Botany 2020 Genome Assembly

This module was written by Jacob Landis (jbl256@cornell.edu) for the *de novo* genome assembly and annotation workshop for the Virtual Botany 2020 meeting. All of the data used here are publicly available from the Short Read Archive and other data repositories. This module uses the CyVerse framework and a virtual machine that has been created with all of the necessary programs already installed. The associated presentations and analyses scripts for the entire module can be found on GitHub (https://github.com/bcbc-group/Botany2020NMGWorkshop). If you have questions about anything, especially trying to run some of these with your own data and on your local server/cluster, please contact me.

Data gathering

Before we get into everything, let's make sure we have all the data needed for this part of the module. We will need to have the whole genome sequencing data, the index for error correction, and some pre-generated assemblies and BUSCO results (just to keep things moving). If you haven't done so already, please navigate to scratch (cd /scratch) and we'll create the necessary folders and get data moved over. You should already have icommands setup either on the webterminal or on your own machine.

If this is the first time using icommands on your partition, you may first need to log on to the webshell and type `iinit` following the prompts type in:

```
data.cyverse.org
1247
Username
iplant
your password
```

After you do this on webshell and you have icommands installed on your local machine (https://learning.cyverse.org/projects/data_store_guide/en/latest/step2.html), you can use icommands from your local machine while doing SSH to move the necessary files to your partition.

```
cd /scratch
iget -rPT /iplant/home/shared/Botany2020NMGWorkshop
```

Data subset

To make it possible for most of these steps to actually complete in the allotted time we have, we are going to use small subset of the raw data. It might seem that a lot of these analyses are fast to complete, but I would encourage you to try them on your own with the full data sets to get an idea of how long they might take even for a species with a small genome.

```
cd /scratch/Botany2020NMGWorkshop/raw_data/Ugibba/genome
```

```
zcat SRR10676752_1.fastq.gz | head -n 400000 > Polish_test_R1.fastq.gz
zcat SRR10676752_2.fastq.gz | head -n 400000 > Polish_test_R2.fastq.gz
```

```
gunzip Ugibba_PacBio_to_use.fastq.gz
head -n 200000 Ugibba_PacBio_to_use.fastq > Ugibba_PacBio_subset.fastq
```

Short-read assembly only

Using adapter cleaned Illumina short read data (paired end 150 bp) whole genome sequencing, the first step would be to use MaSuRCa to run an assembly with only Illumina reads. In the sr_config.txt of Masurca we will tell it only use the Illumina data. To run the program, there are only two lines of code to use:

```
cd /scratch
mkdir Masurca_Illumina
cd Masurca_Illumina
```

```
scp /scratch/Botany2020NMGWorkshop/Genome_assembly/scripts/sr_config_Illumina_only.txt /scratch/Masurca_Illumina
```

```
cat sr_config_Illumina_only.txt
```

```
/opt/MaSuRCA-3.4.1/bin/masurca sr_config_Illumina_only.txt
```

```
./assemble.sh
```

Even though we won’t use this for any of the downstream analyses, we will compare this assembly in terms of number contigs, N50, assembled base pairs, and presence of BUSCO genes to the other assemblies we run.

Long-read assembly only

There are many options for using long-read only data, both Nanopore and PacBio. A fast option is wtdbg2, which takes raw long-reads and uses a fuzzy De Bruijin graph to assemble. Another option which gives better results is Flye, which can take either raw or error-corrected long-reads. The program was built for raw reads, so those results are usually better than using error corrected data.

For wtdbg2, it is two lines of code again

```
cd /scratch
mkdir wtdbg2
cd wtdbg2

/opt/wtdbg2/wtdbg2 -x rs -g 1.0g -i /scratch/Botany2020NMGWorkshop/raw_data/Ugibba/genome/Ugibba_PacBio_subset.fastq -t 6 -fo Ugibba_PacBio

/opt/wtdbg2/wtpoa-cns -t 6 -i Ugibba_PacBio.ctg.lay.gz -fo Ugibba_PacBio.ctg.fasta
```

For Flye, which does a better job in the assembly but takes longer to complete, it is one line of code. Here we are going to be using the uncorrected PacBio data, 6 threads in the assembly, 8 iterations of polishing with minimap2. Polishing with the error prone long reads will not fix all of the problems that may arise, but it does help some. We also need to tell the program and estimated genome size, in our case, 78 MB. This analyses takes a while to run, so we are not going to run it here today. Using the same exact data as wtdbg2, this analyses will take about 10 hours to run compared to 10 minutes.

```
cd /scratch
mkdir Flye
cd Flye

nohup /opt/MaSuRCA-3.4.1/Flye/bin/flye --pacbio-raw /scratch/Botany2020NMGWorkshop/raw_data/Ugibba/genome/Ugibba_PacBio_subset.fastq --out-dir Flye_assembly_raw/ --genome-size 78m --threads 6 --iterations 8 > my.log 2>&1 &
echo $! > save_pid.txt
```

