LINUX WORKSHOP

02 Variant Calling

By Jiajia Li (ANU Biological Data Science Institute)

28/02/2025

Australian National University

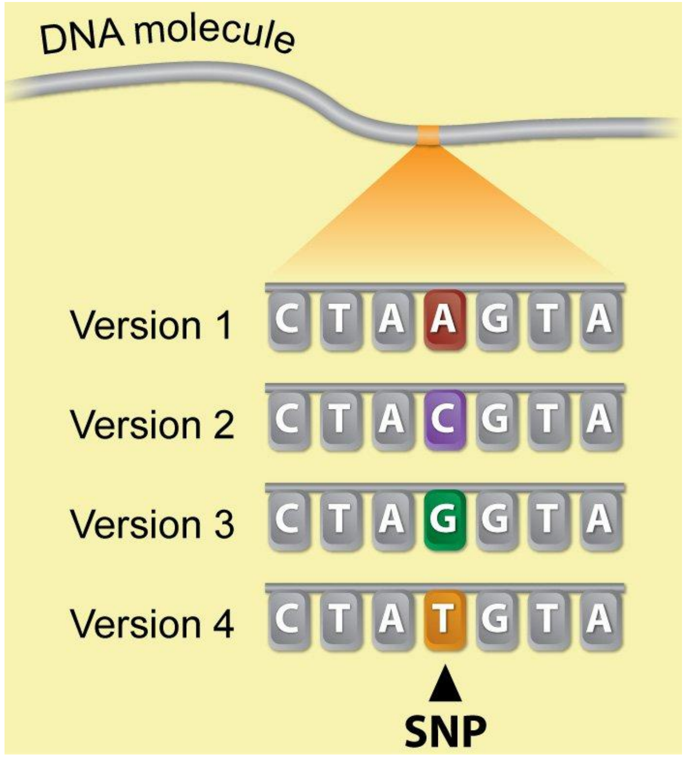
Learning Objectives

- What is variant calling?
- Background of data
- Paired-end sequencing
- FASTQ file format
- Run FastQC for quality control
- Run Trimmomatic to remove low quality sequence
- Use for loop with Trimmomatic

Variant Calling

Variant calling is a bioinformatics process used in genomics to identify **genetic variations** between the genome of an individual and a reference genome.

These variants can include **single-nucleotide polymorphisms** (SNPs), small insertions and deletions (indels), and larger structural variants.



Data

The data we are going to use is from *E. coli*, and it is part of a long-term evolution experiment led by Richard Lenski.

The experiment was designed to assess adaptation in *E. coli*. A population was propagated more than 40,000 generations in a **glucose-limited** minimal medium. This medium was supplemented with citrate. Which E. coli cannot metabolise in the aerobic conditions of the experiment.

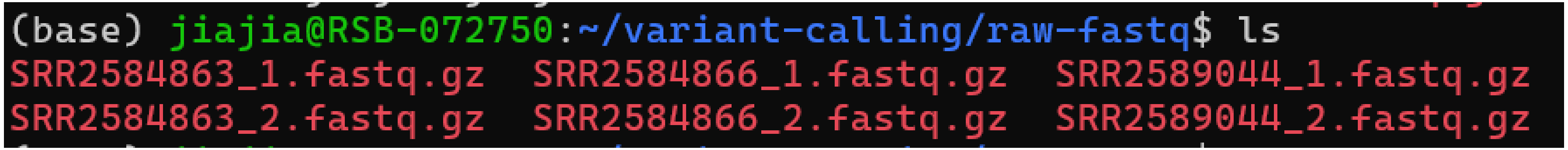
Sequencing of the populations at regular time points revealed that spontaneous citrate-using variant (**Cit+**) appeared between 31,000 and 31,500 generations, causing an increasing in population size and diversity.

Data

We will be working with 3 sample events from the Ara-3 strain of this experiment, one from 5,000 generations, one from 15,000 generations, and one from 50,000 generations.

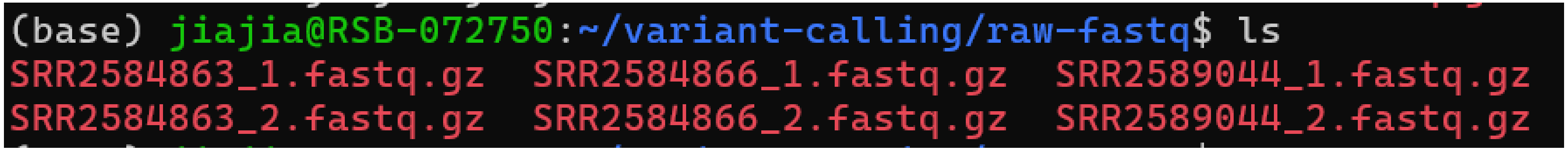
We will explore how many SNPs within each sample.

We have downloaded the data before, and it should be in directory `~/variant-calling/raw-fastq`.



```
(base) jiajia@RSB-072750:~/variant-calling/raw-fastq$ ls
SRR2584863_1.fastq.gz  SRR2584866_1.fastq.gz  SRR2589044_1.fastq.gz
SRR2584863_2.fastq.gz  SRR2584866_2.fastq.gz  SRR2589044_2.fastq.gz
```

Data



```
(base) jiajia@RSB-072750:~/variant-calling/raw-fastq$ ls
SRR2584863_1.fastq.gz  SRR2584866_1.fastq.gz  SRR2589044_1.fastq.gz
SRR2584863_2.fastq.gz  SRR2584866_2.fastq.gz  SRR2589044_2.fastq.gz
```

- .gz?
- Check the size of the files

The _1 and _2 of our files means it is **paired-end sequencing**. Our samples were sequenced from both ends. In the analyses, we will use both files.

FASTQ file format

FASTQ file format is a text-based format for storing both a **biological sequence** and its corresponding **quality scores**. It is the standard format for storing the output of high-throughput sequencing instruments.

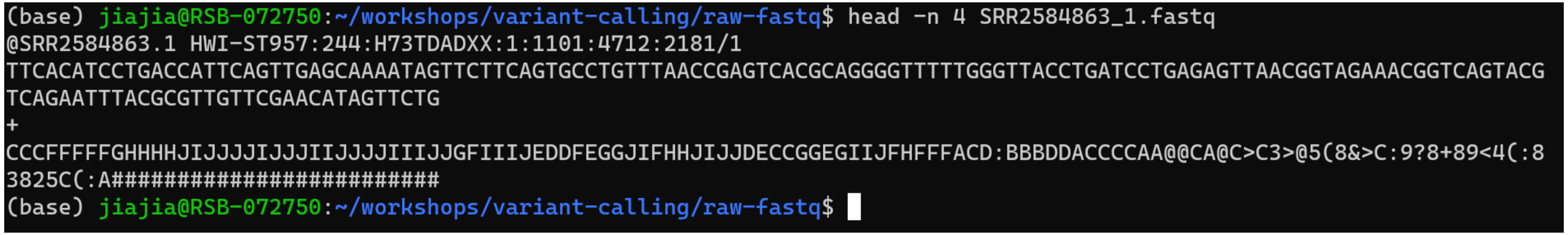
A FASTQ file has four line-separated fields per sequence:

- Line 1 begins with an **@** character and is followed by a sequence identifier and an optional description.
- Line 2 is the raw sequence letters. A, T, G, C
- Line 3 begins with a + character and is optionally followed by the same sequence identifier again.
- Line 4 encodes the quality values for the sequence in line 2 and must contain the same number of symbols.

