Genome assembly practical

Main steps

1. Data examination
2. Assembly
3. Evaluation
4. Repeat!

For the fast and curious: see ‘Real-World Assembly’ and ‘Treasure hunt’.

Context

Bad news! The beers at the music bar taste off, and a bacterial spoilage is suspected. Guy L. is about to resign. Luckily, you’re in Český Krumlov, the ~~bohemian~~ world city with the highest density of bioinformaticians per capita! Your mission: assemble the microbial genomes from the suspect beer samples to identify the culprit. Is there a safety concern lurking in your pint? Or maybe just a flavor profile to avoid? Either way, your findings will determine what’s safe to drink tonight, and restore everyone’s sanity.

Available Data

- Illumina paired-end short reads (**`Paired_Illumina.fa`**)
- Ultra-long Oxford Nanopore reads (**`UltraLong_ONT.fa`**)
- Pacific Biosciences HiFi reads (**`HiFi_PB.fa`**)

In this practical, we also provide the reference genome for comparison purposes. However, do not examine it too closely, as this may spoil the fun!

Using one (or several) of these datasets, you should be able to assemble the genome of our bacteria successfully, or at least we hope so!

Analyze your data

In this section, you will use **Seqkit** tools to investigate the properties of the datasets. With these tools, you should answer the following questions for each dataset:

- How many bases and reads are there?
- What are the mean read length and the N50?
- What are the lengths of the longest and shortest reads?
- Can you estimate the coverage, given that the bacteria's genome size is approximately 2 Mb?

Your First Assembly!

The goal of this section is to perform an initial assembly of the genome using one of the three datasets provided.

You may choose any of the available assemblers listed below. "But which one should I use?" you ask. The purpose of this practical session is to encourage you to take the initiative and make a choice! For this first attempt, if the assembler requires any parameters, try to estimate reasonable values without overthinking. At this stage, the quality of the assembly is not critical.

Available Assemblers

Below is a list and a brief description of the tools available for your assembly tasks:

Short Read Assemblers

Input: `Paired_Illumina.fq`

- SPAdes
- MEGAHIT
- IDBA

SPAdes is an assembler designed for Illumina sequencing data, particularly effective for prokaryotic and small eukaryotic genomes. It performs well on bacterial genomes, small metagenomes, and both multi-cell and single-cell datasets. SPAdes uses multiple k-mer sizes in the de Bruijn graph to construct high-quality contigs, but it can be resource-intensive in terms of time and memory.

TIP: To speed up SPAdes, you can:
- Use a single k-mer size with the `-k` option.
- Skip the correction step (especially without a FASTQ file) using the `--only-assembler` option.

MEGAHIT is optimized for large metagenomic datasets and excels at assembling even low-coverage regions. It is known for being fast and memory-efficient while also supporting multiple k-mer sizes.

TIP: Like SPAdes, MEGAHIT allows you to specify k-mer sizes using the `--k-list` parameter.

IDBA (Iterative De Bruijn Graph Assembler) is widely used for de novo assembly of genomic data. It iteratively increases k-mer sizes to resolve complex genomic regions and repeats. IDBA is versatile and can handle datasets with varying sequencing depths.

Variants:
- **IDBA-UD:** Designed for single-cell and metagenomic sequencing data.
- **IDBA-Hybrid:** Combines short- and long-read sequencing data for hybrid assembly.

Optional Exploration

Hybrid Assembly

Many short-read assemblers now support hybrid assemblies by integrating long reads. Try assembling both short reads alone and in combination with long reads to compare results.

Paired-End vs. Unpaired Reads

Short-read assemblers often use paired-end data to improve assembly via scaffolding. You can experiment by ignoring pairing information and compare results with interleaved paired-end data.

k-mer Sizes

Assemblers relying on k-mers can produce different results depending on the k value. Experiment with k sizes or examine intermediate assemblies to observe the effect on assembly statistics.

Long Read Assemblers

Input: **`UltraLong_ONT.fq`** and/or **`HiFi_PB.fq`**

- Flye
- Raven
- HiFiasm
- Shasta
- Verkko

Flye is a long-read assembler based on an original paradigm. It builds a repeat graph (conceptually similar to de Bruijn graphs) using approximate matches between reads. Flye is capable of assembling both genomes and metagenomes, making it versatile for a wide range of applications.

Raven is a long-read assembler that utilizes a streamlined overlap-layout-consensus (OLC) approach. It is specifically optimized for nanopore reads, providing fast assemblies with good contiguity. Raven is particularly effective for datasets with moderate coverage and is known for its simplicity and speed in handling noisy long-read data.

