

# Module 4

## Mapping Short Reads

### Introduction

The **re-sequencing** of a genome typically aims to capture information on **Single Nucleotide Polymorphisms (SNPs)**, **INsertions and DEletions (INDELs)** and **Copy Number Variants (CNVs)** between representatives of the same species, usually in cases where a reference genome already exists (at least for a very closely related species). Whether one is dealing with different bacterial isolates, with different strains of single-celled parasites, or indeed with genomes of different human individuals, the principles are essentially the same. Instead of assembling the newly generated sequence reads *de novo* to produce a new genome sequence, it is easier and much faster to **align or map the new sequence data to the reference genome** (please note that we will use the terms “aligning” and “mapping” interchangeably). One can then readily identify SNPs, INDELs, and CNVs that distinguish closely related populations or individual organisms and may thus learn about genetic differences that may cause drug resistance or increased virulence in pathogens, or changed susceptibility to disease in humans. One important prerequisite for the mapping of sequence data to work is that the reference and the re-sequenced subject have the same genome architecture. Once you are familiar with viewing short read mapping data you may also find it helpful for quality checking your sequencing data and your *de novo* assemblies.

The computer programme **Artemis** allows the user to view and edit **genomic sequences** and EMBL/GenBank (NCBI) **annotation** entries in a highly interactive graphical format. Artemis also allows the user to view “**Next Generation Sequencing**” (NGS) data from Illumina, 454 or Solid machines. See <http://www.sanger.ac.uk/resources/software/artemis/ngs/>.

### Aims

- 1) To introduce the biology & workflow
- 2) To introduce mapping softwares, SAMtools, SAM/BAM and FASTQ file format
- 3) To show how **Next Generation Sequencing data** can be viewed in Artemis alongside your chosen reference using *Chlamydia* as an example: navigation, read filtering, read coverage, views,
- 4) To show how **sequence variation data** such as SNPs, INDELs, CNVs can be viewed in single and multiple BAM files, and BCF variant filtering
- 5) To show how short-read mapping can be executed with a script, and working with NGS data in eukaryotes: *Plasmodium*

# Background

## Biology

To learn about sequence read mapping and the use of Artemis in conjunction with NGS data we will work with real data from the bacterial pathogen *Chlamydia* as well as the eukaryotic single-celled parasites *Plasmodium* that cause malaria.

### *Chlamydia trachomatis*

*C. trachomatis* is one of the most prevalent human pathogens in the world, causing a variety of infections. It is the leading cause of **sexually transmitted infections (STIs)**, with an estimated 91 million new cases in 1999. Additionally, it is also the leading cause of preventable infectious blindness with some 84 million people thought to have active disease. The STI strains can be further subdivided into those that are restricted to the genital tract and the more invasive type known as the lymphogranuloma venereum or LGV biovar. Despite the large differences in the site of infection and the disease severity and outcome there are few whole-gene differences that distinguish any of the different types of *C. trachomatis*. As you will see most of the variation lies at the level of SNPs.

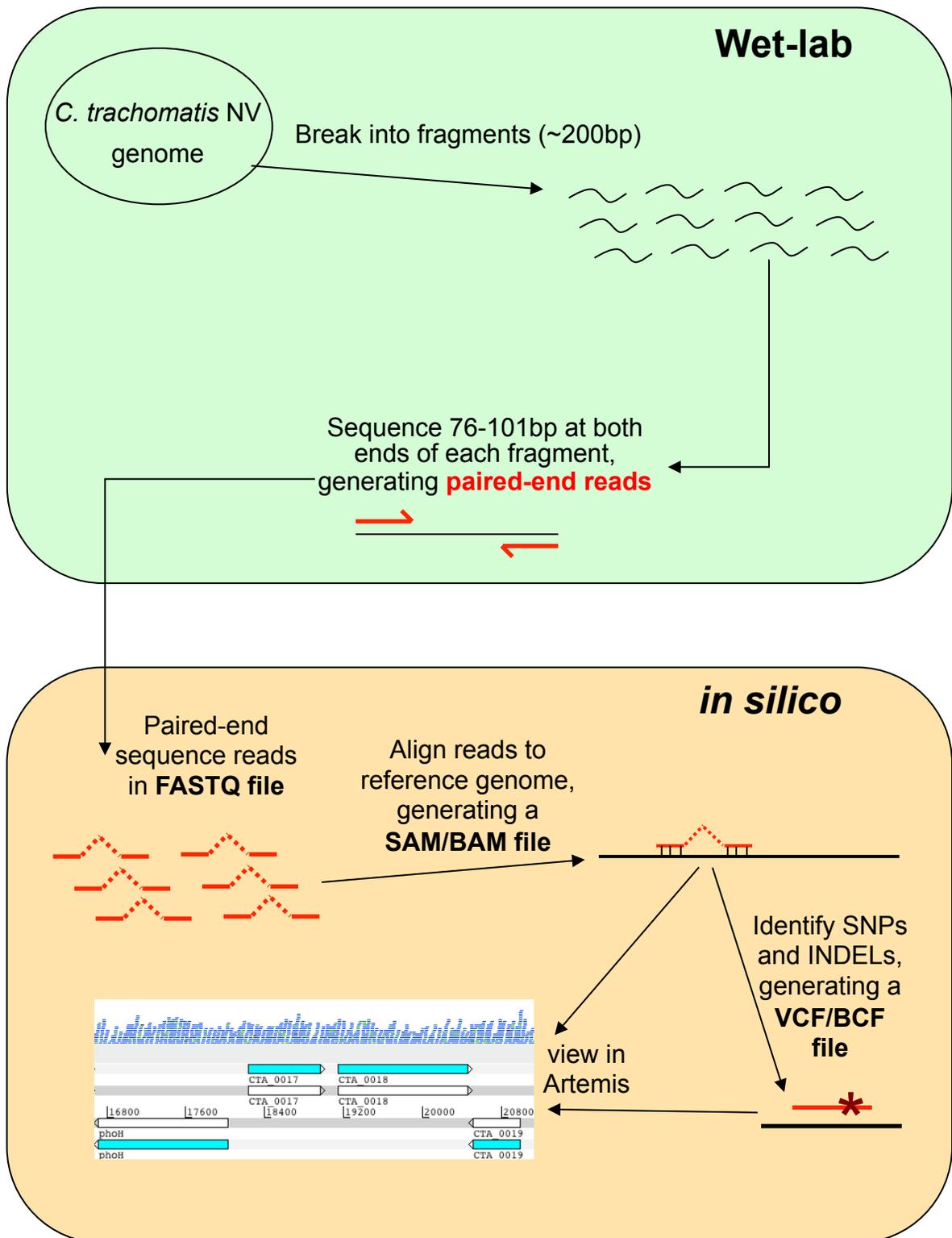
In this part of the course we will align against a reference sequence (L2) the Illumina reads from a recently isolated genital tract new variant Swedish STI *C. trachomatis* strain (known as NV) that caused a European health alert in 2006. During this time it became the dominant strain circulating in some European countries and began to spread world wide. The reason for this was that it **evaded detection by the widely used PCR-based diagnostic test**. During the course of this exercise you will identify the reason why this isolate confounded the standard assay.

### *Plasmodium falciparum*

*P. falciparum* is the causative agent of **the most dangerous form of malaria in humans**. The reference genome for *P. falciparum* strain 3D7 was determined and published about 10 years ago (Gardener et al., 2002). Since then the genomes of several other species of *Plasmodium* that infect humans or animals have been elucidated. Malaria is widespread in tropical and subtropical regions, including parts of Asia, Africa, and the Americas. Each year, there are approximately 350–500 million cases of malaria killing more than one million people, the majority of whom are young children in sub-Saharan Africa.

To date, the genomes of several strains of *P. falciparum* have been sequenced completely. For this exercise we will examine 76bp paired-end sequence read data from the malaria strains Dd2 and IT. In particular the *P. falciparum* Dd2 strain is well known for its **resistance to commonly used antimalarial drugs such as chloroquine**. Working with the mapped sequence data and Artemis we will have a closer look at some SNPs and CNVs that contribute directly to the drug-resistance phenotype of this deadly parasite.

## Workflow of re-sequencing, alignment, and *in silico* analysis



## Short-Read Alignment Software

There are multiple short-read alignment programs each with its own strengths, weaknesses, and caveats. Wikipedia has a good list and description of each. Search for “Short-Read Sequence Alignment” if you are interested. We are going to use BWA:

### BWA: Burrows-Wheeler Aligner

I quote from <http://bio-bwa.sourceforge.net/> the following:

*“Burrows-Wheeler Aligner (BWA) is an efficient program that aligns relatively short nucleotide sequences against a long reference sequence such as the human genome. It implements two algorithms, bwa-short and BWA-SW. The former works for query sequences shorter than 200bp and the latter for longer sequences up to around 100kbp. Both algorithms do gapped alignment. They are usually more accurate and faster on queries with low error rates.”*

Although BWA does not call Single Nucleotide Polymorphisms (SNPs) like some short-read alignment programs, e.g. MAQ, it is thought to be more accurate in what it does do and it outputs alignments in the SAM format which is supported by several generic SNP callers such as SAMtools and GATK.

BWA has a manual that has much more details on the commands we will use. This can be found here: <http://bio-bwa.sourceforge.net/bwa.shtml>

Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler Transform. *Bioinformatics*, 25:1754-60. [PMID: 19451168]

The first thing we are going to do in this Module is to align or map raw sequence read data that is in a standard short-read format (FASTQ) against a reference genome. This will allow us to determine the differences between our sequenced strain and the reference sequence without having to assemble our new sequence data *de novo*.

The FASTQ sequence format is shown over-page.



# 1. Exercise with data from *Chlamydia trachomatis*

To map the reads using BWA follow the following series of commands which you will type on the command line when you have opened up your terminal and navigated into the correct directory. Do a quick check to see if you are in the correct directory: when you type the UNIX command 'ls' you should see the following folders (in blue) and files (in white) in the resulting list.

```
wt@Pathogens: ~/Module_4_Mapping
wt@Pathogens:~/Module_4_Mapping$ ls
BCFs      L2b.bam.bai  L2_cat.fasta  NV_1.fastq
L2b.bam   L2_cat.embl  malaria       NV_2.fastq
```

## Stage 1:

Our **reference sequence** for this exercise is a *Chlamydia trachomatis* LGV strain called **L2**. The sequence file against which you will align your reads is called **L2\_cat.fasta**. This file contains a concatenated sequence in FASTA format consisting of the genome and a plasmid. To have a quick look at the first 10 lines of this file, type: `head L2_cat.fasta`

Most alignment programs need to index the reference sequence against which you will align your reads before you begin. To do this for BWA type:

```
bwa index L2_cat.fasta
```

The command and expected output are shown below. Be patient and wait for the command prompt to return before proceeding to Stage 2.

```
wt@Pathogens: ~/Module_4_Mapping
wt@Pathogens:~/Module_4_Mapping$ bwa index L2_cat.fasta
[bwa_index] Pack FASTA... 0.01 sec
[bwa_index] Construct BWT for the packed sequence...
[bwa_index] 0.29 seconds elapse.
[bwa_index] Update BWT... 0.01 sec
[bwa_index] Pack forward-only FASTA... 0.02 sec
[bwa_index] Construct SA from BWT and Occ... 0.20 sec
[main] Version: 0.7.5a-r405
[main] CMD: bwa index L2_cat.fasta
[main] Real time: 0.681 sec; CPU: 0.536 sec
wt@Pathogens:~/Module_4_Mapping$
```

**Stage 2:**

We will now align both the forward and the reverse reads separately against our now indexed reference sequence. The forward and reverse reads are contained in files NV\_1.fastq and NV\_2.fastq, and the output will be saved to files F.sai and R.sai, respectively. To have a quick look at the first 20 lines of one of the FASTQ files, type:

```
head -20 NV_1.fastq
```

Perform the alignment with the following two commands (wait until the first command is finished before starting the second command, it will take a little while):

```
bwa aln -q 15 L2_cat.fasta NV_1.fastq > F.sai
```

```
bwa aln -q 15 L2_cat.fasta NV_2.fastq > R.sai
```

```
wt@Pathogens: ~/Module_4_Mapping
wt@Pathogens:~/Module_4_Mapping$ bwa aln -q 15 L2_cat.fasta NV_1.fastq > F.sai
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_read_seq] 0.3% bases are trimmed.
[bwa_aln_core] calculate SA coordinate... 9.26 sec
[bwa_aln_core] write to the disk... 0.02 sec
[bwa_aln_core] 262144 sequences have been processed.
[bwa_read_seq] 0.2% bases are trimmed.

wt@Pathogens: ~/Module_4_Mapping
wt@Pathogens:~/Module_4_Mapping$ bwa aln -q 15 L2_cat.fasta NV_2.fastq > R.sai
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
```

Note the 'flag' `-q 15` tells the program to discard (parts of) sequence reads that are below a minimum quality score. Poor quality reads will therefore not be aligned.

