Introduction to short read assembly

Frank Aylward

Abstract

Here is a short intro on short read assembly using the SPAdes assembler.

Some notes on tool installation:

To install SPAdes or other tools into your PATH you may wish to use a package manager called Miniconda. For this go to the website here and download the 64-bit Linux distribution:
https://conda.io/miniconda.html
# then, once you have located this file on your computer, type
bash Miniconda2-latest-Linux-x86_64.sh
# and follow the installation instructions. When asked to append the PATH info say "yes".
# After this you will need to close your terminal and re-open it before beginning again. Then you should be able to install tools using conda. For the module today we will need to install two tools: the sra-toolkit and an assembler called spades.
conda install -c bioconda sra-tools
conda install -c bioconda spades

Citation: Frank Aylward Introduction to short read assembly. **protocols.io**
dx.doi.org/10.17504/protocols.io.pifdkbn
Published: 17 Apr 2018

Protocol

Get the reads

Step 1.

# today we will be working with the raw reads from the Staphylococcus phage 812 genome sequencing project.

# To get the raw reads we need to use a program called the sra-toolkit. This is a tool maintained by NCBI to allow users to download data easily from the command line.
# The toolkit allows users to specify the unique accession number for a project and then download the associated reads.
# Here we will also use a command -X, which specifies how many reads we want to download. Since these datasets can be quite large we want to start with a small number. Here we will use 10,000

# we also want to use the --split-3 flag, which for Illumina data makes sure the forward and reverse reads are split into separate files.
fastq-dump -X 10000 --split-3 SRR6764339

Run spades and get the assembly

Step 2.

# Now once we have the raw reads we can begin assembling them using a program calls SPAdes.
# Now actually running the assembly is fairly easy- we just specify the read files, an output folder name, the number of threads we want to use, and the k-mer length.

```
spades.py -1 SRR6764339_1.fastq -2 SRR6764339_2.fastq -o phage -t 4 -k 21 &> log.txt
```

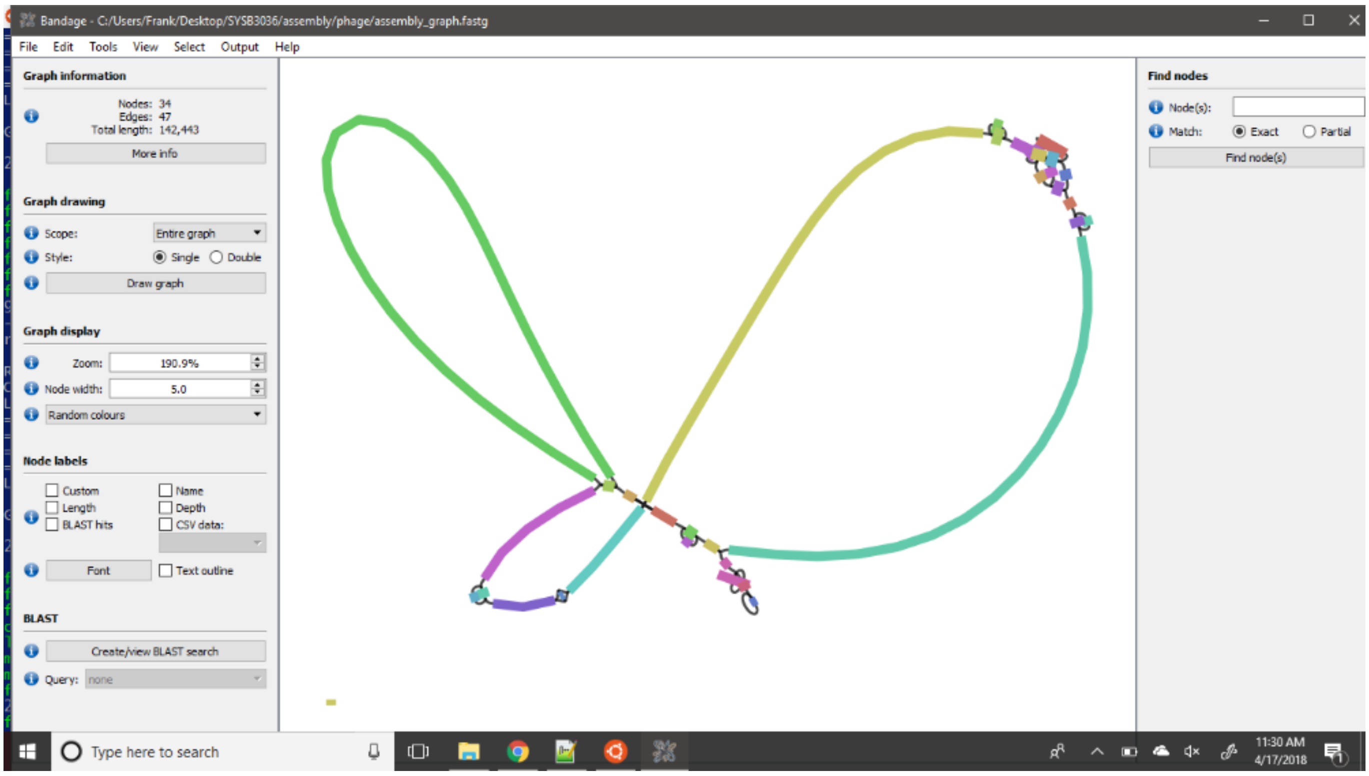
Visualize the assembly

Step 3.

