Bioinformaticians assemble !

10 janvier 2024

1 Context

Welcome, valiant Bioinformaticians of the Galaxy !
Brace yourselves as we embark upon an interstellar mission of utmost urgency ! Our recon satellites have detected an uncharted bacterial threat loitering through the expanses of the cosmos, and it's headed towards Earth at warp speed. The galaxy's renowned scientists have managed to intercept and sequence this enigmatic entity, yet its genomic assembly has perplexed minds across the universe !
Enter : You, our elite assembly specialists, are summoned to the digital realm of the Galaxy website to unravel the genomic mysteries of this cosmic bacterium ! Our salvation rests in your capable hands and agile minds. Armed with sequencers that harness the power of the stars and datasets extracted with technology from a distant future, your mission is to assemble, analyze, and neutralize the threat that looms over our existence. Will you decipher the genomic code in time and save our beloved home from potential calamity ? The race against time begins now and may the best genomes win !


2 Available Data

For this practical, you can access three available datasets :

- Illumina paired-end short-reads (**Paired_Illumina.fa**)
- Ultra-long Oxford Nanopore reads (**Ultra_Long_ONT.fa**)
- Pacific Biosciences High Fidelity reads (**HiFi_PB.fa**)

For the purpose of the practical we also give you the reference genome for comparison, but do not look at it too closely, it will spoil the fun !
Using one or several of these datasets, you should be able to get a good assembly of our pathogen, hopefully...

3 Take a look at your data

In this part, you will use **Seqkit** tools to investigate the properties of the different datasets. Using this tool, you should be able to answer the following questions for each dataset :

- How many bases/reads are there ?
- What is the mean read length/the N50 ?
- How long is the longest (shortest) read ?
- Can you estimate a coverage knowing that we gauge the pathogen genome size to be around 300kb ?

```
$ seqkit stats Paired_Illumina.fa
```

If a dataset is too large you can build smaller datasets via sub-sampling for faster and better results (50X is a good aim).

```
$ seqkit head -n 200000 Paired_Illumina.fa > Paired_Illumina_sub.fa
```

4 Your first assembly !

This part aims at obtaining a quick, initial assembly of the genome using any of the three datasets. You could use any of the available assemblers listed below. "But, which one", you ask ? The point of this practical session is to let you take the initiative and pick one ! So, for this first attempt, if the assembler asks for any parameter, try to guess a reasonable value but don't over-think it. At this point, it does not matter if the assembly is of poor quality.

5 Available Assemblers

Here we list and provide a brief description of the available tools you can use :

5.1 SPAdes

SPAdes is a short reads assembler designed for prokaryotic and small eukaryotic genomes. It does an excellent job at assembling bacteria with short-read sequencing, either multi-cell or single-cell data, and small metagenomes. It uses multiple sizes of k (k-mer size in the de Bruijn graph) to construct the best possible contigs. It generally takes longer time and memory than other assemblers.

```
$ spades.py --12 Paired_Illumina.fa   -o Nadine_spades --isolate
```

SPAdes can also use long reads to solve remaining repeats of the de Bruijn graph and improve continuity.

5.2 HIFIasm

Hifiasm is a robust and efficient genome assembler developed specifically for high-fidelity (HiFi) reads from Pacific Biosciences (PacBio) sequencing technology. Since HiFi reads are long reads with considerably high accuracy, they provide a strong foundation for assembling genomes.

```
$ hifiasm  HiFi_PB.fa -o Nadine_hifiasm -f 32
```

Hifiasm produce several .gfa file, the most interestings are the Nadine_hifiasm.bp.p_ctg.gfa where p_ctg stand for primary contigs and the Nadine_hifiasm.bp.p_utg where p_ctg stand for primary unitigs (no heuristics used).
Recently Hifiasm integrated a novel feature to integrate ultra long reads to solve repeats in the HiFi overlap graph.

5.3 Flye

Flye is a long-read assembler based on an original paradigm. It builds a repeat graph (conceptually similar to de Bruijn graphs, but uses approximate matches between reads).
Flye can use any kind of long reads, you have to chose the right argument to indicate the expected error rate of each dataset.

```
$ flye --nano-raw Ultra_Long_ONT.fa  --out-dir Nadine_flye_ONT
```

```
flye --pacbio-hifi HiFi_PB.fa --out-dir Nadine_flye_HIFI --min-overlap 3000
```

5.4 Unicycler

Unicycler is a prominent genome assembly software tailored towards bacterial genomics. The interesting feature of Unicycler is that it implements several assemblies strategies. It can build an assembly directly from long reads, and themSincn peform polishing operations to clean the contigs from the sequencing errors.

```
$ unicycler-runner.py  -l Ultra_Long_ONT.fa -o Nadine_unicycler
```

It can also build a hybrid assembly by first constructing a De Bruijn graph from the short reads and then solve repeats using long reads.

