

# Bacterial Genome Assembly

Chris Fields

PowerPoint by Saba Ghaffari  
Edited by Negin Valizadegan 2021-2022

# Introduction

## Exercise

1. Perform a bacterial genome assembly using PacBio HiFi data.
2. Evaluation and comparison of different datasets and parameters.
3. View the best assembly in Bandage.

# Premise

1. We have sequenced the genomic DNA of a bacterial species that we are very interested in. Using other methods, we have determined that its genome size is approximately 1.7 Mbp.
2. We chose to use Pacific Biosciences HiFi technology for performing this analysis because our genome of interest is relatively small and PacBio HiFi gives us both long reads (**700bp-20kbp**) that are 99% accurate.

# Dataset Characteristics

Dataset #	FQ Name	File Size
1	dataset1.fastq.gz	144 MB
2	dataset2.fastq.gz	36 MB
3	dataset3.fastq.gz	18 MB
4	dataset4.fastq.gz	7.1 MB

The assembler can read compressed files, so the FASTQ\* file is compressed to save disk space. We will see what the data look like in a bit.

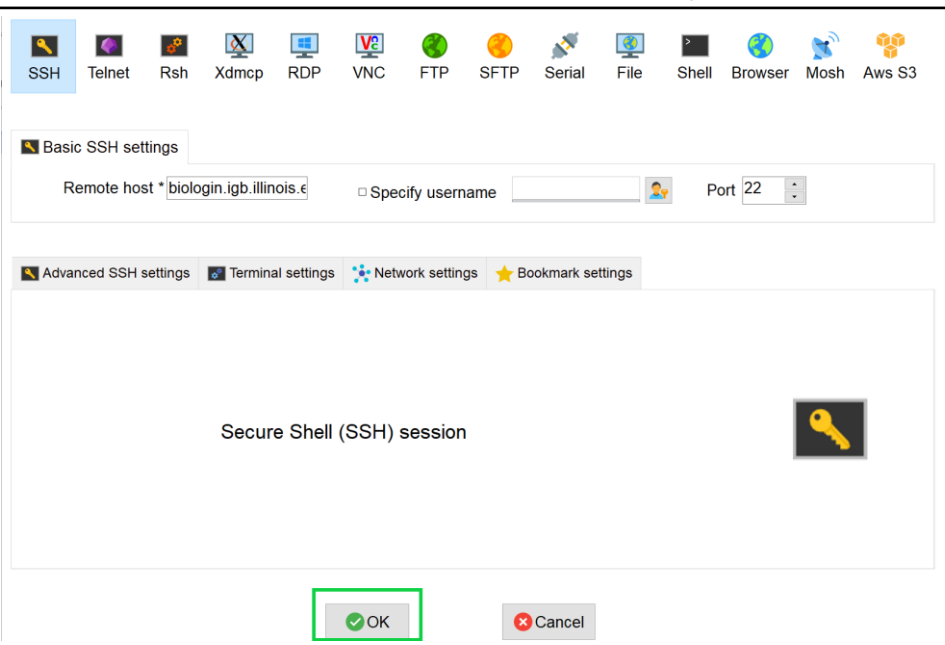
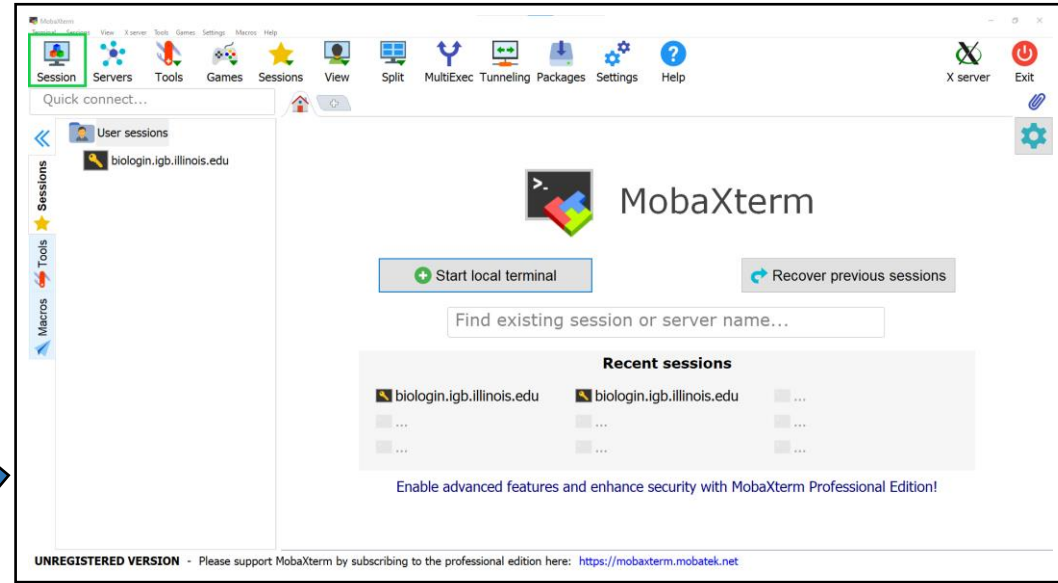
\* FASTQ -> “FASTQ format is a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores. Both the sequence letters and quality scores are encoded with a single ASCII character for brevity”. Excerpted from <http://en.wikipedia.org/wiki/Fastq>

