Lab 3 – Part 1
Genome Assembly and Analysis

Today we are going to see how to assemble a genome, look at some genome assemblies and start trying to assess how good an assembly is.

We're going to see how to assemble an *E. coli* genome from publically available data using the program SPAdes(Nurk et al., 2013). SPAdes is a relatively recent addition to the pantheon of assemblers. It's designed by the Center for Algorithmic Biotechnology in St. Petersburg, Russia. It was originally designed to tackle the problems that arise in assembling genomes from single-cell genomic data. However, it's really good at prokaryotic genome assembly, so over time they have adapted it for that.

SPAdes is a stand-alone program in Python, so no frustrations with R, though plenty of others.

To start open Terminal

You will find it in → Applications→ Utilities

Once in Terminal we are going to use these Unix commands a lot:

cd (change directory)
to go up a directory, follow with ..
to go down a directory, type in directory name
ls (list)
this is used to list things in the directory
ls –lrt
this lists the things in the directory with more info, including size and permissions

Now, let's get into your Udrive

1. $ cd ..
2. $ ls
   If this list contains Volume skip to line 5
3. $ cd ..
4. $ls
   This list should contain Volume, if not, jump back to 3
5. $cd /Volume/udrive
6. $ ls
   This should give you a list of folders in your udrive. Navigate to the folder Ecoli_Assembly in the Assembly folder.

In this folder you will find a data folder. See what's inside there (in Terminal or Finder). You should find two fastq.gz files. These files are the data files. These are from paired-end reads. The file with the _1 is the forward (first) sequencing reaction the file with the _2 is the reverse (second) sequencing reaction.

We need to tell spades what each of these files is. We do that with the prefixes --pe1-1 for the first set and --pe1-2 for the second. We also need to tell it where to put the data in the end. For that we use the prefix -o

The command looks like this (DO NOT INPUT UNLESS TOLD TO BY INSTRUCTOR):
$ spades.py --pe1-1 s_6_1.fastq.gz --pe1-2 s_6_2.fastq.gz -o SPAdes_test

You can add other types of data too. You can add up to 9 data sets in the command line by substituting the number after the --pe for each data set. You can use mate-pairs by substituting mp for pe. You can add PacBio or Oxford Nanopore data by using the prefixes --pacbio and -- nanopore respectively. In the data there is also a file that is the PacBio fastq for the same strain of *E.coli.*

Let's see how big these data are.
$ ls –lrt
How big are they?

This takes HOURS to run on our computers! You are free to try it if you have time. This program is a memory and CPU hog. It only runs on Linux or Mac. You can download it elsewhere if you want to try this. It is free, and easy to install. Though if you install it, you will need to add it to the PATH or call it with its full directory as a heading.

For today, instead, we are going to walk through what it does and then look at some preassembled data and do some analysis of these assemblies.

Pre-assembled data (AKA Julia Child's Oven)
In the Assembly folder you will find a folder labeled SPAdes. In that folder is a folder labeled Sau_test. This is the result of a SPAdes assembly I did a few months ago using Illumina paired end reads of the *Staphylococcus aureus* genome. These data were used as part of the GAGE project (http://gage.cbcb.umd.edu/index.html). In the Sau_test folder there are a number of things. Let's look at them.

There are a series of folders labeled K*. In these folders are the contigs for the intermediate assemblies using kmers of the number given in the folder name. These are generated in order of size, starting with 21-mers. Theses are used to generate the net step. In the end there are a series of files in the main folder. Below is the explanation of these files given in the SPAdes manual.

```
scaffolds.fasta — resulting scaffolds (recommended for use as
   resulting sequences)
contigs.fasta — resulting contigs
assembly_graph.fastg — assembly graph
contigs.paths — contigs paths in the assembly graph
scaffolds.paths — scaffolds paths in the assembly graph
before_rr.fasta — contigs before repeat resolution
```

Let's do some analysis on these.

Analyze for contiguity

1. Go to http://quast.bioinf.spbau.ru/
    a. Input the scaffolds.fasta file
    b. Open the report

Let's look at the data.
How many contigs are there? How long is the longest? How close is that to the size of the genome of *S. aureus*? What is the total length of all the contigs?

Let's look at how these contigs come together to form the genome. Click on the View in Icarus button.
Here we can visually see what the N50 and N75 are.
N50 is the contig (ordered largest to smallest) that makes up 50% of the assembly.
N75 is the contig (ordered largest to smallest) that makes up 75% of the assembly.

What is the N50? The N75? Why is it useful for comparing between different assemblies/assemblers on the same data or genome? Why is this not a good measure if you are comparing between different genomes?

Let's visualize our assembly. Remember these are De Bruijn graphs.

