Search for mutations responsible for antibiotic resistance in E. coli genome

Data preparation

Get needed information with command `wget`:

`wget https://ftp.ncbi.nlm.nih.gov/genomes/genbank/bacteria/Escherichia_coli/reference/GCA_000005845.2_ASM584v2/GCA_000005845.2_ASM584v2_genomic.fna.gz`
 	
 `wget https://ftp.ncbi.nlm.nih.gov/genomes/genbank/bacteria/Escherichia_coli/reference/GCA_000005845.2_ASM584v2/GCA_000005845.2_ASM584v2_genomic.gff.gz`
	Download reference genome of E.Coli strain K-12 substrain MG1655 (sequence in .fna.gz and annotation in .gff.gz)

`wget https://figshare.com/articles/dataset/amp_res_2_fastq_zip/10006541/*`	Download raw Illumina sequencing reads from shotgun sequencing of an E. coli strain that is resistant to the antibiotic ampicillin (.fastq.gz)

Unpack files with `gunzip`:

`gunzip -k <file.gz>`	`-k` -- don't delete input file(s)

Rename files with `mv`:

`mv <old_name> <new_name>`	rename files so that: ref_seq_EColi.fna -- ref genome sequence ref_seq_EColi.gff -- ref genome annotation amp_res_1.fastq -- forward reads amp_res_2.fastq -- reverse reads

Inspect files manually with head, to see what's inside:

`head -20 <file>`	`-n` -- first n lines will be printed to stdout

N.B. Don't try to use `head` with archives (like .tar, .gz, ...), because they contain compressed information in binary format (the result will be not-human-readable). If you want to inspect some file inside an archive, you can use `zcat <file> | head` instead.

Calculate number of reads with `wc -l`:

`wc -l amp_res_*`	`-l` -- print the new line counts

Output:
```
1823504 amp_res_1.fastq
1823504 amp_res_2.fastq
3647008 total
```

The result of this command is the number of new lines in the file. We have to divide it by 4 to get the actual number of reads (each read in .fastq has 4 lines of information). As a result, there are 455876 reads in each file.

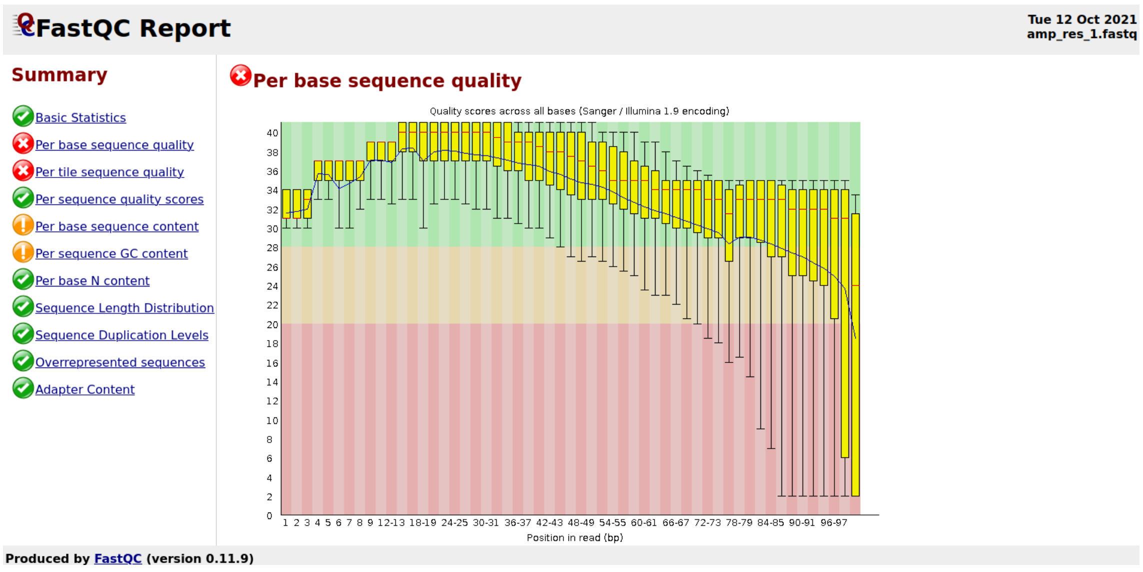
FastQC

We can inspect reads quality with `fastqc`:

`fastqc <seqfile1> <seqfile2>`	We can specify more than one fastq file

As a result, we get a report (consisting of .html file and .zip archive with images and all calculated statistics) for each file. Html file contains different sections with many graphics, and contains information about different aspects of our data. For example, in the "Basic Statistics" section we can see what encoding type was used while sequencing (Sanger / Illumina 1.9, which uses Phred33 scale [more info]) or how many reads we have (455876, which coincides with the result that we got from `wc -l`).