**Stage 3:**

Now the separate alignments for the forward and reverse sequence reads need to be merged. Do this with the following command:

```
bwa sampe L2_cat.fasta F.sai R.sai NV_1.fastq NV_2.fastq > mapping.sam
```

The resulting file will be in SAM format.

```

wt@Pathogens: ~/Module_4_Mapping
wt@Pathogens:~/Module_4_Mapping$ bwa sampe L2_cat.fasta F.sai R.sai NV_1.fastq NV_2.fastq > mapping.sam
[bwa_read_seq] 0.3% bases are trimmed.
[bwa_read_seq] 0.2% bases are trimmed.
[bwa_sai2sam_pe_core] convert to sequence coordinate...
[infer_isize] (25, 50, 75) percentile: (135, 273, 329)
[infer_isize] low and high boundaries: 37 and 717 for estimating avg and std
[infer_isize] inferred external isize from 230491 pairs: 239.639 +/- 104.250
[infer_isize] skewness: -0.227; kurtosis: -1.309; ap_prior: 2.77e-05
[infer_isize] inferred maximum insert size: 830 (5.66 sigma)
[bwa_sai2sam_pe_core] time elapses: 3.96 sec
[bwa_sai2sam_pe_core] changing coordinates of 138 alignments.
[bwa_sai2sam_pe_core] align unmapped mate...
[bwa_paired_sw] 14791 out of 19324 Q17 singletons are mated.
[bwa_paired_sw] 6 out of 2874 Q17 discordant pairs are fixed.
[bwa_sai2sam_pe_core] time elapses: 0.65 sec
[bwa_sai2sam_pe_core] refine gapped alignments... 0.25 sec
[bwa_sai2sam_pe_core] print alignments... 1.30 sec
[bwa_sai2sam_pe_core] 262144 sequences have been processed.

```

**Please note:**

The `sampe` part of the command is for paired-end reads.

The last part of the command line `mapping.sam` determines the name of the output file that will be created in SAM format.

**SAM (Sequence Alignment/Map) format** is a generic format for storing large nucleotide sequence alignments that is illustrated on the next page. Creating our output in SAM format allows us to use a complementary software package called SAMtools.

**SAMtools** is a collection of utilities for manipulating alignments in SAM format.

See <http://samtools.sourceforge.net/> for more information. There are numerous options that control the way the SAMtools utilities run, a few of which are explained below. To get brief explanations of the various utilities and the different options or flags that control each utility, type `samtools` or `samtools` followed by one particular utility on the command line like e.g.:

```

samtools
samtools view

```

To have a quick look at the first lines of the SAM file you just generated, type:

```
head mapping.sam
```

The SAM/BAM file format is illustrated on the next page.

## File format: SAM / BAM (each line: one aligned sequence read)

The SAM/BAM file format is very powerful. It is unlikely that you will need to work with the contents of a SAM/BAM file directly, but it is very informative to visualize it in a viewer and it is a great format to do further analysis with. The format specifications are at <http://samtools.sourceforge.net/SAM1.pdf>. Below is a brief overview of the information contained in such files.

QNAME	FLAG	RNAME	POS	MAPQ	CIGAR	RNEXT	PNEXT	TLEN	SEQ	QUAL
IL23_4880:2:1:13282:2618#11	163	AM884176.1	1	60	108M	=	148	255	ATGACAAGGCCTCCATTACTAAAACGACCTCGCAGAAACCGAAAAAGTGCAGCCGTCGATCTATAATTCAGAAACCCAACCTCTGTCTAGTACTGACTTGACTCTGGCCC	
IL23_4880:2:13:17651:14038#11	99	AM884176.1	2	60	108M	=	97	203	TGACAAGGCCTCCATTACTAAAACGACCTCGCAGAAACCGAAAAAGTGCAGCCGTCGATCTATAATTCAGAAACCCAACCTCTGTCTAGTACTGACTTGACTCTGGCCC	
IL23_4880:2:1:13282:2618#11	83	AM884176.1	148	60	108M	=	1	-255	GAGAGTATGCCTGGAGTATACAGATGGAGTTTAGACATGGTCTCTAAAGAGTTAGAGACTTTGTACGATAGGATGAAAGCAGTTATCTCTTTCCTGTAATTGAT	
IL23_4880:2:13:17651:14038#11	147	AM884176.1	97	60	108M	=	2	-203	TTGATCTGGCCCATCTTTCTTAAAGATGGCTCTCGAATTCGAGAAGAAATAGAGACTATGCCTGGAGTATACAGATGGAGTTTAGACATGGTCTCTAAAGAGTTAGAG	

Next we want to change the file format from SAM to BAM. While files in SAM format store their information as plain text, the BAM format is a binary representation of that same information. One reason to keep the alignment files in BAM rather than in SAM format is that the binary files are a lot smaller than the plain text files, i.e. the BAM format saves expensive storage space (sequence data are generated at an ever increasing rate!) and reduces the time the computer has to wait for slow disk access to read or write data.

Many visualization tools can read BAM files. But first a BAM file has to be sorted (by chromosome/reference sequence and position) and indexed, which enables fast working with the alignments.

**Stage 4:**

To convert our SAM format alignment into BAM format run the following command:

```
samtools view -b -S mapping.sam > mapping.bam
```



Flag: output in BAM format    Flag: input in SAM format – **note: this is a capital S**

```
wt@Pathogens: ~/Module_4_Mapping
wt@Pathogens:~/Module_4_Mapping$ samtools view -b -S mapping.sam > mapping.bam
```

**Stage 5:**

Next we need to sort the mapped read sequences in the BAM file by typing this command:

```
samtools sort mapping.bam NV
```

↙ Prefix for output file

This will take a little time to run.

By default the sorting is done by chromosomal/reference sequence and position.

```
wt@Pathogens: ~/Module_4_Mapping
wt@Pathogens:~/Module_4_Mapping$ samtools sort mapping.bam NV
```

**Stage 6:**

Finally we need to index the BAM file to make it ready for viewing in Artemis:

```
samtools index NV.bam
```

```
wt@Pathogens: ~/Module_4_Mapping
wt@Pathogens:~/Module_4_Mapping$ samtools index NV.bam
```

**Stage 7:**

We are now ready to open up Artemis and view our newly mapped sequence data.

## 1. Start up Artemis.

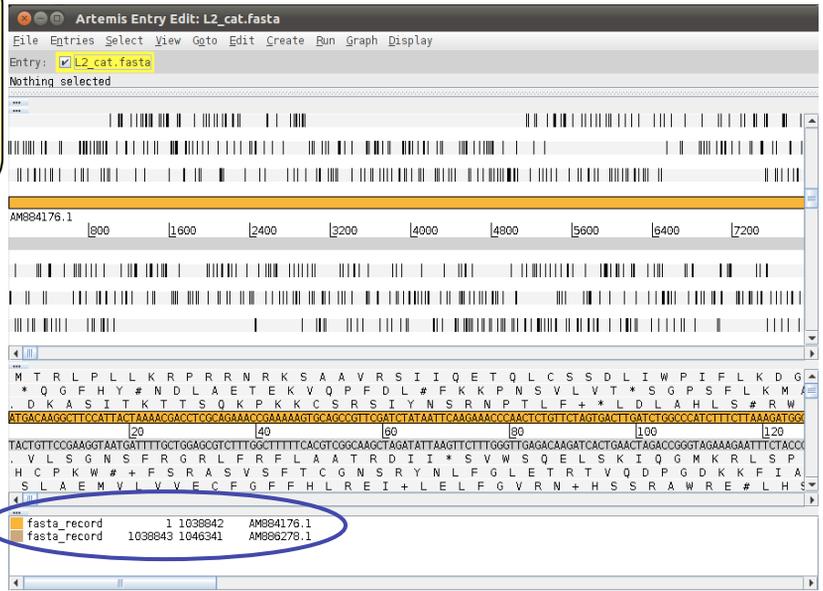
Double click on the Artemis Icon or type 'art &' on the command line of your terminal window and press return. We will read the reference sequence into Artemis that we have been using as a reference up until now.

Once you see the initial Artemis window, open the file `L2_cat.fasta` via **File – Open**. Just to remind you, this file contains a concatenated sequence consisting of the *C. trachomatis* LGV strain 'L2' chromosome sequence along with its plasmid.

Hopefully you will now have an Artemis window like this!

If not, please ask a demonstrator for assistance.

Since the `L2_cat.fasta` is a concatenation of two DNA sequences (chromosome and plasmid) it draws two features automatically to represent them, one in orange and the other brown.



## 2. Now load up the annotation file for the *C. trachomatis* LGV strain L2 chromosome.

1

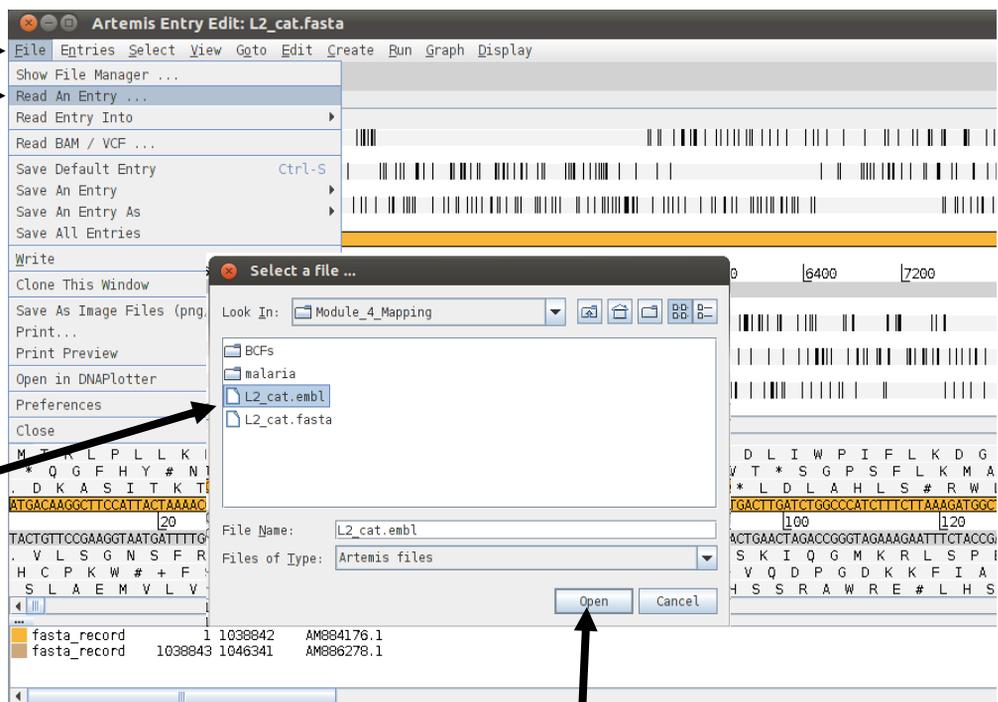
Click 'File' then 'Read An Entry'

