Introduction to Sequence Informatics

Institute of Infection and Global Health

Introduction to sequence informatics

Session 2 - Genome sequencing and annotation

Next-generation sequencing has revolutionised genetics by allowing individual research groups to sequence and analyse genomes and transcriptomes on a regular basis. Reference sequences once took years to obtain, it is now possible to generate and assemble draft sequences in under a week. The most common bench-top sequencers are those made by Illumina, which produce high-throughput, short-read data in the form of millions of reads that are 75-300bp long.

This introduction will focus on short-read Illumina data as it is a common technology but it important to be aware of the existence of 3rd generation sequencers made by PacBio and Oxford Nanopore that are capable of producing reads that are tens (or even hundreds) of kilobases in length. At the moment these sequencers make up only a small proportion of the market but are likely to become common over the next few years.

Sequencing a genome is only the first step in the process. The sequencing process results in the genome being fragmented into millions of pieces, which we must stitch back together. There are two approaches to this, de novo assembly and mapping to a reference sequence. During this session, we will focus on de novo assembly. We will cover mapping during session four. Once we have assembled our genome we will then annotate it, which requires identifying features of interest (such as genes) and attempting to characterise them. This is an essential step when sequencing new organisms and many tools have been developed to help with the process. As always, it is important to validate annotation where possible by biologically confirming the existence and functions of annotated features.

Overview of Illumina Sequencing Data

Unlike traditional Sanger sequencing, the reads from 2nd and 3rd generation sequencers are reported in a fully digital format. Short read data from Illumina sequencers is typically stored in a plain text format, known as a fastq file, that looks like this:

```
@SRR292770.1 FCB067LABXX:4:1101:1155:2103/1
GGAGTCATCATACGGCGCTGATCGAGACCGCAACGACTTTAAGGTCGCA
+
FFFFCFGDCGGGFCGBGFFFAEGFG;B7A@GEFBFGGFFGFGEFCFFFB
```

The four lines above represent a single read and can be broken down as follows:

```
@SRR292770.1 FCB067LABXX:4:1101:1155:2103/1
```

This is the sequence ID line that identifies the run (SRR292770), the read number (.1), information about the machine and the run (FCB067LABXX:4:1101:1155:2103) and if the read is part of a pair (/1 with the second member of the pair being marked /2). It is standard to separate read pairs into two files, such as reads_1.fastq and reads_2.fastq.

```
GGAGTCATCATACGGCGCTGATCGAGACCGCAACGACTTTAAGGTCGCA
```

The second line of the fastq file contains the actual sequence of the read. Current Illumina machines typically generate reads that are 75 to 150bp long, although some may go up to 300bp. The third line is a simple

```
+
```

and marks the separation of the read from the quality data on the fourth line.

```
FFFFCFGDCGGGFCGBGFFFAEGFG;B7A@GEFBFGGFFGFGEFCFFFB
```

These letters and symbols encode the quality score for the corresponding bases on line two. Quality scores are represented in an ascii format, where a single character represents a fixed number. For example an exclamation mark, !, encodes the ascii value of 33.

Sequencing quality is measured using a Phred score, which represents the probability that a base call at a given position is incorrect. The Phred score is calculated as:

$Q = -10 \times \log_{10}(P)$

Where P is the probability of an incorrect call and Q is the Phred score. Phred quality scores are logarithmically linked to error probabilities. The current Illumina sequencing chemistry can achieve Phred scores up to 40, representing an accuracy of 99.99%.

Phred Quality Score (Q)	**Base call accuracy**	**Probability of incorrect call**	**Probability of incorrect call (P)**
10	0.90	1 in 10	0.10
20	0.99	1 in 100	0.01
30	0.999	1 in 1000	0.001
40	0.9999	1 in 10,000	0.0001
50	0.99999	1 in 100,000	0.00001
60	0.999999	1 in 1,000,000	0.000001

After calculation of the Phred score for each base it is converted into a single character using the simple formula of

Ascii character = Phred score + 33

Character	!	”	#	$	%	&	‘	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E	F	G	H	I
Phred score	0										10										20										30										40

It is worth noting that while this is the current standard, some older sequencers used the formula Phred + 64 in the conversion process.

If a base has a quality score of ‘B’ what is its Phred score and its accuracy?

What would an accuracy of 99.5 be encoded as using the Illumina format?

Exercise 1

During this exercise, we are going to use Illumina sequence data to assemble the genome of *E. coli* and compare the assembled contigs to a reference sequence. This exercise has been adapted from a 2013 tutorial 'Beginner's guide to comparative bacterial genome analysis using next-generation sequence data' by David Edwards and Kathryn Holt. The original tutorial is available from:

http://www.microbialinformaticsj.com/content/3/1/2/

While an updated version 2 is available from:

https://holtlab.net/2017/07/01/update-to-comparative-bacterial-genomics-tutorial/

1. Log on to the tryps01 server and copy the session2 folder into your home directory. The session2 folder is located at: /archived/Informatics/session2

2. Enter the session2 folder in your home directory and examine the files that are present. The directory should contain the following files:
   SRR292770_1.fastq
   SRR292770_2.fastq
   Ecoli_ref1.fasta
   Ecoli_ref2.fasta
   Ecoli_ref1_annotation.gff3
   Ecoli.sorted.bam
   rename.pl

3. Examine the fastq files, which contain the data for each individual read, and compare them to the examples we have already discussed. How many reads are present in each file? (Remember, each read is made up of 4 lines of data)

4. After sequencing our DNA the first step is to check the quality of the data. If the quality of our sequencing is low then we will struggle to re-assemble our genome. The quality of a read tends to be highest at the start and then slowly decrease. To improve the accuracy of our assemblies we can do two things. The first is to discard reads where the quality is low over the entire length of the read. The second thing we can do is to trim the reads, removing the low-quality bases at the end until the average read quality meets our threshold. For quality control, it is typical to set the threshold to a Phred score of ~15. We have already trimmed the reads for this exercise but you can look at the overall quality by running the command

   fastqc

   This will open up a window. Load the reads in SRR292770_1.fastq using

   File -> Open…

   and then select the file. Fastqc will then produce a quality report, have a look over it to get a feel for the different statistics and then close the program.

