

Module 6

Genome Assembly and Analysis

Introduction

One of the greatest challenges of sequencing a genome is determining how to arrange sequencing reads into chromosomes and plasmids. This process of determining how the reads fit together by looking for overlaps between them is called **genome assembly**. In this module we concentrate on genome assembly using Illumina sequence platform data, although the techniques you will learn are applicable to other technologies (e.g. 454 GS FLX and ABI SOLiD). Current Illumina machines can produce around 600 Gigabases of sequence data in ten days. This is the equivalent of around 300 human genomes or 300,000 bacterial genomes!

The data from the Illumina machine comes as relatively short stretches (35-150 base pairs) of DNA – around 6 billion of them. These individual sequences are called **sequencing reads**. There are a range of assembly programs that have been specifically designed to assemble genomes from next generation sequence (NGS) data. Genome assembly using sequence reads of around 100bp is complicated due to the high frequency of repeats longer than the sequence read length in genomes, for example: IS elements, rRNA operons; and the massive amount of data the programs have to handle. In addition to finding overlaps in the sequence, the assembly programs can also use information from the predicted insert size where paired reads are used, to link and position reads in an assembly.

Where a genome is piecing together without any reference sequence to compare it to, or scaffold it against, is termed a *de novo* assembly. Due to the previously mentioned challenges of assembly, *de novo* assembly will not produce complete genomes, but will be fragmented into multiple contiguous sequences (**contigs**), the order of which is arbitrary, and does not necessarily bear any relation to their real order in the genome.

Where a closely related reference sequence is available, it is possible to improve an assembly by ordering the contigs in comparison to the reference, and also transferring annotation. In this case, nearly all of the genome will be present, i.e. genes and features, but there will be some regions that will contain gaps, or contigs that will not be accurately placed, because they are not present in the reference used. Although technically incomplete, ordered genome assemblies can provide valuable insights into the genetics and biology of an organism.

The aims of this exercise are:

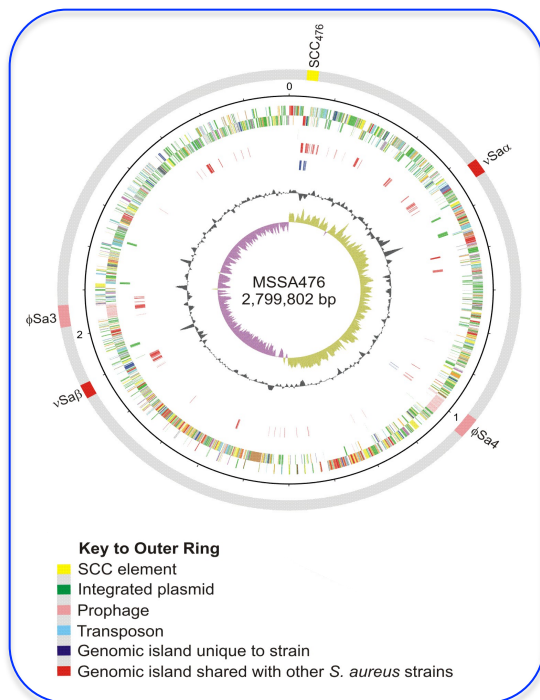
- i) To show how **Next Generation Sequencing data** can be assembled into a draft genome.
- ii) Order the draft genome against a reference sequence, and **transfer annotation** from the reference to the draft genome
- iii) To show how, using **comparative genomics**, regions of difference that distinguish genomes can be identified and analysed.

Background

Staphylococcus aureus is a bacterial pathogen that has gained notoriety in recent years due to its ability to evolve new virulent and drug resistant variants. In particular, the spread of *S. aureus* in hospitals has placed an increased burden on health care systems; *S. aureus* is the most common cause of hospital-acquired infection. Accompanying the spread of this bacterium has been an increase in the resistance to antibiotics. In parts of Europe, the US and Japan, 40-60% of all hospital *S. aureus* are now resistant to the β -lactam antibiotic methicillin. Methicillin resistant *S. aureus* (MRSA) strains were first described in the 1960s and successful clones of MRSA have spread round the globe.

The example

In this module we will assemble the genome of a strain of *S. aureus*, 16B, that was sequenced as part of an MRSA outbreak investigation, (Köser *et al.*, 2012, N Engl J Med. 366:2267-75). Using multi locus sequence typing (MLST) the isolate was identified as belonging to sequence type 1 (ST1), a lineage of *S. aureus* that is more frequently associated with infections in the community rather than in hospitals, and tends to be less resistant to antibiotics than the *S. aureus* commonly associated with hospital-acquired infection.



The exercises

We are going to generate a 16B assembly, and compare it to the chromosomes of 2 other ST1 isolates: MSSA476, which was isolated in the UK (Holden *et al.*, 2004, PNAS. 101:9786-91), and MW2, which was isolated in the USA (Baba *et al.*, 2002, Lancet 359:1819-27). Both MSSA476 and MW2 have been completely sequenced, annotated and deposited in EMBL. The MSSA476 genome consists of a 2.8 Mb chromosome (left) and a 20.6 kb plasmid, pSAS. We will use the MSSA476 chromosome to order the 16B assembly, and also transfer annotation to it. In order to check the assembly for misassemblies and copy number variants we will map the 16B sequence reads back to the ordered 16B assembly.

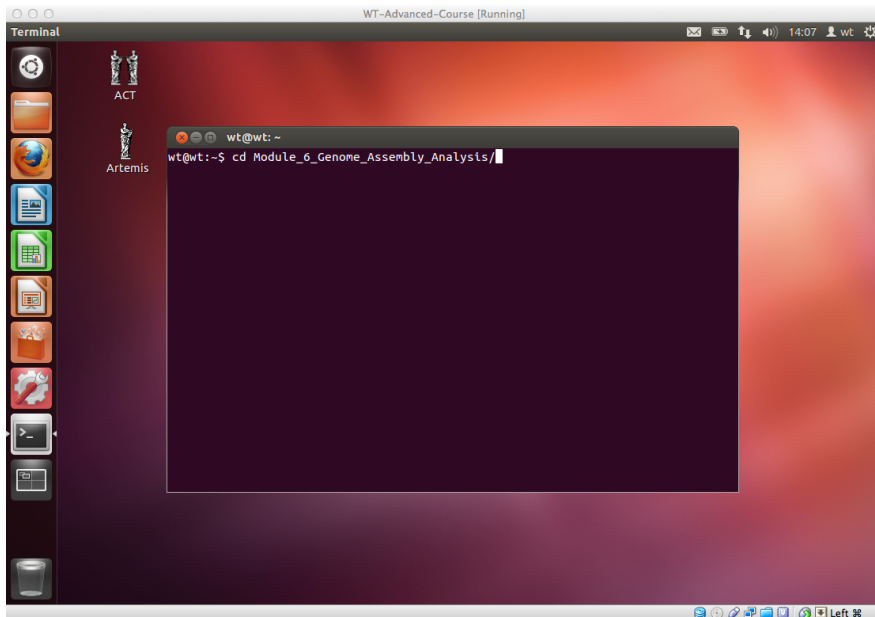
The research question

The three ST1 isolates are closely related but exhibit different antibiotic resistance profiles: 16B is resistant to penicillin, fusidic acid, methicillin and erythromycin; MSSA476 is resistant to penicillin and fusidic acid; and MW2 is resistant to penicillin and methicillin.

Using a comparative genomic approach we will identify regions of difference, and investigate the genetic basis of the antibiotic resistance in 16B, and genetic mechanisms that driving the evolution of resistance.

A: Generating a *de novo* assembly

To begin the exercise we need to open up a terminal window just like the one you used for the UNIX Module. We will then need to move into the 'Module_6_Genome_Assembly_Analysis' directory using the UNIX command **cd**



To generate the *de novo* assembly you are going to use a assembly package called Velvet (Zurbino *et al.*, 2008, Genome Res. 8:821-9); other assembly programs are available, e.g. SOAP (soap.genomics.org.cn) and ABySS (Simpson *et al.*, 2009, ABySS: a parallel assembler for short read sequence data. Genome Research, 19:1117-2). Velvet takes in short read sequences, removes errors then produces high quality unique contigs. It then uses paired-end read and long read information, when available, to retrieve the repeated areas between contigs.

The algorithm at the heart of Velvet is based on de Bruijn graphs (a mathematical structure used to model relationships between objects). When doing assembly with short reads the first step is to find all the possible overlaps between all the reads. One efficient way is to look for k-mers (words/nucleotide patterns of a specific length) in each read. If two reads contain the same k-mer they might also overlap. Each read contains several k-mers, and k-mers from the same read can be connected in a graph. Velvet represents overlaps between k-mers in a de Bruijn graph. By simplifying the graph Velvet can try to generate connected sequences, where k-mers in the graph are connected and thereby it is able to piece together sequences and generate contigs.

In this module we are not going to explore the options available in Velvet, but are going to run it with basic parameters. If you would like to know more about theory behind Velvet, or the various options, see the Velvet web site (www.ebi.ac.uk/~zerbino/velvet/).

To perform the assembly you are going to run a series of commands that you will type on the command line. Make sure that you type the commands carefully as UNIX is case sensitive.

First of all, 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 files in the resulting list.

```
wt@wt: ~/Module_6_Genome_Assembly_Analysis
wt@wt:~/Module_6_Genome_Assembly_Analysis$ cd Module_6_Genome_Assembly_Analysis/
wt@wt:~/Module_6_Genome_Assembly_Analysis$ ls
16B_1.fastq  emb1      MSSA476.dna  MSSA476_MGEs.tab  MW2.embl
16B_2.fastq  Extra_Files  MSSA476.embl  MW2.dna           MW2_MGEs.tab
```

The forward and reverse reads for the isolate 16B (16B_1.fastq and 16B_2.fastq) were generated using an Illumina HiSeq machine and are 75bp paired-end reads.

The Velvet package contains two programs: **velveth** and **velvetg**, which are run in succession to generate the assembly. **velveth** helps you construct the dataset for the following program, **velvetg**, and indicates to the system what each sequence file represents. **velvetg** is the core of Velvet where the de Bruijn graph is built then manipulated, and which ultimately produces the assembly that we are interested in.

The first program of the velvet package we are going to use is **velveth**.

At the prompt type and return the command line:

```
velveth S_aureus_16B.49 49 -shortPaired -fastq -separate
16B_1.fastq 16B_2.fastq
```

S_aureus_16B.49 - is the directory into which results are written

49 - is the the k-mer value we are using (i.e. 49 nucleotides). Other k-mers can be used and can alter the performance of assembly, however for this module we will run it with a value of 49 which will perform adequately.

-shortPaired -fastq -separate - tells the program that the input sequence data is forward and reverse short paired reads in fastq format in separate files

16B_1.fastq 16B_2.fastq - forward and reverse fastq files

```

wt@wt: ~/Module_6_Genome_Assembly_Analysis
wt@wt:~/Module_6_Genome_Assembly_Analysis$ velveth S_aureus.49 49 -shortPaired -
fastq -separate 16B_1.fastq 16B_2.fastq
[0.000020] Reading FastQ file 16B_1.fastq;
[0.019747] Reading FastQ file 16B_2.fastq;
[46.635256] 4000000 sequences found in total in the paired sequence files
[46.637369] Done
[46.637429] Reading read set file S_aureus.49/Sequences;
[48.728650] 4000000 sequences found
[63.966501] Done
[63.967617] 4000000 sequences in total.
[63.968013] Writing into roadmap file S_aureus.49/Roadmaps...
[68.638575] Inputting sequences...
[68.639211] Inputting sequence 0 / 4000000
[88.153786] Inputting sequence 1000000 / 4000000
[105.023258] Inputting sequence 2000000 / 4000000
[123.308468] Inputting sequence 3000000 / 4000000
[141.210452] === Sequences loaded in 72.571919 s
[141.210952] Done inputting sequences
[141.211159] Destroying splay table
[141.353248] Splay table destroyed
wt@wt:~/Module_6_Genome_Assembly_Analysis$

```

The next program of the velvet package we are going to use is **velvetg**.