For every section we can see whether everything is OK or not. It can be seen, for example, that per base sequence quality is low on the ends of reads.


FastQC Report

Tue 12 Oct 2021
amp_res_1.fastq

Summary

- ✓ Basic Statistics
- ✗ Per base sequence quality
- ✗ Per tile sequence quality
- ✓ Per sequence quality scores
- ! Per base sequence content
- ! Per sequence GC content
- ✓ Per base N content
- ✓ Sequence Length Distribution
- ✓ Sequence Duplication Levels
- ✓ Overrepresented sequences
- ✓ Adapter Content

✗ Per base sequence quality

Produced by FastQC (version 0.11.9)

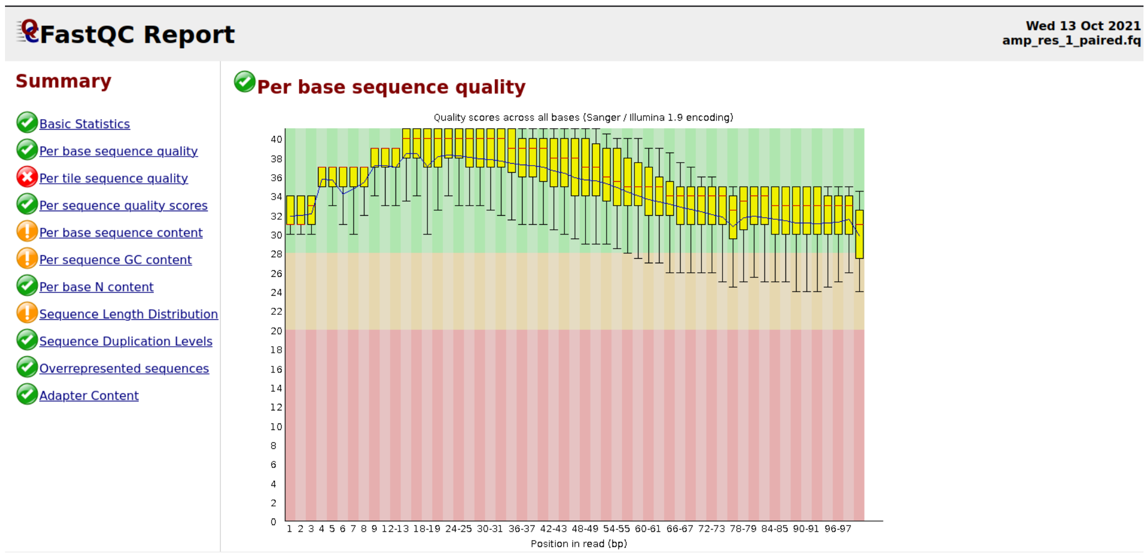
Filtering the reads

We can try to increase quality by filtering bad parts of the reads using `Trimmomatic`:

`TrimmomaticPE -phred33`
`amp_res_1.fastq.gz`
`amp_res_2.fastq.gz`
`amp_res_1_paired.fq`
`amp_res_1_unpaired.fq`
`amp_res_2_paired.fq`
`amp_res_2_unpaired.fq`
`LEADING:20 TRAILING:20`
`SLIDINGWINDOW:10:20`
`MINLEN:20`
	-phred33 / -phred64 -- type of coding quality score
	input: 2 .fastq.gz files with with forward and reverse reads
	output: 4 .fq files, in <paired> files BOTH the forward and its matching reverse read passed the trimming filter, and <unpaired> are singleton files where only one read passed the trimming filter
	For each parameter we have to set certain quality threshold (=20 in our command)

Output: Input Read Pairs: 455876 Both Surviving: 446259 (97.89%) Forward Only Surviving: 9216 (2.02%) Reverse Only Surviving: 273 (0.06%) Dropped: 128 (0.03%)

97.89% of our reads passed the trimming filter. If we set the quality threshold to 30, 79.01% of our reads will pass. But if we look at the FastQC report, this will not lead to significant differences in the quality of reads compared to quality = 20. So we ended up with quality threshold =20.


FastQC Report

Wed 13 Oct 2021
amp_res_1_paired.fq

Summary

- ✓ Basic Statistics
- ✓ Per base sequence quality
- ✗ Per tile sequence quality
- ✓ Per sequence quality scores
- ! Per base sequence content
- ! Per sequence GC content
- ✓ Per base N content
- ! Sequence Length Distribution
- ✓ Sequence Duplication Levels
- ✓ Overrepresented sequences
- ✓ Adapter Content

✓ Per base sequence quality

The mean quality has increased from 32.9±4.4 to 34.5±2.6, and we got rid of big errors at the ends of reads.