2

Single click to select EMBL file

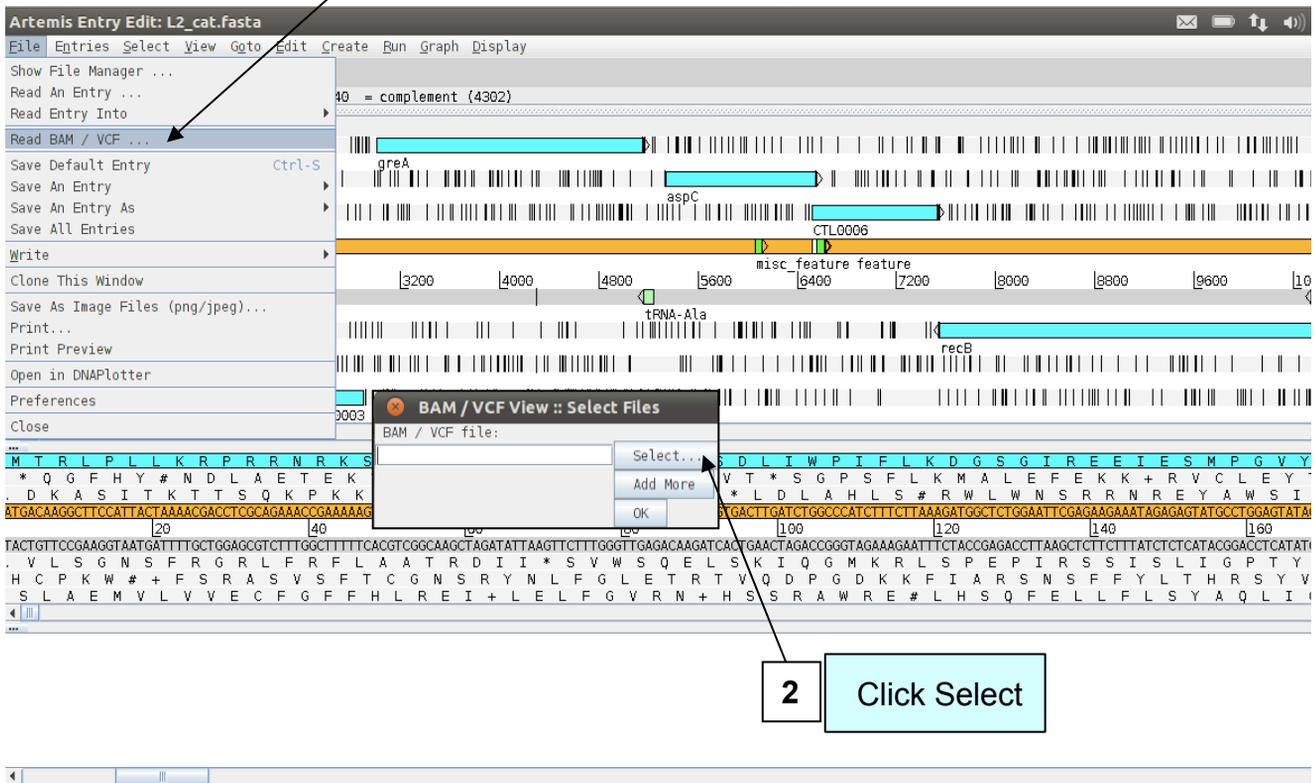
3

Single click to open file in Artemis then wait

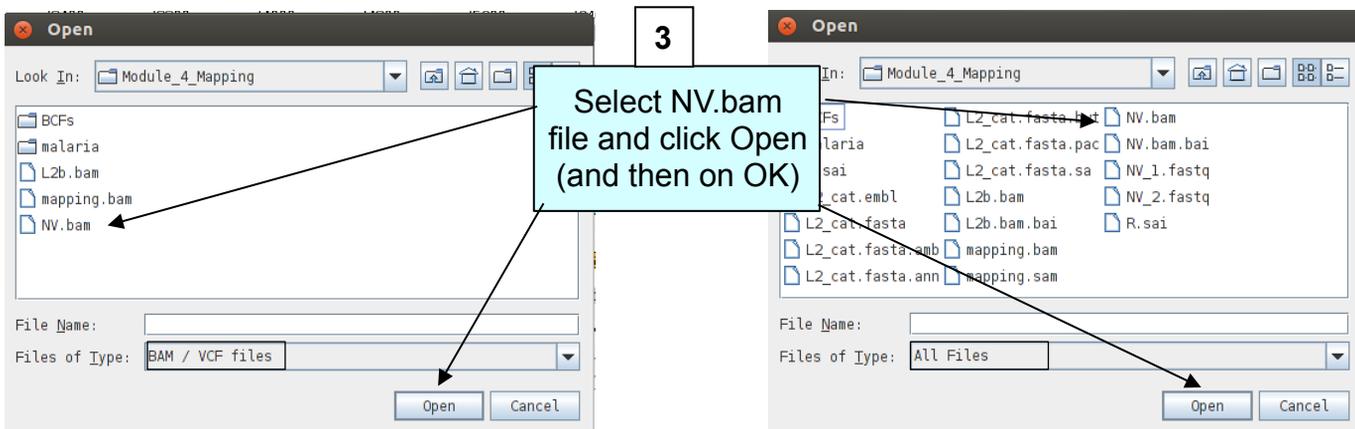


To examine the read mapping we have just performed we are going to read our BAM file containing the mapped reads into Artemis as described below. Please make sure you do not go to a zoomed-out view of Artemis, but stay at this level, as display of BAM files does take time to load!

1 Read in a BAM file



2 Click Select



3 Select NV.bam file and click Open (and then on OK)

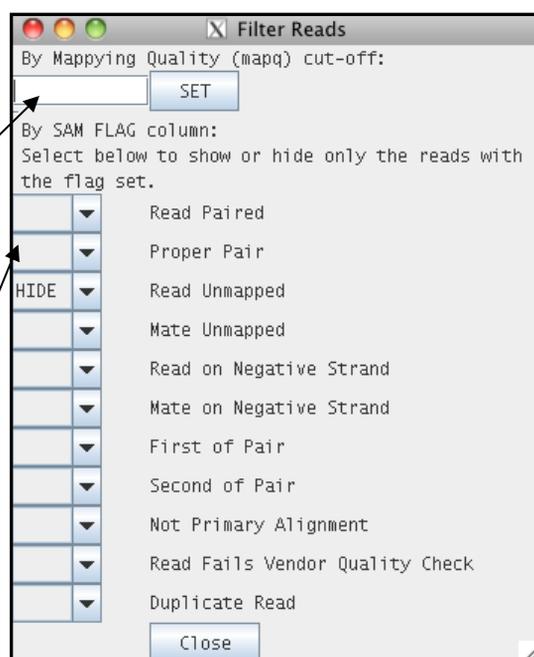
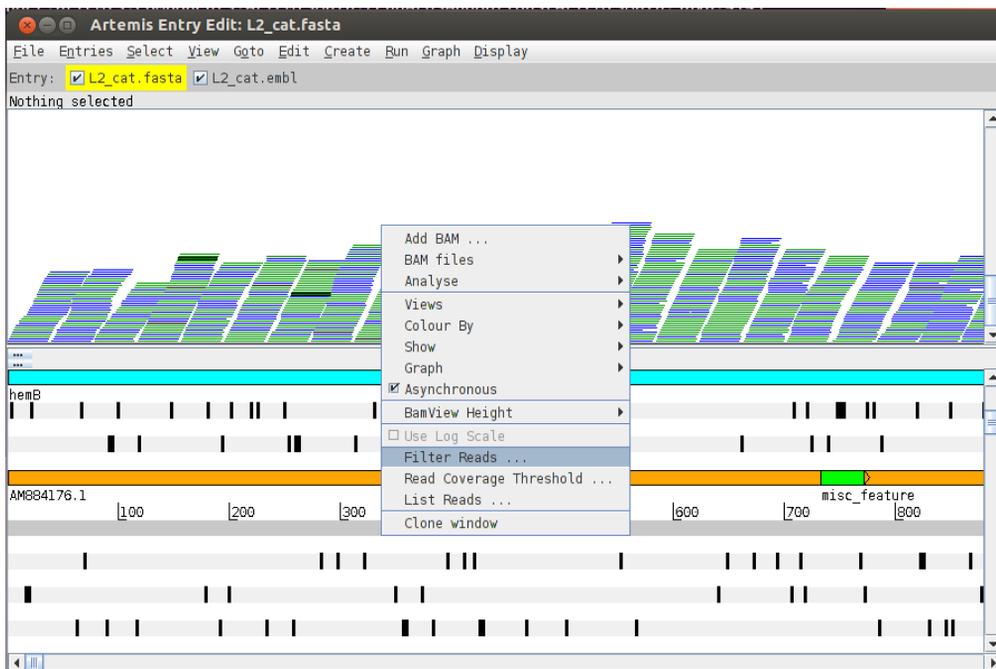
You should see the BAM window appear as in the screen shot below. Remember these reads are of the Swedish NV strain mapped against the LGV strain L2 reference genome. In the top panel of the window each little horizontal line represents a sequencing read. Notice that some reads are blue which indicates that these are unique reads, whereas green reads represent “duplicated” reads that have been mapped to exactly the same position on the reference sequence. To save space, if there are duplicated reads only one is shown, which means that there could be a large number of duplicated reads at a given position but the software only depicts one.



If you click a read (1 & 2) its mate pair will also be selected. Also note that if the cursor hovers over a read for long enough details of that read will appear in a small box (3). If you want to know more then right-click and select ‘Show details of: READ NAME’ from the menu (4). A window will appear (5) detailing the mapping quality (see over page), coordinates, whether it’s a duplicated read etc. If this read(s) covers a region of interest, being able to access this information easily can be really helpful.

**“Mapping quality”** - The mapping quality depends on the number of mismatches between the read and the reference sequence as well as the repetitiveness of the reference sequence. The maximum quality value is 99, whereas a value of 0 means that the read mapped equally well to at least one other location and is therefore not reliably mapped.

You can actually use several details relating to the mapping of a read to filter the reads from the BAM file that are shown in the window. To do this, right-click again over the stack plot window showing the reads and select “Filter Reads...”. A window will appear with many options for filtering, as shown below.



Reads with less than the mapping quality are not shown. Try 60.

HIDE the proper pairs. What happened?

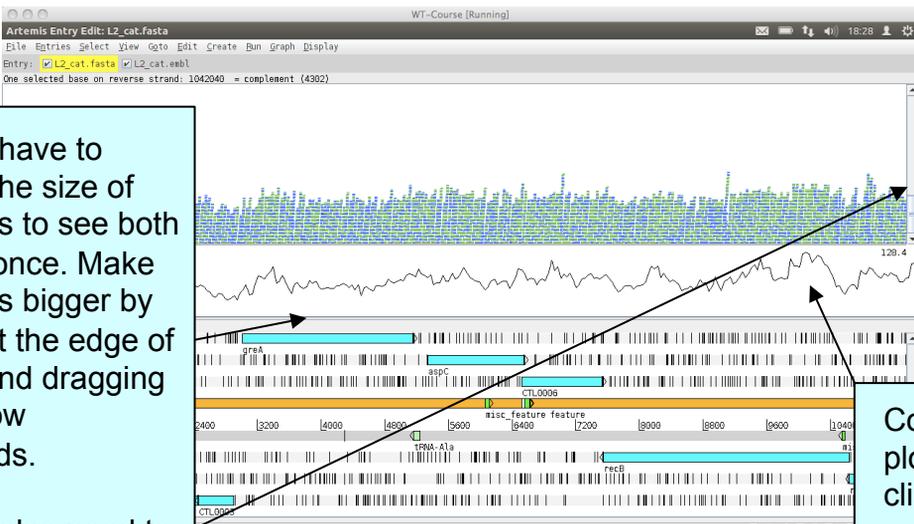
Filtering reads for repetitive regions or seeing properly-paired-reads only can be really helpful.

As mentioned before, to save space if there are duplicated reads only one is represented. But often one may want to know the actual read coverage on a particular region or see a graph of this coverage. You can do this by adding additional graphs as detailed below.



1

There are different views and graphs to display that you can choose from, for example: right click here and select 'Graph' then 'Coverage' from the menu.  
See below.



2

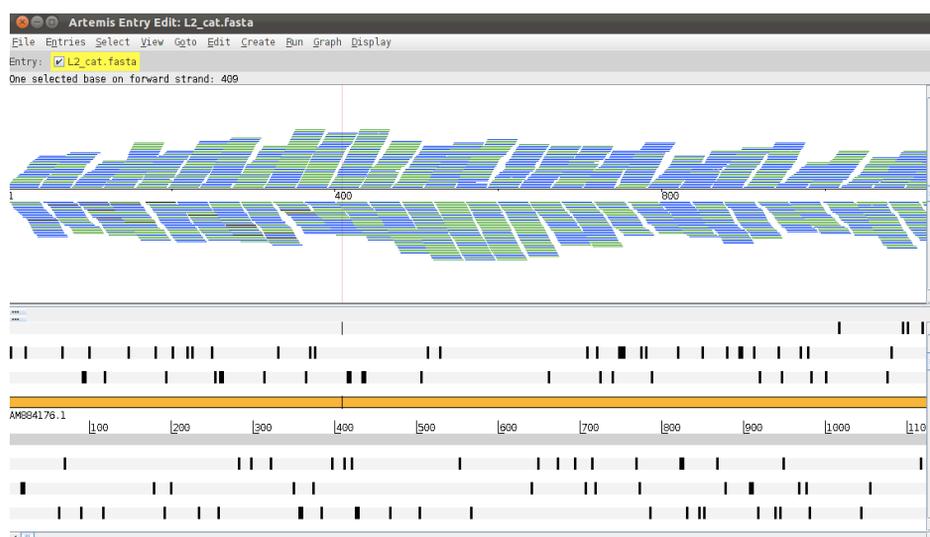
You may have to readjust the size of the panels to see both views at once. Make the panels bigger by clicking at the edge of a panel and dragging the window downwards.  
You may also need to use this slider to adjust the Stack View too.

Coverage plot. Right click for more options.

There are several other ways to view your aligned read information. Each one may only be subtly different but they are very useful for specific tasks as hopefully you will see. To explore the alternative read views right-click in the BAM panel (1 below) and select the 'Views' menu option (2 below):

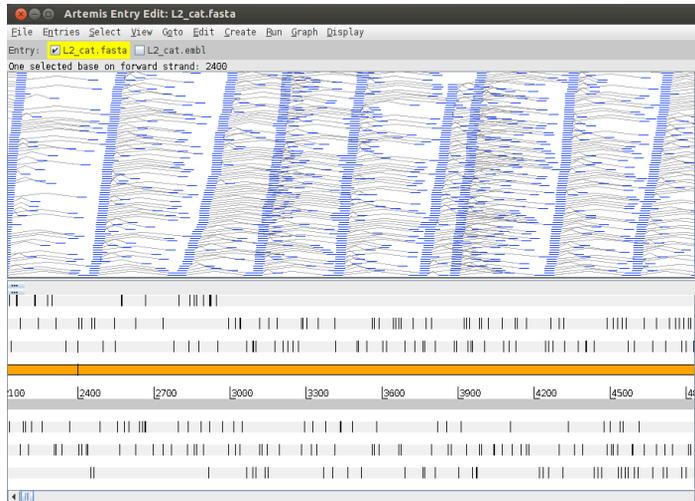


- We have already looked at 'Stack' view.
- The **Coverage** view: just like adding the coverage plot above you can also convert the Stack view to a coverage view. This can be useful when multiple BAM files are loaded as a separate plot is shown for each. You can also look at the coverage for each strand individually by using the **Coverage by Strand** option. You can now also view the coverage as a **Heat Map**, with darker colours displaying higher coverage.
- The '**Strand Stack**' view (shown below), with the forward and reverse strand reads above and below the scale respectively. Useful for strand specific applications or for checking for strand-specific artifacts in your data. See picture below.