5. To assemble the reads into contigs we will be using a de novo assembler called Velvet. It consists of two programs, velveth and velvetg. Velveth examines the individual reads and builds a hash database of the sequences. The hash, commonly referred to as a kmer, is a short DNA sequence and the database contains a list of every location that that hash sequence can be found in the reads.

   For example, if we build a database using a hash size of four and the two reads below:

   Read 1: ACTTCTATTAGGCGATTATCG

   and

   Read 2: ATTGCGCGATAGCTATCGAT

   it will have entries that are equivalent to the following:

   ATTA Read 1: 7, 15
   CTAT Read 1: 5; Read 2: 13

   which indicates that the first sequence (ATTA) is found twice in read 1, starting at bases 7 and 15 while the second sequence is found in both read 1 and read 2 at the listed start positions.

The second program, Velvetg, takes this information and attempts to build contigs by joining the reads where they share overlapping hash sequences. Small contigs are then joined until no more reads can be added. In an ideal world, this would allow us to rebuild an entire chromosome, however, it is rare to be able to do this. We instead end up with hundreds, or thousands, of smaller contigs that velvet has been unable to join together. This may be because the sequence in the gap has not been sequenced, because it is repetitive or because velvet could not determine which contigs should have been joined to one another.

Read 1 and 2 in our above example share the CTAT motif but why wouldn't velvet join them together to create a contig?

6. Build your hash database using the following command:

   velveth Ecoli_35 35 -fastq -shortPaired -separate SRR292770_1.fastq SRR292770_2.fastq

   The command can be broken down as:

Ecoli_35	Output the results into a folder called Ecoli_35
35	Use a hash length of 35 nucleotides
-fastq	The reads are in a fastq format
-shortPaired	We are using short, paired end read data
-separate	The reads from each pair are in separate files
SRR292770_1.fastq	The name of the file with the first read from each pair
SRR292770_2.fastq	The name of the file with the second read from each pair

7. The next step is to generate the contigs using velvetg. This takes the hash database we have just created and tries to assemble the reads into increasingly large contigs by overlapping them where they share sequence identity. The command for this is:

   velvetg Ecoli_35 -clean yes -exp_cov 21 -cov_cutoff 2.81 -min_contig_lgth 200

   The options here break down as:

Ecoli_35	Use the hash library in folder Ecoli_35
-clean	Remove unneeded intermediary files
-exp_cov 21	The expected genome coverage is approximately 21 reads
-cov_cutoff 2.81	Discard any contigs that have an average coverage of less than 2.81
-min_contig_lgth 200	Only save contigs that are at least 200bp long

The values used here have already been optimised for the dataset, were you running the program with your own sequencing data you would need to spend time determining the best settings to assemble your contigs. Velvetg will create a number of output files and save them into the Ecoli_35 directory. We are most interested in the file called contigs.fa, which contains the assembled sequences.

8. After velvetg has finished running enter the Ecoli_35 folder and examine its contents. Find contigs.fa and examine it before moving it to the session2 directory.

9. One of the ways that we can assess the quality of the assembly is to determine the number of contigs, the total size of the assembled genome and the N50 value. Ideally, we want to have a small number of contigs and large values for the total genome size and N50 value.

   The N50 statistic is defined as the minimum contig length needed to cover 50% of the genome. It is calculated by collecting all of the contigs, ordering them by size and then adding together the largest contigs until their combined size is >50% of the total genome size. The N50 value is the size of the smallest contig that has been included.

   For example, if we had 4 contigs with sizes of 1kb, 2kb, 4kb and 5kb then our total genome size would be 12kb. To cover half the genome (6kb) we need to combine the two largest contigs, our N50 value is, therefore, 4kb.

   For our assembly, we can calculate the summary statistics using

   assembly-stats contigs.fa

   This will summarise the assembly using a number of statistics. What is the N50 value for the assembly? How many contigs were required to cover 50% of the total genome?

   You should have an assembly that is approximately 5.3mb in size. As the genome of *E. coli* has been sequenced many times before we know that it has a size of 4-6mb, dependent upon the strain being sequenced. Your assembly size should fall into the middle of this range, which is a good indicator that we have used sensible options when running velvetg.

10. We know that the reads we used in our assembly came from *E. coli* and that its genome has been sequenced many times before. We can use one of these existing sequences as a reference to assess the completeness of our assembly using a program you are already familiar with - MUMmer. As we did in session one we will use MUMmer to identify and plot the regions of our assembly that are collinear with the reference. The commands for this are

    ```
    mummer -mum -b -l 50 -c contigs.fa Ecoli_ref1.fasta > contigs-ref1.mums
    mummerplot --x11 -p contigs-ref1 contigs-ref1.mums
    ```

    Given these sequences came from the same organism do you get the result that you expected?

11. It should be clear that something is not right in our comparison of contigs.fa and Ecoli_ref1.fa. Check that mummer is working correctly by comparing Ecoli_ref1.fa with Ecoli_ref2.fa. Are these sequences collinear?

12. If you were to look closely you would find that the Ecoli_ref2.fa genome is not actually a reference sequence. It was assembled from the same reads that you have used in your assemblies. The difference is that we have reordered the contigs using Ecoli_ref1.fa as a guide and a piece of software called Mauve.

