Lab Report 3

the Assembly of a Reference Genome Using Contigs and Scaffolding

Helen King- A15362722

Abstract: The de novo assembly pipeline is used to construct a genome through raw reads which have had their read quality improved through trimmed and filtering out of low quality reads. This creates an accurate gene model of the organism, bacteria *Staphylococcus aureus*, through using contigs and then scaffolding.

Intro: De novo assembly is used in only used in certain circumstances as reference based alignment is easier, detects repeats and tolerates sequencing errors. Using de novo alignment needs no reference and detects non-collinear transcripts. It has a significant impact when forming a reference genome, or to recreate the original sequence, as it is unbiased. The organism being used is the *Staphylococcus aureus,* bacteria, which has a genome size of around 3 million bp. We used this as a model as it has got a small genome relative to other organisms like Homo sapiens have 3 billion bp and even yeast have around 12 million bp. The library is set up by assembling contigs from the paired end longer reads. Then the mate-pair end "short_jump" reads are used as scaffolding for the assembled contigs.

Methods: Use commands such as git clone, git add, git commit and git push to control the repository for this week. There are four fastq files in the public library, two for short_jump reads and two for normal reads. K-mer size can vary and need to be trialled and then determined by first using jellyfish to determine the k-mer distribution within the data. Jellyfish count uses flag –m for k-mer length, -C flag used to ignore directionality, -s is used hast table estimate, value larger than genome 10000000 and –o flag is used for the output name. Use jellyfish histo on each of the files. Then use the .histo file, and plot it using ipython in terminal.

Use the k-mer distribution approach for error correction of the poor data, remove the low frequency k-mers displayed in the histogram graph (Figure 2) as the peak before the curve. Emacs a file list with the names of the .fastq files you want to correct (frag_1 and frag_2). Then run KmerFreqHA command with flags –L for maximum read length set to 101, -k flag to 27 for kmer size, -i flag for initial size of has table to be 10000000, -p set to the output prefix and –l set to the name of the filelist just created. This is so we can do the pipeline with three different kmer sizes and compare assemblies at the end.

Next in the pipeline, we are going to run Corrector_HA command using the flag –k 27 for kmer size, -Q at 33 for the quality value, -o at 2 to output a .fa file, -l to 5 as that's where the valleypoint in the kmer histogram and the input prefix and then the filelist we made. The output is frag_1.fastq.cor.fq

Calculate genome size by finding the depth of coverage (N), kmer peak (M), kmer size (K), average read length (L) and total bases (B).
$N = \frac{M \times L}{L - K + 1}$ and $Genome\ size = \frac{T}{N}$. Find the new valley point for the M value. Then use NR and sum commands to find L. K is then 27 as you set it to. The total number of bases (B) should be found by using grep –v command and then wc and awk.

The next step in the method is to assemble reads with minia command using the –in flag for the input file list, -abundance-min to 2 and –out to be the prefix output. Repeat this three times with –k-mer-size flag for kmer size set to 27,31 and 43 .

The final step for this lab is to analyse these contigs using wc –l and then divide by two to get the number of contigs for each minia.fa file. The length of the shortest is found using cat , awk, sort and then head commands in a pipeline. The length of the longest is ound by doing the same command as shortest but changing the fag for sort to –n. Otherwise, you can just use the data mash command with min 1 and max 1. For further analysis use QUAST which is the quality assessment tool for genome assemblies. Scp your files to you desktop then upload them, skip contigs shorter than 100 and select prokaryotic. The genome is unknown.

The next lab you then use the whole class data to create two scatter plots for kmer size and contig length and then again for kmer size and contig N50 value. These contigs then have to be scaffolding to make a complete genome. This is when the short jump reads come in handy. Each read will hopefully span two contigs one forward read on one and then one backwards read on the other.

The de Bruijn graph is used for connecting contig assembly. Create a new folder called Thursday within this folder to keep it ordered using mkdir. Move all the short_jump files and the minia contig.fa files into it. Emacs a new library file that contains the short jump file names and then the insert size (3500), inset size error (0.5) and orientation (RF). Then run the SSPACE_Basic_v2.0.pl command, citing the new library file with the flag –l, specify in the list of contigs with –s flag, setting the minimum of 100 for the contig size –z flag, -v to 1 for the verbose output and then –b for the output prefix.