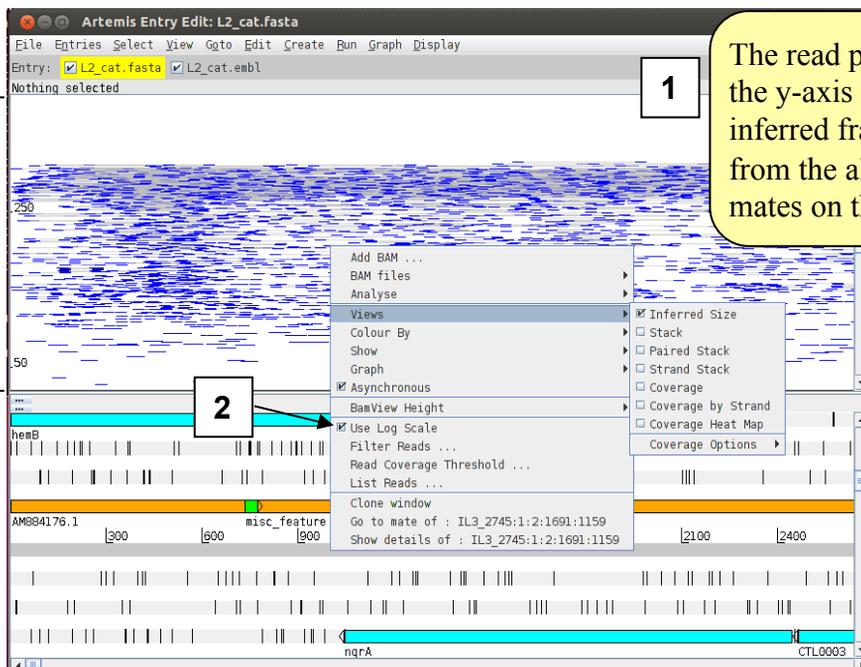
Alternative views continued:

d) The **‘Paired Stack’** view (inverted reads are red) joins paired reads. This can be useful to look for rearrangements and to confirm that regions are close together in the reference and the genome from which the aligned reads originate.



e) The **‘Inferred Size’** is similar to the ‘Paired Stack’ view, but it orders the read pairs along the y-axis by their inferred insert size which is calculated from the aligned positions of the mates on the reference sequence (1). Optionally you can display the inferred insert sizes on a log scale (2). Note that Illumina libraries are usually made from size fractionated DNA fragments of about 250bp-500bp.

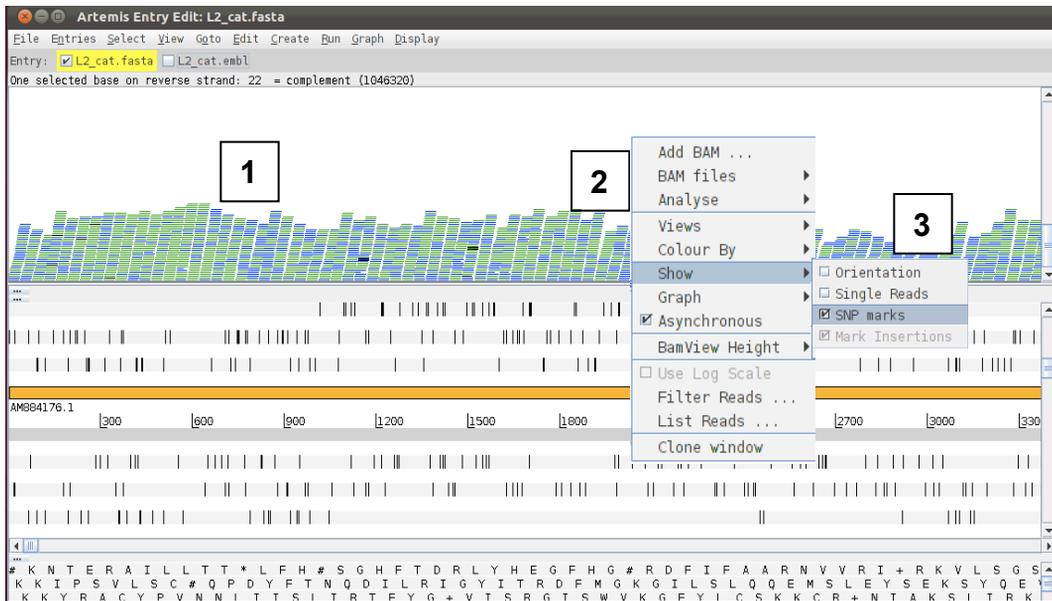
So this is not the actual library fragment size, although you would expect it to correlate closely, and be relatively constant, if your reference was highly conserved with the sequenced strain. The utility of this can seem a little obscure but its not and can be used to look for insertions and deletions as will be shown later in this Module.



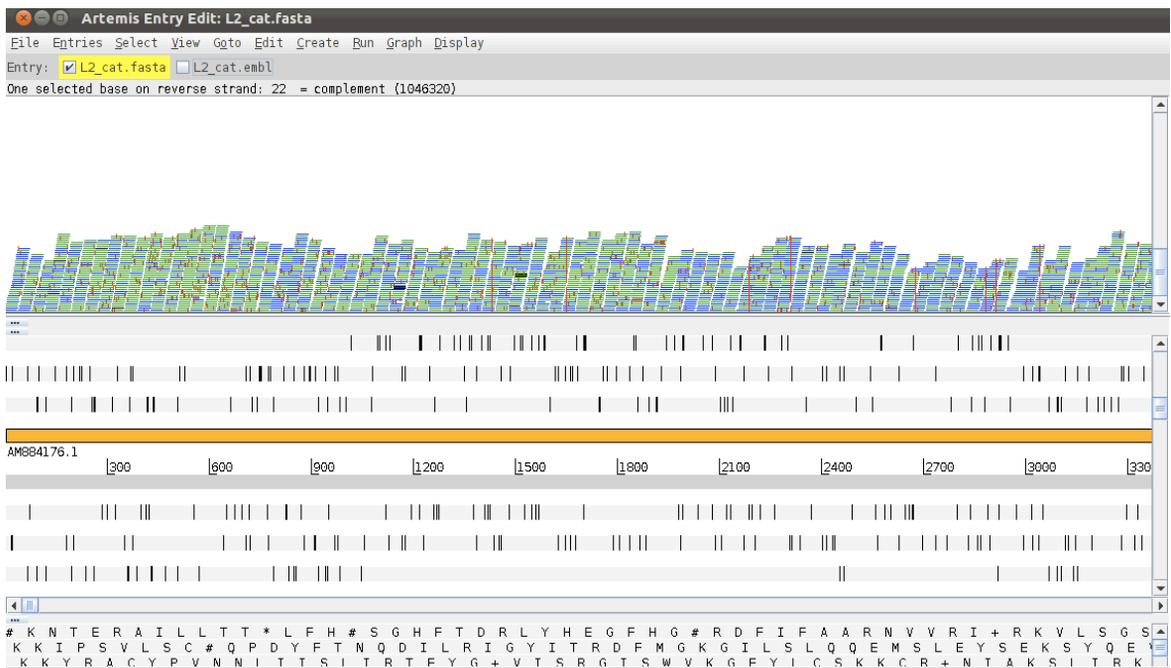
The read pairs are ordered along the y-axis (3) according to their inferred fragment size calculated from the aligned positions of the mates on the reference

## Viewing SNPs

Start by returning your view back to 'Stack' view.



To view SNPs use your right mouse button to click **in** the BAM view window (the panel showing the coloured sequence reads; **1 see above**). Then in the popup menu click on **2 'Show'** and **3** and check the 'SNP marks' box. SNPs in your data in comparison to the reference sequence are shown as red marks on the individual reads as shown below.

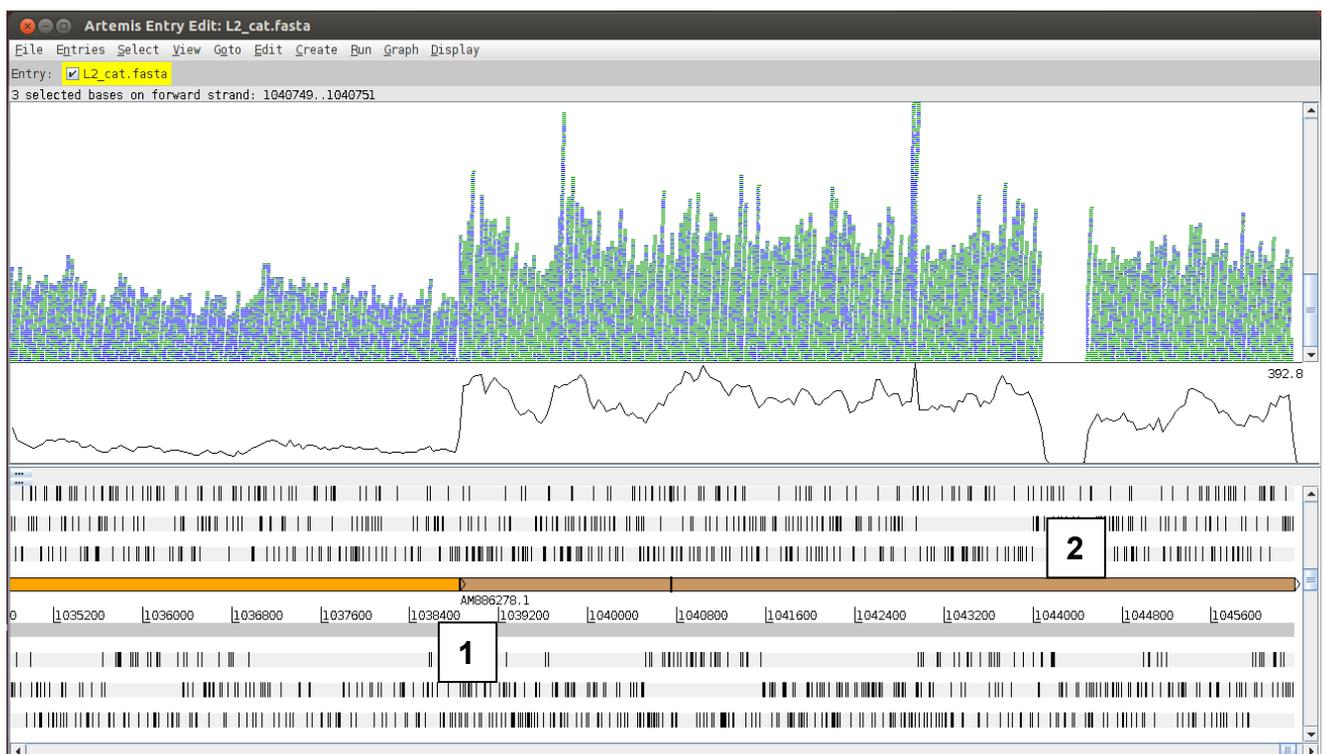




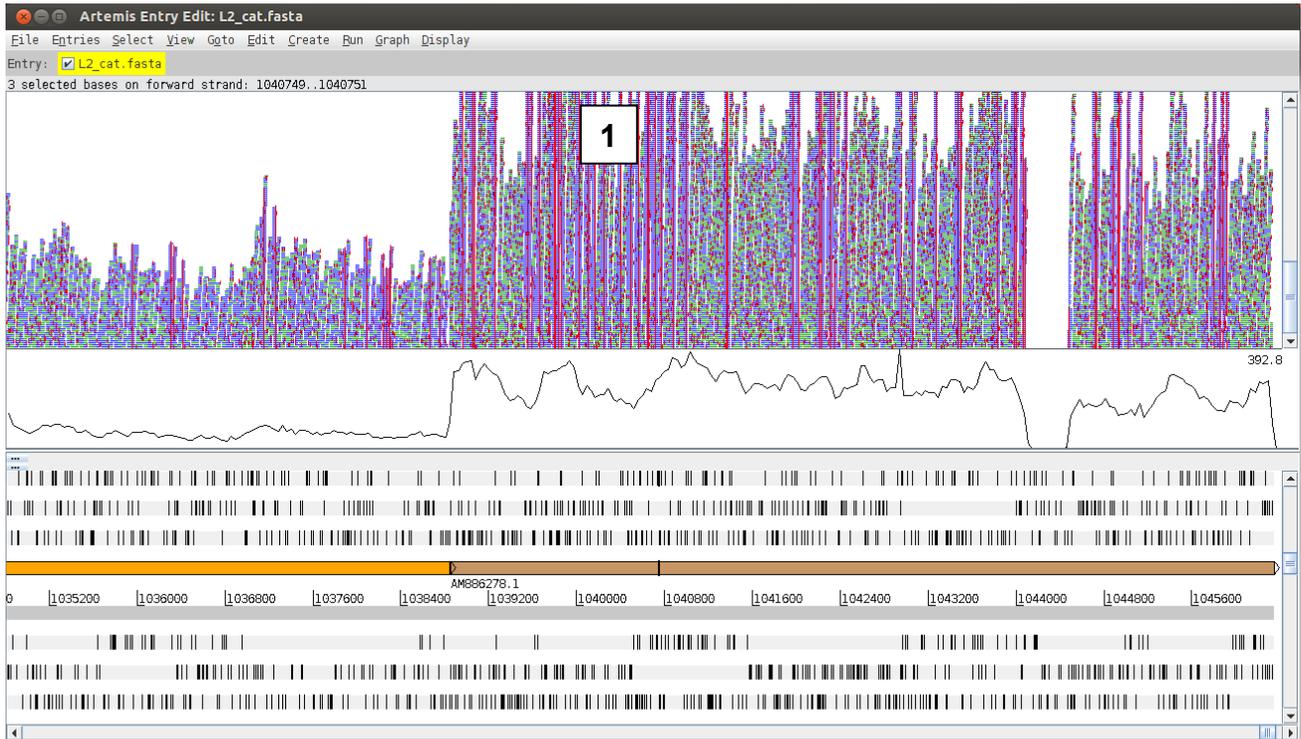
To give you a good biological example for when this type of information and analysis can be really informative and valuable, now do the following: using either the sliders, the GoTo menu or the 'Navigator', go to the end of the sequence or to base position 1043000. Adjust your view so you are in Stack view and have the depth of coverage graph showing. You might also need to adjust the Artemis window as well as the different panels.

