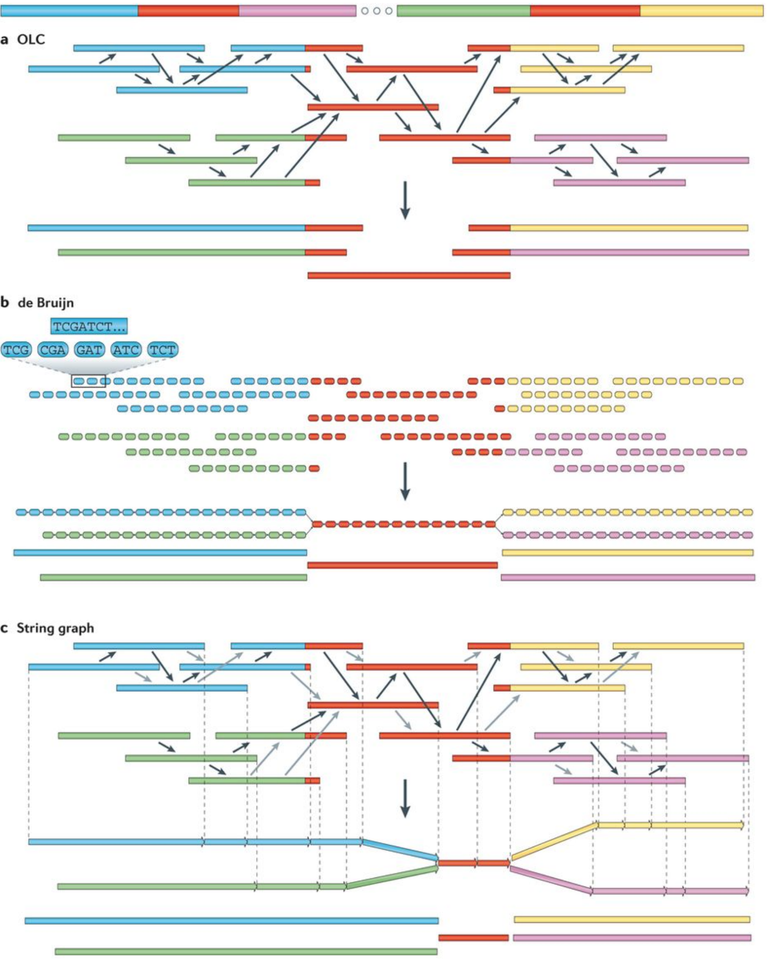
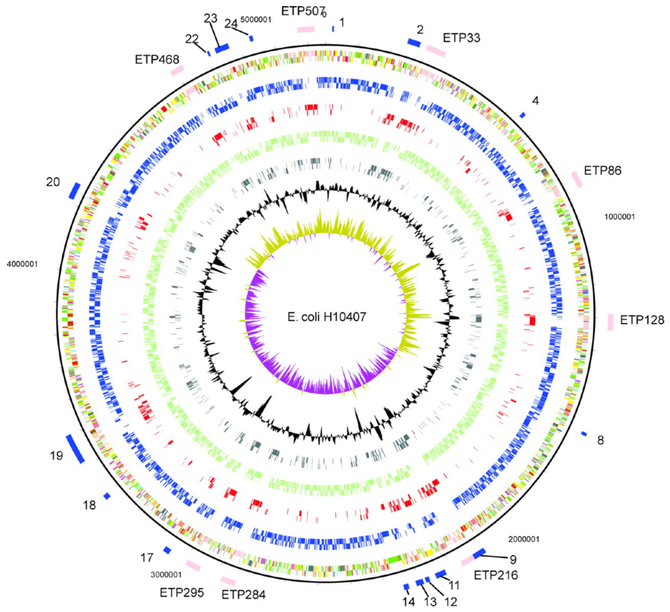
Genome assembly Practical





HTS Workshop Genomics & Transcriptomics
KAUST 2019

Robert Lehmann
Octavio Salazar

Preparation

First Steps - Preparing the folder structure

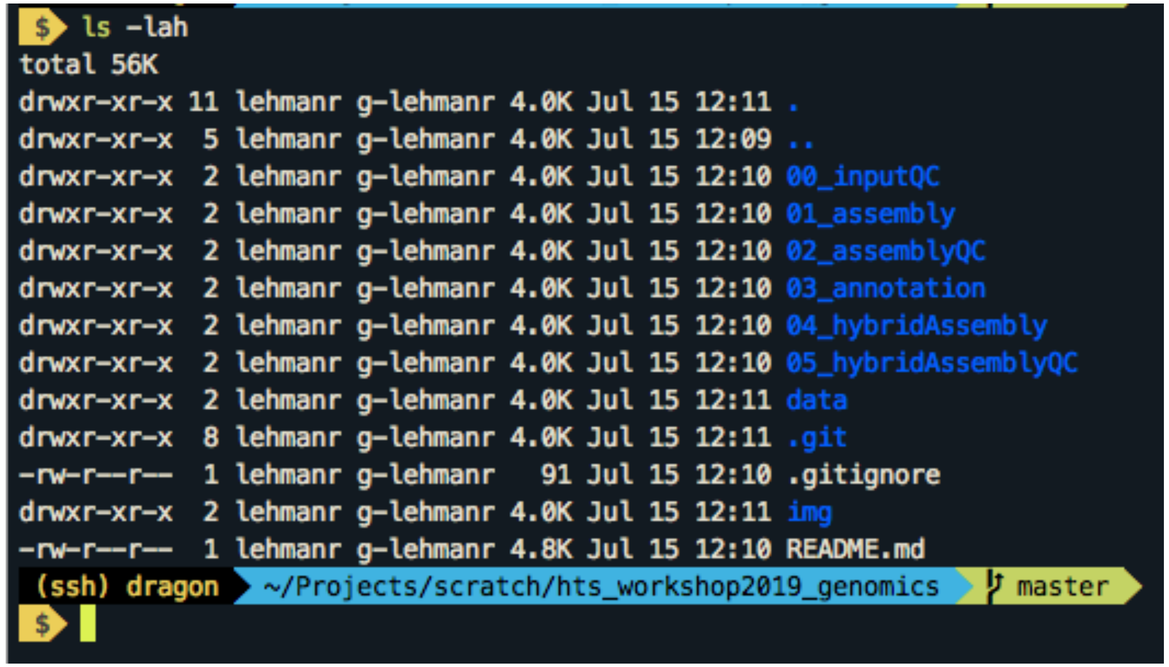
<u>Download scripts and reference data for analyses:</u>

In order to download the scripts and reference data for the analyses, we first need to login to the Ibex cluster:

$ ssh <username>@ilogin.ibex.kaust.edu.sa

Then, we download the prepared scripts from github:

$ mkdir < workshop dir >
$ cd <workshop dir>
$ git clone https://github.com/roblehmann/hts_workshop2019_genomics.git
$ cd *hts_workshop2019_genomics*



First Steps - Preparing the folder structure

Some of the tools we will use require additional data, such as a reference genome assembly and annotation for E. coli. Let's download these data before starting the assembly:

$ cd data && bash copyDataFromIbex.sh

The result should look like this

```
$ ls -laht
total 7.7G
-rw-r--r--  1 lehmanr g-lehmanr 1.7M Jul 15 15:02 GCF_000005845.2_ASM584v2_protein.faa
drwxr-xr-x  3 lehmanr g-lehmanr 4.0K Jul 15 15:02 .
-rw-r--r--  1 lehmanr g-lehmanr 2.8M Jul 15 15:02 GCF_000005845.2_ASM584v2_genomic.gff
-rw-r--r--  1 lehmanr g-lehmanr 4.5M Jul 15 15:02 GCF_000005845.2_ASM584v2_genomic.fna
-rw-r--r--  1 lehmanr g-lehmanr 1.7M Jul 15 15:02 taxo.k2d
-rw-r--r--  1 lehmanr g-lehmanr   56 Jul 15 15:02 opts.k2d
-rw-r--r--  1 lehmanr g-lehmanr 7.5G Jul 15 15:02 hash.k2d
-rwxr-xr-x  1 lehmanr g-lehmanr  570 Jul 15 15:01 copyDataFromIbex.sh
drwxr-xr-x  5 lehmanr g-lehmanr 4.0K Jul 15 15:00 enterobacteriales_odb9
drwxr-xr-x 11 lehmanr g-lehmanr 4.0K Jul 15 14:58 ..
-rw-r--r--  1 lehmanr g-lehmanr  40M Jul 15 14:58 tardigrade_SRR2986339_subsampled_2.fq.gz
-rw-r--r--  1 lehmanr g-lehmanr  39M Jul 15 14:58 tardigrade_SRR2986339_subsampled_1.fq.gz
-rwxr-xr-x  1 lehmanr g-lehmanr 1.1K Jul 15 14:58 downloadData.sh
-rw-r--r--  1 lehmanr g-lehmanr  31M Jul 15 14:58 SRR2627019.1m.80xR2.fq.gz
-rw-r--r--  1 lehmanr g-lehmanr  24M Jul 15 14:58 SRR2627019.1m.80xR1.fq.gz
-rw-r--r--  1 lehmanr g-lehmanr  20M Jul 15 14:58 SRR2627019.1m.40xR2.fq.gz
-rw-r--r--  1 lehmanr g-lehmanr  16M Jul 15 14:58 SRR2627019.1m.40xR1.fq.gz
-rw-r--r--  1 lehmanr g-lehmanr 7.6M Jul 15 14:58 SRR2627019.1m.20xR2.fq.gz
-rw-r--r--  1 lehmanr g-lehmanr 6.0M Jul 15 14:58 SRR2627019.1m.20xR1.fq.gz
-rw-r--r--  1 lehmanr g-lehmanr 3.9M Jul 15 14:58 SRR2627019.1m.10xR2.fq.gz
-rw-r--r--  1 lehmanr g-lehmanr 3.1M Jul 15 14:58 SRR2627019.1m.10xR1.fq.gz
-rw-r--r--  1 lehmanr g-lehmanr 4.4M Jul 15 14:58 SRR1284073.1m.5x.pacbio.fastq.gz
-rw-r--r--  1 lehmanr g-lehmanr 1.7K Jul 15 14:58 README.md
```

Quality control

Step 0 - Read data quality control

00_inputQC

```
$ cd 00_inputQC && ls -lah
total 0
drwxr-xr-x  2 lehmanr g-lehmanr 4.0K Jul 15 14:58 .
drwxr-xr-x 11 lehmanr g-lehmanr 4.0K Jul 15 14:58 ..
-rwxr-xr-x  1 lehmanr g-lehmanr  414 Jul 15 14:58 0-runFastQC.sh
-rwxr-xr-x  1 lehmanr g-lehmanr 1.1K Jul 15 14:58 1-runTrimming.sh
-rwxr-xr-x  1 lehmanr g-lehmanr  295 Jul 15 14:58 2-runMultiQC.sh
-rwxr-xr-x  1 lehmanr g-lehmanr  542 Jul 15 14:58 3-runKraken.sh
-rwxr-xr-x  1 lehmanr g-lehmanr  490 Jul 15 14:58 4-runKAT.sh
```

Job configuration part for slurm

Comment

Make fastqc available

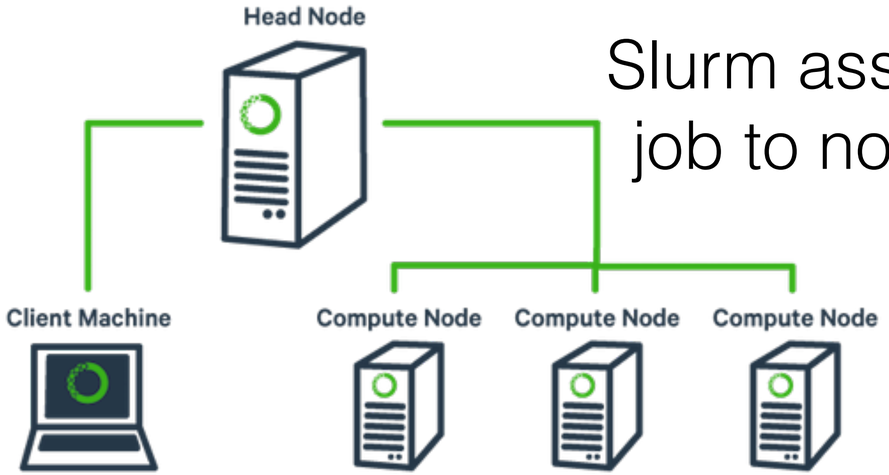
Actual command

```
#!/bin/bash
#SBATCH --job-name=fastqc
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=2
#SBATCH --time=2:00:00
#SBATCH --mem=5000
#SBATCH --mail-type=end

# fastQC
# https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

set -vex

module load fastqc

outDir='output'
mkdir -p $outDir

threads=2

for f in ../data/*fq.gz; do
        echo "fastQC'ing file"${f}
        fastqc $f -o ${outDir} -t ${threads}
done
```

how to edit file:
$ nano 0-runFastQC.sh

Slurm - Small refresher

00_inputQC



Slurm assigns job to nodes

How to give a new job to Slurm?

sbatch <job_script.sh>
or
run <job_script.sh>

What is the cluster doing?

```
$ squeue | head -n 10
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
5457607_[338-2000] group-sts   T-2-17   lenzia PD       0:00      1 (Resources)
           5458107     batch BandR+UF  bordjiz PD       0:00      4 (Priority)
$ squeue | tail -n 10
           5451125     batch   case_2   zhongp  R   14:55:06      1 ds505-17
           5451101     batch   case_2   zhongp  R   14:55:07      1 ds506-25
           5451103     batch   case_2   zhongp  R   14:55:07      1 ds503-13
           5451104     batch   case_2   zhongp  R   14:55:07      1 ds505-29
           5451105     batch   case_2   zhongp  R   14:55:07      1 ds506-05
           5435734     batch g09_m1_0  minenky  R 1-03:17:42      1 ds506-29
           5292423     batch hc_00_pm  minenky  R 3-16:38:16      1 dbn302-06-r
           5292441     batch hc_00_pm  minenky  R 3-16:38:16      1 dbn302-07-l
           5292442     batch hc_00_pm  minenky  R 3-16:38:16      1 dbn302-07-r
           5458042     batch     bash  ramazam  R    3:38:55      1 cn603-07-r
```

