Project 2: RNA-Seq analysis

Algorithms in Bioinformatics
Mahboobeh (Mariya) Golchinpour leili

Objective
In this project, you will learn how to **extract a gene expression matrix** from **fastq files** and perform advanced analysis on the resulting data. Specifically, you will analyze the next-generation sequencing expression profiles of unpaired normal and COVID-19 FFPE bronchoalveolar lavage or lung samples from a study with accession number GSE190496. The table below provides information about the selected samples from this study that will be used in your analysis.

Tissue	SRR Accession number
FFPE Normal human lung	SRR17172463, SRR17172465, SRR17172466, SRR17172467
FFPE COVID19 patient lung	SRR17172468, SRR17172469, SRR17172470, SRR17172471
FFPE Normal human brochoalveolar lavage cells	SRR17172480, SRR17172481, SRR17172482
FFPE COVI19 Patient brochoalveolar lavage cells	SRR17172485, SRR17172486, SRR17172487

My files

Normal: **SRR17172481 :single end**

Patient: **SRR17172485: single end**

SRR17172481: Accession: GSM5724768: NB2

SRR17172485: Accession: GSM5724772: CB1

Data preparation

Each student is supposed to preprocess paired data of normal and covid19 lung tissue samples.
To do this project, you need to download fastq files from SRA using sratoolkit using the
fastq-dump (Hint: use fastq-dump [options] file.fastq.gz)

```
#download data
prefetch SRR17172481
prefetch SRR17172485

#converting multiple SRA to fastq
fastq-dump SRR17172481.sra
fastq-dump SRR17172485.sra
```

Part a- Quality control and trimming

In this step, first, assess the **read qualities** using the **FastQC** software.

Then use the **Trimmomatic** software to **improve the read qualities** through the **read trimming**. Recheck the read qualities to make sure the problems are solved. (Hint: you may use fastqc and TrimmomaticPE)

Please answer the following questions.

1. What is the average number of reads across samples before and after the read trimming?
2. Compare the read length averages in different samples before and after the read trimming?
3. Compare the read quality distributions over all sequences before and after the read trimming.
4. What does the Adaptor Content warning indicate?
5. Why do we first remove the Adapter sequences for the reads and then the low-quality bases?
6. What does the quality of bases mean, and how is it obtained?

File: SRR17172481 Before Trimming

FastQC Report

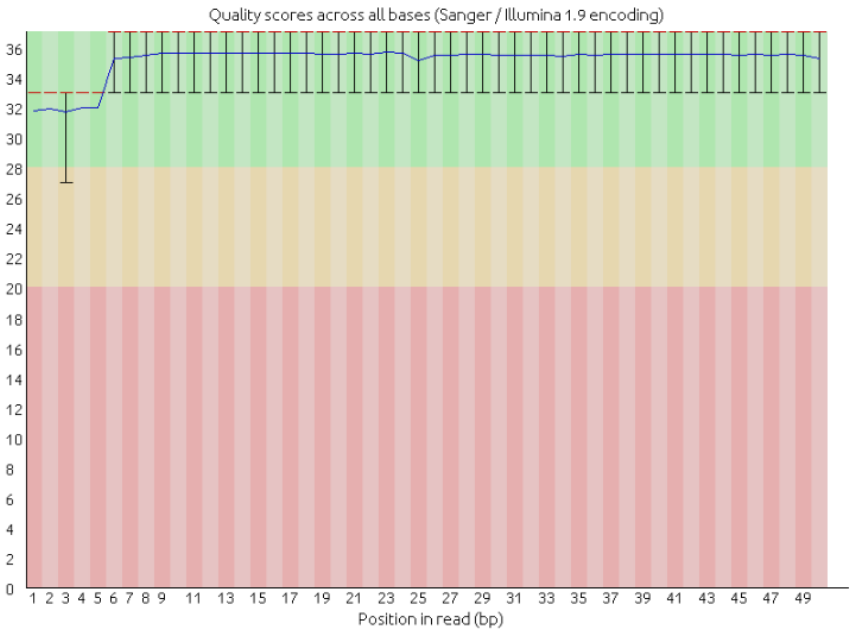
Summary

- 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
- Kmer Content

Basic Statistics

Measure	Value
Filename	SRR17172481.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	7521289
Sequences flagged as poor quality	0
Sequence length	50
%GC	50

Per base sequence quality



Trimming

```
trimmomatic SE -phred33 SRR17172481.fastq SRR17172481-trimed.fastq LEADING:32 TRAILING:32 SLIDINGWINDOW:4:15 MINLEN:36
```

- `SE`: This option specifies that the input file contains **single-end reads.**
- `phred33`: TQuality scores in the input file are encoded using the Phred+33 format.
- `LEADING:32`: This option specifies that any bases with a **quality score below 32 at the leading end of a read should be trimmed.**
- `TRAILING:32`: This option specifies that any bases with a **quality score below 32 at the trailing end of a read should be trimmed.**
- `SLIDINGWINDOW:4:15`: This option specifies that **a sliding window of 4 bases will be used to scan the read, and if the average quality score within the window falls below 15, the window will be trimmed.**
- `MINLEN:36`: This option specifies that any reads that are shorter than 36 bases after trimming should be discarded.

File: SRR17172481 After Trimming

FastQC Report

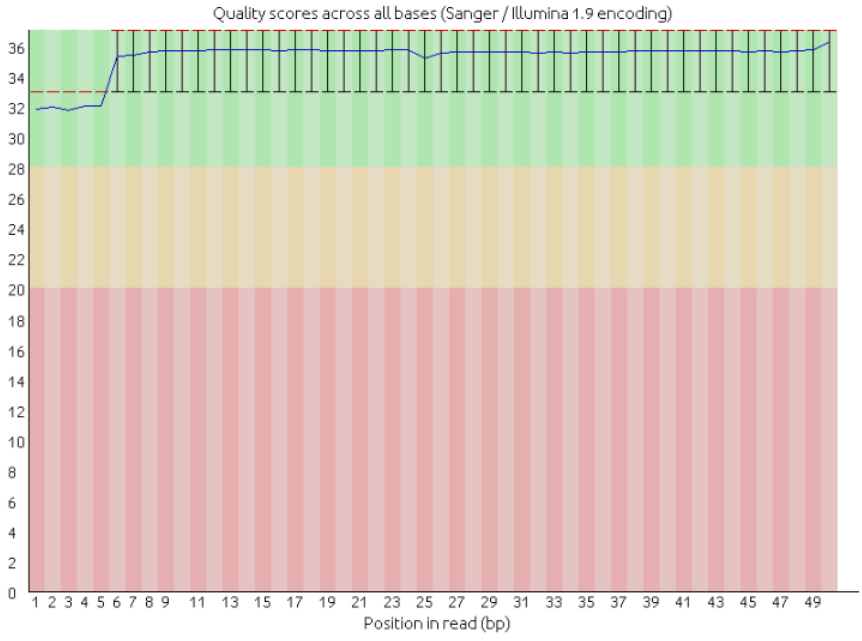
Summary

- 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
- Kmer Content

Basic Statistics

Measure	Value
Filename	SRR17172481-trimed.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	7427994
Sequences flagged as poor quality	0
Sequence length	36-50
%GC	50

Per base sequence quality



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

	before Trimming	after Trimming:
SRR17172481	7521289	7427994
SRR17172485	5013115	4932884

File: SRR17172481

Total Sequences before Trimming: 7521289

Total Sequences after Trimming: 7427994

(7521289+7427994)/2= 7474641.5

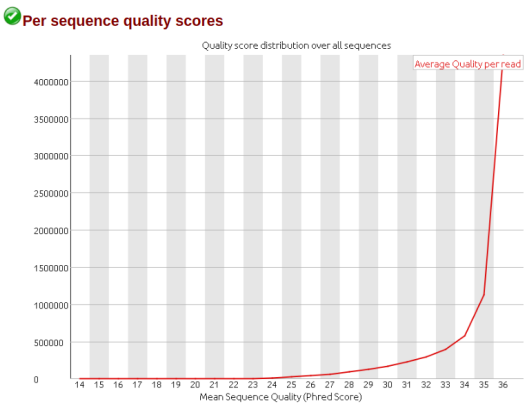
2. Compare the read length averages in different samples before and after the read trimming?

Sequence length before Trimming: 50

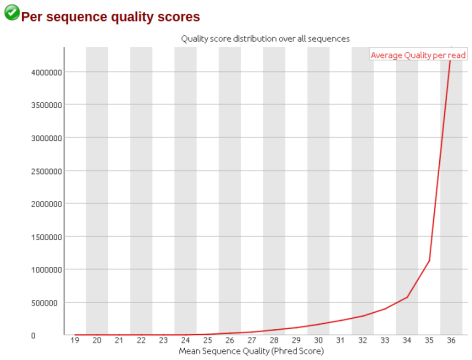
Sequence length after Trimming: 36-50

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

Per sequence quality scores before Trimming



Per sequence quality scores after Trimming

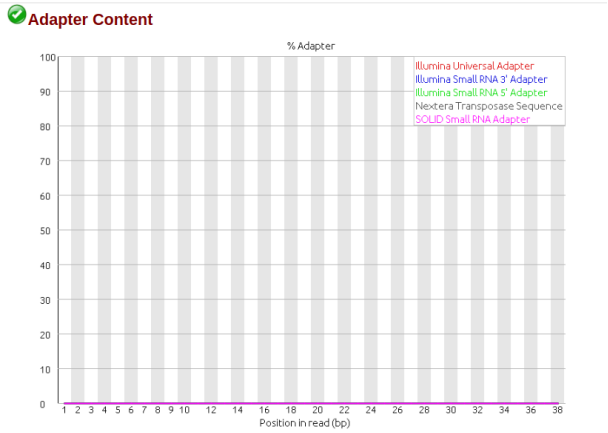


4. What does the Adaptor Content warning indicate?

The "Adaptor Content" warning in FastQC indicates that the **sequencing reads contain a significant amount of adapter sequence**, which may indicate that the adapter was not properly removed during the library preparation or sequencing process. **Adapters** are short sequences that are used to ligate the sequencing library fragments to the sequencing platform.

The "Adaptor Content" warning in FastQC is triggered when the adapter sequence is detected in a significant proportion of the sequencing reads. The warning message will indicate the type of adapter sequence that was detected and the percentage of reads that contain this sequence.

To address this warning, you may need to re-evaluate the library preparation or sequencing process and ensure that the adapter removal step is performed properly. This may involve optimizing the adapter removal protocol, increasing the sequencing depth to compensate for the loss of reads due to adapter contamination, or using bioinformatic tools to remove adapter sequences from the raw sequencing data. It is important to address the "Adaptor Content" warning before proceeding with downstream analysis to ensure that the quality and integrity of the sequencing data are not compromised.



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

because adapter contamination can significantly affect the quality and accuracy of the sequencing data. By removing adapter sequences before removing low-quality bases, we can reduce the impact of adapter contamination and improve the overall quality of the sequencing data. After adapter removal, we can then focus on removing low-quality bases and other types of noise from the sequencing reads.

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

The quality of bases in a sequencing read refers to the confidence level or probability that each base call is correct. Quality scores are assigned to each base call in the sequencing read based on the base call quality, which is determined using the Phred score system.

The Phred score system is a logarithmic scale used to represent the base call quality scores, which range from 0 to 40. A quality score of 0 represents a base call with a 100% chance of being incorrect, while a quality score of 40 represents a base call with a 99.99% chance of being correct. In general, a higher quality score indicates a higher confidence level in the base call.

