Mar 19, 2019

Working

SYSB 3036 W09: Short Read Assembly

Frank Aylward[1]

[1]Virginia Tech

dx.doi.org/10.17504/protocols.io.v4se8we

Frank Aylward
Virginia Tech

ABSTRACT

PROTOCOL STATUS

Working

We use this protocol in our group and it is working

SAFETY WARNINGS

1 First we need to download the raw read data, which will be in FASTQ format. This is how the data looks when it comes right off of a sequencer.

Raw sequencing read data is quite large and can be unwieldy, so that's why we need the sra-toolkit to download it. This tool is designed to interface with the NCBI Sequence Read Archive to get the data. We can use the fastq-dump sub-command in the sra-toolkit:

fastq-dump -X 10000 --split-3 SRR6764339

A few things about this command:
- The purpose of this command is to download FASTQ files of raw sequencing reads that belong to a specific genome project, and in this case SRR6764339 corresponds to the Staphylococcus phage we want to analyze. FASTQ files are similar to FASTA files, but in addition to sequence information they also contain quality score information. This is because the sequencer is not always 100% confident in the bases that it calls, and we want to be able to discern between low-quality and high-quality sequence for our downstream analysis.

-Here "-X 1000" specifies that we only want 1000 reads. This is quite a small number, but it will suffice for what we need.

-The "--split-3" flag specifies that we want the reads split into three files- one for forward reads, one for reverse reads, and one for unpaired reads. These reads come from an Illumina sequencing project, and Illumina reads typically are "paired-end", meaning that a single DNA fragment was sequenced from both ends. It is useful to know if two sequence reads were from the same DNA fragment, so this information is retained in the FASTQ files.

After this runs you should see two new files, SRR6764339_1.fastq and SRR6764339_2.fasta. All the reads are paired, so there is no unpaired read file.

2 Now that we have the raw sequence reads we can assemble with SPAdes.

Before we start with SPAdes we need to update our PATH so we can see the program. I have installed the tool on my personal partition and you need to enter into a few commands so that your terminal knows where everything is located.
The code for this is:

PATH=$PATH:/home/faylward/SPAdes-3.13.0-Linux/bin

and then:

export PATH

After this we can take a look at how to run SPAdes.
First let's take a look at the SPAdes options:

spades.py

Check out the options and think about what kind of input command we might want.

3 Let's get started with a simple command.

spades.py -1 SRR6764339_1.fastq -2 SRR6764339_2.fastq -o phage_17 -k 17 &> log.txt

-Here we are specifying the two FASTQ input files with the -1 and -2 flags.
-We specify an output folder that we want the results to go into using the "-o" flag.
-We can specify the k-mer length that we want to use for de Bruijn graph construction with the -k flag. SPAdes usually tries a variety of k-mers and settles on the best one, but here we will use only 21 for demonstration purposes.
- I put "&> log.txt" to capture the standard output and standard error streams into the log.txt file. This isn't necessary, but it makes sure we can go back at the log file later.

You can take a quick look in the log.txt file if you want to see what SPAdes was saying while it was running.

more log.txt

4 Now let's navigate into the phage_21 folder and see what the output files look like.

cd phage_17
ls -lh

You should see a variety of files that SPAdes created. The important ones we want to look at here are "scaffolds.fasta" and "assembly_graph.fastg".

The scaffolds that were assembled are in the scaffolds.fasta file. Let's take a look with seqkit:

You should see 20 or so scaffolds that range in size from ~15 nt to ~60,000 nt. Scaffolds smaller than 1,000 nt are not that useful, but at least we have a few large ones. Not that bad for a first-pass assembly, especially considering we are only using 10,000 reads. What we would really like is a complete genome.

5 The other interesting file that spades created is the assembly_graph.fastg file. The G in FASTG stands for Graph, so we have a de Bruijn graph that we can visualize here.

To visualize this we will use a tool called Bandage. This tool is a bit different than the ones we have usually been using since it will not run in the command line. Instead we will download it via the command line, unzip the file, navigate to the file in the File Explorer, and then double-click on the Bandage file to start it. A GUI (Graphical User Interface) will pop up and we will work there.

