Project 2’ s report

Narges Ghanei

610399155

Data preparation:

```
bio_narges_ghanei@ibbadmin-X10DAi:~$ fastq-dump --split-files SRR6191645.sra
```

```
bio_narges_ghanei@ibbadmin-X10DAi:~$ fastq-dump --split-files SRR6191646.sra
```

This will create 4 files called SRR6191645_1.fastq.gz, SRR6191645_2.fastq.gz, SRR6191646_1.fastq.gz SRR6191646_2.fastq.gz in the same directory as the SRA file.

Part a- Quality control and trimming

assess the read qualities using the FastQC software:

```
(bowtie2) bio_narges_ghanei@ibbadmin-X10DAi:~/project2/data$ fastqc -o /share_large/intro2bioinformatics/home/bio_narges_ghanei/project2/data /share_large/intro2bioinformatics/data/fastq/SRR6191645_1.fastq.gz
application/gzip
Started analysis of SRR6191645_1.fastq.gz
Approx 5% complete for SRR6191645_1.fastq.gz
Approx 10% complete for SRR6191645_1.fastq.gz
Approx 15% complete for SRR6191645_1.fastq.gz
Approx 20% complete for SRR6191645_1.fastq.gz
Approx 25% complete for SRR6191645_1.fastq.gz
Approx 30% complete for SRR6191645_1.fastq.gz
Approx 35% complete for SRR6191645_1.fastq.gz
Approx 40% complete for SRR6191645_1.fastq.gz
Approx 45% complete for SRR6191645_1.fastq.gz
Approx 50% complete for SRR6191645_1.fastq.gz
Approx 55% complete for SRR6191645_1.fastq.gz
Approx 60% complete for SRR6191645_1.fastq.gz
Approx 65% complete for SRR6191645_1.fastq.gz
Approx 70% complete for SRR6191645_1.fastq.gz
Approx 75% complete for SRR6191645_1.fastq.gz
Approx 80% complete for SRR6191645_1.fastq.gz
Approx 85% complete for SRR6191645_1.fastq.gz
Approx 90% complete for SRR6191645_1.fastq.gz
Approx 95% complete for SRR6191645_1.fastq.gz
Analysis complete for SRR6191645_1.fastq.gz
```

```
(bowtie2) bio_narges_ghanei@ibbadmin-X10DAi:~/project2/data$ fastqc -o /share_large/intro2bioinformatics/home/bio_narges_ghanei/project2/data /share_large/intro2bioinformatics/data/fastq/SRR6191645_2.fastq.gz
application/gzip
Started analysis of SRR6191645_2.fastq.gz
Approx 5% complete for SRR6191645_2.fastq.gz
Approx 10% complete for SRR6191645_2.fastq.gz
Approx 15% complete for SRR6191645_2.fastq.gz
Approx 20% complete for SRR6191645_2.fastq.gz
Approx 25% complete for SRR6191645_2.fastq.gz
Approx 30% complete for SRR6191645_2.fastq.gz
Approx 35% complete for SRR6191645_2.fastq.gz
Approx 40% complete for SRR6191645_2.fastq.gz
Approx 45% complete for SRR6191645_2.fastq.gz
Approx 50% complete for SRR6191645_2.fastq.gz
Approx 55% complete for SRR6191645_2.fastq.gz
Approx 60% complete for SRR6191645_2.fastq.gz
Approx 65% complete for SRR6191645_2.fastq.gz
Approx 70% complete for SRR6191645_2.fastq.gz
Approx 75% complete for SRR6191645_2.fastq.gz
Approx 80% complete for SRR6191645_2.fastq.gz
Approx 85% complete for SRR6191645_2.fastq.gz
Approx 90% complete for SRR6191645_2.fastq.gz
Approx 95% complete for SRR6191645_2.fastq.gz
Analysis complete for SRR6191645_2.fastq.gz
```

```
(bowtie2) bio_narges_ghanei@ibbadmin-X10DAi:~$ fastqc -o /share_large/intro2bioinformatics/home/bio_narges_ghanei/project2/data /share_large/intro2bioinformatics/data/fastq/SRR6191646_1.fastq.gz
application/gzip
Started analysis of SRR6191646_1.fastq.gz
Approx 5% complete for SRR6191646_1.fastq.gz
Approx 10% complete for SRR6191646_1.fastq.gz
Approx 15% complete for SRR6191646_1.fastq.gz
Approx 20% complete for SRR6191646_1.fastq.gz
Approx 25% complete for SRR6191646_1.fastq.gz
Approx 30% complete for SRR6191646_1.fastq.gz
Approx 35% complete for SRR6191646_1.fastq.gz
Approx 40% complete for SRR6191646_1.fastq.gz
Approx 45% complete for SRR6191646_1.fastq.gz
Approx 50% complete for SRR6191646_1.fastq.gz
Approx 55% complete for SRR6191646_1.fastq.gz
Approx 60% complete for SRR6191646_1.fastq.gz
Approx 65% complete for SRR6191646_1.fastq.gz
Approx 70% complete for SRR6191646_1.fastq.gz
Approx 75% complete for SRR6191646_1.fastq.gz
Approx 80% complete for SRR6191646_1.fastq.gz
Approx 85% complete for SRR6191646_1.fastq.gz
Approx 90% complete for SRR6191646_1.fastq.gz
Approx 95% complete for SRR6191646_1.fastq.gz
Analysis complete for SRR6191646_1.fastq.gz
```

```
(bowtie2) bio_narges_ghanei@ibbadmin-X10DAi:~$ fastqc -o /share_large/intro2bioinformatics/home/bio_narges_ghanei/project2/data /share_large/intro2bioinformatics/data/fastq/SRR6191646_2.fastq.gz
application/gzip
Started analysis of SRR6191646_2.fastq.gz
Approx 5% complete for SRR6191646_2.fastq.gz
Approx 10% complete for SRR6191646_2.fastq.gz
Approx 15% complete for SRR6191646_2.fastq.gz
Approx 20% complete for SRR6191646_2.fastq.gz
Approx 25% complete for SRR6191646_2.fastq.gz
Approx 30% complete for SRR6191646_2.fastq.gz
Approx 35% complete for SRR6191646_2.fastq.gz
Approx 40% complete for SRR6191646_2.fastq.gz
Approx 45% complete for SRR6191646_2.fastq.gz
Approx 50% complete for SRR6191646_2.fastq.gz
Approx 55% complete for SRR6191646_2.fastq.gz
Approx 60% complete for SRR6191646_2.fastq.gz
Approx 65% complete for SRR6191646_2.fastq.gz
Approx 70% complete for SRR6191646_2.fastq.gz
Approx 75% complete for SRR6191646_2.fastq.gz
Approx 80% complete for SRR6191646_2.fastq.gz
Approx 85% complete for SRR6191646_2.fastq.gz
Approx 90% complete for SRR6191646_2.fastq.gz
Approx 95% complete for SRR6191646_2.fastq.gz
Analysis complete for SRR6191646_2.fastq.gz
```

-o option to specify the output directory.

This will generate HTML reports for each fastq file. I analyzed the output files below.

SRR6191645_1_fastqc.html:

Basic Statistics

Measure	Value
Filename	SRR6191645_1.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	64135799
Total Bases	9.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	52

- The file name is SRR6191645_1.fastq, which suggests that it is the first file of a paired-end sequencing experiment. Paired-end sequencing is a method that generates two reads from each DNA fragment, one from each end.

- The file type is conventional base calls, which means that the file contains the raw nucleotide sequences and their corresponding quality scores. The quality score is a measure of how confident the sequencer is that it called the correct base. The higher the score, the lower the probability of an error.

- The encoding is Sanger / Illumina 1.9, which means that the quality scores are encoded using ASCII characters with an offset of 33. This is the most common encoding scheme for FASTQ files. The quality scores range from 0 to 93.

- The total sequences is 64,135,799, which means that the file contains more than 64 million reads. This is a large number of reads, which can indicate a high sequencing depth or a large genome size. The sequencing depth is the average number of times a nucleotide is sequenced. A higher depth can increase the accuracy and sensitivity of the analysis.

- The total bases is 9.6 Gbp, which means that the file contains more than 9.6 billion nucleotides. This is the product of the total sequences and the sequence length.

The sequence length is 150, which means that each read is 150 nucleotides long. This is a typical length for paired-end sequencing.

- The sequences flagged as poor quality is 0, which means that none of the reads have a quality score below a certain threshold.

- The %GC is 52, which means that the proportion of guanine and cytosine nucleotides in the reads is 52%. The GC content can affect the efficiency and accuracy of the sequencing and the analysis.

Some of the summary files are not ticked green. They are red or orange which means it should be modified to get a higher quality.