File: SRR17172485 Before Trimming

FastQC Report

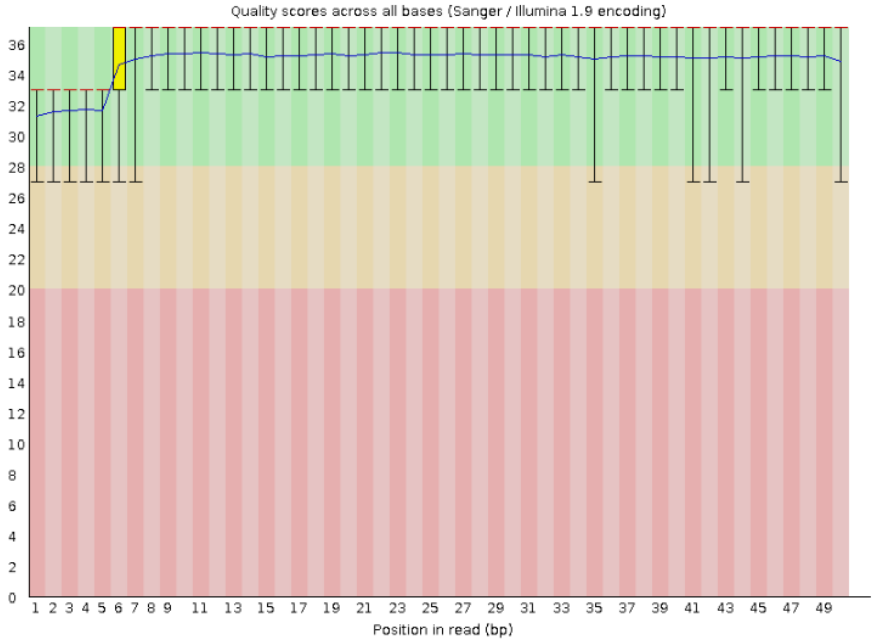
Summary

- 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	SRR17172485.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	5013115
Sequences flagged as poor quality	0
Sequence length	50
%GC	55

Per base sequence quality



File: SRR17172485 After Trimming

FastQC Report

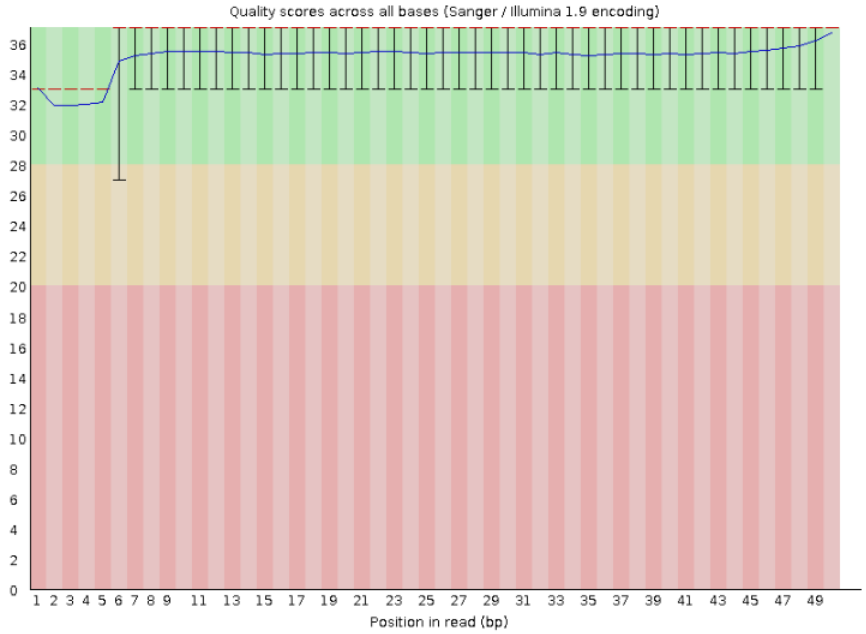
Summary

- 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	SRR85.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	4932884
Sequences flagged as poor quality	0
Sequence length	36-50
%GC	55

Per base sequence quality



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

File: SRR17172485

Total Sequences before Trimming: 5013115

Total Sequences after Trimming: 4932884

(5013115+ 4932884)/2 = 4972999.5

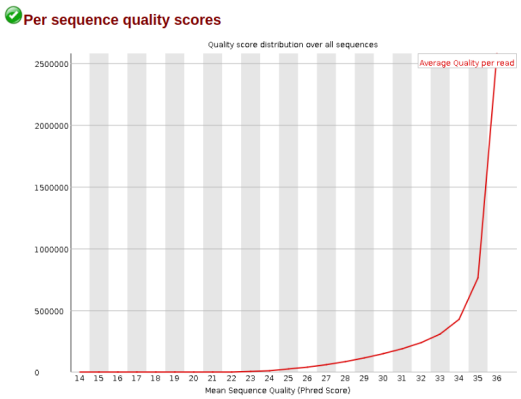
2. Compare the read length averages in different samples before and after the read trimming?

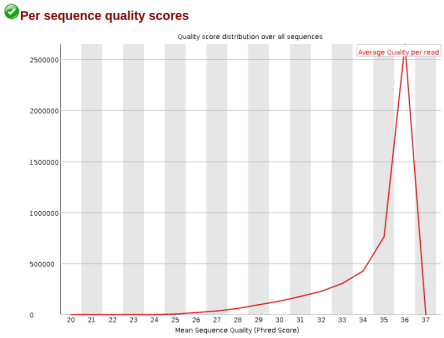
Sequence length before Trimming: 50

Sequence length after Trimming: 36-50

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

Per sequence quality scores before Trimming	Per sequence quality scores after Trimming





Untitled

Part b- Read mapping

In the second step, map the reads to the reference genome using the HISAT2 software. To map reads to the reference, the HISAT2 software uses a graph-based alignment and a variety of metrics. The output of this step will be a SAM file for each sample. (Hint: you may use hisat2 [options]* -x <ht2-idx> {-1 <m1> -2 <m2> } [-S <sam>])

Use the Samtools software to convert the HISAT2 SAM files to BAM files (Hint: you may use samtools view [options] <in.sam>). Make sure you delete the SAM files afterwards.
This step uses the hisat2-build software to index the Homo sapiens (GRCh38) reference genome (available at this link). The fasta files of the genome have already been downloaded and placed in the path " /home/studmin/Proj_02/Homo_Genome/Homo_sapiens.GRCh38.dna.toplevel.fa" . Do not copy the genome fasta file into your directory. Refer to the HISAT2 manual for a more detailed explanation. Please answer the following questions. (Hint: you may use hisat2-build [options]* <reference_in> <ht2_index_name>)

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

SAM and BAM files are both file formats used **to store alignment information from sequence read mapping to a reference genome**, but SAM (Sequence Alignment/Map) is a text-based file format that stores the alignment information in a human-readable format. SAM files contain a header section that describes the reference genome and the sequencing data, followed by a body section that lists the alignment information for each read. The body section includes a set of fields that describe the alignment position, quality, and other attribtes of each read.

BAM (Binary Alignment/Map) is a binary format that stores the same alignment information as a SAM file, but in a more compact and efficient format. BAM files are compressed, indexed, and can be read more quickly than SAM files. BAM files are also more suitable for storage and analysis of large-scale sequencing data, as they occupy less disk space and can be processed more efficiently.

The main differences between SAM and BAM files are:

1. File format: SAM is a text-based format, while BAM is a binary format.
2. Size: BAM files are smaller than SAM files because they are compressed.
3. Speed: BAM files can be read more quickly than SAM files because they are binary and indexed, which makes them more efficient to process.
4. Human-readability: SAM files are human-readable, while BAM files are not.

2. What is the purpose of indexing the genome?

The purpose of indexing the genome is to enable efficient and rapid searching of the genome for specific sequences or regions of interest. Genome indexing involves creating an index or hash table that maps the sequence information to a set of unique identifiers, which can be used to quickly retrieve the corresponding sequence information from the genome database.

Indexing the genome can significantly improve the efficiency and accuracy of bioinformatics analysis by enabling rapid sequence searches and reducing the computational resources required for data processing. Genome indexing also allows for the integration and sharing of genomic data across different research groups, which can facilitate collaboration and accelerate scientific discovery.

There are several types of genome indexing methods, including hash-based indexing, suffix arrays, and Burrows-Wheeler Transform (BWT). Each method has its advantages and disadvantages, and the choice of indexing method may depend on the specific application and the characteristics of the genome dataset being analyzed.

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

SRR17172481	74.56% aligned exactly
SRR17172485	16.91% aligned exactly

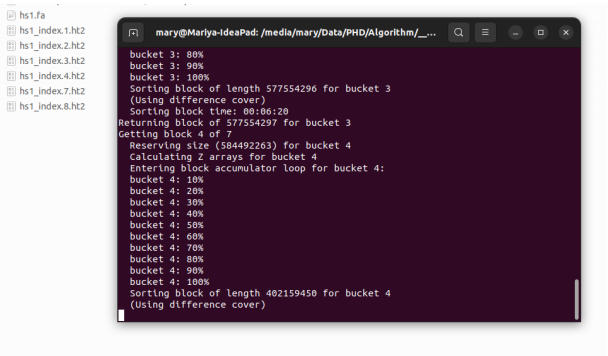
A low percentage of mapped reads could be due to a number of factors, including:

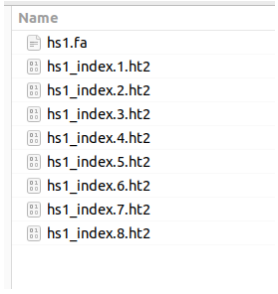
1. Poor quality of the sequencing data: If the sequencing reads are of low quality, it can be difficult to map them to a reference genome. This can result in a low mapping percentage.
2. Insufficient coverage of the reference genome: If the reference genome is incomplete or has gaps in coverage, it may be difficult to map all of the sequencing reads to the genome. This can result in a low mapping percentage.
3. Presence of contaminants or non-target sequences: If the sequencing data contains contaminants or non-target sequences (e.g., from host organisms or environmental samples), it can reduce the number of reads that can be mapped to the reference genome. This can result in a low mapping percentage.
4. Alignment parameters: The mapping software used to align the reads to the reference genome may have different parameters and settings that can affect the mapping percentage. For example, if the software is set to be very stringent, it may reject many reads that could have been mapped with a more relaxed setting.

The reference genome (`hs1.fa.gz`) downloaded from this link: https://hgdownload.soe.ucsc.edu/goldenPath/hs1/bigZips/

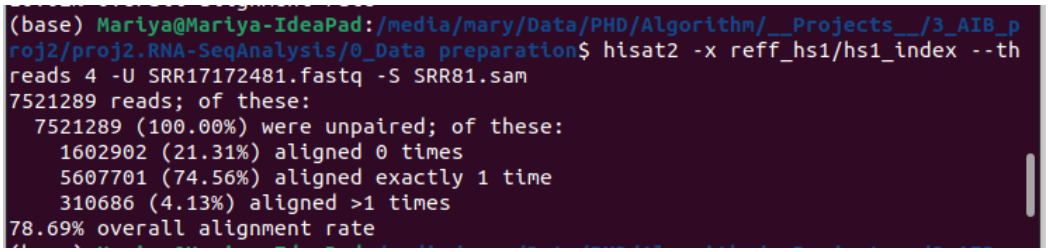
Index a reference genome (hs1.fa) using the HISAT2 software.

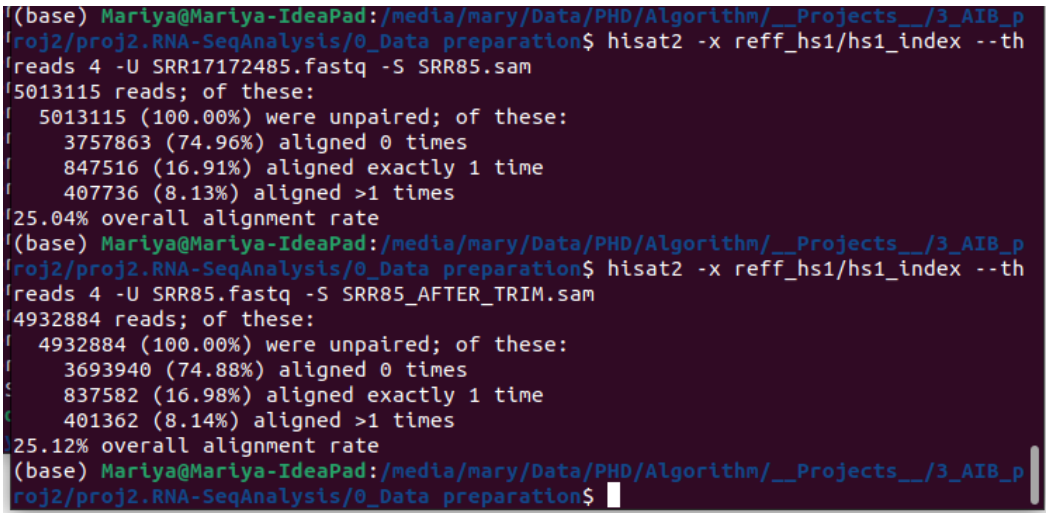
```
#1. index reference genome
hisat2-build hs1.fa hs1_index
```



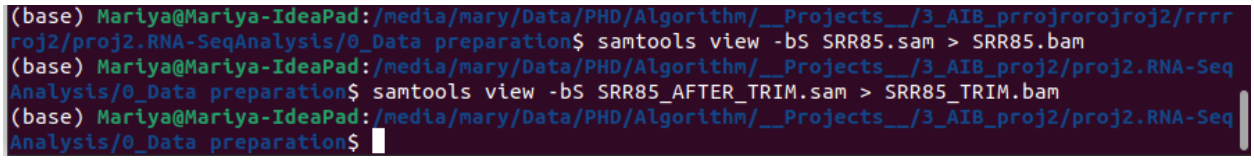


```
#2. map SRR17172481 reads to the reference and create sam file
hisat2 -x reff_hs1/hs1_index --threads 4 -U SRR17172481.fastq -S SRR81.sam
#2. map SRR17172485 reads to the reference and create sam file
hisat2 -x reff_hs1/hs1_index --threads 4 -U SRR17172485.fastq -S SRR85.sam
```






```
#3. convert sam file to bam file
samtools view -bS SRR81.sam > bam_files/SRR81.bam
```




```
#sort bam file
samtools sort SRR81.bam -o SRR81_SORTED.bam

#index sorted bam file
samtools index SRR81_SORTED.bam
```

Part c- Building gene expression matrix

In the third step, run **htseq-count** on count aligned reads for differential expression analysis. You can use the HTSeq documentation for further explanation. In this step, you need a gene/transcript annotation file that you can download from this link. **Merge results files into a single matrix for use in the edgeR package.**

Please answer the following question after completing this step.