    Reordering contigs in this way is also known as scaffolding and is often used to create pseudo-chromosome sequences. This is useful when sequencing has been unable to produce a single contig that spans an entire chromosome. Here we used a sequence from the same species to order the contigs, but as we saw in Session One closely related species often share regions of collinearity and can, therefore, be used as a reference sequence.

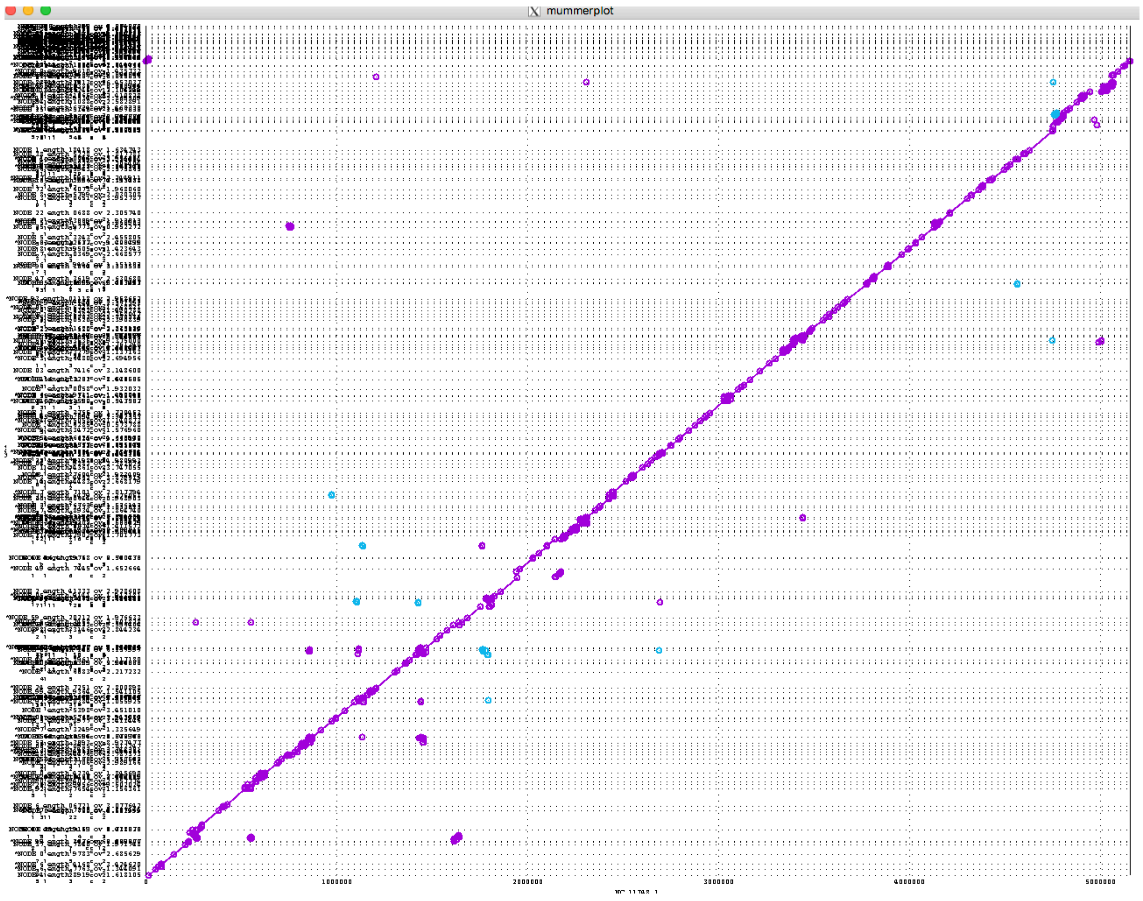
    Targeted sequencing approaches may subsequently be used fill the gaps between ordered contigs. This is a process called finishing, which also includes identifying and correcting errors in the sequence. These approaches are also important in confirming the order of contigs, as while many species may contain large regions of collinearity it is not unusual for chromosomes to undergo large-scale rearrangements or duplications. While 3rd generation, long read sequencing platforms are capable of generating reads many kilobases in length it is still unusual to obtain contigs that span an entire chromosome when dealing with eukaryotic chromosomes.

13. While we used Mauve to create the Ecoli_ref2.fa file it is possible to perform a basic realignment process using the nucmer function of mummer. The commands for this are:

    nucmer -p ref1-contigs-order Ecoli_ref1.fasta contigs.fa

    mummerplot -x11 --layout --filter ref1-contigs-order.delta -R Ecoli_ref1.fasta -Q contigs.fa

    This should produce a plot similar to the one below, with Ecoli_ref1 on the x-axis and the reordered sequences from contigs.fa on the y-axis. The numerous labels on the y-axis are present because our assembly consists of multiple contigs, as opposed to the single chromosomal sequence used for the x-axis.



Exercise Two

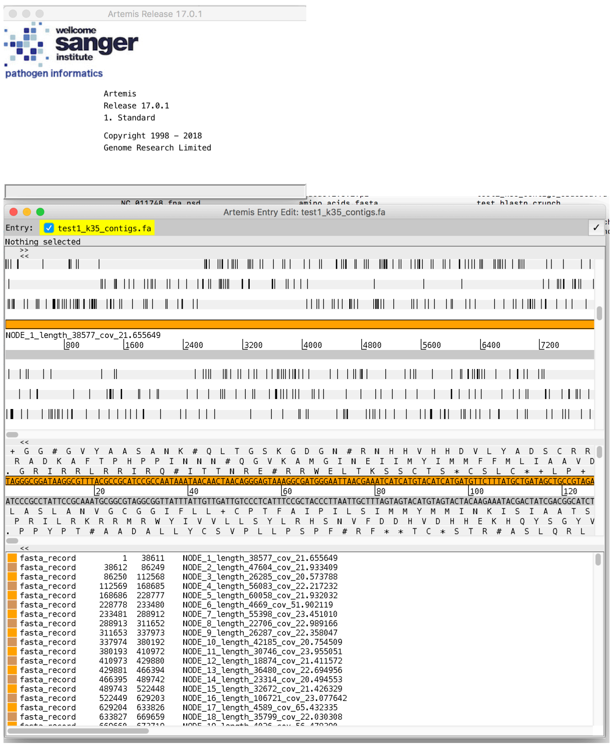
After assembling a new genome the next step in the process is to annotate it, by which we mean identifying and labelling the features that are present. These features may include genes, centromeres, non-coding RNAs or microsatellite repeat sequences. While software exists that can automatically annotate a genome it remains a time consuming process and it is often necessary to manually correct / validate features. This is especially true of features that are unique to a genome, as unusual sequences are the least likely to be properly annotated using automated software.