6 Assembly evaluation

Once you get an assembly, you will want to assess your contigs quality. For this, you have multiple options.

6.1 *De novo* evaluation

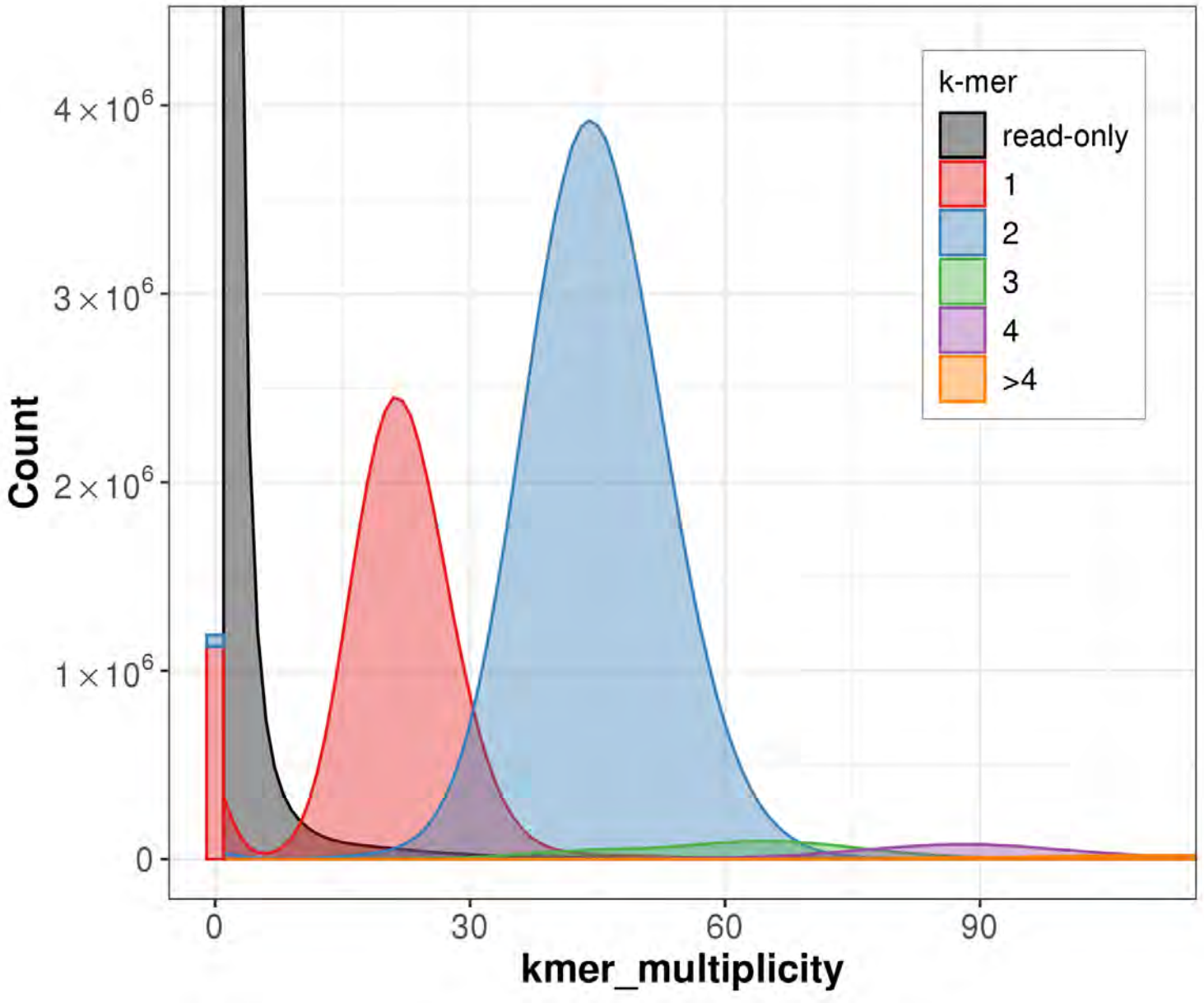
You can use the **Merqury** tool to evaluate your contigs *without* using a reference genome. The idea is to compare the k-mer spectrum of a sequencing experiment (Illumina or HiFi preferably) of your individual with the k-mers present in your contigs. You expect the frequent k-mers to be present in your contigs and the rare k-mers to be absent. To run **Merquery**, first you need to perform a kmer counting operation with **Meryl**.

```
$ meryl k=31 count  Paired_Illumina.fa output Nadine.meryl
```