N.B. Bad reads won't map to reference genome, so no need to try to make them perfect.

Aligning sequences to reference

Index our reference genome with `bwa index`:

```
bwa index ref_seq_EColi.fna
```

Align reads with `bwa mem`:

```
bwa mem ref_seq_EColi.fna amp_res_1_paired.fq amp_res_2_paired.fq
> alignment.sam
```

N.B. Be sure that all index files from ”`bwa index`” are in the same folder with reference genome (ref_seq_EColi.fna).

Compress alignment (.sam) file with `samtools view`:

`samtools view -S -b alignment.sam > alignment.bam`	`-S` -- ignored (input format is auto-detected) `-b` -- output .bam

Now we can see some statistics about our mapping using `samtools flagstat`:

```
samtools flagstat
alignment.bam
```

Output:

```
892776 + 0 in total (QC-passed reads +
QC-failed reads)
0 + 0 secondary
258 + 0 supplementary
0 + 0 duplicates
891649 + 0 mapped (99.87% : N/A)
892518 + 0 paired in sequencing
446259 + 0 read1
446259 + 0 read2
888554 + 0 properly paired (99.56% : N/A)
890412 + 0 with itself and mate mapped
979 + 0 singletons (0.11% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr
(mapQ>=5)
```

99.87% of our reads (or 97.45% of all reads that we had before trimming filter) mapped correctly.

Alignment (.bam) file should be sorted and indexed for faster search:

`samtools sort alignment.bam -o alignment_sorted.bam`	sort file

`samtools index alignment_sorted.bam`	index file

Variant calling

Create intermediate (.mpileup) file with info about how many reads have a mutation at the same position:

```
samtools mpileup -f ref_seq_EColi.fna alignment_sorted.bam >
my.mpileup
```

Call variants with VarScan v2.3.9:

`java -jar VarScan.v2.3.9.jar mpileup2snp my.mpileup --min-var-freq 0.8 --variants --output-vcf 1 > VarScan_results.vcf`	mpileup2snp -- review only SNPs (there is also mpileup2indel command for reviewing indels) --min-var-freq -- the minimum % of non-reference bases at a position required to call it a mutation in the sample --variants -- only output positions that are above our threshold --output-vcf 1 -- output in .vcf format

Output:

```
4641343 bases in pileup file
9 variant positions (6 SNP, 3 indel)
1 were failed by the strand-filter
5 variant positions reported (5 SNP, 0 indel)
```

As a result, we get 5 SNP in the .vcf file.

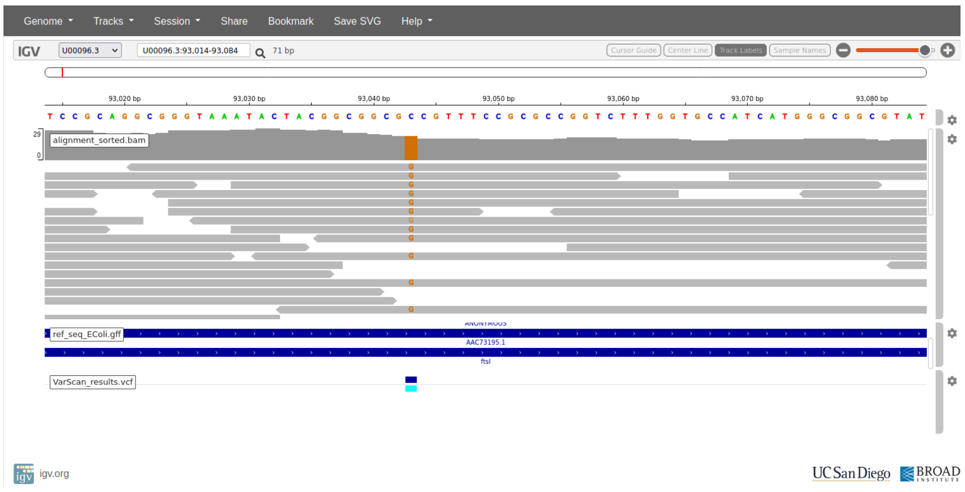
IGV

In the end, we can look at our alignment using IGV browser. First, we have to create index file for reference genome:

`samtools faidx ref_seq_EColi.fna`	Create index file for .fasta

Then, we can load everything in IGV browser (https://igv.org/app/):

Genome -> Local File… ->	Load genome and index (.fna and .fai)
Tracks -> Local File… ->	Load sorted alignment file with index (.bam and .bam.bai)
Tracks -> Local File… ->	Load genome annotation (.gff)



To get the same information in console, we can use:

bedtools intersect -loj -a VarScan_results.vcf -b ref_seq_EColi.gff	Report overlaps between two feature files (-a and -b)