# Intro to NGS

## — Assembly: practical examples —

*Zemin Ning, Francesca Giordano*

# How to install the course folder

## — Already installed in the EBI pc! —

- Open a terminal and clone the folder from github:

  $ git clone https:/github.com/fg6/EBI_NGS_Assembly.git

- Move inside the newly created folder EBI_NGS_Assembly:

  $ cd EBI_NGS_Assembly

- Launch the install script:

  $ ./scripts/install.sh

(More details : https://github.com/fg6/EBI_NGS_Assembly)

# What's in the course folder

$ ls *

https://github.com/fg6/EBI_NGS_Assembly


README.md

data:

Escherichiacoli-K-12.fasta  fastq  results

scripts:

check_assembly.sh  install.sh  miniasm.sh  runACT.sh  spades_hybrid.sh spades.sh

src:

forACT miniasm  minimap2  MUMmer3.23  spades

# Data folder

$ ls data/

Escherichiacoli-K-12.fasta  fastq/  results/

E.Coli Reference Assembly  (Circular genome)

Bases= **4,641,652**  Seqs= **1**

$ head -2 data/Escherichiacoli-K-12.fasta

> gi|556503834|ref|NC_000913.3|  Escherichia coli str. K-12 substr. MG1655, complete genome

AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC

# Data folder

$ ls data/fastq

miseq1.fastq  miseq2.fastq  nanopore.fastq

**Nanopore long reads:**

Bases= 225,086,343 Seqs= 30,364 **Mean_length= 7,413 Longest= 45,588**

Read depth (Rough estimate):  Sum_of_Bases / Reference_Bases:

225,086,343/4,641,652 = **49 x**

**Illumina short, paired reads:**

miseq1.fastq:  Bases= 1,927,922,599  Seqs= 12,808,979  **Length=151**  bases

Read depth (Rough estimate):  Sum_of_Bases / Reference_Bases:

2* 1,927,922,599/4,641,652 = **388 x**

# Data folder

$ ls data/results/*

data/results/act:

act_hybridspades_contigs  act_hybridspades_scaffolds  act_miniasm
act_spades_contigs  act_spades_scaffolds

data/results/assemblies:

hybridspades_contigs.fasta  hybridspades_scaffolds.fasta  miniasm.fasta
spades_contigs.fasta  spades_scaffolds.fasta

These are the assemblies and the alignment files for the ACT visualisation that the pipelines create. They are added here to compare with your results, but also because some of the pipelines takes too long and we will not be running them: if you wish you can try to run them on your own.

# Check an assembly Stats:

## scripts/check_assembly.sh

Once an assembly is generated it needs to be assessed: evaluating a few stats can be helpful to understand the quality of your assembly.

We can use the script scripts/check_assembly.sh to estimates a few stats and assess the average identity with respect to the reference.

check_assembly.sh has 2 steps:

1. an executable called n50 that estimates assembly total number of bases, total number of contigs and more stats

2. dnadiff: a pipeline from MUMmer that map the assembly against the reference and estimate percentage of the reference covered and average identity of the assembly with respect to the reference (among other things..)

```
$ ./scripts/check_assembly.sh

    Usage: check_assembly.sh assembly.fasta reference.fasta
```

# Check an assembly stats

```
$ ./scripts/check_assembly.sh  data/results/assemblies/spades_contigs.fasta  \
    data/Escherichiacoli-K-12.fasta

        Bases= 4611519 contigs= 279 mean_length= 16529 longest= 414005 N50= 129100 n= 12

        Assembly stats saved in /home/training/EBI_NGS_Assembly/results/report/
        report_spades_contigs/n50.stats

        Dnadiff results are in file /home/training/EBI_NGS_Assembly/results/report/
        report_spades_contigs/out.report

        AvgIdentity                99.97           99.97
```

# Check an assembly stats

$ head -25 /home/training/EBI_NGS_Assembly/results/report/report_spades_contigs/out.report

| | [REF] | [QRY] |
|---|---|---|
| [Sequences] | | |
| TotalSeqs | 1 | 279 |
| AlignedSeqs | 1(100.00%) | 158(56.63%) |
| UnalignedSeqs | 0(0.00%) | 121(43.37%) |
| | | |
| [Bases] | | |
| TotalBases | 4641652 | 4611519 |
| AlignedBases | 4641625(100.00%) | 4575366(99.22%) |
| UnalignedBases | 27(0.00%) | 36153(0.78%) |
| | | |
| [Alignments] | | |
| 1-to-1 | 134 | 134 |
| TotalLength | 4570587 | 4570589 |
| AvgLength | 34108.86 | 34108.87 |
| AvgIdentity | 100.00 | 100.00 |
| | | |
| M-to-M | 296 | 296 |
| TotalLength | 4666135 | 4666149 |
| AvgLength | 15763.97 | 15764.02 |
| AvgIdentity | 99.97 | 99.97 |