Then you run **Merquery** using the meryl database and your contigs

```
$ export MERQURY=/home/genomics/software/merqury-1.3/
$ merqury.sh Nadine.meryl/ Nadine_flye/assembly.fa  Nadine_merqury
```

After that you can watch your beautiful k-mer spectra in .png and read the reports.



6.2 Reference-based evaluation

You can compare your assembly to a reference using the **QUAST** tool to assess its quality. The idea is to align your contigs on your reference genome and compute stats by analyzing their alignment. Optionnally you

can also give quast a read set so it can perform additionnal checks.
The main questions that Quast will answer for you are :

- How many large/small misassemblies were made by the assembler
- What percentage of your genome is covered by your contigs
- How contiguous is your assembly (N50/N75/NGA50/NGA75 metrics)
- How close to the reference are your contig (%mismacthes)

Note that you can give **QUAST** multiple assemblies simultaneously for comparison purposes.

```
$ quast.py Nadine_flye/assembly.fa Nadine_unicycler/assembly.fa -r Reference.
  fa -o Nadine_quast_ref
```

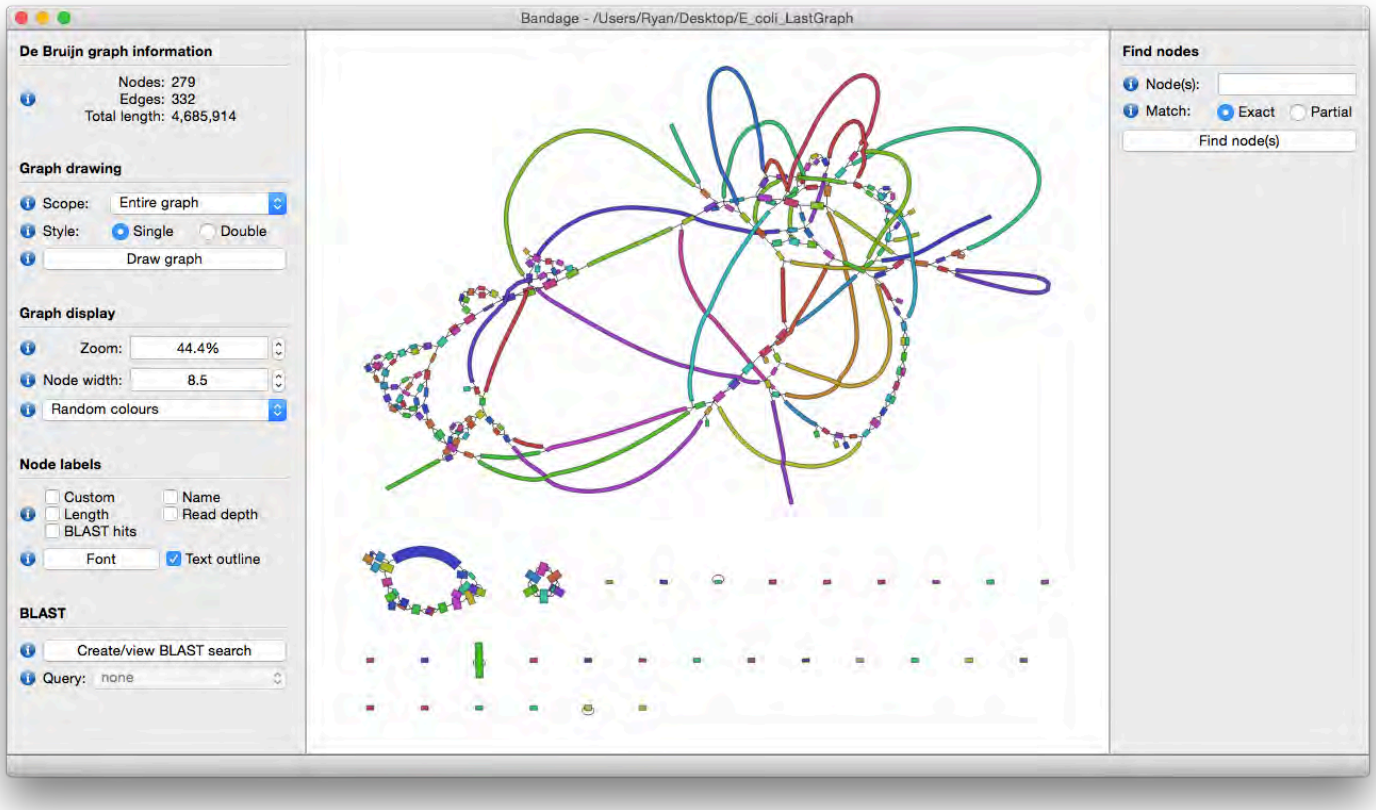
Without reference genome the reads and k-mer analysis can be performed

```
$ quast.py Nadine_flye/assembly.fa Nadine_unicycler/assembly.fa  -o
  Nadine_quast_denovo --pe12 Paired_Illumina.fa -k --k-mer-size 31
```