1. How many genes are not expressed in control and covid samples? Explain the results.
2. Compare the matrix obtained at this stage with the corresponding gene expression submatrix of the main study. Discuss the differences.
3. What are other software available to do this step? Name two other software and discuss their advantages and disadvantages.

Inputs:

catLiftOffGenesV1.gtf

sorted bam file SRR81_SORTED.bam

catLiftOffGenesV1.gtf contain below items, and i used **gene_name** for htseq-count

```
(base) Mariya@Mariya-IdeaPad:/media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data_preparation/SRR17172481$ head catLiftOffGenesV1.gtf
chr1    Liftoff transcript      11136   12457   .       -       .       transcript_id "LOFF_T0000001"; gene_id "LOFF_G0000001"; gene_name "AL627309.3"
chr1    Liftoff exon    11136   11635   .       -       .       transcript_id "LOFF_T0000001"; gene_id "LOFF_G0000001"; gene_name "AL627309.3";
chr1    Liftoff exon    11639   12457   .       -       .       transcript_id "LOFF_T0000001"; gene_id "LOFF_G0000001"; gene_name "AL627309.3";
chr1    Liftoff transcript      11630   13433   .       +       .       transcript_id "LOFF_T0000002"; gene_id "LOFF_G0000002"; gene_name "AP006222.2"
chr1    Liftoff exon    11630   11831   .       +       .       transcript_id "LOFF_T0000002"; gene_id "LOFF_G0000002"; gene_name "AP006222.2";
chr1    Liftoff exon    12900   13433   .       +       .       transcript_id "LOFF_T0000002"; gene_id "LOFF_G0000002"; gene_name "AP006222.2";
chr1    CAT     transcript      14253   21099   8940    +       .       transcript_id "CHM13_T0000001"; gene_id "CHM13_G0000001"; gene_name "AC114498.1"
chr1    CAT     exon    14253   14325   .       +       .       transcript_id "CHM13_T0000001"; gene_id "CHM13_G0000001"; gene_name "AC114498.1";
chr1    CAT     exon    20566   21099   .       +       .       transcript_id "CHM13_T0000001"; gene_id "CHM13_G0000001"; gene_name "AC114498.1";
chr1    CAT     transcript      14292   20905   8500    +       .       transcript_id "CHM13_T0000002"; gene_id "CHM13_G0000001"; gene_name "AC114498.1"
```

Definitions

- `htseq-count`: This is a software tool used for counting reads mapped to genomic features (such as genes) in an RNA sequencing experiment. It takes a BAM file (a binary alignment file) and a GTF file (a file containing genomic annotations) as input.
- `format bam`: This specifies that the input file format is BAM.
- `gene_name`: This specifies that the feature IDs in the GTF file are in the "**gene_name**" attribute field.
- `SRR81_SORTED.bam`: This is the input BAM file containing the RNA sequencing reads that have been aligned to the genome.
- `catLiftOffGenesV1.gtf`: This is the input GTF file containing the genomic annotations (such as gene locations and names) that will be used to count the reads.
- `count_SRR81_SORTED_gene_name.txt`: This specifies that the output of the command should be written to a text file named "count_SRR81_SORTED_gene_name.txt". The output will
  contain the counts of reads that map to each gene in the GTF file

```
# htseq-count create count file for SRR17172485
htseq-count htseq-count --format bam gene_name SRR85_SORTED.bam
catLiftOffGenesV1.gtf > count_SRR85_SORTED_gene_name.txt
```

```
(base) Mariya@Mariya-IdeaPad:/media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/SRR17172485$
head count_SRR85_SORTED_gene_name.txt -n 20
5S_rRNA 0
5_8S_rRNA       0
7SK     0
A1BG    0
A1BG-AS1        0
A1CF    0
A2M     0
A2M-AS1 0
A2ML1   0
A2ML1-AS1       0
A2ML1-AS2       0
A2MP1   0
A3GALT2 0
A4GALT  0
A4GNT   0
AA06    0
AAAS    37
AACS    0
AACSP1  0
AADAC   21
```

```
# htseq-count create count file for SRR17172481
htseq-count --format bam gene_name SRR81_SORTED.bam
catLiftOffGenesV1.gtf > count_SRR81_SORTED_gene_name.txt
```

```
(base) Mariya@Mariya-IdeaPad:/media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/SRR17172481$
head count_SRR81_SORTED_gene_name.txt -n 20
5S_rRNA 0
5_8S_rRNA       0
7SK     0
A1BG    0
A1BG-AS1        0
A1CF    0
A2M     530
A2M-AS1 0
A2ML1   0
A2ML1-AS1       0
A2ML1-AS2       0
A2MP1   0
A3GALT2 0
A4GALT  207
A4GNT   20
AA06    0
AAAS    72
AACS    121
AACSP1  0
AADAC   0
```

The **output file generated by HTSeq-count** contains the **read counts for each genes** in **the annotation file** that were **overlapped by the RNA sequencing reads.** The format of the output file is a **tab-separated table** with two columns: the **genes name** and the **read count** for that feature.

some genes repeated multiple time this because of isoform.

```
 7 A2M      530
 8 A2M-AS1 0
 9 A2ML1   0
10 A2ML1-AS1        0
11 A2ML1-AS2        0
12 A2MP1   0
```

Merge two coun files of SRR17172481 and SRR17172485

Each files has 60836 line

```
7  nrow(counts81)
8  nrow(counts85)
9
11:1  (Top Level)

Console  Terminal  Background Jobs
R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/RNA_SEQ
> nrow(counts81)
[1] 60835
> nrow(counts85)
[1] 60835
> nrow(counts85)
```

Merge two files

```
1  #count matrix SRR81
2  counts81 <- read.table("Files/count_SRR81_SORTED_gene_name.txt", header = TRUE, row.names = 1)
3  counts81
4  #count matrix SRR81
5  counts85 <- read.table("Files/count_SRR85_SORTED_gene_name.txt", header = TRUE, row.names = 1)
6  counts85
7  nrow(counts81)
8  nrow(counts85)
9  merge_counts <- merge(counts81, counts85, by = "row.names")
10 merge_counts
11 nrow(merge_counts)
12 colnames(merge_counts) <- c("GeneNames", "SRR81.NB2", "SRR85.CB1")
13 write.table(merge_counts, file = "Files/Merge_genename_counts_SRR81_SRR85.txt", sep = "\t", quote = FALSE, row.names =
14
12:65  (Top Level)  R Script

Console  Terminal  Background Jobs
R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/RNA_SEQ/
> head(merge_counts)
  GeneNames SRR81.NB2 SRR85.CB1
1 5_8S_rRNA         0         0
2       7SK         0         0
3      A1BG         0         0
4  A1BG-AS1         0         0
5      A1CF         0         0
6       A2M       530         0
>
```

```
> head(merge_counts,20)
   GeneNames SRR81.NB2 SRR85.CB1
1  5_8S_rRNA         0         0
2        7SK         0         0
3       A1BG         0         0
4   A1BG-AS1         0         0
5       A1CF         0         0
6        A2M       530         0
7    A2M-AS1         0         0
8     A2ML1         0         0
9  A2ML1-AS1         0         0
10 A2ML1-AS2         0         0
11    A2MP1         0         0
12  A3GALT2         0         0
13   A4GALT       207         0
14    A4GNT        20         0
15     AA06         0         0
16     AAAS        72        37
17     AACS       121         0
18   AACSP1         0         0
19    AADAC         0        21
20  AADACL2         0         0
```

Gene A2M showed expression of 530 reads in the normal condition, but no expression was observed in the COVID condition.

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

```
1
2  merg_file<- read.csv("Files/Merge____genename_counts_SRR81_SRR85.csv")
3  allgenes<-nrow(merg_file)
4  allgenes
5  not_expressed_control <- sum(merg_file[, 2] == 0)
6  not_expressed_control
7  not_expressed__covid <- sum(merg_file[, 3] == 0)
8  not_expressed__covid
9  cat ("ALL genes are:",allgenes," genes are not expressed in control samples: ", not_expressed_control,
10       "genes are not expressed in covid samples: ",not_expressed__covid)
```

2:57 (Top Level) R

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/RNA_SEQ/

```
[1] 60836
> not_expressed_control <- sum(merg_file[, 2] == 0)
> not_expressed_control
[1] 50771
> not_expressed__covid <- sum(merg_file[, 3] == 0)
> not_expressed__covid
[1] 53366
> cat ("ALL genes are:",allgenes," genes are not expressed in control samples: ", not_expressed_control,
+        "genes are not expressed in covid samples: ",not_expressed__covid)
ALL genes are: 60836  genes are not expressed in control samples:  50771 genes are not expressed in covid samples:  53366
> num_zeros <- apply(merg_file == 0, 2, sum)
> num_zeros
Gene.names  SRR81.NB2  SRR85.CB1
         0      50771      53366
```

50,771 genes that are not expressed in sample SRR81.NB2 (control sample)

and **53,366 genes that are not expressed in sample SRR85.CB1 (COVID-19 sample)**.

All genes are 60,836, as results shows a large number of genes are not expressed in both the control and COVID-19 samples. This is not unexpected, as it is common for many genes to be silent or not expressed under certain conditions. However, it is important to note that the absence of gene expression does not necessarily mean that the gene is not important or relevant to the biological process being studied.

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

We have two Main files:

controls: ”GSE190496_Gene_counts_NB1_NB5.csv”

covid: “GSE190496_Gene_counts_CB1_CB8.csv’”

we should merge this two files and create a matrix with all control and covid samples. we merge these files by “counts” colomns.

c__comparison.R*

Source on Save | Run

```
2  # ============================================
3  counts_2 = read.csv('../GSE96/DATA/GSE190496_Gene_counts_CB1_CB8.csv')
4  head(counts_2)
5  colnames(counts_2) = c(counts_2[1, 1:5], colnames(counts_2)[6:13])
6  dim(counts_2)
7  head(counts_2)
8  counts_2 = counts_2[-1, ]
9  head(counts_2)
10 counts_2 = counts_2[, -14]
11 head(counts_2)
12 counts_1 = read.csv('../GSE96/DATA/GSE190496_Gene_counts_NB1_NB5.csv')
13 head(counts_1)
14 merged_counts = merge(counts_1, counts_2, by='COUNTS')
15 head(merged_counts)
16 write.csv(merged_counts, "Files/merged_all_counts__NB1_NB5_CB1_CB8.csv")
```

12:51 (Untitled)

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/RNA_SEQ/

```
> merged_counts = merge(counts_1, counts_2, by='COUNTS')
> head(merged_counts)
```

	COUNTS	NB5	NB4	NB1	NB2	NB3	ENTREZ_ID	GENE_SYMBOL	Probe	ENSEMBL_GENE_ID	CB4	CB5	CB6	CB7	CB2	CB3	CB1	CB8
1	A1BG_25586	24	3	39	1	0	1	A1BG	Hs_A1BG_25586	ENSG00000121410	0	0	36	0	0	0	0	52
2	A1CF_87905	0	0	0	0	0	29974	A1CF	Hs_A1CF_87905	ENSG00000148584	25	0	0	0	19	0	0	0
3	A2M_1	0	0	0	187	0	2	A2M	Hs_A2M_1	ENSG00000175899	0	25	0	0	0	0	0	0
4	A2M_12371	31	0	82	603	1	2	A2M	Hs_A2M_12371	ENSG00000175899	16	1	0	0	22	0	0	0
5	A2ML1_18347	0	0	0	0	0	144568	A2ML1	Hs_A2ML1_18347	ENSG00000166535	0	0	0	0	0	0	0	0
6	A3GALT2_23042	0	0	0	0	0	127550	A3GALT2	Hs_A3GALT2_23042	ENSG00000184389	0	0	0	0	0	0	0	0

```
>
```

```r
SRR17172481 = read.csv('../SRR17172481/count_SRR81_SORTED_gene_name.txt', sep = '\t', header = F)
head(SRR17172481, n=10)

counts_main <- read.csv("Files/Mergecounts_CB1_CB8_NB1_NB5.csv", header = TRUE)
head(counts_main)
gene_names = counts_main[, 'COUNTS']
head(gene_names)
gene_names_2 = sapply(gene_names, function(n){strsplit(n, '_')[[1]][1]})
head(gene_names_2)
length(gene_names_2)
length(unique(gene_names_2))
# isoform
counts_main$gene_name_2 = gene_names_2
head(counts_main)