# Exercise 1: Assembly the long reads using MiniAsm

The pipeline has two steps:

1. First maps the reads all against all (using the aligner MiniMap2)

2. Then draws a graph from the mapping and finds the best path through the graph

Generate a MiniAsm assembly from long reads launching:

```
$ ./scripts/miniasm.sh
```

Questions:

1. How much time the script ran for?
2. How many bases has the assembly? How many contigs?

# Exercise 1: — Results

Questions:

1. How much time the script ran for?

   Launch in this way to find out:

   ```
   $ time ./scripts/miniasm.sh
   ```

2. How many bases has the assembly? How many contigs?

   Bases: 4,544,451

   Contigs: 1

# Exercise 2: — Compare long reads and short reads-based assemblies

Check the assembly stats for the long read assembly:

results/miniasm/miniasm.fasta

and the short read assembly from SPAdes:

data/results/assemblies/spades_contigs.fasta

Questions:

1. Which has the higher average identity? Can you guess why?
2. Which has the longest contigs? Can you guess why?
3. How many sequences are aligned in both cases? How much of the reference is covered?

# Exercise 2: — Compare long reads and short reads -based assemblies

Questions:

1. Which has the higher average identity? Can you guess why

The SPAdes assembly has higher avg id: 99.97%, against the miniasm avg id: 88.90%.

The reason is that the illumina reads are more accurate and contain less base-errors (<~1%) than the Oxford Nanopore reads (up to 20%).

Also: higher depth means higher accuracy, and we saw the short reads have higher depth.

2. Which has the longest contigs? Can you guess why?

MiniAsm generates a unique contig that cover the whole reference. The shorter the reads the more fragmented the assembly.

3. How many sequences are aligned in both cases? How much of the reference is covered?

MiniAsm : 1 seq aligned, covers 95% of reference; SPAdes: 158 seqs aligned, cover 100% of reference

# Exercise 3: — Hybrid assembly

Check the assembly stats for the hybrid assembly from SPAdes:

data/results/assemblies/hybridspades_contigs.fasta

Questions:

1. What do you think “hybrid” stands for?
2. How many contigs has the assembly?
3. Which is the avg id?
4. How many sequences are aligned? How much of the reference is covered?
5. Compare these findings with the MiniAsm and SPAdes cases

# Exercise 3: — Hybrid assembly

1. What do you think “hybrid” stands for?

An hybrid assembler uses both short and long reads. HybridSPAdes assembles the short reads as SPAdes, but then uses the long reads to scaffold the obtained contigs.

2. How many sequences has the assembly? 128 sequences

3. Which is the avg id? 99.99%

4. How many sequences are aligned? How much of the reference is covered?

7 sequences are aligned, and they cover the 100% of the reference

5. Compare these findings with the MiniAsm and SPAdes cases

An hybrid assembler is able to take the good of both short reads (high accuracy) and long reads (less fragmented contigs).

# Exercise 3: — Hybrid assembly

The 10 Longest Seqs are:

Chr NODE_1_length_3108520_cov_429.620249 lenght= 3,108,520 bp

Chr NODE_2_length_1006019_cov_433.455439 lenght= 1,006,019 bp

Chr NODE_3_length_515997_cov_420.219643 lenght= 515,997 bp

Chr NODE_4_length_5463_cov_4151.251578 lenght= 5,463 bp

Chr NODE_5_length_2120_cov_350.857562 lenght= 2,120 bp

Chr NODE_6_length_457_cov_409.468421 lenght= 457 bp

Chr NODE_7_length_417_cov_386.900000 lenght= 417 bp

Chr NODE_8_length_406_cov_0.662614 lenght= 406 bp

Chr NODE_9_length_348_cov_0.675277 lenght= 348 bp

Chr NODE_10_length_333_cov_0.726562 lenght= 333 bp

The Shortest Seq is NODE_127_length_78_cov_41671.000000 lenght= 78 bp

# Visualisation: Artemis ACT

http://www.sanger.ac.uk/science/tools/artemis

ACT usage:

```
$ act  reference.fasta  alignments.al  assembly.fasta
```

The alignments.al is an alignment file that list all the assembly sequences aligned to the reference in a specific format required by ACT;  assembly.fasta is not the original assembly but has the sequences positioned in order of appearance in the reference for a clearer visualisation.

