Bioinformatics Final Project

Team 2: Dan Marten and Stanley Nicholson

December 11, 2021

Introduction

In this project, groups were provided with both Nanopore and Illumina data, and a suspected *Escherichia coli* genome in the organism was constructed and annotated through the use of various assemblies, taxonomized BLAST scripts, QUAST for filtered assembly comparisons, and both PROKKA and DFAST for annotation. With a complete genome assembled from this, we were also able to run a mock read-mapping experiment with our original reads (Nanopore and Illumina) against the assembled reference genome. Our best assembly was selected as that from Unicycler using Nanopore data, as it constructed a single contig of approximately 4.7 Mbp, approximately the same length of the the expected *Escherichia coli*. While other assemblies with Nanopore data produced a similarly sized contig, some additional rigor in Unicycler's methods lead us to choose the selected assembly. Further analyses can be seen in the remainder of the report. In the future, performing a hybrid Unicycler assembly with both Nanopore and Illumina data simultaneously could produce improved results. In summary, though, we were able to assemble and annotate *Escherichia coli* genome using the instructions outlined in the report.

Step 1: Visualize Data / Step 2: Filter

In this section, we filtered our Nanopore data using `NanoFilt` and our Illumina data with `fastp`. Both methods yield further `.fastq` files, optionally zipped with the `.gz` extension. We use `fastqc` to visualize the `.fastq` files and the data's quality.

To contextualize the read data and filtering decisions, it is important to inderstand the key discrepancies between Nanopore and Illumina reads. In summary, Nanopore reads are relatively long (average sequence length of 10,000 bp) and less accurate than Illumina reads. When the DNA strands are first being read by Nanopore's sequencer, the first 50 bases are of lower quality, and filtering these out is of utmost importance, as well as removing generally low quality reads. Illumina reads are generally shorter (50-300 bp) but of higher quality. However, for Illumina reads, it is generally the last of the reads, after approximately 250-275 bp, that are of lower quality. These considerations are important when moving forward.

Concerning the Illumina data, our major considerations were simply sequence quality and removing overall poor quality reads. As such, we implemented a filter with a quality cutoff of 30 and a