```

20:1 D NERGE

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

```
[1] 0.886874
> # isoform
> counts_main$gene_name_2 = gene_names_2
> head(counts_main)
    COUNTS CB1 CB2 CB3 CB4 CB5 CB6 CB7 CB8 NB1 NB2 NB3 NB4 NB5 gene_name_2
1    A2M_1   0   0   0   0  25   0   0   0   0 187   0   0   0         A2M
2   AAMP_2   0  48   0   0  64   0   0 128  64 384   0   1 132        AAMP
3   AATF_4  16   0   0   0   0   0   0   0   0  24  32   0   0        AATF
4 ABCB1_12   0   0   0   0   0   0   0  36   0   0   0   0   0       ABCB1
5 ABCC2_22   1   0   0   0   0   0  22   0   0   0   0   0  28       ABCC2
6 ABCC5_26   0  49  39  28  18  30  22  15 217 269  63  59 435       ABCC5
>
```

Then we compare these files with my count matrix SRR81-SRR85

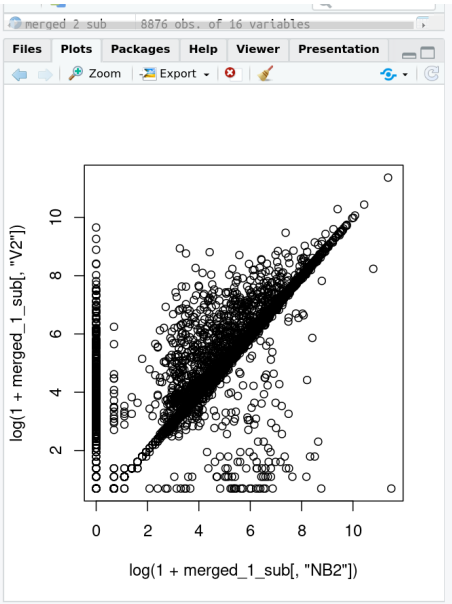
```r
head(counts_1)
merged_counts = merge(counts_1, counts_2, by='COUNTS')
write.csv(merged_counts, "Files/merged_all_counts__NB1_NB5_CB1_CB8.csv")
write.csv(merged_counts, "Data/merged_all_counts__NB1_NB5_CB1_CB8.csv")
head(merged_counts)
#D1. How many genes are given to edgeR?
nrow(merged_counts)#22339
cat("gene expression submatrix of the main study=>",nrow(merged_counts))
count_matrix_SRR81_SRR85<- read.table( file = "Files/Merge_count_matrix_SRR81_SRR85.txt", header = TRUE, row.names
# head(count_matrix_SRR81_SRR85)
cat("gene expression submatrix of SRR81 and SRR85=>",nrow(count_matrix_SRR81_SRR85))
```

```
> #D1. How many genes are given to edgeR?
> nrow(merged_counts)#22339
[1] 22339
> cat("gene expression submatrix of the main study=>",nrow(merged_counts))
gene expression submatrix of the main study=> 22339
> count_matrix_SRR81_SRR85<- read.table( file = "Files/Merge_count_matrix_SRR81_SRR85.txt", header = TRUE, row.names = 1)
> # head(count_matrix_SRR81_SRR85)
> cat("gene expression submatrix of SRR81 and SRR85=>",nrow(count_matrix_SRR81_SRR85))
gene expression submatrix of SRR81 and SRR85=> 64217
>
```

Combine **SRR81 count matrix** with **Main matrix** and create colomn gene_name2 and V2, by compairing colomn NB2 that is normal main file with V2 that is our SRR81 count file we have following results with **cor: 0.88 %**

```r
# Identify and remove duplicate values in SRR17172481
SRR17172481_nodup <- SRR17172481[!duplicated(SRR17172481$V1), ]
# Merge the two data frames using the cleaned-up columns
merged_1 <- merge(counts_main_nodup, SRR17172481_nodup, by.x='gene_name_2', by.y='V1')
head(merged_1)

merged_1_sub = merged_1 %>% filter(V2!=0)
head(merged_1_sub)
dim(merged_1_sub)

plot(log(1+merged_1_sub[, 'NB2']), log(1+merged_1_sub[, 'V2']), )
cor(x = merged_1_sub[, 'NB2'], y=merged_1_sub[, 'V2'], method = 'spearman')
```

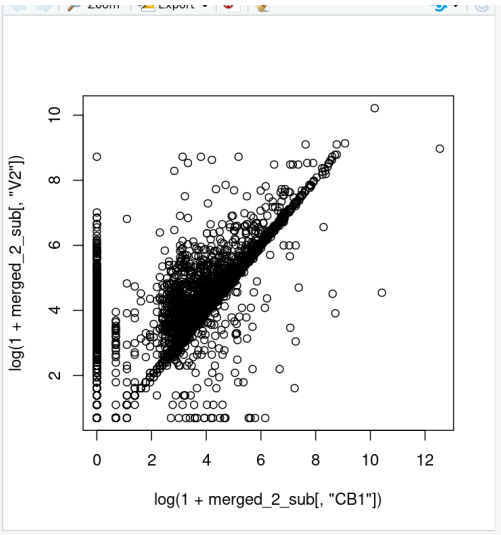
```
> head(merged_1_sub)
  gene_name_2       COUNTS CB1 CB2 CB3 CB4 CB5 CB6 CB7 CB8 NB1 NB2 NB3 NB4 NB5  V2
1         A2M         A2M_1   0   0   0   0  25   0   0   0   0 187   0   0   0 530
2      A4GALT A4GALT_88955   0  58   0   0  39  15   0  42  55 208   0 155 729 207
3       A4GNT  A4GNT_89151   0   0   0   0   0   0   0  25  11  28   0   0   0  20
4        AAAS   AAAS_23947  39   0  22   0   0   0   0  25   0  81   0   0  58  72
5        AACS   AACS_17112   0  43   0   0   0  14   0  16   0 108   0  44 182 121
6       AAGAB  AAGAB_23638  46   0  13  79 314  15   0 183  21 840   2 128 661 695
> dim(merged_1_sub)
[1] 9615   16
> plot(log(1+merged_1_sub[, 'NB2']), log(1+merged_1_sub[, 'V2']), )
> cor(x = merged_1_sub[, 'NB2'], y=merged_1_sub[, 'V2'], method = 'spearman')
[1] 0.886874
>
```



Combine **SRR85 count matrix** with **Main matrix** and create column gene_name2 and V2, by comparing column CB1 that is COVID main file with V2 that is our SRR85 count file we have following results with **cor: 0.72 %**

```
121 ########################################################################
122 merged_2 = merge(counts_main, SRR17172485, by.x='gene_name_2', by.y='V1')
123 head(merged_2)
124
125 merged_2_sub = merged_2 %>% filter(V2!=0)
126 head(merged_2_sub)
127 dim(merged_2_sub)
128
129 plot(log(1+merged_2_sub[, 'CB1']), log(1+merged_2_sub[, 'V2']), )
130 cor(x = merged_2_sub[, 'CB1'], y=merged_2_sub[, 'V2'], method = 'spearman')
131
132
```

```
Console  Terminal  Background Jobs
R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SE
  gene_name_2        COUNTS CB1 CB2 CB3 CB4 CB5 CB6 CB7 CB8 NB1 NB2 NB3 NB4 NB5 V2
1        AAAS    AAAS_23947  39   0  22   0   0   0   0  25   0  81   0   0  58 37
2       AADAC   AADAC_87800  21   0   0   0   0   0   0  27   0   0   0   0  25 21
3     AADACL3 AADACL3_28573  18   0   0   0   0   0   0  27   0   0   0   0   0 18
4       AAGAB   AAGAB_23638  46   0  13  79 314  15   0 183  21 840   2 128 661 44
5        AAR2    AAR2_25893  27   0   0   0   0  54   0 118   0   0   0   0   0 27
6       AASDH   AASDH_19738  36   0   0   0   0   0   0   0  27  11   0   0   0 60
> dim(merged_2_sub)
[1] 8876   16
> plot(log(1+merged_2_sub[, 'CB1']), log(1+merged_2_sub[, 'V2']), )
> cor(x = merged_2_sub[, 'CB1'], y=merged_2_sub[, 'V2'], method = 'spearman')
[1] 0.7237963
>
```



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

we can use different statistics and some plots such as correlation

```
# Get a list of the count files
count_files <- list.files("SRR17172481", pattern = "count_SRR81_SORTED.txt", full.names = TRUE)
# count_files <- list.files("SRR17172481", pattern = "count_SRR81__TRIM_SORTED.txt", full.names = TRUE)

count_files
# Load each count file into a separate data frame
count_data <- lapply(count_files, read.table, header = TRUE, stringsAsFactors = FALSE)
count_data
# Set the row names of each data frame to the gene IDs
count_data <- lapply(count_data, function(x) {row.names(x) <- x[,1]; x[,2]})
count_data
# Combine the data frames into a single matrix
count_matrix <- do.call(cbind, count_data)
count_matrix
# Write the count matrix to a text file
write.table(count_matrix, file = "SRR17172481/count_matrix_SRR81.txt", sep = "\t", quote = FALSE, row.names = TRUE)
# write.table(count_matrix, file = "SRR17172481/count_matrix_SRR81__TRIM.txt", sep = "\t", quote = FALSE, row.names = TRUE)
```

```
################################### merge ########################################################################
# Load the first count matrix
counts1 <- read.table("SRR17172481/count_matrix_SRR81.txt", header = TRUE, row.names = 1)
counts1
# Load the second count matrix
counts2 <- read.table("SRR17172485/count_matrix_SRR85.txt", header = TRUE, row.names = 1)
counts2
# Merge the two matrices column-wise
counts_merged <- cbind(counts1, counts2)
counts_merged
write.table(counts_merged, file = "Merge_count_matrix_SRR81_SRR85.txt", sep = "\t", quote = FALSE, row.names = TRUE)
```

```
#C-comparison
SRR17172481 = read.csv('../SRR17172481/count_SRR81_SORTED_gene_name.txt', sep = '\t', header = F)
head(SRR17172481, n=10)
# V1-> genename  V2:counts
#
# counts_main <- read.csv("Files/Mergecounts_CB1_CB8_NB1_NB5.csv", header = TRUE)
# head(counts_main)
```

```
# gene_name_2 -> removed numbers from gene
merged_1 = merge(counts_main, SRR17172481, by.x='gene_name_2', by.y='V1')
# Identify and remove duplicate values in counts_main
counts_main_nodup <- counts_main[!duplicated(counts_main$gene_name_2), ]

# Identify and remove duplicate values in SRR17172481
SRR17172481_nodup <- SRR17172481[!duplicated(SRR17172481$V1), ]
# Merge the two data frames using the cleaned-up columns
merged_1 <- merge(counts_main_nodup, SRR17172481_nodup, by.x='gene_name_2', by.y='V1')
head(merged_1)

merged_1_sub = merged_1 %>% filter(V2!=0)
head(merged_1_sub)
dim(merged_1_sub)

plot(log(1+merged_1_sub[, 'NB2']), log(1+merged_1_sub[, 'V2']), )
cor(x = merged_1_sub[, 'NB2'], y=merged_1_sub[, 'V2'], method = 'spearman')

#########################################################################
merged_2 = merge(counts_main, SRR17172485, by.x='gene_name_2', by.y='V1')
head(merged_2)

merged_2_sub = merged_2 %>% filter(V2!=0)
head(merged_2_sub)
dim(merged_2_sub)

plot(log(1+merged_2_sub[, 'CB1']), log(1+merged_2_sub[, 'V2']), )
cor(x = merged_2_sub[, 'CB1'], y=merged_2_sub[, 'V2'], method = 'spearman')
```

Part D- Differential gene expression analysis

Use edgeR to perform differential gene expression analysis. The edgeR is an open-source Bioconductor package designed for differential expression analysis based on count-based RNA-seq data. Use the expression matrix of the main study (all samples) to answer the following questions.

1. How many genes are given to edgeR? How many of them are differentially expressed in covid versus normal samples? How do you define statistical significance in this context?
2. Determine the percentage of differentially expressed genes with |log2FoldChange| > 1.5.
3. Explain the difference between P-value and FDR?