To create the alignment.al file and the re-ordered assembly.fasta you can use **script/runACT.sh**: this will create the files and then launch ACT.
Alternatively, you will find all the act files in the data folder  **data/results/act**:

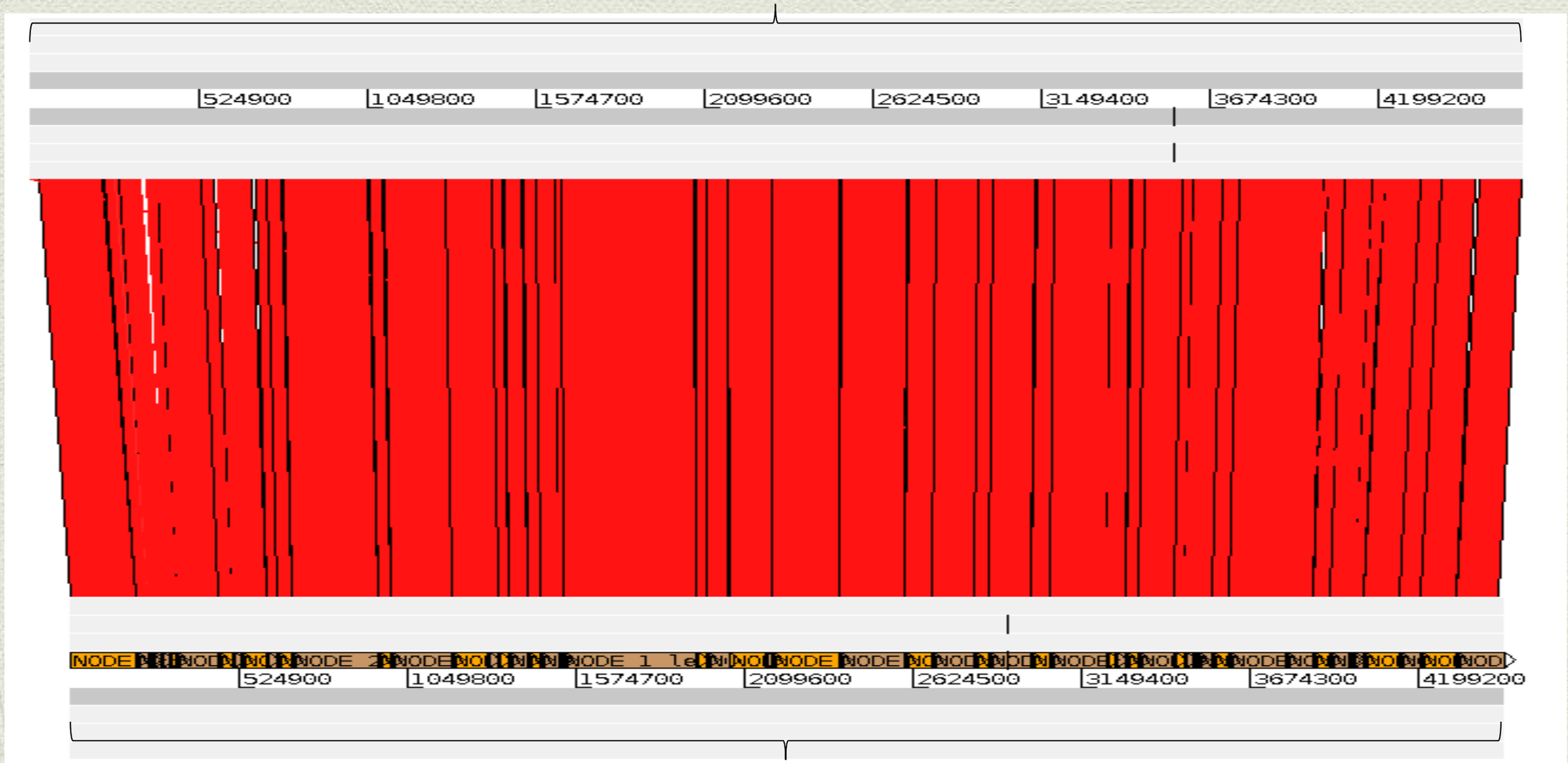
```
act_miniasm/  act_spades_contigs/ act_hybridspades_contigs/
```