TIP: For some reason Racon polishing is not working today use -p 0 to get a Raven assembly!

HiFiasm is a state-of-the-art assembler designed specifically for PacBio HiFi reads. It uses a haplotype-resolved assembly approach, making it ideal for diploid genomes and highly accurate long-read data. HiFiasm leverages the high accuracy of HiFi reads to produce highly contiguous assemblies with minimal polishing required. However, it is not suited for noisy long reads, such as those from Oxford Nanopore sequencing.

Shasta is a long-read assembler optimized for PacBio HiFi and other high-quality long reads. It uses a specialized representation of overlaps to accelerate assembly and is designed to gen-

erate highly contiguous assemblies with minimal resource usage. Shasta's built-in consensus algorithm focuses on speed and efficiency.

Verkko is a hybrid, graph-based genome assembly pipeline designed to produce telomere-to-telomere (T2T) assemblies by combining highly accurate long reads (typically PacBio HiFi, or high-accuracy ONT reads) with Oxford Nanopore ultra-long reads. It first builds an assembly graph from the accurate reads, then uses the ultra-long reads to traverse repeats and ambiguities, improving contiguity while preserving base-level accuracy. Verkko is therefore particularly effective when repeats prevent a HiFi-only (or ONT-only) assembly from becoming fully contiguous, but it can be more computationally demanding than single-technology assemblers.