FASTQ file format

We can view the first complete read in one of our sequence files:

- `head –n 4 SRR2584863_1.fastq`
- You have to unzip the .gz file to view it.
- But we can also use `less`, and it works on .gz files.



```
(base) jiajia@RSB-072750:~/workshops/variant-calling/raw-fastq$ head -n 4 SRR2584863_1.fastq
@SRR2584863.1 HWI-ST957:244:H73TDADXX:1:1101:4712:2181/1
TTCACATCCTGACCATTCAGTTGAGCAAAATAGTTCTTCAGTGCCTGTTTAACCGAGTCACGCAGGGGTTTTTGGGTTACCTGATCCTGAGAGTTAACGGTAGAAACGGTCAGTACG
TCAGAATTTACGCGTTGTTCGAACATAGTTCTG
+
CCCFFFFFGHHHHJIJJJJIJJJIIJJJJIIIJJGFIIIJEDDFEGGJIFHHJIJJDECCGGEGIIJFHFFFACD:BBBDDACCCCAA@@CA@C>C3>@5(8&>C:9?8+89<4(:8
3825C(:A#########################
(base) jiajia@RSB-072750:~/workshops/variant-calling/raw-fastq$
```

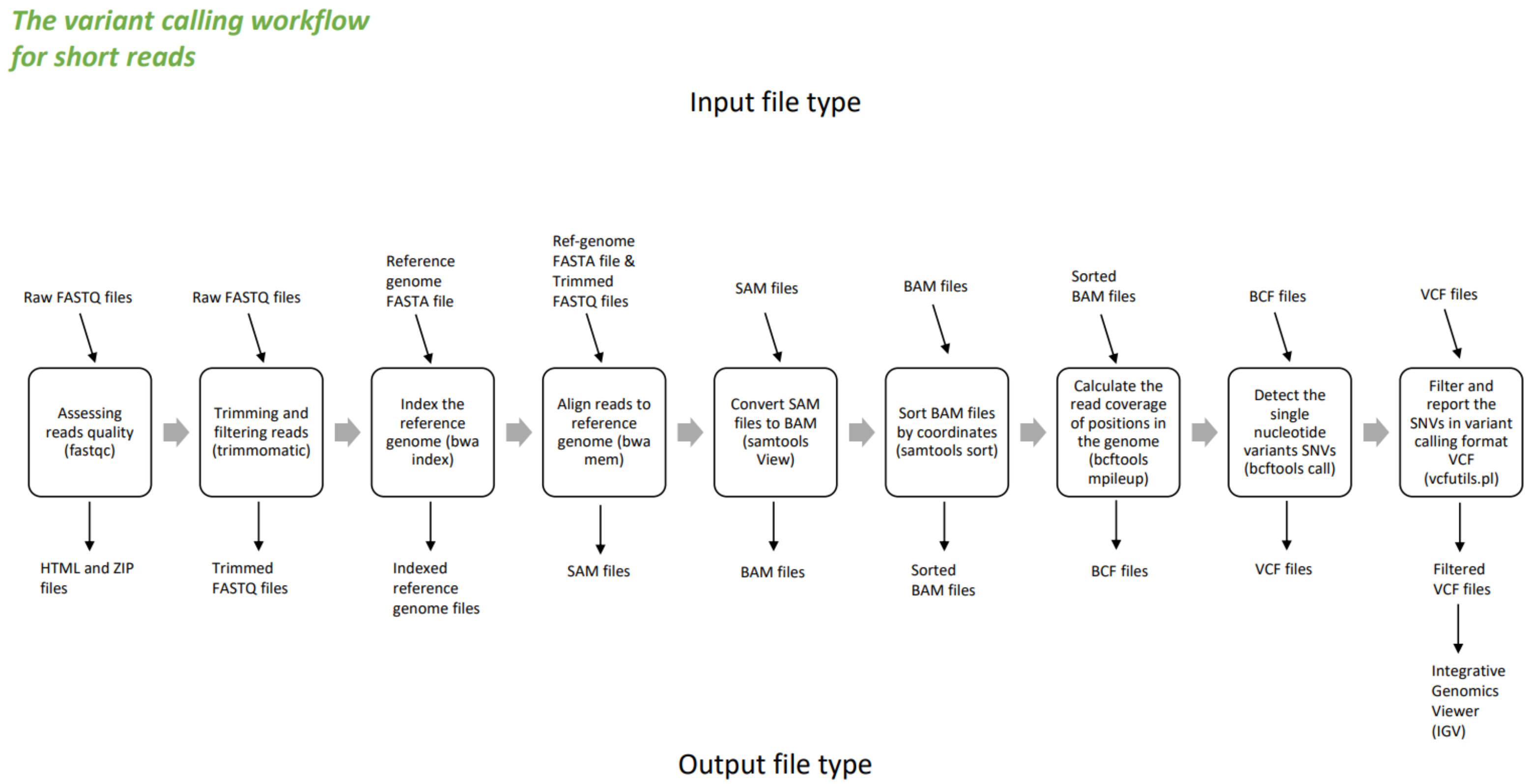
The variant calling workflow

When working with high-throughput sequencing data, the raw reads you get from the sequencer need to pass through several bioinformatics tools to generate your desired output.

The execution of this set of tools in a specific order is commonly referred to as a workflow or pipeline.

There are 9 steps in our variant calling workflow, and we will learn it step by step together.

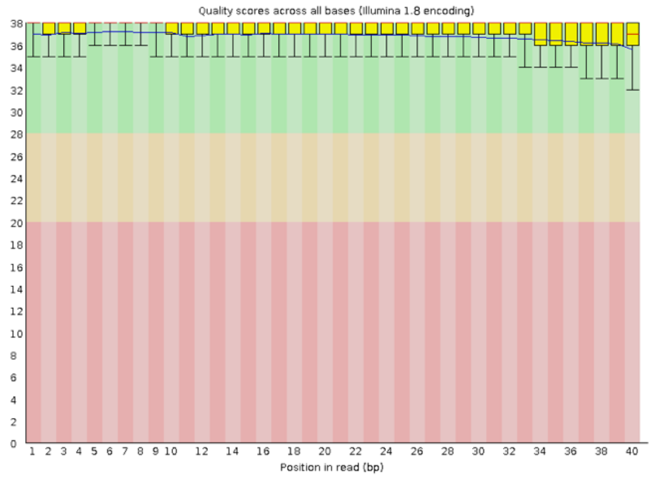
The variant calling workflow

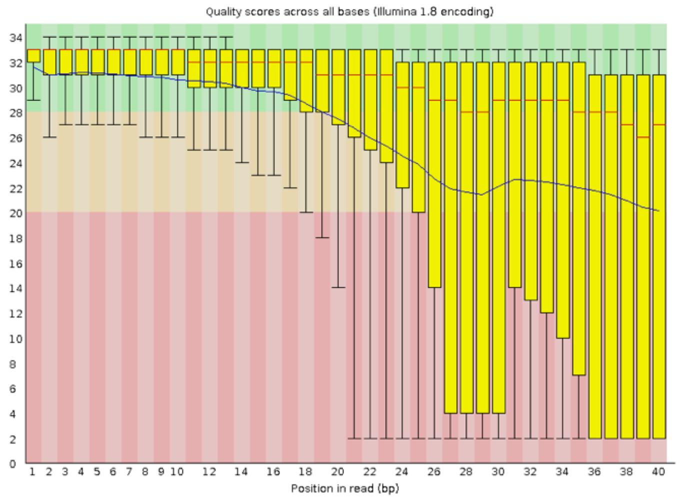

The variant calling workflow
for short reads
Input file type
Raw FASTQ files
Raw FASTQ files
Reference genome FASTA file
Ref-genome FASTA file & Trimmed FASTQ files
SAM files
BAM files
Sorted BAM files
BCF files
VCF files
Assessing reads quality (fastqc)
Trimming and filtering reads (trimmomatic)
Index the reference genome (bwa index)
Align reads to reference genome (bwa mem)
Convert SAM files to BAM (samtools View)
Sort BAM files by coordinates (samtools sort)
Calculate the read coverage of positions in the genome (bcftools mpileup)
Detect the single nucleotide variants SNVs (bcftools call)
Filter and report the SNVs in variant calling format VCF (vcfutils.pl)
HTML and ZIP files
Trimmed FASTQ files
Indexed reference genome files
SAM files
BAM files
Sorted BAM files
BCF files
VCF files
Filtered VCF files
Integrative Genomics Viewer (IGV)
Output file type

Step 1 – Assessing reads quality using FastQC

FastQC is a widely used bioinformatics tools for assessing the quality of high-throughput sequence data, particularly FASTQ files.

It provides detailed reports on **sequence quality**, **GC content**, **adapter contamination** and more.