# Start the VM

- Follow instructions for starting VM (This is the Remote Desktop software).
- The instructions are different for UIUC and Mayo participants.
- Find the instructions for this on the course website under Lab set-up:  
<https://publish.illinois.edu/compgenomicscourse/2022-schedule/>

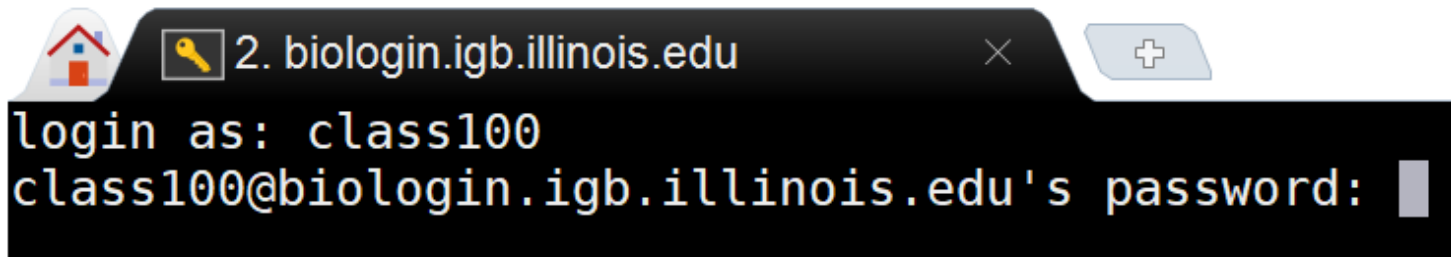
# Step 0A: Accessing the IGB Biocluster

- Open **MobaXterm** on your desktop
- In a new session, select **SSH** and type the following host name:  
`biologin.igb.illinois.edu`
- Click **OK**



# Step 0A: Accessing the IGB Biocluster

- Enter login credentials assigned to you.
- Example username: **class100**.
- You will not see any characters on screen when typing in password. Just type it.



A screenshot of a terminal window. The title bar shows a home icon, a key icon, and the text "2. biologin.igb.illinois.edu". The terminal content displays the following text:

```
login as: class100
class100@biologin.igb.illinois.edu's password: █
```

# Step 0A: Accessing the IGB Biocluster

If you have done this before, just double-click on the session you created once and type username and password.



MobaXterm

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help

Quick connect...

User sessions

- biologin.igb.illinois.edu

biologin.igb.illinois.edu

Type: SSH  
Host: biologin.igb.illinois.edu  
User:  
Port: 22

Start local terminal

Rec

Find existing session or server name

Recent sessions

- biologin.igb.illinois.edu
- ...
- ...
- ...

Enable advanced features and enhance security with MobaXterm

VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings

Quick connect...

User sessions

- biologin.igb.illinois.edu

2. biologin.igb.illinois.edu

login as: █



# Step 0B: Lab Setup

The lab is located in the following directory:

```
/home/classroom/hpcbio/mayo_workshop/02_Genome_Assembly
```

This directory contains the initial data and the finished version of the lab (i.e. the version of the lab after the tutorial). Consult it if you unsure about your runs.

You don't have write permissions to the lab directory. Create a working directory of this lab in your home directory for your output to be stored. Note `~` is a symbol in Unix path referring to your home directory.

```
$ mkdir ~/02_Genome_Assembly
```

Copy the following script to your local directory line by line and run it. You do not need to copy the \$.

```
$ cp /home/classroom/hpcbio/mayo_workshop/02_Genome_Assembly/src/setup.sh ~  
$ bash setup.sh  
$ cd ~/02_Genome_Assembly
```

# Assembly

We will be using the hifiasm assembler, which is a very fast assembler developed for highly accurate long reads such as PacBio HiFi.