We have two file:

controls: ”GSE190496_Gene_counts_NB1_NB5.csv”

	A	B	C	D	E	F
1	COUNTS	NB5	NB4	NB1	NB2	NB3
2	A1BG_25586	24	3	39	1	0
3	A1CF_87905	0	0	0	0	0
4	A2M_1	0	0	0	187	0
5	A2M_12371	31	0	82	603	1
6	A2ML1_18347	0	0	0	0	0
7	A3GALT2_23042	0	0	0	0	0
8	A4GALT_88955	729	155	55	208	0
9	A4GNT_89151	0	0	11	28	0
10	AAAS_23947	58	0	0	81	0
11	AACS_17112	182	44	0	108	0
12	AADAC_87800	25	0	0	0	0
13	AADACL2_20478	0	0	0	0	0
14	AADACL3_28573	0	0	0	0	0
15	AADACL4_22336	0	0	0	0	0
16	AADAT_11332	18	0	0	0	0
17	AAGAB_23638	661	128	21	840	2
18	AAK1_10468	95	80	158	757	0
19	AAMDC_12899	34	0	70	74	0
20	AAMP_2	132	1	64	384	0
21	AANAT_91124	18	0	0	0	0
22	AAR2_25893	0	0	0	0	0
23	AARD_13511	0	0	0	0	0
24	AARS_3	265	32	57	666	35
25	AARS2_26243	142	2	80	98	35
26	AARSD1_26785	1	0	0	27	0
27	AASDH_19738	0	0	27	11	0
28	AASDH_26786	0	0	41	78	0
29	AASDHPPT_21936	284	69	95	290	0

covid: “GSE190496_Gene_counts_CB1_CB8.csv”

A1 f_x Σ =

	A	B	C	D	E	F	G	H	I	J	K	L
1					CB4	CB5	CB6	CB7	CB2	CB3	CB1	CB8
2	ENTREZ_ID	GENE_SYMBOL	Probe	ENSEMBL_GENE_ID								
3	2	A2M	Hs_A2M_1	ENSG00000175899	0	25	0	0	0	0	0	0
4	14	AAMP	Hs_AAMP_2	ENSG00000127837	0	64	0	0	48	0	0	128
5	16	AARS1	Hs_AARS1_3	ENSG00000090861	0	17	0	0	0	0	0	0
6	26574	AATF	Hs_AATF_4	ENSG00000275700	0	0	0	0	0	0	16	0
7	5243	ABCB1	Hs_ABCB1_12	ENSG00000085563	0	0	0	0	0	0	0	36
8	1244	ABCC2	Hs_ABCC2_22	ENSG00000023839	0	0	0	22	0	0	1	0
9	10057	ABCC5	Hs_ABCC5_26	ENSG00000114770	28	18	30	22	49	39	0	15
10	6059	ABCE1	Hs_ABCE1_32	ENSG00000164163	15	8	15	33	1	0	23	0
11	23	ABCF1	Hs_ABCF1_35	ENSG00000204574	0	59	0	24	0	0	88	155
12	55324	ABCF3	Hs_ABCF3_36	ENSG00000161204	0	0	0	0	0	0	0	0
13	9619	ABCG1	Hs_ABCG1_37	ENSG00000160179	0	0	0	0	0	0	0	9
14	8714	ABCC3	Hs_ABCC3_38	ENSG00000108846	0	0	0	0	18	0	0	0
15	9429	ABCG2	Hs_ABCG2_39	ENSG00000118777	48	0	0	11	0	0	104	32
16	9429	ABCG2	Hs_ABCG2_40	ENSG00000118777	0	0	19	0	0	0	0	0
17	9429	ABCG2	Hs_ABCG2_41	ENSG00000118777	0	0	0	0	0	0	28	0
18	57406	ABHD6	Hs_ABHD6_45	ENSG00000163686	0	0	0	4	0	0	22	0
19	25	ABL1	Hs_ABL1_46	ENSG00000097007	0	0	0	0	0	0	42	2
20	33	ACADL	Hs_ACADL_49	ENSG00000115361	0	0	0	1	0	0	0	0
21	39	ACAT2	Hs_ACAT2_56	ENSG00000120437	0	151	59	26	24	0	0	107
22	64746	ACBD3	Hs_ACBD3_59	ENSG00000182827	0	44	11	0	21	0	0	0
23	65057	ACD	Hs_ACD_60	ENSG00000102977	0	0	0	0	0	0	0	0
24	1636	ACE	Hs_ACE_61	ENSG00000159640	0	0	0	0	0	0	0	0

I tried to create **new colomn “COUNTS”** from “Probe” colomn.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1						CB4	CB5	CB6	CB7	CB2	CB3	CB1	CB8	
2	ENTREZ_ID	GENE_SYMBOL	Probe	ENSEMBL_GENE_ID	COUNTS									
3	2	A2M	Hs_A2M_1	ENSG00000175899	A2M_1	0	25	0	0	0	0	0	0	
4	14	AAMP	Hs_AAMP_2	ENSG00000127837	AAMP_2	0	64	0	0	48	0	0	128	
5	16	AARS1	Hs_AARS1_3	ENSG00000090861	AARS1_3	0	17	0	0	0	0	0	0	
6	26574	AATF	Hs_AATF_4	ENSG00000275700	AATF_4	0	0	0	0	0	0	16	0	
7	5243	ABCB1	Hs_ABCB1_12	ENSG00000085563	ABCB1_12	0	0	0	0	0	0	0	36	
8	1244	ABCC2	Hs_ABCC2_22	ENSG00000023839	ABCC2_22	0	0	0	22	0	0	1	0	
9	10057	ABCC5	Hs_ABCC5_26	ENSG00000114770	ABCC5_26	28	18	30	22	49	39	0	15	
10	6059	ABCE1	Hs_ABCE1_32	ENSG00000164163	ABCE1_32	15	8	15	33	1	0	23	0	
11	23	ABCF1	Hs_ABCF1_35	ENSG00000204574	ABCF1_35	0	59	0	24	0	0	88	155	
12	55324	ABCF3	Hs_ABCF3_36	ENSG00000161204	ABCF3_36	0	0	0	0	0	0	0	0	
13	9619	ABCG1	Hs_ABCG1_37	ENSG00000160179	ABCG1_37	0	0	0	0	0	0	0	9	
14	8714	ABCC3	Hs_ABCC3_38	ENSG00000108846	ABCC3_38	0	0	0	0	18	0	0	0	
15	9429	ABCG2	Hs_ABCG2_39	ENSG00000118777	ABCG2_39	48	0	0	11	0	0	104	32	
16	9429	ABCG2	Hs_ABCG2_40	ENSG00000118777	ABCG2_40	0	0	19	0	0	0	0	0	
17	9429	ABCG2	Hs_ABCG2_41	ENSG00000118777	ABCG2_41	0	0	0	0	0	0	28	0	
18	57406	ABHD6	Hs_ABHD6_45	ENSG00000163686	ABHD6_45	0	0	0	4	0	0	22	0	
19	25	ABL1	Hs_ABL1_46	ENSG00000097007	ABL1_46	0	0	0	0	0	0	42	2	
20	33	ACADL	Hs_ACADL_49	ENSG00000115361	ACADL_49	0	0	0	1	0	0	0	0	
21	39	ACAT2	Hs_ACAT2_56	ENSG00000120437	ACAT2_56	0	151	59	26	24	0	0	107	
22	64746	ACBD3	Hs_ACBD3_59	ENSG00000182827	ACBD3_59	0	44	11	0	21	0	0	0	
23	65057	ACD	Hs_ACD_60	ENSG00000102977	ACD_60	0	0	0	0	0	0	0	0	
24	1636	ACE	Hs_ACE_61	ENSG00000159640	ACE_61	0	0	0	0	0	0	0	0	
25	43	ACHE	Hs_ACHE_63	ENSG00000087085	ACHE_63	0	0	0	0	0	0	0	13	
26	47	ACLY	Hs_ACLY_66	ENSG00000131473	ACLY_66	0	24	0	0	0	0	25	54	
27	23597	ACOT9	Hs_ACOT9_69	ENSG00000123130	ACOT9_69	0	17	0	0	0	0	0	0	
28	2180	ACSL1	Hs_ACSL1_75	ENSG00000151726	ACSL1_75	328	1437	113	152	382	20	271	1291	
29	60	ACTB	Hs_ACTB_80	ENSG00000075624	ACTB_80	336	2535	503	510	405	31	311	644	
30		ACTBP9	Hs_ACTBP9_82	ENSG00000266920	ACTBP9_82	0	34	17	13	0	10	16	27	
31	88	ACTN2	Hs_ACTN2_90	ENSG00000077522	ACTN2_90	0	0	0	0	0	0	0	0	
32	55860	ACTR10	Hs_ACTR10_93	ENSG00000131966	ACTR10_93	17	53	1	0	0	0	39	13	
33	102	ADAM10	Hs_ADAM10_102	ENSG00000137845	ADAM10_102	125	364	0	0	21	0	17	93	
34	8745	ADAM22	Hs_ADAM22_104	ENSG00000114948	ADAM22_104	0	0	0	0	0	0	0	0	

```
> counts_2 = read.csv('../GSE96/DATA/GSE190496_Gene_counts_CB1_CB8.csv')
> head(counts_2)
          X         X.1         X.2             X.3      X.4 CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8 X.5
1 ENTREZ_ID GENE_SYMBOL       Probe ENSEMBL_GENE_ID   COUNTS  NA  NA  NA  NA  NA  NA  NA  NA  NA
2         2         A2M    Hs_A2M_1 ENSG00000175899    A2M_1   0  25   0   0   0   0   0   0  NA
3        14        AAMP   Hs_AAMP_2 ENSG00000127837    AAMP_2   0  64   0   0  48   0   0 128  NA
4        16       AARS1  Hs_AARS1_3 ENSG00000090861   AARS1_3   0  17   0   0   0   0   0   0  NA
5     26574        AATF   Hs_AATF_4 ENSG00000275700    AATF_4   0   0   0   0   0   0  16   0  NA
6      5243       ABCB1 Hs_ABCB1_12 ENSG00000085563 ABCB1_12   0   0   0   0   0   0   0  36  NA
> counts_1 = read.csv('../GSE96/DATA/GSE190496_Gene_counts_NB1_NB5.csv')
> head(counts_1)
         COUNTS NB5 NB4 NB1 NB2 NB3
1    A1BG_25586  24   3  39   1   0
2    A1CF_87905   0   0   0   0   0
3         A2M_1   0   0   0 187   0
4     A2M_12371  31   0  82 603   1
5   A2ML1_18347   0   0   0   0   0
6 A3GALT2_23042   0   0   0   0   0
>
```

Now we can merge these two files based on common colomns “COUNTS” and create a matrix with all control and covid samples.

```
counts_2 = read.csv('../GSE96/DATA/GSE190496_Gene_counts_CB1_CB8.csv')
head(counts_2)
colnames(counts_2) = c(counts_2[1, 1:5], colnames(counts_2)[6:13])
dim(counts_2)
head(counts_2)
counts_2 = counts_2[-1, ]
head(counts_2)
counts_2 = counts_2[, -14]
head(counts_2)
counts_1 = read.csv('../GSE96/DATA/GSE190496_Gene_counts_NB1_NB5.csv')
head(counts_1)
merged_counts = merge(counts_1, counts_2, by='COUNTS')
head(merged_counts)
write.csv(merged_counts, "Files/merged_all_counts__NB1_NB5_CB1_CB8.csv")
```

21:26 (Untitled)

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/RNA_SEQ/

```
> head(merged_counts)
        COUNTS NB5 NB4 NB1 NB2 NB3 ENTREZ_ID GENE_SYMBOL           Probe ENSEMBL_GENE_ID CB4 CB5 CB6 CB7
1    A1BG_25586  24   3  39   1   0         1        A1BG    Hs_A1BG_25586 ENSG00000121410   0   0  36   0
2    A1CF_87905   0   0   0   0   0     29974        A1CF    Hs_A1CF_87905 ENSG00000148584  25   0   0   0
3         A2M_1   0   0   0 187   0         2         A2M         Hs_A2M_1 ENSG00000175899   0  25   0   0
4     A2M_12371  31   0  82 603   1         2         A2M     Hs_A2M_12371 ENSG00000175899  16   1   0   0
5   A2ML1_18347   0   0   0   0   0    144568       A2ML1   Hs_A2ML1_18347 ENSG00000166535   0   0   0   0
6 A3GALT2_23042   0   0   0   0   0    127550     A3GALT2 Hs_A3GALT2_23042 ENSG00000184389   0   0   0   0
  CB2 CB3 CB1 CB8
1   0   0   0  52
2  19   0   0   0
```

1.How many genes are given to edgeR?

nrow(merged_counts)# “**22339**” genes are given to edgeR

```
merged_counts = merge(counts_1, counts_2, by='COUNTS')
head(merged_counts)
# How many genes are given to edgeR?
nrow(merged_counts)
```

33:20 (Untitled)

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/RNA_SEQ/

```
> head(merged_counts)
        COUNTS NB5 NB4 NB1 NB2 NB3 ENTREZ_ID GENE_SYMBOL           Probe ENSEMBL_GENE_ID CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1    A1BG_25586  24   3  39   1   0         1        A1BG    Hs_A1BG_25586 ENSG00000121410   0   0  36   0   0   0   0  52
2    A1CF_87905   0   0   0   0   0     29974        A1CF    Hs_A1CF_87905 ENSG00000148584  25   0   0   0  19   0   0   0
3         A2M_1   0   0   0 187   0         2         A2M         Hs_A2M_1 ENSG00000175899   0  25   0   0   0   0   0   0
4     A2M_12371  31   0  82 603   1         2         A2M     Hs_A2M_12371 ENSG00000175899  16   1   0   0  22   0   0   0
5   A2ML1_18347   0   0   0   0   0    144568       A2ML1   Hs_A2ML1_18347 ENSG00000166535   0   0   0   0   0   0   0   0
6 A3GALT2_23042   0   0   0   0   0    127550     A3GALT2 Hs_A3GALT2_23042 ENSG00000184389   0   0   0   0   0   0   0   0
> # How many genes are given to edgeR?
> nrow(merged_counts)
[1] 22339
>
```