In this exercise we will run through a small number of the steps that are involved in annotating a genome. We could easily dedicate the entire course to the process and still only touch on the subject.

1. First we will examine the genome that we have created using a widely used piece of software called Artemis. The command is:

   art contigs.fa &

   You should see the following two windows open. The first is the main Artemis menu window, the second is our assembly window.


Artemis Release 17.0.1
wellcome sanger institute
pathogen informatics
Artemis Entry Edit: test1_k35_contigs.fa

```
Artemis
Release 17.0.1
1. Standard

Copyright 1998 - 2018
Genome Research Limited
```

```
Entry: ✓ test1_k35_contigs.fa
Nothing selected
NODE_1_length_38577_cov_21.655649
```

```
fasta_record          1   38611    NODE_1_length_38577_cov_21.655649
fasta_record      38612   86249    NODE_2_length_47604_cov_21.933409
fasta_record      86250  112568    NODE_3_length_26285_cov_20.573788
fasta_record     112569  168685    NODE_4_length_56083_cov_22.217232
fasta_record     168686  228777    NODE_5_length_60058_cov_21.932032
fasta_record     228778  233480    NODE_6_length_4669_cov_51.902119
fasta_record     233481  288912    NODE_7_length_55398_cov_23.451010
fasta_record     288913  311652    NODE_8_length_22706_cov_22.989166
fasta_record     311653  337973    NODE_9_length_26287_cov_22.358047
fasta_record     337974  380192    NODE_10_length_42185_cov_20.754509
fasta_record     380193  410972    NODE_11_length_30746_cov_23.955051
fasta_record     410973  429880    NODE_12_length_18874_cov_21.411572
fasta_record     429881  466394    NODE_13_length_36480_cov_22.694956
fasta_record     466395  489742    NODE_14_length_23314_cov_20.494553
fasta_record     489743  522448    NODE_15_length_32672_cov_21.426329
fasta_record     522449  629203    NODE_16_length_106721_cov_23.077642
fasta_record     629204  633826    NODE_17_length_4589_cov_65.432335
fasta_record     633827  669659    NODE_18_length_35799_cov_22.030308
```

2. The & symbol at the end of the command tells the terminal to open Artemis in the background. Doing this keeps the command line free for further use, allowing us to run additional commands without having to quit Artemis first.

3. Let’s have a closer look at the Artemis window. At the top, you will find your file menu, containing ‘File’, ‘Options’ and ‘Windows’ menus. These menus provide a number of standard options for opening files and loading them into Artemis. We loaded the contigs.fa file directly using the command line but you could also click File -> Open… and then selected contigs.fa, which would open the second window you can see.

4. The second window is our Artemis entry window, it contains a series of subsections.

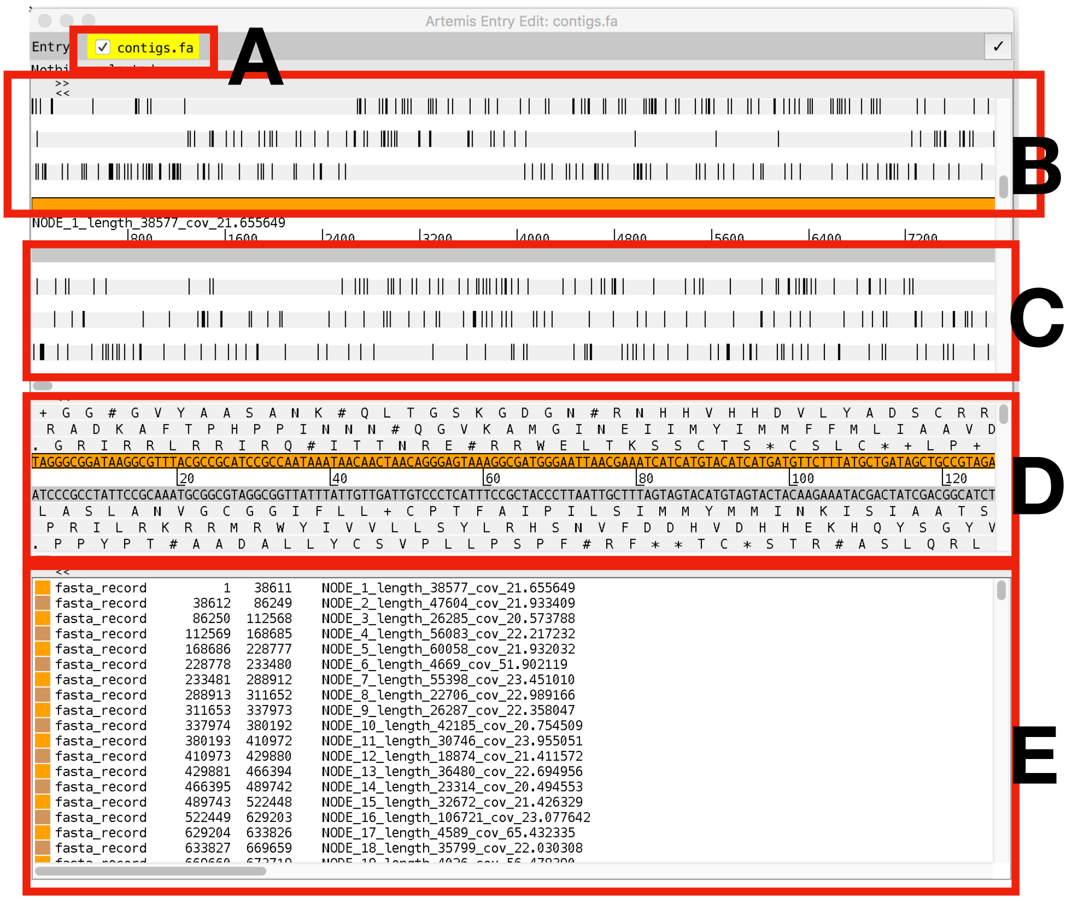
   A - The entry name, with a checkbox allowing us to turn it on and off. We have only loaded a single entry but Artemis allows for multiple entries to be loaded on top of one another.