# Step 1: Get sequence statistics

We know the size of the files, but we don't know **how many reads** there are, what the **maximum and minimum length** is, **total bases**, and so forth. This would be good to know, since we want to make sure we have adequate coverage for an assembly.

Once you are logged into the biocluster with your classroom account, **type** the following commands.

```
$ sbatch run-fastq-stats.slurm.sh  
$ squeue -u <used_id>
```

The first command ('**sbatch**') submits a **job script** (in this case, **run-fastq-stats.slurm.sh**) to the cluster for running.

The second command will check the status of the job; it should be very fast so this may show nothing if the job is already finished.

# Step 1: Get sequence statistics

What's in the job script?

```
#!/bin/bash
#SBATCH -n 1
#SBATCH --mem=4000
#SBATCH -p classroom
#SBATCH -J fastq-stats
```

Tells the cluster 'job manager' what resources you want (1 core, 4GB memory, run on the 'classroom' nodes, and name the job 'fastq-stats')

```
### Load Modules
module load seqkit/0.12.0
```

Load the software. We are using a tool called 'seqkit' to generate some basic stats on the sequence data

```
### Run app on file
seqkit stats dataset*.fastq.gz > stats.txt
```

Run the tool on all the files ('dataset\*.fastq.gz') and redirect the output ('>') to a text file

# Step 1: Get sequence statistics

You can use the Linux 'cat' or 'less' command to get a peek at the 'stats.txt' file. Run the next command.

```
$ cat stats.txt
```

file	format	type	num_seqs	sum_len	min_len	avg_len	max_len
dataset1.fastq.gz	FASTQ	DNA	7,286	75,079,593	793	10,304.6	28,055
dataset2.fastq.gz	FASTQ	DNA	3,611	37,452,304	793	10,371.7	27,426
dataset3.fastq.gz	FASTQ	DNA	1,800	18,723,557	793	10,402	27,426
dataset4.fastq.gz	FASTQ	DNA	724	7,471,070	1,010	10,319.2	25,180

Note the sum length for each file; the expected size of our genome is supposed to be **1.7 Mbp**. So, what is the expected sequence coverage for each?

(Hint: 1.7Mbp is 1,700,000 bases)

# Step 2A: Run Assembly 1

For this 1<sup>st</sup> assembly we use **dataset1.fastq.gz** (144MB). Once you are logged into the biocluster with your classroom account, type the following command:

```
$ sbatch run-assembly1.slurm.sh  
$ squeue -u <userID>
```

Again, the first command (**'sbatch'**) submits a **job script** (in this case, **run-fastq-stats.slurm.sh**) to the cluster for running. The second command will check the status of the job (this may be a bit slower, say a couple of minutes, depending on the computer load at the time). While this is running, let's look at the script.

# Step 2A: Run Assembly 1

This script is a little more complex, we are running several steps. **Note the comments!**

You can type 'less run-assembly1.slurm.sh' to see the text in your script (hit 'q' to exit).

```
#!/bin/bash
#SBATCH -n 4
#SBATCH --mem=8000
#SBATCH -p classroom
#SBATCH -J hifiasm

### Load Modules
module load hifiasm/0.5-IGB-gcc-8.2.0

# Step 1: make working directory
mkdir dataset1

# Step 2: run assembly
hifiasm -o dataset1/full.asm -f 0 -t $SLURM_NPROCS -a 3 dataset1.fastq.gz 2> dataset1/full.log
```

Tells the cluster 'job manager' what resources you want (4 cores, 8GB memory (memory listed in MB), run on the 'classroom' nodes, and name the job 'hifiasm')

Load the software. We are using 'hifiasm' to assemble the data.

## Step 2B: Run Assemblies 2, 3 & 4

For this 2<sup>nd</sup> run, we will assemble both **dataset2.fastq.gz** and **dataset3.fastq.gz**. The coverage for these are lower (dataset1 is ~44x, dataset2 is ~22x, dataset3 is ~11x, and dataset4 is ~4x).

For the last assembly (dataset4) we are really pushing the boundaries for adequate coverage for this tool (preferably above 25x).

```
$ sbatch run-assembly234.slurm.sh  
$ squeue -u <userID>
```

This may take about 2 minutes, or may finish quicker than that. We can look over the script again while this is running.