At the prompt type the following and press enter:

```
velvetg S_aureus_16B.49 -exp_cov auto -min_contig_lgth 200
-cov_cutoff auto -ins_length 350
```

S_aureus_16B.49 - is the directory in which velvetg can find the velveth output files which are necessary to run velvetg: Sequences and Roadmaps files

-exp_cov auto - allow the system to infer the expected coverage of unique regions

-min_contig_lgth 200 - minimum contig length of 200 bp exported to the output file

-cov_cutoff auto - allow the system to infer removal of low coverage nodes

-ins_length 350 - expected distance between two paired-end reads in the short-read dataset: 350 bp

```

wt@wt: ~/Module_6_Genome_Assembly_Analysis
[229.591042] === Nodes Scaffolded in 0.367209 s
[229.663480] Preparing to correct graph with cutoff 0.200000
[229.672253] Cleaning memory
[229.673531] Deactivating local correction settings
[229.673828] Pebble done.
[229.674509] Concatenation...
[229.892636] Renumbering nodes
[229.893349] Initial node count 633
[229.893627] Removed 383 null nodes
[229.893824] Concatenation over!
[229.894006] Removing reference contigs with coverage < 17.599285...
[229.894209] Concatenation...
[229.894411] Renumbering nodes
[229.894592] Initial node count 250
[229.894774] Removed 0 null nodes
[229.894957] Concatenation over!
[230.104953] Writing contigs into S_aureus.49/contigs.fa...
[233.774574] Writing into stats file S_aureus.49/stats.txt...
[233.801795] Writing into graph file S_aureus.49/LastGraph...
[236.431359] Estimated Coverage = 35.198570
[236.431656] Estimated Coverage cutoff = 17.599285
Final graph has 250 nodes and n50 of 225963, max 613961, total 2777979, using 38
47086/4000000 reads
wt@wt:~/Module_6_Genome_Assembly_Analysis$

```

There is a lot of output printed to the screen, but the most important is in the last line:

Final graph has 250 nodes and n50 of 225963, max 613961, total 2777676, using 3847086/4000000 reads. (Result might differ depending on the velvet version used).

This line first gives you a quick idea of the result. **250 nodes** (contigs) are in the final graph. An **n50 of 225963** means that 50% of the assembly is in contigs of at least 225963 bases. This **n50** parameter is most commonly used as an indicator of assembly quality. The higher, the better! **max** is the length of the longest contig. **total** is the size of the assembly, here it is 2.78 Mb. The last two numbers tell us how many reads were used from the 4 million pairs.

A typical *S. aureus* genome is 2.8 Mb in size, therefore the *de novo* assembly that we have produced should contain over 99% of this isolate's genome.

All of the results are written into the directory you specified, e.g. **S_aureus_16B.49**.

Use the UNIX **cd** command to move into this directory, and the **ls** command to look at the contents.

```

wt@wt: ~/Module_6_Genome_Assembly_Analysis/S_aureus.49
wt@wt:~/Module_6_Genome_Assembly_Analysis/S_aureus.49$ ls
contigs.fa Graph2 LastGraph Log PreGraph Roadmaps Sequences stats.txt

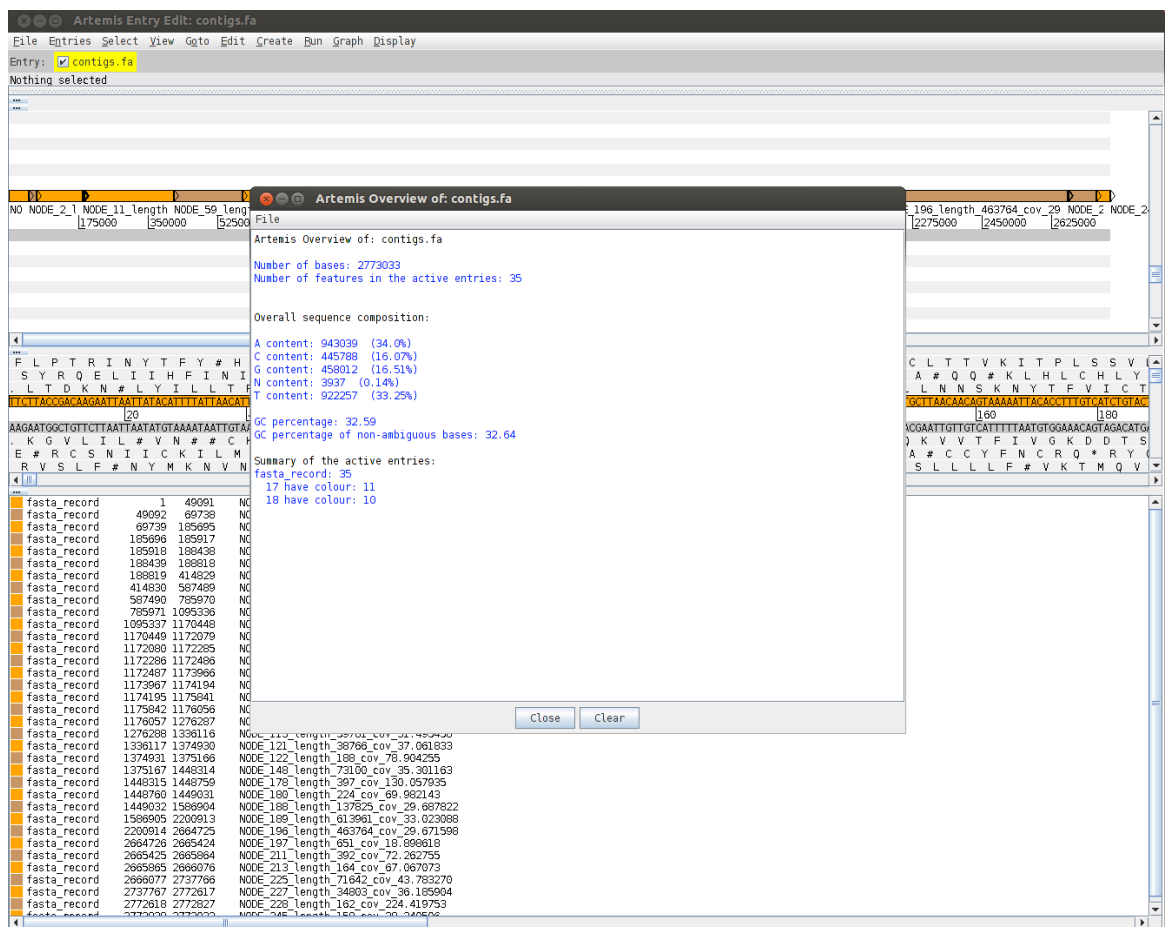
```

The final contigs are in **contigs.fa**. This file contains the contigs in multifasta format, i.e. the sequence of each contig is written as a separate fasta sequence, with all the contigs fasta sequences concatenated together. The **stats.txt** file holds some information about each contig, its length, the coverage, etc.. The other files contain information for the assembler.

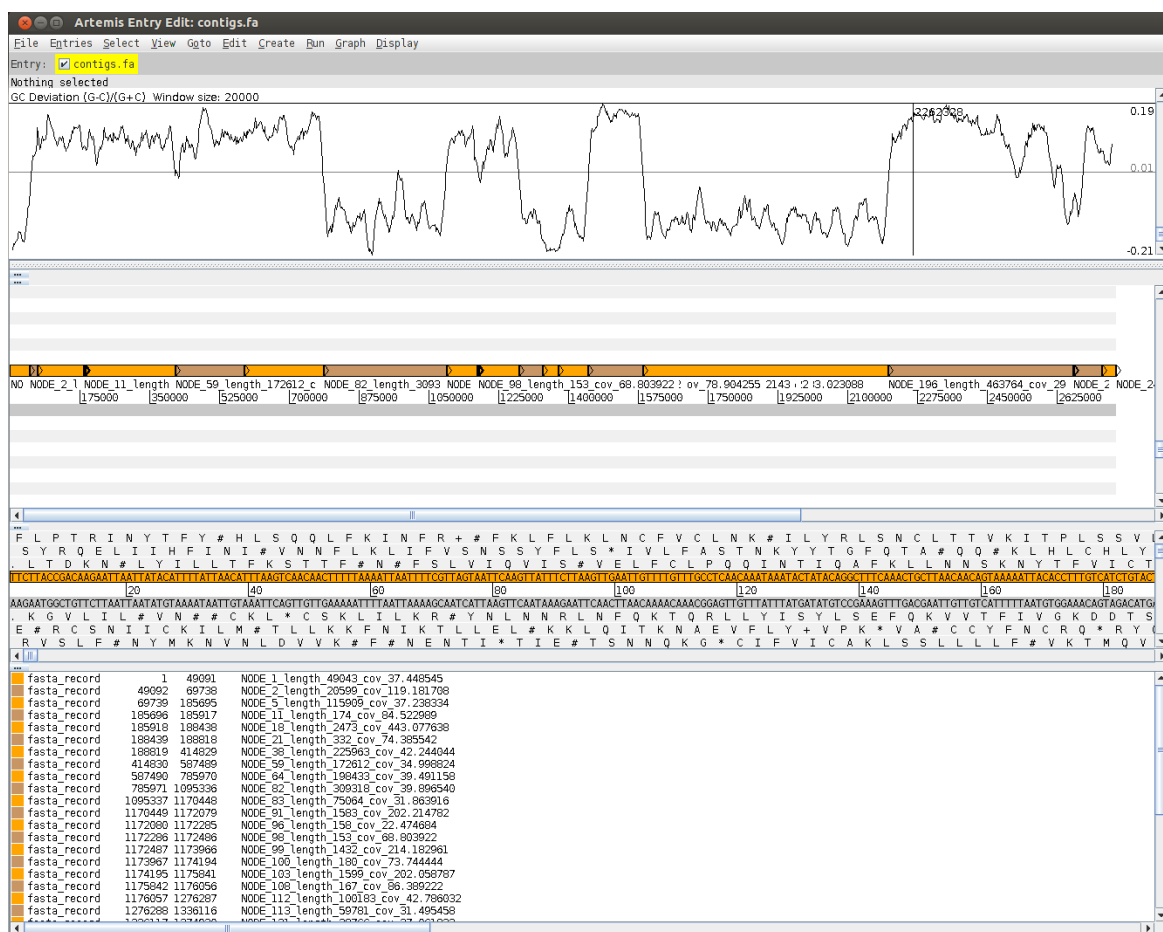
We are now going to look at the assembly in Artemis.

Double click on the Artemis Icon or type '**art &**' on the command line of your terminal window and press return. Once you see the initial Artemis window, open the contigs.fa file via **File, Open**.

Once open, zoom out so you can see the whole sequence in your window. The individual contigs in the multifasta file are alternately coloured orange and brown and displayed on the forward DNA line in the sequence view window. To look at a summary of the contigs.fa, click **View**, then **Overview**. You should see that there are 35 contigs in total (35 Number of features in active entry).



From the **Graph** menu, open **GC Deviation (G-C)/(G+C)** by clicking on the button next to it. Rescale the plot for to a more appropriate window size for this zoomed out view: Right click on the graph, and click **Maximum Window Size** , and select **20000**. Then move the graph slider of the right hand side of the screen down to the bottom of the bar.



From the graph you can see that plot generally varies about a upper level and a lower level across the assembly, with shifts occurring at contig boundaries. As you will have seen in the previous Module looking at the **GC Deviation** plot of the *Salmonella* Typhi sequence, there is a GC skew across the chromosome that is caused by a mutation basis that means that the leading strand of the replication fork is G and T rich, as opposed to the lagging strand, which is C and A rich. If you look at the circular diagram on page 2 of this Module, you can see the GC skew for the MSSA476 chromosome (the purple and olive inner plot on the figure). The origin and the terminus of replications are approximately half way round the chromosome (the origin is at the top, and the terminus is at the bottom), therefore there is a strong signal of GC deviation between these sites, i.e. as you move round the chromosome the GC Deviation plot will be either be at a high or low level, and after the origin or terminus of replication, the plot will shift to converse level.