Now I analyzed other html files and here is the results to compare to each other:

Basic Statistics

Measure	Value
Filename	SRR6191645_1.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	64135799
Total Bases	9.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	52

Basic Statistics

Measure	Value
Filename	SRR6191645_2.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	64135799
Total Bases	9.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	52

Basic Statistics

Measure	Value
Filename	SRR6191646_1.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	57973670
Total Bases	8.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	53

Basic Statistics

Measure	Value
Filename	SRR6191646_2.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	57973670
Total Bases	8.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	53

The difference between them are just in total bases and %GC. For each SRA file they are the same. All of the files have broken adapter content that we try to fix them in the next part.

use the Trimmomatic software to improve the read qualities through the read trimming:

```
(bowtie) bio_narges_ghaneI@ibbadmin-X10DAi:~$ trimmomatic PE -phred33 -trimlog trim_log /share_large/intro2bioinformatics/data/fastq/SRR6191645_1.fastq /share_large/intro2bioinformatics/data/fastq/SRR6191645_2
.fastq output_R1_paired.fastq output_R1_unpaired.fastq output_R2_paired.fastq output_R2_unpaired.fastq ILLUMINACLIP:TruSeq3-PE.fa:2:30:10:2:True LEADING:3 TRAILING:3 SLIDINGWINDOW:4:30 MINLEN:80
TrimmomaticPE: Started with arguments:
 -phred33 -trimlog trim_log /share_large/intro2bioinformatics/data/fastq/SRR6191645_1.fastq /share_large/intro2bioinformatics/data/fastq/SRR6191645_2.fastq output_R1_paired.fastq output_R1_unpaired.fastq outpu
t_R2_paired.fastq output_R2_unpaired.fastq ILLUMINACLIP:TruSeq3-PE.fa:2:30:10:2:True LEADING:3 TRAILING:3 SLIDINGWINDOW:4:30 MINLEN:80
Using PrefixPair: 'TACACTCTTTCCCTACACGACGCTCTTCCGATCT' and 'GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT'
ILLUMINACLIP: Using 1 prefix pairs, 0 forward/reverse sequences, 0 forward only sequences, 0 reverse only sequences
Input Read Pairs: 64135799 Both Surviving: 33581670 (52.36%) Forward Only Surviving: 16260978 (25.35%) Reverse Only Surviving: 2414746 (3.77%) Dropped: 11878405 (18.52%)
TrimmomaticPE: Completed successfully
```

```
(bowtie) bio_narges_ghaneI@ibbadmin-X10DAi:~$ trimmomatic PE -phred33 -trimlog trim_log /share_large/intro2bioinformatics/data/fastq/SRR6191646_1.fastq /share_large/intro2bioinformatics/data/fastq/SRR6191646_2
.fastq output2_R1_paired.fastq output2_R1_unpaired.fastq output2_R2_paired.fastq output2_R2_unpaired.fastq ILLUMINACLIP:TruSeq3-PE.fa:2:30:10:2:True LEADING:3 TRAILING:3 SLIDINGWINDOW:4:30 MINLEN:80
TrimmomaticPE: Started with arguments:
 -phred33 -trimlog trim_log /share_large/intro2bioinformatics/data/fastq/SRR6191646_1.fastq /share_large/intro2bioinformatics/data/fastq/SRR6191646_2.fastq output2_R1_paired.fastq output2_R1_unpaired.fastq out
put2_R2_paired.fastq output2_R2_unpaired.fastq ILLUMINACLIP:TruSeq3-PE.fa:2:30:10:2:True LEADING:3 TRAILING:3 SLIDINGWINDOW:4:30 MINLEN:80
Using PrefixPair: 'TACACTCTTTCCCTACACGACGCTCTTCCGATCT' and 'GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT'
ILLUMINACLIP: Using 1 prefix pairs, 0 forward/reverse sequences, 0 forward only sequences, 0 reverse only sequences
Input Read Pairs: 57973670 Both Surviving: 35182854 (60.69%) Forward Only Surviving: 11162298 (19.25%) Reverse Only Surviving: 2229299 (3.85%) Dropped: 9399227 (16.21%)
TrimmomaticPE: Completed successfully
```

Command explanation:

trimmomatic PE -phred33 -trimlog trim_log input_R1.fastq.gz input_R2.fastq output_R1_paired.fastq output_R1_unpaired.fastq output_R2_paired.fastq output_R2_unpaired.fastq ILLUMINACLIP:TruSeq3-PE.fa:2:30:10:2:True LEADING:3 TRAILING:3 SLIDINGWINDOW:4:30 MINLEN:80 -> This command will trim and filter the reads according to criteria and output four files: two for the paired reads that survived the trimming, and two for the unpaired reads that were discarded.

- ILLUMINACLIP:TruSeq3-PE.fa:2:30:10:2:True -> This means that the adapter sequence must match with at least 30 quality score to be clipped. The ILLUMINACLIP option should have a higher quality threshold for adapter clipping. I set it to 30.
- SLIDINGWINDOW:4:30 -> This means that the average quality within the 4-base wide window must be at least 30 to keep the read.
- MINLEN:80 -> This means that the read must be at least 80 bases long after trimming to be retained.

Recheck the read qualities to make sure the problems are solved:

```
(bowtie) bio_narges_ghanei@ibbadmin-X10DAi:~$ fastqc -t 5 -o ./project2 output_R1_paired.fastq output_R2_paired.fastq
```

```
(bowtie) bio_narges_ghanei@ibbadmin-X10DAi:~$ fastqc -t 5 -o ./project2 output2_R1_paired.fastq output2_R2_paired.fastq
```

To compare the fastqc results before and after trimming, we need to run the fastqc command on the trimmed fastq files (output_R1_paired.fastq and output_R2_paired.fastq). The -o option to specify the output directory for the trimmed files. This will generate HTML reports for the trimmed fastq files in the directory.

Compare reads before and after trimming:

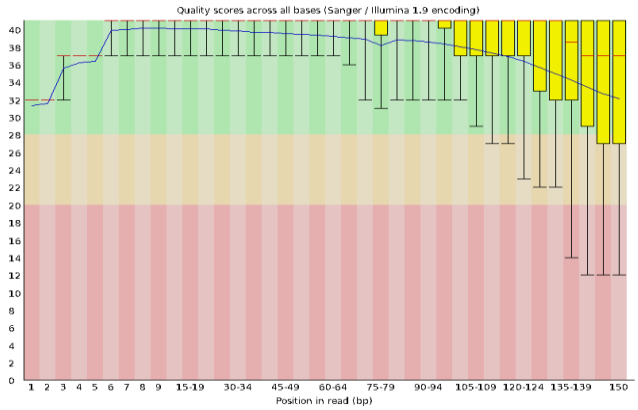
SRR6191645_1

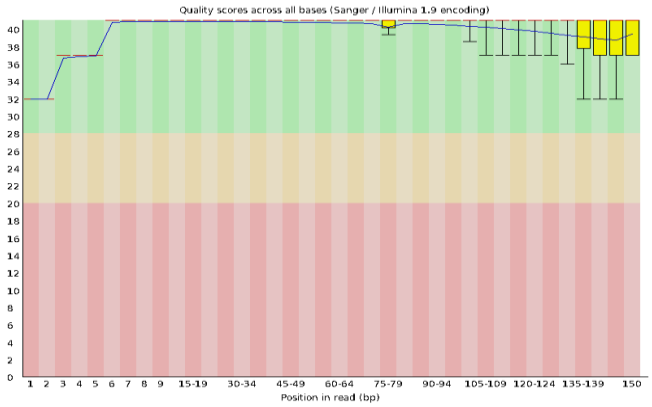
Basic Statistics

Measure	Value
Filename	SRR6191645_1.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	64135799
Total Bases	9.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	52

Basic Statistics

Measure	Value
Filename	output_R1_paired.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	33581670
Total Bases	4.6 Gbp
Sequences flagged as poor quality	0
Sequence length	80-150
%GC	53



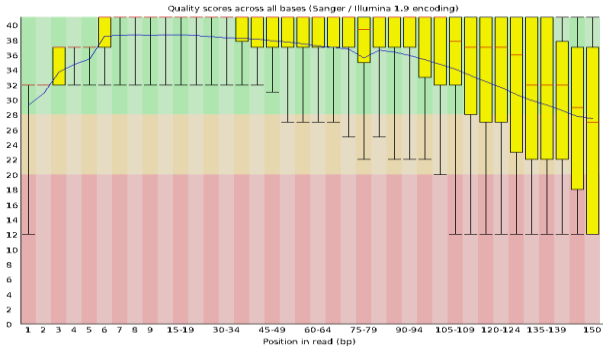


We can see that number of sequences are decreased and the quality of reads become extremely better. Olso adapter sequence that was red before, is now green.

SRR6191645_2

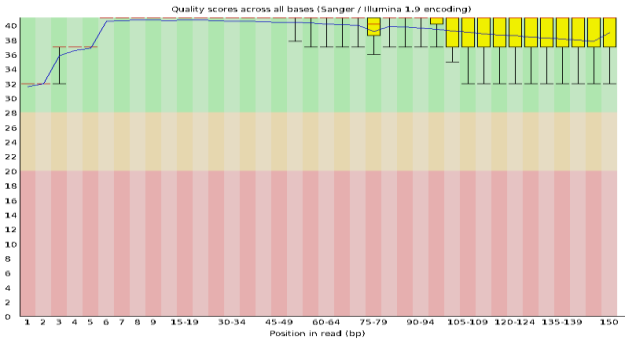
Basic Statistics

Measure	Value
Filename	SRR6191645_2.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	64135799
Total Bases	9.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	52



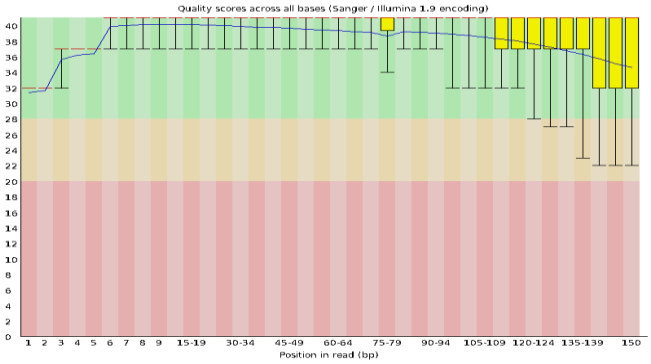
Basic Statistics

Measure	Value
Filename	output_R2_paired.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	33581670
Total Bases	4.1 Gbp
Sequences flagged as poor quality	0
Sequence length	80-150
%GC	53