# Step 2B: Run Assemblies 2, 3 & 4

This script is a little more complex, we are running several steps. *Note the comments!*

You can type 'less run-assembly234.slurm.sh' to see the text in your script (hit 'q' to exit).

```
#!/bin/bash
#SBATCH -n 4
#SBATCH --mem=8000
#SBATCH -p classroom
#SBATCH -J hifiasm
```

Tells the cluster 'job manager' what resources you want (4 cores, 8GB memory, run on the 'classroom' nodes, and name the job 'hifiasm')

```
### Load Modules
module load hifiasm/0.5-IGB-gcc-8.2.0
```

Load the software. We are using 'hifiasm' to assemble the data.

```
# Step 1: make working directories (yes you can do more than one)
mkdir dataset2 dataset3 dataset4
```

```
# Step 2: run assemblies
```

```
hifiasm -o dataset2/half.asm -f 0 -t $SLURM_NPROCS -a 3 dataset2.fastq.gz 2> dataset2/half.log
hifiasm -o dataset3/quarter.asm -f 0 -t $SLURM_NPROCS -a 3 dataset3.fastq.gz 2> dataset3/quarter.log
hifiasm -o dataset4/tenth.asm -f 0 -t $SLURM_NPROCS -a 3 dataset4.fastq.gz 2> dataset4/tenth.log
```

## Step 3: Convert formats

**This is a really common task:** you have format 'X' and need format 'Y'. In this case, hifiasm assemblies are in a reference graph format called **GFA**. We will look at one of these in a bit, but it is pretty complex.

We want a simpler file that represents the final assembly in **FASTA** format. We need this to gather assembly stats.

```
$ sbatch run-conversion.slurm.sh
```

# Step 3: Convert formats

Want to convert all files. This script will run all of them.

```
#!/bin/bash
#SBATCH -n 1
#SBATCH --mem=2000
#SBATCH -p classroom
#SBATCH -J GFA-to-FASTA
```

```
### Load Modules
module load gfatools/0.4-IGB-gcc-4.9.4
```

```
# Convert GFA (reference graph) into FASTA (one at a time)
gfatools gfa2fa dataset1/full.asm.p_ctg.gfa > dataset1/full.asm.p_ctg.fasta
gfatools gfa2fa dataset2/half.asm.p_ctg.gfa > dataset2/half.asm.p_ctg.fasta
gfatools gfa2fa dataset3/quarter.asm.p_ctg.gfa > dataset3/quarter.asm.p_ctg.fasta
gfatools gfa2fa dataset4/tenth.asm.p_ctg.gfa > dataset4/tenth.asm.p_ctg.fasta
```

Tells the cluster 'job manager' what resources you want (1 core, 2GB memory, run on the 'classroom' nodes, and name the job 'GFA-to-FASTA')

Load the software. We are using 'gfatools' to convert formats

## Step 3: Convert formats

Then **type** the below to list the files.

```
$ ls -l dataset*/*.fasta
```

```
-rw-rw-r-- 1 cjfields cjfields 1691621 May 23 2020 dataset1/full.asm.p_ctg.fasta  
-rw-rw-r-- 1 cjfields cjfields 1673187 May 23 2020 dataset2/half.asm.p_ctg.fasta  
-rw-rw-r-- 1 cjfields cjfields 1461613 May 23 2020 dataset3/quarter.asm.p_ctg.fasta  
-rw-rw-r-- 1 cjfields cjfields 1370488 May 23 2020 dataset4/tenth.asm.p_ctg.fasta
```

# Results:

The following instructions guide you to the location of the results. As the needed output for the rest of the lab is provided in the VM, you could skip this slide.

You can find the results of all previous runs in folders **dataset1**, **dataset2**, **dataset3**, and **dataset4**:

```
~/02_Genome_Assembly
```

You can go to each folder by typing the following command:

```
cd ~/02_Genome_Assembly/[Folder-Name]
```

To see the files in the each of these directories, type “ls” command.

Make sure that you return to your previous working directory for the rest of the lab by typing

```
cd ~/02_Genome_Assembly
```

The description of the results is provided in the next slide.

# HiFiAsm Output: Legend

Once everything is finished you will have numerous files. Key ones are highlighted below:

File	Meaning
<i>prefix.p_ctg.gfa</i>	Primary assembly (a <i>haploid</i> representation). GFA format
<i>prefix.p_ctg.fasta</i>	Primary assembly (a <i>haploid</i> representation). FASTA format
<i>prefix.a_ctg.gfa</i>	Alternate assembly contig graph (alleles not in primary assembly). GFA format
<i>prefix.r_utg.gfa</i>	Raw <a href="#">unitig</a> graph. GFA format. Keeps all haplotype information, including somatic mutations and recurrent sequencing errors.