If you adjust the zoom using the side sliders you should get a view similar to the one below. Notice two things: 1) the depth of coverage steps up at the beginning of the brown DNA line feature and 2) the coverage falls to zero within a region of this feature.

What could this mean?



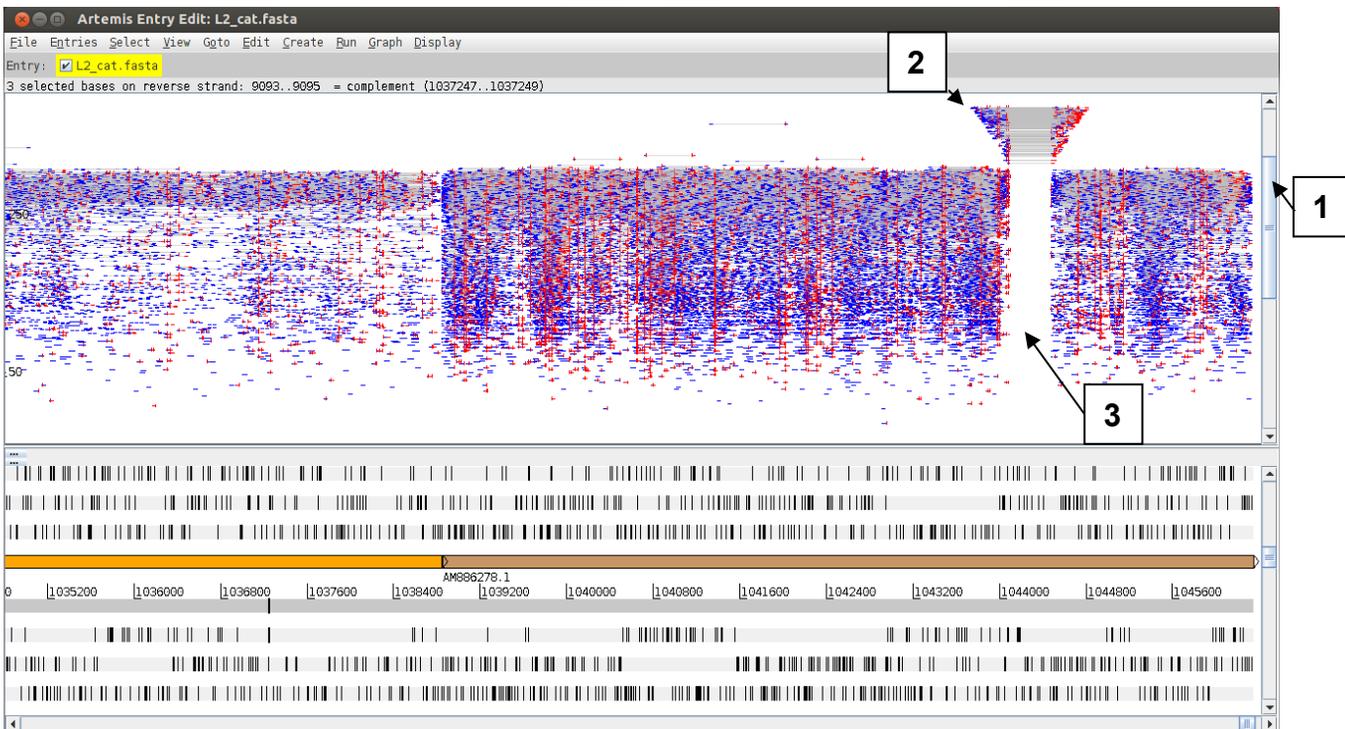
Note that the display changes when you switch on the display of SNPs (right click – Show SNP marks). This is due to a difference in display of duplicate reads. Reads having the same start and end position after mapping are considered duplicates and are displayed in green in the bam view. However, apart from the true SNPs, these duplicate reads are likely to differ in the sequencing errors, thus have to be displayed individually when the SNPs are displayed (1).



Coming back to the increase in coverage, the answer is that since part of the sequence you have been viewing is a plasmid (brown DNA feature) it is present in multiple copies per cell, whereas the chromosome is only present in one copy per cell (orange DNA feature). Therefore each part of the plasmid is sequenced more often than the rest of the genome leading to a higher read coverage in this area of the plot.

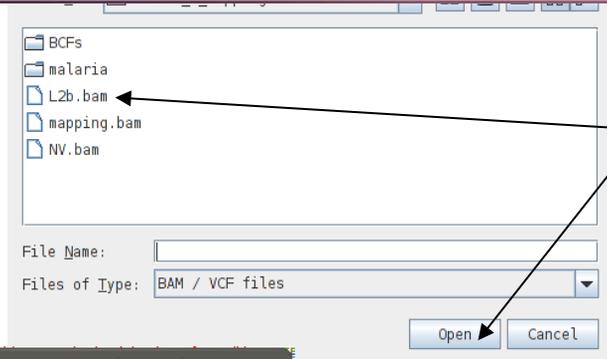
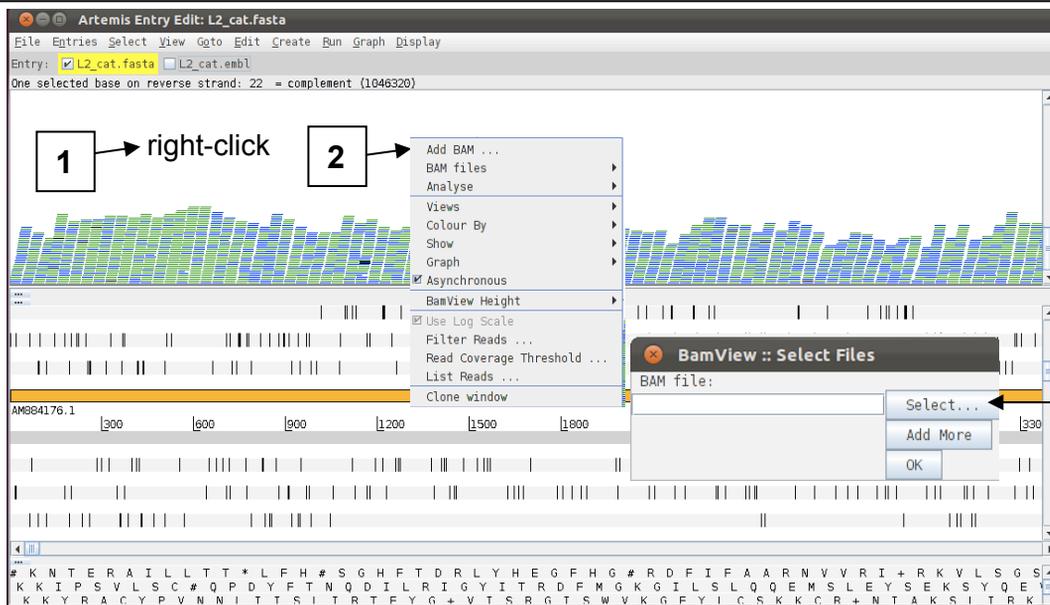
What about the region in the plasmid where no reads map?

This is where the Inferred Size view for the reads is useful. If you change the view as before to 'Inferred Size' and use the log scale you will see an image similar to the one below. You may have to adjust the view (1) to actually see the subset of reads that are shown above almost all other reads in this plot (2). The inferred insert size calculated from the alignment for this subset of reads is far bigger than the normal size range of other read pairs in this region (2) and there are no grey lines linking paired reads within the normal size range crossing this region (3). Together, this is indicative of a **deletion** in the DNA of the sequenced strain compared to the reference!

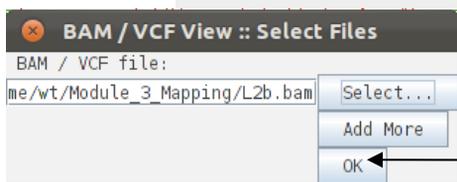


You can also view **multiple BAM files at the same time**. Remember that a BAM file is a processed set of aligned reads from (in this case) one bacterium aligned against a reference sequence. So in principle we can view multiple different bacterial isolates mapped against the same reference concurrently. The *C. trachomatis* isolate you are going to read in is *C. trachomatis* strain L2b. It is more closely related to the reference sequence that we have been using, hence the similar name.

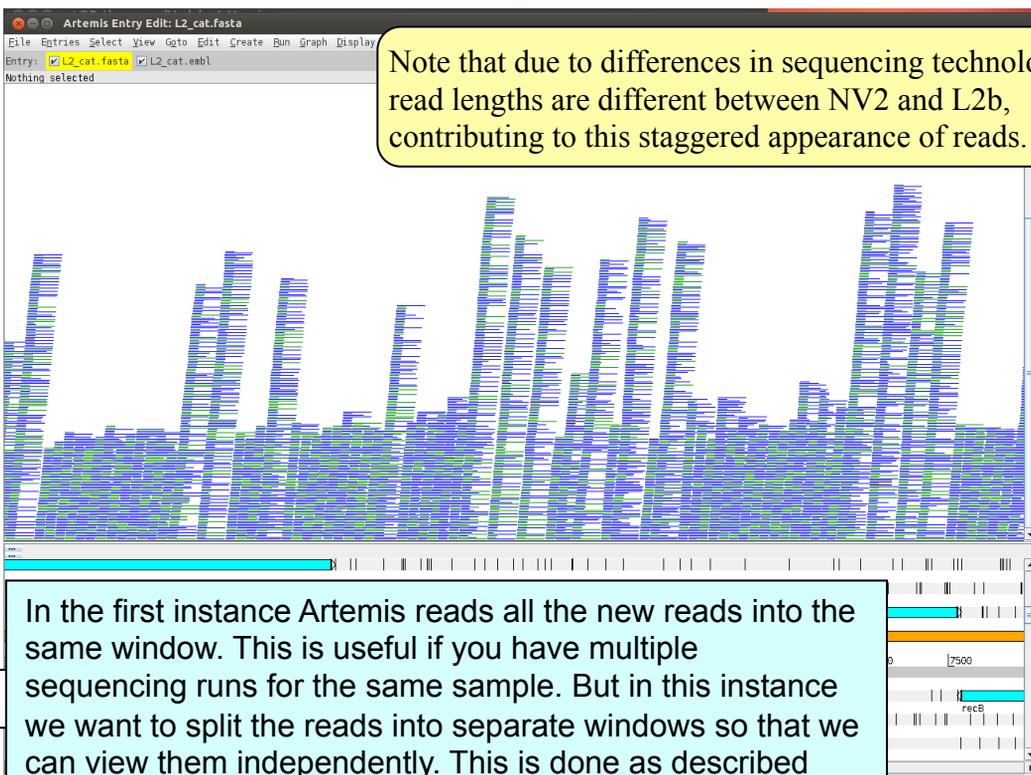
We are not going to redo the mapping for a new organism, instead we have pre-processed the relevant FASTQ data for you. The file you will need is called **L2b.bam**. Follow the instructions below. Start by going back to a normal stacked read view and zooming in more detail.



4 Within the Module 3 directory choose L2b.bam



5



6

In the first instance Artemis reads all the new reads into the same window. This is useful if you have multiple sequencing runs for the same sample. But in this instance we want to split the reads into separate windows so that we can view them independently. This is done as described below.

Artemis Entry Edit: L2\_cat.fasta

File Entries Select View Goto Edit Create Run Graph Display

Entry:  L2\_cat.fasta  L2\_cat.embl

Nothing selected

Nothing selected

AM884176.1 misc\_feature

800 1600 2400 3200 4000 4800 5600 6400 7200 8000 8800 9600 10400 11200

7

First, **clone** the BAM view window. Right-click over the BAM window and select 'Clone window'.

Artemis Entry Edit: L2\_cat.fasta

File Entries Select View Goto Edit Create Run Graph Display

Entry:  L2\_cat.fasta  L2\_cat.embl

Nothing selected

Nothing selected

AM884176.1 misc\_feature

800 1600 2400 3200 4000 4800 5600 6400 7200 8000 8800 9600 10400 11200

8

If you right-click over the top BAM window and select BAM files you can individually select the files as desired. This means you can display each BAM file in its own window by de-selecting one or the other file.

Now go back to the plasmid region at the end of the genome sequence and have a look at the previously un-mapped region located around base position 1044200. You can see that the newly added BAM file (for L2b) shows no such deletion with reads covering this region (as shown below). Have a look at the inferred read sizes, too.