1. How many of them are differentially expressed in covid versus normal samples?

with cpm for normal, keep <- rowSums(cpm(y[, group == "normal"]) >= 1) >= 3 :

with p$table$FDR < 0.05 , Number of differentially expressed genes: 345

with p$table$PValue < 0.05, Number of differentially expressed genes: 1300

With **cpm** for both normal and covid # keep <- rowSums(cpm(y) >= 1) >= n_samples_per_group

Number of differentially expressed genes FDR: 740

Number of differentially expressed genes PValue: 1532

With **cpm** for both normal

```r
# Calculate the adjusted p-value or FDR using the Benjamini-Hochberg method
p$table$FDR <- p.adjust(p$table$PValue, method = "BH")
head(p$table)
n_genes <- sum(p$table$FDR < 0.05 )
cat("Number of differentially expressed genes:", n_genes, "\n")
n_genes_pval <- sum(p$table$PValue < 0.05 )
cat("Number of differentially expressed genes:", n_genes_pval, "\n")
```

42:41 (Top Level)

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data pre

```
> # ===
> # Calculate the adjusted p-value or FDR using the Benjamini-Hochberg method
> p$table$FDR <- p.adjust(p$table$PValue, method = "BH")
> head(p$table)
        logFC   logCPM          F    PValue       FDR
1  -1.5871687 3.864233 0.55701164 0.4554683 0.8587060
4   1.9018741 4.386053 0.98614603 0.3206882 0.7807979
7   1.5707142 5.526700 1.08409204 0.2977861 0.7633686
9  -1.7796541 3.983381 0.69090204 0.4058595 0.8324445
10  0.3522970 4.273847 0.03260178 0.8567136 0.9795567
16  0.1264678 6.442037 0.01004239 0.9201764 0.9895555
> n_genes <- sum(p$table$FDR < 0.05 )
> cat("Number of differentially expressed genes:", n_genes, "\n")
Number of differentially expressed genes: 345
> n_genes_pval <- sum(p$table$PValue < 0.05 )
> cat("Number of differentially expressed genes:", n_genes_pval, "\n")
Number of differentially expressed genes: 1300
```

With **cpm** for both normal and covid

```r
# Calculate the adjusted p-value or FDR using the Benjamini-Hochberg method
p$table$FDR <- p.adjust(p$table$PValue, method = "BH")
head(p$table)
n_genes <- sum(p$table$FDR < 0.05 )
cat("Number of differentially expressed genes FDR:", n_genes, "\n")
n_genes_pval <- sum(p$table$PValue < 0.05 )
cat("Number of differentially expressed genes PValue:", n_genes_pval, "\n")
```

40:1 # exactTest

Console | Terminal | Render | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Dat

```
> head(p$table)
         logFC   logCPM           F    PValue       FDR
1  -1.67047375 3.925512 0.619813455 0.4311183 0.8313202
4   1.80101719 4.380785 0.914040183 0.3390468 0.7694629
7   1.50231671 5.538668 1.021796487 0.3120955 0.7450408
10  0.26583023 4.303716 0.019177075 0.8898603 0.9822185
16  0.06202358 6.454334 0.002476756 0.9603081 0.9946831
17  1.02478643 5.572694 0.483727300 0.4867403 0.8568486
> n_genes <- sum(p$table$FDR < 0.05 )
> cat("Number of differentially expressed genes FDR:", n_genes, "\n")
Number of differentially expressed genes FDR: 740
> n_genes_pval <- sum(p$table$PValue < 0.05 )
> cat("Number of differentially expressed genes PValue:", n_genes_pval, "\n")
Number of differentially expressed genes PValue: 1532
>
```

```r
#Filename=> D_Merg_covid_normal.R
# ============D MERGE ===================================
counts_2 = read.csv('../GSE96/DATA/GSE190496_Gene_counts_CB1_CB8.csv')
head(counts_2)
colnames(counts_2) = c(counts_2[1, 1:5], colnames(counts_2)[6:13])
dim(counts_2)
head(counts_2)
counts_2 = counts_2[-1, ]
head(counts_2)
counts_2 = counts_2[, -14]
head(counts_2)
counts_1 = read.csv('../GSE96/DATA/GSE190496_Gene_counts_NB1_NB5.csv')
head(counts_1)
merged_counts = merge(counts_1, counts_2, by='COUNTS')
write.csv(merged_counts, "Files/merged_all_counts__NB1_NB5_CB1_CB8.csv")
write.csv(merged_counts, "Data/merged_all_counts__NB1_NB5_CB1_CB8.csv")
#####################################################################
#D1. How many genes are given to edgeR?
#####################################################################
head(merged_counts)
nrow(merged_counts)#22339
#####################################################################
#D1. how many of them are differentially expressed in covid versus normal samples?
#####################################################################
library(limma)
library(edgeR)
new_df <- merged_counts[, c("NB5", "NB4", "NB1", "NB2", "NB3", "CB4", "CB5", "CB6", "CB7", "CB2", "CB3", "CB1", "CB8")]
head(new_df)
y <- DGEList(counts = new_df) # NA error
# --------Remove NA ----------------
# Find and insert 0 insted of NA
which(is.na(new_df))
```

```
# [1] 113383 122287 126037
new_df[is.na(new_df)] <- 0
which(is.na(new_df))
head(new_df)
# --------------------
y <- DGEList(counts = new_df) # NA error
head(y)
# Create a vector of sample names for normal and COVID groups
normal_samples <- colnames(new_df)[1:5]
covid_samples <- colnames(new_df)[6:13]

# Create a factor variable for the groups
group <- factor(c(rep("normal", length(normal_samples)), rep("covid", length(covid_samples))))
names(group) <- c(normal_samples, covid_samples)

y <- DGEList(counts = new_df, group = group)
head(y)
### cpm Filter normal ##############
# cpm/// Filter normal the genes that have low expression and/or low variability
keep <- rowSums(cpm(y[, group == "normal"]) >= 1) >= 3
# cpm/// Filter normal and covid the genes that have low expression and/or low variability
# n_samples_normal <- ncol(counts_subset[,1:5])
# n_samples_covid <- ncol(counts_subset[,6:13])
# n_samples_per_group <- min(n_samples_normal, n_samples_covid)
# keep <- rowSums(cpm(y) >= 1) >= n_samples_per_group
####################################
y <- y[keep, , keep.lib.sizes = FALSE]
y <- calcNormFactors(y)
design <- model.matrix(~group)
y <- estimateDisp(y, design)
fit <- glmQLFit(y, design)
p <- glmQLFTest(fit, coef = 2)
head(p)
# Calculate the adjusted p-value or FDR using the Benjamini-Hochberg method
p$table$FDR <- p.adjust(p$table$PValue, method = "BH")
head(p$table)
n_genes <- sum(p$table$FDR < 0.05 )
cat("Number of differentially expressed genes based on FDR:", n_genes, "\n")
n_genes_pval <- sum(p$table$PValue < 0.05 )
cat("Number of differentially expressed genes based on PValue:", n_genes_pval, "\n")
#########################################################################
# Get the differentially expressed genes with |log2FoldChange| > 1.5
#########################################################################
de_genes <- rownames(p$table)[abs(p$table$logFC) > 1.5 & p$table$FDR < 0.05]
# Calculate the percentage of differentially expressed genes with |log2FoldChange| > 1.5
perc_de_genes <- length(de_genes) / nrow(p$table) * 100
cat("Percentage of differentially expressed genes with |log2FoldChange| > 1.5: ", round(perc_de_genes, 2), "%\n")
```

```r
# Calculate the adjusted p-value or FDR using the Benjamini-Hochberg method
p$table$FDR <- p.adjust(p$table$PValue, method = "BH")
head(p$table)
n_genes <- sum(p$table$FDR < 0.05 )
cat("Number of differentially expressed genes based on FDR:", n_genes, "\n")
n_genes_pval <- sum(p$table$PValue < 0.05 )
cat("Number of differentially expressed genes based on PValue:", n_genes_pval, "\n")
##########################################################################
# Get the differentially expressed genes with |log2FoldChange| > 1.5
##########################################################################
de_genes <- rownames(p$table)[abs(p$table$logFC) > 1.5 & p$table$FDR < 0.05]
# Calculate the percentage of differentially expressed genes with |log2FoldChange| > 1.5
perc_de_genes <- length(de_genes) / nrow(p$table) * 100
cat("Percentage of differentially expressed genes with |log2FoldChange| > 1.5: ", round(perc_de_genes, 2), "%\n")
```

9:27 D MERGE

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

```
> n_genes <- sum(p$table$FDR < 0.05 )
> cat("Number of differentially expressed genes based on FDR:", n_genes, "\n")
Number of differentially expressed genes based on FDR: 740
> n_genes_pval <- sum(p$table$PValue < 0.05 )
> cat("Number of differentially expressed genes based on PValue:", n_genes_pval, "\n")
Number of differentially expressed genes based on PValue: 1532
> ##########################################################################
> # Get the differentially expressed genes with |log2FoldChange| > 1.5
> de_genes <- rownames(p$table)[abs(p$table$logFC) > 1.5 & p$table$FDR < 0.05]
> # Calculate the percentage of differentially expressed genes with |log2FoldChange| > 1.5
> perc_de_genes <- length(de_genes) / nrow(p$table) * 100
> cat("Percentage of differentially expressed genes with |log2FoldChange| > 1.5: ", round(perc_de_genes, 2), "%\n")
Percentage of differentially expressed genes with |log2FoldChange| > 1.5:  7.33 %
>
```

1. How do you define statistical significance in this context?

I used **Pvalue** and **FDR** with thereshould of **0.05**

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

with cpm for normal:

Percentage of differentially expressed genes with |log2FoldChange| > 1.5: 7.33 %

cpm normal

```r
##########################################################################
# Get the differentially expressed genes with |log2FoldChange| > 1.5
de_genes <- rownames(p$table)[abs(p$table$logFC) > 1.5 & p$table$FDR < 0.05]

# Calculate the percentage of differentially expressed genes with |log2FoldChange| > 1.5
perc_de_genes <- length(de_genes) / nrow(p$table) * 100
cat("Percentage of differentially expressed genes with |log2FoldChange| > 1.5: ", round(perc_de_genes, 2), "%\n")

```

74:1 (Untitled)

Console | Terminal | Render | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/RNA_SEQ/

```
> cat("Percentage of differentially expressed genes with |log2FoldChange| > 1.5: ", round(perc_de_genes, 2), "%\n")
Percentage of differentially expressed genes with |log2FoldChange| > 1.5:  7.33 %
>
```

cpm normal-covid

```
91  ######################### E ##########################################################################################
92  # To accomplish this step, select the genes with FDR < 0.1 and an absolute value of Log2FoldChange > 1.5.
93  # Get the differentially expressed genes with |log2FoldChange| > 1.5
94  E_de_genes <- rownames(p$table)[abs(p$table$logFC) > 1.5 & p$table$FDR < 0.1]
95  head(E_de_genes)
96  # Calculate the percentage of differentially expressed genes with |log2FoldChange| > 1.5
97  perc_E_de_genes <- length(E_de_genes) / nrow(p$table) * 100
98  cat("Percentage of differentially expressed genes with |log2FoldChange| > 1.5 and FDR<0.1: ", round(perc_E_de_genes, 2), "%\n")
99
100
92:67  # E
```

```
Console  Terminal  Render  Background Jobs
R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/RNA_SEQ/
> cat("Percentage of differentially expressed genes with |log2FoldChange| > 1.5 and FDR<0.1: ", round(perc_E_de_genes, 2), "%\n")
Percentage of differentially expressed genes with |log2FoldChange| > 1.5 and FDR<0.1:  9 %
>
```

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

P-value and FDR are both statistical measures used in hypothesis testing to evaluate the significance of a result. However, they differ in their interpretation and the type of error they control.

P-value is defined as the **probability of obtaining a test statistic as extreme or more extreme than the observed statistic**, assuming that the null hypothesis is true. In other words, it measures the strength of evidence against the null hypothesis. A small P-value (typically less than 0.05) indicates that the observed result is unlikely to have occurred by chance alone, and therefore supports the rejection of the null hypothesis in favor of the alternative hypothesis. However, P-value does not control the rate of false positives (Type I errors), which is the probability of rejecting the null hypothesis when it is actually true. Therefore, multiple hypothesis tests with small P-values can lead to an inflated false positive rate.

