Genome Assembly Workshop: June 25-27; 2024:

Day1: Prokaryotic Genome Assembly:

These are the details and step-by-step details for day 1 projects:

Here is the data project ID hosted on European Nucleotide Archives:
https://www.ebi.ac.uk/ena/browser/view/PRJNA935550

If you want to know more about the dataset we are going to use, go through this publication:
https://www.sciencedirect.com/org/science/article/pii/S2576098X23004425#s2-1

We are only going to download the data from Britain:

Here are the commands for the data downloads i.e, if you choose to download them yourself:
Britain:

- Note: We already have the rawdata at the following folder:

```
/project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/00_RawData

Files:
Illumina Fwd: SRR24502289_1.fastq.gz
Illumina Rev: SRR24502289_2.fastq.gz
Nanopore: SRR24502288_1.fastq.gz
```

- you can make a softlink/shortcut of the files in your own folder):

However, if you want to download the data yourself please use the data transfer node (`ssh atlas-dtn-1`):

```
# Illumina Fwd:
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR245/089/SRR24502289/SRR24502289_1.fastq.gz

# Illumina Rev
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR245/089/SRR24502289/SRR24502289_2.fastq.gz

# Nanopore
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR245/088/SRR24502288/SRR24502288_1.fastq.gz
```

Question 1: How does the Illumina data look?

We can use `fastqc` for analyzing the Illumina Reads:

NOTE: Never run a computationally intensive job on the head node. Before we proceed, ensure that you are in a compute node.

```
srun -A gif_vrsc_workshop -N1 -n4 -c8 --pty --preserve-env bash
####################
-A: our account "gif_vrsc_workshop"
-N1: One node
-n2: Two tasks/processes
-c4: Four CPUs per processes
--pty: pseudo terminal for interactive use
--preserve-env: preserve the user's environment
bash: The command/executable
```

Now let's check Illumina reads quality using `fastqc` :

```
mkdir /project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/01a_Fastqc

cd  /project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/01a_Fastqc

# Softlink the Illumina Reads:

ln -s /project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/00_RawData/SRR24502289_* .

# Load the module:
module load fastqc
fastqc -o . SRR24502289_1.fastq.gz &> SRR24502289_1.log &

fastqc -o . SRR24502289_2.fastq.gz &> SRR24502289_2.log &

#
```

Though we only have two samples here, it may still be a good idea to collate the html files, plots and data into a single folder. We can use `multiqc` for that. Atlas does not have a module so we will use a `singularity/apptainer sif` file that we have already downloaded for `multiqc`. We do need to load the module for `apptainer` though.

```
module load apptainer

apptainer exec /project/gif_vrsc_workshop/software/multiqc_latest.sif multiqc -p --title "SRR2450228_Illumina
```

Copy the `multiqc` outputs to your local computer: The ideal way to do this is using globus. For those of you with a Linux system and/or a Windows subsystem for linux, you can also use `scp` as follows:

On your local computer:

```
scp -r $USER@atlas-dtn-1.hpc.msstate.edu:/project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/01a_Fastqc/SR
```

** What does fastqc tell us about the Illumina datasets?**

Here are a few tips:

1. Check the general statistics
2. Sequence quality
3. Per Sequence quality score
4. Other Per base stats

Question 2: How does the Nanopore data look?

We could use `nanoQC` to examine nanoppre data. We use `conda` to install this program:

```
module load miniconda3
conda create -n nanoqc
source activate nanoqc
conda install -c bioconda nanoqc
nanoQC -h

###

ln -s /project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/00_RawData/SRR24502288_1.fastq.gz .

source activate nanoqc

nanoQC SRR24502288_1.fastq.gz -o SRR24502288_Nano >& SRR24502288_Nano.log&
```

To visualize the `html` file copy the file to your local computer as discussed earlier for `multiqc`

Another good optiom to quickly gets stats on the nanopore reads is `nanoplot`:

```
conda create -n conda_software python=3.11
source activate conda_software
pip install NanoPlot

mkdir /project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/01c_Nanoplots

cd /project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/01c_Nanoplots

ln -s /project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/00_RawData/SRR24502288_1.fastq.gz

NanoPlot -t4  --fastq SRR24502288_1.fastq.gz --maxlength 40000 --plots dot
```

Here are some of the salient Nanopore stats:

```
General summary:
Mean read length:               12,186.3
Mean read quality:                  10.1
Median read length:              5,595.5
Median read quality:                11.1
Number of reads:                29,076.0
Read length N50:                26,626.0
STDEV read length:              14,531.3
Total bases:               354,329,504.0
```

Question 3: What is the approximate coverage of the Illumina and Nanopore data?

Coverage = (Number of Reads * Read length avg)/Approx Genome size

For Illumina, each of fwd and reverse reads have 963100 reads. The approx. Ecoli genome size is 4.5MB and the avg read length is ~250.

Coverage is:

```
Illumina
{[963100+963100]*250}/450000= 1000 X
```

Can you do the same for Nanopore data?

Unicycler:

https://github.com/rrwick/Unicycler

Unicycler is a tool designed for assembling bacterial genomes, capable of handling Illumina reads, long reads (PacBio or Nanopore), or a combination of both. Here are the key points and steps to get started with Unicycler:

Introduction
Unicycler offers three types of assemblies:

Illumina-only assembly: Optimizes SPAdes to produce cleaner assembly graphs.