Error correcting/polishing

This section consists of two parts, error correcting the long-reads (either PacBio or Nanopore) with the much higher quality Illumina data prior to use in an assembly, such as for the MaSuRCA hybrid assembly we will do next, or using the Illumina data to polish a completed assembly such as with the Flye assembly we just did. Polishing an assembly that is done primarily with Illumina data will help fix some mistakes that were created during the assembly but it doesn't greatly change the overall quality in terms of percent of BUSCO genes found. However, polishing a long-read only assembly with the Illumina data can make a huge difference in terms of what genes can be found. We will start first with error correcting the long-reads to be used for an assembly. To do this we will use the program FMLRC (FM Index Long-Read Correction). The first step is to build a Multi-String Burrows Wheeler Transform (MSBWT) index from the Illumina reads. This takes a long time so we are going to skip it for now and we are providing the end results in the form of the msbwt.npy file.

```
pip install msbwt
```

```
msbwt cffq -p 8 Ugibba_error_correction_msbwt/ SRR10676752_1.fastq SRR10676752_2.fastq
```

This part can take quite a bit of time to complete, depending on the amount of Illumina data. Once this is complete, FMLRC will independently correct each read by comparing low frequency k-mers in the long-reads and replace them with the closet matching high frequency k-mer in the short read data. The format here is call the program, specify the directory where the FM-index resides, then specify the raw data to be correct, and the output file, finally specify the number of threads. The error correction step may only take a few minutes to run.

```
cd /scratch/Botany2020NMGWorkshop/Genome_assembly/Error_correction
```

```
/opt/fmlrc/fmlrc msbwt.npy /scratch/Botany2020NMGWorkshop/raw_data/Ugibba/genome/Ugibba_PacBio_subset.fastq
Ugibba_PacBio_subset_corrected.fasta -p 6
```

For polishing an assembled genome, there are many options. For one we will use POLCA (POLishing by Calling Alternatives), which is bundled with MaSuRCA. This approach maps the Illumina data to the assembly using bwa mem, then it calls variants using FreeBayes. Variants replace the assembled bases with the highest scoring alternative allele, resulting in a polished consensus. To save some time, instead of using all of the Illumina data we have, we are going to use a small subsample of 100,000 sequences.

```
cd /scratch
mkdir Polca_polishing
cd Polca_polishing
```

```
scp /scratch/Botany2020NMGWorkshop/Genome_assembly/Completed_assemblies/Ugibba_FLYE_assembly.fasta
/scratch/Polca_polishing
```

```
/opt/MaSuRCA-3.4.1/bin/polca.sh -a Ugibba_FLYE_assembly.fasta -r '
/scratch/Botany2020NMGWorkshop/raw_data/Ugibba/genome/Polish_test_R1.fastq.gz
/scratch/Botany2020NMGWorkshop/raw_data/Ugibba/genome/Polish_test_R2.fastq.gz' -t 6 -m 1000M
```

Hybrid approach

Now with the addition of the error correct long-reads, the next assembly will be a hybrid assembly in MaSuRCA using both the Illumina short-reads and the error corrected long-reads. In the MaSuRCA control file make to specify the location of all the reads needed. To start the program is the exact same for the Illumina only run:

```
cd /scratch
mkdir Masurca_hybrid
cd Masurca_hybrid
scp /scratch/Botany2020NMGWorkshop/Genome_assembly/scripts/sr_config_hybrid.txt /scratch/Masurca_hybrid
```

```
cat sr_config_hybrid.txt
```

```
/opt/MaSuRCA-3.4.1/bin/masurca sr_config_hybrid.txt
```

```
./assemble.sh
```

Quality control of assembly

To test the quality of the assemblies, there are a few metrics that can be explored. These include number of contigs, N50, assembled bases, and presence of BUSCO genes. For the structural characteristics, there are several options. One quick option is NanoPlot. This will create plots and also give a summary of the input. You can either specify a fasta file for an assembly or fastq for PacBio/Nanopore raw reads:

```
scp -r /scratch/Botany2020NMGWorkshop/Genome_assembly/Completed_assemblies /scratch/completed_assemblies

cd /scratch/completed_assemblies

docker run -v /scratch/completed_assemblies/:/work/ -w /work nanozoo/nanoplot NanoPlot --fasta Ugibba_pruned_assembly.fasta --N50
```

I recommend checking the stats for the three different assemblies we made (Illumina only, MaSuRCA hybrd, and Flye) and the published assembly from 2017.

Another way to determine how good an assembly is, either genome or transcriptome, is to look at the parentage of BUSCO (Benchmarking Universal Single-Copy Orthologs) genes found. There are several lineage specific libraries in BUSCO and you should use the one that is the most specific for your taxa of interest. If you study grasses you would use Poales, or tomato would use Solanales. The more narrow you can get the more BUSCO genes you will be looking for. We will use Embryophyta which encompasses seed plants and includes 1,375 BUSCO genes. First is to make sure we have the embryophyta library uploaded:

```
cd /scratch
mkdir busco
cd busco
scp /scratch/completed_assemblies/*.fasta /scratch/busco
scp -r /scratch/Botany2020NMGWorkshop/embryophyta_odb9 /scratch/busco

docker run  -v /scratch/busco/:/busco_wd -w /busco_wd/  upendradevisetty/busco:v3.0 -i Ugibba_pruned_assembly.fasta --out
Ugibba_pruned --lineage_path embryophyta_odb9 --mode geno
```

Once you run this for the four different assemblies that you already investigated, move the short_summaries.txt files from each run into a new folder called “Ugibba_assembly_comparison”. You should have four files. Now we will use the supplied python script with the Busco package to plot the results using ggplot in R:

```
python3 /scratch/Botany2020NMGWorkshop/Genome_assembly/scripts/generate_plot.py -wd Busco_Summaries
```