   B - The forward features pane. Each vertical black line in this pane represents the position of a putative stop codon while the orange bar at the bottom highlights the current contig. The rows represent each of the three possible reading frames. We can control our position using the slider at the bottom of the pane, allowing us to move along the length of the sequence.

   C - The reverse features pane, which is the same as the forward pane, but using the negative strand of the contig.

   D - A duplicate of the forward and reverse panes. This allows us to examine sequences at different positions or, as in the default view, at different zoom levels. In this example, D is zoomed in to show individual codons of each reading frame. The zoom level can be controlled using the slider on the right.

   E - Our list of features. At the moment this contains only a list of our individual contigs, in a fully annotated genome it would contain each gene and feature.


Artemis Entry Edit: contigs.fa
Entry
contigs.fa
A
B
NODE_1_length_38577_cov_21.655649
800
1600
2400
3200
4000
4800
5600
6400
7200
C
D
E

```
+ G G # G V Y A A S A N K # Q L T G S K G D G N # R N H H V H H D V L Y A D S C R R
 R A D K A F T P H P P I N N N # Q G V K A M G I N E I I M Y I M M F F M L I A A V D
. G R I R R L R R I R Q # I T T N R E # R R W E L T K S S C T S * C S L C * + L P +
TAGGGCGGATAAGGCGTTTACGCCGCATCCGCCAATAAATAACAACTAACAGGGAGTAAAGGCGATGGGAATTAACGAAATCATCATGTACATCATGATGTTCTTTATGCTGATAGCTGCCGTAGA
                  20                 40                 60                 80                 100                120
ATCCCGCCTATTCCGCAAATGCGGCGTAGGCGGTTATTTATTGTTGATTGTCCCTCATTTCCGCTACCCTTAATTGCTTTAGTAGTACATGTAGTACTACAAGAAATACGACTATCGACGGCATCT
 L A S L A N V G C G G I F L L + C P T F A I P I L S I M M Y M M I N K I S I A A T S
  P R I L R K R R M R W Y I V V L L S Y L R H S N V F D D H V D H H E K H Q Y S G Y V
. P P Y P T # A A D A L L Y C S V P L L P S P F # R F * * T C * S T R # A S L Q R L
```

```
fasta_record        1   38611   NODE_1_length_38577_cov_21.655649
fasta_record    38612   86249   NODE_2_length_47604_cov_21.933409
fasta_record    86250  112568   NODE_3_length_26285_cov_20.573788
fasta_record   112569  168685   NODE_4_length_56083_cov_22.217232
fasta_record   168686  228777   NODE_5_length_60058_cov_21.932032
fasta_record   228778  233480   NODE_6_length_4669_cov_51.902119
fasta_record   233481  288912   NODE_7_length_55398_cov_23.451010
fasta_record   288913  311652   NODE_8_length_22706_cov_22.989166
fasta_record   311653  337973   NODE_9_length_26287_cov_22.358047
fasta_record   337974  380192   NODE_10_length_42185_cov_20.754509
fasta_record   380193  410972   NODE_11_length_30746_cov_23.955051
fasta_record   410973  429880   NODE_12_length_18874_cov_21.411572
fasta_record   429881  466394   NODE_13_length_36480_cov_22.694956
fasta_record   466395  489742   NODE_14_length_23314_cov_20.494553
fasta_record   489743  522448   NODE_15_length_32672_cov_21.426329
fasta_record   522449  629203   NODE_16_length_106721_cov_23.077642
fasta_record   629204  633826   NODE_17_length_4589_cov_65.432335
fasta_record   633827  669659   NODE_18_length_35799_cov_22.030308
```

5. Examine your assembly using Artemis to become familiar with the basic controls. Spend at least a few minutes doing this. If you right-click on any of the panes it will bring up an extensive options menu. Use one of these menu's to label the position of putative start codons, which will be highlighted in pink.

6. Turn off the start codons and look closely at the position of the stop codons (the black marks) along the assembly. Are there any patterns that you can spot?

7. You will have hopefully spotted that along each contig there are regions where there are no stop codons. These gaps are putative Open Reading Frames and may indicate that this region codes for a gene.

   We can annotate these putative ORFs as Coding Sequences (CDS) by clicking:

   Create -> Mark Open Reading Frames…

   Using this option annotate all of the putative ORFs in your assembly with a minimum size of 1000 amino acids. Ensure that you check the 'break at contig boundaries' box as we know that our assembly is unordered and does not represent a finished genome.

This will create a new entry at the top of the window called 'ORFS_1000+' and will annotate all of your ORFs in blue along our sequence. As these putative CDS are large there will not be many of them, you may need to scan along the assembly to find them. If we were annotating the genome for real we would have set our minimum size much smaller than this.

