

Genome assembly and annotation

Day 3: Genome assembly

Igor Pessi

Department of Microbiology – UH

igor.pessi@helsinki.fi

Aims for this part of MMB-114

Day 1: Basics of UNIX and working with the command line

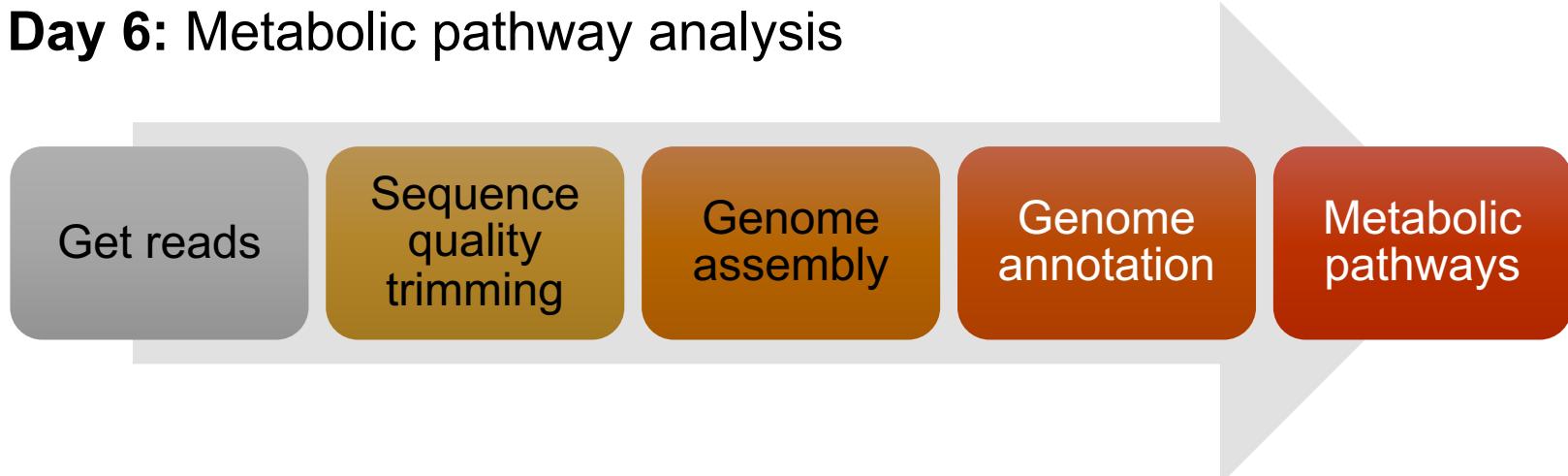
Day 2: Handling of Illumina data

Day 3: Genome assembly

Day 4: Check-up and report

Day 5: Genome annotation

Day 6: Metabolic pathway analysis



Before we start...

How does the raw data look like?

How many sequences we have for each genome? Are these numbers the same for the R1 and R2 files?

Is the length of the reads the same in the R1 and R2 files?

The "Per base sequence quality" module shows a problem. Why?

- Which part of the reads have the best quality? Beginning, middle, end?

What is the mean sequence quality for the majority of the reads?

The "Per base sequence content" module shows a problem. Why?

The "Adapter content" module also shows a problem. Why?

- Are there adapters in our reads?
- In which part of the reads?
- What are adapters and why should we remove them?

Which of the three genomes has the best sequence quality?

How does the data look like now?

CUTADAPT

How many times adapters were trimmed in R1 and R2?

How many reads were removed because they were too short?

How many low-quality bases were trimmed?

FASTQC

Do the "Per base sequence quality" plots look different now?

Why?