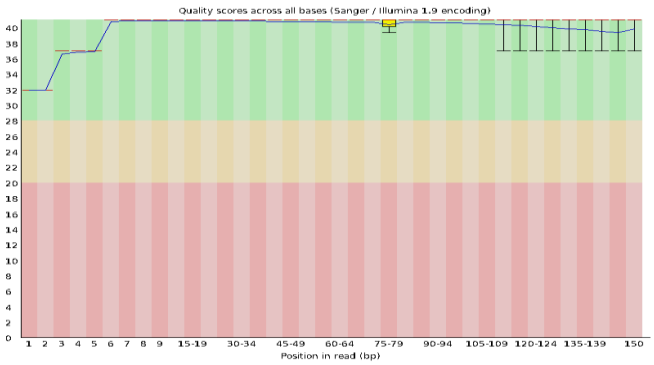


SRR6191646_1

Measure	Value
Filename	SRR6191646_1.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	57973670
Total Bases	8.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	53



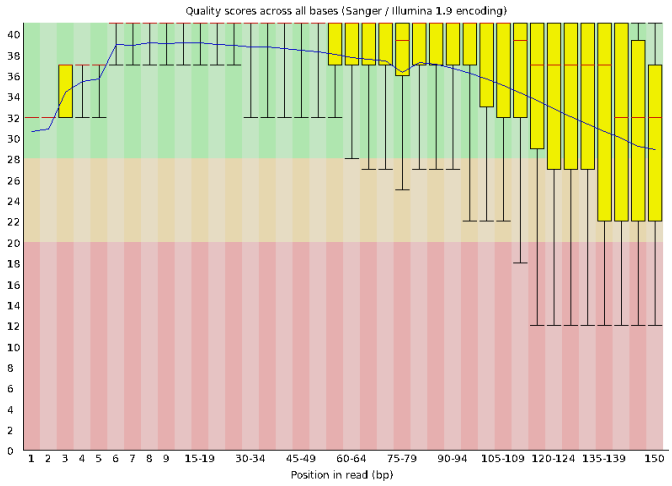
Measure	Value
Filename	output2_R1_paired.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	35182854
Total Bases	5 Gbp
Sequences flagged as poor quality	0
Sequence length	80-150
%GC	53

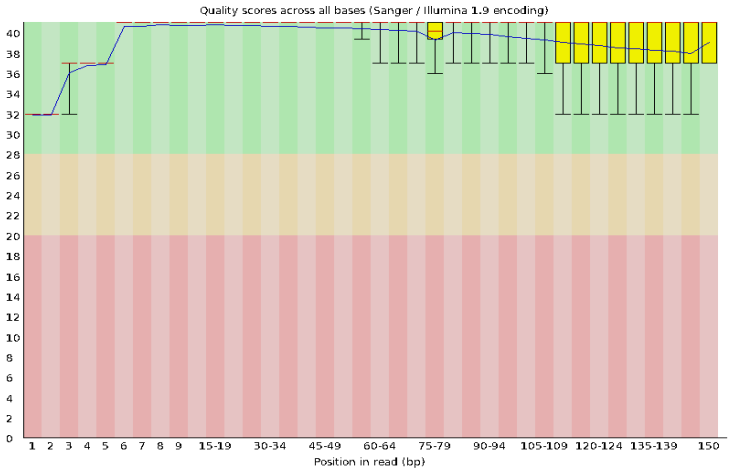


SRR6191646_2

Measure	Value
Filename	SRR6191646_2.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	57973670
Total Bases	8.6 Gbp
Sequences flagged as poor quality	0
Sequence length	150
%GC	53

Measure	Value
Filename	output2_R2_paired.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	35182854
Total Bases	4.4 Gbp
Sequences flagged as poor quality	0
Sequence length	80-150
%GC	53





Questions:

1. **What is the average number of reads across samples before and after the read trimming?**

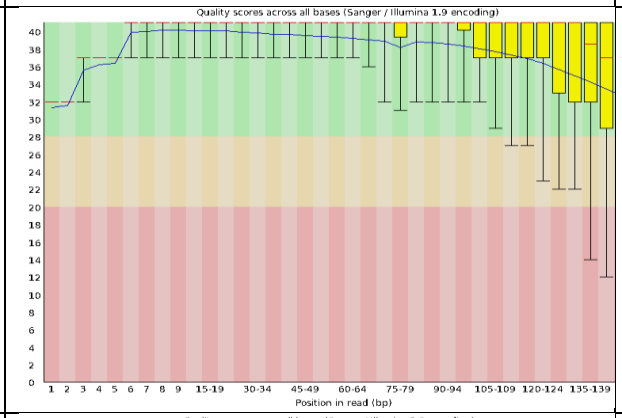
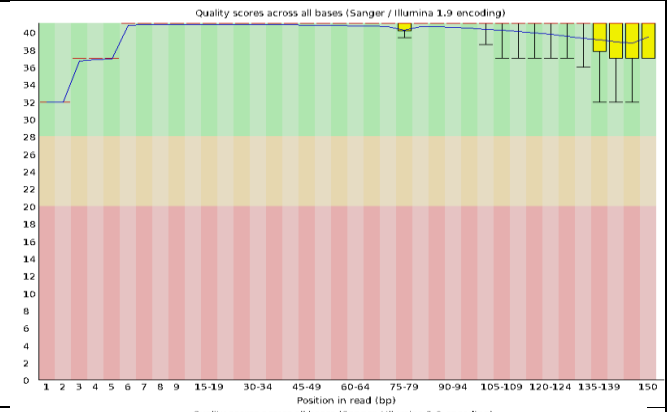
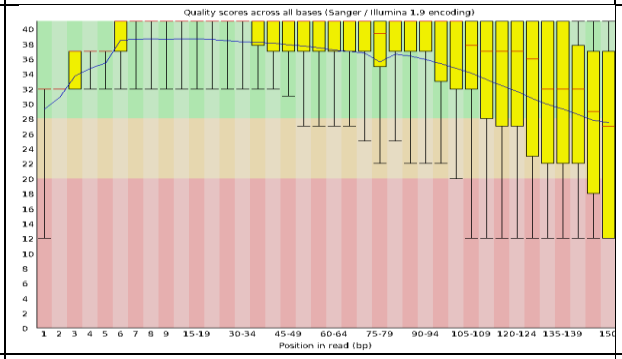
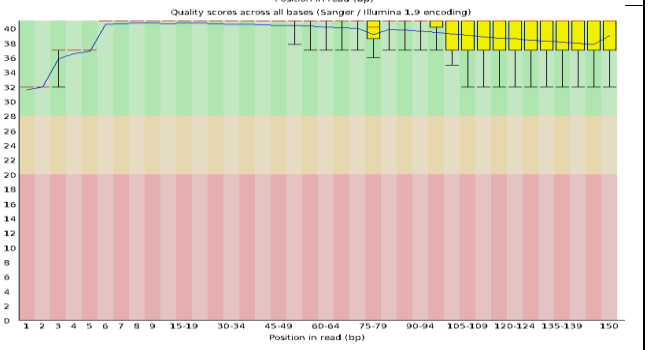
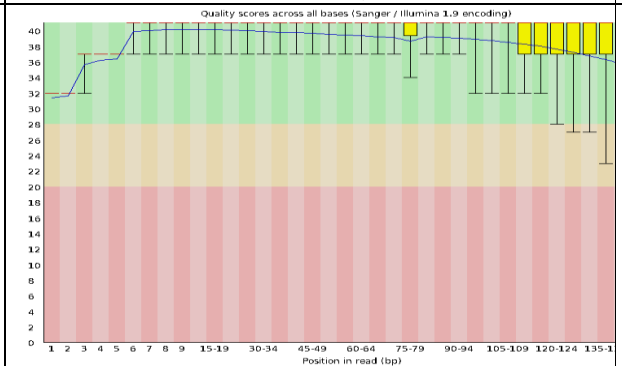
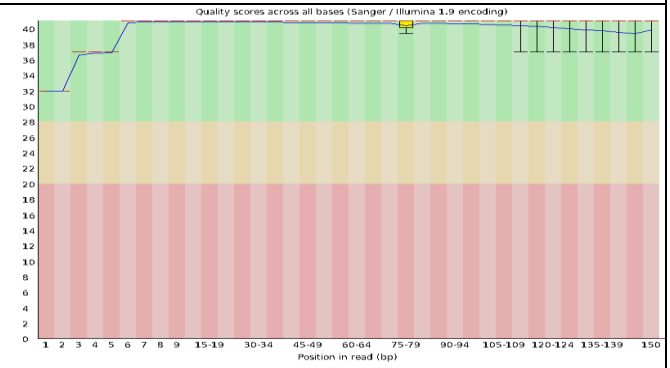
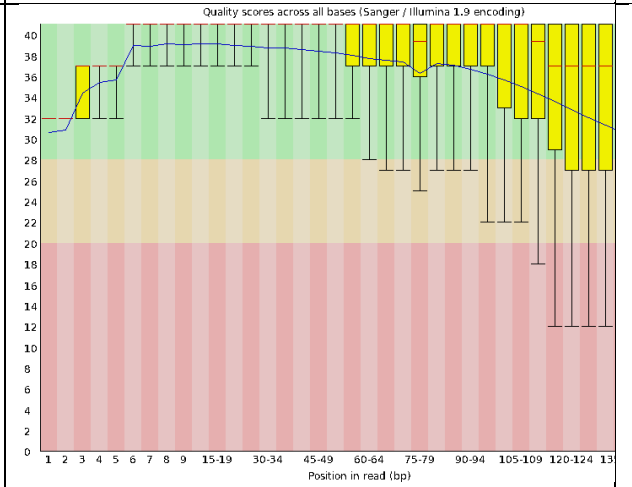
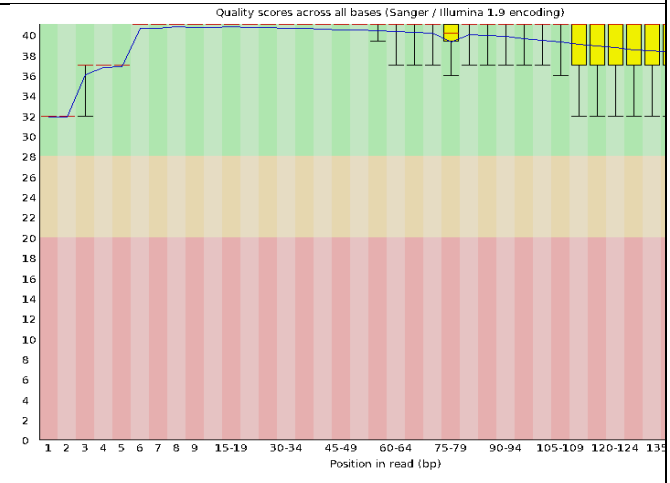
	Number of reads before trimming	Number of reads after trimming
SRR6191645	64135799	33581670
SRR6191646	57973670	35182854