Artemis Entry Edit: L2\_cat.fasta

File Entries Select View Goto Edit Create Run Graph Display

Entry:  L2\_cat.fasta  L2\_cat.embl

Nothing selected

Nothing selected

AM884176.1  Hide Close

AM884176.1 misc\_feature

800 1600 2400 3200 4000 4800 5600 6400 7200 8000 8800 9600 10400 11200

AM884176.1

CTL0895

tRNA-Pro

AM886278.1

1036000 1036800 1037600 1038400 1039200 1040000 1040800 1041600

repeat\_region

repeat\_r

repeat\_region r

repeat\_regi

repeat\_region

1042400 1043200 1044000 1044800 1045600

pL2-02

## Looking at SNPs in more detail

So far we have looked at SNP variation rather superficially. In reality you would need more information to understand the effect that the sequence change might have on for example coding capacity. For this we can view a different data type called Variant Call Format (**VCF**). In analogy to the SAM/BAM file formats, VCF files are essentially plain text files while **BCF** files represent the binary, usually compressed versions of VCF files. VCF format was developed to represent variation data from the 1000 human genome project and is likely to be accepted as a standard format for this type of data.

We will now take our NV.bam file and generate a BCF file from it which we will view in Artemis.

To do so go back to the terminal window and type on the command line:

```
samtools mpileup -DSugBf L2_cat.fasta NV.bam > NV_temp.bcf
```

```
wt@Pathogens: ~/Module_4_Mapping
wt@Pathogens:~/Module_4_Mapping$ samtools mpileup -DSugBf L2_cat.fasta NV.bam > NV_temp.bcf
[fai_load] build FASTA index.
[mpileup] 1 samples in 1 input files
<mpileup> Set max per-file depth to 8000
```

There are two more steps required before we can view out SNPs in Artemis. First, do the actual SNP calling:

```
bcftools view -bcg NV_temp.bcf > NV.bcf
```

Second, as before we have to index the file before viewing in in Artemis:

```
bcftools index NV.bcf
```

Now, before we view our SNP calls in the Artemis session that's still open, let's do a bit of house keeping because many of the files we have created are large and are no longer needed. So please delete the following files:

```
NV_temp.bcf F.sai R.sai mapping.sam mapping.bam
L2_cat.fasta.amb L2_cat.fasta.ann L2_cat.fasta.bwt
L2_cat.fasta.pac L2_cat.fasta.sa L2_cat.fasta.fai
```

You can do this either in your terminal window with UNIX command rm (see below):

```
rm files
```

OR you can use the more conventional file manager if you prefer.

```
wt@Pathogens:~/Module_4_Mapping$ rm -f NV_temp.bcf F.sai R.sai mapping.bam mapping.sam L2_cat.fasta.amb L2_cat.fasta.ann L2_cat.fasta.bwt L2_cat.fasta.pac L2_cat.fasta.sa L2_cat.fasta.fai
```

## File format: VCF / BCF (each line: one position in alignment)

Reference sequence name

Position

REF: base call in reference  
ALT: alternative base call in sequence data

Quality score of base call

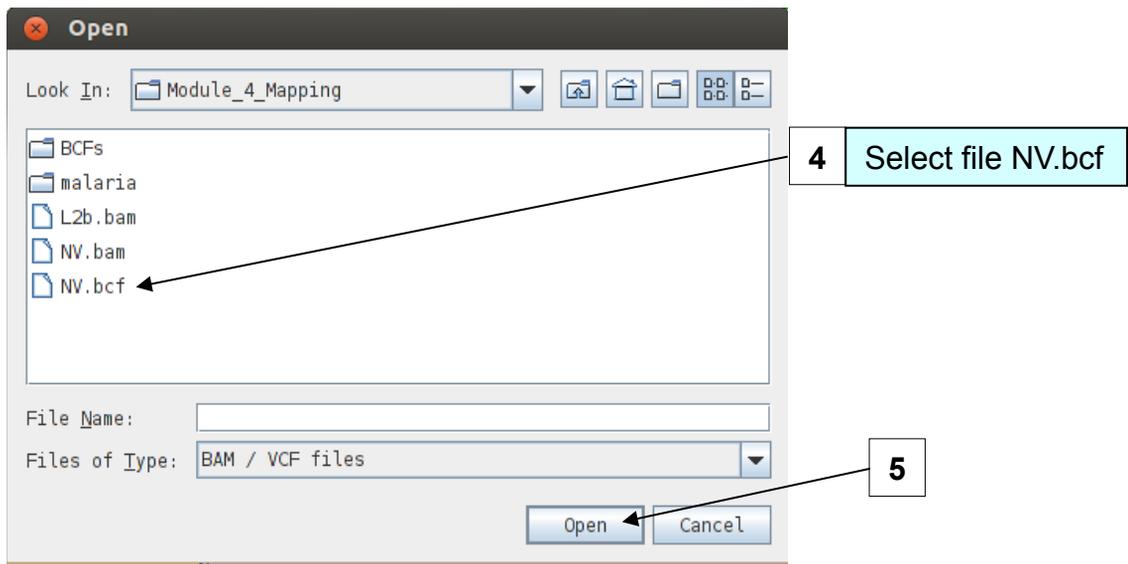
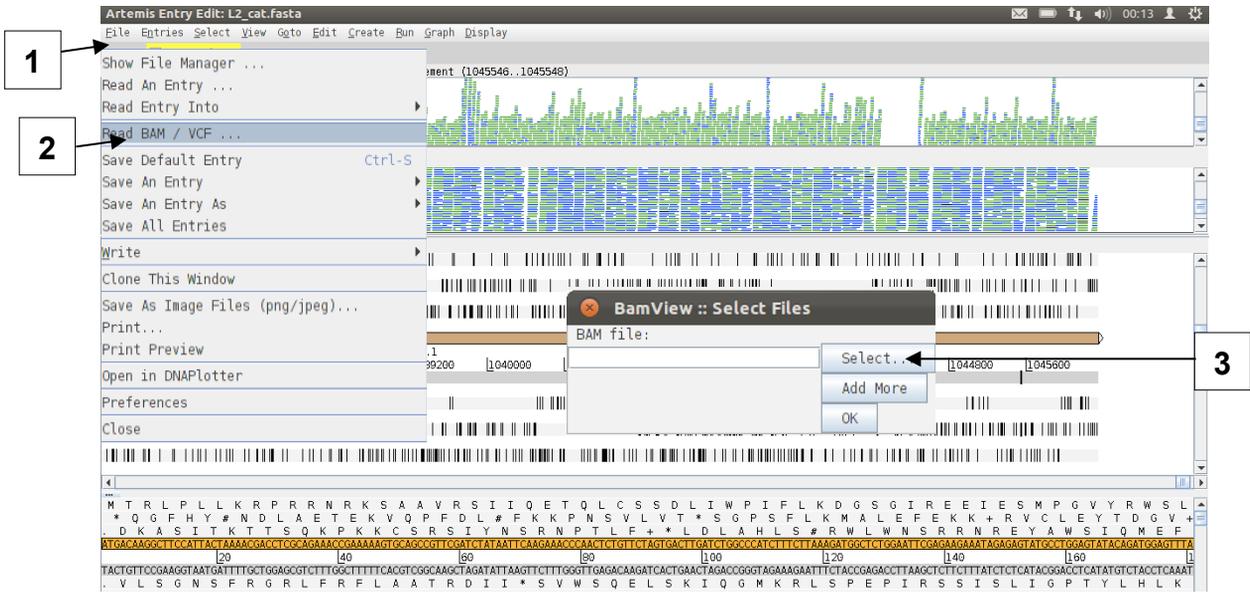
Detailed information:  
DP=read depth  
DP4=REF,REF,ALT,ALT  
MQ=mapping quality

Genotype call info

#CHROM	POS	ID	REF	ALT	QUAL	INFO	FORMAT
AM884176.1	24267	.	C	.	283	DP=159;AF1=0;AC1=0;DP4=75,84,0,0;MQ=60;FQ=-282	PL:DP:SP
AM884176.1	24268	.	C	.	283	DP=159;AF1=0;AC1=0;DP4=73,84,0,0;MQ=60;FQ=-282	PL:DP:SP
AM884176.1	24269	.	T	.	283	DP=156;AF1=0;AC1=0;DP4=75,81,0,0;MQ=60;FQ=-282	PL:DP:SP
AM884176.1	24270	.	G	A	222	DP=157;VDB=0.1063;AF1=1;AC1=2;DP4=0,0,75,82;MQ=60;FQ=-282	GT:PL:DP:SP:GQ

To look at a region with some interesting sequence variation, go again to the end of the sequence or to base position 1043000 using either the sliders, the GoTo menu or the 'Navigator'.

Next read the BCF file that you have just created into Artemis by selecting menus and options as shown below.



BCF window showing only sequence variation.

Adjust the slider and the size of the BAM & BCF panel window to look like this

Artemis Entry Edit: L2\_cat.fasta  
File Entries Select View Goto Edit Create Run Graph Display  
Entry: L2\_cat.fasta  
Nothing selected

AM884176.1  
AM884176.1  
AM886278.1  
Combine References

WT-Course [Running]  
Run Graph Display

Combine References Hide

You may have noticed that the SNPs in the BCF window are only shown for the chromosome. To show SNPs also on the plasmid click and select 'Combine References' here. The SNPs for the chromosome and plasmid should now be visible.

Below are the details of the three possible colour schemes for the variants in the BCF window panel (change the colour scheme via Right-click and Colour By). Note that this includes both SNPs and INDELS. Scroll along the sequence and see how many different kinds of variants you can find.

<b>1. Variant</b>	
Variant A	Green
Variant G	Blue
Variant T	Black
Variant C	Red
Multiple Alleles	Orange, with circle at top
Introducing stop codon	Circle in the middle, colour of variant
Insertion	Magenta
Deletion	Grey
Non-variant	Light grey
<b>2. Synonymous / Non-synonymous</b>	
Synonymous SNP	Red
Non-synonymous SNP	Blue
<b>3. Quality Score</b>	
Variants are all on a red colour scale with those with a higher score being darker red	

You can read in multiple BCF files from different related bacterial isolates. To do this right-click over the BCF window and select 'Add VCF' (remember BCF and VCF are essentially the same thing).

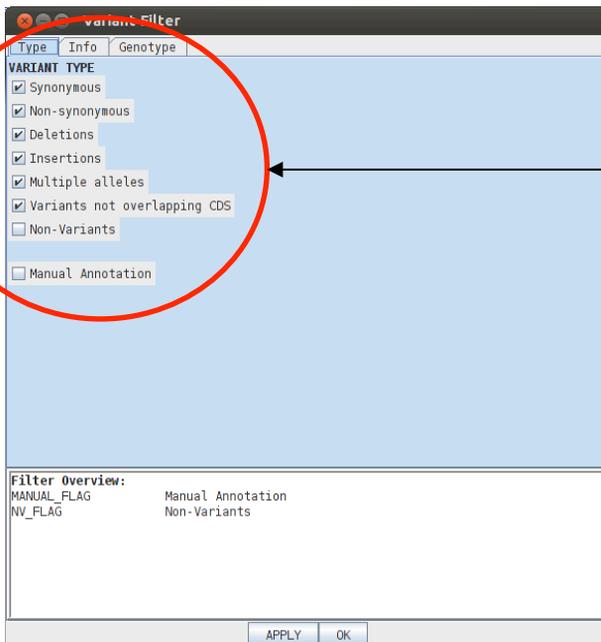
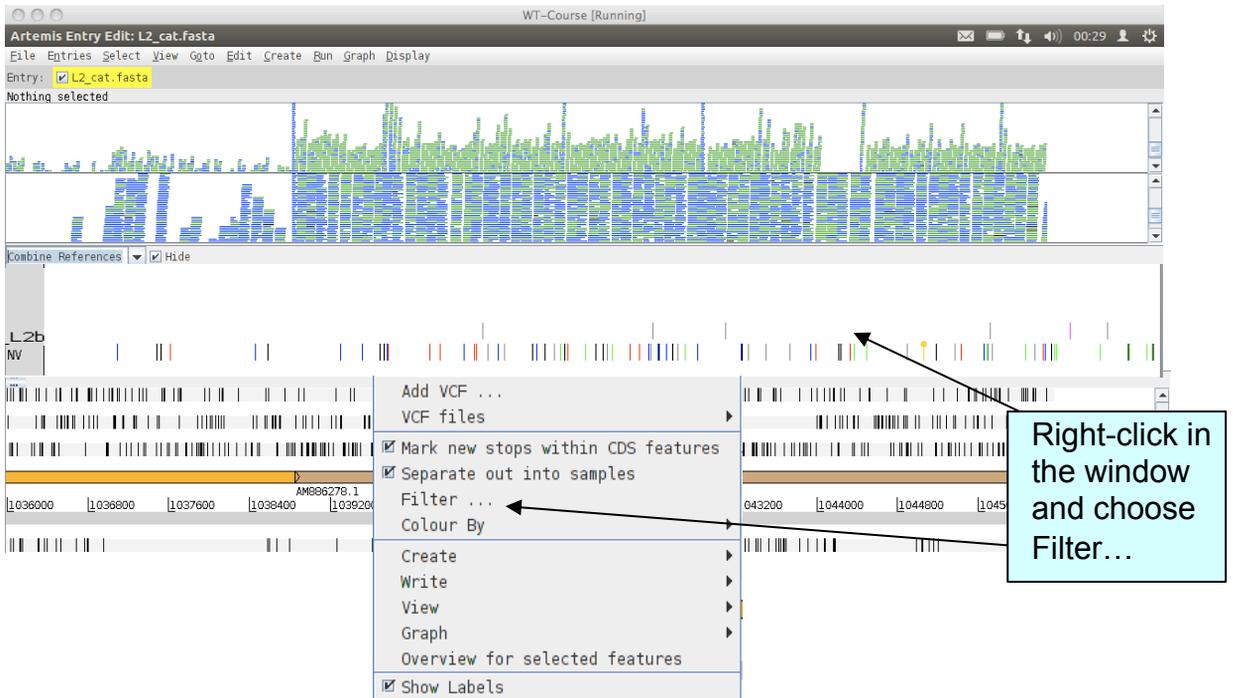
We have provided BCF files for the L2b LGV *C. trachomatis* strain. They are in the directory called "BCFs" in the Module\_4\_Mapping directory that you have been working in until now.