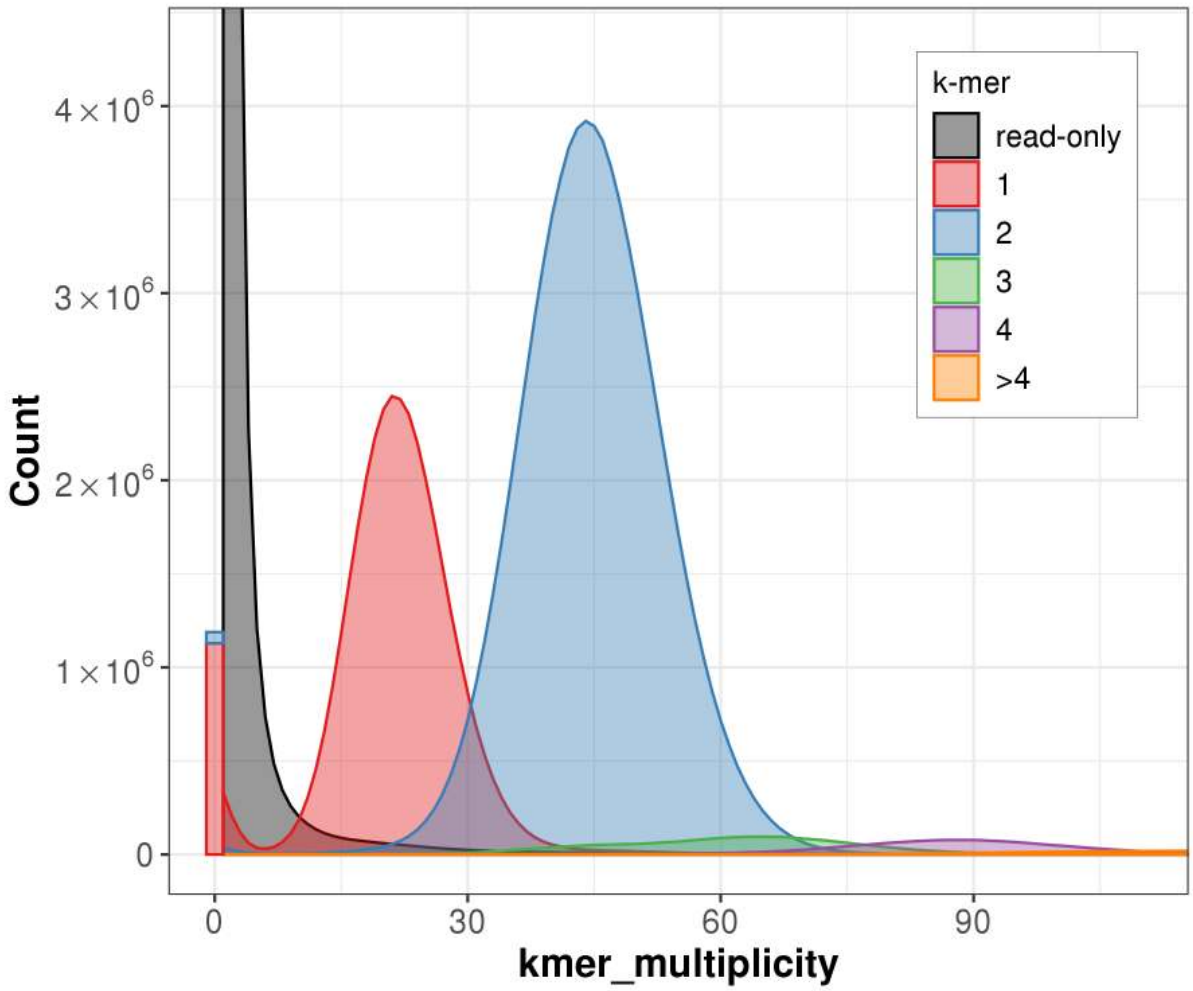
Assembly Evaluation

Once you obtain an assembly, you will want to assess the quality of your contigs. For this, you have several options.

De Novo Evaluation

You can use the **Merqury** tool to evaluate your contigs *without* relying on a reference genome. The idea is to compare the k-mer spectrum of a sequencing experiment (preferably Illumina or HiFi) of your individual to the k-mers present in your contigs. You would expect frequent k-mers to be present in your contigs and rare k-mers to be absent.

To run **Merqury**, first, perform a k-mer counting operation with **Meryl** (Input: **`Paired_Illumina.fq`**). Then, run **Merqury** using the database and your contigs (Input: **`read-db.meryldb`** + **`my_contigs.fa`**).



Reference-Based Evaluation

You can compare your assembly to a reference genome using the **QUAST** tool to assess its quality. This involves aligning your contigs to the reference genome and analyzing the alignment statistics.

Input: `ReferenceGenome.fa` + `my_contigs.fa`

Optionally, you can also provide a read set for additional checks

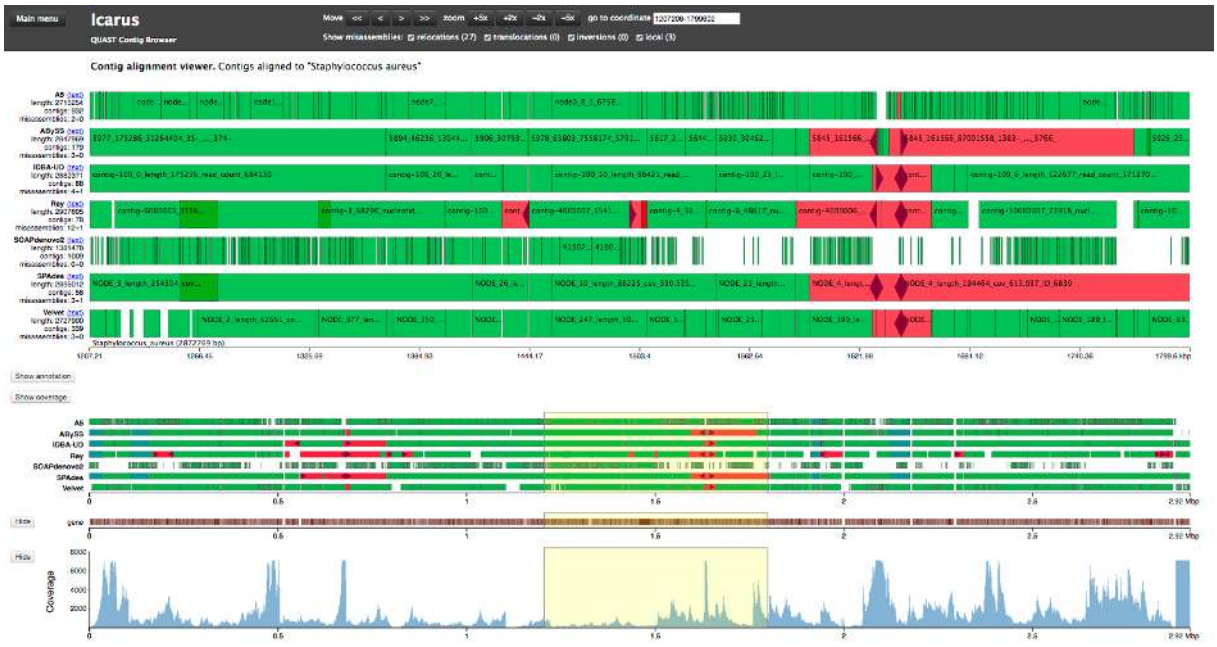
Input: `ReferenceGenome.fa` + `my_contigs.fa` + `Paired_Illumina.fq`.