# Assembly Evaluation

What metrics do we use to evaluate the assembly?

# Assembly Evaluation: Skeleton

	dataset1	dataset2	dataset3	dataset4
Genome Size (Mb)				
N50 (Mb)				
Number of contigs				
Longest contig (Mb)				
Shortest contig (bp)				
Mean contig size (Kb)				
GC content				

## Definition of N50:

“Given a set of contigs, each with its own length, the *N50* length is defined as the shortest sequence length at 50% of the genome. It can be thought of as the point of half of the mass of the distribution. For example, 9 contigs with the lengths 2,3,4,5,6,7,8,9,and 10, their sum is 54, half of the sum is 27. 50% of this assembly would be 10 + 9 + 8 = 27 (half the length of the sequence). Thus the N50 = 8”.

Excerpted from [https://en.wikipedia.org/wiki/N50,\\_L50,\\_and\\_related\\_statistics#N50](https://en.wikipedia.org/wiki/N50,_L50,_and_related_statistics#N50)



# Step 4A: Evaluate Assembly 1

We will evaluate the results of the 1<sup>st</sup> assembly (dataset 1) using a perl script:

`assemblathon_stats.pl`

```
# Use a perl script to determine the various metrics for Assembly 1
$ perl assemblathon_stats.pl ~/02_Genome_Assembly/dataset1/full.asm.p_ctg.fasta
```

# Step 4B: Output of Assembly 1 Evaluation (top)

```
Number of scaffolds          1
Total size of scaffolds     1663877
Longest scaffold            1663877
Shortest scaffold           1663877
Number of scaffolds > 1K nt  1 100.0%
Number of scaffolds > 10K nt 1 100.0%
Number of scaffolds > 100K nt 1 100.0%
Number of scaffolds > 1M nt  1 100.0%
Number of scaffolds > 10M nt 0  0.0%
Mean scaffold size          1663877
Median scaffold size        1663877
N50 scaffold length         1663877
L50 scaffold count          1
scaffold %A                  30.50
scaffold %C                   19.49
scaffold %G                   19.68
scaffold %T                   30.34
scaffold %N                    0.00
scaffold %non-ACGTN          0.00
Number of scaffold non-ACGTN nt 0

Percentage of assembly in scaffolded contigs 0.0%
Percentage of assembly in unscaffolded contigs 100.0%
Average number of contigs per scaffold 1.0
Average length of break (>25 Ns) between contigs in scaffold 0
```

# Step 5: Evaluate Assemblies 2, 3 and 4

We will evaluate the results of the stringent assembly using a perl script:

`assemblathon_stats.pl`

```
# Use a perl script to determine the various metrics for Assembly 2-4
perl assemblathon_stats.pl ~/02_Genome_Assembly/dataset2/half.asm.p_ctg.fasta

perl assemblathon_stats.pl ~/02_Genome_Assembly/dataset3/quarter.asm.p_ctg.fasta

perl assemblathon_stats.pl ~/02_Genome_Assembly/dataset4/tenth.asm.p_ctg.fasta
```

## Step 6: Compare Assembly Statistics

	dataset1	dataset2	dataset3	dataset4
Genome Size (Mb)	1.663877	1.645745	1.437603	1.347800
N50 (Mb)	1.663877	1.645745	0.799713	0.105349
Number of contigs	1	1	4	18
Longest contig (bp)	1,663,877	1,645,745	799,713	325,120
Shortest contig (bp)	1,663,877	1,645,745	14,705	17,658
Mean contig size (bp)	1,663,877	1,645,745	359,401	74,878
GC content	39.17	39.19	39.03	39.17

Genome size is ~1.7 Mb; two of these assemblies are close. The genome coverage (the number of times each base is covered by a read) for dataset1 is about 44x, dataset2 is 22x, dataset3 is 11x and dataset4 is 4x. The lower the coverage the fewer bases recovered and more fragmented the genome.

Also note how many contigs each has; datasets 1 & 2 have fully assembled chromosomes!

Exit MobaXterm, by either closing the window or typing 'exit' in the command prompt.

# Assembly Visualization

Use Bandage to visualize the assembly graph.