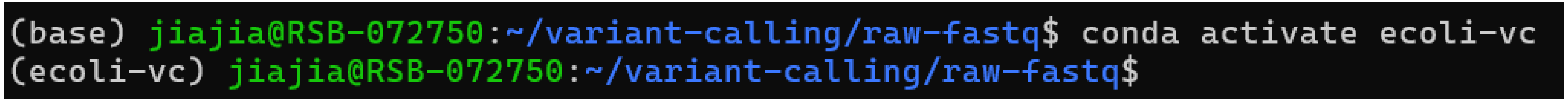




Step 1 – Assessing reads quality using FastQC

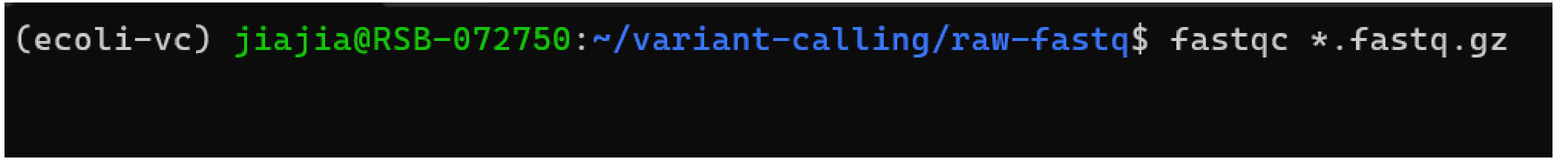
Now, let’s run FastQC on our sequences.

First, we must activate our conda environment.



```
(base) jiajia@RSB-072750:~/variant-calling/raw-fastq$ conda activate ecoli-vc
(ecoli-vc) jiajia@RSB-072750:~/variant-calling/raw-fastq$
```

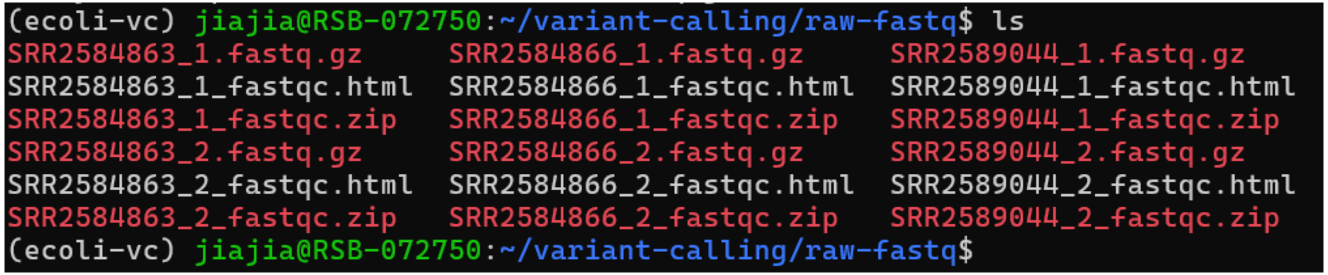
Then, we can use wildcard * to match our file names so FastQC can run for all our samples together.



```
(ecoli-vc) jiajia@RSB-072750:~/variant-calling/raw-fastq$ fastqc *.fastq.gz
```

Step 1 – Assessing reads quality using FastQC

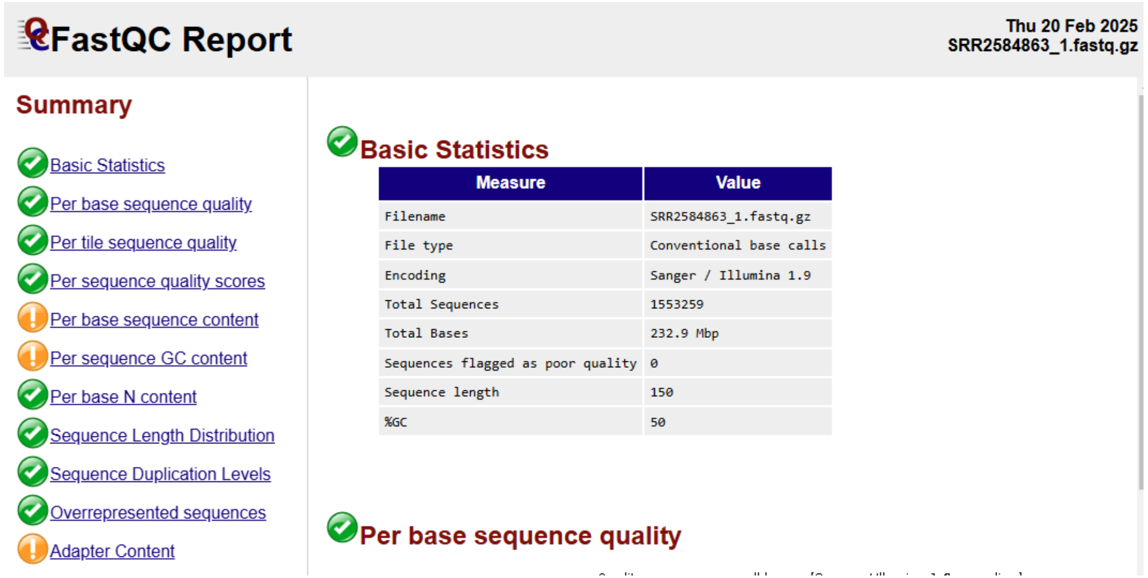
After finish running, you will get a `fastqc.html` and a `fastqc.zip` file for each FASTQ input.



```
(ecoli-vc) jiajia@RSB-072750:~/variant-calling/raw-fastq$ ls
SRR2584863_1.fastq.gz       SRR2584866_1.fastq.gz       SRR2589044_1.fastq.gz
SRR2584863_1_fastqc.html    SRR2584866_1_fastqc.html    SRR2589044_1_fastqc.html
SRR2584863_1_fastqc.zip     SRR2584866_1_fastqc.zip     SRR2589044_1_fastqc.zip
SRR2584863_2.fastq.gz       SRR2584866_2.fastq.gz       SRR2589044_2.fastq.gz
SRR2584863_2_fastqc.html    SRR2584866_2_fastqc.html    SRR2589044_2_fastqc.html
SRR2584863_2_fastqc.zip     SRR2584866_2_fastqc.zip     SRR2589044_2_fastqc.zip
(ecoli-vc) jiajia@RSB-072750:~/variant-calling/raw-fastq$
```

You can use browser to open the HTML file and read the report.

Step 1 – Assessing reads quality using FastQC



FastQC Report

Thu 20 Feb 2025
SRR2584863_1.fastq.gz

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

✓ **Basic Statistics**

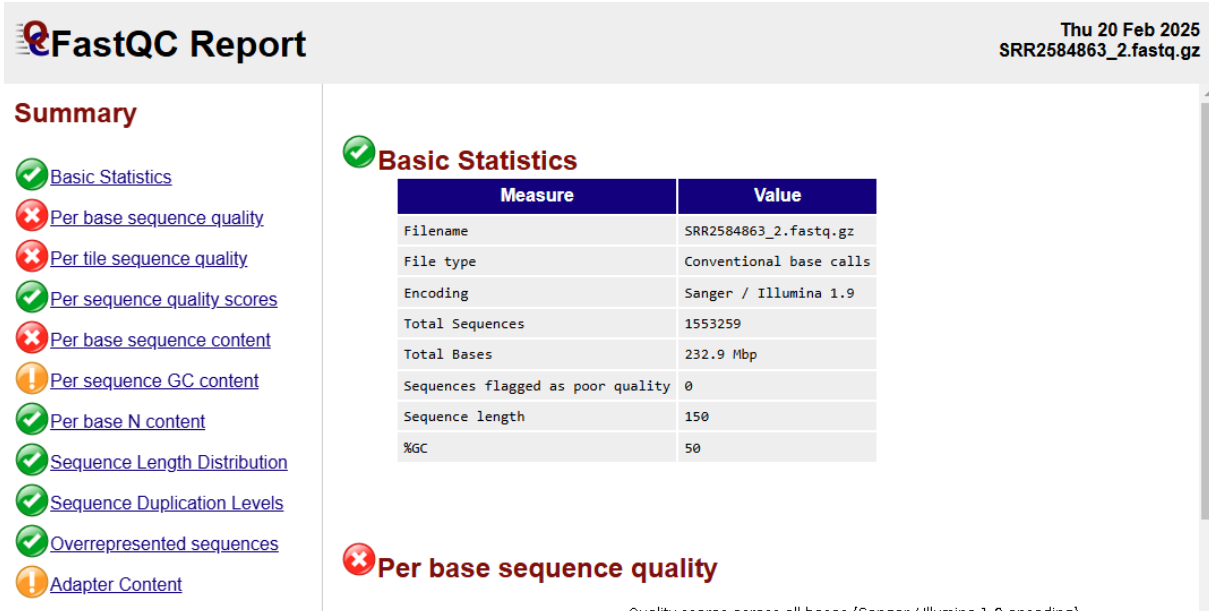
Measure	Value
Filename	SRR2584863_1.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1553259
Total Bases	232.9 Mbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	50

✓ **Per base sequence quality**

- Pass
- Warning – may need attention
- Red – likely an issue

SRR2584863_1.fastq.gz seems to have good quality.

