Bioinformatics I
WS 15/16
Tutor: Alexander Seitz

1	2	$\sum$ (10)

Jonas Ditz
& Benjamin Schroeder

Blatt 10

(Abgabe am 18. January 2016)

Theoretical Assignment - *Assembly using de Bruijn graph*

The de Bruijn graph for the reads provided on the assignment sheet would look like the graph in Figure 1, if one chooses $k = 4$. As one can see easily, there are not just one but four different Eulerian paths in the graph. Each path results in one of the following superstring:

$$S_1 = ACCGTTAACGTAAACGT$$
$$S_2 = ACCGTAAACGTTAACGT$$
$$S_3 = ACCGTTAAACGTAACGT$$
$$S_4 = ACCGTAACGTTAAACGT$$

This happens due to the fact that there are nodes with more than one outgoing an incoming edge. If one chooses $k = 5$ the resulting de Bruijn graph would look like the graph in Figure 2. It is obvious that the number of nodes is depended on k. With a bigger k there are more nodes in the resulting de Bruijn graph. The resulting graph for $k = 5$ is non-connected. So the superstring for this graph is not just one but three strings:

$$S_{part_1} = TAAACTG$$
$$S_{part_2} = CGTAACGTTAA$$
$$S_{part_3} = ACCGT$$

One can see that $S_{part_1} = f_5$ and $S_{part_3} = f_1$, while S_{part_2} is the result of an overlap alignment between f_2, f_3 and f_4. In general the second graph is more realistic. It is very unlikely to get a connected graph with real-life data. So it is possible that just f_2, f_3 and f_4 come from the same area in the target DNA. f_1 and f_5 could come from a different contig and, hence, result in this graph (Figure 2). Since this is a minimal example the first graph (Figure 1) is not wrong but one should not expect to get a connected graph all the time. In fact a connected graph is suspicious, normally.

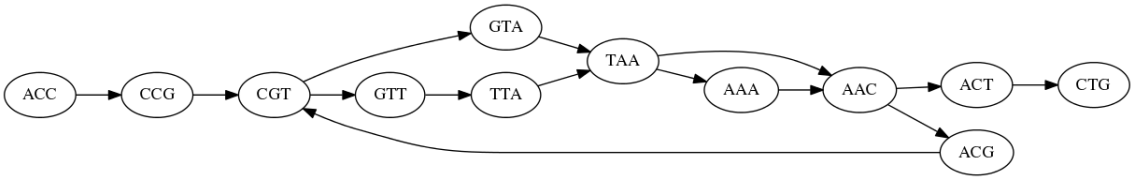


Figure 1: De Bruijn graph for the given reads and $k = 4$.

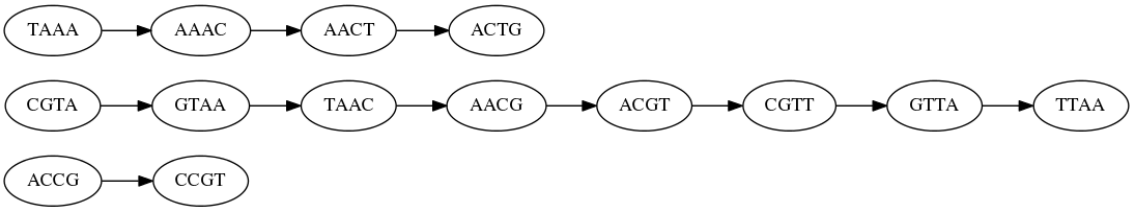


Figure 2: De Bruijn graph for the given reads and $k = 5$.

Practical Assignment - *Assemble the human mitochondrial genome*

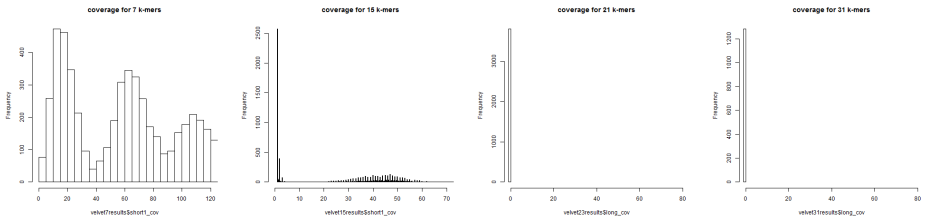
Introduction

The mitochondrial genome is rather a small genome in our world. But even this rather small genome, with its 10^4 basepairs is a though nut, when trying to discover its sequence from reads. For this purpose we used two different alignment algorithms. The Velvet algorithm by Zebrino and Birney is using de Bruijn graphs to assemble the reads.[1] Likewise does the short oligonucleotide alignment program (SOAP) denovo2 algorithm by Luo et al. use a de Bruijn graph [2]. The differences of both algorithms lie in their detailed structure.