Next stage is to clean up the original shortjump read files and see if the scaffold improves. There is a compromise between data quality and data quantity. Must trim to improve the average quality of the reads using the sickle pe command. Use the flags –t set to sanger as it is sanger sequencing reads, -f to the short jump 1.fastq and the –r to the other short jump file. –o and –p flags are set to the output prefix and the –s flag is set to singletrim. Then scaffold again using the same space command as before but create new library file including the trimmed.fastq files.

The final thing needed to do is to use GapCloser command to close the gaps between the scaffolds created. –o flag is set to the name of the output .fa file, -a flag set to the input scaffolds file .fasta, -l to 101 for the maximum read length in your experiments and -b for the configuration file describing the read libraries to input. The configuration file is give on the SOAP webpage, edit it to specify the input files. Be careful to consider the outie direction of the short jump reads.

The way we are going to evaluate this assembly is through QUAST. First, go to NCBI and download both the fasta file and gene annotation in GFF3 form for the reference sequence of Staphylococcus aureus strain known as NC_010079. Sign into QUAST and add the files to upload your assembly., check scaffolds, find genes and prokaryotic boxes. Upload the reference genome and its annotations. Then repeat without choosing the scaffolds button so then we can visualise it in the Icarus browser. There will be two outputs per run of GapCloser, one as a split version of the assemblies and one of the assemblies as scaffolds. The contigs are reconstructed by the broken files created by

QUAST. After that, results for real scaffold and reconstructed contigs can you see if this GapCloser step was useful. Repeat all steps for the three kmer steps and then you can also compare N50 and max contig values for them.

Results:

Fastq File	Output	Number of Reads
frag_1	6230376	1557594
frag_2	6230376	1557594
short_jump_1	4445728	1111432
short_jump_2	4445728	1111432

Table 1. the number of reads in each raw file given in the public.

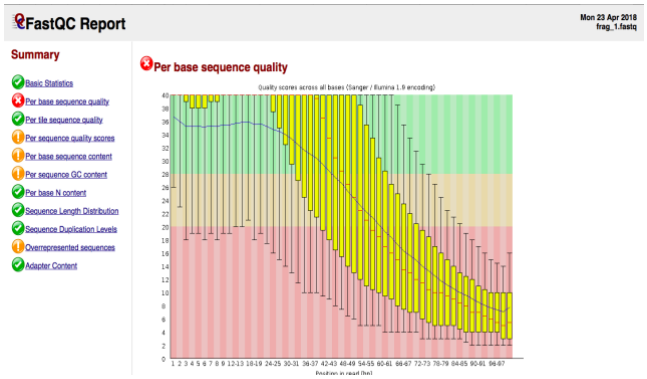
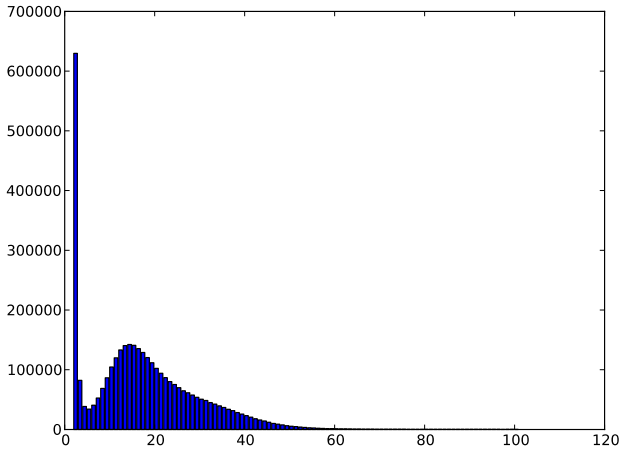
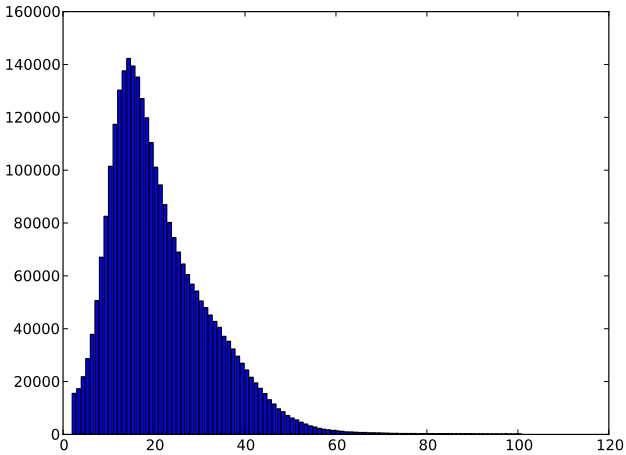


Figure 1. frag_1.fastq quality report.