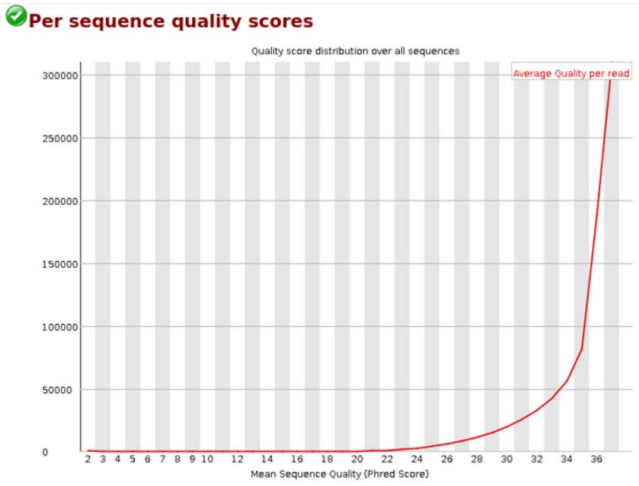

Figure 1: Raw Illumina Data

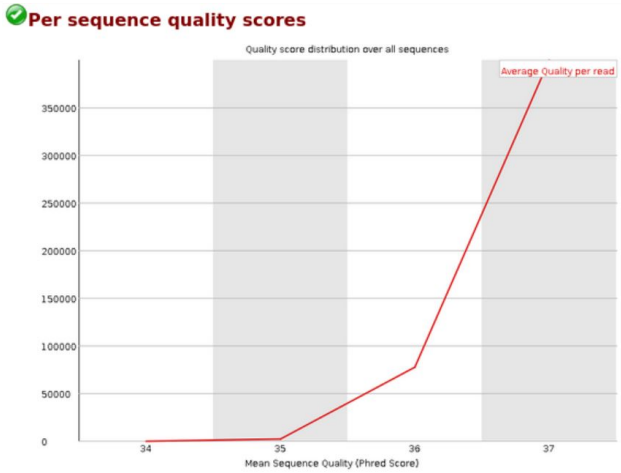

Figure 2: Filtered Illumina Data

Figure 3: Before and after filtering Illumina data

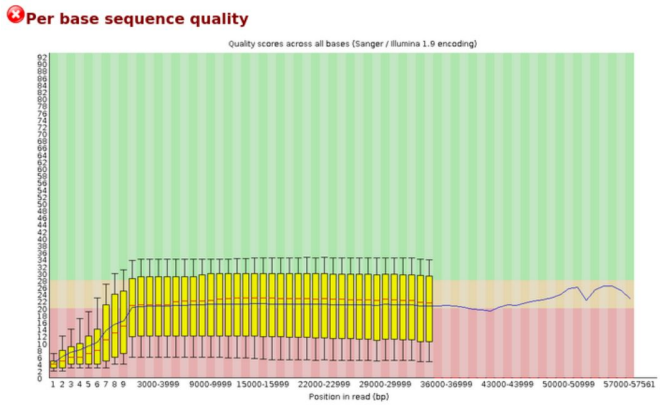

Figure 4: Raw Nanopore data

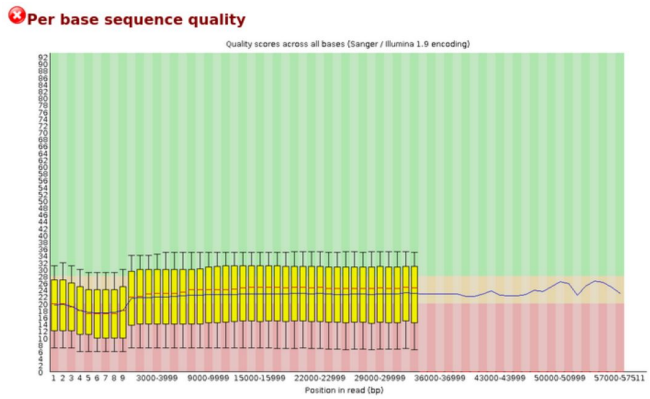

Figure 5: Filtered Nanopore data

Figure 6: Before and after filtering Nanopore data

minimum length of 100 bp. This code is seen in `Code 1`. This removed low quality reads, as seen in **Figure 1**. Interestingly enough, our minimum read quality from this was raised to 34 instead of 30. Though not displayed, our read count was 812,000 before filtering and 479,000 after filtering, to the thousands, for the forward reads.

For Nanopore, **Figure 6** shows clearly that the early low quality reads have been removed. This is due to filtering for a minimum PHRED quality of 10 and removing the first 50 bp. With only the lowest of quality filters removed, our number of total sequences decreased from 120,000 to 98,000, to the nearest thousand. Interestingly enough, our first effort included a minimum quality cutoff of 17. However, it cut our number of sequences into less than one tenth of what it previously was, despite a quality of 17 being less than the mean sequence quality.

1. `fastp -w 2 -i F21_illumina_R1.fastq.gz -I F21_illumina_R2.fastq.gz -o F21R1illumina.fastq.gz -O F21R2illumina.fastq.gz -M 30 -r -l 100`

2. `gunzip -c F21_nanopore_team2.fastq.gz | NanoFilt -q 10 --headcrop 50 | gzip > nanoq10test50head.fastq.gz`

Step 3: Assemblies

After filtering the reads, the Nanopore and Illumina data was aligned and assembled into contiguous sequences using pre-constructed and open-source assembler. Unicycler was used for both Nanopore and Illumina data, while Flye and Shasta were used for Nanopore data as well. SPAdes was then used for Illumina data as well. The code for running these assemblies is shown below, with discrepancies and interesting design decisions listed as relevant.

Unicycler

Illumina

```
/usr/local/bin/unicycler-1
↪ /home/biF21_07/projectSymLinkT02/Illumina/finalFiltered/F21R1illumina.fastq.gz
↪ -2
↪ /home/biF21_07/projectSymLinkT02/Illumina/finalFiltered/F21R2illumina.fastq.gz
↪ -o ./ --threads 4 --spades_path /opt/SPAdes-3.13.0-Linux/bin/spades.py
↪ --pilon_path /opt/pilon/pilon-1.24.jar
```

Nanopore

```
/usr/local/bin/unicycler -l
↪ /home/biF21_07/projectSymLinkT02/Nanopore/filtered/nano_q10_h50_FINAL_FILTER/

nanopore_q10_h50.fastq.gz -o ./ --threads 4
```

Flye

```
/opt/Flye-2.8.3/bin/flye --nano-corr
↪ /home/biF21_07/projectSymLinkT02/Nanopore/filtered/nano_q10_h50_FINAL_FILTER/
↪ nanopore_q10_h50.fastq.gz --genome-size 4.5m --out-dir . --threads 2
```