Looking at the GC Deviation plot in Artemis of the 16B assembly you can see there are multiple shifts from high to low indicating that the contigs in the assembly as displayed, are not in the correct order and orientation relative to the true origin and terminus of replication of the 16B chromosome.

B: Ordering the assembly against a reference chromosome

At the Wellcome Trust Sanger Institute we have developed a tool called ABACAS (Assefa *et al.*, 2009) to order contigs against a reference sequence. Any spaces between the contigs (gaps) can be filled in with “N” characters to ‘pad’ the sequence with equivalent sized regions to those on the reference that may be missing in the assembly. The result is called a pseudo-molecule. This can be loaded into ACT along with the reference sequence and then be analyzed.

The sequence we are going to use as a reference belongs to an ST1 MSSA strain, MSSA476 (EMBL accession number BX571857). Before we begin, make sure you are back in the Module 6 directory. To check where you are use the UNIX **pwd** command. If you were in the `S_aureus.49` directory, use the **cd ..** command to move into the directory above.

At the prompt type and return the command line:

```
abacas.1.3.1.pl -r MSSA476.dna -q S_aureus.49/contigs.fa -p  
nucmer -b -d -a -c -o 16B.ordered
```

- r** - reference sequence in a single fasta file (**MSSA476.dna**)
- q** - contigs in multi-fasta format (**contig.fas** file in the **S_aureus.49** directory)
- p** - MUMmer program to use: **nucmer** (nucleotide-nucleotide comparison)
- d** - use default nucmer parameters, which is in this case is faster
- b** - generate a bin of contigs that don't map. This is very important
- a** - append contigs in bin to the pseudo-molecule
- o** - prefix for the output file name (**16B.ordered**)
- c** - reference sequence is circular

To see a complete list the option available you can type the command:

```
abacas.1.3.1.pl -h
```

Once ABACAS is done, it indicates which files it generated and how to load them into Act:

```
FINISHED CONTIG ORDERING

To view your results in ACT
    Sequence file 1: MSSA476.dna
    Comparison file 1: 16B.ordered.crunch
    Sequence file 2: 16B.ordered.fasta

    ACT feature file is: 16B.ordered.tab

    Contigs bin file is: 16B.ordered.bin

    Gaps in pseudomolecule are in: 16B.ordered.gaps
```

Before opening the files in ACT, we are going to generate a BLASTN comparison file, rather than using the comparison file that ABACAS generates. This is because the ABACUS generated file is based on MUMMER and just aligns the contigs and does not report smaller matches within contigs.

Previously in Module 3 we generated the comparison rules using a web site, WebACT. This time you are going to do it yourself using the locally installed version of BLAST. We will run two programs: **formatdb**, which formats one of the sequences as a BLAST database; and the other **blastall**, runs the BLAST comparison.

At the prompt type and return the command line:

```
formatdb -p F -i MSSA476.dna
```

-p - is the sequence protein (True or False). Ours is DNA sequence therefore we use **F**
-i - input sequence to format (**MSSA476.dna**)

Next type and return the command line:

```
blastall -p blastn -m 8 -d MSSA476.dna -i 16B.ordered.fasta -o  
MSSA476.dna_vs_16B.ordered.fasta
```

-p - BLAST program to use (**blastn** = nucleotide blast)
-m - alignment output type (**8**, one line per entry)
-d - database file (**MSSA476.dna**). This must be the file used for the formatdb command
-i - query File (**16B.ordered.fasta**)
-o - output file name (**MSSA476.dna_vs_16B.ordered.fasta**)

We are now going to look at the Abacus ordered 16B assembly in ACT with the BLASTN comparison file we have just generated.

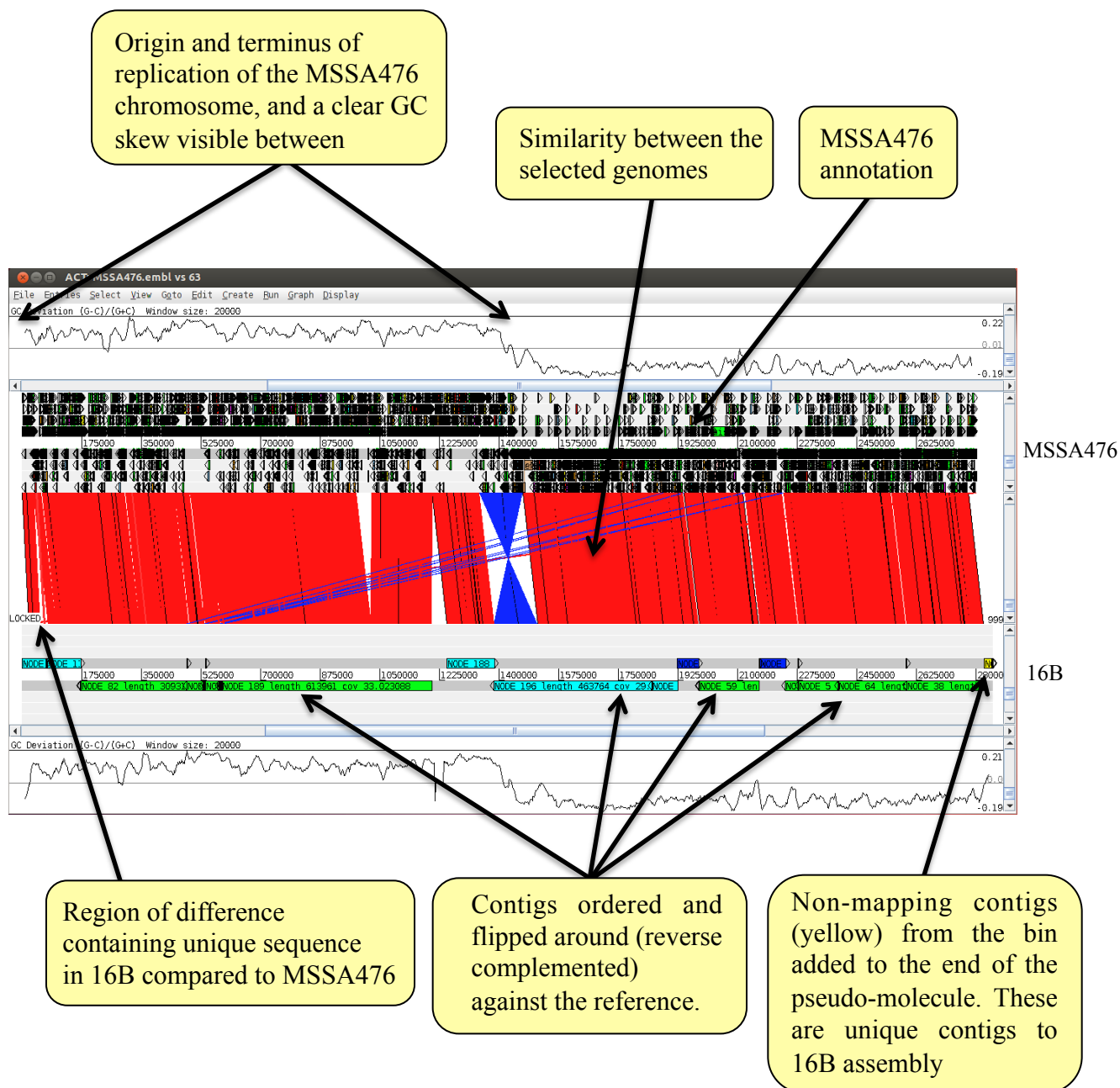
At the prompt type and return the command line:

```
act MSSA476.embl MSSA476.dna_vs_16B.ordered.fasta <(cat  
16B.ordered.tab 16B.ordered.fasta) &
```

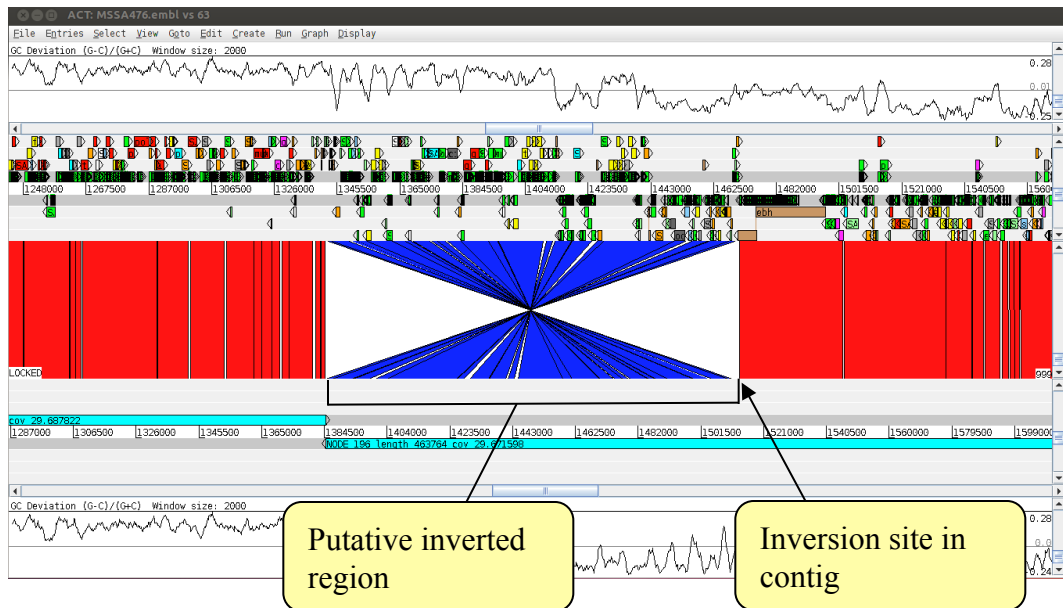
The command **<(cat 16B.ordered.tab 16B.ordered.fasta)** loads both the tab file containing contig features and fasta file at the same time

Once ACT has opened, zoom out so you can see the whole of the sequences (you may have to re-size the ACT window) and reduce the size of the BLASTN footprint that is displayed, by moving the slider on the right-hand side of the comparison window down to the bottom of the bar.

As before, display the GC Deviation (G-C)/(G+C) plots for both of the sequences (under the **Graph** menu there will be two sequences, top and bottom sequences, click on each to open the graphs for each). Remember to rescale the plot for a more appropriate window size (use 20000 as before, then move the graph slider of the right hand side of the screen down to the bottom of the bar).



In the ACT figure there are several regions of interest that are worth investigating. The first region we are going to look at is the inverted region in the centre of the assembly that is covered by the hourglass shaped blue matches in the comparison panel. This 130 kb region spans the terminus of replication region, and is present at one end of a contig. At the other end of the putative inverted region there is a contig break. In order to check if this is a real inversion or a mis-assembly produced by Velvet, we are going to map the 16B sequence reads back to 16B assembly we have just created, to see if there are reads that span inversion point within the contig.



C: Mapping reads back to the ordered assembly

In this next exercise you are going to use the same mapping method as you did in Module 4, to map the 16B strain forward and reverse reads (16B_1.fastq and 16B_2.fastq) against the pseudo-molecule that you created using ABACAS (16B.ordered.fasta). Can you think of a quick way to do this?

