list of features, along with genome coordinates, orientation, gene ID, gene product, nucleotide and protein sequences, etc.

A RAST annotated PT000033 genome is available for you under the RAST_output sub-directory in Module2 directory. Open a terminal and type:

```
$ cd ./Module2/RAST_output
# Then start artemis:
$ art
```

Then open the GenBank file as you did with the Prokka annotation file. Is this annotation better?

Also, try to open the Excel spreadsheet with LibreOffice Calc in the Virtual Machine. LibreOffice Calc is an Open-Source alternative to Microsoft Excel. You can easily extract gene information from your strain using this file as well (and maybe even reading it into R!).

:: Introduction to Phylogenetics ::

Introduction

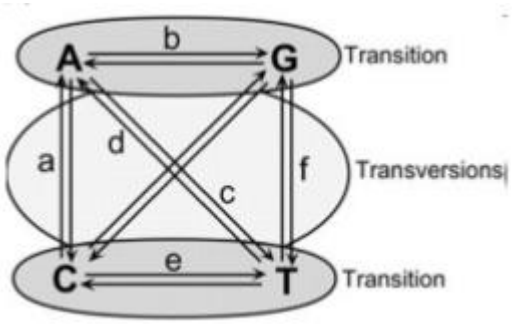
Phylogenetics pertains the **study of the evolutionary relationships** between organisms (species, strains, etc), genes or even genomes and, should represent the evolutionary history of such entities by showing how closely related they are [25, 26]. Usually this is depicted as a phylogenetic tree. Initially, phylogenetic reconstruction relied on morphological and physiological characters but the problem with to using such characters in phylogenetic analysis are: i) these are prone to convergent evolution which leads to homoplasies; and, iii) the number of characters to infer from are often limited. With advances in molecular biology, DNA and protein sequences rapidly became the preferred characters for phylogenetic analysis. From molecular-based phylogenies, **we can infer on how a given sequence has reached its current state** (*e.g.* how has it changed in the face of selective pressures) or **how is it expected to change in the future**. In the scope of this course, phylogenetic analysis can be an invaluable tool to identify the origin of pathogens, how they have spread and even **shed some light on cryptic transmission chains** that were otherwise undetected by conventional epidemiological investigation. In this regard, phylogenies are an increasingly recognized tool to **evaluate and extract biologically relevant information** from large molecular datasets [25-27].

To start a phylogenetic analysis, you generally **start from a multiple sequence alignment** constructed from a given dataset. The fundamental principle in a multiple sequence alignment is, that each column in the alignment should contain homologous residues, that is, with the same evolutionary origin and not the result of insertions or deletions in the sequence. As such, every position in an alignment can be considered an evolutionary marker and given the length of a simple gene it becomes immediately clear the increased robustness of molecular-based phylogenetics. It is therefore important to retain that a proper alignment contains an enormous amount of information.

Constructing a phylogenetic tree from a multiple sequence alignment can be done using methods that generally fall within two categories: **distance-matrix methods** (e.g. Unweighted Pair Group Method with Arithmetic Mean [UPGMA] or Neighbour-Joining) or **discrete data methods** (e.g. Maximum Likelihood or Bayesian methods). Distance based methods look at all pairwise comparison between sequences to calculate a simple distance metric, usually the percent of sequence difference to group isolates into a tree,

while discrete data methods look at each column of the alignment and calculate the best tree that represents the individual information at each of these sites [25, 27].

