Assembly Practical

6 janvier 2025

1 Main steps

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

For the fast and curious : Real data assemblies and treasure hunt

2 Context

Bad news ! The beers at the music bar are tasting off, and a bacterial spoilage is suspected. Luckily, you're in Český Krumlov—the bohemian 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

3 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**)

For the purposes of 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 !

4 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 ?

5 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.

6 Available Assemblers

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

6.1 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.

6.2 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 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.

6.3 Long Read Assemblers

Input : UltraLong_ONT.fq or **HiFi_PB.fq**

- Flye
- Raven
- HiFiasm
- Shasta
- Miniasm

Flye
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
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.

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

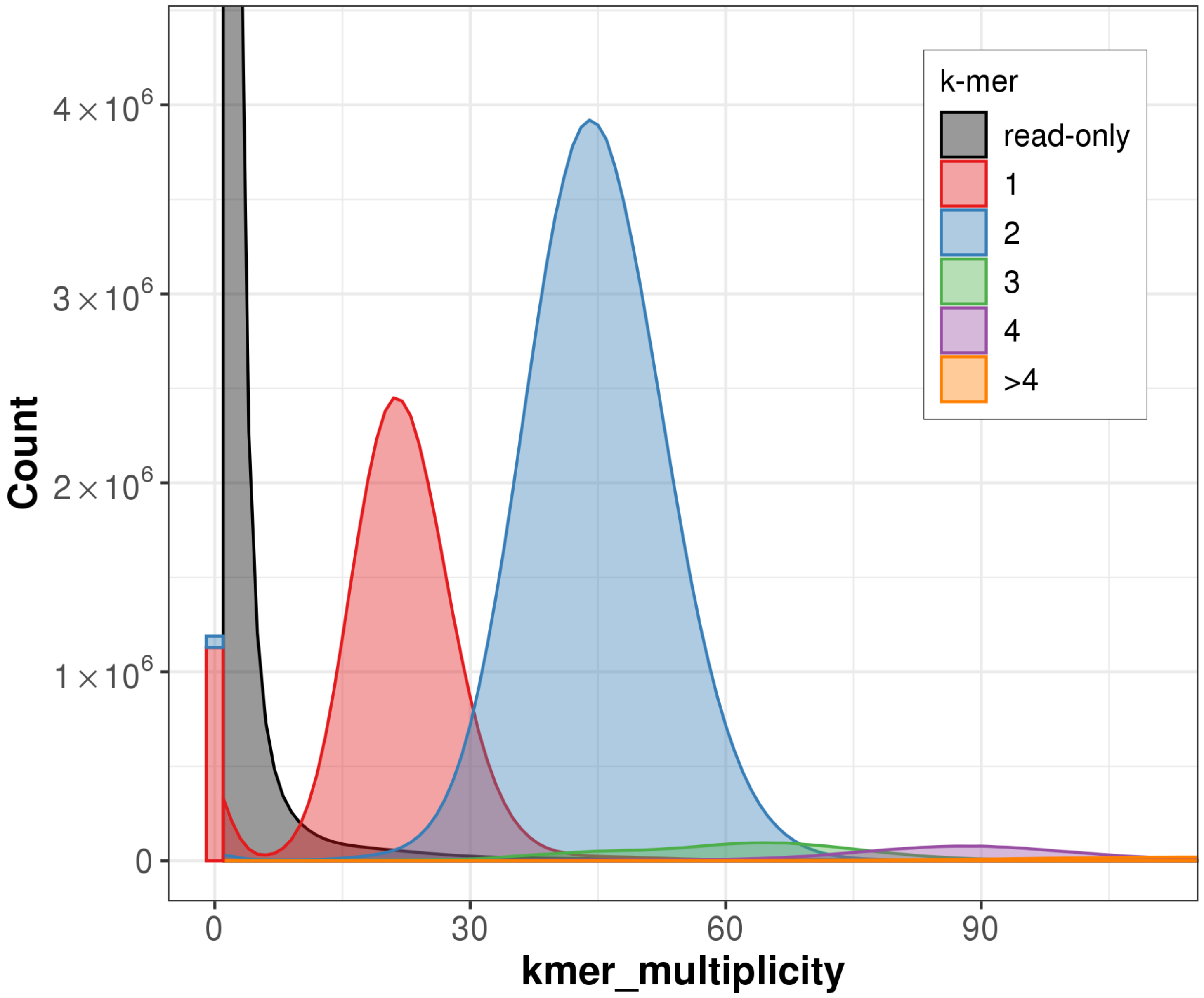
Miniasm *Warning : Using Miniasm is a "do-every-step-yourself" assembler.*
Codeveloped alongside Minimap, it serves as a proof of concept, demonstrating that erroneous long reads can be directly assembled. To run Miniasm, you need a PAF file, which is obtained by aligning your long reads against themselves (using the `ava-pb` or `ava-ont` options in Minimap2). Additionally, the resulting contigs will contain numerous errors since Miniasm does not perform any form of consensus or polishing. You must perform these steps yourself using tools like **Racon** or other long-read polishing workflows.