What am I doing (on the cluster)?

```
$ squeue -u lehmanr
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

Lets try to run FastQC

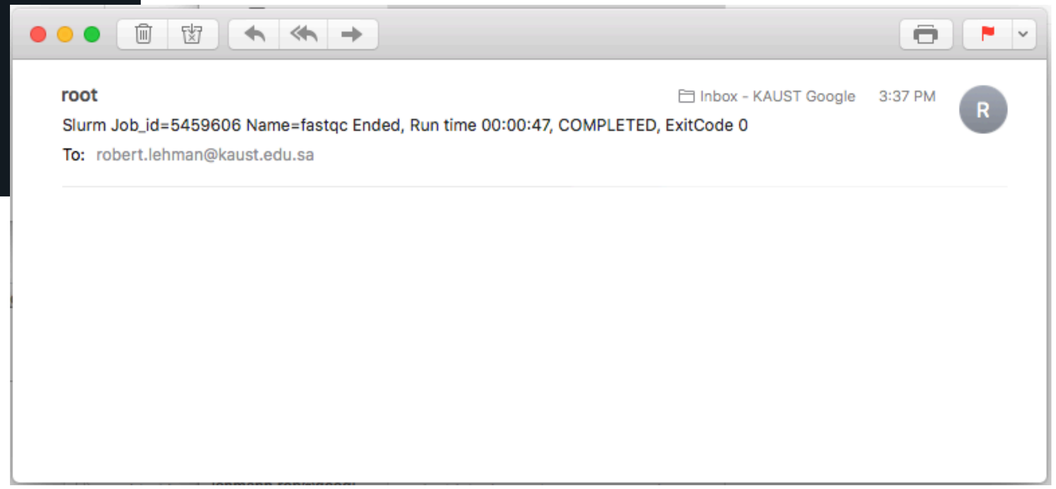
00_inputQC

Submit FastQC job

Check if its running

```
$ sbatch 0-runFastQC.sh
Submitted batch job 5459606
$ squeue -u lehmanr
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           5459606     batch   fastqc  lehmanr  R       0:44      1 cn605-07-l
$ ls -lah
total 256K
drwxr-xr-x  3 lehmanr g-lehmanr 4.0K Jul 15 15:36 .
drwxr-xr-x 11 lehmanr g-lehmanr 4.0K Jul 15 14:58 ..
-rwxr-xr-x  1 lehmanr g-lehmanr  414 Jul 15 14:58 0-runFastQC.sh
-rwxr-xr-x  1 lehmanr g-lehmanr 1.1K Jul 15 14:58 1-runTrimming.sh
-rwxr-xr-x  1 lehmanr g-lehmanr  295 Jul 15 14:58 2-runMultiQC.sh
-rwxr-xr-x  1 lehmanr g-lehmanr  542 Jul 15 14:58 3-runKraken.sh
-rwxr-xr-x  1 lehmanr g-lehmanr  490 Jul 15 14:58 4-runKAT.sh
drwxr-xr-x  2 lehmanr g-lehmanr 4.0K Jul 15 15:36 output
-rw-r--r--  1 lehmanr g-lehmanr  15K Jul 15 1[illegible]7 slurm-5459606.out
```

Email when job is done



FastQC output

Slurm's log file

00_inputQC

How to look at the results

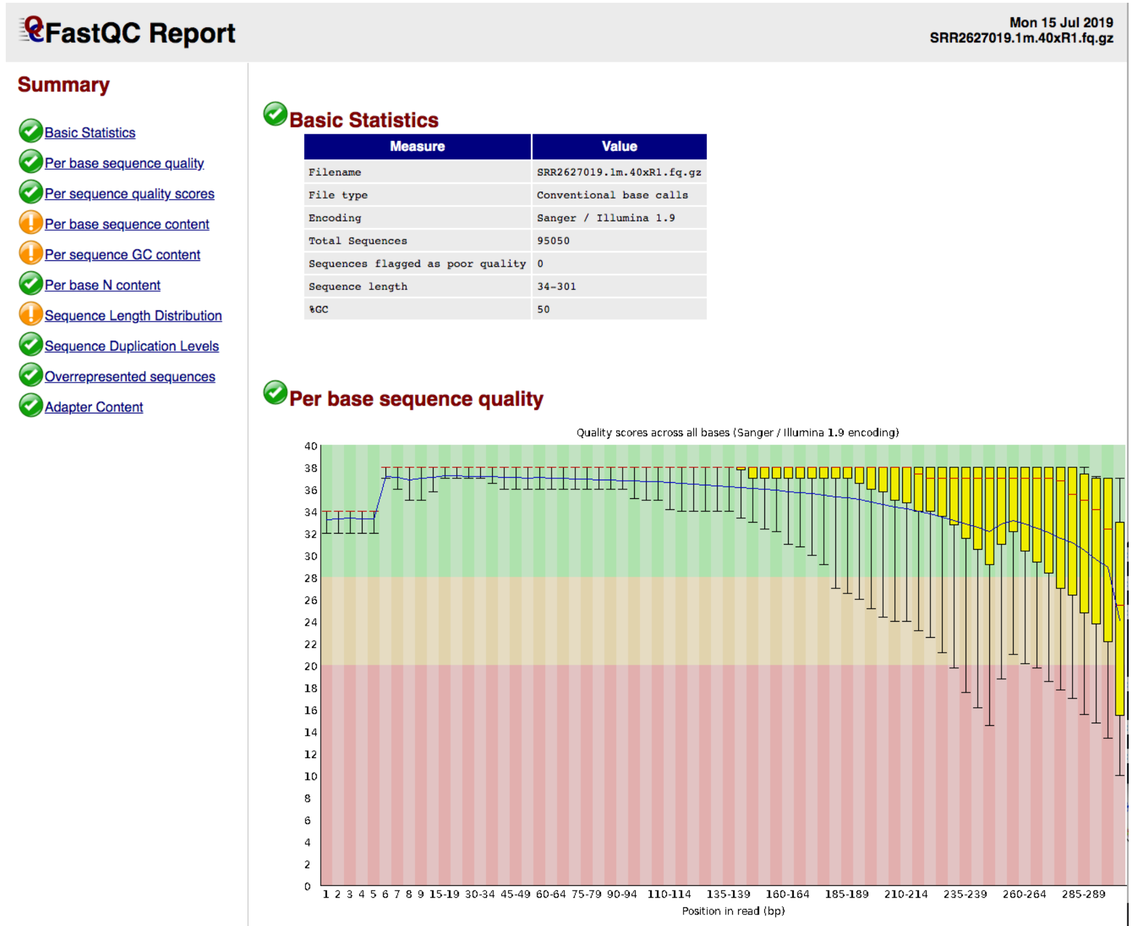
Check the content of the “output” folder

$ ls -lah output

Download the html files to your local computer and open:

$ scp <user>@ilogin.ibex.kaust.edu.sa:<path to output folder>/SRR2627019.1m.40xR1_fastqc.html
$ scp <user>@ilogin.ibex.kaust.edu.sa:<path to output folder>/SRR2627019.1m.40xR2_fastqc.html

This is very tedious for many samples -> need to summarize reports…



Trimming the reads

Read script 1-runTrimming.sh:

$ nano 1-runTrimming.sh

Submit trimming job for one coverage dataset to cluster:

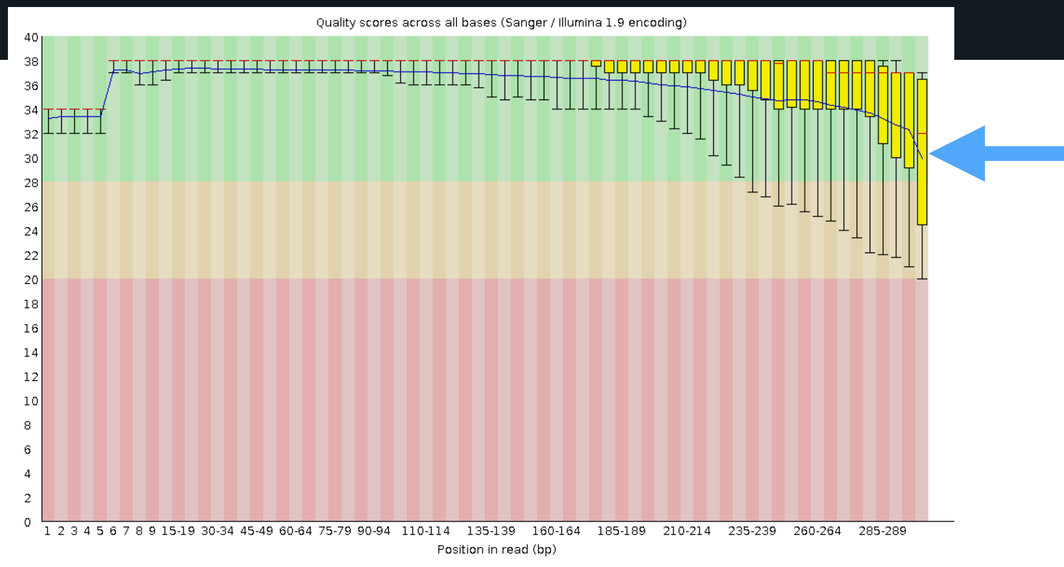
$ sbatch 1-runTrimming.sh

Have a look at the trimming result:

$ nano output/ecoli_40x_trimmomatic.summary
$ scp <user>@ilogin.ibex.kaust.edu.sa:<path to output folder>/SRR2627019.1m.40x_trimmed_R1_fastqc.html

Edit script to trim datasets for varying coverage and resubmit…

```
 GNU nano 2.3.1                          File: ecoli_40x_trimmomatic.summary

Input Read Pairs: 95050
Both Surviving Reads: 88019
Both Surviving Read Percent: 92.60
Forward Only Surviving Reads: 6638
Forward Only Surviving Read Percent: 6.98
Reverse Only Surviving Reads: 151
Reverse Only Surviving Read Percent: 0.16
Dropped Reads: 242
Dropped Read Percent: 0.25
```



Contamination check with Kraken2

00_inputQC

Have a look at the script

$ nano 2-runKraken.sh

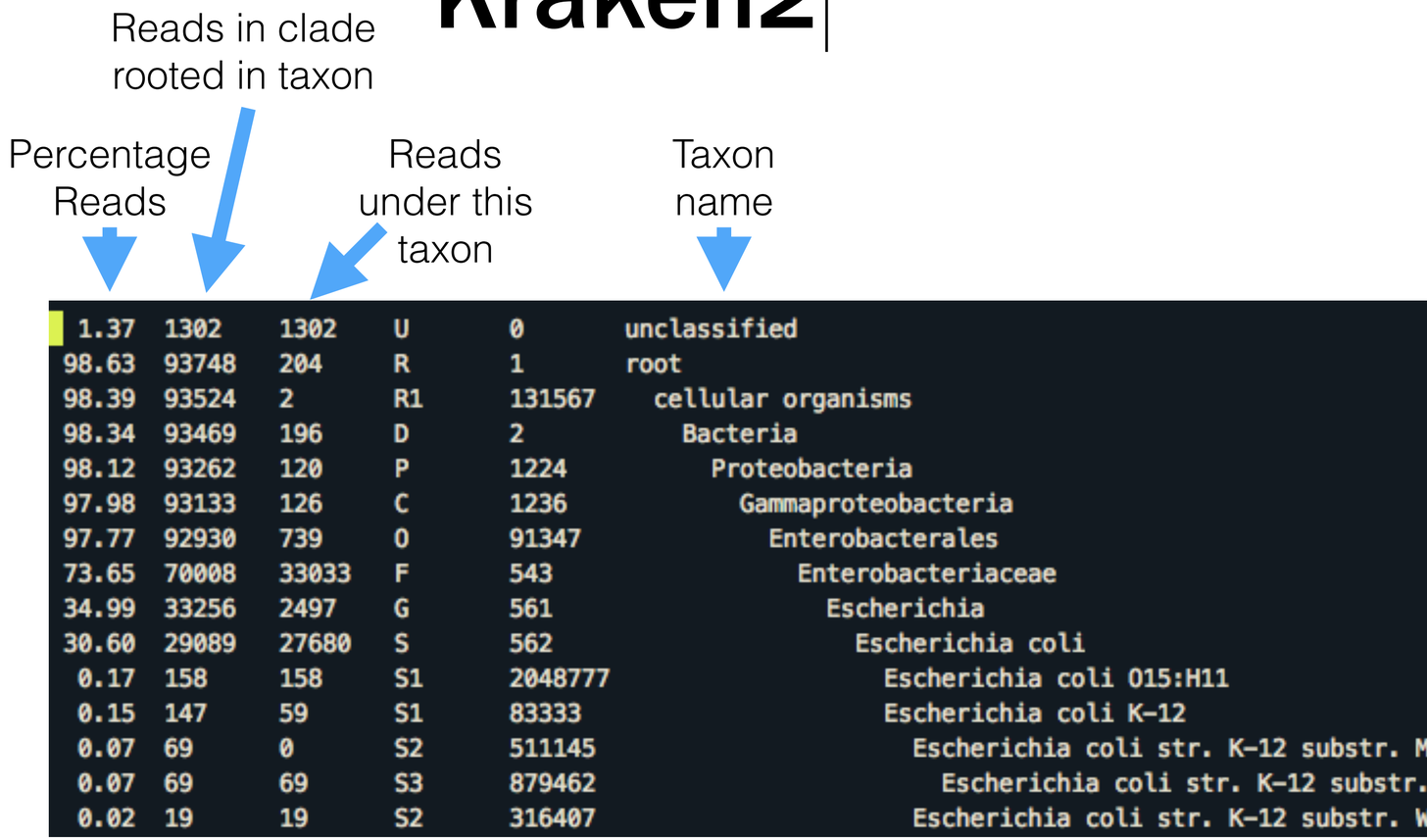
And submit as before:

$ batch 2-runKraken.sh

The output file is a flatfile, so we can check the result on IBEX using nano:

$ nano output/ecoli_40x_kraken2_report.txt

And modify parameters to run on the other input datasets as before…



How to find which taxons have many reads assigned?
-> awk

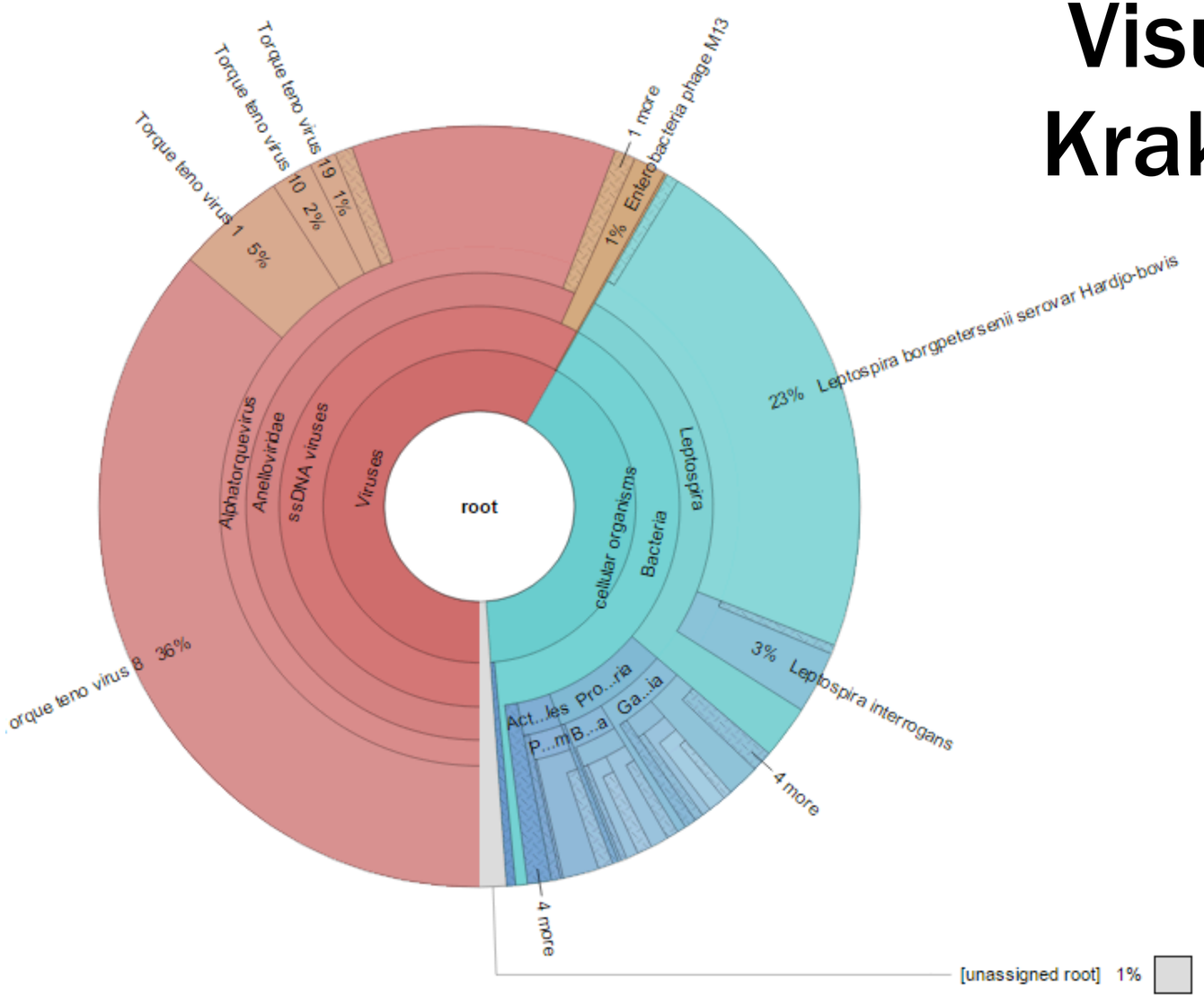
Contamination check with Kraken2