To compute these genetic distances and put it into a matrix. But to compute this matrix it is necessary to statistically model the substitution of nucleotides using a **nucleotide substitution model**. The simplest of these models is the Jukes and Cantor model introduced in 1969 (**JC69**) which assumes that the equilibrium frequencies for all four nucleotides are 25% and that any nucleotide has an equal probability of being replaced by any other nucleotide. On the other hand, the Kimura 1980 model (**K80**) distinguishes between transversions and transitions; and the General Time Reversible model (**GTR**) assumes a different rate for every possible substitution:



- JC69 : a=b=c=d=e=f
  A=C=G=T=1/4
- K80 : b=e, a=c=d=f
  A=C=G=T=1/4
- HKY85 : b=e, a=c=d=f
  A ≠ C ≠ G ≠ T
- TN93 : b, e, a=c=d=f
  A ≠ C ≠ G ≠ T
- GTR : a, b, c, d, e, f
  A ≠ C ≠ G ≠ T

There are other parameters that increase the complexity of these models of nucleotide substitution: unequal base frequencies (+F); varying proportions of invariant sites (+I); or variations in the substitution rate across sites (+G). A sequence may be evolving under any of these models, so it is important to determine which is the best fit to our data [26, 27].

But, how to evaluate the robustness of a tree and its branches? The most widely used test is called **bootstrapping**, where it randomly samples the entire alignment dataset column by column with replacement until achieving a sequence alignment with the length of the original one. This process is repeated several times (usually 1000 iterations) and the entire tree is recalculated for the pseudo-alignment every single time. By the end, all branches of the original tree are compared to check its presence in the trees obtained from the alignment sampling process. If a branch is present in 70% of these latter trees, it is generally assumed that this is a reliable grouping [25].

This module is composed by a set of 4 simple exercises to demonstrate how we can make a simple phylogenetic tree from WGS data obtained from *M. tuberculosis* clinical isolates. The main goal will be to assess the distances between these isolates using simple distance metrics as well as more robust models of molecular evolution and translate the latter into a phylogenetic tree that could be used to assess TB transmission dynamics and microevolution. The number of isolates to be used will be limited to cope with the limitations of the course computing system.

Exercise 1 – Core-genome SNP Alignment

Continuing from the previous modules, we are going to perform a simple phylogenetic analysis of five M. tuberculosis clinical isolates:

- PT000033
- PT000049
- PT000050
- PT000271
- PT000279

You can look up at these isolates characteristics such as spoligotyping lineage, shared-type or drug resistance profile in http://cplp-tb.ff.ulisboa.pt.

The idea here is to start by constructing a DNA pseudo-molecule from concatenated SNPs. Of course, this will vary according to the isolates that we are about to use, if we included another isolate with a divergent sub-set of SNPs it would result in a longer sequence. These SNPs will be obtained from comparison against a reference sequence using a mapping approach similar to Module 1. This reference sequence will be the same as in Module 1: *M. tuberculosis* H37Rv (GenBank accession: NC000962.3). Whith this in mind, alignement won't be necessary since we will be constructing this sequence from a SNP table. Such alignment can be called a core SNP alignment since each SNP occurs at a core site, that is, a site that is present at all sites.

In this exercise we will use Snippy, which finds SNPs between a haploid reference genome and your NGS sequence reads (https://github.com/tseemann/snippy). It will find both substitutions (SNPs) and insertions/deletions (indels). This script is designed with speed in mind, and produces a consistent set of output files in a single folder. It can then take a set of Snippy results using the same reference and generate a core SNP alignment (and ultimately a phylogenomic tree).

Let's get started, open a terminal window and type:

```
$ cd Module3

$ snippy --cpus 1 --outdir PT000033 --ref ../NC000962_3.gbk --R1
../course_files/PT000033_1.fastq.gz --R2
../course_files/PT000033_2.fastq.gz
```

The command above specifies: the number of CPU cores to be used, one in this case due to limitations in the computing system (usually with a quad-core computer you can specify 8 CPUs); an output directory containing the output files (using the –outdir option); the reference (--ref) genbank file of the reference and your reads.