You can see the labels (taken from the file name) for the BCF files you have read into the window by right-clicking and choosing "Show labels".

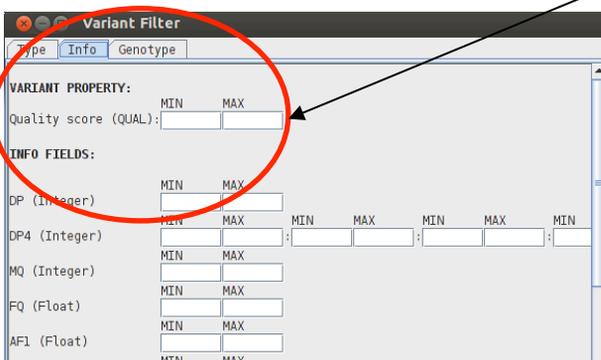
Once you have read the additional BCF file into Artemis right-click in the BCF window and check the 'Show Labels' box to make it easier to see which BCF file is which.

**What you should notice is that L2b has far fewer SNPs and INDELS than NV compared to the reference.** This is because L2b is an LGV strain of *Chlamydia* and NV is an STI strain. We will come back to these relationships later in the next Module.

As you may expect by now, Artemis also allows you to filter your VCF file.



Have a look through the variant filter window that pops up. You can select or unselect different SNP types or variants to modify your view. Non-variant sites are important because they differentiate sites where the data confirm that the sequence is the same as the reference from regions that appear not to contain SNPs simply because no reads map to them.



Like the BAM views you can also remove or include SNPs etc based on for example mapping score, depth of coverage or sequencing quality in the PROPERTY section listed under the INFO tab.

Useful cutoff values are e.g. DP of at least 10 and Qual of at least 30.

## 2. Exercise with data from *Plasmodium falciparum*

To give you a second example with exercises on how to use sequence read mapping, SNP calling and Artemis to identify relevant genomic variation, let's now turn to the data from *Plasmodium falciparum*, the eukaryotic pathogen that causes malaria in humans.

In the terminal, switch to the folder called 'malaria' using the Unix command 'cd':  
`cd malaria`

### Running a Bash script to do the work for us...

Mapping and aligning raw reads to a reference sequence is a common task in bioinformatics. To save time and to show you one example of how scripts can automate tasks we will use a **Bash script** to perform the following key tasks:

- map sequence reads from the malaria parasite strain IT to the reference sequence (3D7) using the BWA program
- call SNPs for the IT sequence data in comparison to the reference using the mpileup component of SAMtools

This shell script is very generic, and thus can be used over and over again to map different samples (also known as lanes) of sequence data.

To actually run the script, type the following on the command line:

```
./map_lanes.sh IT.Chr5_1.fastq IT.Chr5_2.fastq Pf3D7_05.fasta
BWA.IT.Chr5
```

Note that **BWA.IT.Chr5** still belongs to your command line! Please also note that this script will run for several minutes, so please be patient. Lots of information about the progress of the mapping will be printed to the screen, but its rare you'd ever need to look at it.

The commands performed by the script are listed on the next page. While the script is running, we can have a look at the commands, and the BASH structure. If you are not sure about certain commands, have a look back at the previous parts of this module or ask a course demonstrator or a class mate.

```
1  #!/bin/bash
2
3  #read in values from command line
4  fastq1=$1
5  fastq2=$2
6  ref=$3
7  output=$4
8
9  #index the reference file
10 bwa index $ref
11
12 #map the sequence data
13 bwa aln $ref $fastq1 > F.sai
14 bwa aln $ref $fastq2 > R.sai
15 bwa sampe -a 700 $ref F.sai R.sai $fastq1 $fastq2 > $output.sam
16
17 #create a sorted and indexed bam file
18 samtools view -b -S $output.sam > $output.tmp.bam
19 samtools sort $output.tmp.bam $output
20 samtools index $output.bam
21
22 #generate a BCF file and index it
23 samtools mpileup -ugf $ref $output.bam > $output.tmp.bcf
24 bcftools view -bcvg $output.tmp.bcf > $output.bcf
25 bcftools index $output.bcf
26
27 #clean up your directory of temporary files
28 rm -f $output.tmp.bcf F.sai R.sai $output.sam $output.tmp.bam
29
30 #clean up your directory of unnecessary files
31 rm -f $ref.amb $ref.ann $ref.bwt $ref.pac $ref.sa $ref.fai
```

- **Line 1** tells the computer which program to use to execute or interpret this file, in this case it is the **bash** program.
- Empty lines have been inserted for clearer structure and are not interpreted. Lines starting with a **#** are comments and are not executed either.
- **Lines 4-7** read in the values passed to the script from the command line. These values are called command line arguments and will be discussed in more detail later.
- **Line 10** indexes the reference file.
- **Lines 13-14** aligns the fastq files to the reference genome.
- **Line 15** extracts alignments from bwa's proprietary binary `.sai` file to a `.sam` file
  - Also, if you would like to know e.g. what the switch `"-a 700"` in the command `"bwa sampe -a 700"` does, you can check the program itself on the command line like this: **bwa sampe**
- **Line 18** converts the `.sam` file into a `.bam` file.
- **Lines 19-20** sort and index the `.bam` file so that it can be viewed in Artemis.
- **Lines 23-25** generate a `.bcf` file and index it.
- **Lines 28 and 31** remove temporary and unnecessary files.

### Variables

In bash scripting, as in any scripting language, you use containers called variables to store data, change it, and access it later. New variables can be created like this:

```
name=value
```

In a bash script, you must do it exactly like this, with no spaces on either side of the equals sign, the variable name must contain only alphanumeric characters and underscores, and it cannot start with a numeric character. Accessing the values stored in a variable can be done like this:

```
$name
```

In the `map_lanes.sh` script we create four different variables and use them to store the values that are passed to the script from the command line.

```
fastq1=$1
fastq2=$2
ref=$3
output=$4
```

Later in the script we access the values stored in these variables. For example, we index the reference genome by passing the value that is stored in the `ref` variable to the `bwa index` command.

```
bwa index $ref
```

## Command Line Arguments

Since we want to use the `map_lanes.sh` script on different datasets, it takes some arguments on the command line telling it what to work on. These arguments are:

- Name of the input fastq files
- Name of the reference file to use
- A prefix to use when writing output files (e.g. `<prefix>.bam`).

Remember we have run the `map_lanes.sh` script with the following command line arguments

```
./map_lanes.sh IT.Chr5_1.fastq IT.Chr5_2.fastq
                Pf3D7_05.fasta BWA.IT.Chr5
```

A shell script can have any number of command line arguments which can be accessed in the script using the variables `$0`, `$1`, `$2`, `$3`, `$4`, `$5` etc.

- The variable `$0` is the script's name, when run with the command above this variable will contain the value `"/map_lanes.sh"`
- The variable `$1` is the first argument passed to the script, when run with the command above this variable will contain the value `"IT.Chr5_1.fastq"`
- Similarly, the variable `$2` is the second argument and will contain the value `"IT.Chr5_2.fastq"`
- `$3` is the third argument and will contain the value `"Pf3D7_05.fasta"`
- `$4` is the fourth argument and will contain the value `"BWA.IT.Chr5"`
- The total number of arguments is stored in `$#`.

When the `map_lanes.sh` script is finished running, type `ls` to see the contents of the directory. You should see a new file called `BWA.IT.Chr5.bam` which contains the results of mapping the files `IT.Chr5_1.fastq` and `IT.Chr5_2.fastq` to the `Pf3D7_05.fasta` reference sequence.

Why is the file called `BWA.IT.Chr5.bam`?

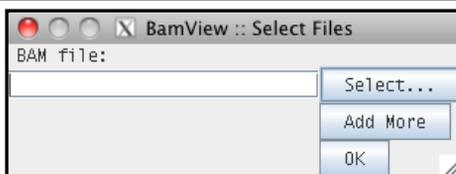
Now back to biology. It is thought that a duplication in the *mdr1* gene of *P. falciparum* is associated with drug resistance against the antimalarial mefloquine and that it may also modulate susceptibility to chloroquine, another antimalarial drug. For more information have a look in PubMed, e.g. at Borges et al. (2011) [PMID: 21709099] or at Mungthin et al. (2010) [PMID: 20449753]!

Please start up Artemis using the following command:

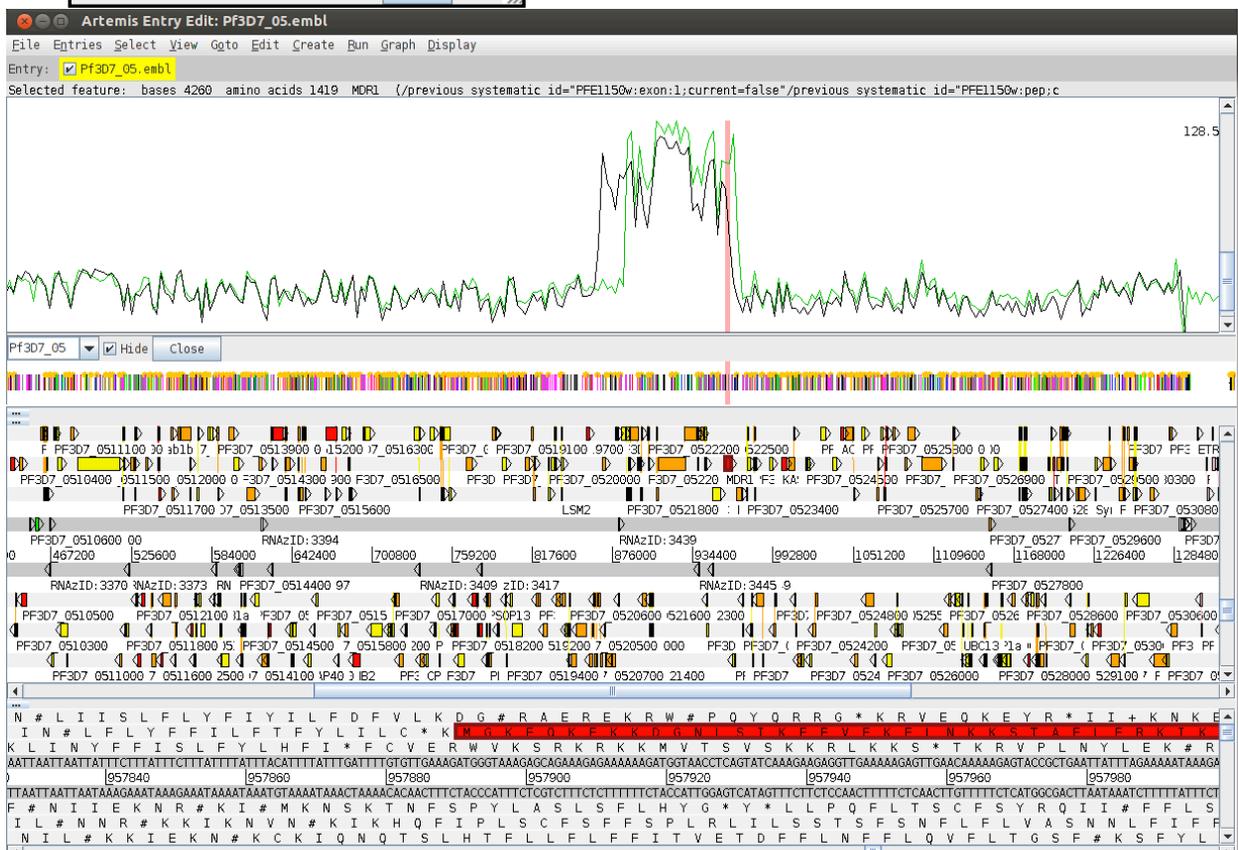
```
art -Dbam=BWA.IT.Chr5.bam,BWA.IT.Chr5.bcf Pf3D7_05.embl &
```

Once Artemis has started running on your screen **navigate to the *mdr1* gene locus** using e.g. the Navigator (Goto – Navigator... – Goto Feature With Gene Name). What can you say about the read coverage at this locus? (you may have to zoom out to get a good look at the whole region which is between 866,000 and 965,000bp).