```
$ awk '$1 > 1 {print}' ecoli_kraken2_report.txt | wc -l
13
```

```
[lehmanr@dbn503-35-r output]$ awk '$1 > 1' ecoli_kraken2_report.txt
 99.67  94735   298     R       1       root
 99.35  94436   2       R1      131567    cellular organisms
 99.30  94387   203     D       2           Bacteria
 99.08  94175   257     P       1224          Proteobacteria
 98.80  93909   69      C       1236            Gammaproteobacteria
 98.72  93831   917     O       91347             Enterobacterales
 74.22  70550   39261   F       543                 Enterobacteriaceae
 31.88  30298   1603    G       561                   Escherichia
 29.75  28282   27968   S       562                     Escherichia coli
 23.52  22356   15      F       1903411             Yersiniaceae
 23.50  22337   205     G       629                   Yersinia
 23.28  22124   16426   G1      1649845                 Yersinia pseudotuberculosis complex
  5.76  5475    5410    S       632                       Yersinia pestis
```

00_inputQC

Visualization of Kraken2 results



Krona

https://github.com/marbl/Krona

Contamination check with KAT

00_inputQC

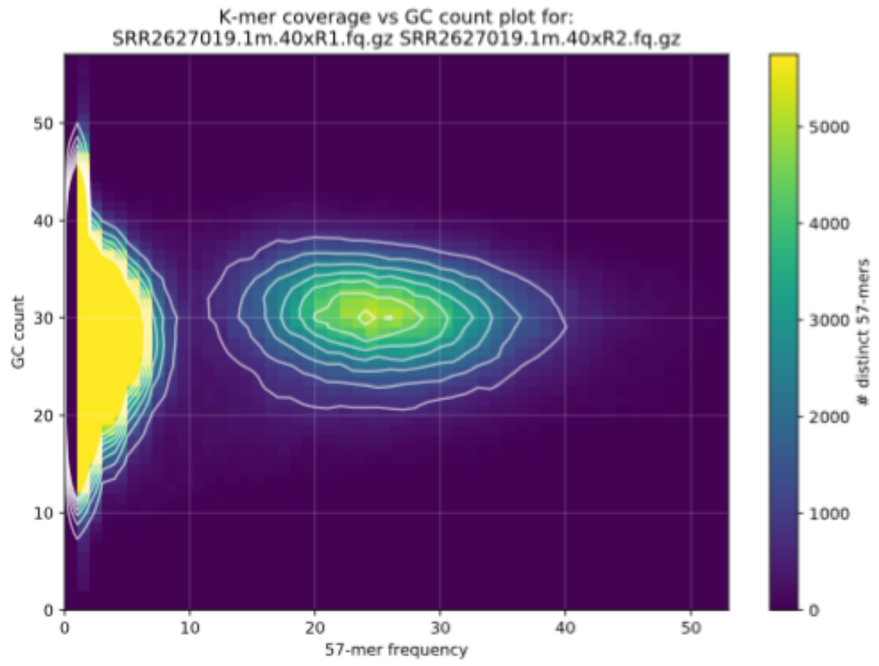
And also the GC vs. Coverage plots:

$ nano 3-runKAT.sh
$ sbatch 3-runKAT.sh

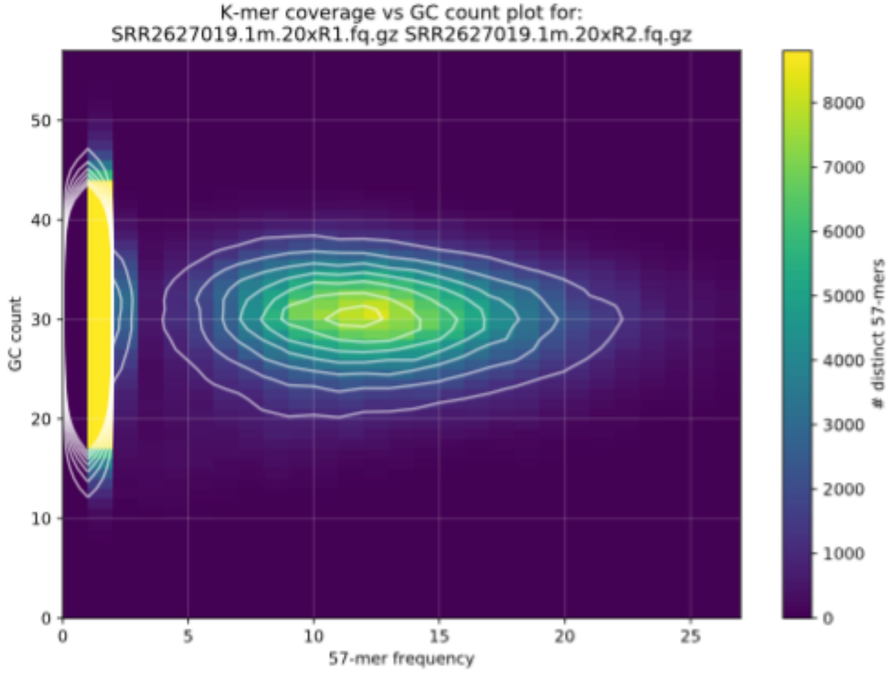
Change to dataset with different coverage, and re-run.
Then download all run results to your machine:

$ scp "<user>@ilogin.ibex.kaust.edu.sa:<path to output folder>/00_inputQC/output/*pdf" .

40x dataset



20x dataset



Now we’re stuck with many individual reports, how to compressed all that information?

MultiQC to the rescue

Summarize QC results

00_inputQC

Run MultiQC:

$ sbatch 4-runMultiQC.sh

Copy report to the local machine:

$ scp -r "<user>@idragon:/home/lehmanr/Projects/scratch/hts_workshop2019/00_inputQC/multiqc_*" .

And view the results:

$ open multiqc_report.html

Copy table | Configure Columns | Plot | Showing 20/20 rows and 4/5 columns.

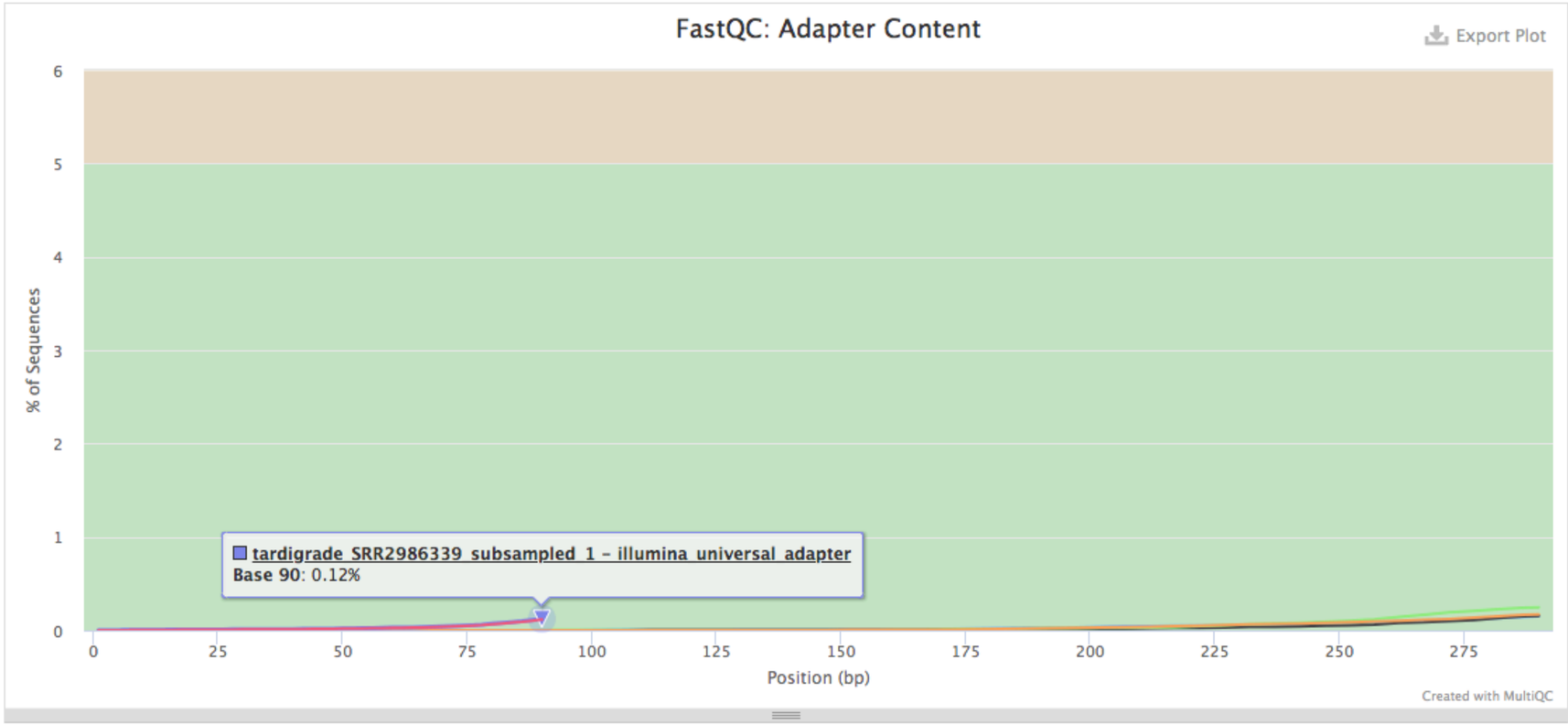
Sample Name	% Dups	% GC	Length	M Seqs
SRR2627019.1m.10xR1	2.2%	51%	278 bp	0.0
SRR2627019.1m.10xR2	1.9%	52%	279 bp	0.0
SRR2627019.1m.10x_trimmed_R1	2.0%	51%	263 bp	0.0
SRR2627019.1m.10x_trimmed_R2	1.8%	51%	220 bp	0.0
SRR2627019.1m.20xR1	4.6%	51%	278 bp	0.0
SRR2627019.1m.20xR2	3.6%	52%	278 bp	0.0
SRR2627019.1m.20x_trimmed_R1	4.3%	51%	263 bp	0.0
SRR2627019.1m.20x_trimmed_R2	3.5%	51%	220 bp	0.0
SRR2627019.1m.40xR1	6.5%	50%	268 bp	0.1
SRR2627019.1m.40xR2	5.4%	51%	269 bp	0.1
SRR2627019.1m.40x_trimmed_R1	6.1%	50%	254 bp	0.1
SRR2627019.1m.40x_trimmed_R2	5.4%	50%	215 bp	0.1
SRR2627019.1m.80xR1	16.9%	51%	278 bp	0.1
SRR2627019.1m.80xR2	14.4%	52%	278 bp	0.1
SRR2627019.1m.80x_trimmed_R1	15.1%	51%	263 bp	0.1
SRR2627019.1m.80x_trimmed_R2	13.7%	51%	220 bp	0.1
SRR2986339_subsampled_1_trimmed_R1	0.5%	50%	97 bp	0.4
SRR2986339_subsampled_2_trimmed_R2	0.3%	50%	90 bp	0.4
tardigrade_SRR2986339_subsampled_1	0.6%	51%	101 bp	0.4
tardigrade_SRR2986339_subsampled_2	0.5%	51%	101 bp	0.4

FastQC: Sequence Counts

Export Plot

Adapter content

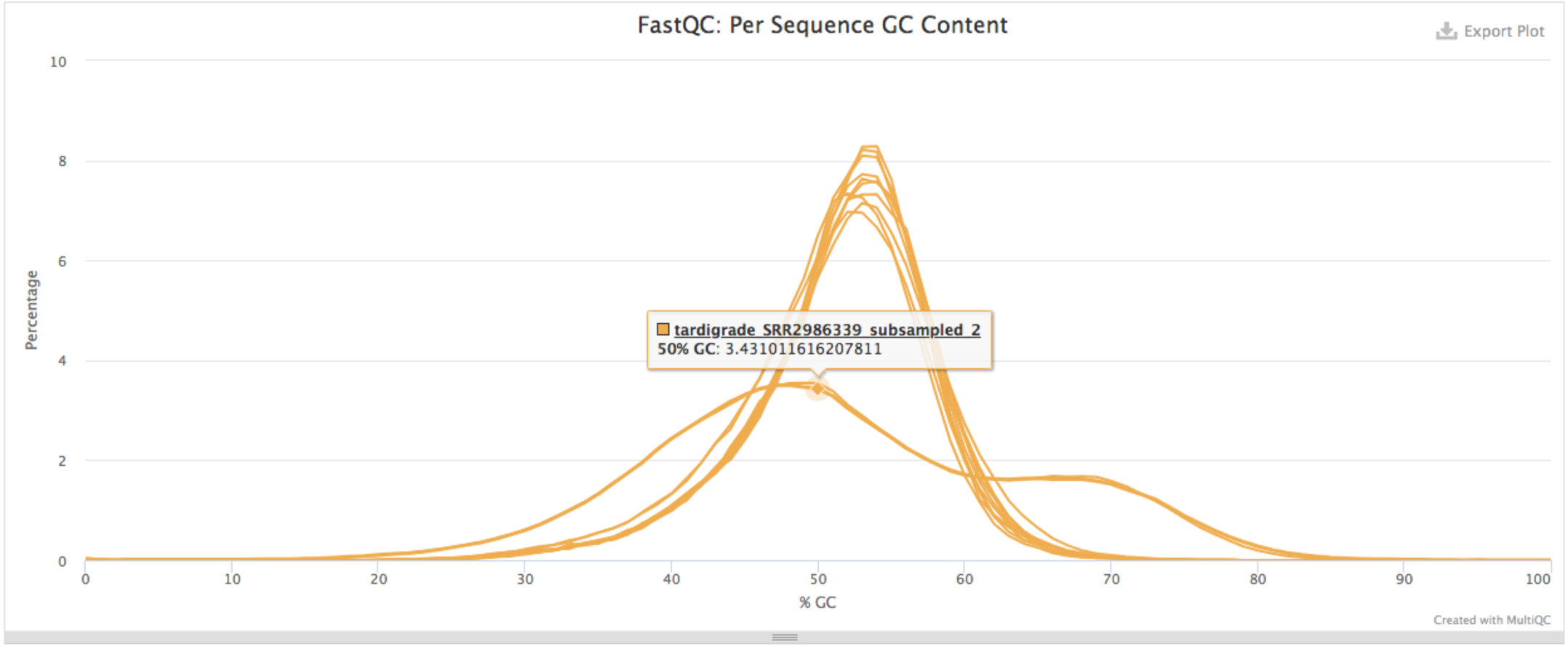
- Adapter content low
- removal by Illumina pipeline worked already well



00_inputQC

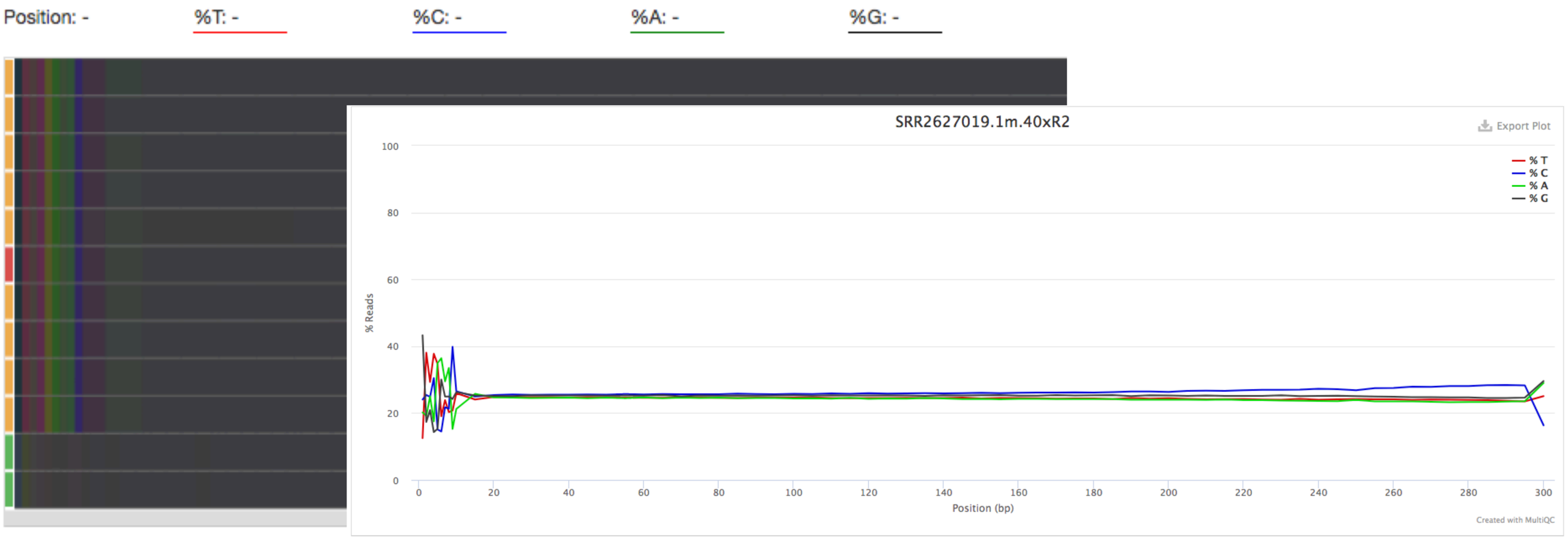
GC distribution

- Tardigrade bimodal GC content looks suspicious…
- E. coli looks ok



Base Bias

- Some samples show bias to certain base at the very beginning
- virtually always the case, due to not-really random priming
- usually not a problem + cutting biased bases wouldn't help



Assembly

01_assembly

Assemble short reads