**We are going to do visualization on VM**

# Step 0: Local Files

For viewing and manipulating the files needed for this laboratory exercise, the path on the VM will be denoted as the following:

**[course\_directory]**

We will use the files found in:

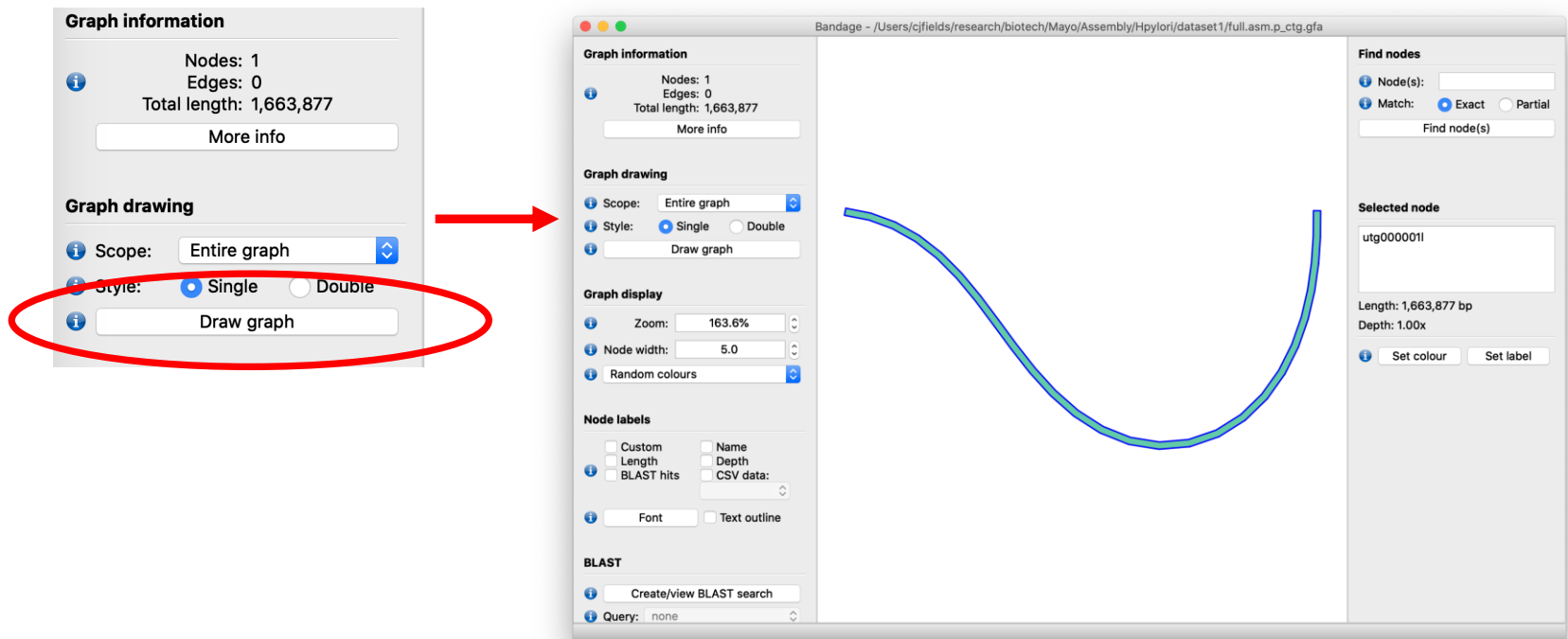
**[course\_directory]\02\_Genome\_Assembly\results**

**[course\_directory]= Desktop\Labs    **UIUC****

**[course\_directory]= Desktop\VM    **Mayo****

# Step 1: Assembly Visualization

- Open Bandage shortcut on Desktop
- Under **File**, go to **Load Graph** and open the `full.asm.p_ctg.gfa` file in the **results** directory:  
**[course\_directory]/02\_Genome\_Assembly/results/dataset1/**
- Click on the 'Draw Graph' button on left panel. Select the single node.