For instance for SPAdes:

```
$ act  data/Escherichiacoli-K-12.fasta  \
    data/results/act/act_spades_contigs/foractnonoise0.2_minid10.al \
    data/results/act/act_spades_contigs/foractnonoise0.2_minid10.fasta
```

# ACT for SPAdes contigs

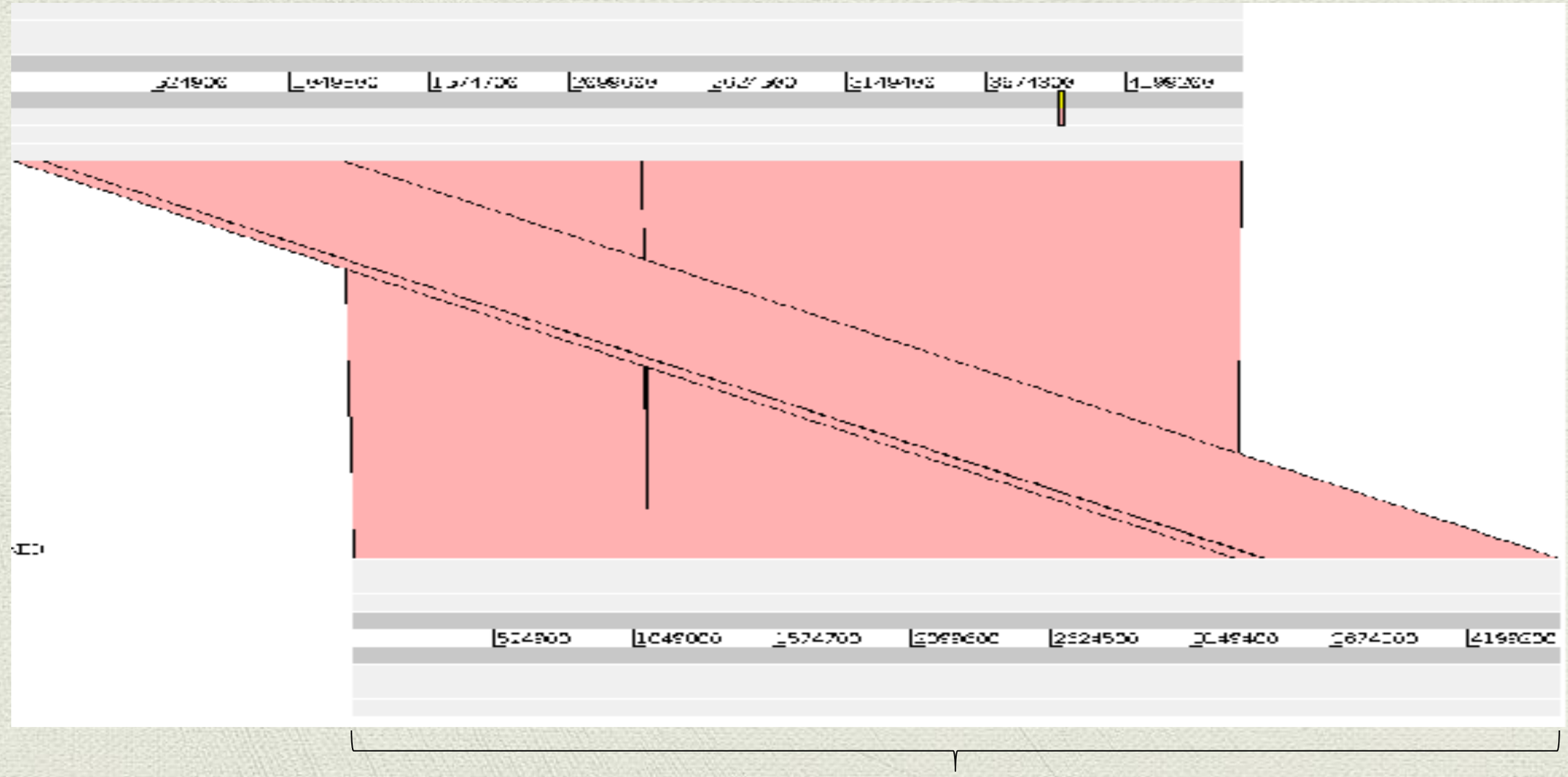
Reference: 1 Sequence, ~4.6 M bases