```
#!/bin/bash
#SBATCH --job-name=spades
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=5
#SBATCH --time=6:00:00
#SBATCH --mem=5000
#SBATCH --mail-type=end
#SBATCH --reservation=bioscience_ml

# SPAdes assembler
# https://www.liebertpub.com/doi/abs/10.1089/cmb.2012.0021

module load spades

set -vex

# run assembler on trimmed 40x ecoli data
spades.py -m30 -t5 -o ./ecoli_assembly-SRR2627019_40x \
-1 ../00_inputQC/output/SRR2627019.1m.40x_trimmed_R1.fq.gz \
-2 ../00_inputQC/output/SRR2627019.1m.40x_trimmed_R2.fq.gz \
> spades_ecoli_40x.log
```

The result

```
$ ls -laht ecoli_assembly-SRR2627019_40x
total 20M
-rw-r--r--  1 lehmanr g-lehmanr 200K Jul 16 17:20 spades.log
-rw-r--r--  1 lehmanr g-lehmanr  532 Jul 16 17:20 warnings.log
drwxr-xr-x  2 lehmanr g-lehmanr 4.0K Jul 16 17:20 misc
drwxr-xr-x 11 lehmanr g-lehmanr 4.0K Jul 16 17:20 .
drwxr-xr-x  2 lehmanr g-lehmanr 4.0K Jul 16 17:20 tmp
-rw-r--r--  1 lehmanr g-lehmanr 216K Jul 16 17:20 contigs.paths
-rw-r--r--  1 lehmanr g-lehmanr 6.1M Jul 16 17:20 assembly_graph.fastg
-rw-r--r--  1 lehmanr g-lehmanr 3.1M Jul 16 17:20 assembly_graph_with_scaffolds.gfa
-rw-r--r--  1 lehmanr g-lehmanr 216K Jul 16 17:20 scaffolds.paths
-rw-r--r--  1 lehmanr g-lehmanr 3.0M Jul 16 17:20 scaffolds.fasta
-rw-r--r--  1 lehmanr g-lehmanr 3.0M Jul 16 17:20 contigs.fasta
-rw-r--r--  1 lehmanr g-lehmanr 3.0M Jul 16 17:20 before_rr.fasta
drwxr-xr-x  4 lehmanr g-lehmanr 4.0K Jul 16 17:20 K127
drwxr-xr-x  3 lehmanr g-lehmanr 4.0K Jul 16 17:19 K99
drwxr-xr-x  3 lehmanr g-lehmanr 4.0K Jul 16 17:19 K77
drwxr-xr-x  3 lehmanr g-lehmanr 4.0K Jul 16 17:19 K55
drwxr-xr-x  3 lehmanr g-lehmanr 4.0K Jul 16 17:19 K33
drwxr-xr-x  3 lehmanr g-lehmanr 4.0K Jul 16 17:18 K21
-rw-r--r--  1 lehmanr g-lehmanr  137 Jul 16 17:18 dataset.info
drwxr-xr-x  3 lehmanr g-lehmanr 4.0K Jul 16 17:18 corrected
-rw-r--r--  1 lehmanr g-lehmanr 1.9K Jul 16 17:18 params.txt
-rw-r--r--  1 lehmanr g-lehmanr  324 Jul 16 17:17 input_dataset.yaml
drwxr-xr-x  4 lehmanr g-lehmanr 4.0K Jul 16 17:17 ..
```

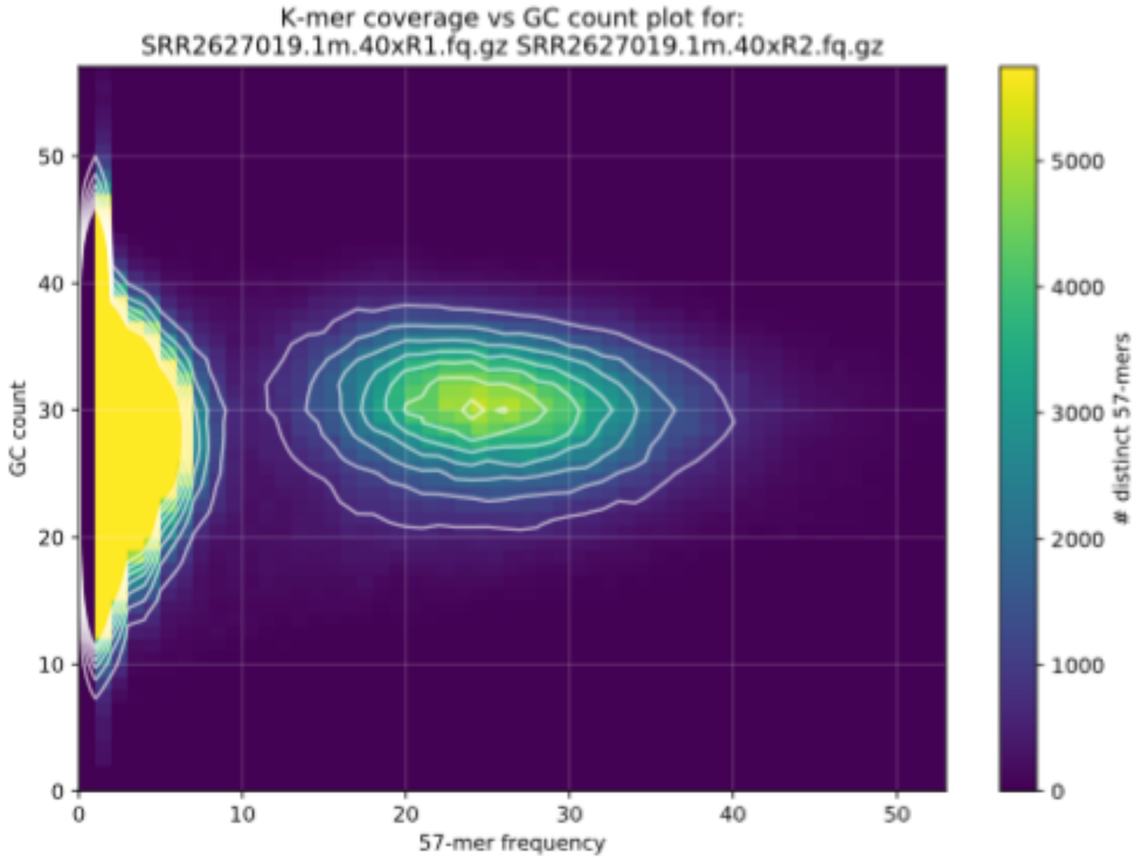
- have a look at spades.log
  did SPAdes terminate
  successfullly?

- where are the candidate
  assemblies for varying k?

Good for reproducibility:

- params.txt lists parameters
- spades.log gives detailed idea of
  what happened

Now edit runSpades.sh and assemble the other datasets!

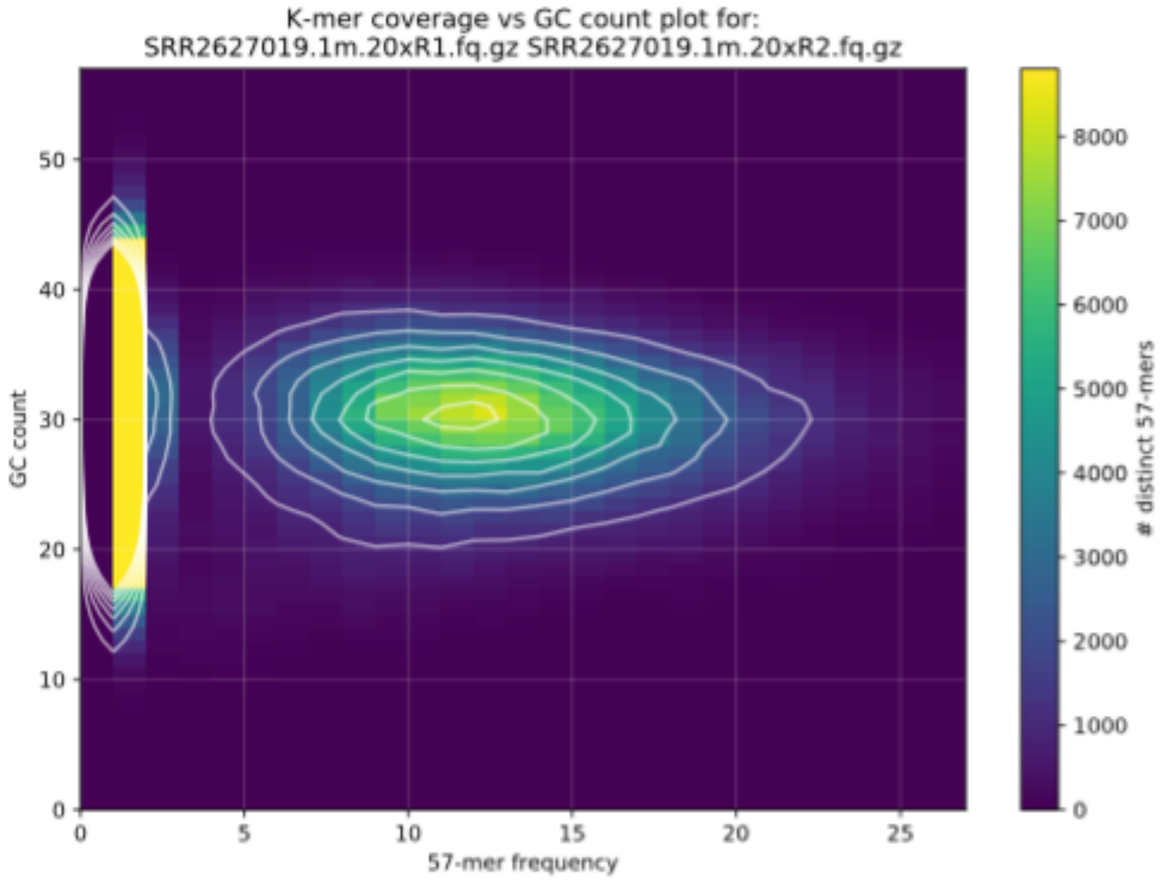


Read the Log Files!

01_assembly

40x