After QUAST completion, you may take a look at the various reports (like report.txt) in the shell or look at the report.html in a browser.

6.3 Visualize your assembly graph

Use the **Bandage** software to visualize the assembly graph (**Input :** my_assembly_graph.gfa) You may skip this step if you do not have a gfa file.



Once you can look at your graph, you can make the following observations :

- **Is your graph well connected ?**
- **How many disjoint components are there ?**
- **Does the graph have many branching nodes ?**

Disjoint components may indicates whether you had sufficient coverage to perform the assembly. It could also mean that the sequencer produced some sequences that did not overlap with one another. Branching nodes may indicates variants or repetitions that the assembler could not resolved.

7 Let the fun begin !

Now is the time for your creative side to shine ! Try different datasets or combinations of datasets with the available tools and various parameters. Test your contigs with Merqury, Quast, or Bandage to see if they have good properties. When you are satisfied, please keep your assembly safe in a folder, as it will be evaluated and compared with the other participant assemblies. To do so, please fill the following online form https://docs.google.com/spreadsheets/d/1MSrHwLdz-a1Q_9UKJOqrOypGgaYEtRZJSrpvqBHXuXM/edit?usp=sharing
Once you get a ”good” assemblies, you should be able to answer the following questions :

1. **What is the drawback of the short reads assembly ?**
2. **What is the drawback of the long reads assembly ?**
3. **What is the drawback of the HiFi reads assembly ?**
4. **Should we be afraid of this ”novel” bacteria ?**
5. **From which country (or planet) the bacteria may be from ?**

8 Real world assembly

As you may have guessed, this practical uses synthetic data (i.e., a handcrafted genome made for the purpose of this practical). We chose to build a small but hard-to-assemble bacteria, for the computing steps to be fast (and to avoid long and boring waiting time). In the real world, even a bacteria's genome is an order of magnitude larger than this. To get insight into real data assembly, you can assemble a small known bacteria (around 1.6Mb). Keep in mind that most bacteria are larger (around five megabases), and you may understand why we chose to assemble a small one.
To perform this real assembly you have two datasets :

- Illumina Miseq paired-ended short-reads (SRR9043691)
- Oxford Nanopore reads (SRR10342325)

You also have a reference genome (Campylobacter jejuni CFSAN032806) to evaluate this assembly (you can use Merqury too).
For high coverage datasets, you **need** to build smaller datasets via sub-sampling for faster and better results (50X is a good aim). If the sub-sampled datasets can contain the longest reads it is even better! For this you can use **Seqkit** to do so.

```
$ seqkit head -n 1000 sorted_file.fa > top_1000_longest_reads.fa
```

Illumina reads all have the same size, but for long reads it is quite interesting to keep only the longest reads. To do so we can sort them by size before to construct our subsample.

```
$ seqkit sort -l -r your_file.fa > sorted_file.fa
```

You can do the whole practical once again using this data!

9 Treasure hunt

If you are brave enough, you can try to assemble by hand this set of ”reads” :
CAGATGTCCT
GACCTTTTCT
TAAGATCTTT
TCTTTAGCCG
TTTCTTCTTT
GAGCTTTACT
TTACTTCATT
TCATTCACAT
CACATGAGCT
GTCCTGAACA
GAGCTGATCA
TCTTTCAGAT
AGCCGGACCT
TATGATCATT
TCATTAGGCG
AGGCGGAGCT
HINT : There are NO sequencing errors
HINT : Overlaps are at least of length 5