At the prompt type and return the command line:

```
bwa index 16B.ordered.fasta
```

Once this is finished next type and return the command line:

```
bwa aln -q 15 16B.ordered.fasta 16B_1.fastq > F.sai
```

Once this is finished next type and return the command line:

```
bwa aln -q 15 16B.ordered.fasta 16B_2.fastq > R.sai
```

Once this is finished next type and return the command line:

```
bwa sampe 16B.ordered.fasta F.sai R.sai 16B_1.fastq  
16B_2.fastq > mapping.sam
```

Once this is finished next type and return the command line:

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

Once this is finished next type and return the command line:

```
samtools sort mapping.bam 16B
```

Once this is finished next type and return the command line:

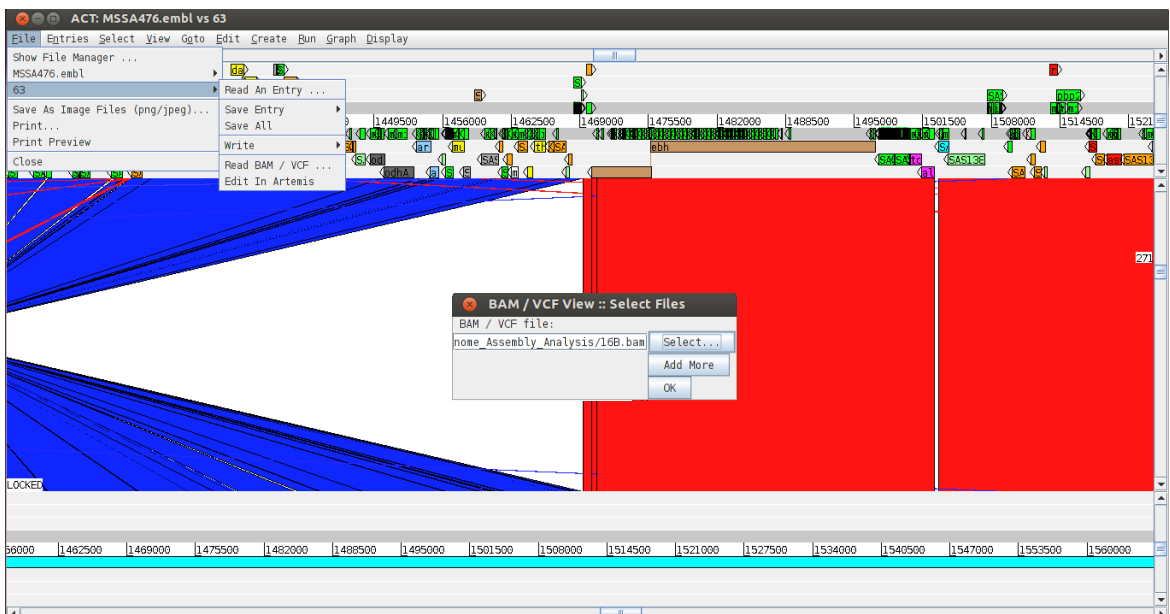
```
samtools index 16B.bam
```

Once this has finished you will now have a BAM file that you can load up into ACT.

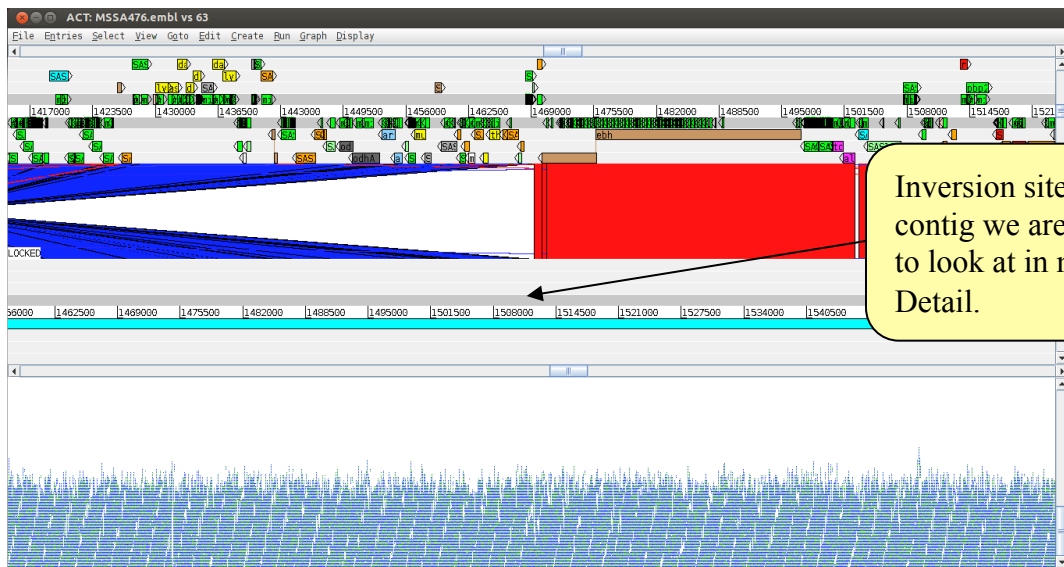
If you closed ACT before running the mapping, you can open it as before: **act
MSSA476.embl MSSA476.dna_vs_16B.ordered.fasta <(cat
16B.ordered.tab 16B.ordered.fasta) &**

To load the BAM file into ACT, click **File** on the menu and then click the **63** entry, and then the **Read BAM / VCF**.

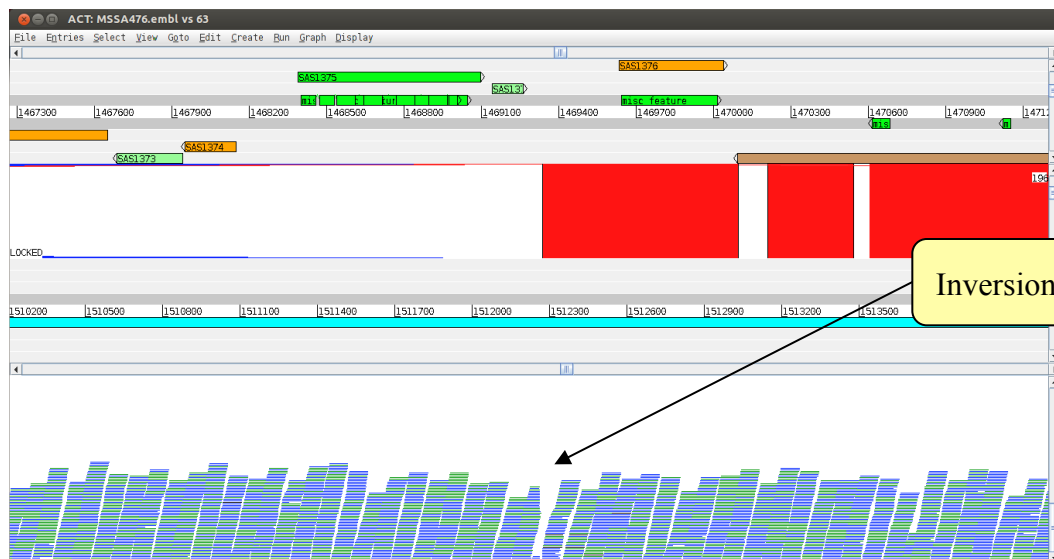
In the pop-up box click **Select**, select the **16B.bam** file, click **Open**, then click **OK**.



If you are not already there, go to the inversion region, and the inversion point in the contig (the region below illustrated in the image). You should see the BAM view as a panel at the bottom of the screen, rather than at the top of the screen as you will have done previously in Artemis. This is because we are looking at the bottom sequence, 16B.ordered.fasta.

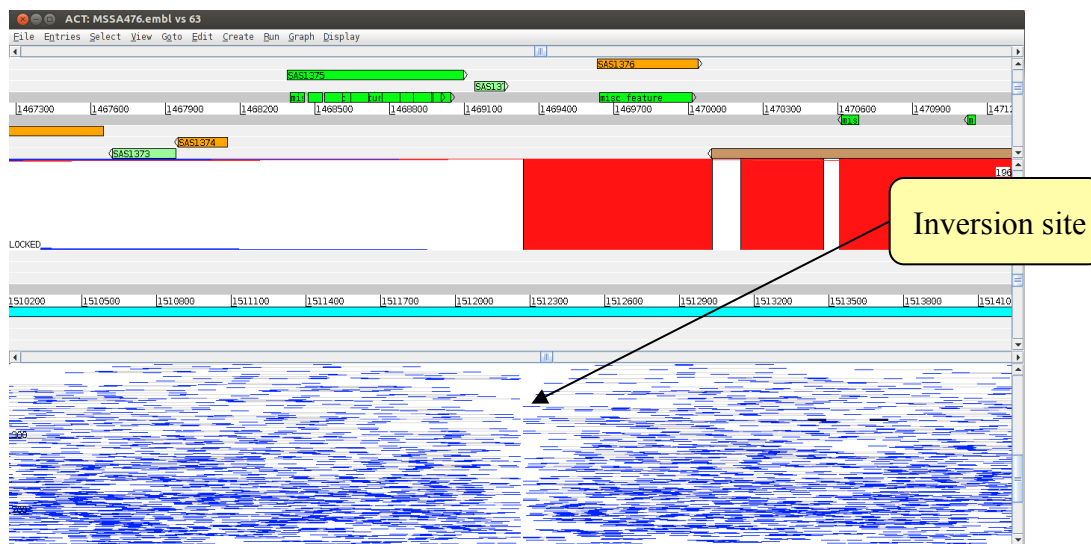


Zoom in further keeping the inversion site in the centre of the ACT screen.



The reads in the BAM view appear to break at the junction of the inversion indicated by the BLASTN match; no reads span the junction point (click on the reads around the junction to see their pair) suggesting that there may be problems with the assembly of the 16B DNA across this region.

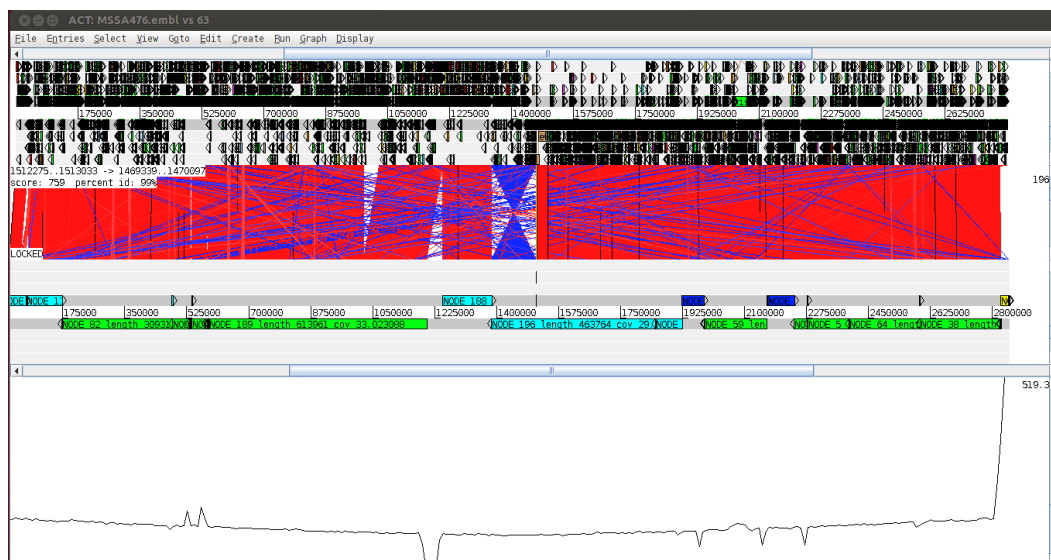
To get another perspective of the mapping to this region, change the BAM view to show the inferred size of the insert. To do this right click on the BAM view window, move the cursor over **Views**, and click **Inferred Size**.



From the inferred size view you can see that there are no reads with predicted inserts that span this region. This suggests that the inversion may not be present, and that the sequence generated by Velvet in this region has not assembled correctly, and needs further investigation. To check if this is a misassembly, you could change the parameters of the original Velvet runs, or alternatively design PCR primers and do a PCR to check for the orientation of this region in the genomic DNA.