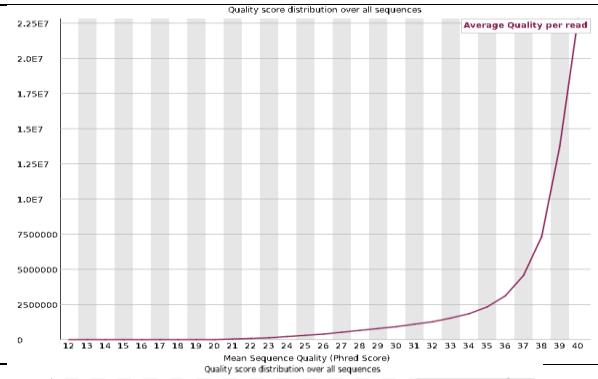
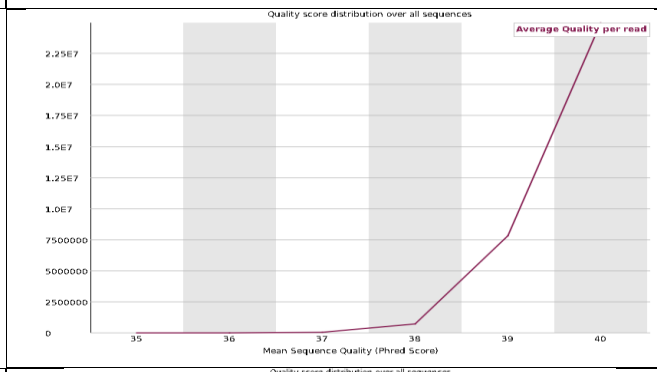
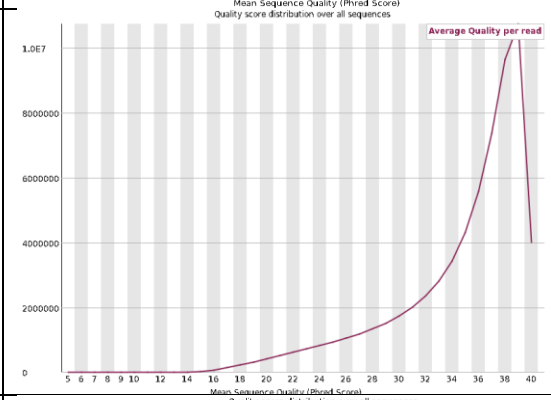
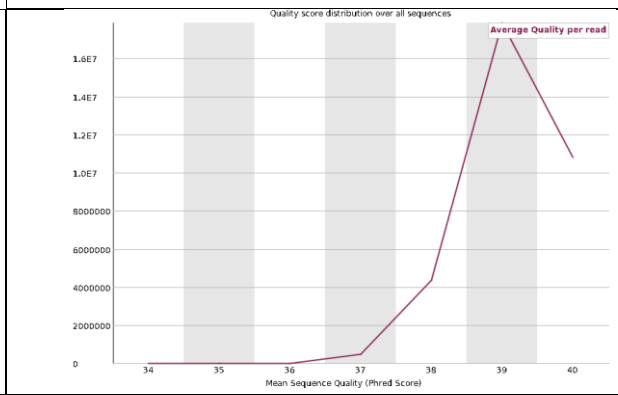
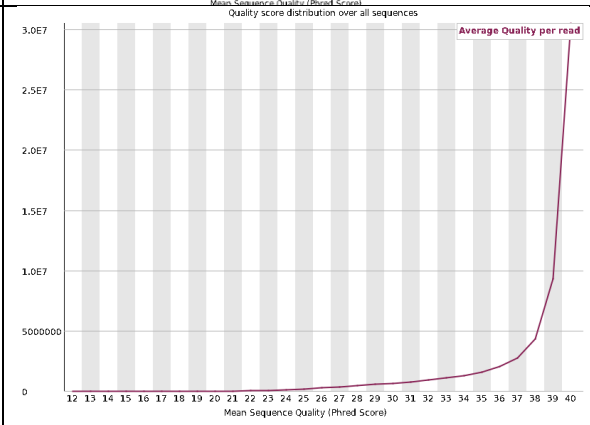
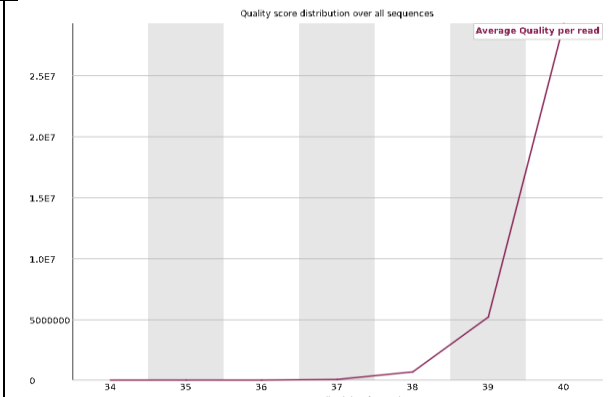
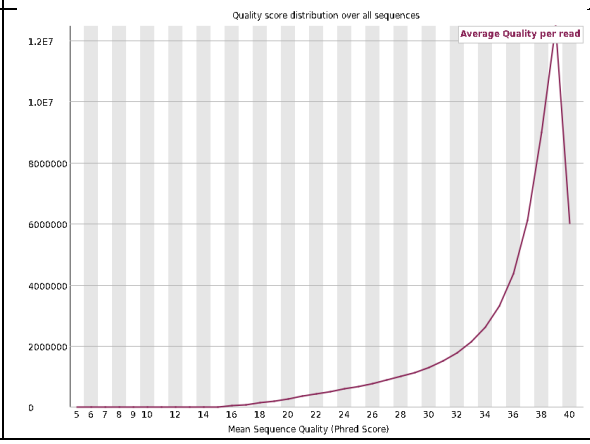
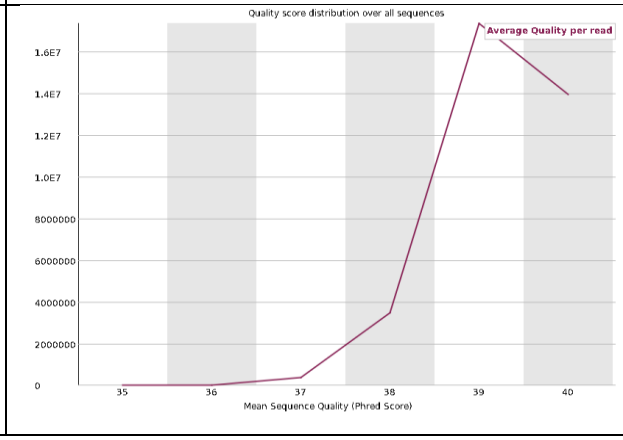
2. **Compare the read length averages in different samples before and after the read trimming?**

	Read length average before trimming	Read length average after trimming
SRR6191645	150	80-150
SRR6191646	150	80-150

3. Compare the read quality distributions over all sequences before and after the read trimming.

as I mentioned above, the quality plots are as below:

	Per base sequence quality before trimmimg	Per base sequence quality after trimmimg
SRR6191645_1		
SRR6191645_2		
SRR6191646_1		
SRR6191646_2		

	Per sequence quality scores before trimmimg	Per sequence quality scores after trimmimg
SRR6191645_1		
SRR6191645_2		
SRR6191646_1		
SRR6191646_2		

After trimming, the distribution of quality scores change. Trimming removes low-quality bases or adapters from the ends of reads, which can improve the overall quality of sequences. As a result, we see a shift towards higher quality scores in the Per sequence quality scores plot after trimming. The distribution becomed narrower with fewer low-quality sequences.