8. Now that we have identified our putative CDSs we need to identify them. One way of doing this is by blasting them against a database of known sequences like we did in Session One. To do select the first 5 CDS entries in the features pane by clicking on them while holding down shift. We can then write them to a file using:

   File -> Write -> Amino Acids of Selected Features

   And save the file with the name ORF1000

9. During Session One we performed our blast against a local database. This time, however, we will use the entire NCBI database as a reference. Return to the command line and press enter to get the prompt back. Before we can run the BLAST command we need to quickly rename our sequences. When creating the CDS entries Artemis gives each the same name of 'CDS' but for a few of the programs we are about to use the sequences must have unique names. We will use a quick perl script to do this, which is run using the command:

   ```
   perl rename.pl ORF1000 > ORF1000-v2
   ```

   We will then use the following command to blast our sequences:

   ```
   blastp -db nr -task blastp-fast -query ORF1000-v2 -num_descriptions 10 -
   num_alignments 10 -remote -evalue 1e-15 -out myBlastResults &
   ```

   This step may take some time so we will run it in the background by adding the & symbol at the end of the command. You can check if the command is still running by using the command

   ```
   jobs
   ```

   This will return a list of all jobs you are running in the background and an ID for it in brackets on the left (your job should be listed as [1]).If we wanted to return it to the foreground then we would use the command

fg %1

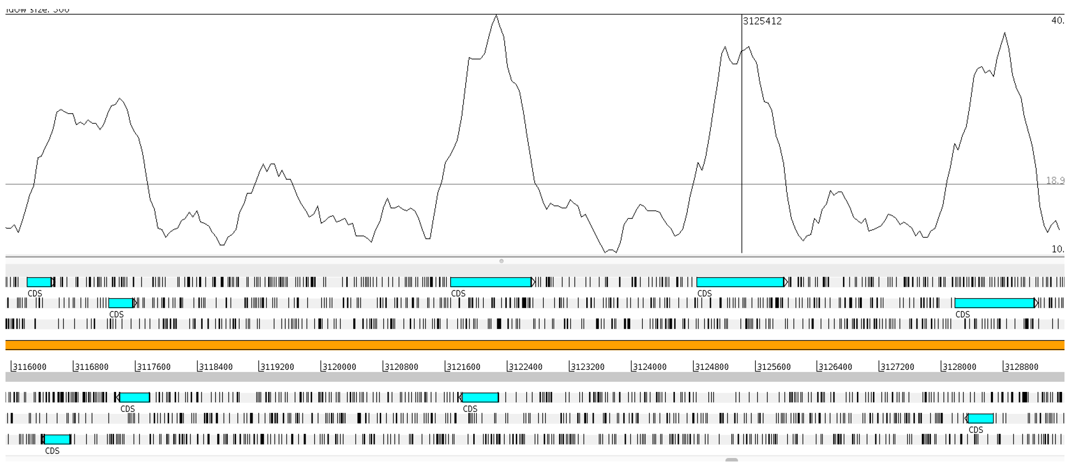
If we had wanted to bring job 2 to the foreground we simply replace the 1 with 2. While you await the blast results return to Artemis and continue working through the exercise, we will check back on BLAST later in the exercise.

10. BLAST is a powerful tool but it can run into problems when a sequence is novel and shares little similarity with entries in the database. Because of this it is common to use a range of tools during annotation, including some built in to Artemis. One of these is the ability to plot the base composition across the genome. To produce this plot click on

    Graph -> G/C Content

    You should see a new panel open up at the top of the Artemis window which plots the G/C content across a region. Scan through the genome and look at whether the base composition changes where you have identified putative CDS.

    While there is not a clear correlation in *E. coli* for some species it is possible to clearly identify coding sequences based on the variation in base composition. The figure below is from *P. falciparum,* one of the causative agents of malaria. The genome is, on average, comprised of only 20% GC but there is a significant increase in GC content associated with genes, allowing for their approximate position to be identified and annotated.



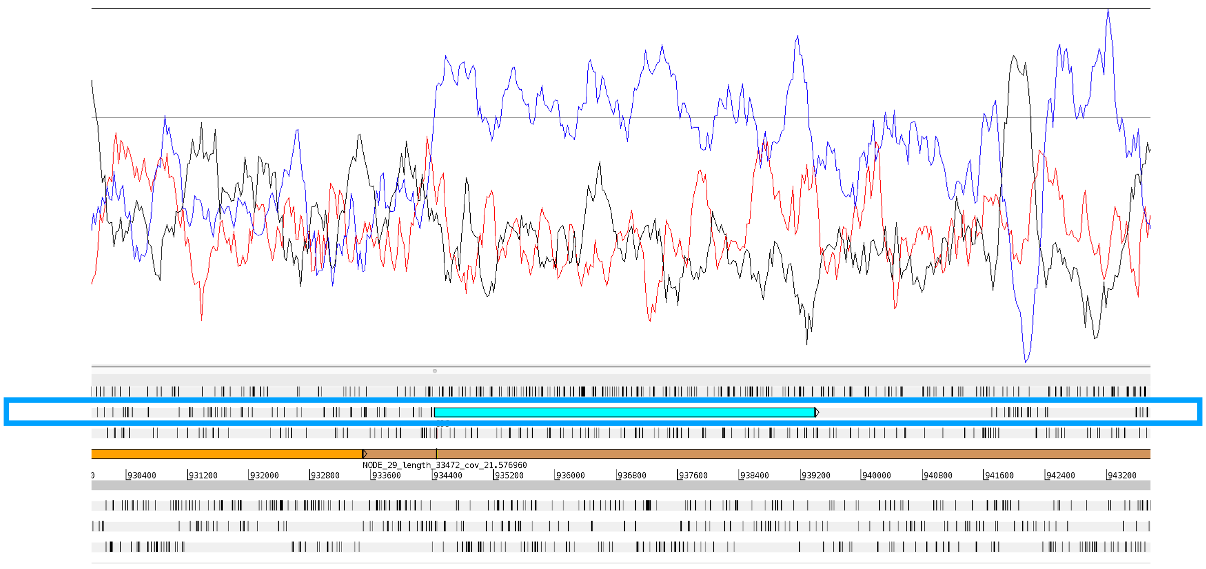
11. A second useful plotting feature is the correlation score, which can be found under

    Graph -> Correlation Scores

    and

    Graph -> Reverse Correlation Scores

    Turn off the GC plot pane and select the Correlation Scores plot. You should see a plot comprised of three coloured lines, with annotation in the top right hand corner. Each of these lines represent the three possible reading frames for the forward strand of your assembly.