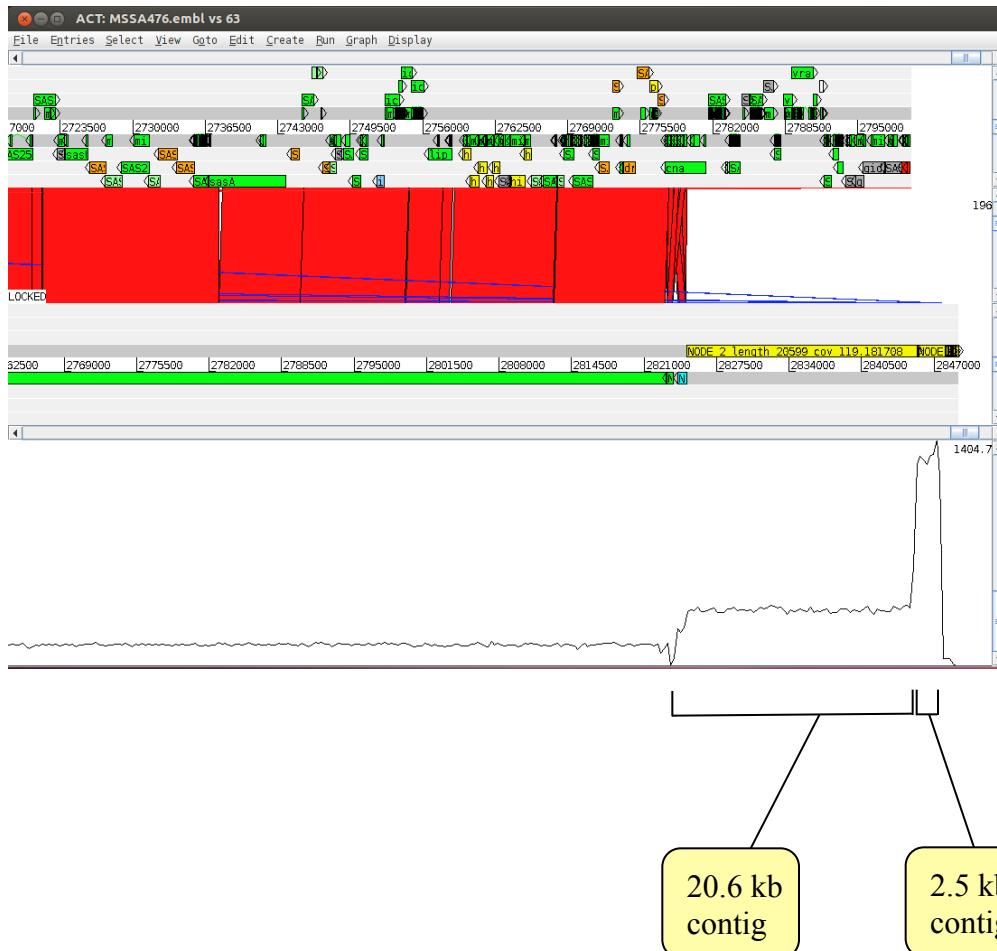
In addition to allowing us to check for potential mis-assemblies we can also use the mapping data to look for copy number variants in the assembly, as you did in Module 4.

In ACT change the read view back to Stack view, and zoom out to see the whole sequence



From this view of you can see that the average coverage across the whole 16B sequence is about 120 fold, and that there is subtle reduction in coverage from the origin to the terminus of replication. You can also see that the non-mapping sequences from the bin at the right-hand side of the sequence have a higher level of coverage than the rest of the sequence that matches to the MSSA476 chromosome.

Zoom into this region to look in more detail.



The non-mapping contigs are indicated by the yellow features. There are 7 contigs and the two larger sequences are 20.6 kb and 2.5 kb. The read coverage across these regions increases considerably from the average (120 fold), to about 400 fold for the 20.6 kb contig, and 1400 fold for the 2.5 kb contig. It is therefore likely that these two contigs are separate multicopy plasmids that are part of the 16B genome.

D: Annotation transfer

Now we have the contigs ordered against the reference, and have mapped back the reads to identify a possible misassembly, and also identified putative plasmid sequences. However we are still not yet in a position to drill down into the biology of the strain. For this we need to add some annotation to the newly assembled genome. To do this we can transfer the annotation of reference strain we used in ABACAS, as this has been annotated and is clearly highly related. We have developed a tool called RATT (Otto *et al.*, 2011, Nucleic Acids Res 39:e57) that can do this.

In the first step the similarity between the two sequences is determined and a synteny map is constructed. This map is used to map the annotation of the reference onto the new sequence. In a second step, it tries to correct gene models. One advantage of RATT is that the complete annotation is transferred, including descriptions. Thus careful manual annotation from the reference becomes available in the newly sequenced genome. Obviously, where no synteny exists, no transfer can be done. Let's see if this will work for our assembly.

As input we use the reference genome's annotation (the MSSA476 genome consists of a single chromosome and plasmid therefore are going to use them both) and the output of abacas (16B.ordered.fasta).

At the prompt type and return the command line:

```
start.ratt.sh embl 16B.ordered.fasta 16B Assembly > out.ratt.txt
```

embl – directory which contains all the EMBL files to be transferred (EMBL files for the MSSA476 chromosome and plasmid, pSAS)

16B.ordered.fasta – multifasta file to which the annotation will be mapped

16B – prefix to give results files

Assembly – Transfer type: Assembly, transfer between different assemblies

out.ratt.txt – output summary file

For a full list of RATT options type and return:

```
start.ratt.sh -h
```

RATT generates a lot of output, such as synteny block information, which genes were corrected, and most importantly how many genes were transferred. A summary of this is in the file **out.ratt.txt**.

To take a look at the contents of this file type at prompt and return the command line:

more out.ratt.txt

```
wt@wt: ~/Module_6_Genome_Assembly_Analysis
wt@wt:~$ cd Module_6_Genome_Assembly_Analysis/
wt@wt:~/Module_6_Genome_Assembly_Analysis$ more out.ratt.txt
Of the reference chromosome MSSA476      7.56 per cent  has no synteny with the query
Of the reference chromosome MSSA476pSAS 11.39 per cent  has no synteny with the query
Of the query chromosome ordered_staph-55e08.q2c2068      8.51 per cent  has no synteny with the reference
Nucmer is done. Now transfer the annotation.
perl /usr/local/ratt/main.ratt.pl embl nucmer.16B.snp nucmer.16B.filter.coords 16B
working on MSSA476
working on MSSA476pSAS
Overview of transference of annotation elements:
7689  elements found.
7111  Elements were transferred.
0     Elements could be transferred partially.
0     Elements split.
755   Parts of elements (i.e.exons tRNA) not transferred.
578   Elements couldn't be transferred.

CDS:
2643  Gene models to transfer.
2428  Gene models transferred correctly.
0     Gene models partially transferred.
0     Exons not transferred from partial CDS matches.
215   Gene models not transferred.

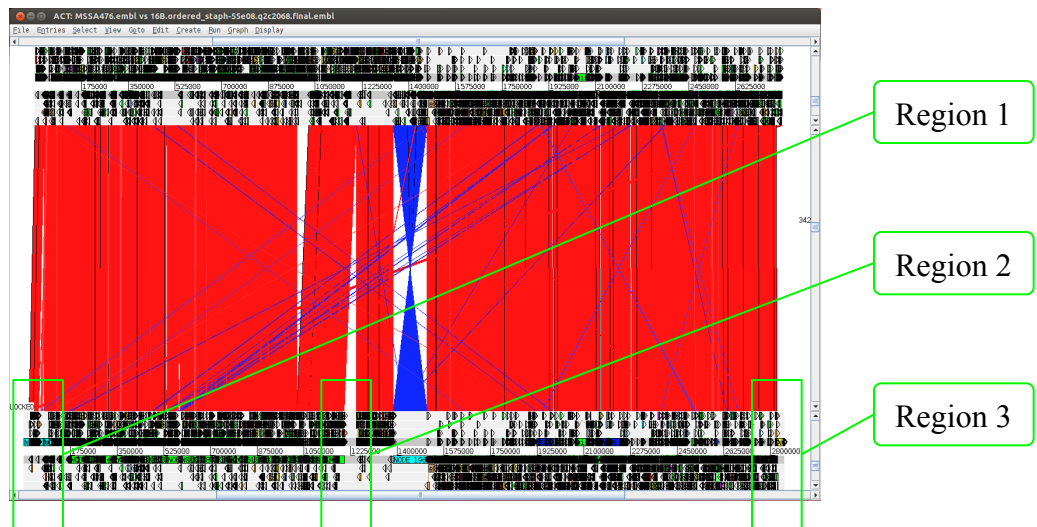
1 Sequences where generated
Done.
Nucmer is done. Now Correct the annotation for chromosome .
work on ordered_staph-55e08.q2c2068
***** Correction *****
Using the default specifications for start/codons and splice sites.
done.
Please see the file 16B.ordered_staph-55e08.q2c2068.Report.txt for reporting the errors and the file 16B.ordered_staph-55e08
q2c2068.Report.gff for reporting the error in Artemis or other sequence viewer.
*****
If you want to start artemis on this replicon:
art 16B.ordered_staph-55e08.q2c2068.final.embl + 16B.ordered_staph-55e08.q2c2068.Report.gff + Query/16B.ordered_staph-55e08
q2c2068.Mutations.gff
wt@wt:~/Module_6_Genome_Assembly_Analysis$
```

RATT produces an EMBL format file containing the assembly and transferred annotation ending in the suffix **.final.embl** (e.g. **16B.ordered_staph-55e08.q2c2068.final.embl**)

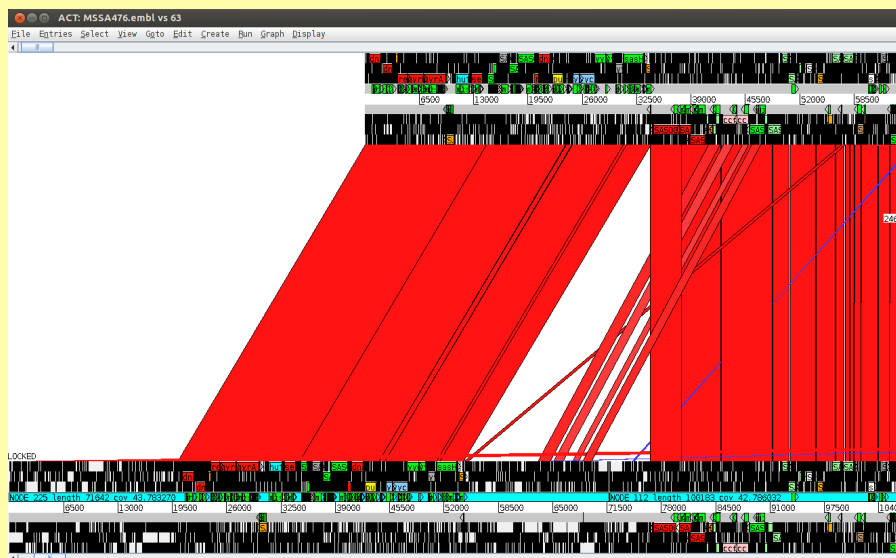
Load this up into ACT with the MSS476 reference chromosome. At the prompt type and return the command line:

**act MSSA476.embl MSSA476.dna_vs_16B.ordered.fasta <(cat
16B.ordered.tab 16B.ordered_staph-55e08.q2c2068.final.embl) &**

RATT has transferred 2432 gene features to the reference, and if you look in ACT you will see that most of the 16B assembly now has annotation. There are a few regions that do not have annotation, and these mainly coincide with regions that do not share DNA-DNA with the reference. We will quickly have a look at these regions.