If everything runs ok the output files will be in the PT000033 directory. To go there type:

```
$ cd PT000033
```

Check if the following output files are present:

Extension	Description
.tab	A simple tab-separated summary of all the variants
.csv	A comma-separated version of the .tab file
.html	A HTML version of the .tab file
.vcf	The final annotated variants in VCF format
.vcf.gz	Compressed .vcf file via BGZIP
.vcf.gz.tbi	Index for the .vcf.gz via TABIX
.bed	The variants in BED format
.gff	The variants in GFF3 format
.bam	The alignments in BAM format. Note that multi-mapping and unmapped reads are not present.
.bam.bai	Index for the .bam file
.raw.vcf	The unfiltered variant calls from Freebayes
.filt.vcf	The filtered variant calls from Freebayes
.log	A log file with the commands run and their outputs
.consensus.fa	A version of the reference genome with all variants instantiated
.aligned.fa	A version of the reference but with - at position with depth=0 and N for 0 < depth < --mincov (**does not have variants**)
.depth.gz	Output of samtools depth for the .bam file
.depth.gz.tbi	Index for the .depth.gz (*currently unused*)

For further information on file columns, fields or variant types you can check Snippy's documentation at: https://github.com/tseemann/snippy (Torsten Seemann)

Notice that Snippy already produces a Bam and annotated VCF file!

Let's go back to the Module3 directory:

```
$ cd ..

## Now let's look at what is in this directory:

$ ls
```

After listing its contents you may have realize that this directory already has the four other Snippy sub-directories for the remaining isolates. These have been computed earlier using commands similar to the one above.

For this next step it is necessary to ensure that all Snippy directories have been created using the same reference genome. This is the case. Now we will use the snippy-core command to produce the core SNP listing and sequences:

```
$ snippy-core PT000033 PT000049 PT000050 PT000271 PT000279
```

This command will read-in the output files from the previous snippy commands (and within the respective directories) and produce another set of output files containing data from all strains specified above. List the contents of the present directory and you should find the following output files:

Extension	Description
.aln	A core SNP alignment in the --aformat format (default FASTA)
.full.aln	A whole genome SNP alignment (includes invariant sites)
.tab	Tab-separated columnar list of core SNP sites with alleles and annotations
.txt	Tab-separated columnar list of alignment/core-size statistics

The file core.aln will contain the core SNP alignment in FASTA format. Let's look at it:

```
$ more core.aln
```

Does it seem ok?

At a first glimpse it might, but when dealing with genome-wide SNPs you might want to filter out some of these because some of these positions are associated with hard-to-map regions usually due to its repetitive nature. This is the case for PE/PPE genes which are usually not considered for *M. tuberculosis* phylogenetic analysis. As such, we will disregard this FASTA core SNP alignment file and we will build a new one from the core.tab file. This file contains the SNPs along with the respective position on the reference genome along with some data on gene product and gene id associated with the SNPs.

Give this file a look by typing:

```
$ more core.tab
```

Now, to use this file we will use an R-written script to concatenate the columns of this table but not before removing:

- Positions associated with PE/PPE genes;

- Positions with a unique K-mer length below 49/50 – gives an estimate of the mappability of the region by considering only positions that have a unique K-mer ending at that position of 49/50 bp in length (with the k-mer length set to 56bp).

This script requires some pre-computed data containing the k-mer scores at each position and the positions associated with PE/PPE genes (at ~/library_files/). To execute this script just type (assuming you are still within the ~/Module3/PT000033/ directory):

```
$ SNPtable_filter_Mtb.R core.tab
```

This command should have produced a file called *coreSNP_alignment_filtered.fas* from the SNP table. Let's look at it:

```
$ more coreSNP_alignment_filtered.fas
```

Similar to the core.aln file, right? The difference is that it does not contain a number of positions that may effectively influence your phylogenetic analysis.

By the end of this exercise you should have a file containing your aligned SNPs, ready for phylogenetic reconstruction.

Exercise 2 – Estimating simple distances: Hamming distance

For this quick exercise we will compare sequences by simply counting the number of difference between these. This distance metric is called the Hamming distance and if we express this as the proportion of the number of sites that are different we get the p-distance. Of course, these metrics do not account for multiple substitutions, substitution rate biases or differences in the evolutionary rates across sites.