So far you looked at the **IT strain**. What about the *mdr1* locus in the **Dd2 strain** of the malaria parasite? The Dd2 clone is known to be chloroquine resistant. Add its mapped reads (a file already prepared before the course) by right-clicking on the BAMview and choosing 'Add BAM...'. Select the file DD2.Chr5.bam



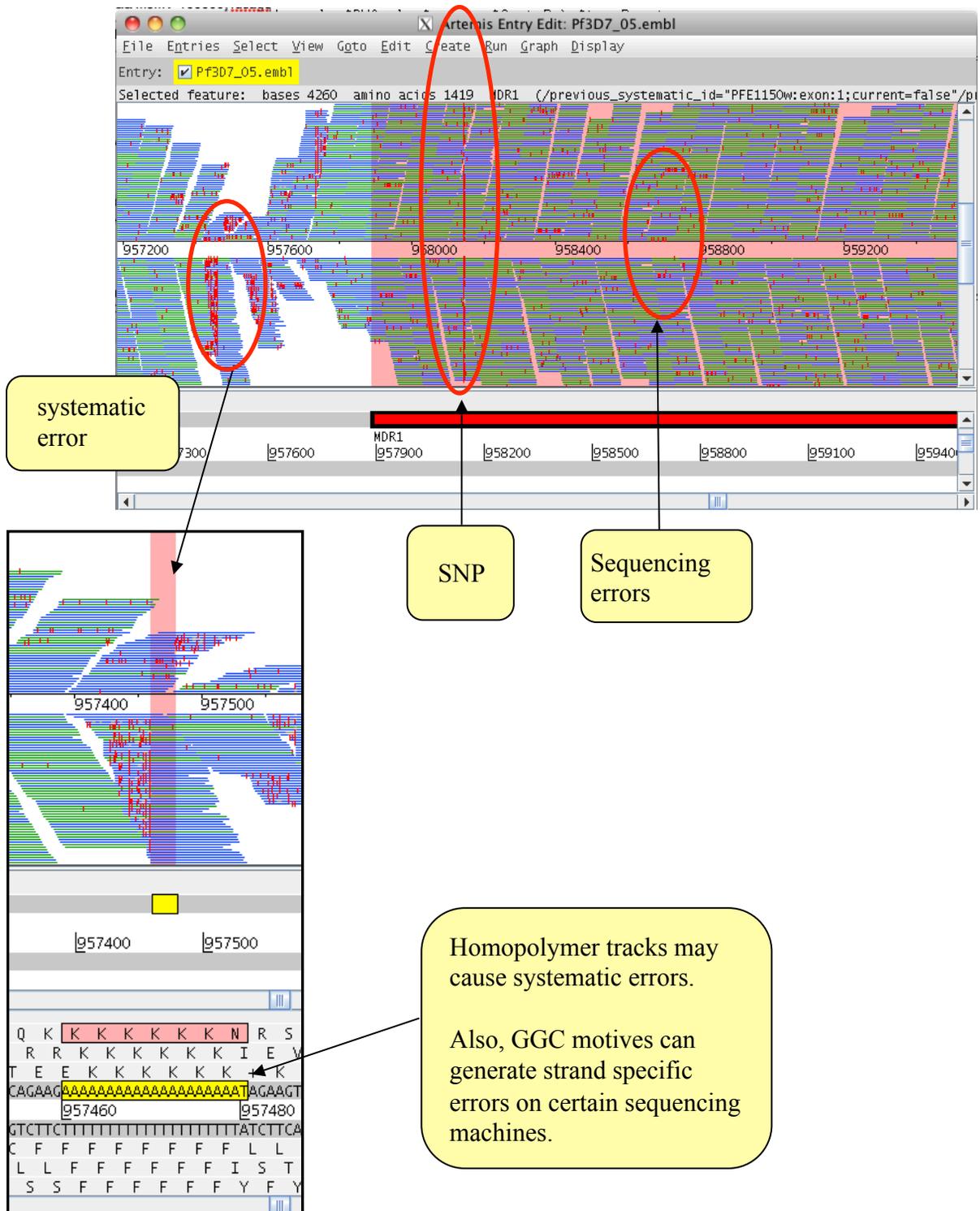
Select DD2.Chr5.bam



It looks like that both the IT and the Dd2 clone have a copy number variation (assuming the reference was assembled correctly). Is the duplication the same in both clones?

## SNPs

Let's have a look at SNPs now. To do so, **zoom in on the *mdr1* gene and make sure you are in Strand Stack view and have Show SNP marks selected.** As mentioned before, in addition to true SNPs some differences between the reference sequence and the mapped reads are due to sequencing errors. On average, 1 in every 100 bases in the reads is expected to be incorrect. In particular, some sequencing errors may be due to a systematic problem as illustrated below.



When you zoom in on position 958145 as far as you can go, you can see a SNP: here, both strain IT and Dd2 have a thymine where the reference (3D7) has an adenine.

What is the consequence of this SNP? Can we tell what effect it will have for the clones? Is this the mutation N86Y (or also simply referred to as Y86) that may modulate the degree of resistance to chloroquine? (See e.g. Mula et al. (2011) [PMID: 21810256])

Reference

Mapping reads

>>  
<<

I I R R N I T F F Y I C V W C N I K E I E F R \* \* Y # S Y N I I I S I Y R F S T I Y I

L S G G G T L P F F I S V F G V L K N M N G D I N P I I L S L V S I G L V O F I

Y Y Q E E H Y L F L Y L C L V # Y # R T \* I + V M I L I L # Y Y H + Y L + V + Y N L Y

80 |958100 |958120 |958140 |958160 |958180 |958200

To answer the question mentioned just above, right-click on the gene annotation of the *mdr1* gene and then choose View – Amino Acids Of Selection: here you can see which amino acids are normally coded for around amino acid position 86. Note also that AAT codes for Asn (N) and TAT codes for Tyr (Y).

Now have a look at neighbouring position 958146. What could this be? Is it from IT or DD2? To answer this question clone the window and display the reads of only one or the other parasite strain in each window, just as we did earlier in this module.

Add BAM ...

BAM files

Analyse

Views

Group BAMs ...

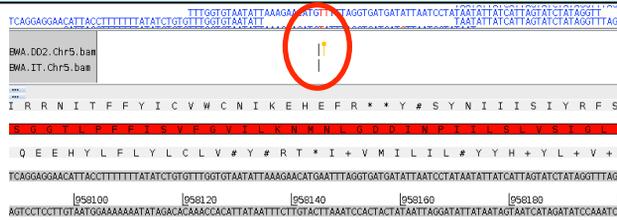
BWA, IT, Chr5, bam

DD2, Chr5, bam

This BAM shows the IT reads

Is there another SNP in Dd2?

Now also open the BCF file for the Dd2 clone to the window by right-clicking on the VCF/BCF pane of the window, choosing Add VCF..., and selecting file "DD2.Chr5.bcf".

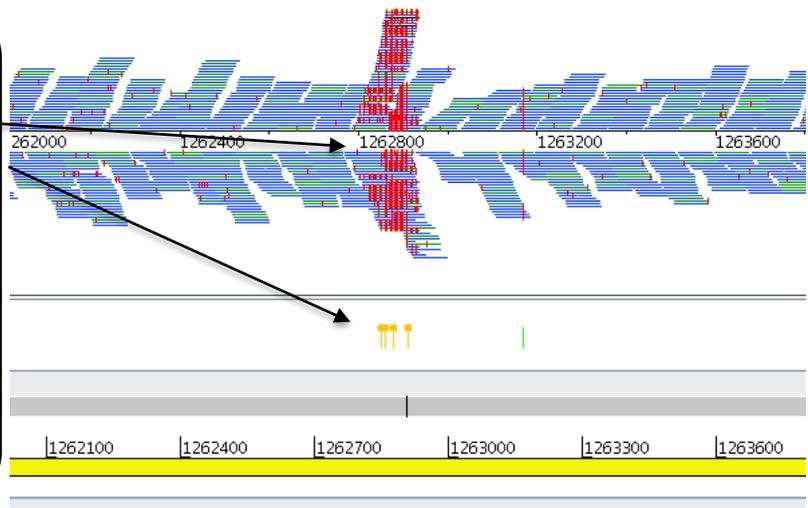


Note that the malaria parasite is a haploid organism. Yet the position above like many other in the genome have apparently more than one allele. What might be the explanation for this?

If you like, explore the genome a bit further, maybe you find some other surprising SNP calls and interesting genome variants!

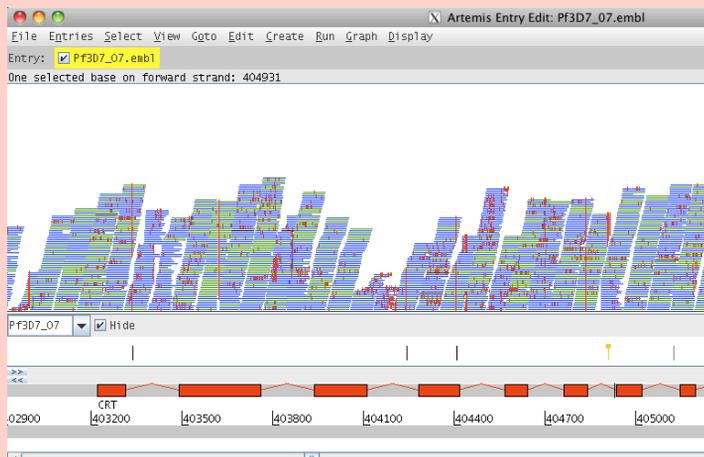
Here is another example of apparently "heterozygous" SNPs in this haploid organism. Note also that the read coverage in this region is also unusually high.

A region like this would be best resolved by *de novo* sequence assembly.



**Extra exercise** for those who are interested in drug resistance in *P. falciparum*: Can you find the *pfert* mutation associated with chloroquine resistance? It is on chromosome 7 at amino acid position 76 of the gene (K76T) – see e.g. Djimde et al (2001) [PMID: 11172152]. For this exercise you can start Artemis from the command line like this:

**art -Dbam=DD2.Chr7.bam,DD2.Chr7.bcf Pf3D7\_07.embl**



Note that AAA codes for Lys (K) while ACA codes for Thr (T).

**Extra exercises for bash scripting****Trouble-shooting – error checking**

Try running the `map_lanes.sh` script with the following command line arguments:

```
./map_lanes.sh IT.Chr5_1.fastq IT.Chr5_2.fastq
                Pf3D7.fasta IT.Chr5
```

Did the script run successfully? If not, why not?

Often, the difference between a good script and a poor script is assessed in terms of the robustness of the script. That is, the ability of the script to handle situations in which something goes wrong. In this case, does the `map_lanes.sh` script handle the situation where a file supplied by the user does not exist?

In this example we will look at improving the robustness of the `map_lanes.sh` script by adding some argument and error checking to the script. Using your preferred text editor open the file `map_lanes_validate_inputs.sh`. You should see the shell script shown on the next page.

Note that apart from error checking, this script also only performs the mapping, it does not generate the bcf files, and it does not clean up after itself!

This script performs some checks on the values passed to it from the command line and then performs a set of standard mapping tasks:

**Lines 1-7** tell the computer which program to use to execute this file and reads in the values passed to the script from the command line.

**Lines 10-13** checks that the correct number of command line arguments have been passed to the script. The lines say if the number of command line arguments passed to the script is NOT EQUAL TO 4, print a message to the screen telling the user what the correct usage is and exit the script.

**Lines 16-19** checks that all the files passed to the script exist. The lines say if the first fastq file does not exist OR the second fastq file does not exist OR the reference file does not exist, print an error message to the screen and exit the script.

**Lines 22-32** perform a set of standard mapping tasks.

**Please note:**

`$#` is the number of command line arguments

`!=` means NOT EQUAL TO

`$0` is the name of the script

`!` is the NOT operator

`-f` checks if a file exists

`||` means OR

```
1  #!/bin/bash
2
3  #read in values from command line
4  fastq1=$1
5  fastq2=$2
6  ref=$3
7  output=$4
8
9  #check the correct number of parameters have been passed to the
   script
10 if [ $# != 4 ]; then
11     echo "Usage: `basename $0` fastq1 fastq2 reference_file
   output_prefix"
12     exit
13 fi
14
15 #check the fastq and reference files passed to the script exist
16 if [ ! -f $fastq1 ] || [ ! -f $fastq2 ] || [ ! -f $ref ]; then
17     echo "Error: One of the input files does not exist"
18     exit
19 fi
20
21 #index the reference file
22 bwa index $ref
23
24 #map the sequence data
25 bwa aln $ref $fastq1 > F.sai
26 bwa aln $ref $fastq2 > R.sai
27 bwa sampe -a 700 $ref F.sai R.sai $fastq1 $fastq2 > $output.sam
28
29 #create a sorted and indexed bam file
30 samtools view -b -S $output.sam > $output.tmp.bam
31 samtools sort $output.tmp.bam $output
32 samtools index $output.bam
```

If you want to you could modify the script to check to see if the output file already exists. If it does exist, print a warning message and exit from the script.

**Decision Statements**

Sometimes you will want to perform different tasks depending on whether a condition is true or false. In bash this can be achieved with the keyword, `if`. The `if` statement consists of a condition that is evaluated, and a block of code that is run if the condition evaluates to true.

```
if [ CONDITION ]; then
# instructions to follow if condition is true
fi
```

**Loops**

In bioinformatics we often have to perform the same action/analysis multiple times. For example, its quite common to multiplex a 96 well plate of samples into a single Illumina lane, so to analyze your data you'll need to run the same commands on all 96 sets of sequencing data. Rather than

running a script over and over again, you can use a loop. It will keep running a set of commands until a condition is met, for example, loop over all files in a directory and run the commands on each file.

In bash this can be achieved with the keyword `FOR`. The `FOR` statement consists of a list and a variable name, then a block of commands to run. In the example below, the `ls` command is run to get a list of files in the current directory. Each file is then taken in turn and is assigned to

the variable `i`. The block of code is then run, and `$i` contains the name of the file. Here we just print out the filename, but you can use any command.

```
FOR i in $( ls ); DO
echo $i
DONE
```

**End of module...**

**ANY QUESTIONS?**

**Please feel free to ask at any time!**

Please close down Artemis, ready to start the next module.