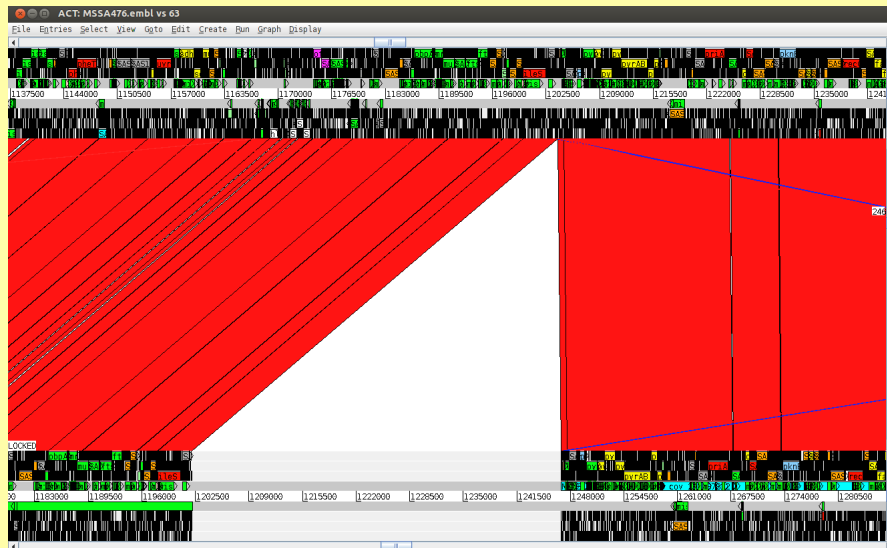
Region 1



In this region near at the left hand side of the reference chromosome and near the origin of replication you can see that there are two regions without annotation transferred. The first is at the very end of the assembly. If you look this region, it matches to DNA in the reference chromosome. This contig spans the origin of replication and therefore matches two separate regions of the reference (left and right ends of the M55A476 chromosome), therefore RATT has failed transfer annotation to the whole of this contig because it has effectively been split and separated in comparison to the reference.

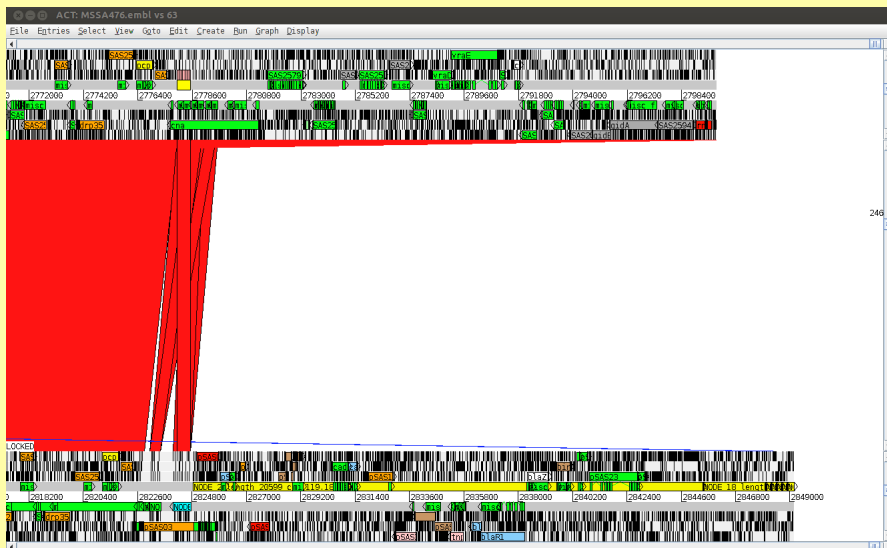
The second region lacking annotation spans two contigs. This ~22 kb region, contains BLASTN hits in the middle of the sequence, that match sequence in the M55A476 reference (top) that is also present in the 16B assembly (positions 85000 to 90000). This suggest that the ~22 kb region shares some similarity with the region downstream.

Region 2



From the ACT figure it would appear that there is a large insert in the 16B assembly relative to the MSSA476. If you zoom in and look at the sequence you will see that is composed of Ns rather than bases (in the figure you can make out regions with Ns, as they do not have any black lines that indicate stop codons on the forward and reverse translations). In this case ABACAS has mis-predicted a gap in this region, and therefore RATT has not transferred annotation.

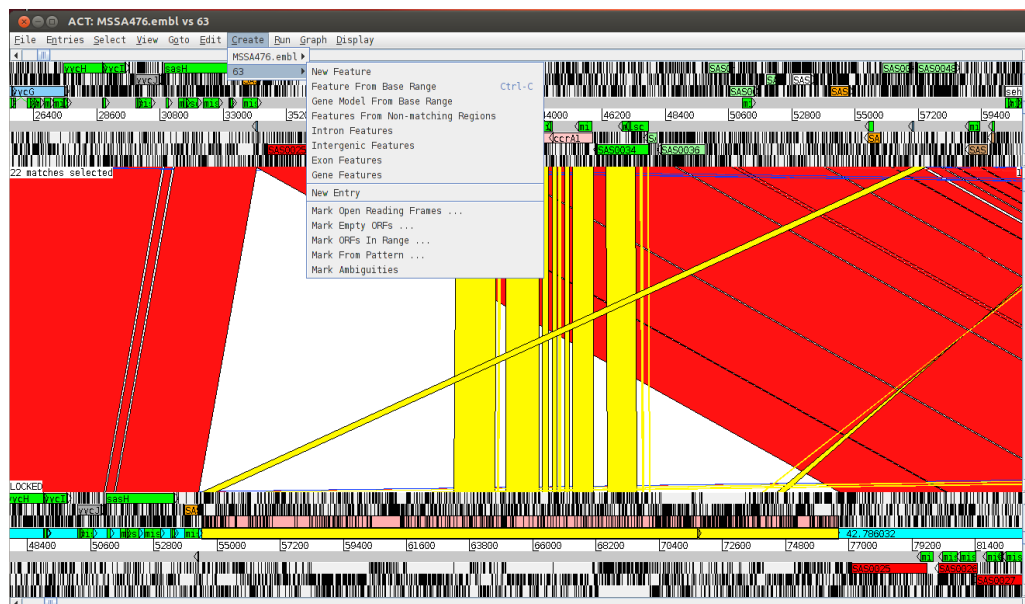
Region 3



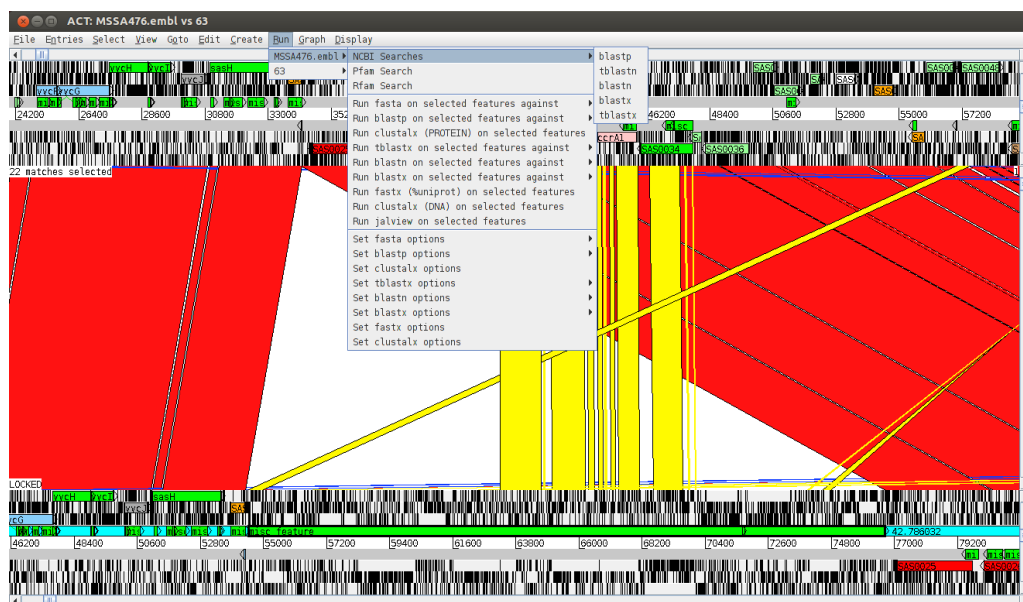
In this region near at the right hand side of the assembly, we have the non-mapping contigs (yellow). Previously we have seen that the two largest contigs are likely to be separate plasmids. The larger of the contigs has annotation transferred to it, however if you look in ACT, you will see that there it has no BLASTN matches to the MSSA476 chromosome. If you then look at the the annotation that has been transferred, you will see that it has come from the MSSA476 plasmid, pSAS, rather than the chromosome, this is because we included EMBL files for both the plasmid and chromosome in the RATT transfer. This indicates that 16B contains a similar plasmid to that found in MSSA46.

For the regions of difference that do not have any annotation, we can use a useful function of ACT (and also Artemis) to see what similar regions there are in the public sequence databases. To do this we are going to run a BLAST search at the NCBI from the **Run** menu in ACT.

Navigate yourself back to Region 1. Select the DNA region in the 16B assembly that is unique (**Right click** and hold, drag the cursor to the end of the region and release). Left click the **Create** menu, and move the cursor over the lower entry (**63**), and click **Feature From Base Range**. In the pop up feature box, change the **Key** to `misc_feature`, then click **Apply**.

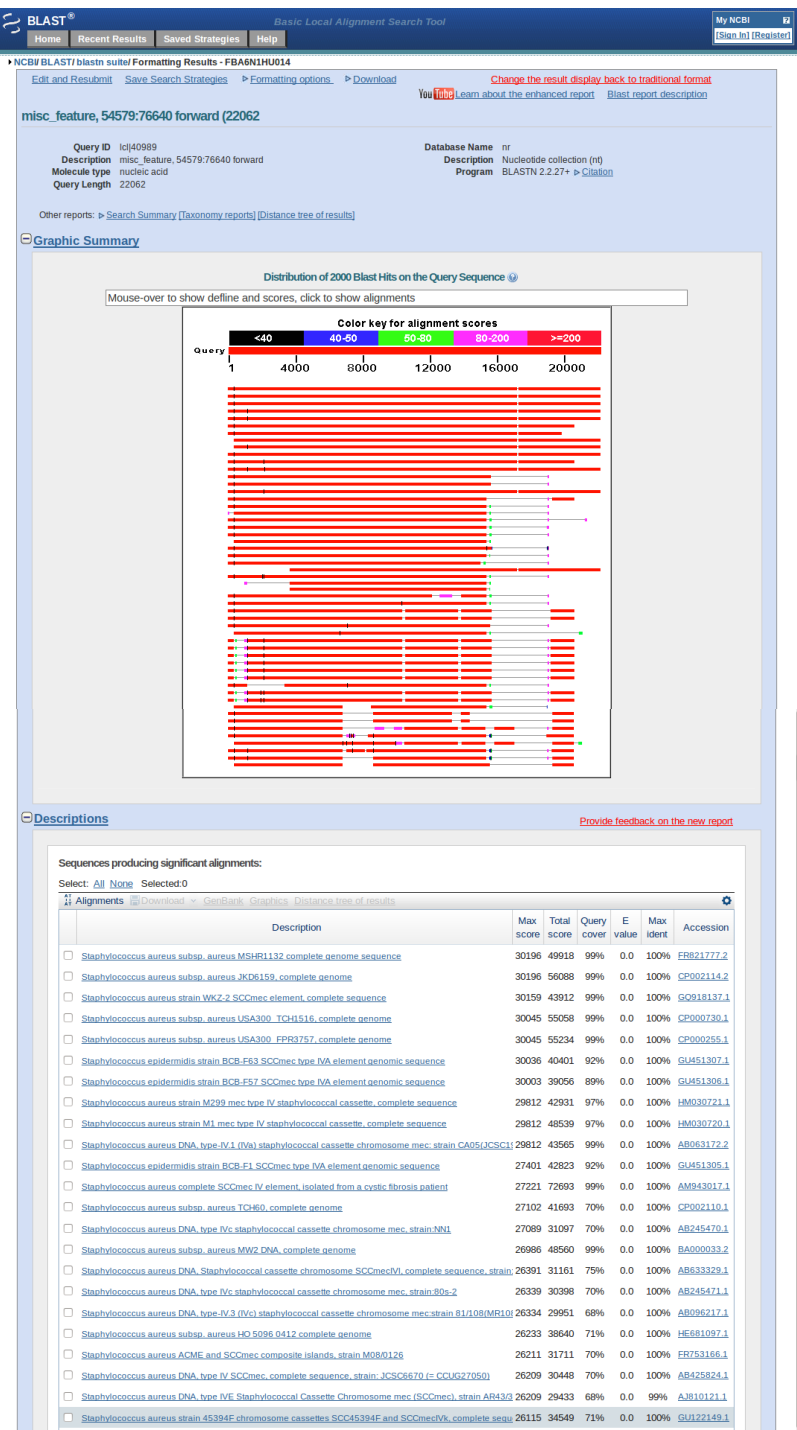


Click on the `misc_feature` you have just created. Click the **Run** menu, and move the cursor over lower entry (**63**), then over **NCBI searches**.