Once you get the assemlby you can visualize the De Bruijn graph using a program called Bandage. It can be installed here:

https://rrwick.github.io/Bandage/

Once you get it installed you can upload the .fastg file and have Bandage draw the graph. You should get something that looks like this:



Bandage - C:/Users/Frank/Desktop/SYSB3036/assembly/phage/assembly_graph.fastg

Graph information
Nodes: 34
Edges: 47
Total length: 142,443

Graph drawing
Scope: Entire graph
Style: [✓ Single] Double

Graph display
Zoom: 190.9%
Node width: 5.0
Random colours

Node labels
Custom, Length, BLAST hits, Name, Depth, CSV data:, Text outline

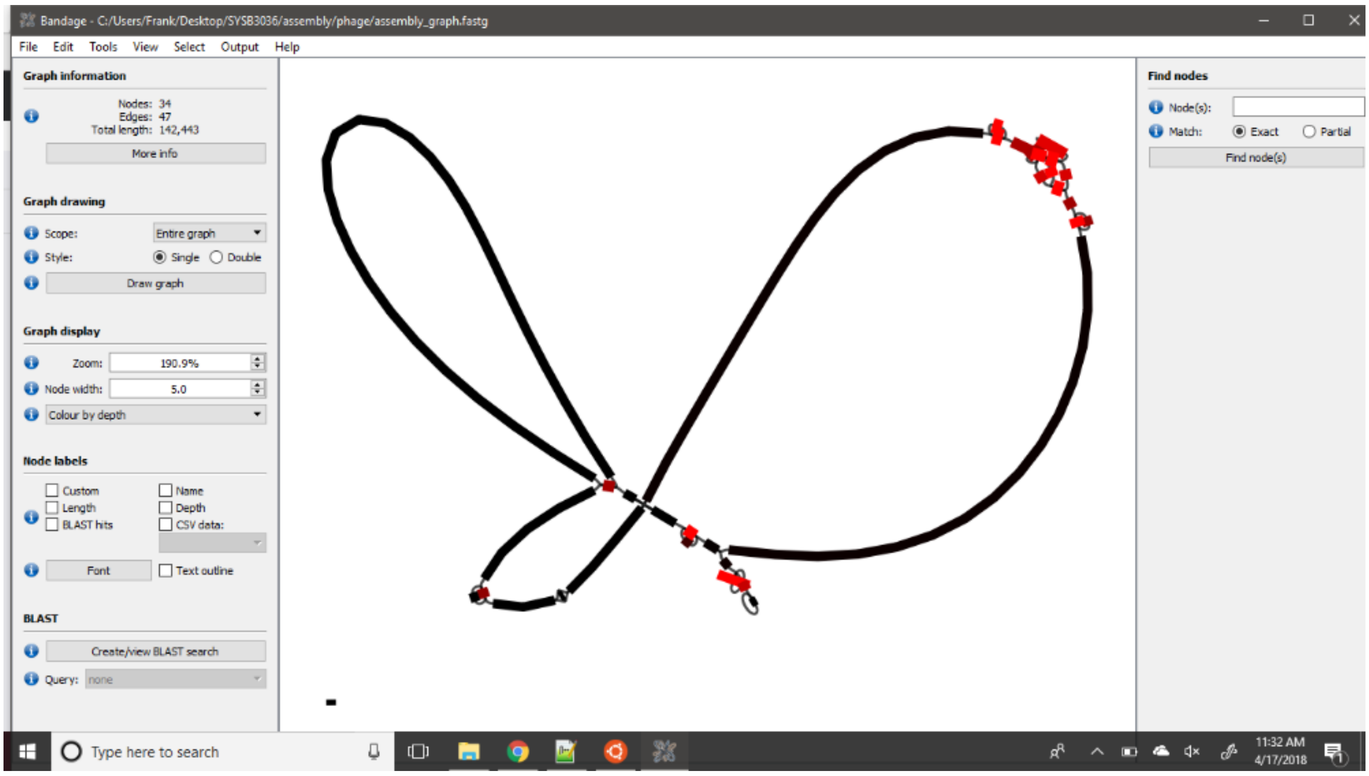
BLAST
Query: none

Find nodes
Match: [✓ Exact] Partial

Visualize the assembly

Step 4.

You can also color the De Bruijn graph by depth of


Bandage - C:/Users/Frank/Desktop/SYSB3036/assembly/phage/assembly_graph.fastg

Graph information

Nodes: 34
Edges: 47
Total length: 142,443

Graph drawing

Scope: Entire graph
Style: [✓ Single] Double

Graph display

Zoom: 190.9%
Node width: 5.0
Colour by depth

Node labels

Custom, Length, BLAST hits, Name, Depth, CSV data:, Text outline

BLAST

Query: none

Find nodes

Node(s):
Match: [✓ Exact] Partial

coverage on the left-hand side. This can give you an idea of if high-coverage repetitive elements might be causingn problems. This should look like this:

Visualize the assembly

Step 5.

Now go back to the SPAdes command and play around with the k-mer length. How does k-mer length change the topology of the de Bruijn graph?

What about the number of reads used- what if you downloaded 10X fewer or 10X more reads and used those for assembly? How does that change the topology of the de Bruijn graph?