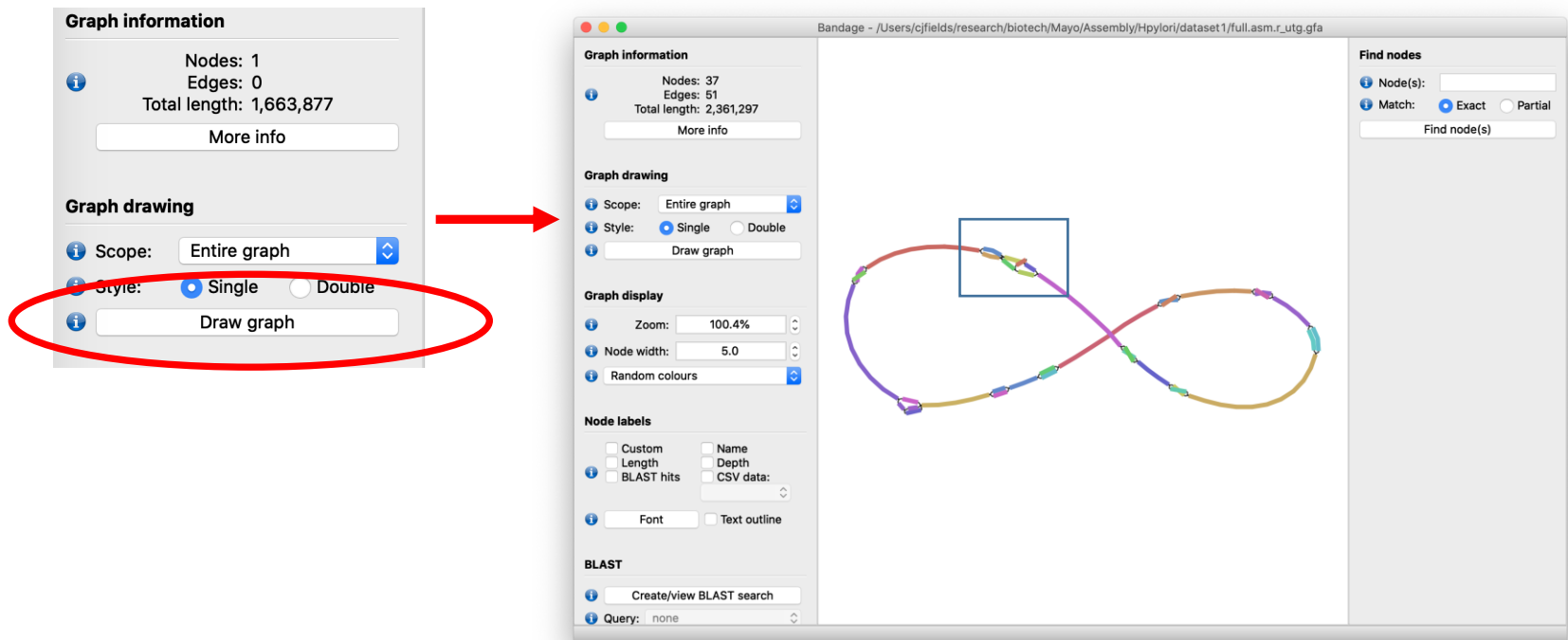


<https://rrwick.github.io/Bandage/>



# Step 1: Assembly Visualization

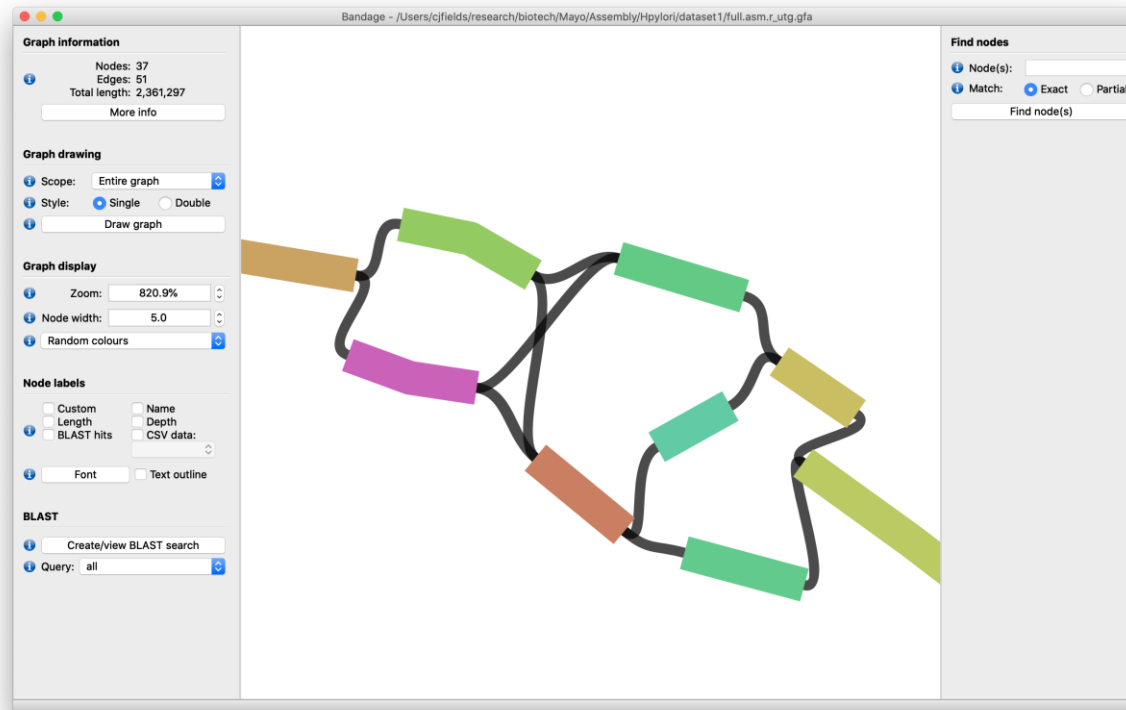
- Kind of boring...
- Now load in the *raw* data from file **full.asm.r\_utg.gfa** (which also contains sequence bubbles likely due to errors).