1. Download software I forgot to have them install. You may need to reinstall if you want to do it again.
    a. Go to http://rrwick.github.io/Bandage/
    b. Download the appropriate version
    c. Install
        i. If on Mac, unzip by double-clicking and drag folder to Applications
2. Start Bandage

Before we look at our data, let's look at some fun *E. coli* data.

3. In the File Menu choose Load Graph → Select the file E_coli_LastGraph in the Data for Bandage folder.
4. In the graph drawing panel on the left, select the scope as entire graph and click draw graph.
5. Let's look at specific regions. On the right hand panel use the find node to find node 18. Click on it,
6. Change the scope of the graph to around node 18 with a distance of 10. Draw with double nodes.
7. In the Graph Display choose color by contiguity. Determine the contiguity from 18+ node.
8. Together we'll look at this section of the graph and try to identify it.
9. Repeat what we just did with the section around node 4+. What is this section of this graph?

What is the big section in the middle? Why is it so complicated? How would you go about trying to get the actual genome without any additional data? If you could get

additional data, what would it be? Why? What do you think the small, unconnected contigs at the bottom are? Why do you think that (what data are you basing that on)?

Paths through the graph:

As we discussed in class, we assemble these with graphs. These graphs of k-mers eventually get consolidated into contigs. However, there are often places where we can't decide what the next contig is, so we end up with a section of a graph that looks like this:

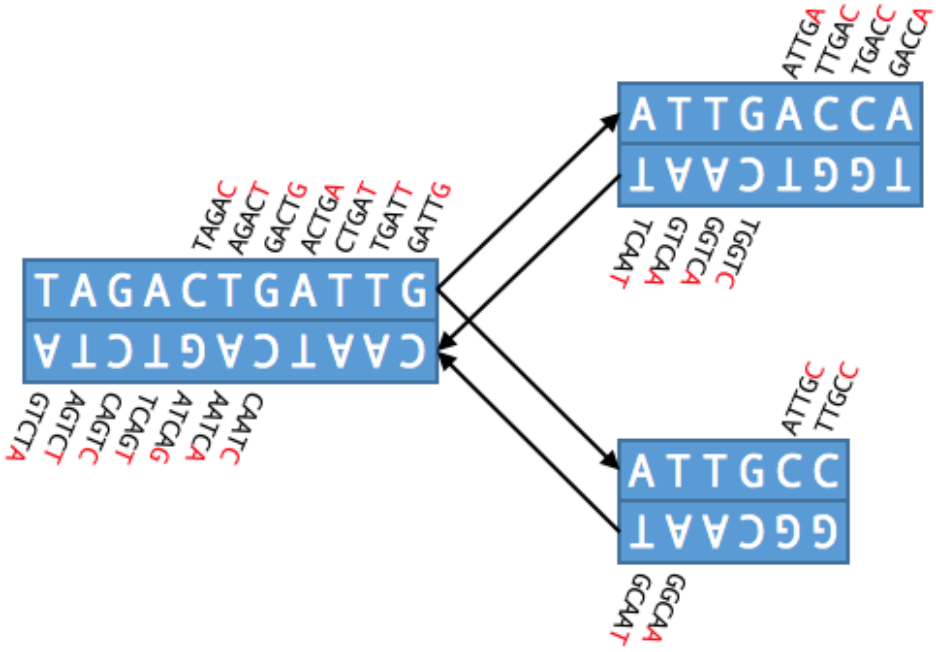

Figure 1: Visual representation of how SPAdes creates the assembly graph.
Image from Bandage manuals
(https://github.com/rrwick/Bandage/wiki/Assembler-differences)

As Bandage knows how to read this, it allows us to output the sequence we want into a fasta file we can use. Bandage does have the ability to use BLAST within the program, but we don't have the standalone BLAST installed on our machines, so we'll have to do it manually.

Now let's look at the *S. aureus* data.

1. Load the graph (assembly_graph.fastg).
   What do you see? How is this different than the *E. coli* data?
2. Are there any extra-genomic regions in this data?
   How did you determine this?
3. Trace through the path starting at node 207-.
   Where does it go? What happens here? There are a couple of loops. Can you resolve them? Hypothesize what you think the structure of this section of the genome is. See if you can confirm this using resources available to you. If this were a previously unsequenced organism (a newly discovered fish for example) how would your confirmation techniques change?

References:

Nurk, S., Bankevich, et al.; (2013). Assembling Genomes and Mini-metagenomes from Highly Chimeric Reads.

In M. Deng, R. Jiang, F. Sun, & X. Zhang (Eds.), *Research in Computational Molecular Biology: 17th Annual International Conference, RECOMB 2013, Beijing, China, April 7-10, 2013. Proceedings* (pp. 158–170). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-37195-0_13