To get the SNP distance separating our isolates we will use another R-written script from the command-line: HammingFasta.R. This script will output a distance matrix on the screen and will produce a csv file (you can try to open it with LibreOffice Calc) containing the matrix. To do this type:

```
$ HammingFasta.R coreSNP_alignment_filtered.fas
```

Did you get something like this on the screen?

What does this mean? What can be the public health implications from this comparison?

Which strains can be from patients with putative epidemiological links?

What can you learn from these strains in CPLP-TB database (http://cplp-tb.ff.ulisboa.pt)?

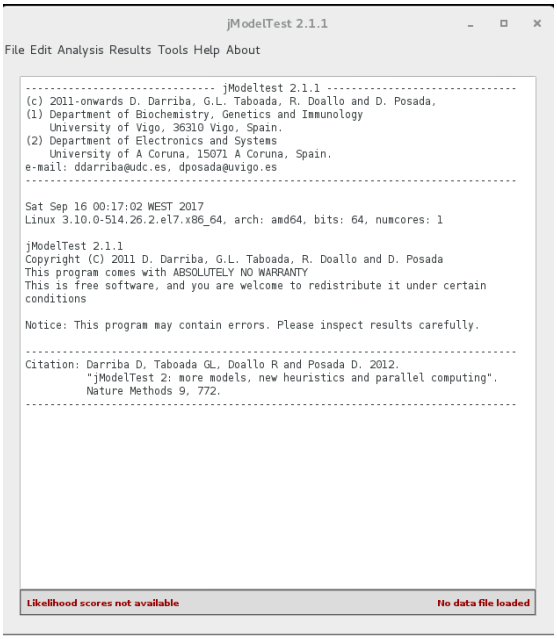
Exercise 3 – Phylogenetic Reconstruction

In the previous exercise you have already evaluate pairwise distance between isolates, but you did not obtain any phylogeny. Although it would be straightforward if we want to know in more detail the transmission dynamics and how these strains are evolving we need more robust methods to reconstruct the phylogeny of these strains.

In this exercise we will use Maximum Likelihood methods to do that. We will start with the alignment in the *coreSNP_alignment_filtered.fas* file. By the way, we are done with the command-line for the day. From now on it's just clicking!

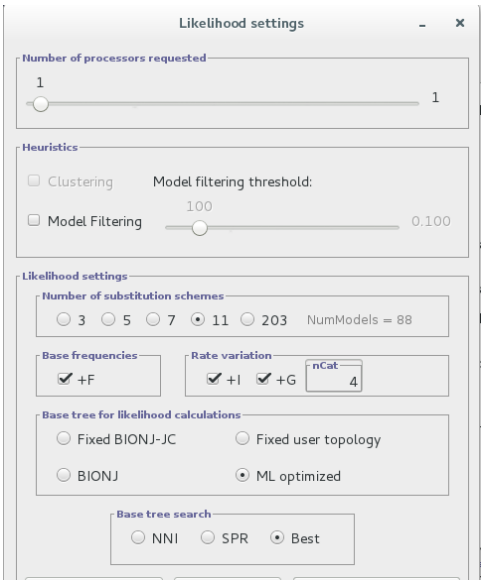
The first step will consist in evaluating what is the model of nucleotide substitution that best fits our data. To do that we will use a program called jModelTest2, which is written in JAVA [28]. To start it just double-click on its shortcut that you can find in the desktop.

Once it starts, you should see something like:



Now, go to *File > Load DNA alignment* and open your alignment file (*coreSNP_alignment_filtered.fas*). Once it opens it will tell you the number of sequences and the number of sites in your alignment.