Step 1 – Assessing reads quality using FastQC



FastQC Report

Thu 20 Feb 2025
SRR2584863_2.fastq.gz

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

Basic Statistics

Measure	Value
Filename	SRR2584863_2.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1553259
Total Bases	232.9 Mbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	50

Per base sequence quality

SRR2584863_2.fastq.gz seems to have a few issue.

That's the quality of raw sequence data.

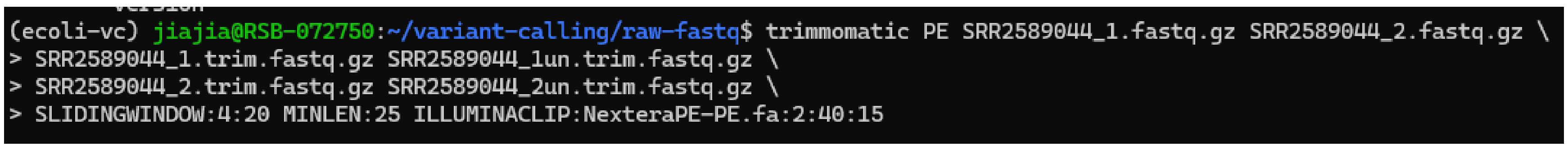
Next, we are going to do a couple standard cleaning process such as **remove low quality reads** and **trim adapter.**

Step 2 – Trimming and filtering reads (Trimmomatic)

From the FastQC results, we know that some of our samples have failed a few quality metrics. This does not mean that our samples should be thrown out. It is very common to have some quality metrics fail, and this may or may not be a problem for your downstream application.

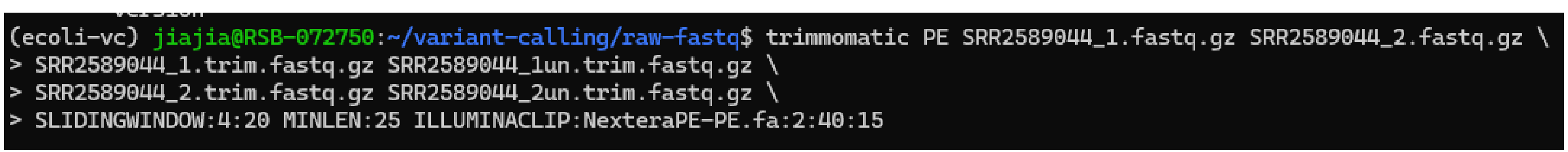
Trimmomatic is a widely used bioinformatics tool for preprocessing Illumina NGS data. It removes low-quality bases, adapter sequences, and short reads.

Basic usage on paired-end reads:



```
(ecoli-vc) jiajia@RSB-072750:~/variant-calling/raw-fastq$ trimmomatic PE SRR2589044_1.fastq.gz SRR2589044_2.fastq.gz \
> SRR2589044_1.trim.fastq.gz SRR2589044_1un.trim.fastq.gz \
> SRR2589044_2.trim.fastq.gz SRR2589044_2un.trim.fastq.gz \
> SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
```

Step 2 – Trimming and filtering reads (Trimmomatic)



```
(ecoli-vc) jiajia@RSB-072750:~/variant-calling/raw-fastq$ trimmomatic PE SRR2589044_1.fastq.gz SRR2589044_2.fastq.gz \
> SRR2589044_1.trim.fastq.gz SRR2589044_1un.trim.fastq.gz \
> SRR2589044_2.trim.fastq.gz SRR2589044_2un.trim.fastq.gz \
> SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
```

- `PE` - indicates paired-end reads
- `trim.fastq.gz` - reads were survived from filtering
- `un.trim.fastq.gz` - reads were dropped from filtering

- `SLIDINGWINDOW:4:20` - perform sliding window trimming, cutting once the average quality within the window falls below a threshold. Window size 4, cutting once average below 20.
- `MINLEN:25` - drop an entire read if it is below 25 bases.
- `ILLUMINACLIP:NexteraPE-PE.fa:2:40:15` - perform adapter removal.

Step 2 – Trimming and filtering reads (Trimmomatic)

Before run the command, we need to have the **adapter file** ready in our folder, so the software knows where it is.

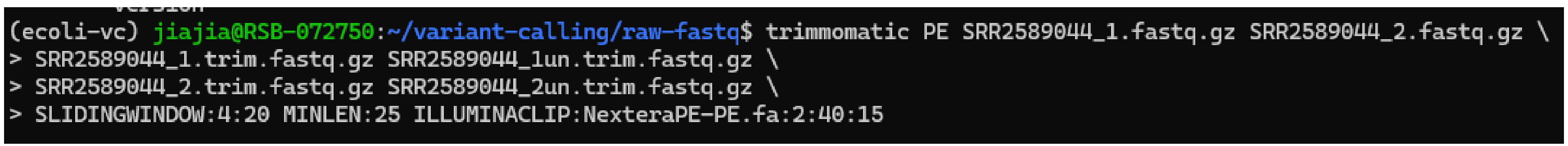
The adapter file we need is “**NexteraPE-PE.fa**”, and it was downloaded already when we install Trimmomatic. But it is in a different folder, it stores in the same place as we install Trimmomatic.

In my case, the adapter file is in `~/miniconda3/pkgs/trimmomatic-0.39-hdfd78af_2/share/trimmomatic-0.39-2/adapters/NexteraPE-PE.fa`.

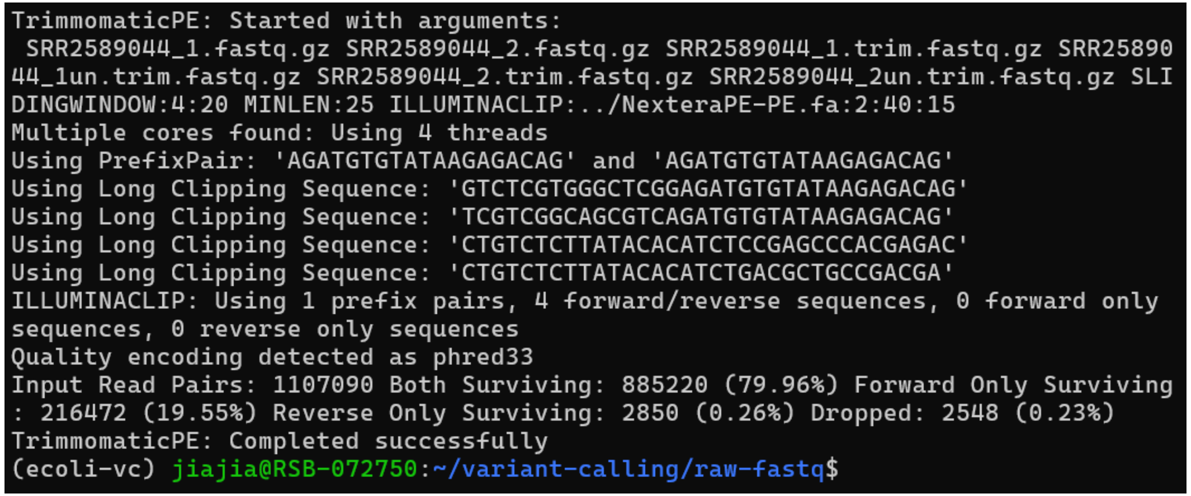
Yours could be in `~/anaconda3/pkgs/trimmomatic-version/…`, you can go through the directories to find it and copy it to `~/variant-calling/raw-fastq`.