7 Assembly Evaluation

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

7.1 *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 Meryl database and your contigs (Input : **read-db.meryldb** + **my_contigs.fa**).

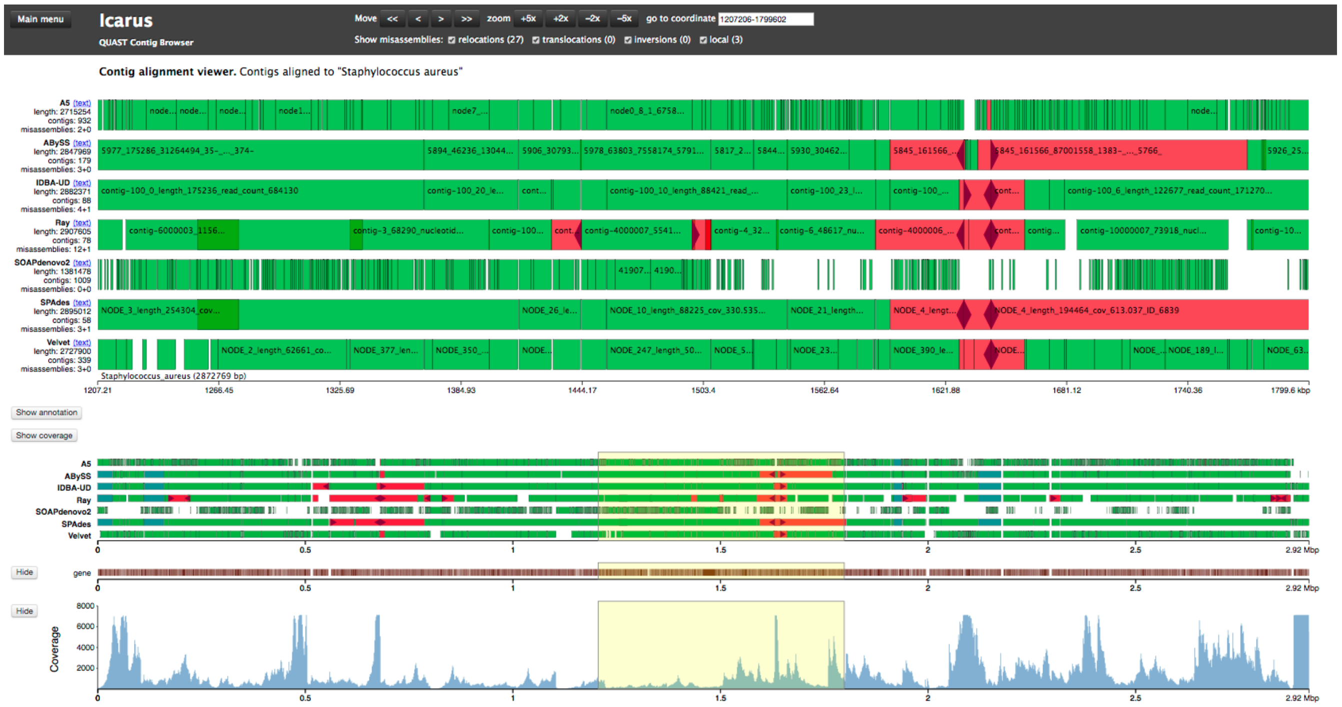


7.2 Reference-Based Evaluation

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 :