    Each of the lines plot the correlation between the amino acid composition of globular proteins in TREMBL (a part of the UniProt reference database) and the composition of the base translation in each of the three reading frames. Putative gene sequences are associated with a clear increase in the correlation of one reading frame, as shown in the above figure.

    Identify two regions of your assembly where the correlation score suggests a gene may be present. Add annotation by highlighting the region (make sure to select the correct reading frame), right clicking and selecting

    Create -> Feature from base range

12. After creating these two features save their amino acids to file and start a new blast search using it. You will need to change the file names to avoid overwriting your existing results.

13. By now your first BLAST search should have finished. Return to the terminal and press Enter, you should see a message appear that looks something like

    [2]- Done. blastp -db nr -task blastp-fast…

    If you do not then type

    jobs

    To check if BLAST is still running. Assuming BLAST has finished examine your results file using less. As we have assembled a well studied organism your results are likely to have multiple hits that match the majority of your input sequences. Scan through the BLAST hits for the first gene in your list - from the information alone can you identify its likely function?

14. Now that we have some annotation information for our genes we need to add it in to our assembly. Return to the Artemis window, select one of the putative CDS that you ran a blast search on and right click on it. Using the menu that appears open the features editor using:

    Edit -> Selected features in editor

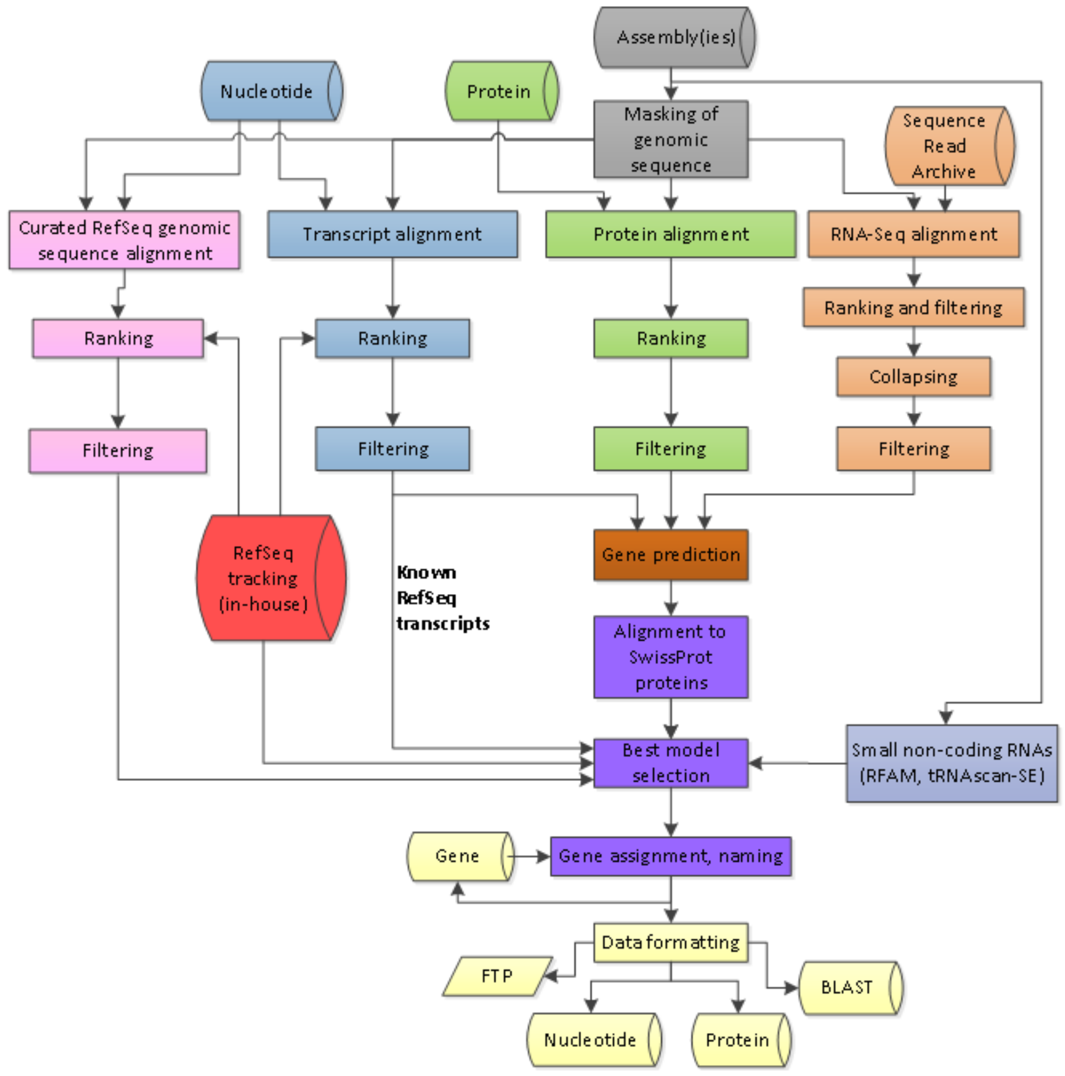
    A new window should appear with a text box that includes the line:

    /note="none"

    Replace this with the following lines:
    /gbkey="CDS"
    /product="*gene function from blast*"
    /inference="BLAST"
    /note="*Add in some details from your BLAST*"

    Replace the text in italics on the second line with information you obtained from your BLAST search, you can expand on this by adding additional details using the note line.