In the NCBI searches sub-menu you will see the various flavours of BLAST that you can run. We are going to run a BLASTN (DNA-DNA comparison) and also a BLASTX (translated DNA-Protein comparison) search for the feature. First click **blastn**. An **Options for blastn** window will appear that allows you to change the blast parameters. We are going to run it with the default settings, therefore click **OK**.

The BLAST job is now sent by ACT to the NCBI, and the Web browser window will open, and the results will appear when they have finished.

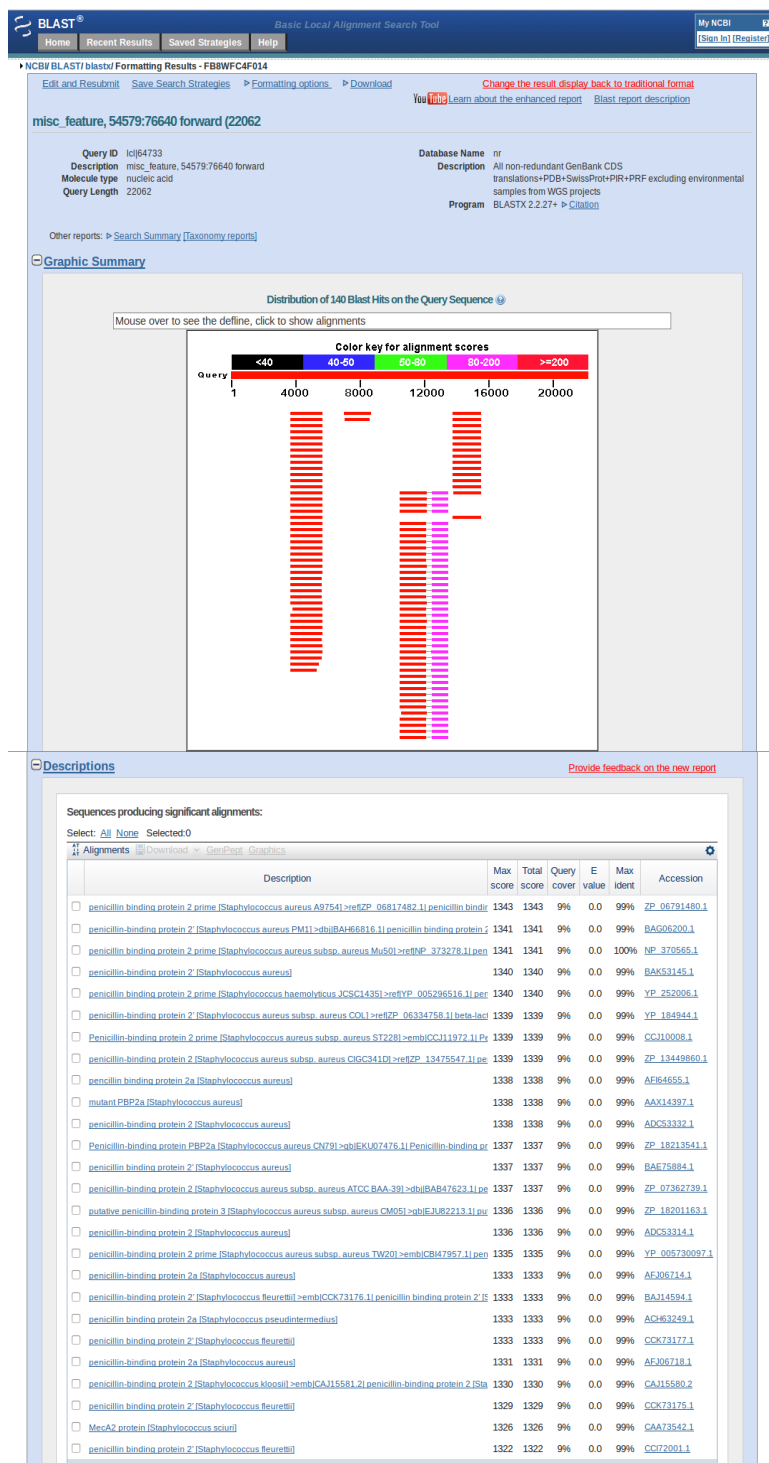


Look at the BLASTN results and see what matches there are, and how much coverage there is of the region we are interested in.

What is the identity of some of these sequences?

Does it correspond to particular type of mobile genetic element (MGE) and what genes would you expect to find on this element?

Having seen the DNA-DNA matches, we are now going to repeat the NCBI search with BLASTX this time (Click the **Run** menu, and move the cursor over the lower entry (**63**), then over **NCBI searches**, and click **blastx**). This will search for protein coding sequences in the region of interest that have BLAST matches to proteins in UniProt.



What is the identity of the matching sequences and their predicted function?

How does this relate to the antibiotic resistance?

If you want to easily locate the proteins sequences in the 16B assembly that correspond to the BLASTX matches, you can select sequence from the BLAST alignment (displayed in the browser) and use the **Navigator** function found under the the **Goto** menu (copy and paste a small portion of the match into the **Find Amino Acid String** of the **Navigator**).

Now that you have done this for Region 1, go and have a look at Region 3. What is the identity of the second largest contig in the non-mapping contigs, and what does it encode?

E: Examining the evolution of drug resistance in ST1 *S. aureus*

Up until now we have compared the 16B assembly to only one other ST1 *S. aureus* strain, MSSA476. We are now going to introduce another strain to the comparison, MW2, and start looking at the genetic differences between the isolates that may impact on their biology. Although MW2 was isolated in a different country (USA), many thousands of miles away from 16B and MSSA476 (both UK), it still belongs to the same clone, and probably share a common ancestor tens rather than hundreds of years ago. A clinically important phenotypic difference between these isolates are their antibiotic resistances:

16B - penicillin^R, fusidic acid^R, methicillin^R, erythromycin^R

MSSA476 – penicillin^R, fusidic acid^R

MW2 – penicillin^R, methicillin^R

As you will hopefully have just discovered, it is possible to use genome sequence data to find the genes responsible for antibiotic resistance. Examining the genetic context of these genes helps us to understand the mechanism that are driving the evolution of resistance in these *S. aureus* isolates. In this final part of the Module you are going to use the comparisons with MW2 and MSSA476 to identify regions of difference between regions that distinguish the isolates, and explain the differences in the antibiotic resistance phenotypes.

Before we begin this exercise close down any ACT session you have open.

In order to examine the regions of difference in the 16B assembly with MW2 we are going to generate a comparison file that we can load in ACT, as we did previously for MSSA476.

At the prompt type and return the command line:

```
formatdb -p F -i 16B.ordered.fasta
```

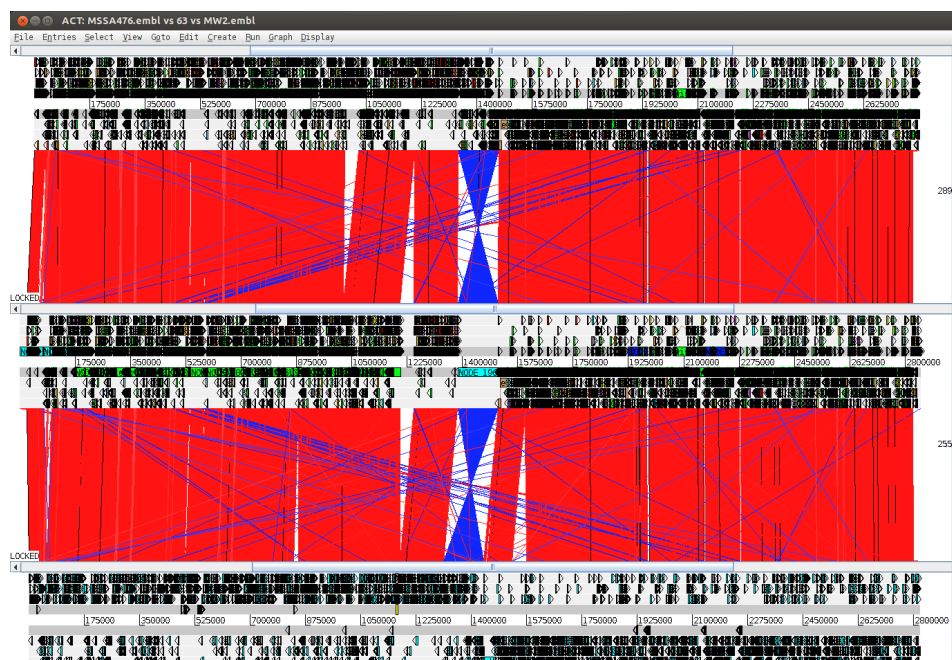
Next type and return the command line:

```
blastall -p blastn -m 8 -d 16B.ordered.fasta -i MW2.dna -o 16B.ordered.fasta_vs_MW2.dna
```

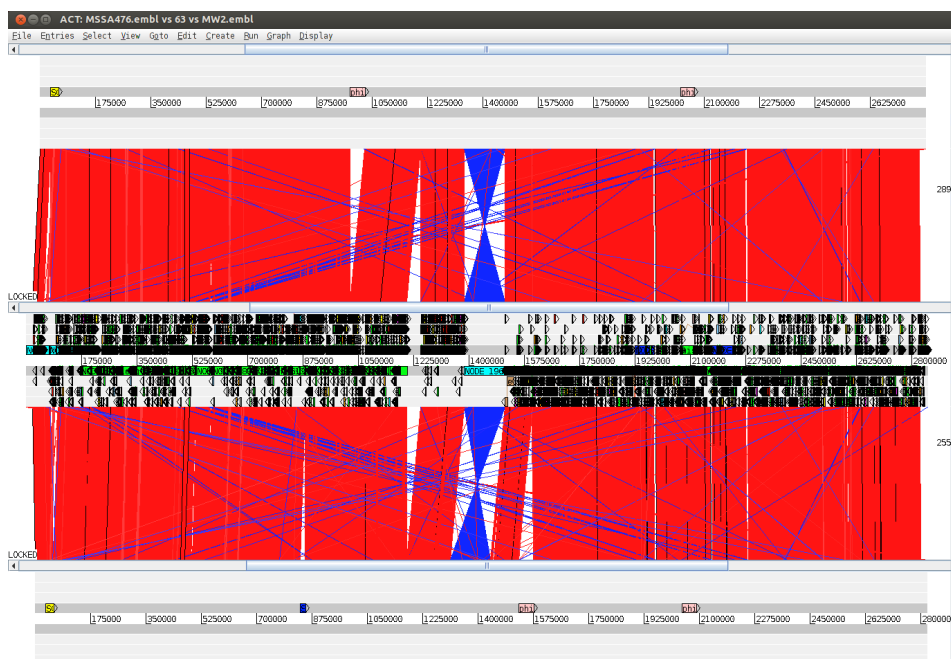
We are now going to load up the three sequences and relevant comparison files into ACT. You can do this either from the command line or by clicking on the ACT icon. If you prefer to do it from the command line you can type:

```
act MSSA476.embl MSSA476.dna_vs_16B.ordered.fasta <(cat 16B.ordered.tab 16B.ordered_staph-55e08.q2c2068.final.embl) 16B.ordered.fasta_vs_MW2.dna MW2.embl &
```