```
======= SPAdes pipeline finished WITH WARNINGS!

=== Error correction and assembling warnings:
 * 0:00:09.658   112M / 3G    WARN    General                 (kmer_coverage_model.cpp    : 218)   Too many erroneous kmers, the estimates might be unreliable
 * 0:00:12.967    96M / 3G    WARN    General                 (simplification.cpp         : 578)   The determined erroneous connection coverage threshold may be determined
======= Warnings saved to /scratch/dragon/intel/lehmanr/bscratch/lehmanr/hts_workshop2019/01_assembly/ecoli_assembly-SRR2627019_40x/warnings.log
```

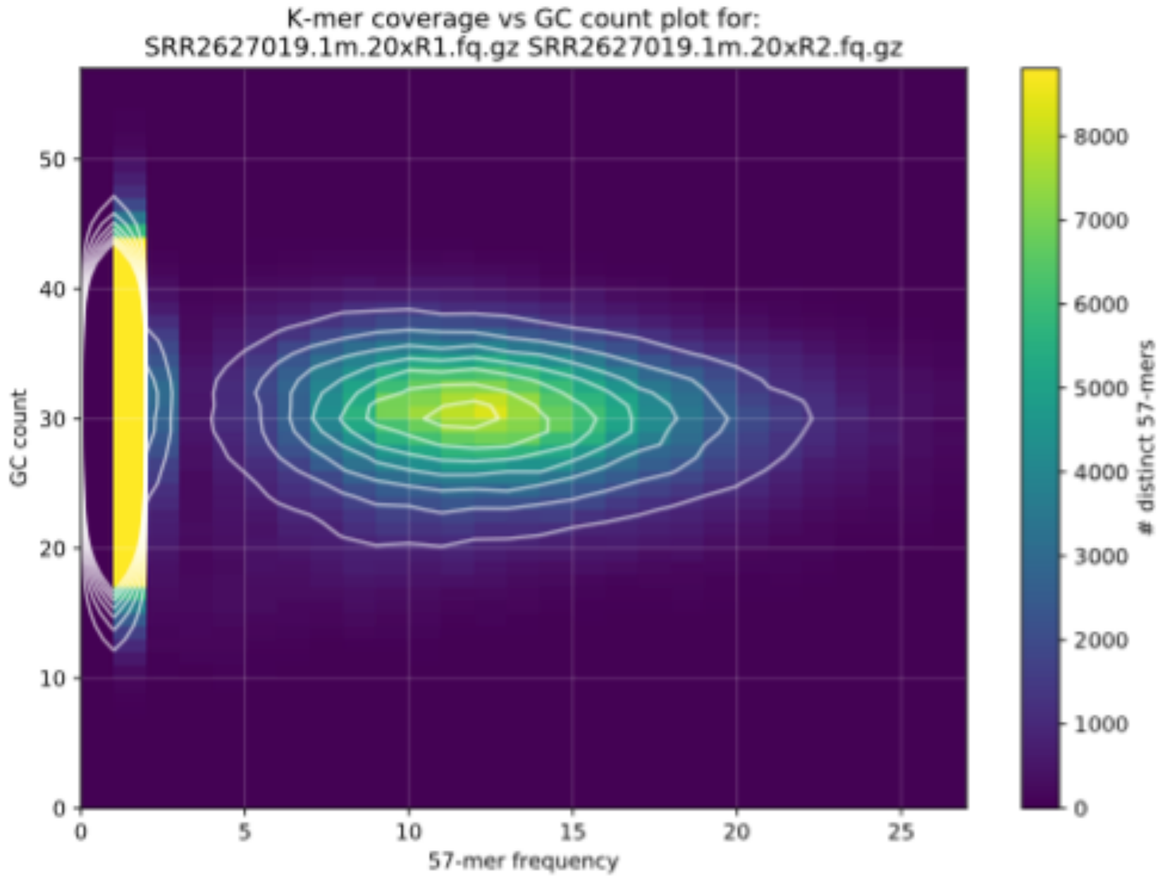


Read the Log Files!

01_assembly

20x

```
======= SPAdes pipeline finished WITH WARNINGS!

=== Error correction and assembling warnings:
 * 0:00:12.785    44M / 3G    WARN    General                 (kmer_coverage_model.cpp    : 366)   Failed to determine erroneous kmer threshold. Threshold set to: 0
======= Warnings saved to /scratch/dragon/intel/lehmanr/bscratch/lehmanr/hts_workshop2019/01_assembly/ecoli_assembly-SRR2627019_20x/warnings.log
```



01_assembly

Read the Log Files!

no error in 80x assembly process

```
======= SPAdes pipeline finished.

SPAdes log can be found here: /scratch/dragon/intel/lehmanr/bscratch/lehmanr/hts_workshop2019/01_assembly/ecoli_assembly-SRR2627019_80x/spades.log
```

How many pieces were assembled?


```
$ grep -c '>' K*/final_contigs.fasta
K127/final_contigs.fasta:2668
K21/final_contigs.fasta:6306
K33/final_contigs.fasta:4915
K55/final_contigs.fasta:3494
K77/final_contigs.fasta:3223
K99/final_contigs.fasta:2972
```

SPAdes chose the K=127 assembly!

```
$ grep -c '>' contigs.fasta
2668
```

```
$ grep -c '>' scaffolds.fasta
2666
```

And scaffolding closed 2 gaps

But isn't 2666 scaffolds on 1 Mb a bit much?

What about the other datasets?

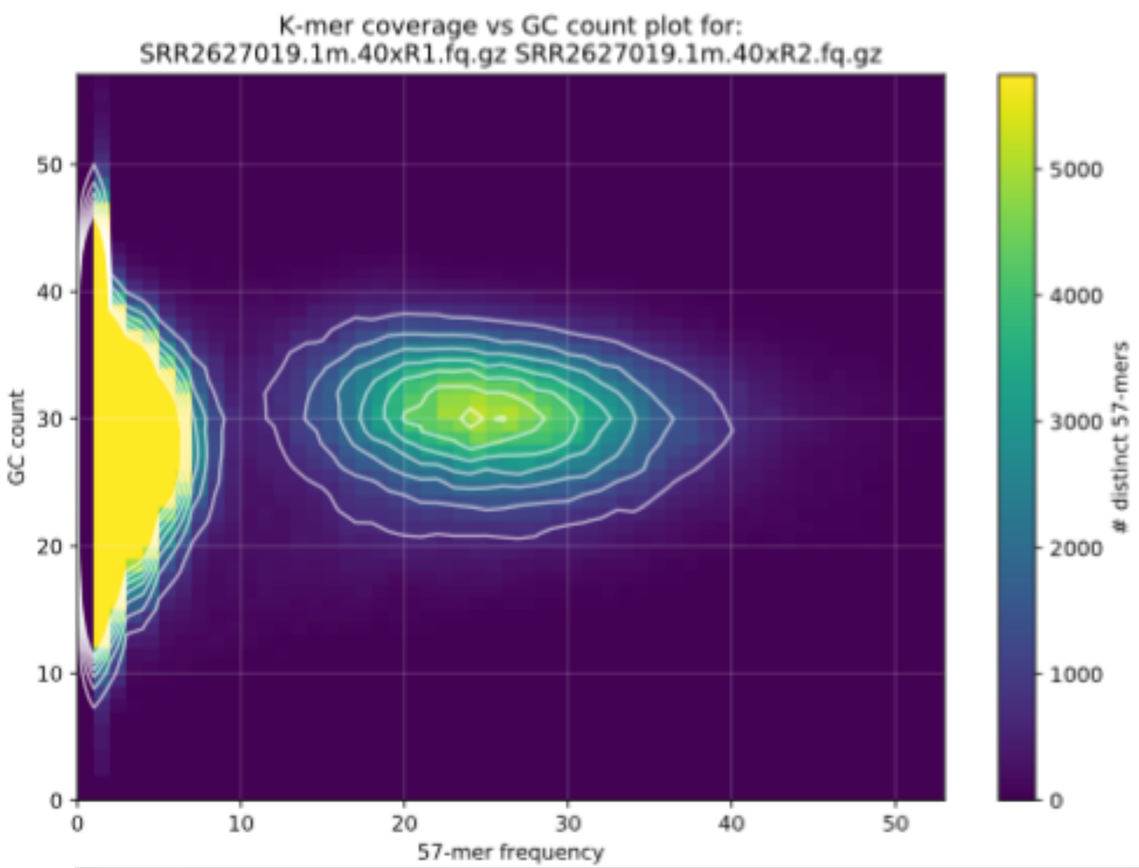

```
$ grep -c '>' K*/final_contigs.fasta
K127/final_contigs.fasta:23
K21/final_contigs.fasta:482
K33/final_contigs.fasta:278
K55/final_contigs.fasta:88
K77/final_contigs.fasta:79
K99/final_contigs.fasta:41
```

80x - only 23 contigs!
Due to higher coverage?