Very poor quality at lower base sequence. As seen in figure 1 of the frag_1. This is seen in both direction of the reads. Suggests the need for trimming or some form of downstream processing to remove this. As said in the methods section in order to make accurate in situ processing, you need to compromise between quality and quantity of reads.





Figures 2 and 3. Histograms of frag1 and frag2 k-mer (x-axis) against depth (y-axis). Figure 3 is before correction and figure 4 is after k-mer based error correction.

Figures 2 and 3 display the before and after of k-mer based correction. The low frequency k-mers using KmerFreq_HA and Corrector_HA. Then jellyfish is used to display the distribution of k-mers.

The genome size I got was around 13 million bp which is off 3 million bp by 6 times more. I don't think I have used the correct values because this is horribly off. I used the calculation seen above and then

$$N1 = \frac{142225 \times 95.631}{95.631 - 27 + 1} = 19.22755$$

$$G1 = \frac{247286808}{19.22755} = 13015095$$

If T is higher- the quality of reads could mean they should have been removed but haven't so increase the number of bases. N is lower- Due to kmer peak being too low.

k-mer size	Number of contigs	Longest contig	Shortest contig	Average scaffold size	N50
27	323	258295	101	9041	112792
31	284	258872	100	10191	128380
43	239	358369	100	12108	235666

Table 2 contig analysis

The reads have been assembled using minia. This is the range of k-mer sizes. An dthe number of contigs lower when k-mer size increases. The size of the longest contig makes sharp increase between 31 and 43 compared to a k-mer size of 31 and 27. The N50 value displays the length of the contig that is needed to cover half of the genome, its like mean but seen as more representative assembly metric. The N50 value increases the with k-mer size, so 43 is a better k-mer size based of the N50.

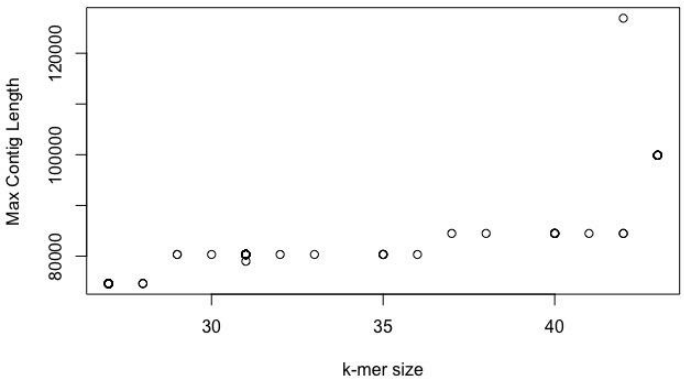


Figure 4. Scatter plot on k-mer size against max contig class at variety of k-mer sizes.

Figure 4 and 5 is using class wide results the same as displayed in table 2, but compiled. Figure 4 shows a small increase in max contig length with the increase of k-mer size until about 43 k-mer and then a massive spike in max contig length with a few reads. These might be erroneous. This is analysis of the contigs is done by uploading to QUAST, a web tool which forms a quality assessment for genome assemblies.