FDR (False Discovery Rate), on the other hand, is defined as the expected proportion of false positives among all the significant results. It controls the rate of false positives (Type I errors) and is a more conservative measure compared to P-value. FDR is calculated by adjusting the P-values using a correction method such as the Benjamini-Hochberg procedure. By controlling the FDR at a
specific level (typically 0.05), we can limit the number of false positives among the significant results. Therefore, FDR is widely used in multiple hypothesis testing scenarios where a large number of tests are performed simultaneously.

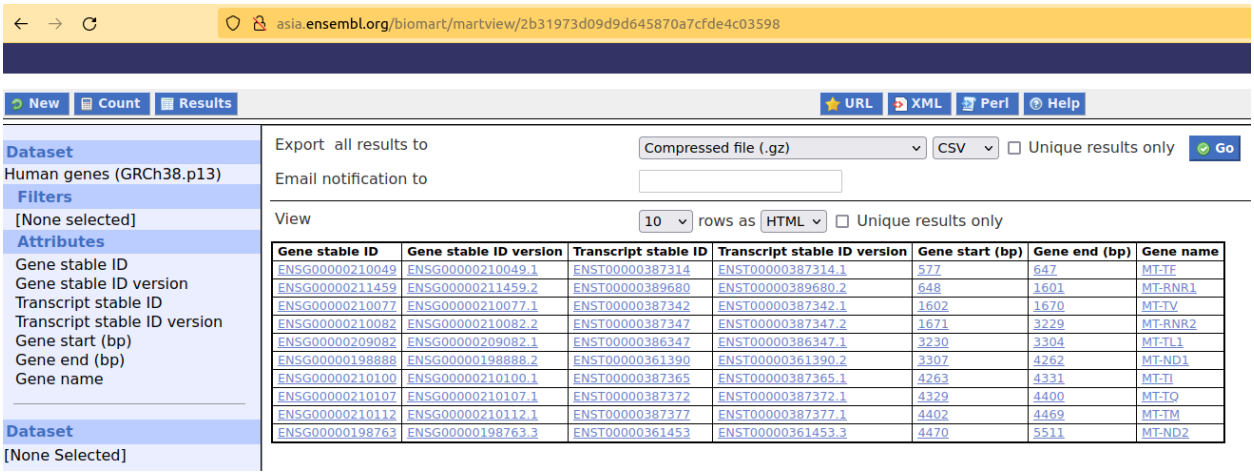
Part e- Gene Ontology enrichment analysis

In the final step, to obtain the distinct biological functions present in cancer samples, use the GOseq package in R to perform Gene Ontology (GO) enrichment analysis. The GO enrichment analysis statistically assesses the overrepresentation of differentially expressed genes in common GO functional branches. To accomplish this step, select the genes with FDR < 0.1 and an absolute value of Log2FoldChange > 1.5. Please answer the following question after completing this step.

1. Display results related to Biological Process, Molecular Function, Cellular Component, and KEGG as separate plots using an R package of your choice.
2. Do a brief study of each of the significant terms and discuss which terms you think may play an important role.
3. Write a general biological conclusion about the final results of the project.

For this step we need gene length, so we we calculate it from website. this file contain these titles: **Gene stable ID, Gene stable ID version, Transcript stable ID, Transcript stable ID version, Gene start (bp), Gene end (bp), Gene name.**

http://asia.ensembl.org/biomart/martview/2b31973d09d9d645870a7cfde4c03598



Gene stable ID	Gene stable ID version	Transcript stable ID	Transcript stable ID version	Gene start (bp)	Gene end (bp)	Gene name
ENSG00000210049	ENSG00000210049.1	ENST00000387314	ENST00000387314.1	577	647	MT-TF
ENSG00000211459	ENSG00000211459.2	ENST00000389680	ENST00000389680.2	648	1601	MT-RNR1
ENSG00000210077	ENSG00000210077.1	ENST00000387342	ENST00000387342.1	1602	1670	MT-TV
ENSG00000210082	ENSG00000210082.2	ENST00000387347	ENST00000387347.2	1671	3229	MT-RNR2
ENSG00000209082	ENSG00000209082.1	ENST00000386347	ENST00000386347.1	3230	3304	MT-TL1
ENSG00000198888	ENSG00000198888.2	ENST00000361390	ENST00000361390.2	3307	4262	MT-ND1
ENSG00000210100	ENSG00000210100.1	ENST00000387365	ENST00000387365.1	4263	4331	MT-TI
ENSG00000210107	ENSG00000210107.1	ENST00000387372	ENST00000387372.1	4329	4400	MT-TQ
ENSG00000210112	ENSG00000210112.1	ENST00000387377	ENST00000387377.1	4402	4469	MT-TM
ENSG00000198763	ENSG00000198763.3	ENST00000361453	ENST00000361453.3	4470	5511	MT-ND2

Merge output file with our merged file based on common rows of <counts['ENSEMBL_GENE_ID']> and <mart_export['Gene.stable.ID']>.

Then we calculate new colomn “Gene.length” by subtraction of two colomns:

merged_data$Gene.length <- merged_data$Gene.end..bp. - merged_data$Gene.start..bp.

```
########################## E ############################################################3
# Read in the two CSV files
counts <- read.csv("../GSE96/DATA/merged_all_counts__NB1_NB5_CB1_CB8.csv", header = TRUE, row.names = 1)
counts['ENSEMBL_GENE_ID']
mart_export <- read.csv("../GSE96/DATA/mart_export.csv", header = TRUE)
mart_export['Gene.stable.ID']
# Merge the two data frames based on the common columns
merged_data <- merge(counts, mart_export, by.x = "ENSEMBL_GENE_ID", by.y = "Gene.stable.ID", all.x = TRUE)
head(merged_data)
# Create a new column for gene length
merged_data$Gene.length <- merged_data$Gene.end..bp. - merged_data$Gene.start..bp.
# View the first few rows of the updated data frame
head(merged_data)
# Save the merged data frame to a new CSV file
write.csv(merged_data, file = "../GSE96/DATA/Gene_Length_NB1_NB5_CB1_CB8.csv", row.names = TRUE)
write.csv(merged_data, file="Files/Gene_Length_NB1_NB5_CB1_CB8.csv", row.names=FALSE)

################################################
```

```r
# Read in the two CSV files
counts <- read.csv("../GSE96/DATA/merged_all_counts__NB1_NB5_CB1_CB8.csv", header = TRUE, row.names = 1)
counts['ENSEMBL_GENE_ID']
mart_export <- read.csv("../GSE96/DATA/mart_export.csv", header = TRUE)
mart_export['Gene.stable.ID']
# Merge the two data frames based on the common columns
merged_data <- merge(counts, mart_export, by.x = "ENSEMBL_GENE_ID", by.y = "Gene.stable.ID", all.x = TRUE)
head(merged_data)
# Create a new column for gene length
merged_data$Gene.length <- merged_data$Gene.end..bp. - merged_data$Gene.start..bp.
# View the first few rows of the updated data frame
head(merged_data)
# Save the merged data frame to a new CSV file
write.csv(merged_data, file = "../GSE96/DATA/mart_export_merged_NB1_NB5_CB1_CB8.csv", row.names = TRUE)

```

90:104 (Untitled)

Console | Terminal | Render | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/RNA_SEQ/

```
> head(merged_data)
  ENSEMBL_GENE_ID       COUNTS NB5 NB4 NB1 NB2 NB3 ENTREZ_ID GENE_SYMBOL           Probe CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1                 PRR21_17743   0   0   0   0   0      NULL       PRR21  Hs_PRR21_17743   0   5  82  47  22   0   0 108
2 ENSG00000000003 TSPAN6_89866 483  40   0  86   0      7105      TSPAN6 Hs_TSPAN6_89866   0   0   0   0  21   0   0   0
3 ENSG00000000003 TSPAN6_89866 483  40   0  86   0      7105      TSPAN6 Hs_TSPAN6_89866   0   0   0   0  21   0   0   0
4 ENSG00000000003 TSPAN6_89866 483  40   0  86   0      7105      TSPAN6 Hs_TSPAN6_89866   0   0   0   0  21   0   0   0
5 ENSG00000000003 TSPAN6_89866 483  40   0  86   0      7105      TSPAN6 Hs_TSPAN6_89866   0   0   0   0  21   0   0   0
6 ENSG00000000003 TSPAN6_89866 483  40   0  86   0      7105      TSPAN6 Hs_TSPAN6_89866   0   0   0   0  21   0   0   0
  Gene.stable.ID.version Transcript.stable.ID Transcript.stable.ID.version Gene.start..bp. Gene.end..bp. Gene.name Gene.length
1                   <NA>                <NA>                          <NA>             NA            NA      <NA>          NA
2     ENSG00000000003.15      ENST00000496771            ENST00000496771.5       100627108     100639991    TSPAN6       12883
3     ENSG00000000003.15      ENST00000614008            ENST00000614008.4       100627108     100639991    TSPAN6       12883
4     ENSG00000000003.15      ENST00000612152            ENST00000612152.4       100627108     100639991    TSPAN6       12883
5     ENSG00000000003.15      ENST00000373020            ENST00000373020.9       100627108     100639991    TSPAN6       12883
6     ENSG00000000003.15      ENST00000494424            ENST00000494424.1       100627108     100639991    TSPAN6       12883
>
```

Explain some items.

The input for `goseq` is a vector that indicates, for **each gene**, **whether or not it is significantly differentially expressed**.

This should be **a named vector,** where the **names are the gene ids** and the **values are** `1` **if the gene is significant and** `0` **if it is not.** In this case we can use the **Ensembl gene IDs**.

DGELIST Inputs

The `group` argument specifies the grouping variable that indicates the experimental condition or treatment of each sample.

The resulting `y` object has two components:

1. `y$counts`: a matrix of raw count data, where each row corresponds to a gene and each column corresponds to a sample. Each element in the matrix represents the number of reads
   that map to a particular gene in a particular sample.
2. `y$samples`: a data frame that provides metadata about each sample, including the experimental group, library size (total number of reads in each sample), and normalization factors used to adjust for differences in sequencing depth and composition.

```
> y <- DGEList(counts = merge_subset, group = group)
> head(y)
An object of class "DGEList"
$counts
  NB5 NB4 NB1 NB2 NB3 CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1   0   0   0   0   0   0   5  82  47  22   0   0 108
2 483  40   0  86   0   0   0   0   0  21   0   0   0
3 483  40   0  86   0   0   0   0   0  21   0   0   0
4 483  40   0  86   0   0   0   0   0  21   0   0   0
5 483  40   0  86   0   0   0   0   0  21   0   0   0
6 483  40   0  86   0   0   0   0   0  21   0   0   0

$samples
     group lib.size norm.factors
NB5 normal 57227082            1
NB4 normal  8369642            1
NB1 normal 21206839            1
NB2 normal 74687107            1
NB3 normal  3782208            1
8 more rows ...

> dim(y)
[1] 211956     13
```

$samples
group lib.size norm.factors

1. `group`: This column indicates the experimental group or condition that each sample belongs to. In this case, all the samples are from the "normal" group.
2. `lib.size`: This column indicates the library size of each sample, which is the total number of reads obtained from sequencing the sample. In this case, the library sizes range from about 3.5 million reads to over 70 million reads.
3. `norm.factors`: This column indicates the normalization factors that were calculated for each sample to adjust for differences in sequencing depth and composition. The normalization factors are relative scaling factors that adjust the raw read counts for each sample such that they are comparable across samples. In this case, the normalization factors range from about 0.79 to 1.52, indicating that some samples had higher sequencing depth than others.

```
fit <- glmQLFit(y, design)
```

The code snippet provided fits a GLM to the RNA-seq data in the object `y` using the quasi-likelihood method implemented in the R package `edgeR`. The `glmQLFit()` function fits a GLM to the data and estimates the dispersion parameters using a quasi-likelihood approach that accounts for overdispersion and other sources of variability in the RNA-seq data.

The first argument to `glmQLFit()` is `y`, which is an object of class "DGEList" that contains the RNA-seq count data and associated metadata, including library size and normalization factors. The second argument, `design`, is a matrix that specifies the experimental design and the variables to be included in the GLM. Each row of the `design` matrix represents a sample in the experiment, and each column represents a predictor variable or covariate to be included in the model. The `design` matrix must have a column for each group or condition to be compared in the differential gene expression analysis.

The `glmQLFTest()` function is used to perform hypothesis tests for differential expression between two groups, normal and covid, in this case. The `glmQLFTest()` function takes as input the fitted GLM object from `glmQLFit()` (`fit` in this case) and the index of the coefficient of interest (which is the second coefficient in the design matrix, corresponding to the covid group).

The output of `glmQLFTest()` is an object of class "topTags" that contains the results of the hypothesis test, including the log-fold changes, standard errors, and p-values for each gene. By default, `glmQLFTest()` performs a two-sided test and calculates p-values using a Wald test.