```
$ grep -c '>' ecoli_assembly-SRR2627019_*/contigs.fasta
ecoli_assembly-SRR2627019_10x/contigs.fasta:109
ecoli_assembly-SRR2627019_20x/contigs.fasta:30
ecoli_assembly-SRR2627019_40x/contigs.fasta:2668
ecoli_assembly-SRR2627019_80x/contigs.fasta:23
```

- Higher coverage does yield fewer contigs
- BUT: only 40x has much more contigs than all other?

Contamination assembled



The extra bits are likely Yersinia sequence that we detected earlier

```
[lehmanr@dbn503-35-r output]$ awk '$1 > 1' ecoli_kraken2_report.txt
 99.67  94735   298     R       1       root
 99.35  94436   2       R1      131567    cellular organisms
 99.30  94387   203     D       2           Bacteria
 99.08  94175   257     P       1224          Proteobacteria
 98.80  93909   69      C       1236            Gammaproteobacteria
 98.72  93831   917     O       91347             Enterobacterales
 74.22  70550   39261   F       543                 Enterobacteriaceae
 31.88  30298   1603    G       561                   Escherichia
 29.75  28282   27968   S       562                     Escherichia coli
 23.52  22356   15      F       1903411             Yersiniaceae
 23.50  22337   205     G       629                   Yersinia
 23.28  22124   16426   G1      1649845                 Yersinia pseudotuberculosis complex
  5.76  5475    5410    S       632                       Yersinia pestis
```

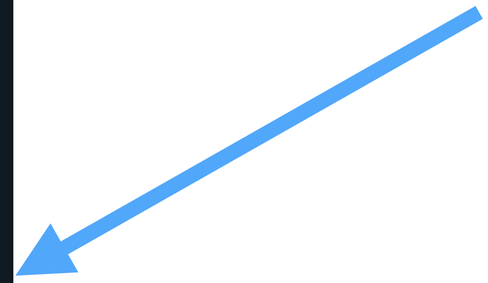
Are all assembled contigs trustworthy?

01_assembly