Step 2 – Trimming and filtering reads (Trimmomatic)

After copy the adapter file, let’s run the code.



```
(ecoli-vc) jiajia@RSB-072750:~/variant-calling/raw-fastq$ trimmomatic PE SRR2589044_1.fastq.gz SRR2589044_2.fastq.gz \
> SRR2589044_1.trim.fastq.gz SRR2589044_1un.trim.fastq.gz \
> SRR2589044_2.trim.fastq.gz SRR2589044_2un.trim.fastq.gz \
> SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
```

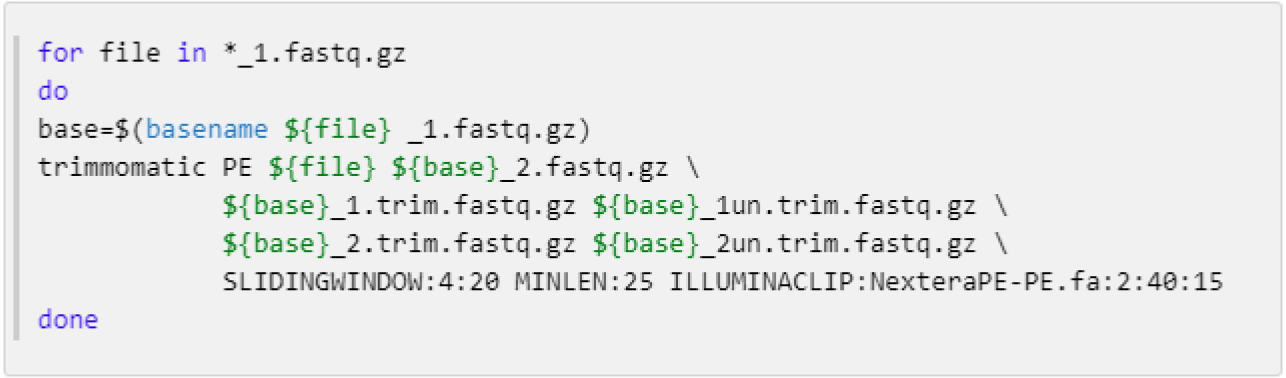


```
TrimmomaticPE: Started with arguments:
 SRR2589044_1.fastq.gz SRR2589044_2.fastq.gz SRR2589044_1.trim.fastq.gz SRR25890
44_1un.trim.fastq.gz SRR2589044_2.trim.fastq.gz SRR2589044_2un.trim.fastq.gz SLI
DINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:../NexteraPE-PE.fa:2:40:15
Multiple cores found: Using 4 threads
Using PrefixPair: 'AGATGTGTATAAGAGACAG' and 'AGATGTGTATAAGAGACAG'
Using Long Clipping Sequence: 'GTCTCGTGGGCTCGGAGATGTGTATAAGAGACAG'
Using Long Clipping Sequence: 'TCGTCGGCAGCGTCAGATGTGTATAAGAGACAG'
Using Long Clipping Sequence: 'CTGTCTCTTATACACATCTCCGAGCCCACGAGAC'
Using Long Clipping Sequence: 'CTGTCTCTTATACACATCTGACGCTGCCGACGA'
ILLUMINACLIP: Using 1 prefix pairs, 4 forward/reverse sequences, 0 forward only
sequences, 0 reverse only sequences
Quality encoding detected as phred33
Input Read Pairs: 1107090 Both Surviving: 885220 (79.96%) Forward Only Surviving
: 216472 (19.55%) Reverse Only Surviving: 2850 (0.26%) Dropped: 2548 (0.23%)
TrimmomaticPE: Completed successfully
(ecoli-vc) jiajia@RSB-072750:~/variant-calling/raw-fastq$
```

Step 2 – Trimming and filtering reads (Trimmomatic)

We have run Trimmomatic on sample `SRR2589044`, there are 2 more samples to run.

We can also write a for loop to run all 3 samples together:



```
for file in *_1.fastq.gz
do
base=$(basename ${file} _1.fastq.gz)
trimmomatic PE ${file} ${base}_2.fastq.gz \
            ${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
            ${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
            SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
done
```

What does `basename` do?

Step 2 – Trimming and filtering reads (Trimmomatic)

Until now, we have been working in the `/raw-fastq` folder, but to keep our project folder clean and organised, we should separate out raw files and result files.

Exercise – create a new folder `trimmed-fastq` and move all the output of Trimmomatic into it.

Exercise – run FastQC again on our trimmed files.

Other trimming and filtering software include **Cutadapt**, if you see other pipelines using these software they are doing the same thing for the sequences.

Step 3 – Alignment to a reference genome (BWA)

Burrows Wheeler Aligner (BWA) is a software for mapping low-divergent sequences against a large reference genome.

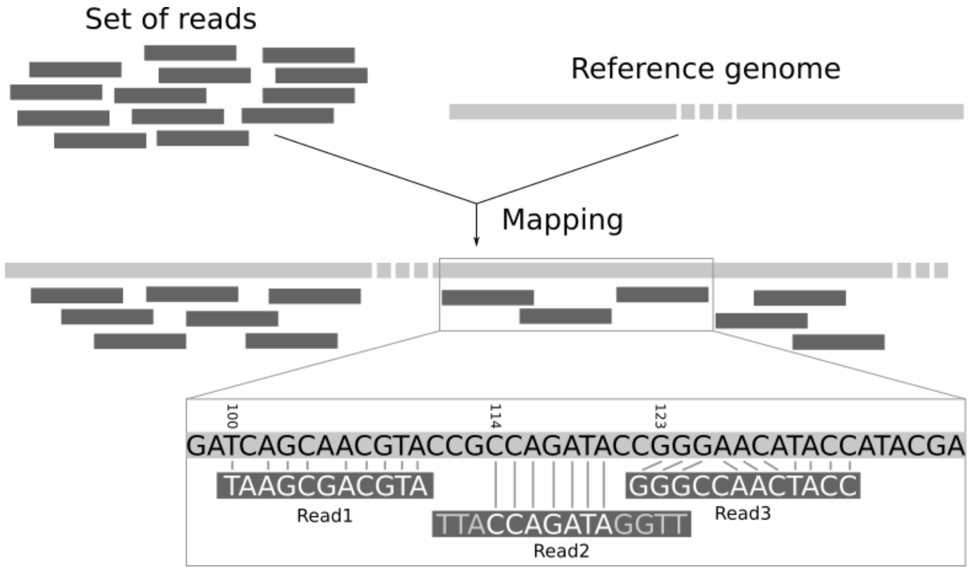
There are other aligners:

Short-read aligners:
- Bowtie2 – fast, memory-efficient, great for small genomes
- STAR – best for RNA-seq

Long-read aligners:
- Minimap2 – fast and efficient for long reads
- NGMLR – good for detecting structural variants

Step 3 – Alignment to a reference genome (BWA)

Read mapping/alignment is the process of aligning short or long sequencing reads to a reference genome. It is a key step in analysing data from high-throughput sequencing technologies like Illumina, PacBio, and Oxford Nanopore.



Step 3 – Alignment to a reference genome (BWA)

Our reference genome – *E. coli* REL606, the original population of our experiment.

Practise – download the reference genome:

1. Create a new folder `~/variant-calling/ref-genome`
2. Go to the new folder
3. Download the reference genome file from this link https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/017/985/GCA_000017985.1_ASM1798v1/GCA_000017985.1_ASM1798v1_genomic.fna.gz

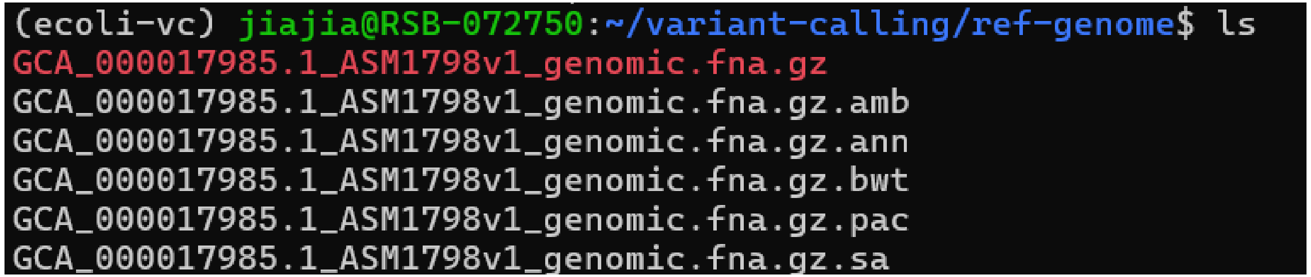
Step 3 – Alignment to a reference genome (BWA)

After we have the genome, we need first to index it and then can start the read alignment.

Indexing allows the aligner to quickly find potential alignment sites for query sequences in a genome, which saves time during alignment.

Indexing only needs to be run once.