The code for downloading and unzipping the tool is:

wget https://github.com/rrwick/Bandage/releases/download/v0.8.1/Bandage_Ubuntu_dynamic_v0_8_1.zip

and

unzip Bandage_Ubuntu_dynamic_v0_8_1.zip

You should see a file called Bandage if you navigate to your current folder in the File Explorer. Double click on it.
A program should pop up with a menu on the top band. Click on File->Load graph and navigate to the FASTG file that SPAdes created. Then click on Draw Graph in the middle of the menu on the left.
You should see something like this:

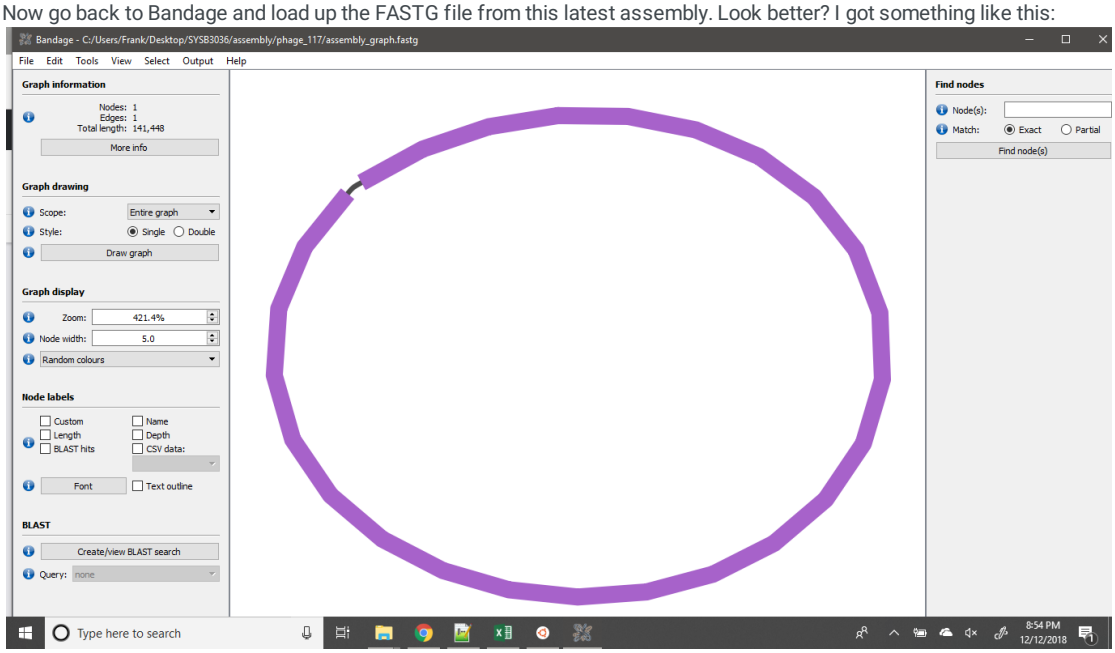
Notice that it's a big tangles mess, but it seems to be in one big chunk for the most part. The individual scaffolds we saw in the previous step are from this big chunk. SPAdes did a pretty good job of traversing the de Bruijn graph and pulling out linear scaffolds, but it couldn't pull out a full linear contiguous genome given how many knots and bubbles and in the graph. That's why we wound up with several scaffolds and not just one.
You can play around with the settings as well. If you color by depth of coverage (middle of left left panel) you will see the knots tend to be in high coverage areas. This is becauses these are repetitive regions where the same k-mers turn up in different parts of the genome. This creates complications for de Bruijn graph assembly.

6 A k-mer size of of 17 is quite short, so we can run SPAdes again with a longer k-mer size and see if that helps resolve a complete genome. So we will move back out of the phage_17 folder and re-run SPAdes with a much longer k-mer length.

cd ..

spades.py -1 SRR6764339_1.fastq -2 SRR6764339_2.fastq -o phage_117 -k 117 &> log2.txt

Now go back to Bandage and load up the FASTG file from this latest assembly. Look better? I got something like this:



So it seems the longer k-mer length really helped resolve those knots and bubbles. This is becuase longer k-mers are less likely to be repeated in different regions of the genome, since they are higher complexity regions. It's unlikely that a 117-bp sequence will be present in different regions, but it's much easier for a 17-bp sequence to be present in multiple locations.

If you look at the scaffolds.fasta file for the new assembly you should also see almost all of the sequence is in one large scaffold. So we have our complete genome!