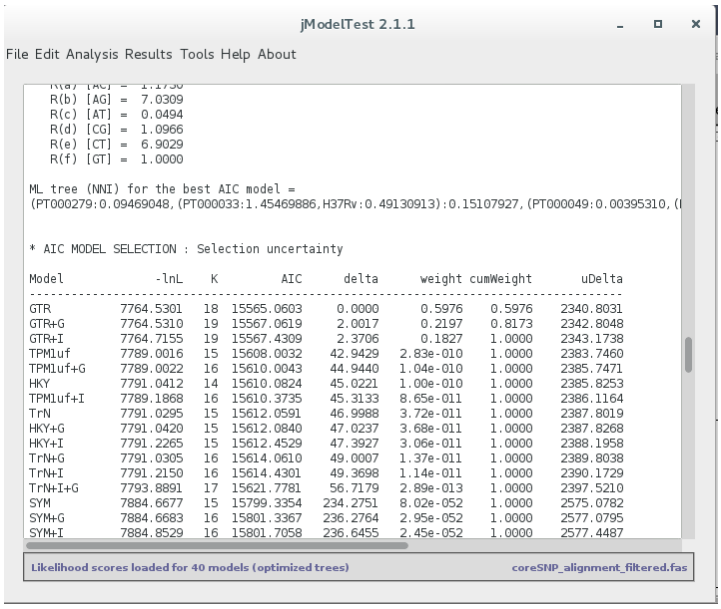
Next, go to Analysis > Compute likelihood scores:



Here you can choose the Likelihood settings. We will keep with the default settings except (to speed-up the analysis) we will only test 5 substitution schemes: Check that unequal base frequencies (+F), rate variation (+G) and invariant sites (+I) are selected. Choose a base tree for likelihood calculation that is ML optimized and for base tree search choose Best of NNI and SPR.

Once this analysis is over the software has calculated the likelihood score for each model. This likelihood score is usually a very low number and it is usually to be presented as the negative of its natural logarithm (-lnL). The lower this -lnL value the more likely the model is. Different model comparison algorithms are implemented in jModelTest. After the module evaluation you can go to Analysis menu and calculate the Akaike information Criterion (AIC), Bayesian information criterion (BIC) and Decision theory method. You can run all three tests with standard settings. The output will be on the screen, you have to scroll up to see the model list from each method, the best model comes first, notice its -lnL value and the weight column.

Here is an example for the AIC method:



According to the output of these programs, which model presents the best fit to our dataset? _____

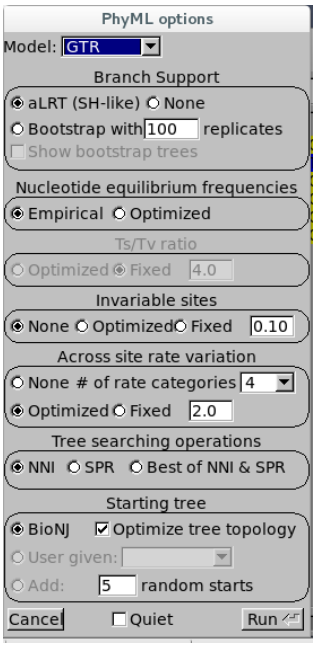
Do all the methods agree on it? _____

And the second best?

AIC is more liberal than AIC and penalizes free parameters less strongly, which may lead to preference for more complex models when simpler models may be a good fit as well.

What can we do if there is no agreement between decision methods?

Since all methods appear to agree on GTR, we will stick with it for the last part. We will reconstruct the phylogenetic tree of these isolates. For this purpose, we will use Seaview. Open Seaview using the shortcut on the desktop and go to *File* > *Open* and open the alignment file. The alignment immediately appears in the alignment window. To construct a Maximum Likelihood phylogenetic tree from this data, go to *Trees* > *PhyML*. A window should appear:



Here, we will set the parameters based on the calculations from jModelTest:

Branch Support: choose aLRT (approximate Likelihood Ratio Test) instead of bootstrapping. This is a more recent method based over likelihood gains from the present branch against a null (collapsing the branch) hypothesis. aLRT is, statistically speaking, less obscure than bootstrap and provides you a *p*-value [26, 29]. For example, the probability of a false positive on branch with aLRT score of 0.95 is 5% (0.05).