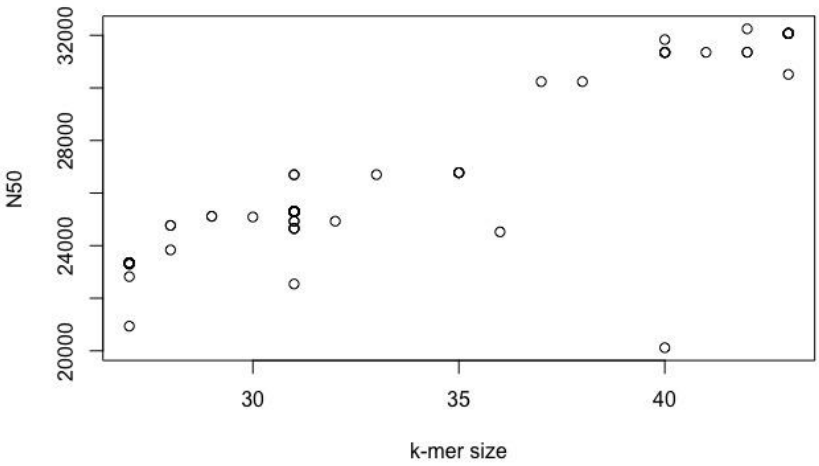


Figure 5. Scatter plot of k-mer size against N50

Figure 5 shows a trend for a proportional relationship between k-mer size and N50. When k-mer size increases N50 also increases. This infers that in general te larger the kmer, the better the quality of the contigs. This also justifies using k-mer size of for genome compilation.

The next stage is to use the second set of reads which are "outties" and named short jumps due to their short read length of 37 bp compared to 101 read length of the frag reads, despite having insert size of 3500 bp. Their function is to act as scaffolding between the contigs. SSPACE_Basic command performs the scaffolding. This is then repeated after trimming the short jump reads to improve the short jump read quality using sickle. This displays a significantly higher N50 and max scaffolding from before to after trimming. GapCloser is finally used to try and close the gaps in the scaffold. Finally, we uploaded these finalised versions of the *Staphylococcus aureus* genome using three different preliminary k-mer sizes. By uploading them to QUAST for analysis we can see when the k-mer size has any effect on the quality of the end genome.

	Before Trimming		After Trimming	
kmer size	Max Scaffold	N50	MaxScaffold	N50
27	74591	23343	112792	258295
31	80334	25284	128380	258872
43	99881	32069	235666	358369

Table 3 before and after the trimming of the reads

There are two outputs when QUAST is run, scaffold assemblies and broken (seen in Table 3). Assemblies are split by fragments greater the size N>100, this is a reconstruction of the contigs. The difference between the N50 and max scaffold between the scaffolding and reconstructed contigs, governs whether the scaffolding step was worthwhile. It could introduce errors by placing unrelated contigs into one scaffold and then also not fixing misassemblies already present in the contigs, compounding the error. We would expect similar number of misassemblies between broken and assembled, or lower in the broken version. Scaffolding can be skipped if found to introduce more errors.

The already known size of the genome is 3 million.

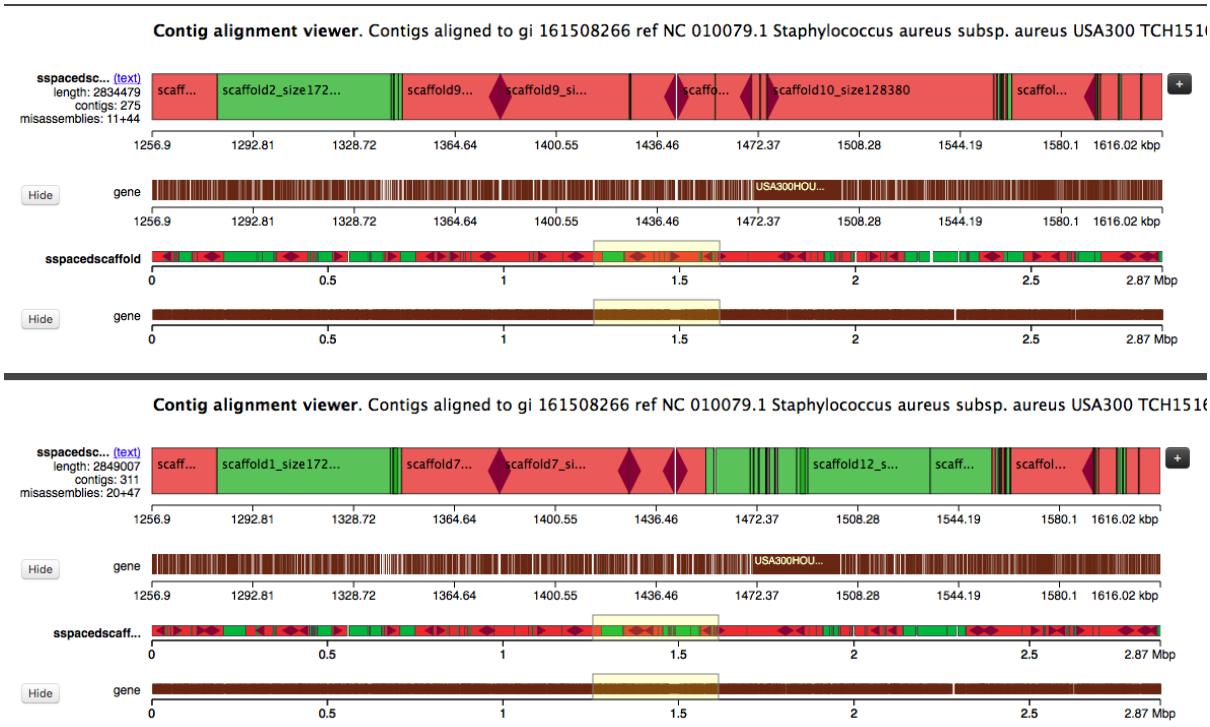
$fraction\ of\ the\ genome\ covered\ by\ scaffolds\ (27) = \frac{2{,}849{,}007}{3{,}000{,}000}$=0.9496…