The assembler Flye was outputting files with 0 contigs when the filtered Nanopore data was given as an argument after `--nano-corr`. Instead, unfiltered data was provided with `--nano-raw`.

```
/opt/Flye-2.8.3/bin/flye--nano-raw
↪ /home/biF21_07/projectSymLinkT02/Nanopore/F21_nanopore_team2.fastq.gz
↪ --genome-size 4.5m --out-dir ./data/ --threads 1
```

Shasta

```
shasta --input nanoq10h50.fastq --assemblyDirectory ./data/ --command assemble
↪  --threads 1 --config Nanopore-Oct2021
```

SPAdes

```
/opt/SPAdes-3.15.2-Linux/bin/spades.py -t 2 --pe-1 1
↪  /home/biF21_07/projectSymLinkT02/Illumina/F21R1illumina.fastq.gz --pe-2 1
↪  /home/biF21_07/projectSymLinkT02/Illumina/F21R2illumina.fastq.gz -o
↪  /media/Data_1/F2021_Team_Assignment/F21_T02/SPAdesWork/SPAdesOutputs/
```

`filteredIlluminaPairedReads1128`

Note that we have changed the output directory to `/work/Assemblies/SPAdesOutputs`.

With the exception of Flye, all of the above assemblies ran as expected. These assemblies are further processed in the following section.

Step 4: Run & Parse BLAST

The above assemblies yielded `.fasta` files for the resultant contigs. In order to construct the genome of an expected *Escherichia coli* organism, a taxonomized BLAST search was performed on the resultant contigs, which returns, in layman's terms, which organisms they most closely resemble, as well as how strong that resemblance is. These results are then parsed and selected only for those contigs which we can be extremely confident could belong to our *Escherichia coli* of interest. However, as it may be a new strain of *Escherichia coli*, not aligning particularly well with previous classified strains of the specific species, we are only searching for outstanding similarity to any species existing in the genus *Escherichia*.

The above steps are performed using the pre-written scripts `runTaxonomizedBLAST.pl` and `parseTaxonomizedBLAST.pl`, as provided by Professor Pombert. Concerning some experimental design parameters, an e-value cutoff of 1e-5 was selected for the first script, and then a much more stringent e-value cutoff of 1e-50 was selected for the parsing and filtering step, for more confidence. Both scripts were included in our own written script, as seen below

```
#!/usr/bin/perl

$inputFasta = $ARGV[0];
$outputName = $ARGV[1];
# no DIE or £usage here, as this is ONLY intended for group/personal use

system("perl ~/projectSymLinkT02/scripts/runTaxonomizedBLAST.pl --threads 8 -p
↪  blastn -a megablast -d /media/Data_1/NCBI/REPGENOMES/ref_prok_rep_genomes
↪  --query $inputFasta --evalue 1e-5 --culling 5");
```

```
# system("perl ~/projectSymLinkT02/scripts/parseTaxonomizedBLAST.pl --blast
↪  *blastn.6 -f $inputFasta --name Escherichia \"Escherichia\" --evalue 1e-50
↪  --output $outputName -v on");
system("perl ~/projectSymLinkT02/scripts/parseTaxonomizedBLAST.pl -b *blastn.6 -f
↪  $inputFasta -n Escherichia \"Escherichia\" --evalue 1e-50 --output
↪  $outputName -v on");
```

Please note that the above script was only intended to simplify our own work flow, so documentation and design may be poor. However, the above script provides an output in the form of a `.fasta` file only containing the contigs which are overwhelmingly likely to belong to *Escherichia*.

Step 5: QUAST

We then input these final output `.fasta` files into QUAST so as to compare the different assemblies. In summary, all Nanopore-based assemblies were able to construct a genome at the approximate length expected, or 4.7 Mbp for each, in one contig. While all Illumina-based assemblies also had reasonable total length, at 4.5 Mbp each, the longest single contig was only 0.40 Mbp. This resulted in better N50 metrics for Nanopore assemblies. As such, we selected the Unicycler assembly of Nanopore to be our best assembly, due to Unicycler's own internal error processes. While the Illumina reads may be more accurate and have better read quality, as an assembly none of them proved to be as good as a Nanopore read-based assembly.

```
/opt/quast/quast.py -o analyses002/
↪  finalFilteredAssemblies/flyNanoUnfiltOut.fasta
↪  finalFilteredAssemblies/shastaNanoFiltOut.fasta
↪  finalFilteredAssemblies/spadesIlluminaFilteredOut.fasta
↪  finalFilteredAssemblies/uniIlluminaFilteredOut.fasta
↪  finalFilteredAssemblies/uniNanoFilteredOut.fasta
```

Step 6: PROKKA + DFast

We then performed genome annotation on these assemblies using PROKKA and DFAST, so as to identify potential protein coding regions. We selected the Flye assembly, the SPAdes assembly, and the Unicycler assembly using Nanopore reads, so as to compare the results.