Nucleotide Equilibrium Frequencies: Choose empirical (it will calculate from sequence) as the model selected was not +F.

Invariable Sites: Choose None as the selected nodel was not +I (otherwise you would choose Optimized to calculate from data or Fixed and you would have to specify a value).

Across site rate variation: Choose None as the model was not +G.

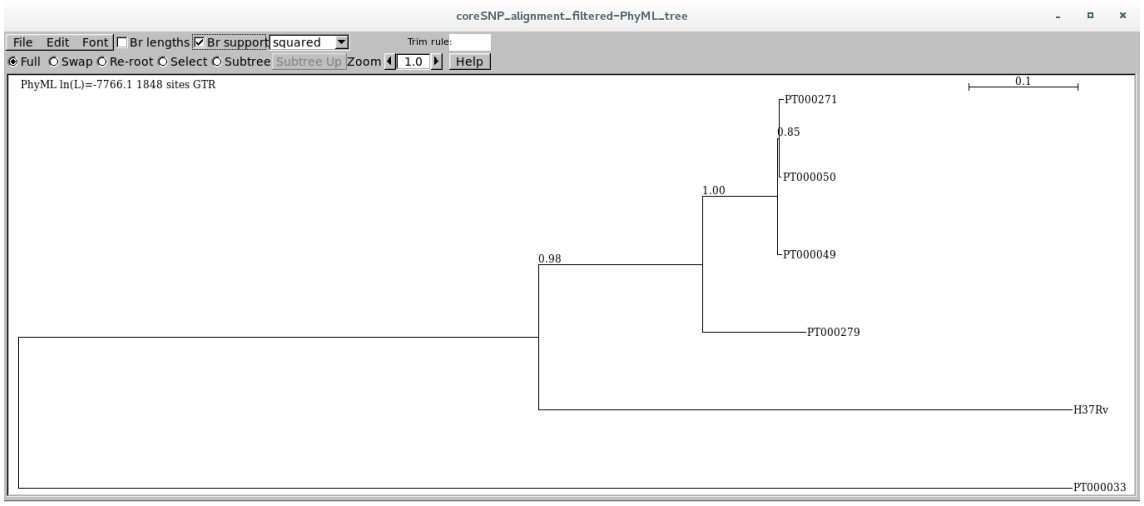
Tree searching operations: Choose Best of NNI & SPR.

Starting Tree: Choose BioNJ to calculate a starting tree and leave *Optimize Tree Selection* selected.

Then click *Run*.

As this alignment does not have many strains it will run rapidly. When the calculations end, click OK.

This will show you the tree. You can check *Br support* checkbox to visualize the aLRT scores:



You can try different visualization options other than squared.

What can you tell from this tree? How about the branches, do they seem statistically robust?

Tip: The tree can be saved in the Newick format (parenthetic). Just go to *File* > *Save as rooted/unrooted tree*.

REFERENCES

1 Loman NJ, Constantinidou C, Chan JZ, et al. High-throughput bacterial genome sequencing: an embarrassment of choice, a world of opportunity. *Nat Rev Microbiol*. 2012; **10**: 599-606.

2 Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics*. 2014; **30**: 2114-2120.

3 Olson ND, Lund SP, Colman RE, et al. Best practices for evaluating single nucleotide variant calling methods for microbial genomics. *Front Genet*. 2015; **6**: 235.

4 Li H, Homer N. A survey of sequence alignment algorithms for next-generation sequencing. *Brief Bioinform*. 2010; **11**: 473-483.

5 Mielczarek M, Szyda J. Review of alignment and SNP calling algorithms for next-generation sequencing data. *J Appl Genet*. 2016; **57**: 71-79.

6 Li H, Handsaker B, Wysoker A, et al. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*. 2009; **25**: 2078-2079.

7 Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*. 2009; **25**: 1754-1760.