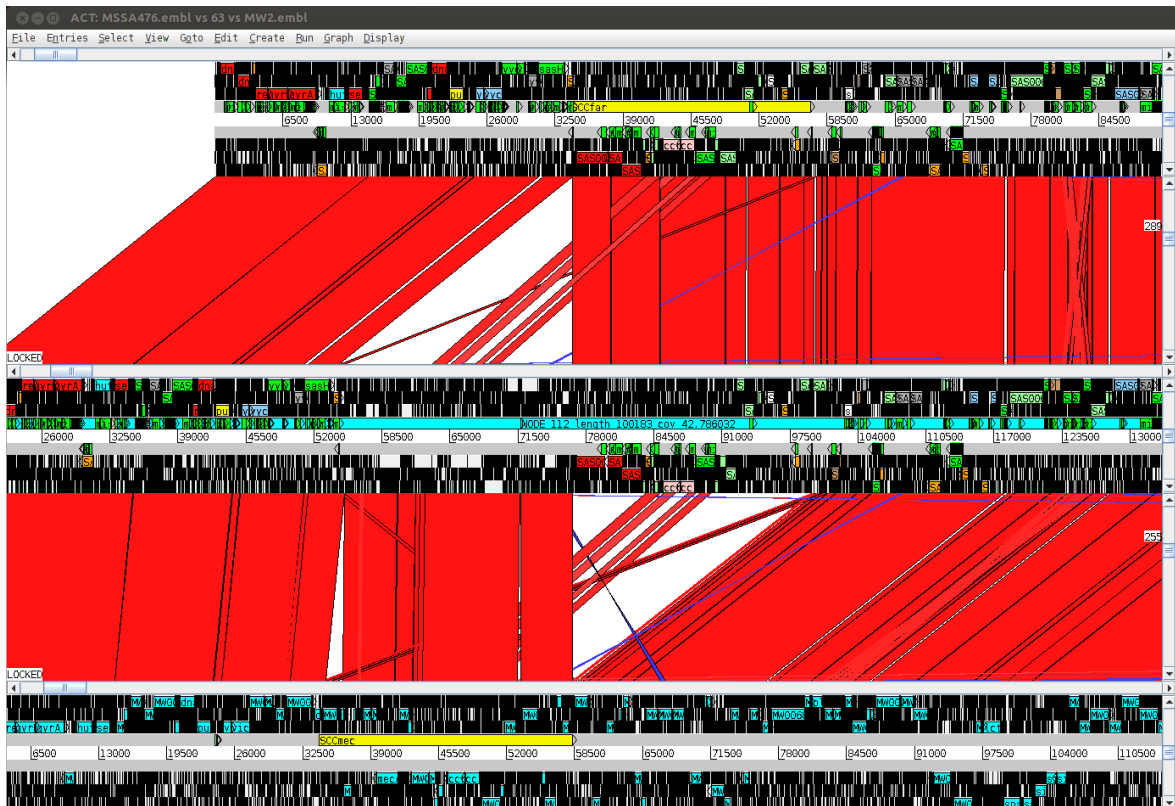
Now that you have included the MW2 sequence to the comparison you should see an ACT view with three DNA panels and two comparison panels separating them. In this zoomed out view, MSSA476 is on the top, 16B is in the middle and MW2 on the bottom. You will also notice that in the ACT menu at the top there are now three entry options.



To help you with your investigations, we have also provided two additional annotation files that contain misc_features which mark the extent of MGEs identified in the MSSA476 and MW2 chromosomes. These can be loaded into the appropriate entry (from the menu click **File**, the entry you want, then **Read An Entry**). The misc_features are colour coded in the ACT view according to the type of MGE (see legend on page 2 of this Module).



Here is the Region 1 that we have looked at previously, now with MW2 at the bottom. The regions of 16B that lacking annotation transferred from MSSA476, contains a matches to a region of the MW2. Does the identity of this MW2 region correspond to what you have seen from the NCBI BLAST searches? What has occurred in this region of the 16B chromosome that could explain the structure of this region in comparison to the other strains?



Compare the other regions containing MGEs. How do these regions vary in the three strains, and what do they encode? Does this explain the differences in the antibiotics phenotypes of the isolates? Can you find any other important genes associated with MGEs that are vary in the isolates that are clinical relevant (clue, think toxins).

Annotated Annotation

In the example we have looked at, we are fortunate that we have annotation for a closely related reference sequence that that we can use to transfer to our isolate of interest's assembly. In this case most of the query isolate's assembly is covered by the transferred annotation. What if you are not so lucky, and you do not have a appropriate reference which you can use? What options are available to you?

If you are dealing which relatively small amount of sequence it may be feasible to annotate it be hand using some of the freely available tools and resources, such as BLAST querying of the public sequence databases, protein motif database searches, TMHMM searches etc. (see Appendix V). If you are dealing with a large amount of sequence, or volume of isolates, this is not going to be practical. One solution is to use some of the automated bacteria genome annotation pipelines that are available via the web.


There are several sites that will allow you to upload your sequence, and then run various tools on it to predict features and functions, which you can then subsequently download and examine. The automated annotation servers vary in their analyses, and also the options that you can apply, but generally they will run an automated gene prediction on the sequence and annotate the genes based on similarity searches to protein databases and motif searches.

Whilst the annotation produced by these servers does not necessarily contain the accuracy or insight that human generated annotation provides, it does provide a valuable start point from which from which you can improve the annotation.

Because of the time that it can take for some of these servers to produce results, even for small sequences (several hours, as they are often busy with many jobs), it is not practical for us to include using one of these servers in the Module exercises. If you are interested in what the results of one of these pipelines looks like we have included the results of an automated annotation of the 16B assembly in the Extra_files directory in the Module_6_Genome_Assembly_Analysis directory.

The resource that we used was RAST (rast.nmpdr.org) (Aziz *et al.*, 2008 BMC genomics. 8;9:75). If you want to have a play with it, you have to register with them it before you use it, and this can can take a couple of hours.

The starting sequence that was used was the 16B.ordered.fasta file that we produced. The default parameters were used (see images on the following page for screen shots from the submission).



RAST Rapid Annotation using
Subsystem Technology version 4.0

The NMPDR, SEED-based, prokaryotic genome annotation service.
For more information about The SEED please visit theSEED.org.

Home Your Jobs 

Info: To monitor RAST's load and view other news and statistics for RAST and the SEED, please visit: ["The Daily SEED."](#)

RAST (Rapid Annotation using Subsystem Technology) is a fully-automated service for annotating bacterial and archaeal genomes. It provides high quality genome annotations for these genomes across the whole phylogenetic tree.

As the number of more or less complete bacterial and archaeal genome sequences is constantly rising, the need for high quality automated initial annotations is rising with it. In response to numerous requests for a SEED-quality automated annotation service, we provide RAST as a free service to the community. It leverages the data and procedures established within the [SEED framework](#) to provide automated high quality gene calling and functional annotation. RAST supports both the automated annotation of high quality genome sequences AND the analysis of draft genomes. The service normally makes the annotated genome available within 12-24 hours of submission.

Please note that while the SEED environment and SEED data structures (most prominently [FIGfams](#)) are used to compute the automatic annotations, the data is NOT added into the SEED automatically. Users can however request inclusion of a their genome in the SEED. Once annotation is completed, genomes can be downloaded in a variety of formats or viewed online. The genome annotation provided does include a mapping of genes to [subsystems](#) and a metabolic reconstruction.

To be able to contact you once the computation is finished and in case user intervention is required, we request that users register with email address.

If you use our service, please cite:

The RAST Server: Rapid Annotations using Subsystems Technology.

Aziz RK, Bartels D, Best AA, DeJongh M, Disz T, Edwards RA, Formosa K, Gerdes S, Glass EM, Kubal M, Meyer F, Olsen GJ, Olson R, Osterman AL, Overbeek RA, McNeil LK, Paarmann D, Paczian T, Parrello S, Pusch GD, Reich C, Stevens R, Vassieva O, Vonstein V, Wilke A, Zagnitko O. *BMC Genomics*, 2008, [\[article \]](#)

This project has been funded in whole or in part with Federal funds from the National Institute of Allergy and Infectious Diseases, National Institutes of Health, Department of Health and Human Services, under Contract No. HHSN272200900040C.

You are already logged in.

» [Go to the Jobs Overview](#)

» [Upload a new job](#)

» [Logout](#)



RAST Rapid Annotation using
Subsystem Technology version 4.0

The NMPDR, SEED-based, prokaryotic genome annotation service.
For more information about The SEED please visit theSEED.org.

Home Your Jobs 

Upload a Genome

Review genome data

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

Contig statistics

Statistic	As uploaded	After splitting into scaffolds
Sequence size	2849152	2769096
Number of contigs	1	64
GC content (%)	32.6	32.6
Shortest contig size	2849152	149
Median sequence size	2849152	24316
Mean sequence size	2849152.0	43267.1
Longest contig size	2849152	192222

Please enter or verify the following information about this organism:

Required information:

Taxonomy ID: (leave blank if NCBI Taxonomy ID unknown)

 Find the taxonomy id for your organism by searching for its name in the [NCBI Taxonomy browser](#).

Taxonomy string:

Domain: ☒ Bacteria ☐ Archaea ☐ Virus

Genus:

Species:

Strain:

Genetic Code: ☒ 11 (Archaea, most Bacteria, most Virii, and some Mitochondria)
☐ 4 (Mycoplasmae, Spiroplasmae, Ureoplasmae, and Fungal Mitochondria)

[Use this data and go to step 3](#)



RAST Rapid Annotation using
Subsystem Technology version 4.0

The NMPDR, SEED-based, prokaryotic genome annotation service.
For more information about The SEED please visit theSEED.org.

Home Your Jobs 

Upload a Genome

Complete Upload

By answering the following questions you will help us improve our ability to track problems in processing your genome:

Optional information:

Sequencing Method: ☐ Sanger ☐ Mix of Sanger and Pyrosequencing ☐ Pyrosequencing ☒ other

Coverage:

Number of contigs:

Average Read Length: (leave blank if unknown)

Please consider the following options for the RAST annotation pipeline:

RAST Annotation Settings:

Select gene caller:

Select FIGfam version for this run:

Automatically fix errors? ☒ Yes

Fix frameshifts? ☐ Yes

Build metabolic model? ☐ Yes

Backfill gaps? ☒ Yes

Turn on debug? ☐ Yes

Set verbose level:

Disable replication: ☐ Yes

[Finish the upload](#)

Please select which type of gene calling you would like RAST to perform. Note that using GLIMMER-3 will disable automatic error fixing, frameshift correction and the backfilling of gaps.
 Choose the version of FIGfams to be used to process this genome.

The automatic annotation process may run into problems, such as gene candidates overlapping RNAs, or genes embedded inside other genes. To automatically resolve these problems (even if that requires deleting some gene candidates), please check this box.
 If you wish for the pipeline to fix frameshifts, check this option. Otherwise frameshifts will not be corrected.
 If you wish RAST to build a metabolic model for this genome, check this option.

If you wish for the pipeline to blast large gaps for missing genes, check this option.
 If you wish debug statements to be printed for this job, check this box.
 Set this to the verbosity level of choice for error messages.
 Even if this job is identical to a previous job, run it from scratch.