Run `bwa index [genome]` to index the genome.



```
(ecoli-vc) jiajia@RSB-072750:~/variant-calling/ref-genome$ ls
GCA_000017985.1_ASM1798v1_genomic.fna.gz
GCA_000017985.1_ASM1798v1_genomic.fna.gz.amb
GCA_000017985.1_ASM1798v1_genomic.fna.gz.ann
GCA_000017985.1_ASM1798v1_genomic.fna.gz.bwt
GCA_000017985.1_ASM1798v1_genomic.fna.gz.pac
GCA_000017985.1_ASM1798v1_genomic.fna.gz.sa
```

These files store the indexing information, so we don't need to run them again.

Step 3 – Alignment to a reference genome (BWA)

Now we can start to align our sequences to the reference genome.

```
bwa mem [ref_genome] [sample_R1.fastq] [sample_R2.fastq] > [output.sam]
```

Run alignment on sample SRR2584863.

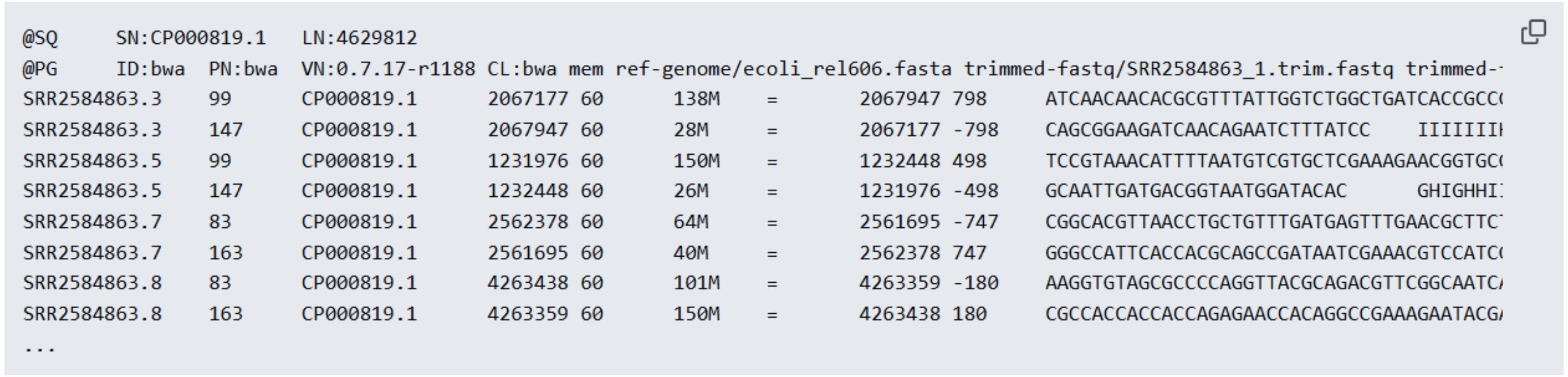
Create a new folder called “results”, then store all the result files in it.

You can name the output file “SRR2584863.aligned.sam”.

Step 3 – Alignment to a reference genome (BWA)

SAM file format: SAM stands for Sequence Alignment Map format.

- It is a TAB-delimited text format consisting of a header section, and an alignment section.
- Header lines start with @, while alignment lines do not.
- Each alignment line has 11 mandatory fields for essential alignment information such as mapping position, and variable number of optional fields for flexible or aligner specific information.



```
@SQ     SN:CP000819.1   LN:4629812
@PG     ID:bwa  PN:bwa  VN:0.7.17-r1188 CL:bwa mem ref-genome/ecoli_rel606.fasta trimmed-fastq/SRR2584863_1.trim.fastq trimmed-
SRR2584863.3    99      CP000819.1      2067177 60      138M    =       2067947 798     ATCAACAACACGCGTTTATTGGTCTGGCTGATCACCGCC
SRR2584863.3    147     CP000819.1      2067947 60      28M     =       2067177 -798    CAGCGGAAGATCAACAGAATCTTTATCC    IIIIIII
SRR2584863.5    99      CP000819.1      1231976 60      150M    =       1232448 498     TCCGTAAACATTTTAATGTCGTGCTCGAAAGAACGGTGC
SRR2584863.5    147     CP000819.1      1232448 60      26M     =       1231976 -498    GCAATTGATGACGGTAATGGATACAC      GHIGHHI
SRR2584863.7    83      CP000819.1      2562378 60      64M     =       2561695 -747    CGGCACGTTAACCTGCTGTTTGATGAGTTTGAACGCTTC
SRR2584863.7    163     CP000819.1      2561695 60      40M     =       2562378 747     GGGCCATTCACCACGCAGCCGATAATCGAAACGTCCATC
SRR2584863.8    83      CP000819.1      4263438 60      101M    =       4263359 -180    AAGGTGTAGCGCCCCAGGTTACGCAGACGTTCGGCAATC
SRR2584863.8    163     CP000819.1      4263359 60      150M    =       4263438 180     CGCCACCACCACCAGAGAACCACAGGCCGAAAGAATACG
...
```

Step 3 – Alignment to a reference genome (BWA)

SAM file is normally big, so to save space, we can convert SAM to BAM. BAM is the compressed binary version of SAM, and it stores the same information.

To convert SAM to BAM:

```
samtools view -b [aligned.sam] > [aligned.bam]
```

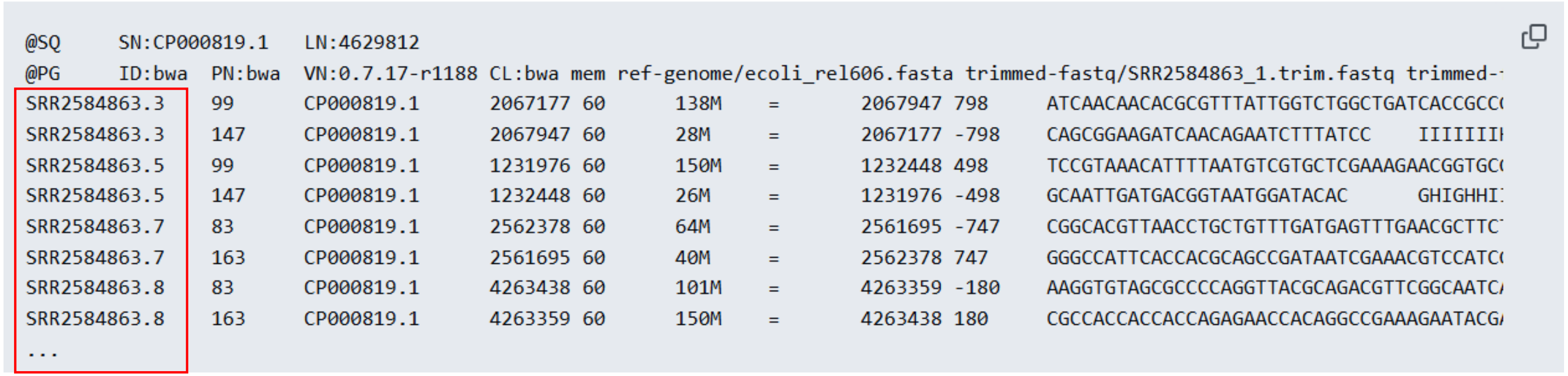
Normally we don't need to see the SAM files, so we can pipe the two steps together to only save the BAM files for downstream analysis.

```
bwa mem [ref_genome] [sample_R1.fastq] [sample_R2.fastq] | samtools view -b > [aligned.bam]
```

Try this on sample SRR2584866.