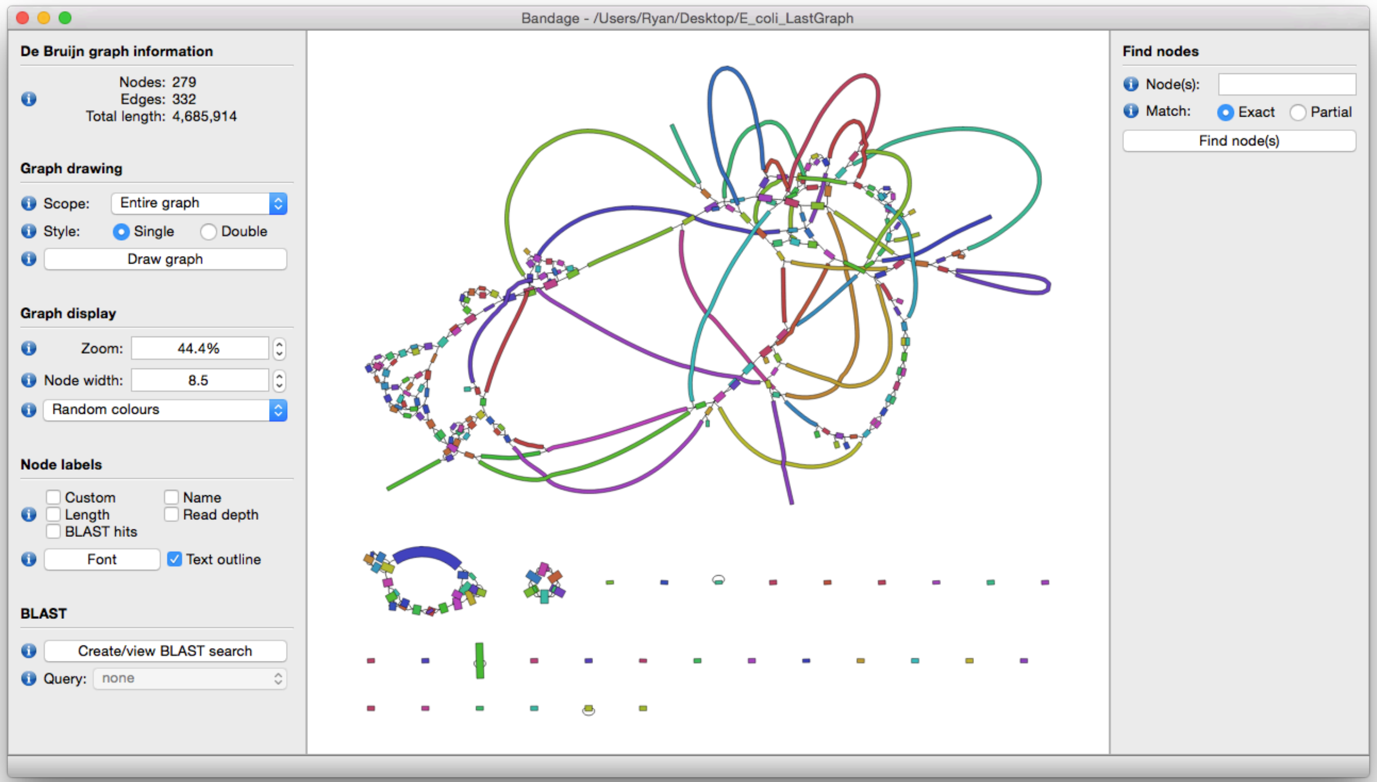
- **How many large/small misassemblies were made by the assembler ?**
- **What percentage of your genome is covered by your contigs ?**
- **How contiguous is your assembly ? (N50, N75, NGA50, NGA75 metrics)**
- **How close to the reference are your contigs ? (% mismatches)**

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.


Main menu
Icarus
QUAST Contig Browser
Move << < > >> zoom +5x +2x −2x −5x go to coordinate 1207206-1799602
Show misassemblies: relocations (27) translocations (0) inversions (0) local (3)
Contig alignment viewer. Contigs aligned to "Staphylococcus aureus"
A5 length: 2715254 contigs: 932 misassemblies: 2+0
ABySS length: 2847969 contigs: 179 misassemblies: 3+0
IDBA-UD length: 2882371 contigs: 88 misassemblies: 4+1
Ray length: 2907605 contigs: 78 misassemblies: 12+1
SOAPdenovo2 length: 1381478 contigs: 1009 misassemblies: 0+0
SPAdes length: 2895012 contigs: 58 misassemblies: 3+1
Velvet length: 2727900 contigs: 339 misassemblies: 3+0
Staphylococcus_aureus (2872769 bp)
Show annotation
Show coverage
Hide
gene
Coverage
2.92 Mbp

7.3 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.



Bandage - /Users/Ryan/Desktop/E_coli_LastGraph

De Bruijn graph information

Nodes: 279
Edges: 332
Total length: 4,685,914

Graph drawing

Scope: Entire graph
Style: [✓ Single] Double
Draw graph

Graph display

Zoom: 44.4%
Node width: 8.5
Random colours

Node labels

Custom, Name, Length, Read depth, BLAST hits
Font [✓ Text outline]

BLAST

Create/view BLAST search
Query: none

Find nodes

Node(s):
Match: [✓ Exact] Partial
Find node(s)

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.

8 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 satisfied with your assembly, submit your results in the following online form : `https://docs.google.com/spreadsheets/d/1MSrHwLdz-a1Q_9UKJ0qr0ypGgaYEtRZJSrpvqBHXuXM/edit?usp=sharing`
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 ?**

9 Real-World Assembly

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).

9.1 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 those real-world data. This will give you hands-on experience with assembling real bacterial genomes and allow you to compare your results with the synthetic dataset.

10 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