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 contigs? (% mismatches)**

However, note that you need a reference genome to use this method. In real-life scenarios, you often do not have an "exact" reference genome. For this practical session, we provide the exact one. Once again, do not examine it too closely—it might spoil the challenge!

Additionally, you can provide QUAST with multiple assemblies simultaneously for comparison.

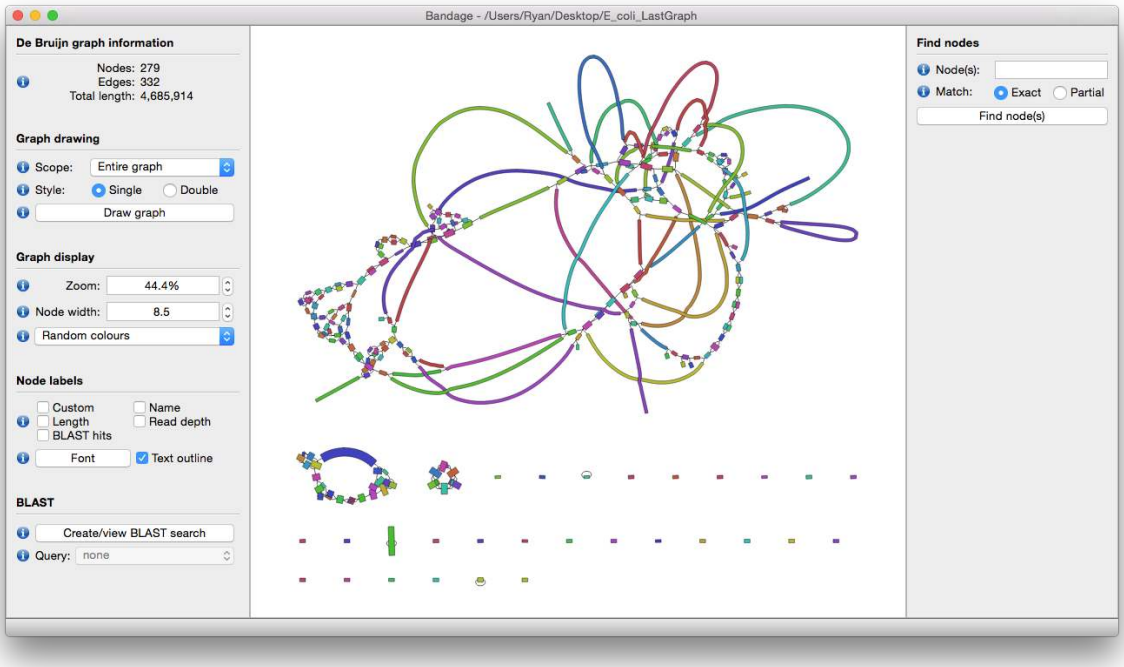


Visualizing Your Assembly Graph

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 examine your graph, consider the following observations:

- **Is your graph well-connected?** This indicates whether you had sufficient coverage to perform the assembly. If not, it could also suggest that the sequencer produced sequences that did not overlap with one another.
- **Does the graph have many branching nodes?** This indicates variants or repeats that the assembler could not resolve.

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 evaluate their properties.

When done with your assembly, submit your results in the following online form.

Once you obtain "good" assemblies, you should be able to answer the following questions:

1. What are the drawbacks of short-read assembly?
2. What are the drawbacks of long-read assembly?
3. What are the drawbacks of HiFi-read assembly?
4. **Should we be concerned about the beers?**

Real-World Assembly (Takes time...)

As you may have guessed, this practical uses synthetic data (i.e., a handcrafted genome created specifically for this practical). To gain insight into real data assembly, you can try assembling a small, known bacterium (around 1.6 Mb). Keep in mind that most bacterial genomes are significantly larger (approximately 5 Mb), which explains why we opted for a smaller example here.

- Illumina MiSeq paired-end short reads (**`SRR9043691`**)
- Oxford Nanopore reads (**`SRR10342325`**)

You also have a reference genome (**`Campylobacter jejuni CFSAN032806`**) available for evaluating your assembly (you can use **Merqury** for this as well).

Handling High-Coverage Datasets

For high-coverage datasets, you **Must** create smaller datasets through sub-sampling to achieve faster and more accurate results. A coverage depth of around 50X is a good target. If the sub-sampled datasets can include the longest reads, it will be even better! This can be performed by sorting reads by length with **Seqkit**.

You can repeat the entire practical using these real-world datasets. This will give you hands-on experience with assembling real bacterial genomes and allow you to compare your results with the synthetic dataset.

Treasure hunt (Hard)

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