```
$ grep '>' scaffolds.fasta
>NODE_1_length_256701_cov_21.974647
>NODE_2_length_210887_cov_21.111340
>NODE_3_length_173798_cov_20.763576
>NODE_4_length_78753_cov_20.050085
>NODE_5_length_69295_cov_20.220275
>NODE_6_length_64076_cov_21.067022
>NODE_7_length_36688_cov_21.507262
>NODE_8_length_32471_cov_19.661792
>NODE_9_length_15646_cov_26.884335
>NODE_10_length_15264_cov_20.240602
>NODE_11_length_10776_cov_17.752371
>NODE_12_length_8831_cov_18.608915
>NODE_13_length_6805_cov_17.641360
>NODE_14_length_4267_cov_21.709903
>NODE_15_length_1345_cov_48.496716
>NODE_16_length_1255_cov_71.635638
>NODE_17_length_1195_cov_68.991573
>NODE_18_length_550_cov_6.846336
>NODE_19_length_258_cov_24.335878
>NODE_20_length_256_cov_41.186047
>NODE_21_length_192_cov_40.476923
>NODE_22_length_136_cov_49.444444
>NODE_23_length_128_cov_245.000000
```

80x Assembly

- a first idea how the assembly looks like
- e.g. noise contigs like Node 23 (short and very high k-mer coverage compared to rest - repetitive / low complexity region?)



Lets compare the candidate assemblies from varying K

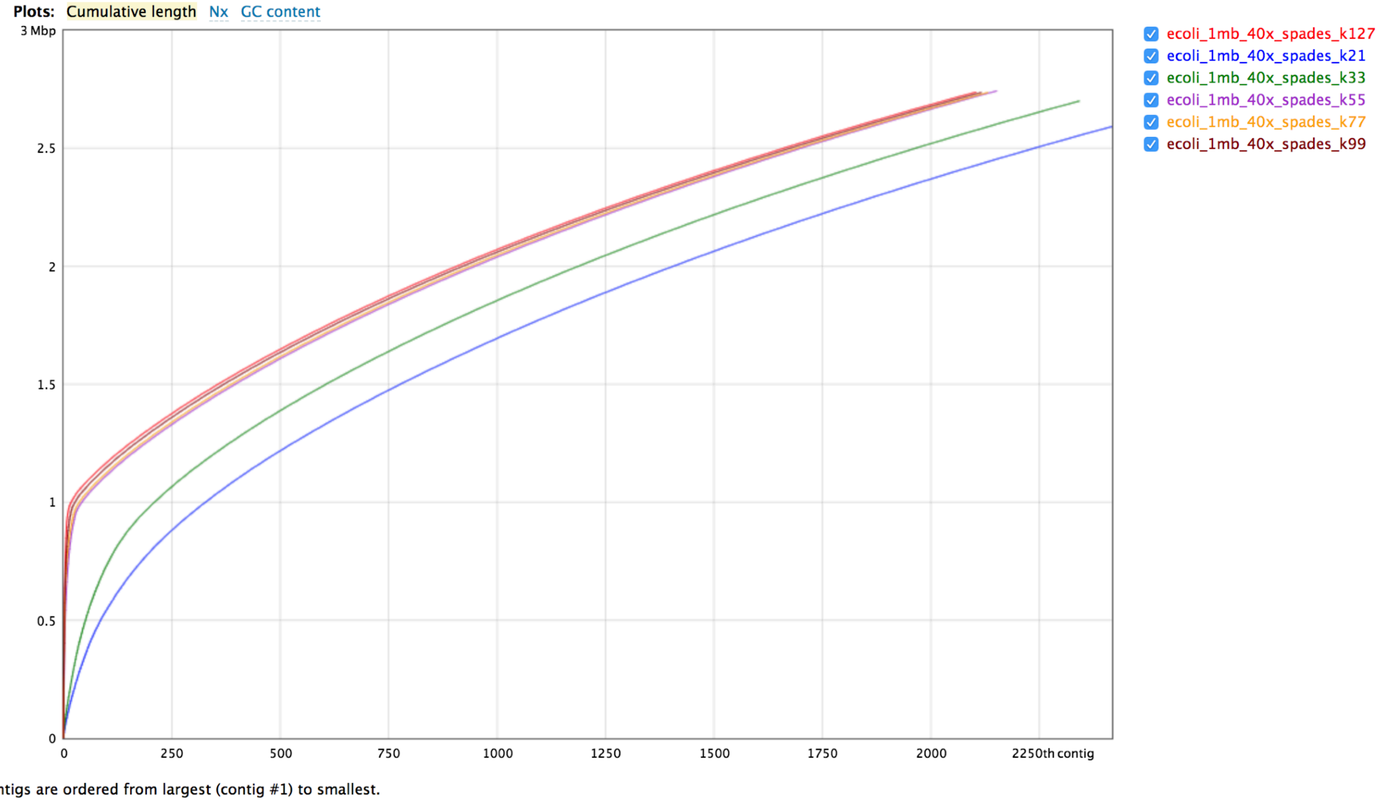
Which K is best?

01_assembly

$ nano 2-runQuast_compK.sh
$ sbatch 2-runQuast_compK.sh

And download the report to the local machine:

$ scp -r "<user>@ilogin.ibex.kaust.edu.sa:<path to output>/quast_results" .
$ open quast_results/latest/report.html



K=127 yields mostmost continuous assembly

- here: contiguity = more completeness -> spades chooses right candidate
- final assembly has some fewer contigs due to scaffolding

Which K is best?

01_assembly

Worst Median Best ☑ Show heatmap

Statistics without reference	ecoli_1mb_40x_spades_k127	ecoli_1mb_40x_spades_k21	ecoli_1mb_40x_spades_k33	ecoli_1mb_40x_spades_k55	ecoli_1mb_40x_spades_k77	ecoli_1mb_40x_spades_k99
# contigs	2105	2418	2343	2153	2132	2117
# contigs (>= 0 bp)	2668	6306	4915	3494	3223	2972
# contigs (>= 1000 bp)	469	660	625	500	494	480
# contigs (>= 5000 bp)	14	52	67	29	25	20
# contigs (>= 10000 bp)	11	5	21	21	18	16
# contigs (>= 25000 bp)	8	0	0	11	8	9
# contigs (>= 50000 bp)	6	0	0	5	4	4
Largest contig	252 996	12 664	20 787	159 849	196 920	204 292
Total length	2 738 532	2 592 217	2 700 462	2 743 708	2 736 906	2 736 802
Total length (>= 0 bp)	2 998 608	3 432 198	3 418 406	3 284 335	3 198 317	3 113 983
Total length (>= 1000 bp)	1 617 788	1 387 918	1 519 521	1 608 478	1 611 886	1 615 215
Total length (>= 5000 bp)	985 580	362 198	590 823	956 258	960 080	976 586
Total length (>= 10000 bp)	960 478	58 450	270 770	893 876	903 326	946 795
Total length (>= 25000 bp)	918 727	0	0	740 990	750 088	841 263
Total length (>= 50000 bp)	849 636	0	0	543 686	606 934	653 923
N50	1241	1065	1137	1214	1226	1241
N75	749	711	731	746	749	749
L50	245	572	467	285	274	257
L75	978	1327	1222	1025	1007	990
GC (%)	49.33	49.3	49.31	49.3	49.32	49.34
Mismatches						
# N's	0	0	0	0	0	0
# N's per 100 kbp	0	0	0	0	0	0

But which dataset yields best assembly?

01_assembly

Lets make a proper comparison
of our assembly candidates with Quast

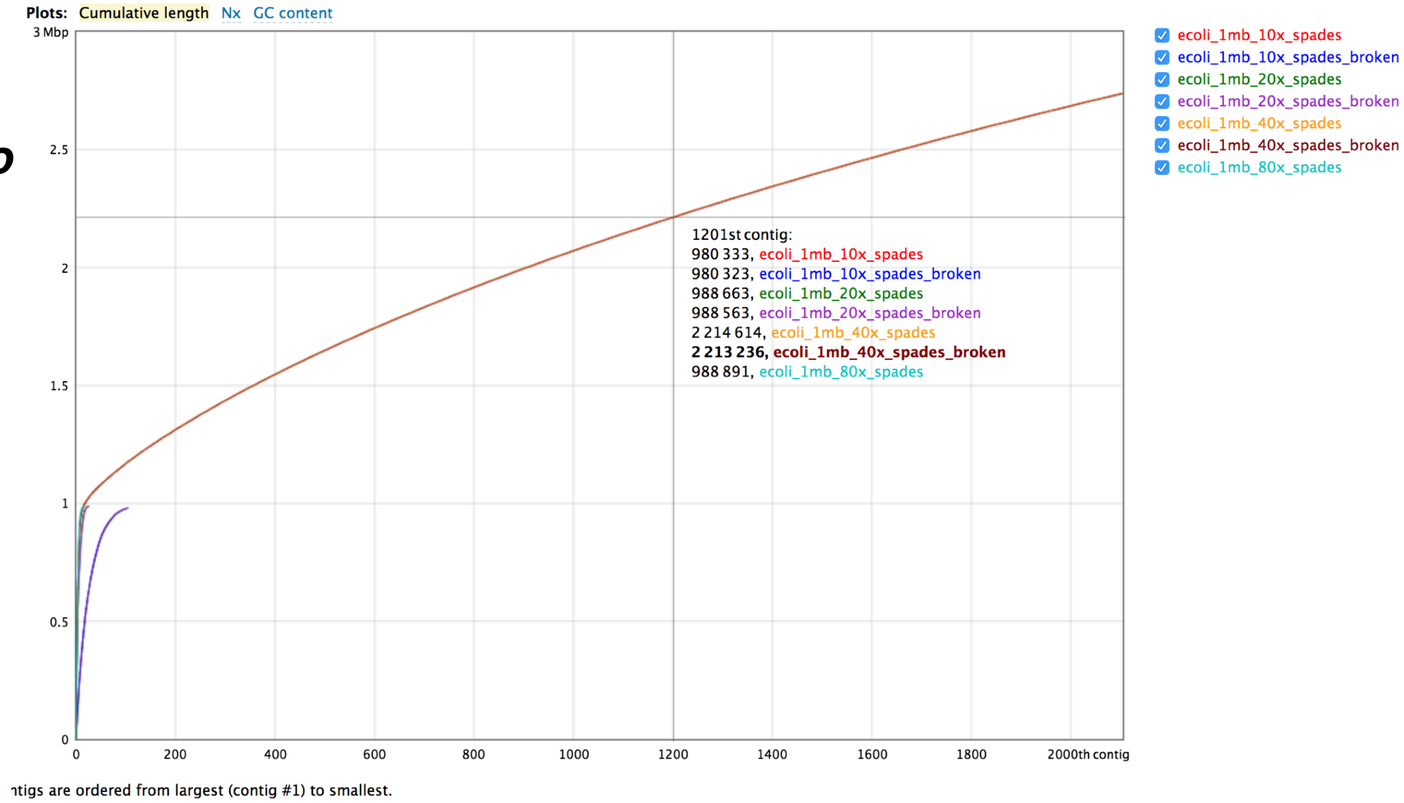
Read the script, it does something new:

$ nano 3-runQuast_compCoverage.sh
$ sbatch 3-runQuast_compCoverage.sh

And download the report to the local machine:

$ scp -r "<user>@ilogin.ibex.kaust.edu.sa:<path to output>/quast_results" .
$ open quast_results/latest/report.html

80x assembly is most continuous!
BUT: we only look at how
long fragments are…



Assembly Quality

How to compare to a genome reference?

02_assemblyQC

Lets compare our assemblies from different K values with a reference.

$ nano 1-runQuastDifferentKmer.sh
$ sbatch 1-runQuastDifferentKmer.sh

And download the report to the local machine:

$ scp -r "<user>@ilogin.ibex.kaust.edu.sa:<path to output>/quast_results" .
$ open quast_results/latest/report.html

SPAdes candidate Assemblies

02_assemblyQC

Worst Median Best — Show heatmap

Genome statistics	K127_final_contigs	K21_final_contigs	K33_final_contigs	K55_final_contigs	K77_final_contigs	K99_final_contigs
Genome fraction (%)	21.276	20.016	20.766	21.156	21.183	21.246
Duplication ratio	1.005	1.005	1.005	1.003	1.004	1.006
# genes	917 + 9 part	634 + 252 part	751 + 143 part	893 + 12 part	900 + 11 part	914 + 10 part
Largest alignment	252 996	12 664	20 787	159 849	196 920	204 292
Total aligned length	988 335	931 785	967 268	982 590	984 089	987 615
NG50	636	582	617	640	637	635
NGA50	-	-	-	-	-	-
LG50	1365	1915	1661	1406	1395	1380
Misassemblies						
# misassemblies	1	0	1	1	1	1
# relocations	1	0	1	1	1	1
# translocations	0	0	0	0	0	0
# inversions	0	0	0	0	0	0
# misassembled contigs	1	0	1	1	1	1
Misassembled contigs length	69 295	0	2749	11 138	11 182	27 947
# local misassemblies	1	1	1	3	3	2
# scaffold gap size misassemblies	0	0	0	0	0	0
# unaligned mis. contigs	3	1	1	1	1	3
Unaligned						
# fully unaligned contigs	2075	2048	2100	2103	2085	2076
Fully unaligned length	1 742 532	1 656 110	1 728 885	1 756 303	1 746 124	1 741 187
# partially unaligned contigs	5	3	3	3	5	5
Partially unaligned length	3962	2498	2510	2532	3962	3962
Mismatches						
# mismatches	38	32	36	45	49	55
# indels	1	0	0	0	0	1
Indels length	1	0	0	0	0	1
# mismatches per 100 kbp	3.85	3.44	3.73	4.58	4.98	5.58
# indels per 100 kbp	0.1	0	0	0	0	0.1
# indels (<= 5 bp)	1	0	0	0	0	1
# indels (> 5 bp)	0	0	0	0	0	0
# N's	0	0	0	0	0	0
# N's per 100 kbp	0	0	0	0	0	0
Statistics without reference						
# contigs	2105	2418	2343	2153	2132	2117
# contigs (>= 0 bp)	2668	6306	4915	3494	3223	2972
# contigs (>= 1000 bp)	469	660	625	500	494	480
# contigs (>= 5000 bp)	14	52	67	29	25	20
# contigs (>= 10000 bp)	11	5	21	21	18	16
# contigs (>= 25000 bp)	8	0	0	11	8	9
# contigs (>= 50000 bp)	6	0	0	5	4	4
Largest contig	252 996	12 664	20 787	159 849	196 920	204 292
Total length	2 738 532	2 592 217	2 700 462	2 743 708	2 736 906	2 736 802
Total length (>= 0 bp)	2 998 608	3 432 198	3 418 406	3 284 335	3 198 317	3 113 983
Total length (>= 1000 bp)	1 617 788	1 387 918	1 519 521	1 608 478	1 611 886	1 615 215
Total length (>= 5000 bp)	985 580	362 198	590 823	956 258	960 080	976 586
Total length (>= 10000 bp)	960 478	58 450	270 770	893 876	903 326	946 795
Total length (>= 25000 bp)	918 727	0	0	740 990	750 088	841 263
Total length (>= 50000 bp)	849 636	0	0	543 686	606 934	653 923
N50	1241	1065	1137	1214	1226	1241
N75	749	711	731	746	749	749
L50	245	572	467	285	274	257
L75	978	1327	1222	1025	1007	990
GC (%)	49.33	49.3	49.31	49.3	49.32	49.34
Similarity statistics						
# similar correct contigs	4	0	1	5	5	5
# similar misassembled blocks	0	0	0	0	0	0

- here: contiguity = more completeness -> spades chooses right candidate
- final assembly has some fewer contigs due to scaffolding

Add all candidate assemblies from different datasets

$ nano 2-runQuast_differentCoverage.sh
$ sbatch 2-runQuast_differentCoverage.sh

Candidate vs. reference

02_assemblyQC

Interesting key figures:
- N50
- total length
- genome fraction*
- # ref. genes covered*
- duplication ratio*

* if ref. Available

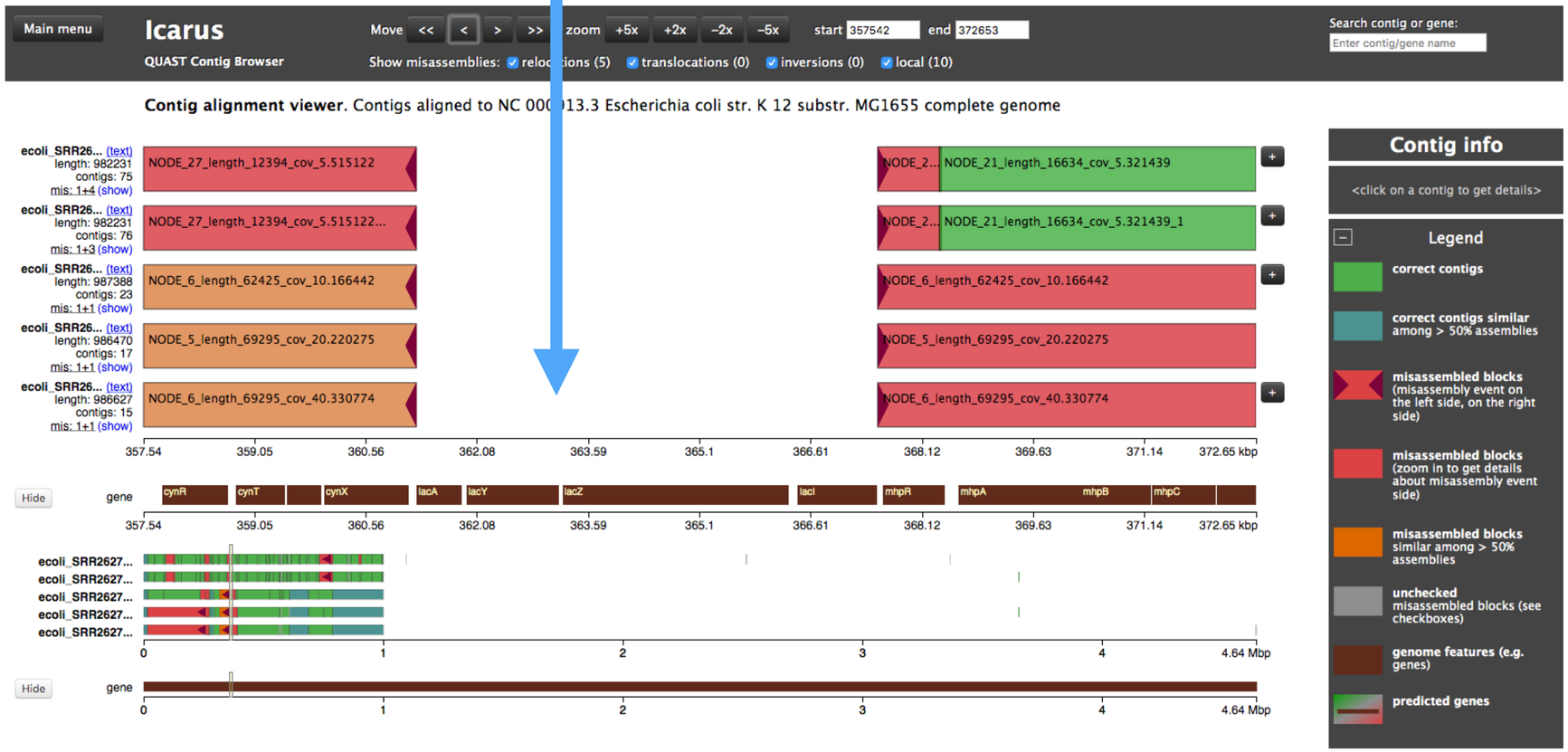
Genome statistics	ecoli_SRR2627019_scaffolds_10x	ecoli_SRR2627019_scaffolds_10...	ecoli_SRR2627019_scaffolds_20x	ecoli_SRR2627019_scaffolds_40x	ecoli_SRR2627019_scaffolds_80x
Genome fraction (%)	21.146	21.14	21.25	21.244	21.242
Duplication ratio	1.002	1.003	1.002	1.001	1.003
# genes	871 + 44 part	871 + 44 part	912 + 8 part	913 + 5 part	914 + 4 part
Largest alignment	60 888	60 888	210 281	256 701	256 701
Total aligned length	982 231	982 231	987 388	986 470	986 627
NGA50	-	-	-	-	-
Misassemblies					
# misassemblies	1	1	1	1	1
Misassembled contigs length	12 394	12 394	62 425	69 295	69 295
Mismatches					
# mismatches per 100 kbp	24.25	24.26	3.85	1.12	1.12
# indels per 100 kbp	0.1	0.1	0	0	0
# N's per 100 kbp	1.02	0	0	0	0
Statistics without reference					
# contigs	76	77	24	18	15
Largest contig	60 888	60 888	210 453	256 701	256 701
Total length	984 319	984 309	988 799	988 603	988 908
Total length (>= 1000 bp)	980 111	979 264	987 668	988 053	988 908
Total length (>= 10000 bp)	828 190	817 978	953 722	964 355	974 041
Total length (>= 50000 bp)	175 489	175 489	763 963	853 510	890 822

Extended report

How to compare our candidate assemblies?

02_assemblyQC

Lac negative E. coli strain sequenced

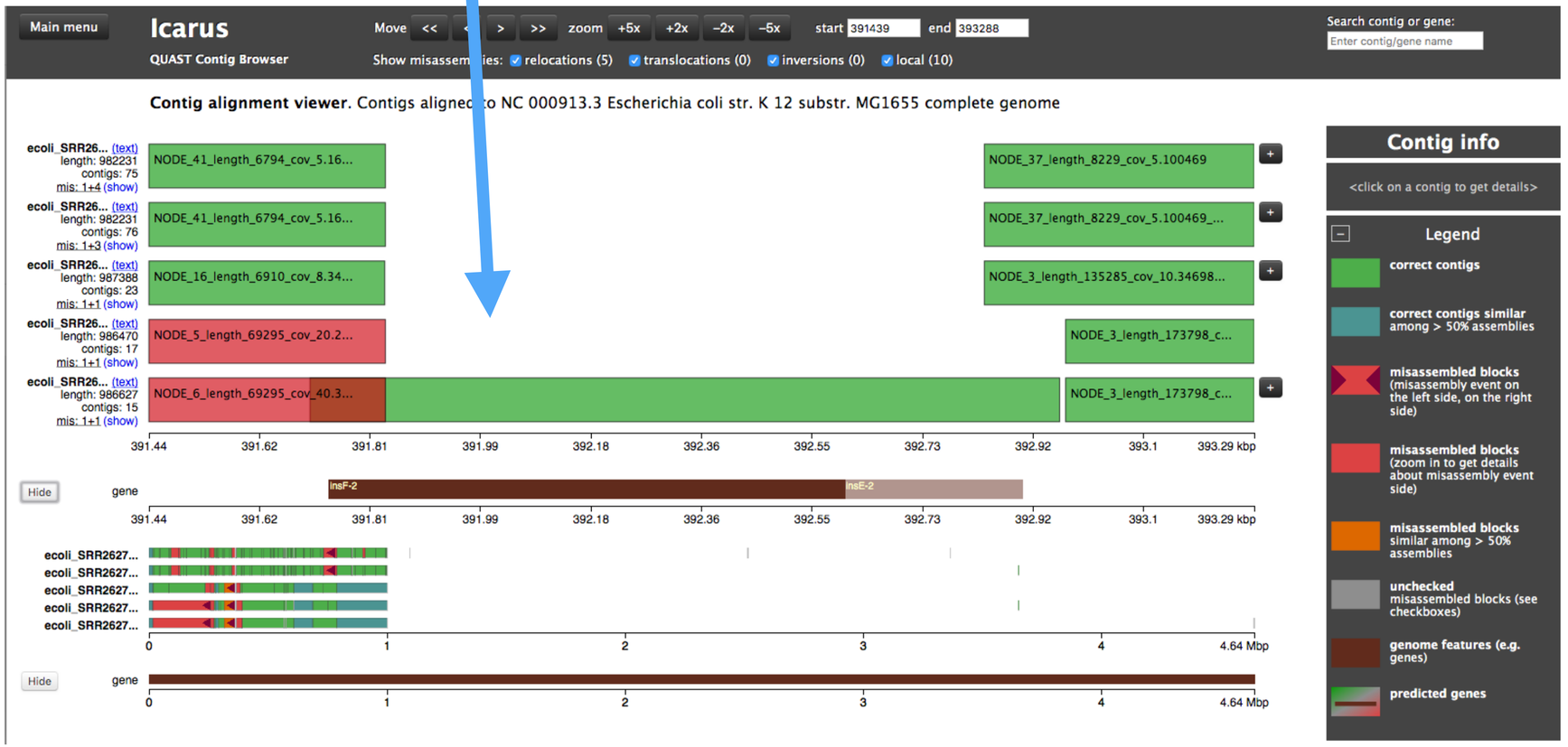


How to compare our candidate assemblies?

02_assemblyQC

Transposon genes

- repetitive sequences hard to assemble
- possibly polymorphic



Gene Annotation + QC

Gene Annotation

03_annotation

Let’s annotate our best assembly

Since we know we are assembling E. coli, we can help the genome annotation pipeline (PROKKA) by supplying the reference protein sequences so that it knows what to look for.

$ nano 1-runProkka.sh
$ sbatch 1-runProkka.sh

Count number of genes in gff:
awk '$3 =="gene"' <annotation.gff> | wc -l

Run annotation for all coverage levels…

BUSCO completeness

Use BUSCO to test how many “typical” bacterial genes we find in the reference annotation and in our new annotation (Keep in mind we only assembled the first 1 Mb)

$ nano 2-runBUSCO.sh
$ sbatch 2-runBUSCO.sh

Count number of genes in gff:
awk '$3 =="gene"' <annotation.gff> | wc -l

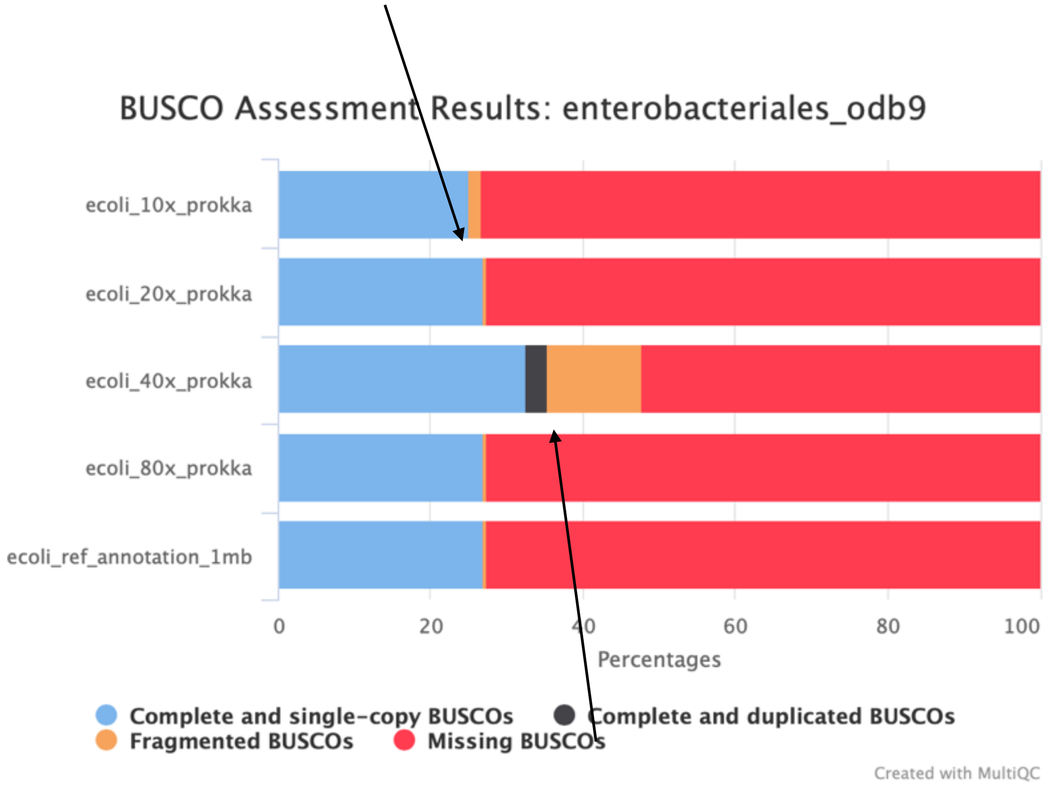
Run annotation for all coverage levels. And summarize again with MultiQC.

$ sbatch 3-runMultiQC.sh

Download report and check the result.

BUSCO completeness

Coverage increases completeness



Contamination creates “artificial” completeness

Hybrid Assembly

Hybrid Assembly

Let’s have a look at the Pacbio reads

FastQC is the tool for the job

$ nano 0-runFastQC.sh
$ sbatch 0-runFastQC.sh

How many reads do we have, how long are they, how are they different to short reads?

Hybrid Assembly

We can use again SPAdes

$ nano 1-runSpades.sh
$ sbatch 1-runSpades.sh

Check if SPAdes had any errors / warnings
Which K did it select?

Hybrid Assembly QC

Hybrid Assembly

05_hybridAssemblyQC

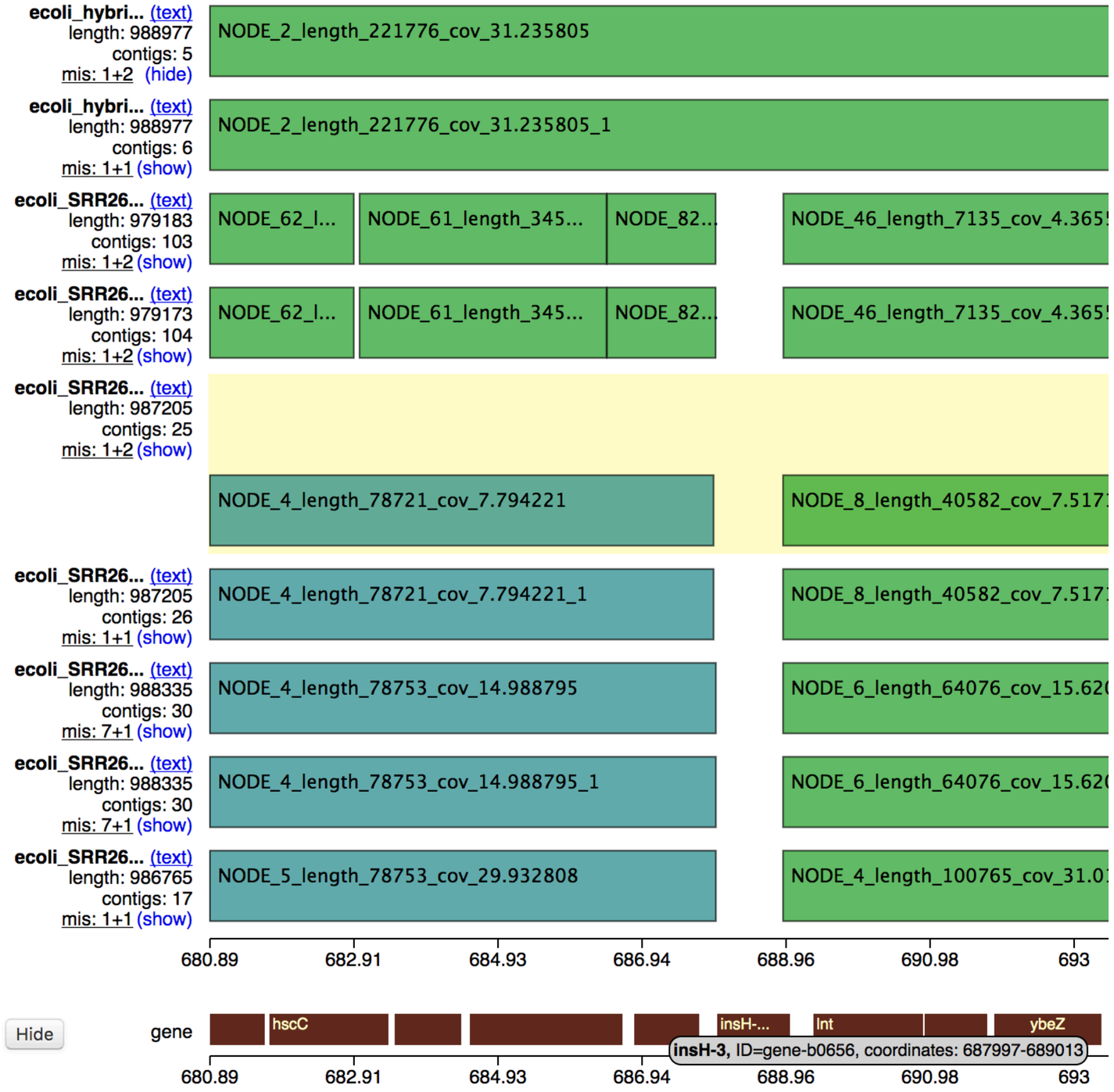
Let’s compare all candidate assemblies

$ nano 1-runQuast.sh
$ sbatch 1-runQuast.sh

Table 1: QUAST report on assembly and annotation

Assembly	10x	20x	40x	80x	80x+5x
# contigs	104	26	2103	18	6
Largest contig	39,006	210,408	252,996	252,996	368,407
GC (%)	51.44	51.4	49.33	51.39	51.41
N50	20,693	135,285	1,244	173,798	221,776
Misassembled contigs length	12,345	62,652	69,295	69,295	368,407
# genes	842+73 part	912+7 part	917+9 part	912+7 part	920+2 part

- long reads boost contiguity + completeness
- would be even more pronounced with lower coverage short read data



Hybrid Assembly

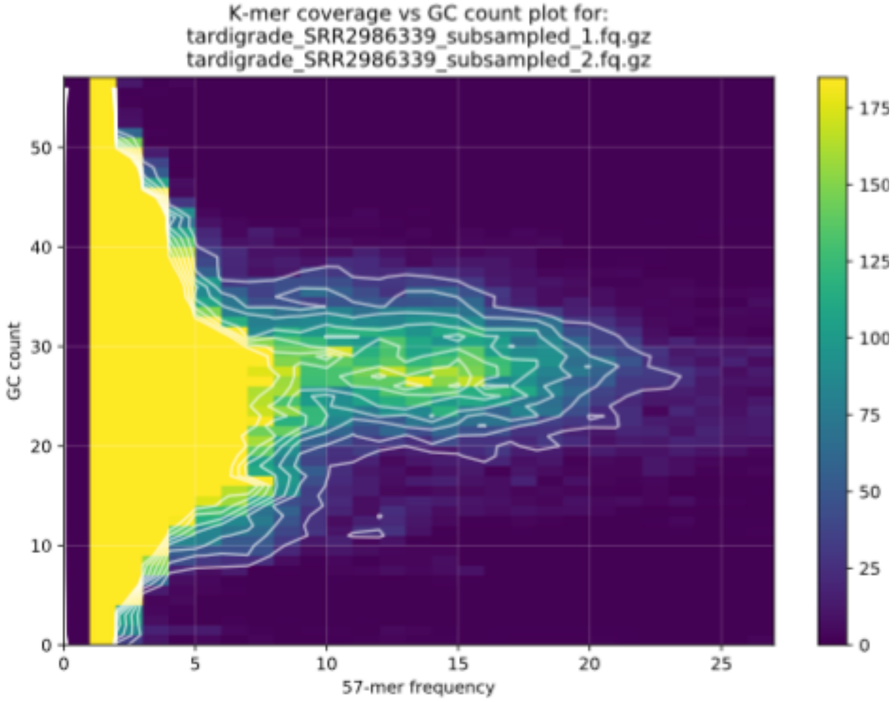
05_hybridAssemblyQC

insH-3 transposase gene only assembled in hybrid assembly

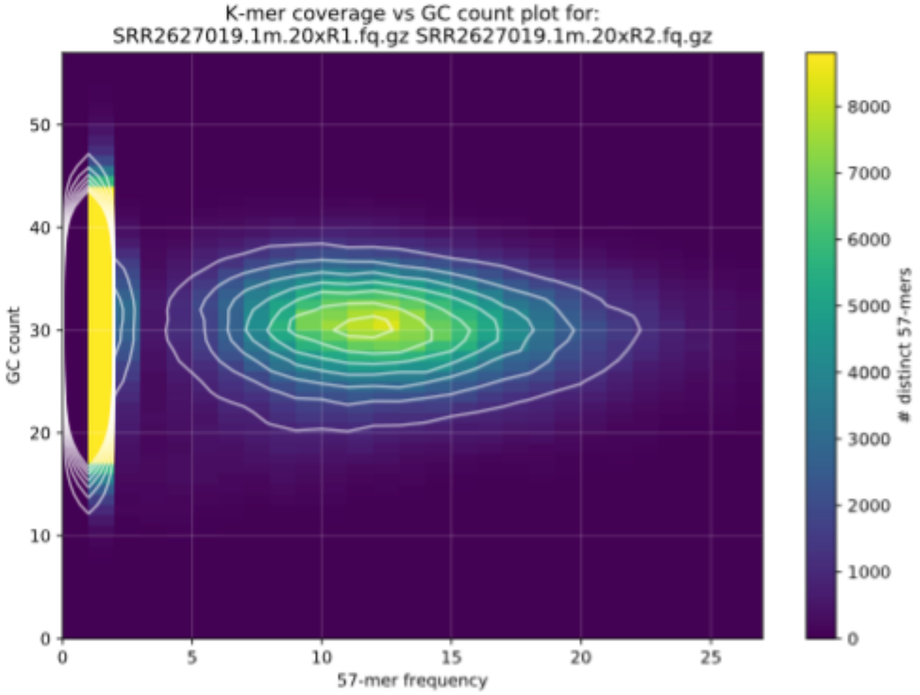
Tardigrade Input data QC

Contamination?

Tardigrade

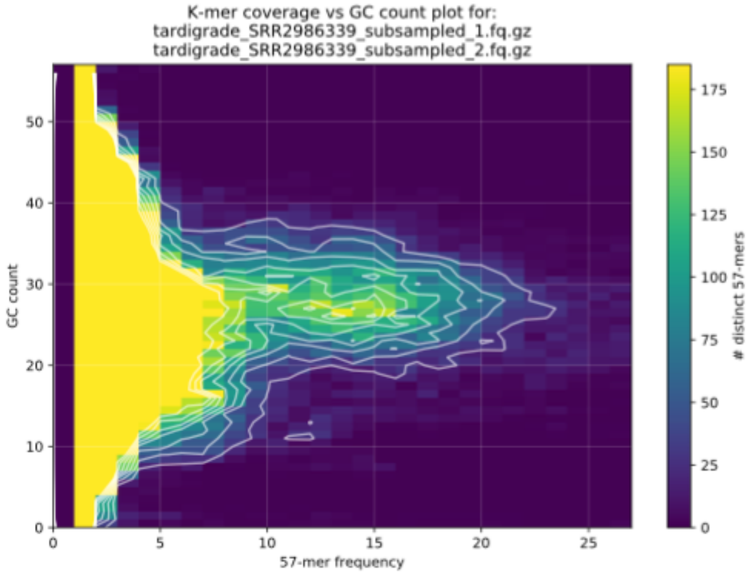


E. coli
20x



Tardigrade Input data QC

Contamination?



- 90 % unclassified - ok since not in database
- 6 % bacteria?
- 3.45 % human?
- no horizontal gene transfer…