After trimming, the Per base sequence quality plot changes. Trimming removes low-quality bases or adapter sequences from the ends of reads, which can improve the quality of bases in these regions. As a result, we observe an increase in the average quality scores at the beginning and end of reads after trimming. Additionally, regions with adapter contamination or other artifacts are removed entirely, leading to smoother and more consistent quality profiles along the length of the reads.

4. **What does the Adaptor Content warning indicate?**

The "Adaptor Content" warning in a FastQC report indicates the presence of adapter sequences in the sequencing reads. Adapter sequences are short oligonucleotides that are used during the library preparation step of sequencing to ligate the DNA fragments to the sequencing adapters. These adapters contain sequences that are necessary for the sequencing process, such as priming sites for PCR amplification or binding sites for the sequencing platform. The "Adaptor Content" warning suggests that adapter contamination may be present in the sequencing reads. Which can have several reasons such as unintended sequencing of adapter fragments, decreased data quality, and potential issues with analysis. To address the "Adaptor Content" warning, it is necessary to perform adapter trimming as part of the data preprocessing step. Adapter trimming involves removing the adapter sequences from the sequencing reads to eliminate contamination and improve data quality. By addressing adapter contamination and performing appropriate quality control measures, we can ensure the accuracy and reliability of sequencing data for subsequent analysis and interpretation.

5. **Why do we first remove the Adapter sequences for the reads and then the low-quality bases?**

We do this for several reasons:

Prevent Misinterpretation: Adapter sequences are introduced during library preparation and are not part of the original biological sequence. If adapters are not removed before quality trimming, the presence of adapter sequences can lead to misinterpretation of the data.

Reduce Noise in Quality Assessment: Adapter sequences typically have lower quality scores compared to the biological sequences. If adapter sequences are not removed before quality trimming, they contribute to the overall quality profile of the reads, potentially masking the true quality distribution of the biological sequences.

Improve Efficiency of Quality Trimming: Adapter trimming typically involves identifying adapter sequences at the ends of reads and removing them. Once adapter sequences are removed, quality trimming algorithms can focus on identifying and removing low-quality bases within the remaining portion of the reads.

Minimize Loss of Information: Quality trimming is a more aggressive process compared to adapter trimming, as it involves discarding bases based on their quality scores. By removing adapter sequences first, researchers can ensure that the maximum amount of information from the biological sequences is retained before quality trimming is performed. This helps minimize the loss of potentially valuable sequencing data during the preprocessing steps.

It ensures accurate downstream analysis and minimizes the risk of misinterpretation due to adapter contamination.

6. What does the quality of bases mean, and how is it obtained?

The quality of bases in sequencing data refers to the confidence level associated with the accuracy of each nucleotide call (A, C, G, T) at a particular position in a sequencing read. It represents the likelihood that the called base is correct, and it's crucial for assessing the reliability of the sequencing data.

Quality scores for bases are typically represented using the Phred scale, which is a logarithmic scale where each integer represents a tenfold difference in the error probability. The Phred quality score (Q) is calculated using the formula:

$$Q = -10 \times \log_{10}(p)$$

Where:Q is the Phred quality score. P is the probability that the base call is incorrect.

The quality score (Q) ranges from 0 to 40 in most modern sequencing platforms, with higher scores indicating higher confidence in the base call:

- A quality score of 30 corresponds to a 1 in 1000 chance of error (base call accuracy of 99.9%).

- A quality score of 40 corresponds to a 1 in 10,000 chance of error (base call accuracy of 99.99%).

Base calling algorithms analyze raw fluorescence intensity data generated during the sequencing process and assign a quality score to each nucleotide call based on the signal strength and other parameters.

Quality control tools such as FastQC analyze these quality scores to generate quality metrics and assess the overall quality of the sequencing data. These metrics helps to identify potential issues, such as low-quality regions, adapter contamination, or sequence biases, which may impact downstream analysis and interpretation of the data.

Part b- Read mapping

Use the hisat2-build software to index the Homo sapiens (GRCh38) reference genome:

```
(bowtie) bio_narges_ghanei@ibbadmin-X10DAi:~$ gunzip  Homo_sapiens.GRCh38.dna.toplevel.fa.gz
```

Unzipping the reference genome.

```
(bowtie) bio_narges_ghanei@ibbadmin-X10DAi:~$ hisat2-build -p 6 -f Homo_sapiens.GRCh38.dna.toplevel.fa GRCh38
```

Running the hisat2-build software to index the reference genome. -p option specifies the number of threads and the -f option indicates that the input file is in FASTA format. This will create 8 index files with the prefix GRCh38 in the current directory.

```
'=1.10'                 GRCh38.4.ht2l
 counting1.txt          GRCh38.5.ht2l
 counting2.txt          GRCh38.6.ht2l
 counts2.txt            GRCh38.7.ht2l
 counts_merge.Rdata     GRCh38.8.ht2l
 counts_merge.txt       Homo_sapiens.GR
 counts.txt             Homo_sapiens.GR
 GRCh38.1.ht2l          htseq-merge_all
 GRCh38.2.ht2l          miniconda3
 GRCh38.3.ht2l          output2_R1_pair
(base) bio_narges_ghanei@ibbadmin-X10
```

GRCh38 files are the indexed fils.

Map the reads to the reference genome using the HISAT2 software:

```
(bowtie) bio_narges_ghanei@ibbadmin-X10DAi:~$ hisat2 -p 7 -x GRCh38 -1 output_R1_paired.fastq -2 output_R2_paired.fastq -S sample.sam --summary-file sample_hisat2_summary.txt
```

```
(base) bio_narges_ghanei@ibbadmin-X10DAi:~$ cat  sample_hisat2_summary.txt
33581670 reads; of these:
  33581670 (100.00%) were paired; of these:
    1883114 (5.61%) aligned concordantly 0 times
    25565117 (76.13%) aligned concordantly exactly 1 time
    6133439 (18.26%) aligned concordantly >1 times
    ----
    1883114 pairs aligned concordantly 0 times; of these:
      117117 (6.22%) aligned discordantly 1 time
    ----
    1765997 pairs aligned 0 times concordantly or discordantly; of these:
      3531994 mates make up the pairs; of these:
        2365681 (66.98%) aligned 0 times
        838373 (23.74%) aligned exactly 1 time
        327940 (9.28%) aligned >1 times
96.48% overall alignment rate
```

```
(bowtie) bio_narges_ghanei@ibbadmin-X10DAi:~$  hisat2 -p 7 -x GRCh38 -1 output2_R1_paired.fastq -2 output2_R2_paired.fas
tq -S sample2.sam --summary-file sample2_hisat2_summary.txt
35182854 reads; of these:
  35182854 (100.00%) were paired; of these:
    1570485 (4.46%) aligned concordantly 0 times
    26720852 (75.95%) aligned concordantly exactly 1 time
    6891517 (19.59%) aligned concordantly >1 times
    ----
    1570485 pairs aligned concordantly 0 times; of these:
      100810 (6.42%) aligned discordantly 1 time
    ----
    1469675 pairs aligned 0 times concordantly or discordantly; of these:
      2939350 mates make up the pairs; of these:
        1779789 (60.55%) aligned 0 times
        791576 (26.93%) aligned exactly 1 time
        367985 (12.52%) aligned >1 times
97.47% overall alignment rate
```

Running the HISAT2 software to map the reads to the reference genome. I used the hisat2 command with the -x option to specify the index prefix, the -1 and -2 options to specify the paired-end fastq files, the -S option to specify the output SAM file, and the --summary-file option to generate a summary report.

This will map the trimmed reads to the reference genome and output a SAM file and a summary report for each sample.

The first file has 96.48% and the second file has 97.47% alignment rate. Other details are attached to my project file.

Use the Samtools software to convert the HISAT2 SAM files to BAM files:

```
bio_narges_ghanei@ibbadmin-X10DAi:~$ samtools view -bS sample.sam > sample.bam
```

Running the Samtools software to convert the SAM files to BAM files. The -b option outputs BAM format, the -S option indicates that the input file is in SAM format, and the -o option specifies the output file name. This will convert the SAM file to a BAM file for each sample.

```
output2_R1_unpaired.fastq    sample2.sam
output2_R2_paired.fastq      sample.bam
output2_R2_unpaired.fastq    sample_hisat2_summary.txt
output_R1_paired.fastq       sample.sam
output_R1_unpaired.fastq     snap
output_R2_paired.fastq       sorted_sample2.bam
output_R2_unpaired.fastq     sorted_sample.bam
project2                     trim_log
sample2.bam                  TruSeq3-PE.fa
sample2_hisat2_summary.txt
```

Sample.bam and sample2.bam are the output files for 2 sam files.

Questions:

1- What is the difference between SAM and BAM files?

SAM (Sequence Alignment/Map) and BAM (Binary Alignment/Map) files are both formats used to represent sequence alignment data, particularly for mapping sequencing reads to a reference genome. They differ in terms of their file structure and the way data is stored. While both SAM and BAM files represent sequence alignment data, BAM files offer advantages in terms of file size, efficiency, and speed of processing due to their binary format and built-in indexing capabilities. However, SAM files remain useful for situations where human readability and compatibility with text-based tools are important.

Format:

- SAM: SAM files are plain text files that store sequence alignment data in a human-readable format. Each line in a SAM file represents a single alignment record and contains various fields, including read name, reference sequence name, alignment position, quality scores, and optional alignment flags and tags.