The Velvet program starts with building an de Bruijn graph, in which each node has a so called twin node, which represents the reverse complement of the node. Both nodes are called block. The construction of the de Broijn graph happens via two databases. one is a hash table fo r the k-mers, and the second recording the overlaps of the k-mers of a read with other reads. Next a simplification step follows. After this two construction steps, the Error removal starts. The error removing focuses on to three aspects, the removing of tips, bubbles and erroneous connections. There are two rules for tip removing, one is the length is smaller than 2k and the other is called minority count. The minority count is true, if there is an alternative path, which is more common. Bubbles are redundant paths, which are removed via the Tour Bus algorithm. The last correction, removing erroneous connections is based on a threshold which is user set. [1] The SOAP program is structured in six modules, which handle read error correction, de Bruijn graph construction, contig assembly, paired-end reads mapping, scaffold construction and gap closure. The de Bruijn graph is since SOAPdenovo2 part of the program.[2] Like velvet, does SOAP use a hash table to build up a de Bruijn graph.[3] But before a correction of the sequencing takes place. and afterwards again corrections are applied.

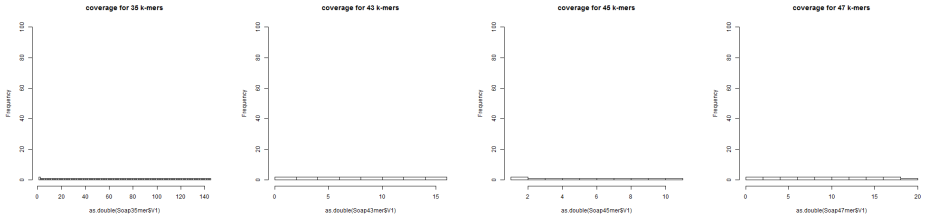
Results

Both algorithms were used to calculate the assembly of the mitochondrial genome separatly. The velvet algorithm was computed on the home source with four different k-mer sizes (from 7 to 31 bases). As a result of the lacking hardware and the missing password for the WSI-Server, the resulting data from the SOAP algorithm was thankfully provided by Sanja Koehlers goup. The statistics for each algorithm are shown in figure 3 and 4. The for the Velvet program are from the stats.txt and for the SOAP from the XYsoap.contig. The coverage was isolated via a java script (see attachment).



(a) k-mer size: 7 (b) k-mer size: 15 (c) k-mer size: 23 (d) k-mer size: 31

Figure 3: The figures show the coverage from the Soap program with differing k-mer size for the mitochondrial DNA assembly.



(a) k-mer size: 35 (b) k-mer size: 43 (c) k-mer size: 45 (d) k-mer size: 47

Figure 4: The figures show the coverage from the Soap program with differing k-mer size for the mitochondrial DNA assembly.

Discussion

The smaller the k-mer the more contigs are the result as it is visible in figure 3a. Larger k-mers have as result more unique contigs [1]. Generally we can not conclude anything from our results for the quality of the algorithms, because we did not use the same k-mer length. But an optimal k-mer size is a compromise between uniqueness of the k-mer (longer) and resulting amount of contigs (shorter).

Acknowledgement

Thanks again to Sanja Koehler an her group for the SOAP-data.

References

[1] Velvet: algorithms for de novo short read assembly using de Bruijn graphs. D.R. Zerbino and E. Birney. Genome Research 18:821-829.

[2] SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. Luo et al. GigaScience 2012 1:18.

[3] SOAP: short oligonucleotide alignment program" (2008) BIOINFORMATICS,Vol. 24 no.5 2008, pages 713–714 doi:10.1093/bioinformatics/btn025