Step 4 – Sort BAM file by coordinates

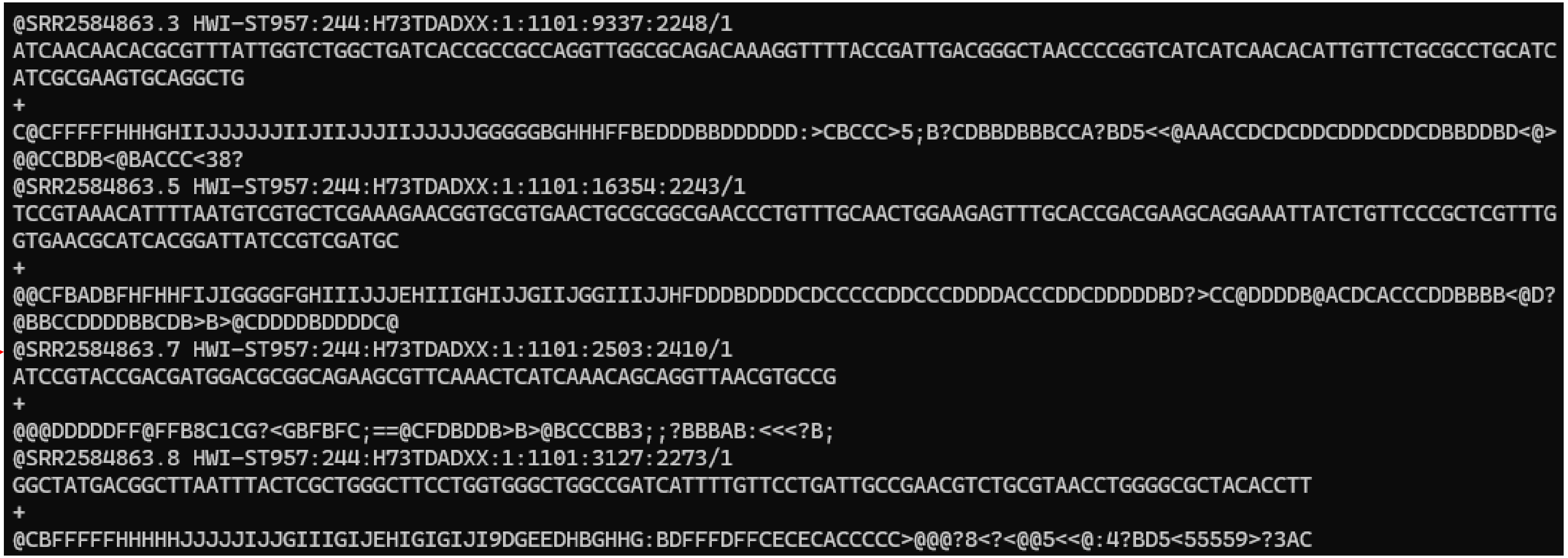
If you have a look of the previous SAM file, the sequences are ordered the same as the fastq file. The reads in fastq files are ordered by the time they go through the sequencing machine.



```
@SQ     SN:CP000819.1   LN:4629812
@PG     ID:bwa  PN:bwa  VN:0.7.17-r1188 CL:bwa mem ref-genome/ecoli_rel606.fasta trimmed-fastq/SRR2584863_1.trim.fastq trimmed-
SRR2584863.3    99      CP000819.1      2067177 60      138M    =       2067947 798     ATCAACAACACGCGTTTATTGGTCTGGCTGATCACCGCC
SRR2584863.3    147     CP000819.1      2067947 60      28M     =       2067177 -798    CAGCGGAAGATCAACAGAATCTTTATCC    IIIIIII
SRR2584863.5    99      CP000819.1      1231976 60      150M    =       1232448 498     TCCGTAAACATTTTAATGTCGTGCTCGAAAGAACGGTGC
SRR2584863.5    147     CP000819.1      1232448 60      26M     =       1231976 -498    GCAATTGATGACGGTAATGGATACAC      GHIGHHI
SRR2584863.7    83      CP000819.1      2562378 60      64M     =       2561695 -747    CGGCACGTTAACCTGCTGTTTGATGAGTTTGAACGCTTC
SRR2584863.7    163     CP000819.1      2561695 60      40M     =       2562378 747     GGGCCATTCACCACGCAGCCGATAATCGAAACGTCCATC
SRR2584863.8    83      CP000819.1      4263438 60      101M    =       4263359 -180    AAGGTGTAGCGCCCCAGGTTACGCAGACGTTCGGCAATC
SRR2584863.8    163     CP000819.1      4263359 60      150M    =       4263438 180     CGCCACCACCACCAGAGAACCACAGGCCGAAAGAATACG
...
```

Step 4 – Sort BAM file by coordinates

If we use `less` to inspect the fastq file of SRR2584863, it looks like:



```
@SRR2584863.3 HWI-ST957:244:H73TDADXX:1:1101:9337:2248/1
ATCAACAACACGCGTTTATTGGTCTGGCTGATCACCGCCGCCAGGTTGGCGCAGACAAAGGTTTTACCGATTGACGGGCTAACCCCGGTCATCATCAACACATTGTTCTGCGCCTGCATC
ATCGCGAAGTGCAGGCTG
+
C@CFFFFFHHHGHIIJJJJJJIIJIIJJJIIJJJJJGGGGGBGHHHFFBEDDDBBDDDDDD:>CBCCC>5;B?CDBBDBBBCCA?BD5<<@AAACCDCDCDDCDDDCDDCDBBDDBD<@>
@@CCBDB<@BACCC<38?
@SRR2584863.5 HWI-ST957:244:H73TDADXX:1:1101:16354:2243/1
TCCGTAAACATTTTAATGTCGTGCTCGAAAGAACGGTGCGTGAACTGCGCGGCGAACCCTGTTTGCAACTGGAAGAGTTTGCACCGACGAAGCAGGAAATTATCTGTTCCCGCTCGTTTG
GTGAACGCATCACGGATTATCCGTCGATGC
+
@@CFBADBFHFHHFIJIGGGGFGHIIIJJJEHIIIGHIJJGIIJGGIIIJJHFDDDBDDDDCDCCCCCDDCCCDDDDACCCDDCDDDDDBD?>CC@DDDDB@ACDCACCCDDBBBB<@D?
@BBCCDDDDBBCDB>B>@CDDDDBDDDDC@
@SRR2584863.7 HWI-ST957:244:H73TDADXX:1:1101:2503:2410/1
ATCCGTACCGACGATGGACGCGGCAGAAGCGTTCAAACTCATCAAACAGCAGGTTAACGTGCCG
+
@@@DDDDDFF@FFB8C1CG?<GBFBFC;==@CFDBDDB>B>@BCCCBB3;;?BBBAB:<<<?B;
@SRR2584863.8 HWI-ST957:244:H73TDADXX:1:1101:3127:2273/1
GGCTATGACGGCTTAATTTACTCGCTGGGCTTCCTGGTGGGCTGGCCGATCATTTTGTTCCTGATTGCCGAACGTCTGCGTAACCTGGGGCGCTACACCTT
+
@CBFFFFFHHHHHJJJJJIJJGIIIGIJEHIGIGIJI9DGEEDHBGHHG:BDFFFDFFCECECACCCCC>@@@?8<?<@@5<<@:4?BD5<55559>?3AC
```

Step 4 – Sort BAM file by coordinates

Here, we will sort the BAM file, so the sequences are in order of the position they were mapped to the genome from left to right.

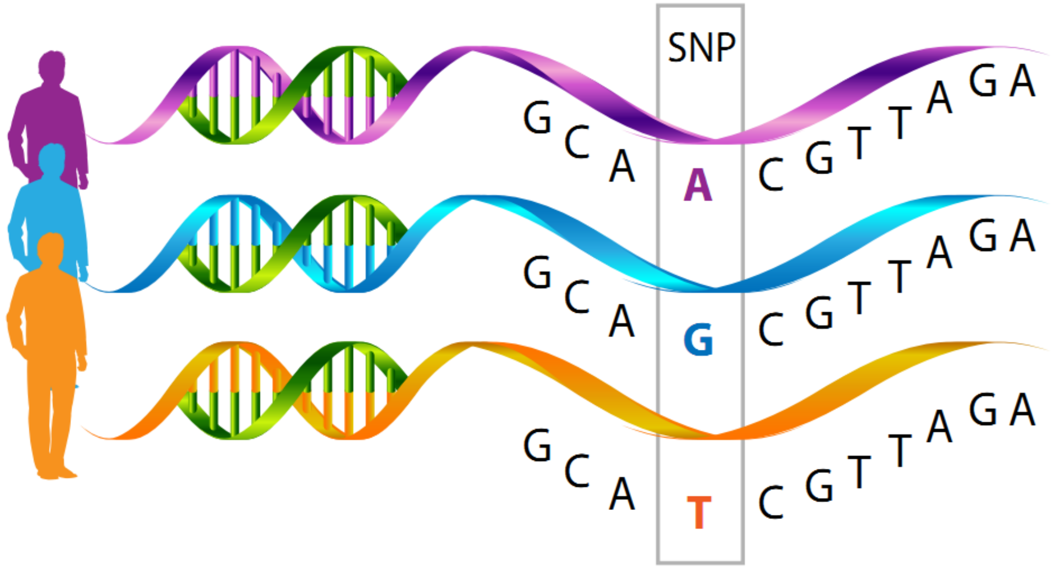
Because many downstream analysis requires the BAM file to be sorted by coordinates including variant calling.

```
samtools sort -o [aligned.sorted.bam] [aligned.bam]
```

Sort sample SRR2584863.

