Introduction (4)

To generate a genome sequence, a biological sample is fed into a sequencing machine. Regardless of the specifics of it, that machine will not provide a single data file with the full sequence. Instead, a long list of small files is created, each containing a small part of the sequence. The quality of each of these reads is not perfect, and they can contain random errors. To counter these errors, redundancy is built in. The end result is a big set of reads with overlapping sequences.

Assembler programs use sophisticated algorithms to turn these reads into a larger sequence. An assembler such as Newbler <bron> uses de bruijn string graphs to connect overlapping reads into a bigger sequence. A de bruijn string graph is a memory and computational efficient way to handle the assembly process. The assembler first breaks up the reads in even smaller overlapping parts, called k-mers. These usually have a length of somewhere around 25-31 basepairs (bp). It iterates through all these k-mers and start noticing overlaps. Not only will it find overlaps between k-mers that originate from the same read, but also from different reads. These will also be grouped, and the combined read will be stored as a graph (Compeau, 2011). This way, overlaps between different reads can be easily found, taken into consideration the size of the dataset.

<plaatje met overlapping k-mers [www.homolog.us/blogs/blog/2011/07/28/debruijn-graphs-i/](http://www.homolog.us/blogs/blog/2011/07/28/debruijn-graphs-i/)>

In the ideal case, this graph consists of only a single node, comprising of all overlapping k-mers, representing the full genomic sequence. However, biological imperfections of the sample, as well as technological artifacts will introduce small errors, which create new nodes and edges between them. Each edge represents a diverging path between two possible sequences. The redundancy in the reads is used to determine which of the possibilities is the correct one, as random errors will be represented less. The algorithm traverses the graph and removes the path that it considers an error.

After this process, a contig is created for each node. Then, a reference sequence of a related species is used as a scaffold, to place each contig in the right place, considering the overall sequence of the genome. This removes even more edges, because the known reference sequence provides additional information on the placement of neighbouring contigs, giving more certainty to the resulting sequence.

Current developments in technology provide researchers with a broader set of tools than before. These can be used to deal with the problems that arise when dealing with newly sequenced organisms.Whena new species is sequenced, a suitably close related reference sequence might not be available.

Another newer branch of research is the creation of a pan-genome. Where a reference sequence consists of a single static sequence, a pan-genome of a specific (group of) species can account for all the variation that is available within that set. Comparing an assembled contig with a reference sequence can make some biological variation look like an error while it isn't, so a pan-genome refrains from using a reference sequence, avoiding skewing the results. A lot of effort is put into optimizing *de novo* assembly, to be able to deliver high-quality sequences without the use of a reference sequence.

Creating a pan-genome is usually done with a specific type of *de novo* assembly, called co-assembly. In a co-assembly, two or more samples are assembled concurrently. In this way, they can be used as a guide or control for each other, and more biological variation is more easily spotted. In the de bruijn graph representing this dataset, each sample is assigned a 'colour'. Then, when traversing the graph, the colour is used as a route. With this, differences in sequence coming from different samples can be easily identified (Nijkamp, 2012).

The field of co-assembly is very young <bron over eerste vermelding/publicatie/tool>, and not a lot of tools have yet been developed specifically for it. There are no widely adopted best practices, as is evident from the variety of available data formats to store large amounts of sequencing data, such as SAM, BAM, vcf, and ace <meerdere bronnen>. Tools that do exist focus more on viewing statistics about the data, and less about visualization <PanGP tool, PanFunPro, PGAT>.

To make a functional visualisation of a co-assembly, a few requirements need to be met. As long as the community doesn't settle on a standard for data storage, the tool needs to work with a variety of input formats. The tool needs to be able to show the full extend of the data and be able to show small details. On the other hand, it needs to prevent 'getting lost' and losing the general overview of the data. Follow-up analysis needs to be possible, so data should be able to be exported to a broad set of analysis tools, such as BLAST or comparative genome analysis tools <bronnenX>. Here, suggestions are done to create a tool that satisfies these requirements.