8 Milne I, Bayer M, Stephen G, Cardle L, Marshall D. Tablet: Visualizing Next-Generation Sequence Assemblies and Mappings. *Methods Mol Biol*. 2016; **1374**: 253-268.

9 Carver T, Harris SR, Berriman M, Parkhill J, McQuillan JA. Artemis: an integrated platform for visualization and analysis of high-throughput sequence-based experimental data. *Bioinformatics*. 2012; **28**: 464-469.

10 Freese NH, Norris DC, Loraine AE. Integrated genome browser: visual analytics platform for genomics. *Bioinformatics*. 2016; **32**: 2089-2095.

11 McKenna A, Hanna M, Banks E, et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome research*. 2010; **20**: 1297-1303.

12 Danecek P, Auton A, Abecasis G, et al. The variant call format and VCFtools. *Bioinformatics*. 2011; **27**: 2156-2158.

13 Cingolani P, Platts A, Wang le L, et al. A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3. *Fly (Austin)*. 2012; **6**: 80-92.

14 Baker U, Tomson G, Some M, et al. 'How to know what you need to do': a cross-country comparison of maternal health guidelines in Burkina Faso, Ghana and Tanzania. *Implement Sci*. 2012; **7**: 31.

15 Ekblom R, Wolf JB. A field guide to whole-genome sequencing, assembly and annotation. *Evol Appl*. 2014; **7**: 1026-1042.

16 Chin FY, Leung HC, Yiu SM. Sequence assembly using next generation sequencing data--challenges and solutions. *Sci China Life Sci*. 2014; **57**: 1140-1148.

17 Compeau PE, Pevzner PA, Tesler G. How to apply de Bruijn graphs to genome assembly. *Nat Biotechnol*. 2011; **29**: 987-991.

18 Bankevich A, Nurk S, Antipov D, et al. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J Comput Biol*. 2012; **19**: 455-477.

19 Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome research*. 2008; **18**: 821-829.

20 Wick RR, Schultz MB, Zobel J, Holt KE. Bandage: interactive visualization of de novo genome assemblies. *Bioinformatics*. 2015; **31**: 3350-3352.

21 Swain MT, Tsai IJ, Assefa SA, Newbold C, Berriman M, Otto TD. A post-assembly genome-improvement toolkit (PAGIT) to obtain annotated genomes from contigs. *Nat Protoc*. 2012; **7**: 1260-1284.

22 Carver TJ, Rutherford KM, Berriman M, Rajandream MA, Barrell BG, Parkhill J. ACT: the Artemis Comparison Tool. *Bioinformatics*. 2005; **21**: 3422-3423.

23 Seemann T. Prokka: rapid prokaryotic genome annotation. *Bioinformatics*. 2014; **30**: 2068-2069.

24 Overbeek R, Olson R, Pusch GD, et al. The SEED and the Rapid Annotation of microbial genomes using Subsystems Technology (RAST). *Nucleic Acids Res*. 2014; **42**: D206-214.

25 Baldauf SL. Phylogeny for the faint of heart: a tutorial. *Trends Genet*. 2003; **19**: 345-351.

26 De Bruyn A, Martin DP, Lefeuvre P. Phylogenetic reconstruction methods: an overview. *Methods Mol Biol*. 2014; **1115**: 257-277.

27 Lemey P, Salemi M, Vandamme AM. *The Phylogenetic Handbook: A Practical Approach to Phylogenetic Analysis and Hypothesis Testing*. Cambridge: Cambridge University Press, 2009.

28 Darriba D, Taboada GL, Doallo R, Posada D. jModelTest 2: more models, new heuristics and parallel computing. *Nat Methods*. 2012; **9**: 772.

29 Anisimova M, Gascuel O. Approximate likelihood-ratio test for branches: A fast, accurate, and powerful alternative. *Syst Biol*. 2006; **55**: 539-552.