Step 5 – Variant Calling

A **variant call** is a conclusion that there is a nucleotide difference compared to the reference genome at a given position, often referred to as a **Single Nucleotide Polymorphism** (SNP).



Step 5 – Variant Calling

Popular variant calling tools:

- For SNPs & Indels – GATK HaplotypeCaller, bcftools, FreeBayes
- For Structural Variants – Manta, Delly, LUMPY
- For Copy Number Variants – CNVkit, GATK gCNV
- For Cancer Variants – Mutect2, Strelka2, VarScan2

We will use bcftools here.

First, we need to calculate the read coverage of positions in the genome:

```
bcftools mpileup -O v -o [sample_raw.vcf] -f [reg_genome] [aligned.sorted.bam]
```

Step 5 – Variant Calling

The result of read coverage calculation looks like:

```
##INFO=<ID=MQ0F,Number=1,Type=Float,Description="Fraction of MQ0 reads (smaller is better)">
##INFO=<ID=I16,Number=16,Type=Float,Description="Auxiliary tag used for calling, see description of bcf_callret1_t in bam2bcf.h
##INFO=<ID=QS,Number=R,Type=Float,Description="Auxiliary tag used for calling">
##FORMAT=<ID=PL,Number=G,Type=Integer,Description="List of Phred-scaled genotype likelihoods">
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  bam/SRR2584863.sorted.aligned.bam
CP000819.1      1       .       A       <*>     0       .       DP=27;I16=1,18,0,0,676,24074,0,0,1140,68400,0,0,0,0,0,0;QS=1,0;
CP000819.1      2       .       G       <*>     0       .       DP=27;I16=2,18,0,0,745,28387,0,0,1200,72000,0,0,26,68,0,0;QS=1,
CP000819.1      3       .       C       <*>     0       .       DP=27;I16=2,18,0,0,736,27594,0,0,1200,72000,0,0,46,140,0,0;QS=1
CP000819.1      4       .       T       <*>     0       .       DP=27;I16=2,18,0,0,741,27889,0,0,1200,72000,0,0,66,252,0,0;QS=1
CP000819.1      5       .       T       <*>     0       .       DP=27;I16=2,18,0,0,754,28670,0,0,1200,72000,0,0,86,404,0,0;QS=1
CP000819.1      6       .       T       <*>     0       .       DP=27;I16=2,18,0,0,778,30440,0,0,1200,72000,0,0,106,596,0,0;QS=
CP000819.1      7       .       T       <*>     0       .       DP=27;I16=2,18,0,0,763,29341,0,0,1200,72000,0,0,126,828,0,0;QS=
CP000819.1      8       .       C       <*>     0       .       DP=27;I16=2,18,0,0,786,30962,0,0,1200,72000,0,0,146,1100,0,0;QS
```

Each line contains information about a position in the genome.

Step 5 – Variant Calling

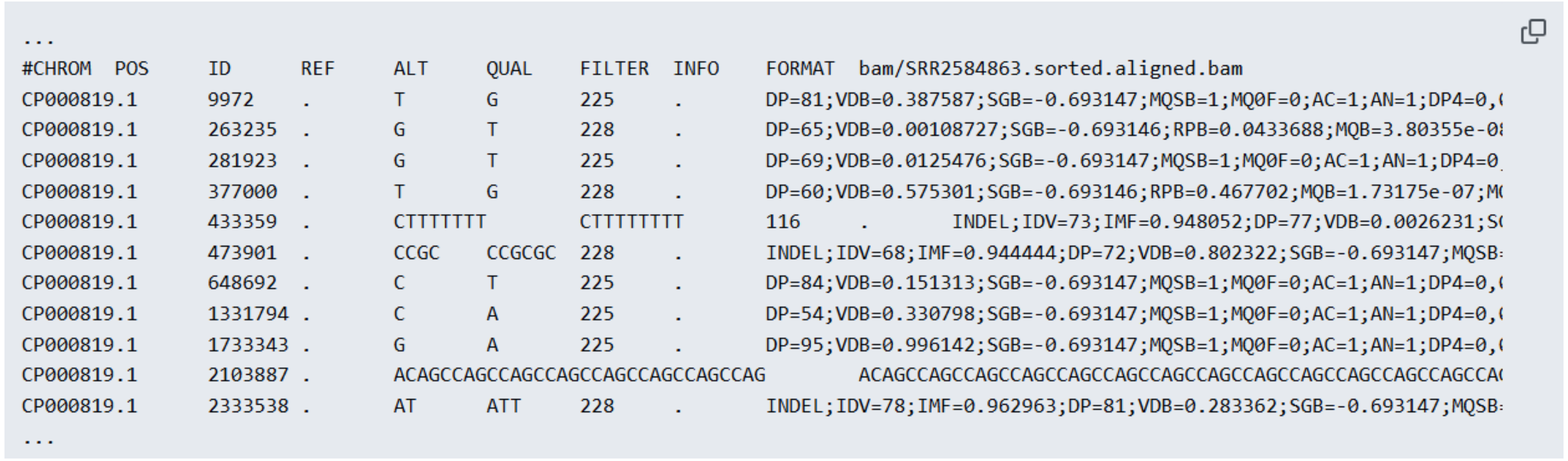
The above result contains everything on every position of the genome. Next, we can detect the SNPs.

```
bcftools call --ploidy 1 -m -v -o [sample_variants.vcf] [sample_raw.bcf]
```

- --ploidy: specify number of ploidy
- -m: allows for multiallelic and rare-variant calling
- -v: to output variant sites only
- -o: specify where to write the output file

Step 5 – Variant Calling

The result of variant call, only variant sites left.



```
...
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  bam/SRR2584863.sorted.aligned.bam
CP000819.1      9972    .       T       G       225     .       DP=81;VDB=0.387587;SGB=-0.693147;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,
CP000819.1      263235  .       G       T       228     .       DP=65;VDB=0.00108727;SGB=-0.693146;RPB=0.0433688;MQB=3.80355e-0
CP000819.1      281923  .       G       T       225     .       DP=69;VDB=0.0125476;SGB=-0.693147;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0
CP000819.1      377000  .       T       G       228     .       DP=60;VDB=0.575301;SGB=-0.693146;RPB=0.467702;MQB=1.73175e-07;M
CP000819.1      433359  .       CTTTTTTT        CTTTTTTTT       116     .       INDEL;IDV=73;IMF=0.948052;DP=77;VDB=0.0026231;S
CP000819.1      473901  .       CCGC    CCGCGC  228     .       INDEL;IDV=68;IMF=0.944444;DP=72;VDB=0.802322;SGB=-0.693147;MQSB
CP000819.1      648692  .       C       T       225     .       DP=84;VDB=0.151313;SGB=-0.693147;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,
CP000819.1      1331794 .       C       A       225     .       DP=54;VDB=0.330798;SGB=-0.693147;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,
CP000819.1      1733343 .       G       A       225     .       DP=95;VDB=0.996142;SGB=-0.693147;MQSB=1;MQ0F=0;AC=1;AN=1;DP4=0,
CP000819.1      2103887 .       ACAGCCAGCCAGCCAGCCAGCCAGCCAGCCAG        ACAGCCAGCCAGCCAGCCAGCCAGCCAGCCAGCCAGCCAGCCAGCCAGCCAGCCA
CP000819.1      2333538 .       AT      ATT     228     .       INDEL;IDV=78;IMF=0.962963;DP=81;VDB=0.283362;SGB=-0.693147;MQSB
...
```

Step 5 – Variant Calling

The above result gives us all the variant sites in our sample, but not all sites have high quality score. In this step, we will filter out low quality variants:

```
vcfutils.pl varFilter [sample_variants.vcf] > [sample_final_variants.vcf]
```

After receiving the final variants result, we can count the lines to see how many variants we have in this sample.

`grep –v ‘^#’ SRR2584863_final_variants.vcf | wc -l`

What is the result?

THANK YOU

Contact Us

Jiajia Li
ANU Biological Data Science Institute

RN Robertson Building, 46 Sullivan's Creek Rd
Canberra ACT 2600

E jiajia.li1@anu.edu.au
W https://bdsi.anu.edu.au/

Australian
National
University