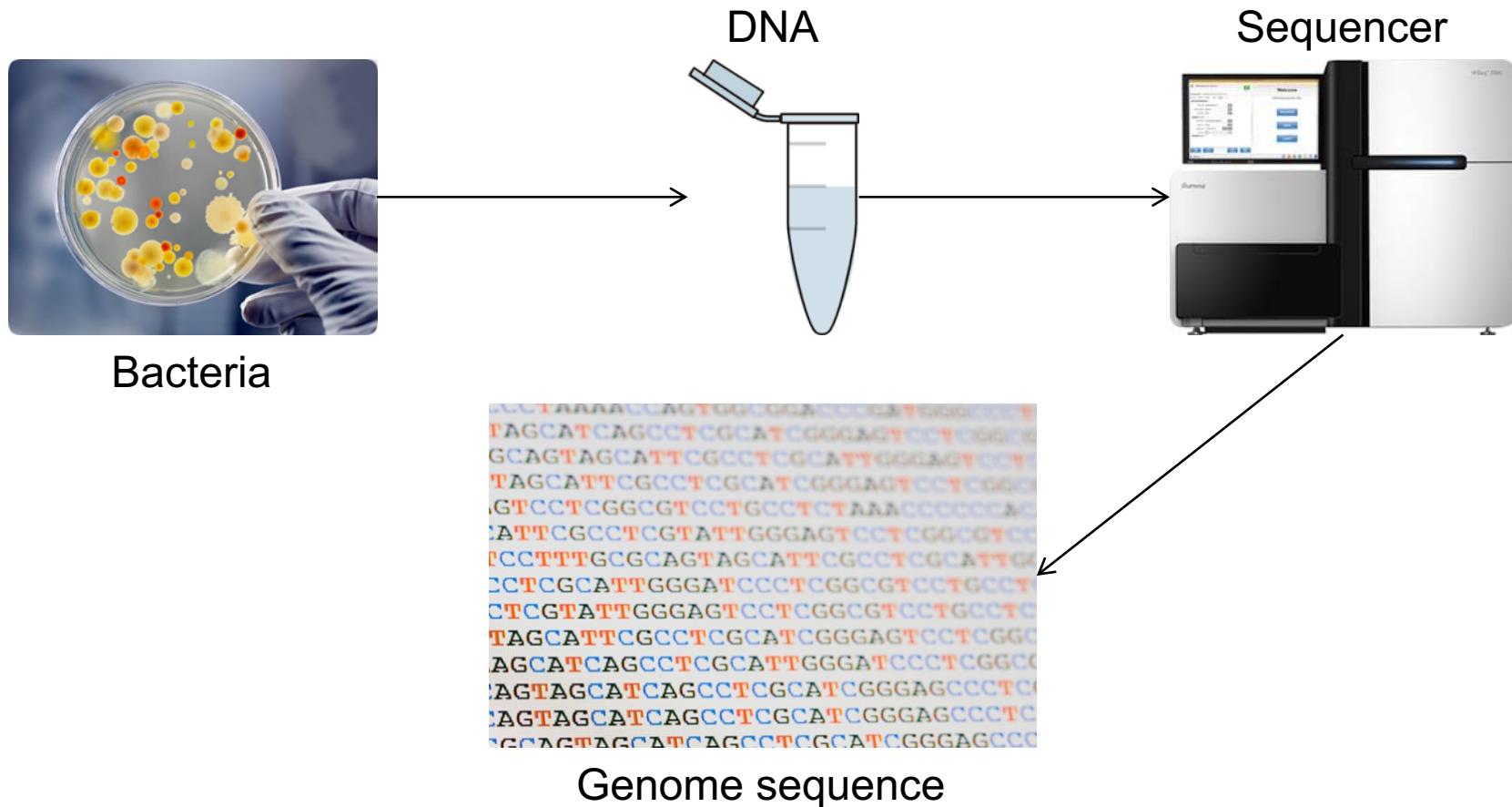
What is the mean sequence quality for the majority of the reads now?

The "Sequence length distribution" module shows a warning now. Have the read lengths changed? Why?

Are there still adapter sequences in our reads?

De novo genome assembly

In an ideal world:

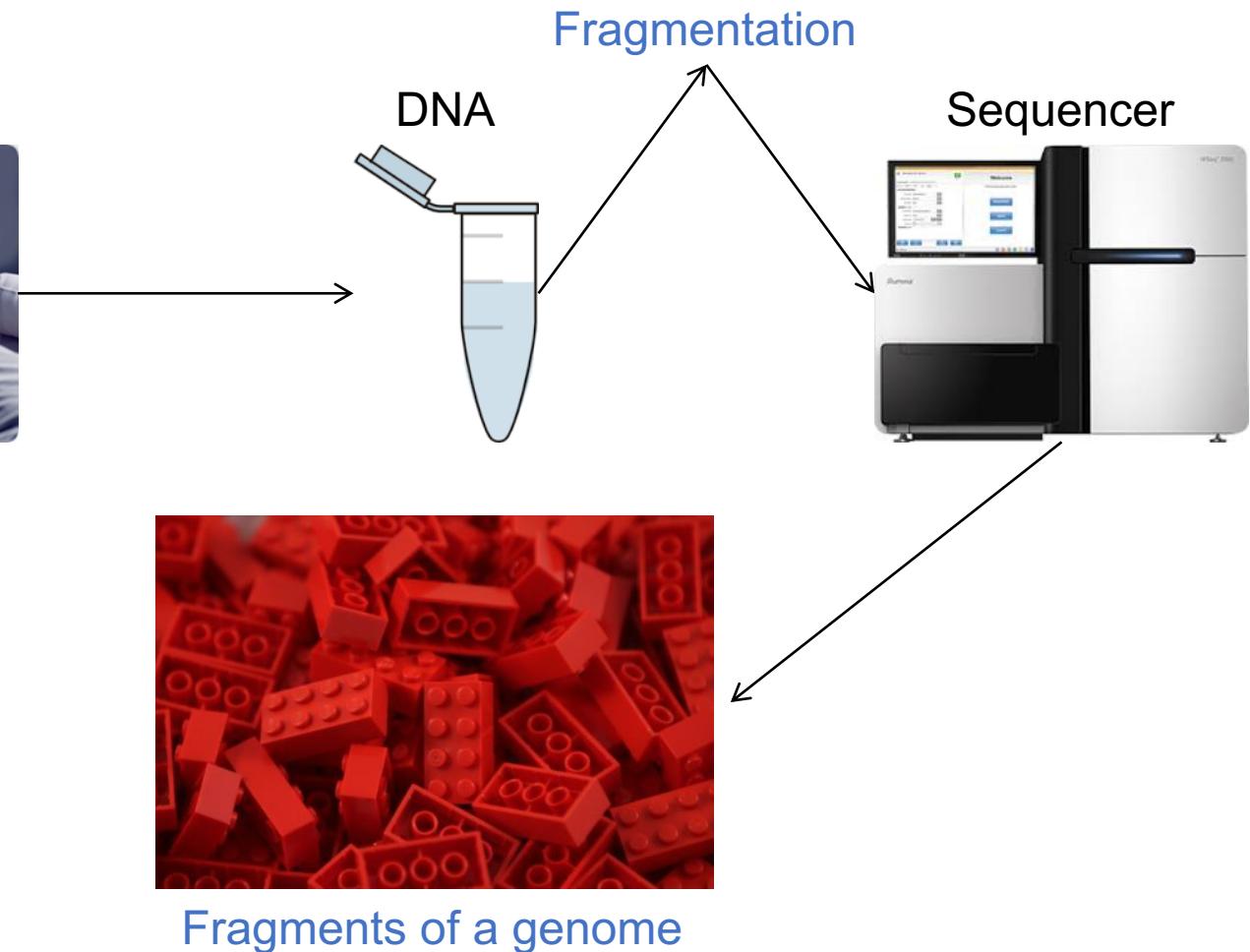


De novo genome assembly

In reality...