PROKKA

Analyzing SPAdes assembled Illumina data:

```
prokka --outdir prokkaSpadesOut1206/ --prefix F21 --locustag F21 --compliant
↪  --species eColi -cpus 4
↪  ~/projectSymLinkT02/work/QUAST/finalFilteredAssemblies/

spadesIlluminaFilteredOut.fasta
```

Analyzing Unicycler assemebled Nanopore data:

Report

	flyNanoUnfiltOut	shastaNanoFiltOut	spadesIlluminaFilteredOut	uniIlluminaFilteredOut	uniNanoFilteredOut
# contigs (>= 0 bp)	1	1	98	99	1
# contigs (>= 1000 bp)	1	1	69	66	1
# contigs (>= 5000 bp)	1	1	49	48	1
# contigs (>= 10000 bp)	1	1	46	45	1
# contigs (>= 25000 bp)	1	1	42	43	1
# contigs (>= 50000 bp)	1	1	28	28	1
Total length (>= 0 bp)	4736672	4737880	4586811	4529886	4736752
Total length (>= 1000 bp)	4736672	4737880	4575640	4518126	4736752
Total length (>= 5000 bp)	4736672	4737880	4532949	4484256	4736752
Total length (>= 10000 bp)	4736672	4737880	4508712	4460760	4736752
Total length (>= 25000 bp)	4736672	4737880	4432119	4422893	4736752
Total length (>= 50000 bp)	4736672	4737880	3884494	3841907	4736752
# contigs	1	1	75	72	1
Largest contig	4736672	4737880	404504	404251	4736752
Total length	4736672	4737880	4579959	4522913	4736752
GC (%)	50.78	50.77	50.77	50.75	50.77
N50	4736672	4737880	137369	136989	4736752
N90	4736672	4737880	41423	41169	4736752
L50	1	1	10	10	1
L90	1	1	34	34	1
# N's per 100 kbp	0.00	0.00	0.00	0.00	0.00

All statistics are based on contigs of size >= 500 bp, unless otherwise noted
(e.g., "# contigs (>= 0 bp)" and "Total length (>= 0 bp)" include all contigs).

Figure 7: QUAST comparison of all the assemblies

```
prokka --outdir prokkaUniNanoOut1206/ --prefix F21 --locustag F21 --compliant
↪  --species eColi -cpus 4
↪  ~/projectSymLinkT02/work/QUAST/finalFilteredAssemblies/

uniNanoFilteredOut.fasta
```

Analyzing Flye assembled Nanopore data:

```
prokka --outdir prokkaFlyeNanoOut1206/ --prefix F21 --locustag F21
↪  --compliant --species eColi -cpus 4
↪  ~/projectSymLinkT02/work/QUAST/finalFilteredAssemblies/

flyNanoUnfiltOut.fasta
```

DFast

Analyzing the Flye assembled Nanopore data:

```
dfast -g ~/projectSymLinkT02/work/QUAST/finalFilteredAssemblies/

flyNanoUnfiltOut.fasta -o flyeNano1206/ --organism eColi --locus_tag_prefix
↪  F21 --step 10
```

Analyzing the Unicycler assembled Nanopore data:

```
dfast -g ~/projectSymLinkT02/work/QUAST/finalFilteredAssemblies/

uniNanoFilteredOut.fasta -o uniNano1206/ --organism eColi --locus_tag_prefix
↪  F21 --step 10
```

Analyzing the SPAdes assembled Illumina data:

```
dfast -g ~/projectSymLinkT02/work/BLAST/spades/illuminaFiltered/

parsedIlluminaFiltered.fasta -o spadesOut1206/ --organism eColi
↪  --locus_tag_prefix F21 --step 10
```