Long-read-only assembly: Uses miniasm and Racon to assemble long reads.

Hybrid assembly: Combines short and long reads for the best assembly quality.

Why Use Unicycler

Circularization: Automatically circularizes replicons.

Plasmid Handling: Effective with plasmid-rich genomes.

Graph Output: Produces assembly graphs viewable in Bandage.

Contamination Filtering: Removes low-depth contigs.

Low Misassembly Rates: It is slow but Ensures high-quality assemblies.

Working Directory:
`/project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/02_Unicycler`

- The main command (with only Illumina) takes about 30-45 mins to run::

```
time /project/gif_vrsc_workshop/software/Unicycler/unicycler-runner.py \
-t 16 \
-1 SRR24502289_1.fastq.gz \
-2 SRR24502289_2.fastq.gz \
--spades_options "-m 2048" \
-o MRE162_Illumina
```

- The main command (with Illumina and Nanopore) takes about 45-70 mins to run:

```
time /project/gif_vrsc_workshop/software/Unicycler/unicycler-runner.py \
-t 16 \
-1 SRR24502289_1.fastq.gz \
-2 SRR24502289_2.fastq.gz \
-l SRR24502288_1.fastq.gz \
--spades_options "-m 2048" \
-o MRE162_Hybrid
```

Assembly stats:

We can check the headers in the `assembly.fasta` file:

```
grep '^>' 02_Unicycler/MRE162_Illumina/assembly.fasta
>1 length=687924 depth=1.00x
>2 length=449850 depth=1.03x
>3 length=420964 depth=1.07x
>4 length=378726 depth=0.95x
.......................
.......................
>94 length=110 depth=1.33x
>95 length=106 depth=1.58x
>96 length=104 depth=2.09x
>97 length=103 depth=1.89x
```

```
grep '^>' MRE162_Hybrid/assembly.fasta
>1 length=4686249 depth=1.00x
>2 length=76850 depth=1.03x circular=true
>3 length=232 depth=1.00x
>4 length=171 depth=1.00x
```

We can also any program to find the stats, for example `bbstats`:

- Illumina Assembly:

```
/project/gif_vrsc_workshop/software/bbmap/stats.sh

MRE162_Illumina/assembly.fasta
A       C       G       T       N       IUPAC   Other   GC      GC_stdev
0.2472  0.2531  0.2526  0.2471  0.0000  0.0000  0.0000  0.5057  0.0487

Main genome scaffold total:             24
Main genome contig total:               24
Main genome scaffold sequence total:    4.785 MB
Main genome contig sequence total:      4.785 MB        0.000% gap
Main genome scaffold N/L50:             3/940.198 KB
Main genome contig N/L50:               3/940.198 KB
Main genome scaffold N/L90:             6/360.277 KB
Main genome contig N/L90:               6/360.277 KB
Max scaffold length:                    1.315 MB
Max contig length:                      1.315 MB
Number of scaffolds > 50 KB:            7
% main genome in scaffolds > 50 KB:     97.79%
```

- Hybrid Assembly:

```
/project/gif_vrsc_workshop/software/bbmap/stats.sh Hybrid/assembly.fasta
A       C       G       T       N       IUPAC   Other   GC      GC_stdev
0.2470  0.2522  0.2535  0.2473  0.0000  0.0000  0.0000  0.5057  0.0363

Main genome scaffold total:             4
Main genome contig total:               4
Main genome scaffold sequence total:    4.765 MB
Main genome contig sequence total:      4.765 MB        0.000% gap
Main genome scaffold N/L50:             1/4.687 MB
Main genome contig N/L50:               1/4.687 MB
Main genome scaffold N/L90:             1/4.687 MB
Main genome contig N/L90:               1/4.687 MB
Max scaffold length:                    4.687 MB
Max contig length:                      4.687 MB
Number of scaffolds > 50 KB:            2
% main genome in scaffolds > 50 KB:     99.99%
```

Assembly Stats terms:

N50: The length of the shortest contig at 50% of the total genome length. A higher N50 indicates better assembly contiguity.

L50: & The smallest number of contigs that cover 50% of the genome. A lower L50 indicates fewer, larger contigs.

Total Length: The combined length of all contigs. It should be close to the estimated genome size.

Number of Contigs: Fewer contigs suggest a more contiguous assembly.

Largest Contig: The length of the longest contig in the assembly. Larger contigs are preferable.

GC Content: The percentage of G and C bases. It should match the known GC content of the organism.

BUSCO Scores: Completeness of the assembly based on benchmarking universal single-copy orthologs. Higher scores indicate a more complete assembly.

Visualization of graphs
One really cool way to actually visualize the GFA files, is to download the GFA file to your local computer and view them on a

software like **Bandage** (https://rrwick.github.io/Bandage/) is a tool for visualizing de novo assembly graphs. It supports various graph formats (Velvet, SPAdes, Trinity, ASQG, GFA) and provides features like automatic node positioning, zooming, panning, node reshaping, color-coding, labeling, and BLAST search integration. Bandage can be used to assess assembly quality, identify problematic regions, resolve ambiguities, and extract sequences. It is available for OS X, Linux, and Windows, with binaries or source code available on GitHub.

A subset of reads:

Directory: `/project/gif_vrsc_workshop/GenomeAssemblyWorkshop/Day1/00_ReadSubset`

You can all try out assembling the subset of reads within this folder. This should only take about 10 mins or so. However the assembly may not be complete.