Bacteria



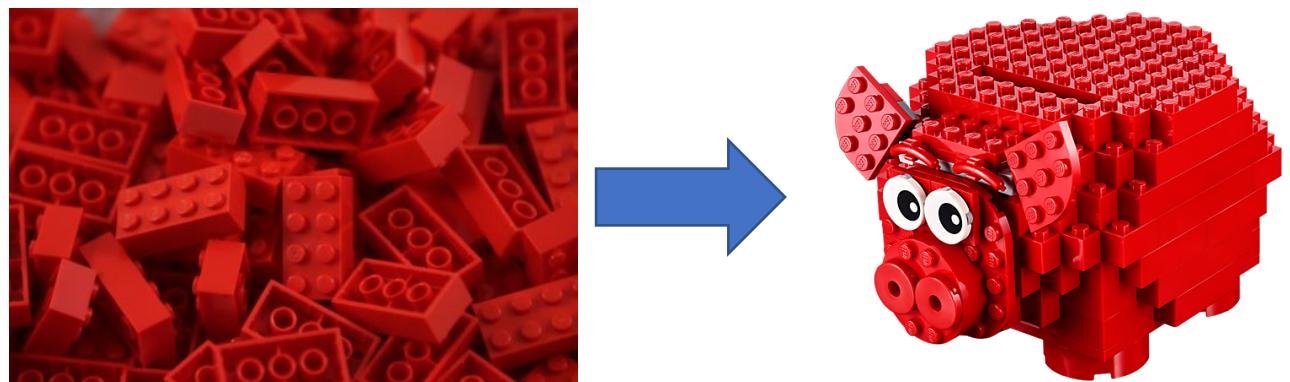
Genome assembly

Reconstruct the original genome from short sequence reads

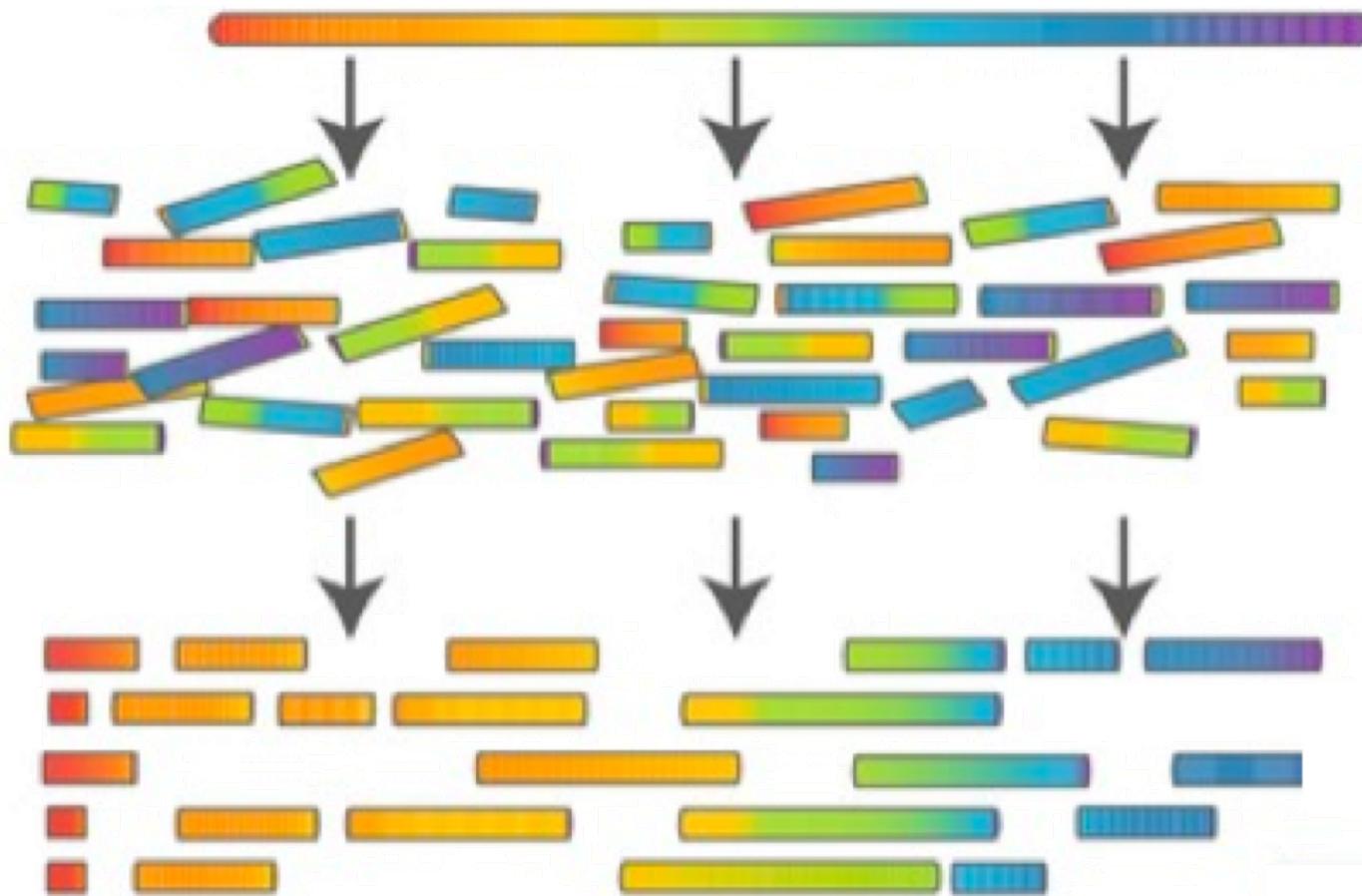
In an ideal world: sequences = one complete genome