The mismatch rate is the rate there is difference between sequence read and a reference genome after alignment. The worse the sequencing run the higher the number of mismatches. There must be a high level of confidence in the reference genome for this

rate to be a valid statistic. The higher number of sequencing errors and the larger difference between the reference and sample genome, the more mismatches you have. For 31-mer and 43-mer, there was 4.22, 6.37 and 4.45 per 100 kbases mismatches respectively. The red scaffolds displayed in Icarus seen in figure 6 shows the partially unaligned contigs, with at least one unaligned fragment of length more than the unaligned threshold.

k-mer size	N50	N50 (broken)	Max. Contig	Max. Contig (broken)
27	110509	84453	258569	258569
31	127459	90496	259002	259002
43	234083	172742	358985	281179

Table 3 final run of QUAST on three k-mer sizes of frag_1 with end genome.



Figures 6 and 7. The Icarus view of the scaffolding files of 31-kmer and 41 k-mer.

Discussion: Overall, 94% of the genome was covered by the scaffolding, which preliminarily sounds promising. To evaluate this properly, we need to look at the quality of the genome using a variety of k-mer sizes to form contigs and then the scaffolding as separate steps. I would not consider this a finished genome as it is not fully covering the known genome size, as well as having misassemblies (seen in figure 6 and 7). The number of genes found from the NCBI reference genome is 2,872 genes. [2]. The three k-mer sizes each have got gene numbers of 2794, 2784 and 2828. This dislays that 41 kmer size, even though does not have the correct number of genes is the best k-mer size to produce the most complete genome out of all we tested. The scaffolds we used do not include complete gene sequence as shown by the max contig and N50 values which demonstrate a low genome quality, and therefore incomplete gene sequences. To improve the de novo assembly, the base problem is more coverage is

needed so that after trimming and filtering of low quality reads, the whole genome is covered and then reconfirmed. In addition to the challenge of poor sequence quality, polymorphism and repetitiveness are other challenges that need to be tackeled but on a more bioinformatics approach. [3] The assisted assembly method found by Gnerre tackels this problem. The variety of k-mer length during assembly was illuminating. Using 41 k-mer seemed to be the best fit to produce higher quality and higher coverage within the genome. This is validified by the higher N50 and max contig value compared to 27 and 31 k-mers. Also, there was a decrease in the unaligned contigs when view in Icarus. The use of scaffolding as well, was useful as it showed a decrease in the N50 values in all our scaffolds, when comparing to the broken scaffolds created by QUAST. Overall, the 41 k-mer still did not output perfect results to create a new reference genome of the *Staphylococcus aureus* genome, especially using only de-novo assembly.

Bibliography

[1] Nature- Technology Feature
De novo genome assembly: what every biologist should know- Monya Baker
[2] https://www.ncbi.nlm.nih.gov/genome/154?genome_assembly_id=299272
[3] Assisted assembly: how to improve a *de novo* genome assembly by using related species
Sante Gnerre*, Eric S Lander*, Kerstin Lindblad-Toh*† and David B Jaffe*
Addresses: *Broad Institute of Harvard and MIT, Cambridge Center, Cambridge, Massachusetts 02142, USA. †Department of Medical Biochemistry and Microbiology, Uppsala University, Husarg.3, Uppsala 751 23, Sweden.