SPAdes assembly: ~200 Sequences, ~4.6 M bases

# ACT for MiniAsm contigs

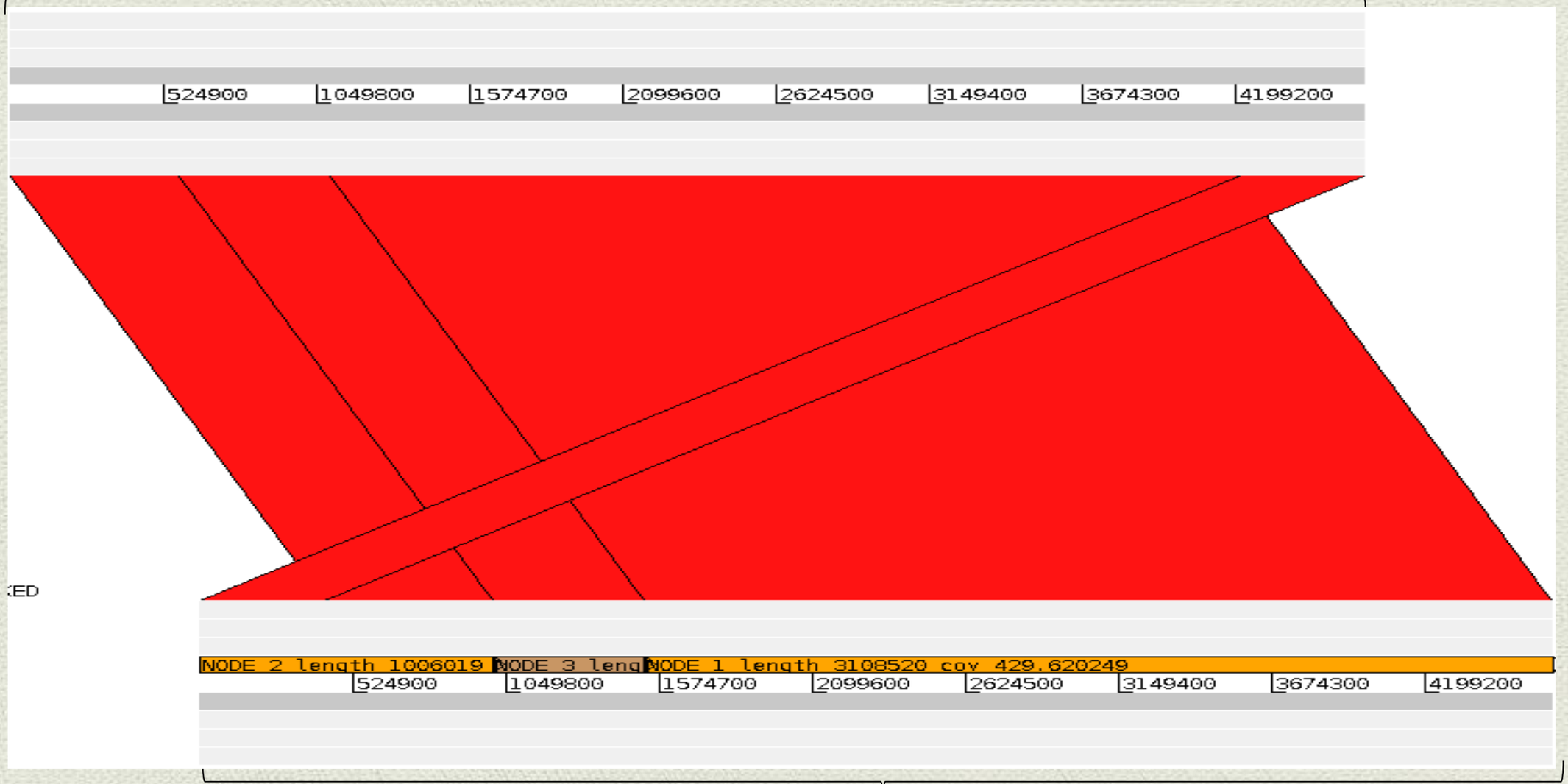
Reference: 1 Sequence, ~4.6 M bases



MiniAsm assembly: 1 Sequence, ~4.5 M bases

# ACT for HybridSPAdes contigs

Reference: 1 Sequence, ~4.6 M bases




HybridSPAdes assembly: 3 Sequences, ~4.6 M bases