- BAM: BAM files are binary versions of SAM files. They use a more compact binary format to store alignment data, which reduces file size and enables faster processing and analysis. BAM files are compressed and indexed, making them more efficient for storage and manipulation compared to SAM files.

Size and Efficiency:

- SAM: SAM files are larger in size compared to BAM files because they store data in plain text format. They are less efficient for storage and processing, especially for large-scale sequencing experiments with millions of reads.

- BAM: BAM files are binary-encoded and compressed, resulting in smaller file sizes compared to SAM files. They are more efficient for storage and manipulation, making them the preferred choice for handling large sequencing datasets.

Speed of Processing:

- SAM: SAM files are slower to process compared to BAM files because they require parsing and interpreting text-based data. Reading and writing SAM files can be slower, especially for large datasets.

- BAM: BAM files are faster to process because they are binary-encoded and compressed. They can be directly accessed and manipulated at the binary level, leading to improved performance in alignment algorithms and downstream analysis pipelines.

Indexing:

- SAM: SAM files do not have built-in indexing capabilities. To efficiently retrieve specific regions of interest or perform random access operations, external index files are required.

- BAM: BAM files include built-in indexing, which allows for efficient random access to specific regions of the alignment data. This makes it easier to query and retrieve alignment information from BAM files without the need for external index files.

2- What is the purpose of indexing the genome?

Genome indexing plays a crucial role in genomics and bioinformatics by facilitating efficient data retrieval, analysis, and interpretation of genomic information. It serves as a foundational component for a wide range of genomic studies, from basic research on genome structure and function to translational and clinical applications in personalized medicine and diagnostics.

Indexing the genome serves several important purposes in genomics and bioinformatics analyses. The primary goals of genome indexing include:

Efficient Searching and Retrieval: Indexing allows for rapid searching and retrieval of specific genomic regions or features of interest. With a properly indexed genome, researchers can quickly locate genes, regulatory elements, variants, or other genomic features relevant to their analysis.

Accelerated Sequence Alignment: Genome indexing is essential for accelerating sequence alignment algorithms, which are used to map sequencing reads to a reference genome.

Enhanced Variant Calling and Analysis: Indexing facilitates variant calling and analysis by enabling efficient comparison of sequencing reads with the reference genome.

Facilitated Genome Annotation: Indexing supports genome annotation efforts by enabling the rapid retrieval of gene annotations, transcript structures, regulatory elements, and other genomic features stored in annotation databases. This facilitates gene expression analysis, functional annotation, and interpretation of genomic data.

Streamlined Data Retrieval and Analysis: Genome indexing streamlines data retrieval and analysis workflows by providing fast access to genomic sequences and annotations. It enables researchers to efficiently query, retrieve, and manipulate genomic data for a wide range of applications, including gene expression profiling, comparative genomics, and evolutionary analysis.

Support for High-Throughput Sequencing: It enables scalable and efficient processing of sequencing data across diverse research domains and applications.

3- Report mapping percentages of all samples in a table. Please explain why a low percentage of reads cannot be mapped.

sample	alignment rate
SRR6191645	96.48%
SRR6191646	97.47%

A low percentage of unmapped reads after alignment with HISAT2 or similar tools can result from various factors related to the quality of sequencing data, complexity of the genome, reference genome characteristics, and alignment algorithm parameters. Analyzing the reasons behind unmapped reads can provide insights into the quality and characteristics of the sequencing data and inform subsequent data processing and analysis steps.

A low percentage of reads that cannot be mapped to the reference genome after alignment with tools like HISAT2 can be attributed to several factors:

Quality of Sequencing Data: Low-quality sequencing data, characterized by high levels of noise, sequencing errors, or contamination, can result in a higher proportion of unmapped reads. Poor-quality reads may fail to align confidently to the reference genome due to mismatches, insertions, deletions, or other sequencing artifacts.

Fragment Length and Insert Size: In paired-end sequencing experiments, the distance between paired reads (insert size) should match the expected fragment length. If the insert size is too large or too small compared to the expected range, one or both reads may fail to align properly, leading to unmapped reads.

Sequence Complexity and Repetitive Regions: Regions of the genome that are highly repetitive or contain complex sequence patterns can pose challenges for alignment algorithms. Reads originating from repetitive regions may map ambiguously to multiple locations in the genome, making it difficult to assign them to specific genomic loci.

Sample Contamination or Cross-Talk: Contamination from environmental DNA, adapter sequences, or cross-talk between samples can introduce extraneous reads that do not originate from the target genome. Reference Genome Quality and Completeness: The quality and completeness of the reference genome can influence the success of read alignment.

Sequence Variation and Structural Variants: Genetic variation, including single nucleotide polymorphisms (SNPs), insertions, deletions, and structural variants, can affect the alignment of reads to the reference genome.

Alignment Parameters and Stringency: The choice of alignment parameters and alignment algorithm settings can impact the mapping efficiency and sensitivity of read alignment.

Part c- Building gene expression matrix

Run htseq-count on count aligned reads for differential expression analysis:

```
(samtoolenv) bio_narges_ghanei@ibbadmin-X10DAi:~$ samtools sort -@ 5 sample.bam -o sorted_sample.bam
```

I sort the BAM files before counting with htseq-count, for two reasons:

- It can improve the performance and memory efficiency of htseq-count, especially for paired-end data. Sorting ensures that most alignment mates appear close to each other in the data and hence the buffer is much less likely to overflow.
- It can avoid potential errors or inconsistencies in the counting results, especially for reads that map to multiple features or locations. Sorting ensures that the reads are processed in a consistent order and that the overlap resolution mode is applied correctly.

Therefore, sorting the BAM files before counting with htseq-count can make the analysis faster, more accurate, and more reliable.

```
(countenv) bio_narges_ghanei@ibbadmin-X10DAi:~$ htseq-count --format=bam --order=pos sorted_sample.bam /share_large/intro2bioinformatics/data
/Homo_sapiens.GRCh38.106.chr.gtf.gz > counting1.txt
```

The --format=bam option tells htseq-count that the input file is in BAM format. This option is required if the input file is not in SAM format.

The --order=pos option tells htseq-count that the input file is sorted by genomic position, which means that the reads are ordered according to their alignment coordinates on the reference genome.

I also can specify the mode of overlap resolution, the strandedness of the reads, and the output file name. this command uses the union mode, the true strandedness as default, and name the output files as counting1.txt.

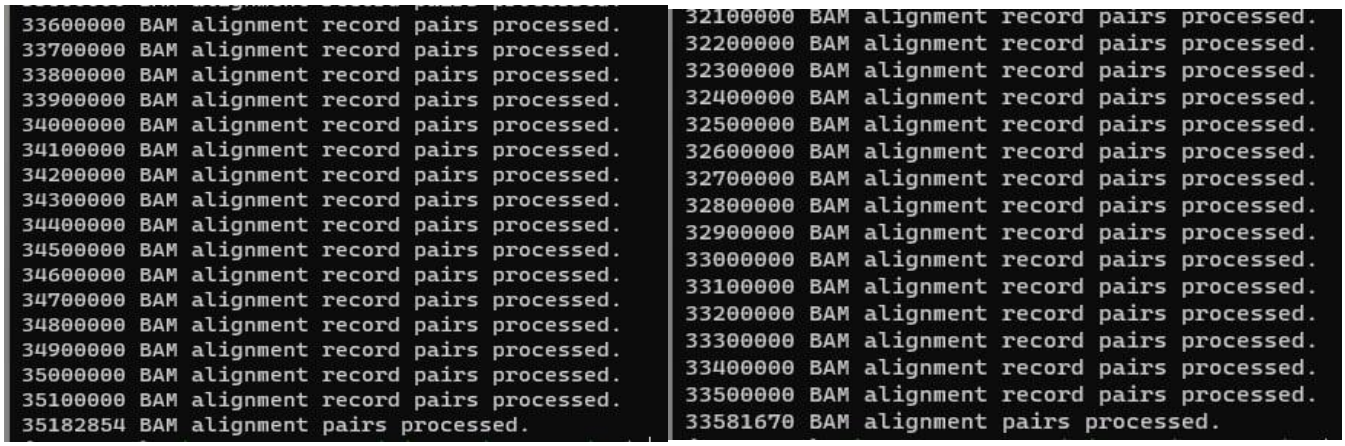
The mode of overlap resolution determines how to deal with reads that overlap more than one feature. For example, if a read overlaps two genes, should it be counted for both, none, or one of them? There are three possible modes: union, intersection-strict, and intersection-nonempty. The union mode counts the read for all the features it overlaps, the intersection-strict mode counts the read only if it overlaps exactly one feature, and the intersection-nonempty mode counts the read only if it overlaps a common set of features. The default mode is union.

The strandedness determines whether the data is from a strand-specific assay or not. A strand-specific assay is a method that preserves the information about which strand of DNA the RNA transcript came from. For example, if a read comes from the sense strand of a gene, should it be counted for that gene or not? There are three possible values for strandedness: yes, no, or reverse. The yes value means that the read has to be mapped to the same strand as the feature, the no value means that the read can be mapped to either strand of the feature, and

the reverse value means that the read has to be mapped to the opposite strand of the feature. The default value is yes, which assumes that the data is strand-specific.

This command uses the htseq-count tool to count the number of reads that map to each feature in a GTF file.