In reality: sequences = multiple contigs

- Contiguous, unambiguous stretches of sequences



Genome assembly



ATGTTCCGATTAGGAAACCTATCTGTAACTGTTCAATTCA GTAAAAGGGAGGAAA

De Bruijn graph

In real life 10^6 sequences cannot be compared with each other

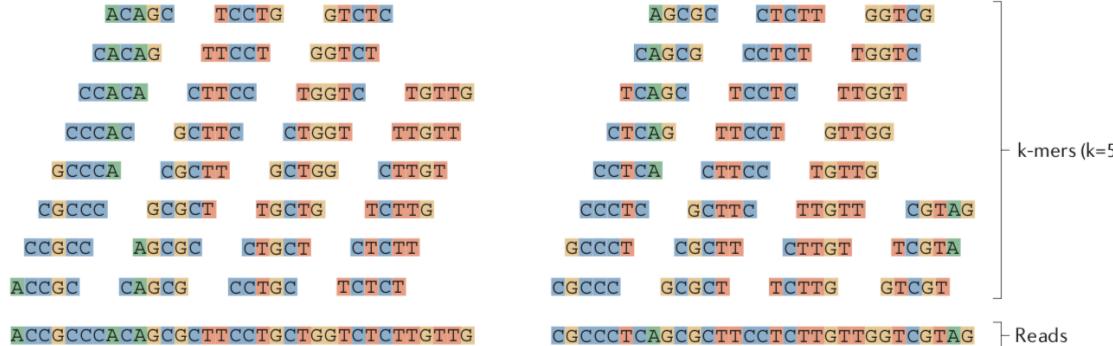
- $10^6 \times 10^6 = 10^{12}$ comparisons

Sequences are reduced to k-mers

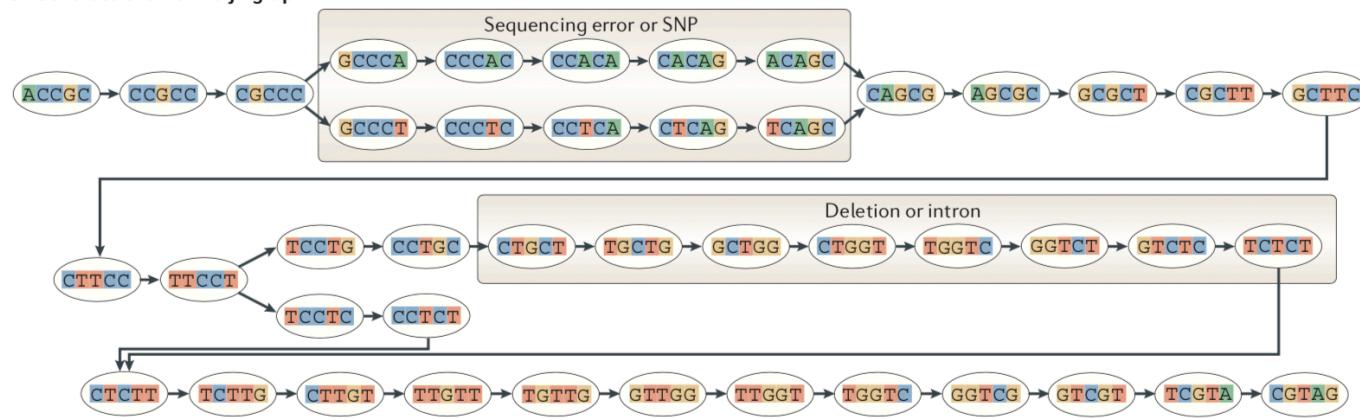
- Smaller subsets of length k



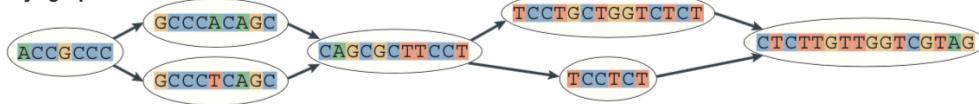
a Generate all substrings of length k from the reads



