## Materials and Methods

### Overview

Stringit is a tool for visualizing co-assemblies. It is a web-based tool, built with javascript and the d3.js library. Its core tasks consist of the following:

* parsing (co-)assembler output
* reading the parsed data and determine underlying structures
* display the graph, accounting for the structures

The first part of these tasks is performed by external scripts, written in Python. They read the data coming from an AMOS databank or a 454 library. Then, they write the relevant data into a unified format that can then be loaded into Stringit itself.

Aside from native javascript code, the d3.js (Data Driven Documents) library is used. This library is used to bind data to DOM elements. Additionally, it provides visualization tools. In Stringit, a force-directed layout is used to display the network, and piecharts are used to provide more information in the node itself.

### Stringit File Format

Before Stringit can use the provided data, it needs to be parsed. Different sources each provide a different syntax

about relevant information and syntax

### Functions

Functions (describe parameters)

readFile

handleContent

groepering

determineTiers

contains

matchcriteria

makeGraaf

radius

gravity

coordinates

future work:

displaying single tier

changing graph based on clicks

displaying information based on selected node

### Datasets

To test the use of Stringit during development, datasets from different sources were used. Two datasets that were used in <Marigold paper> were downloaded, and relevant data about contigs, reads, and edges was extracted with the AMOS bankreport function. The format of this report was used to create the Python script for conversion from AMOS to a format that could be read by Stringit.

An artificial dataset was also created. Reads were simulated from the known *e. coli* and *e. albertii* genomic sequence, and pooled together. These pooled reads were then assembled with the 454 package (“Newbler”). After parsing with the right Python script, this data could be read by Stringit as well.

In previous versions, several other datasets were used. Most notably a graph file in the asqg format, used by the SGA assembler <bron>. However, as more functions were added, support for this format was dropped, as it did not contain all the necessary information. The asqg format only contains information on the assembled reads, and not on the origin of the reads. Without an external read mapper available, Stringit no longer had all the information it needed.

1. Overview. Stringit is a tool that visualises the data inside assembly graph files. The file is loaded in the webapplication, and is processed to a visual graph, which is displayed on screen. There, several tools can be used to analyse the data. The graph can be exported for use in other tools (such as alignment mappers).

2. Stringit. Stringit is a browser-based tool, where users can upload their assembly graphs, after which they can analyse the graph network in the browser window.

It does this by parsing the input file to a javascript object containing all the information on nodes and edges. Using information contained in the graph file, it groups the nodes based on the origin of the sample, creating differently coloured groups and making co-assembly analysis possible. Zooming into different tiers is possible, each consecutive one showing less overall sequence, and more detail. Based on the zooming tier, nodes get aggregated into larger nodes. This way, only information relevant for that level is displayed.

The nodes give information (via a mouse-over HUD) about the sequence(s), and edges denote overlap between nodes. When the sequence diverges because of variation, this is visually represented by two different nodes that converge behind the variation (such as a SNP or indel), which forms a ‘bubble’. The user can analyse any overlap and divergence between species or strains with this combination of zooming tiers, node aggregation, and bubble recognition.

2.1 Input data. As input for Stringit, a graph file is needed. Several assemblers are capable of producing such a file. So far, the graph formats of SGA and 454 ("Newbler") are implemented. However, any graph file is usable, in principle. The formats are recognized internally and processed into a JSON object, containing all the information of the original nodes and edges.

2.2 D3. Stringit uses a powerful javascript library to handle the graph data called d3. D3 is short for Data Driven Documents. It is used to bind data to DOM elements. Stringit uses a force-directed graph layout, one of several ways D3 can be used to display graph data. Nodes representing sequences are connected by edges representing overlap, and the internal d3.js mechanism lay this out on screen. Additional functionality is added with a mouse-over display that shows information on the current node, as well as zooming in and out to regions of interest.

3. Features:

3.1 reading databanks and parsing into usable file

3.2 reading file, determining structures

3.3 displaying graph