15. During the introduction we discussed the use of automated tools for annotating a genome. These tools utilise a range of approaches to predict gene models, identify coding sequences and generate draft annotation. Both NCBI and ENSEMBL maintain automated annotation pipelines that make use of their extensive databases. These pipelines are able to assemble raw sequencing data, align transcripts against the assembly, generate gene prediction models and output draft annotation files. For example, the NCBI Eukaryotic pipeline uses the following steps:



As these pipelines are able to access previously submitted data they are often able to characterise genes with high confidence and transfer annotation from existing reference sequences onto your assembly. There are a couple of downsides however. They require a large amount of processing power and access to large databases, preventing them from being run locally. Additionally, because they are designed to work with a large range of organisms they are also difficult to optimise if you are working with a novel or unknown species.

In these cases it may be better to use a specialised service and to build your own training models. This requires access to annotated sequences from related organisms, which are used to train the prediction algorithms of the software

Some of the most popular approaches are:
NCBI Prokaryotic pipeline - https://www.ncbi.nlm.nih.gov/genome/annotation_prok/
NCBI Eukaryotic pipeline - https://www.ncbi.nlm.nih.gov/genome/annotation_euk/
Augustus - https://github.com/Gaius-Augustus/Augustus
Companion - https://companion.sanger.ac.uk/
SNAP - https://github.com/KorfLab/SNAP
InterProScan - https://www.ebi.ac.uk/interpro/
BRAKER2 - https://github.com/Gaius-Augustus/BRAKER

For the majority of these pipelines the process takes too long to cover during this introductory course, however, InterProScan can be run relatively quickly. InterProScan is used to classify and characterise protein sequences by comparing them to the models built from the ENSEMBL databases. It is capable of predicting both high level structural classifications, such as the presence of alpha-helices or transmembrane domains and low level details such as the function of individual protein domains. Its speed is achieved through its use of a pre-calculated lookup service, if a near identical sequence already exists within the ENSEMBL database the software simply downloads the existing results rather than repeating the entire analysis from scratch.

We are going to run InterProScan on using the ORF1000-v2 file that you created to try and classify the genes that you extracted. The command is:

```
interproscan -goterms -dra -f GFF3 -i ORF1000-v2 -b ORF1000_IPS
```

This will take a couple of minutes and will generate an output file called ORF1000_IPS.gff3. Using less examine the file, based upon the entries can you determine the likely functions of your genes?

In an ideal world we would load our new annotation into Artemis to view it alongside our assembly. Unfortunately the current version of InterProScan outputs the results into a format that Artemis cannot read properly. This is a frequent problem with bioinformatics, one tool or file format is updated while another is not, creating incompatibility between them. It is possible to convert the annotation file but the process is tricky and requires some time. We will instead cheat, by examining the annotation file for our reference sequence.

16. Quit artemis and then reopen it using the following command:

    art Ecoli_ref1.fa + Ecoli_ref1_annotation.gff3 &

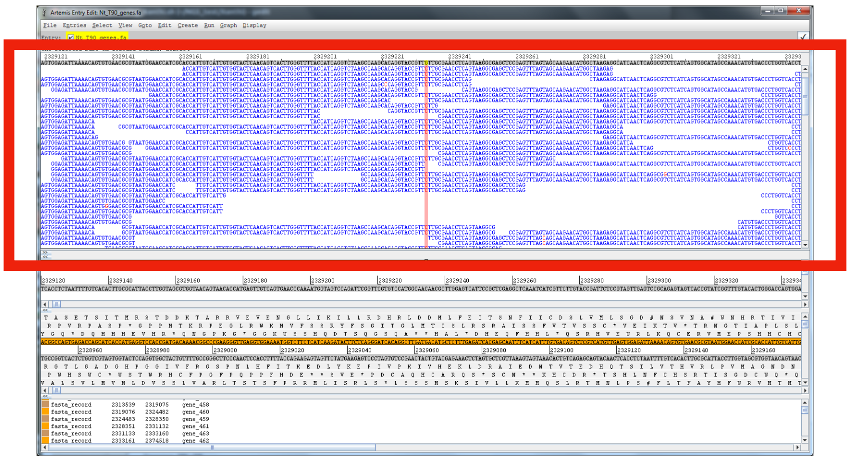
    which will load in both the reference sequence and its annotation at the same time. You will get a warning message when you do this, it is up to you if you wish to view the warnings.

    As *E. coli* has been extensively studied the annotation is very detailed, spend a couple of minutes examining some of the features to get a feel for what a fully annotated genome should look like.

17. There is one additional way of identifying coding sequences that we have not yet looked at - RNA-Seq. We are going to cover this technique properly later in the course but essentially this technique relies upon our ability to sequence RNA and then map it back to a genome assembly. As we know these sequences were expressed from the genome we can use it to identify and annotate transcribed features. Within your folder you should already have a pre-generated BAM file called Ecoli.sorted.bam. To load it into Artemis click

    File -> Read BAM / CRAM / VCF

    And then select the file. You should see a new panel appear in Artemis that looks like the one highlighted in red:



    This is the bam view and it details the position that each read maps to on our genome. As these reads came from RNA this means that that section of the genome is transcribed. Scan through the genome, can you identify some genes that are highly transcribed and some for which there appears to be no expression? What are the functions of those genes?