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

# Step 1: Assembly Visualization

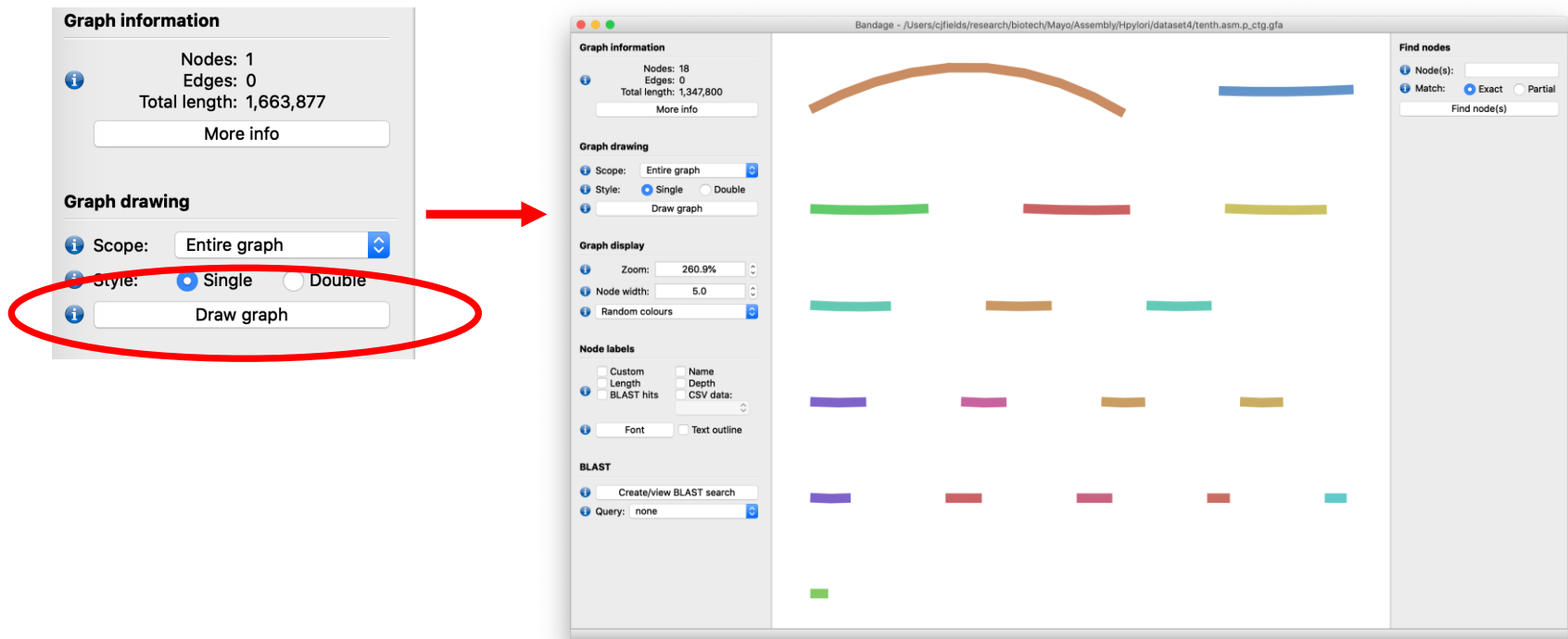
Zoom in using the Zoom option under **Graph display**:



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

# Step 1: Assembly Visualization

- Now load in the primary data `tenth.asm.p_ctg.gfa` from dataset4 (the fragmented example)  
[course\_directory]/02\_Genome\_Assembly/results/dataset4/



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