Result:



Merge results files into a single matrix for use in the edgeR package:

```
(countenv) bio_narges_ghanei@ibbadmin-X10DAi:~$ Rscript htseq-merge_all.R ./count_data/ merged_count_matrix
```

First, I downloaded htseq-merge_all.R from git repository and then merged countings.txt files. It gave me a merged matrix as result.

```
(countenv) bio_narges_ghanei@ibbadmin-X10DAi:~/count_data$ ls
counting1.txt  counting2.txt  merged_count_matrix.Rdata  merged_count_matrix.txt
```

```
1  import pandas as pd
2  df = pd.read_csv ('merged_count_matrix.txt', sep='\t', header=None, names=['name', 'first', 'second'])
3
```

✓ 19.4s

```
1  df ['equal'] = df ['first'].eq (df ['second'])
2
```

✓ 0.0s

```
1  df ['equal'].value_counts ()
2
```

I used this code to count the number of equal rows for 2 genomes. The result is as below:

```
False    41529
True     19975
Name: equal, dtype: int64
```

41529 rows has different values in patient genome and healthy genome and 19975 rows are equal for both.

Questions:

1- **How many genes are not expressed in control and tumor samples? Explain the results.**

Read the counting data

```
1  import pandas as pd
2  df1 = pd.read_csv ('counting1.txt', sep='\t', header=None, names=['name', 'expression'])
3  df2 = pd.read_csv ('counting2.txt', sep='\t', header=None, names=['name', 'expression'])
```

✓ 0.1s

calculate not expressed genes

```
1  import numpy as np
2
3  not_expressed_tumor = 0
4  not_expressed_control = 0
5
6  for i in range(len(df1["expression"])):
7      if df1["expression"][i] == 0:
8          not_expressed_tumor += 1
9
10 for i in range(len(df2["expression"])):
11     if df2["expression"][i] == 0:
12         not_expressed_control += 1
```

I counted the number of not expressed genes in both samples. Not expressed samples have the value of 0 in counting file. The result is as below:

```
Number of genes that are not expressed in tumor samples:  20879
Number of genes that are not expressed in control samples:  22445
percentage of not expressed tumor genes:  33.95 %
percentage of not expressed control genes:  36.49 %
```

2- Compare the matrix obtained at this stage with the corresponding gene expression submatrix of the main study. Discuss the differences.

```
import pandas as pd
cols = ["seqid", "source","type","start", "end", "score","strand", "phase", "gene_id", "gene_version","gene_name","gene_source",
df_homo = pd.read_csv ('Homo_sapiens.GRCh38.106.chr.gtf', sep='\t', comment='#', header=None, names=cols)
```

2m 17.2s Pytho

First I read the gtf file.

```
gene_geneID_map = {}

for i in range(len(df_homo["gene_id"])):
    gene_geneID_map[df_homo["gene_id"][i]] = df_homo["gene_name"][i]
```

1m 29.0s

Then I created a dictionary to map gene names to gene ids using gtf file.

```
import csv

csv_file = 'compair.csv'

# Writing data to the CSV file
with open(csv_file, 'w', newline='') as file:
    writer = csv.writer(file)
    for i in range(1, len(new_df["gene"])):
        geneId = new_df["gene"][i]
        gene_name = gene_geneID_map[geneId]
        for j in range(len(df_merge["name"])):
            if df_merge["name"][j] == gene_name:
                row = [gene_name, new_df["X48C_COUNT"][i], new_df["X48N_COUNT"][i], df_merge["first"][j], df_merge["second"][j]]
                break
        writer.writerows(row)


```

In the last part I wrote a code to write a file that for each gene_name says the count in tumor and control genomes in main matrix and count of tumor and control genomes in my merged matric of this parts result.

To compare more, the number of rows after doing previous parts on the data to number of original data' s rows is as below:

```
0.6075214620187305
```

And percentage of not expressed genes in main matrix is as below, which is different from results that we get in last part:

```
3.82
4.64
```

3- What are other software available to do this step? Name two other software and discuss their advantages and disadvantages.

In addition to HTSeq, there are several other software tools available for building gene expression matrices and performing read counting from high-throughput sequencing data. Two commonly used alternatives to HTSeq are featureCounts and Salmon. FeatureCounts and Salmon offer alternative approaches to read counting and transcript quantification, each with its own strengths and limitations. The choice between these tools depends on factors such as the nature of the sequencing data, the complexity of the transcriptome, computational resources available, and specific analysis requirements. Researchers may choose to evaluate and compare these tools based on their performance and suitability for their particular RNA-seq analysis pipelines.

1.featureCounts:

Advantages:

Speed and Efficiency: featureCounts is known for its speed and efficiency in counting reads, making it suitable for large-scale RNA-seq datasets with millions of reads.

Multimapping and Ambiguity Handling: It provides options for handling multimapping reads and ambiguous assignments, allowing users to control the stringency of read counting.

Compatibility: featureCounts supports various input file formats, including BAM/SAM and BED, making it compatible with a wide range of alignment and analysis pipelines.

Disadvantages:

Limited Transcriptome Annotation: It relies on existing genome annotations and may not perform optimally in cases where transcriptome annotations are incomplete or inaccurate.

Complexity of Options: While featureCounts offers flexibility in handling different scenarios, the numerous options and parameters may require some expertise to optimize for specific datasets and analysis goals.

2.Salmon:

Advantages:

Alignment-Free Quantification: Salmon uses a lightweight, alignment-free approach based on pseudoalignment and expectation-maximization (EM) algorithms, offering fast and accurate transcript quantification.

Robustness to Transcriptome Complexity: Salmon is robust to transcriptome complexity and performs well even in the presence of novel transcripts, alternative splicing events, and isoform switching.

Quantification of Transcript Abundance: It provides estimates of transcript-level abundance, allowing for more detailed analysis of isoform expression and differential splicing.

Disadvantages:

Resource Intensive: While Salmon is efficient in terms of computational time, it may require more memory resources compared to alignment-based methods like featureCounts and HTSeq, particularly for larger datasets and complex transcriptomes.

Dependence on Indexing: Salmon relies on pre-built transcriptome indices, which need to be generated separately for each reference transcriptome. This additional step may increase the setup complexity for users.

Part d- Differential gene expression analysis

Use edgeR to perform differential gene expression analysis:

first I downloaded the main matrix. Then I wrote this code to do wanted analysis.

install and import required libraries

```r
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

BiocManager::install("limma")
BiocManager::install("edgeR")
install.packages("ggplot2")

library('limma')
library('edgeR')
```

this code snippet ensures that the required packages ("limma", "edgeR", and "ggplot2") are installed and available for use in the R environment. It first checks if BiocManager is available and installs it if necessary. Then, it installs the Bioconductor packages "limma" and "edgeR", as well as the CRAN package "ggplot2". Finally, it loads the installed packages into the R environment using the library() function.

Read gene expression matrix

```r
gene_expression_matrix <- read.table('GSE104836_gene_exp.txt', header = TRUE, sep = "\t", stringsAsFactors = FALSE)

gene_expression_matrix <- gene_expression_matrix[,c("gene" ,"X101C_COUNT" , "X101N_COUNT" , "X111C_COUNT" ,"X111N_COUNT" , "X112C_COUNT" ,"X112N_COUNT", "X29C_COUNT" , "X29N_COUNT"
, "X34C_COUNT" , "X34N_COUNT" , "X48C_COUNT", "X48N_COUNT" , "X55C_COUNT" , "X55N_COUNT" , "X57C_COUNT" , "X57N_COUNT","X91C_COUNT" , "X91N_COUNT" , "X94C_COUNT" , "X94N_COUNT")]

group = c("Tumor", "Normal", "Tumor" , "Normal", "Tumor" ,"Normal", "Tumor" , "Normal", "Tumor","Normal", "Tumor", "Normal", "Tumor" ,
"Normal", "Tumor" ,"Normal", "Tumor" , "Normal", "Tumor","Normal")
```

```r
head(gene_expression_matrix,1)
```

A data.frame: 1 × 21

	gene	X101C_COUNT	X101N_COUNT	X111C_COUNT	X111N_COUNT	X112C_COUNT	X112N_COUNT	X29C_COUNT	X29N_COUNT	X34C_COUNT	⋯	X48C_COUNT	X48N_COUNT	X55C_COUNT	X55N_COUNT	X57C_COUNT	X57N_COUN
	<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	⋯	<int>	<int>	<int>	<int>	<int>	<int
1	A1BG	64	208	98	92	109	168	169	87	97	⋯	74	112	108	80	67	11

this code segment loads a gene expression matrix. Subsets it to include only specific columns, defines group labels for each sample, and displays the first row of the resulting dataframe for inspection.

```
dge <- DGEList(counts = gene_expression_matrix[, -1], genes = gene_expression_matrix[, 1] , group=group)
keep <- filterByExpr(dge)
dge <- dge[keep,, keep.lib.sizes=FALSE]
dge <- calcNormFactors(dge)
design <- model.matrix(~ group)
```

The code prepares the data for differential gene expression analysis by creating a DGEList object, filtering out low-expressed genes, normalizing the count data, and constructing a design matrix for modeling the differential expression.

```
dge <- estimateDisp(dge, design)
fit <- glmQLFit(dge, design)
qlf <- glmQLFTest(fit, coef=2)
results <- topTags(qlf, n=Inf)
de_genes <- results$table[results$table$FDR < 0.05 & abs(results$table$logFC) > log2(1.5), ]
```