Step 7: Compare PROKKA DFast

	Dfast		
	FlyeNano	SPAdes Illumina	Unicycler Nano
Total Sequence Length (bp)	4736672	4586126	4736752
Number of Sequences	1	94	1
Longest Sequences (bp)	4736672	404504	4736752
N50 (bp)	4736672	137369	4736752
Gap Ratio (%)	0	0	0
GCcontent (%)	50.8	50.8	50.8
Number of CDSs	**5336**	**4225**	**5230**
Average Protein Length	254.8	315	260.7
Coding Ratio (%)	86.1	87.1	86.3
Number of rRNAs	22	4	22
Number of tRNAs	90	84	90
Number of CRISPRs	2	2	2

Figure 8: DFast analysis

	Prokka		
	FlyeNano	SPAdes Illumina	Unicycler Nano
organism	Genus ecoli strain	Genus ecoli strain	Genus ecoli strain
contigs	1	94	1
bases	4736672	586126	4736752
CDS	**5311**	**4241**	**5206**
CRISPR	2	2	2
gene	5424	4336	5319
rRNA	22	10	22
tRNA	90	84	90
tmRNA	1	1	1

Figure 9: PROKKA analysis

Figure 10: Comparison of PROKKA and DFast

PROKKA and DFAST reasonably agree for each assembly on most key metrics. However, assemblies from Nanopore data have more CoDing Sequence (CDS) counts than assemblies from Illumina data. This is because Illumina reads are more accurate than Nanopore, based on Phred quality, and because Nanopore has a hard time reading repeating base pair sequences (ex: Nanopore reads may read AAAAAAAAAAA as AA or just A). If this error happens in the middle of a protein coding sequence, it will "split" and read it as two protein coding sequences.

Step 8: Get_SNPs.pl

We used our best assembly (as discussed previously, the Unicycler assembly using Nanopore reads) as our reference genome, which we then mapped our filtered Illumina and Nanopore data to. Using this, we could filter out the Illumina and Nanopore reads which do not match what we are looking for. It is important to note that this does not produce an assembly, just individual reads (not contigs) that we have reason to believe belong to this specific *Escherichia coli* strain. We could have performed this step initially, but we did not yet have access to a reference genome of the specific strain we are looking for.

This is performed in the code seen below using `Get_SNPs.pl`, samtools, and a separate script `bam2fastq.pl` written by Professor Pombert.

Illumina

```
get_SNPs.pl --fasta incomingFastas/uniNanoFilteredOut.fasta --pe1
↪  readsFiltered/F21R1illumina.fastq --pe2 readsFiltered/F21R2illumina.fastq
↪  --mapper minimap2 -rmo -bam --var /opt/varscan/VarScan.v2.4.4.jar -preset
↪  sr --outdir trialIllumina/
```

```
samtools bam2fq -f 1 -F 12 -1 danR1out.fastq -2 danR2out.fastq
↪  ../minimap2.BAM/F21R1illumina.fastq.uniNanoFilteredOut.fasta.minimap2.bam
```

Nanopore

```
get_SNPs.pl -fa ../incomingFastas/uniNanoFilteredOut.fasta -fq
↪  ../readsFiltered/nanopore_q10_h50.fastq -rmo -bam --var
↪  /opt/varscan/VarScan.v2.4.4.jar -mapper minimap2
```

```
bam2fastq.pl -b
↪  minimap2.BAM/nanopore_q10_h50.fastq.uniNanoFilteredOut.fasta.minimap2.bam
↪  -o dmOutFastq -t se --auto map
```

The results indicate that the Nanopore data was about 1.5 gigabytes in size whereas after refining with the reference genome became 688 megabytes. As for the Illumina data, the initial filtered data was 61 megabytes with the reassembled being 21 megabytes. The total sequence counts were roughly 48 thousand and 38 thousand.

Conclusion

In this project, we began with raw sequencing data from Nanopore and Illumina sequencers and sought to reconstruct and annotate an *Escherichia coli* genome. Filtering using `fastp` and `Nanofilt` based on widely used quality score cutoffs, we obtained usable sequences that could be used in assemblies. Upon running our Illumina and Nanopore data through four types of assembly programs (Unicycler, Flye, SPAdes, and Shashta), we compared these assemblies to a list of reference genomes that included a *Escherichia coli* genome. Using BLAST and parsing the results we reconstruct these assemblies with only high enough quality contigs. We found the Unicycler assembly of Nanopore data to be the highest quality data after comparison with QUAST. After which we annotated our assemblies to determine the protein coding regions that can be used to map the Illumina and Nanopore assemblies on the reference genome using the `Get_SNPs.pl` script. Our final results present an assembled and annotated genome of *Escherichia coli*.