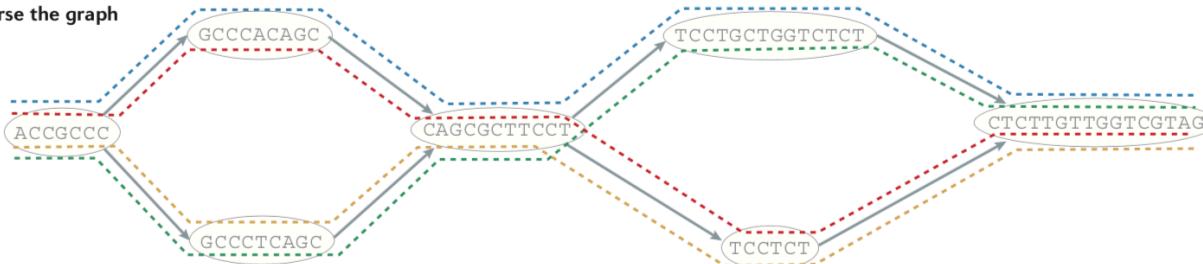
b Generate the De Bruijn graph



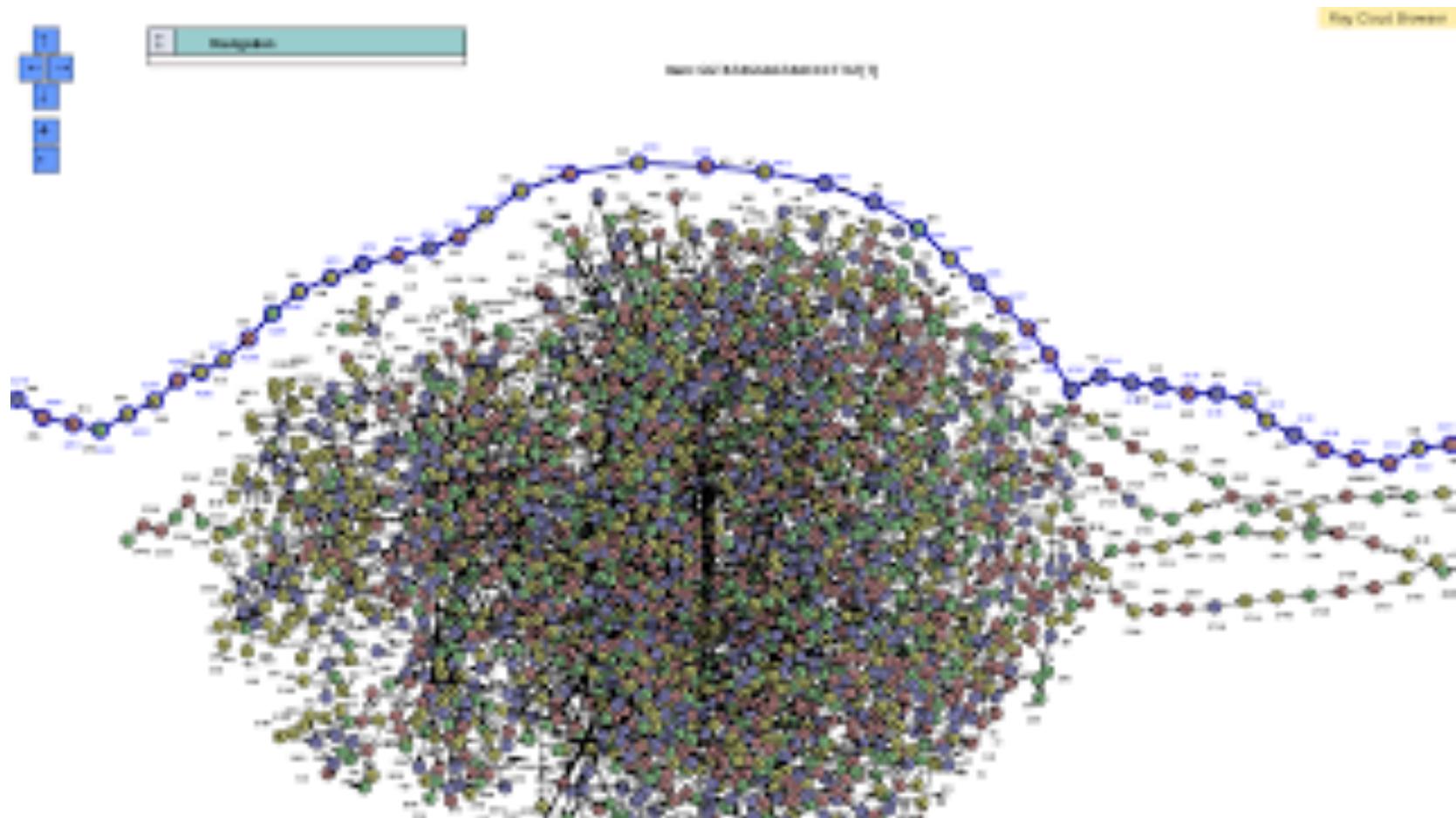
c Collapse the De Bruijn graph



d Traverse the graph



In real life with 10^6 sequences



What makes assembly tricky?

Many pieces (computationally-intensive)

Errors in sequence (which one is correct?)

Missing fragments

Repetitive fragments

Multiple copies (rRNA gene as an example)

Circular genome (no starting point)

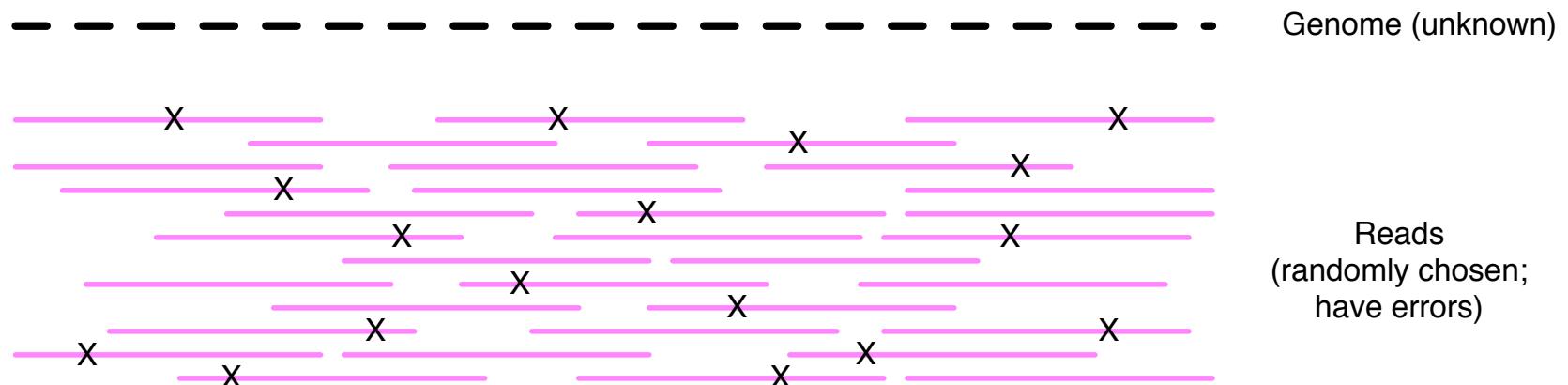
Choice of k-mer

- Too small = misassembly (anything can assemble)
- Too long = no assembly

Sequencing coverage

Coverage describes the number of time that each base of the genome is present in the reads

Assemblers work best at 60-100x coverage



Estimated coverage for our strain

- *Serratia* genomes: ~5 Mb
- Received data
 - 603,082 pairs
 - 326 + 278 bp
- Coverage
 - **Number of bases sequenced/genome size**
 - Bases = $(603,082 \times 326 \text{ bp}) + (603,082 \times 278 \text{ bp})$
 $196,604,732 \text{ bp} + 167,656,796 \text{ bp} = 364,261,528 \text{ bp}$
 - Coverage = $364,261,528 \text{ bp} / 5,000,000 \text{ bp} = 72.8$

Assemblers

Bioinformatic tools that combine short sequencing reads into longer contigs

For our genome, we will use one assembler:

SPADES

- <http://bioinf.spbau.ru/spades>

```
#!/bin/bash -l
#SBATCH --job-name spades
#SBATCH --output spades_out_%j.txt
#SBATCH --error spades_err_%j.txt
#SBATCH --time 2:00:00
#SBATCH --nodes 1
#SBATCH --ntasks-per-node 1
#SBATCH --cpus-per-task 4
#SBATCH --mem 5000
#SBATCH --account project_2001379

module load biokit

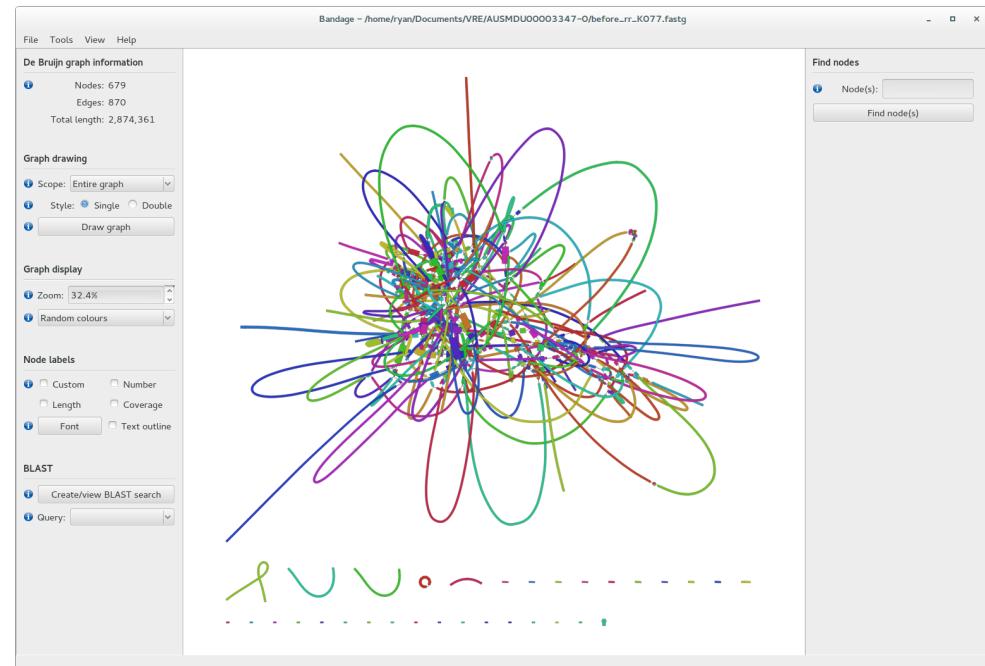
spades.py -1 Antton_R1_trimmed.fastq \
          -2 Antton_R2_trimmed.fastq \
          -o SPADES_ANTTON \
          -t 4 \
          --isolate
```

SPADES output

contigs.fasta: this contains the contigs generated by SPADES

scaffolds.fasta: contigs generated by SPADES after repeat resolution and scaffolding using paired-end information

assembly_graph.fastg: the assembly graph generated by SPADES



Additional notes about Puhti

Batch jobs

`sbatch`

`scancel`

`squeue`

`seff`

Job ID: 3750729

Cluster: puhti

User/Group: stelmach/stelmach

State: COMPLETED (exit code 0)

Nodes: 1

Cores per node: 4

CPU Utilized: 00:15:57

CPU Efficiency: 74.53% of 00:21:24 core-walltime

Job Wall-clock time: 00:05:21

Memory Utilized: 2.49 GB

Memory Efficiency: 51.06% of 4.88 GB

Job consumed 0.40 CSC billing units based on following used resources

CPU BU: 0.36

Mem BU: 0.04

Assembly quality

What is a good assembly?

Good coverage throughout the contigs

Correct size (for bacteria, not 400 k or 14 Mb)

As fewer contigs as possible

Similar GC across contigs

QUAST

<http://bioinf.spbau.ru/quast>

Tool to evaluate the quality of assemblies

After running QUAST, move to your computer with FileZilla:

- report.html
- report.pdf
- icarus.html
- icarus_viewers (folder)

Let's assemble our genome

<https://github.com/igorspp/MMB-114>

(Day 3: Genome assembly)