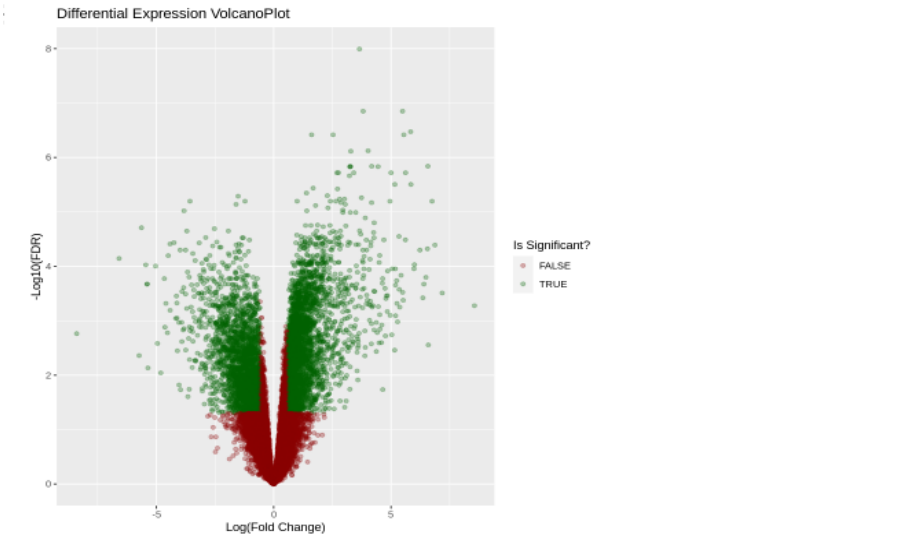
This code conducts a rigorous analysis to identify differentially expressed genes based on the design and count data provided. It estimates dispersion, fits a GLM, performs hypothesis tests, and extracts significant results based on specified criteria.

```
print(de_genes)
write.csv(de_genes, "de_genes.csv")
```

I saved the result matrix in a csv file and attached it to my project.

```r
library("ggplot2")
results_df <- as.data.frame(results$table)
results_df$gene <- rownames(results_df)
results_df$Significant <- results_df$FDR < 0.05 & abs(results_df$logFC) > log2(1.5)
plt <- ggplot(results_df, aes(x = logFC, y = -log10(FDR), color = Significant)) +
            geom_point(alpha = 0.3) +
            scale_color_manual(values = c("darkred", "darkgreen")) +
            labs(
            title = "Differential Expression",
            x = "Log(Fold Change)",
            y = "-Log10(FDR)",
            color = "Is Significant?") +
            theme(legend.position = "right")

plt + theme_gray()
```



This code generates a volcano plot to visualize the results of differential gene expression analysis, highlighting significant genes based on specified criteria for fold change and FDR.

Questions:

1- **How many genes are given to edgeR? How many of them are differentially expressed in tumor versus normal samples? How do you define statistical significance in this context?**

Number of genes

```r
[ ]  total_genes <- nrow(dge)
     print(total_genes)
```

```
[1] 23675
```

Number of differentially expressed gened

```
[ ] significant_genes <- sum(results$table$FDR < 0.05 & abs(results$table$logFC) > log2(1.5))
    print(significant_genes)
```

```
[1] 6224
```

Statistical significance is typically defined based on hypothesis testing. The goal is to identify genes whose expression levels change significantly between different experimental conditions or sample groups.

Statistical significance in the context of differential gene expression analysis using tools like edgeR is defined based on statistical tests, p-value thresholds, and multiple testing correction methods. By applying rigorous statistical criteria, researchers can identify genes that exhibit robust and statistically significant changes in expression levels between experimental conditions or sample groups.

2- **Determine the percentage of differentially expressed genes with |log2FoldChange| > 1.5.**

Percentage of differentially expressed genes

```
percentage_de_genes <- (significant_genes / total_genes) * 100
print(percentage_de_genes)
```

```
[1] 26.28933
```

3- **Explain the difference between P-value and FDR?**

In gene expression analysis, both p-values and false discovery rate (FDR) are important measures used to assess the statistical significance of differential gene expression. However, they serve different purposes and provide complementary information about the results of statistical tests.

P-value:

The p-value is a measure of the strength of evidence against the null hypothesis in a statistical test. It represents the probability of observing the observed data, or more extreme data, under the assumption that the null hypothesis is true. In gene expression analysis, p-values are used to determine whether the observed differences in gene expression levels between experimental conditions or sample groups are statistically significant or merely due to chance. A smaller p-value indicates stronger evidence against the null hypothesis and suggests that the observed differences in gene expression are unlikely to have occurred by random chance alone.

False Discovery Rate (FDR):

The FDR is a method for controlling the rate of false positives, or type I errors, in multiple hypothesis testing scenarios. It represents the expected proportion of falsely rejected null hypotheses among all hypotheses rejected as significant. In gene expression analysis, where thousands of hypotheses (genes) are tested

simultaneously, controlling the FDR is essential to limit the number of false discoveries and ensure the reliability of the results. Unlike p-values, which assess the significance of individual hypotheses, FDR adjustment accounts for multiple testing and controls the overall rate of false positives across all tested hypotheses.

While p-values provide a measure of statistical significance for individual hypotheses (genes), FDR control ensures that the overall rate of false positives is controlled in the context of multiple testing.

Part e- Gene Ontology enrichment analysis

install and import libraries

```
[ ]  if (!require("BiocManager", quietly = TRUE))
       install.packages("BiocManager")
     BiocManager::install("goseq")
     library(goseq)

     if (!require("rtracklayer", quietly = TRUE))
       install.packages("rtracklayer")
     library(rtracklayer)

     if (!require("R.utils", quietly = TRUE))
       install.packages("R.utils")
     library(R.utils)
```

unzipping gtf file

```
[ ]  gunzip("Homo_sapiens.GRCh38.106.chr.gtf.gz", destname = "./Homo_sapiens.GRCh38.106.chr.gtf")
```

reading gtf file

```
gtf <- readGFF("Homo_sapiens.GRCh38.106.chr.gtf")
length <- as.vector(dim(gtf)[2])
```

choose genes

```
[ ]  de <- gene_id %in% gtf
```

Go enrichment

```
pwf = nullp(de,"hg19", "ensGene")

enriched_GO <- goseq(
    gene2cat = gtf,
    method = "Wallenius",
    pwf = pwf
)
```

This part of code is using the **goseq** package in R to perform Gene Ontology analysis for RNA-seq data. It is calculating a probability weighting function (pwf) for a set of genes based on their differential expression (de) status and their length bias. It is using the "hg19" genome and the "ensGene" gene identifier as arguments for the nullp function. It is testing for over-representation of GO categories among the differentially expressed genes using the goseq function. It is using the gene2cat data frame to map genes to categories, the "Wallenius" method to calculate the unbiased category enrichment scores, and the pwf object as input.

Extract significant GO terms and KEGG terms

```
[ ]  significant_terms <- subset(enriched_GO, PLE <= 0.05)
     significant_KEGG <- enriched_GO$over_represented_Kegg_pathways
```

Plot Biological Process

```
[ ]  ggplot(significant_terms[significant_terms$category == "BP", ], aes(x = description, y = PLE)) +
         geom_bar(stat = "identity") +
         theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
         labs(x = "Biological Process", y = "PLE")
```

Plot Molecular Function

```
[ ]  ggplot(significant_terms[significant_terms$category == "MF", ], aes(x = description, y = PLE)) +
         geom_bar(stat = "identity") +
         theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
         labs(x = "Molecular Function", y = "PLE")
```

Plot Cellular Component

```
[ ]  ggplot(significant_terms[significant_terms$category == "CC", ], aes(x = description, y = PLE)) +
         geom_bar(stat = "identity") +
         theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
         labs(x = "Cellular Component", y = "PLE")
```

Plot KEGG

```
[ ]  significant_KEGG_df <- data.frame(KEGG_Term = names(significant_KEGG), Significance = significant_KEGG)


     ggplot(significant_KEGG_df, aes(x = KEGG_Term, y = Significance)) +
       geom_bar(stat = "identity", fill = "blue") +
       labs(title = "Significant KEGG Terms",
            x = "KEGG Terms",
            y = "Significance") +
       theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

Questions

2- Do a brief study of each of the significant terms and discuss which terms you think may play an important role.

- **Biological Process**: This category describes a series of events or molecular activities that contribute to a specific biological objective, such as cell cycle, photosynthesis, or immune response. Biological processes are often complex and involve multiple steps and components. They can be organized into a hierarchical structure, where more general processes are parents of more specific processes.

- **Molecular Function**: This category describes the elemental activities of a gene product at the molecular level, such as binding, catalysis, or transport. Molecular functions are the basic units of biological processes, and they can be shared by different gene products that participate in different processes. They can also be organized into a hierarchical structure, where more general functions are parents of more specific functions.

- **Cellular Component**: This category describes the locations or structures where a gene product is active, such as organelles, membranes, or complexes. Cellular components are the physical contexts of biological processes and molecular functions, and they can provide information about the subcellular localization and interactions of gene products. They can also be organized into a hierarchical structure, where more general components are parents of more specific components.

- **KEGG**: This category is not part of the Gene Ontology, but rather a separate database that provides information about pathways, diseases, drugs, and other aspects of biological systems. KEGG pathways are collections of genes or gene products that are involved in a specific biological process or function, such as glycolysis, apoptosis, or DNA repair. KEGG pathways can help to understand the molecular mechanisms and interactions of biological phenomena.

!!! All codes and output files are attached. Other files are in home directory path. !!!

Thank you for you attention