```
[lehmanr@dbn503-33-r output]$ awk '$1 > 1' tardigrade_kraken2_report.txt
 90.19  394434  394434  U       0       unclassified
  9.81  42880   1066    R       1       root
  9.55  41742   209     R1      131567    cellular organisms
  6.01  26299   1182    D       2           Bacteria
  4.23  18485   1053    P       1224          Proteobacteria
  2.06  8989    686     C       28211           Alphaproteobacteria
  1.07  4675    426     O       356               Rhizobiales
  1.57  6844    189     C       28216           Betaproteobacteria
  1.46  6374    585     O       80840             Burkholderiales
  3.45  15085   0       D       2759        Eukaryota
  3.45  15085   0       D1      33154         Opisthokonta
  3.45  15085   0       K       33208           Metazoa
  3.45  15085   0       K1      6072              Eumetazoa
  3.45  15085   0       K2      33213               Bilateria
  3.45  15085   0       K3      33511                 Deuterostomia
  3.45  15085   0       P       7711                    Chordata
  3.45  15085   0       P1      89593                     Craniata
  3.45  15085   0       P2      7742                        Vertebrata
  3.45  15085   0       P3      7776                          Gnathostomata
  3.45  15085   0       P4      117570                          Teleostomi
  3.45  15085   0       P5      117571                            Euteleostomi
  3.45  15085   0       P6      8287                                Sarcopterygii
  3.45  15085   0       P7      1338369                               Dipnotetrapodomorpha
  3.45  15085   0       P8      32523                                   Tetrapoda
  3.45  15085   0       P9      32524                                     Amniota
  3.45  15085   0       C       40674                                       Mammalia
  3.45  15085   0       C1      32525                                         Theria
  3.45  15085   0       C2      9347                                            Eutheria
  3.45  15085   0       C3      1437010                                           Boreoeutheria
  3.45  15085   0       C4      314146                                              Euarchontoglires
  3.45  15085   0       O       9443                                                  Primates
  3.45  15085   0       O1      376913                                                  Haplorrhini
  3.45  15085   0       O2      314293                                                    Simiiformes
  3.45  15085   0       O3      9526                                                      Catarrhini
  3.45  15085   0       O4      314295                                                      Hominoidea
  3.45  15085   0       F       9604                                                          Hominidae
  3.45  15085   0       F1      207598                                                          Homininae
  3.45  15085   0       G       9605                                                              Homo
  3.45  15085   15085   S       9606                                                                Homo sapiens
```