You can also specify other types of tests and adjust the p-values for multiple testing using the `contrast` and `adjust.method` arguments, respectively.

```
> head(obj_topTags$table)
        logFC   logCPM           F    PValue
9  -0.1216136 2.710031 0.008612991 0.9262738
10 -0.1216136 2.710031 0.008612991 0.9262738
11 -0.1216136 2.710031 0.008612991 0.9262738
12 -0.1216136 2.710031 0.008612991 0.9262738
13 -0.1216136 2.710031 0.008612991 0.9262738
14 -0.1216136 2.710031 0.008612991 0.9262738
>
```

==========

```
32 # Define the experimental design and create the DGEList object
33 group <- factor(c(rep("normal", ncol(uniquegenes__merge_subset[,c(1:5)])), rep("covid", ncol(uniquegenes__merge_subset[,c(6:13)]))))
34 head(group)
35 prep_deg <- DGEList(counts = uniquegenes__merge_subset, group = group)
36 head(prep_deg)
37 dim(prep_deg)
38 ### cpm Filter normal ##############
39 # cpm/// Filter normal the genes that have low expression and/or low variability
29:32  (Untitled)                                                                R Sc

Console  Terminal  Background Jobs
R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/
> head(prep_deg)
An object of class "DGEList"
$counts
   NB5 NB4 NB1 NB2 NB3 CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1    0   0   0   0   0   0   5  82  47  22   0   0 108
2  483  40   0  86   0   0   0   0   0  21   0   0   0
7    0   0   0   0   0   0  20   0   0   0   0   0   0
9  715   0  97 766   1  15 178  20  83  33   0  45  79
25 105  44  69  52   0   0  23   0   0   0   0   0   0
30  78   0   0  34   0   0   0   0   0   0   0   0   0

$samples
     group lib.size norm.factors
NB5 normal  5137371            1
NB4 normal   849862            1
NB1 normal  1738730            1
NB2 normal  5998439            1
NB3 normal   383386            1
8 more rows ...
```

The `DGEList()` function is part of the `edgeR` package and is used to create a `DGEList` object, which is a specialized data structure used for differential gene expression analysis.

The `DGEList` object contains the **count data**, the **experimental group assignments**, and other **metadata**, and is used as **input to other functions** in the `edgeR` package for:

data normalization, **dispersion estimation**, and **hypothesis testing**.

At first we open our merged normal and covid file(merge was based on count colomn)

```
 9  ########################## E ######################################################
10  # Read in the CSV files
11  merged_data <- read.csv("../GSE96/DATA/merged_all_counts__NB1_NB5_CB1_CB8.csv", header = TRUE, row.names = 1)
12  head(merged_data)
13  dim(merged_data)#22339    18
15:1  # E
```

```
Console  Terminal  Background Jobs
R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/
> ########################## E ######################################################
> # Read in the CSV files
> merged_data <- read.csv("../GSE96/DATA/merged_all_counts__NB1_NB5_CB1_CB8.csv", header = TRUE, row.names = 1)
> head(merged_data)
         COUNTS NB5 NB4 NB1 NB2 NB3 ENTREZ_ID GENE_SYMBOL            Probe ENSEMBL_GENE_ID CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1    A1BG_25586  24   3  39   1   0         1        A1BG    Hs_A1BG_25586 ENSG00000121410   0   0  36   0   0   0   0  52
2    A1CF_87905   0   0   0   0   0     29974        A1CF    Hs_A1CF_87905 ENSG00000148584  25   0   0   0  19   0   0   0
3         A2M_1   0   0   0 187   0         2         A2M         Hs_A2M_1 ENSG00000175899   0  25   0   0   0   0   0   0
4     A2M_12371  31   0  82 603   1         2         A2M     Hs_A2M_12371 ENSG00000175899  16   1   0   0  22   0   0   0
5   A2ML1_18347   0   0   0   0   0    144568       A2ML1   Hs_A2ML1_18347 ENSG00000166535   0   0   0   0   0   0   0   0
6 A3GALT2_23042   0   0   0   0   0    127550     A3GALT2 Hs_A3GALT2_23042 ENSG00000184389   0   0   0   0   0   0   0   0
> dim(merged_data)#22339    18
[1] 22339    18
```

Uniqe ENSEMBL_GENE_ID

At first we have 22339 genes , after removing duplicate genes based on “ENSEMBL_GENE_ID” we have 19499 genes.

```
14  # ========== Uniqe ENSEMBL_GENE_ID =================
15  duplicates_ENSEMBL_GENE_ID <- duplicated(merged_data[10])#ENSEMBL_GENE_ID
16  # Subset to unique rows only
17  uniquegenes__ENSEMBL_GENE_ID <- merged_data[!duplicates_ENSEMBL_GENE_ID, ]
18  dim(uniquegenes__ENSEMBL_GENE_ID)
19  head(uniquegenes__ENSEMBL_GENE_ID)
20:1  # Uniqe ENSEMBL_GENE_ID
```

```
Console  Terminal  Background Jobs
R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/
> dim(uniquegenes__ENSEMBL_GENE_ID)
[1] 19499    18
> head(uniquegenes__ENSEMBL_GENE_ID)
         COUNTS NB5 NB4 NB1 NB2 NB3 ENTREZ_ID GENE_SYMBOL            Probe ENSEMBL_GENE_ID CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1    A1BG_25586  24   3  39   1   0         1        A1BG    Hs_A1BG_25586 ENSG00000121410   0   0  36   0   0   0   0  52
2    A1CF_87905   0   0   0   0   0     29974        A1CF    Hs_A1CF_87905 ENSG00000148584  25   0   0   0  19   0   0   0
3         A2M_1   0   0   0 187   0         2         A2M         Hs_A2M_1 ENSG00000175899   0  25   0   0   0   0   0   0
5   A2ML1_18347   0   0   0   0   0    144568       A2ML1   Hs_A2ML1_18347 ENSG00000166535   0   0   0   0   0   0   0   0
6 A3GALT2_23042   0   0   0   0   0    127550     A3GALT2 Hs_A3GALT2_23042 ENSG00000184389   0   0   0   0   0   0   0   0
7  A4GALT_88955 729 155  55 208   0     53947      A4GALT  Hs_A4GALT_88955 ENSG00000128274   0  39  15   0  58   0   0  42
>
```

Set rownames uniquegenes__ENSEMBL_GENE_ID As "ENSEMBL_GENE_ID

```r
# ========== set rownames uniquegenes__ENSEMBL_GENE_ID As "ENSEMBL_GENE_ID"# ==========
rownames(uniquegenes__ENSEMBL_GENE_ID) <- uniquegenes__ENSEMBL_GENE_ID[, 10]
head(uniquegenes__ENSEMBL_GENE_ID)
merge_subset <- uniquegenes__ENSEMBL_GENE_ID[, c(2:6, 11:18)]
# Print the subset of columns
head(merge_subset)
dim(merge_subset)
```

```
> head(uniquegenes__ENSEMBL_GENE_ID)
        COUNTS NB5 NB4 NB1 NB2 NB3 ENTREZ_ID GENE_SYMBOL            Probe ENSEMBL_GENE_ID CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1    A1BG_25586  24   3  39   1   0         1        A1BG    Hs_A1BG_25586 ENSG00000121410   0   0  36   0   0   0   0  52
2    A1CF_87905   0   0   0   0   0     29974        A1CF    Hs_A1CF_87905 ENSG00000148584  25   0   0   0  19   0   0   0
3         A2M_1   0   0   0 187   0         2         A2M         Hs_A2M_1 ENSG00000175899   0  25   0   0   0   0   0   0
5   A2ML1_18347   0   0   0   0   0    144568       A2ML1   Hs_A2ML1_18347 ENSG00000166535   0   0   0   0   0   0   0   0
6 A3GALT2_23042   0   0   0   0   0    127550     A3GALT2 Hs_A3GALT2_23042 ENSG00000184389   0   0   0   0   0   0   0   0
7  A4GALT_88955 729 155  55 208   0     53947      A4GALT  Hs_A4GALT_88955 ENSG00000128274   0  39  15   0  58   0   0  42
>
```

```r
rownames(uniquegenes__ENSEMBL_GENE_ID) <- uniquegenes__ENSEMBL_GENE_ID[, 10]
head(uniquegenes__ENSEMBL_GENE_ID)
# ========== select NB and CB colomns ====================
merge_subset <- uniquegenes__ENSEMBL_GENE_ID[, c(2:6, 11:18)]
head(merge_subset)
dim(merge_subset)
```

```
> # ========== select NB and CB colomns ====================
> merge_subset <- uniquegenes__ENSEMBL_GENE_ID[, c(2:6, 11:18)]
> head(merge_subset)
  NB5 NB4 NB1 NB2 NB3 CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1  24   3  39   1   0   0   0  36   0   0   0   0  52
2   0   0   0   0   0  25   0   0   0  19   0   0   0
3   0   0   0 187   0   0  25   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0   0   0   0
6   0   0   0   0   0   0   0   0   0   0   0   0   0
7 729 155  55 208   0   0  39  15   0  58   0   0  42
> dim(merge_subset)
[1] 19499    13
>
```

Remove NA and uniq duplicate items

```
28 #===========Replace any NA values with 0 ==================
29 merge_subset[is.na(merge_subset)] <- 0
30 head(merge_subset)
31 dim(merge_subset)
32 # ======= REmove duplicates================================
33 duplicates_m <- duplicated(merge_subset)
34 # Subset to unique rows only
35 uniquegenes__merge_subset <- merge_subset[!duplicates_m, ]
36 head(uniquegenes__merge_subset)
37 dim(uniquegenes__merge_subset) #15878    13
```

37:39 # REmove duplicates

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-Seq/

```
  NB5 NB4 NB1 NB2 NB3 CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1  24   3  39   1   0   0   0  36   0   0   0   0  52
2   0   0   0   0   0  25   0   0   0  19   0   0   0
3   0   0   0 187   0   0  25   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0   0   0   0
6   0   0   0   0   0   0   0   0   0   0   0   0   0
7 729 155  55 208   0   0  39  15   0  58   0   0  42
> dim(merge_subset)
[1] 19499    13
> head(uniquegenes__merge_subset)
  NB5 NB4 NB1 NB2 NB3 CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1  24   3  39   1   0   0   0  36   0   0   0   0  52
2   0   0   0   0   0  25   0   0   0  19   0   0   0
3   0   0   0 187   0   0  25   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0   0   0   0
7 729 155  55 208   0   0  39  15   0  58   0   0  42
8   0   0  11  28   0   0   0   0   0   0   0   0  25
> dim(uniquegenes__merge_subset) #15878    13
[1] 15878    13
>
```

calculate DGElist

```
38 # ======= Define the experimental design and create the DGEList object ======================
39 group <- factor(c(rep("normal", ncol(uniquegenes__merge_subset[,c(1:5)])), rep("covid", ncol(uniquegenes__merge_subset[,c(6:13)]))))
40 head(group)
41 prep_deg <- DGEList(counts = uniquegenes__merge_subset, group = group)
42 head(prep_deg)
43 dim(prep_deg)
```

37:44 # REmove duplicates

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

```
> head(prep_deg)
An object of class "DGEList"
$counts
  NB5 NB4 NB1 NB2 NB3 CB4 CB5 CB6 CB7 CB2 CB3 CB1 CB8
1  24   3  39   1   0   0   0  36   0   0   0   0  52
2   0   0   0   0   0  25   0   0   0  19   0   0   0
3   0   0   0 187   0   0  25   0   0   0   0   0   0
5   0   0   0   0   0   0   0   0   0   0   0   0   0
7 729 155  55 208   0   0  39  15   0  58   0   0  42
8   0   0  11  28   0   0   0   0   0   0   0   0  25

$samples
     group lib.size norm.factors
NB5 normal  4535335            1
NB4 normal   765453            1
NB1 normal  1521960            1
NB2 normal  5249365            1
NB3 normal   346127            1
8 more rows ...

> dim(prep_deg)
[1] 15878    13
>
```

Do Normalization , dispersion estimation, fit glm to data by qlmQLFit , calculate toptags genes and set thereshold for FDR and logFC. and create a vector of 0,1 for calculation pwf

```
y <- prep_deg[keep, ]
y <- calcNormFactors(y)
head(y)
dim(y)#[1] 9098   13
design <- model.matrix(~group)
# estimateDisp => estimates the dispersion parameters using a negative binomial model and an empirical Bayes
# Input: prep_deg_norm, design
y <- estimateDisp(y, design)
head(y)
# Perform hypothesis testing for differential gene expression
# glmQLFit => fits a GLM to the data and estimates the dispersion parameters using a quasi-likelihood approach
fit <- glmQLFit(y, design)
obj_topTags <- glmQLFTest(fit, coef = 2)
obj_topTags
# extract the top differentially expressed genes from a glmQLFTest object
top_genes <- topTags(obj_topTags, n = nrow(obj_topTags$table))
head(top_genes)
dge_selected_genes <- as.integer(top_genes$table$FDR<0.1 & abs(top_genes$table$logFC>1.5))
head(dge_selected_genes)#
length(dge_selected_genes) #10415
table(dge_selected_genes)
```

74:1 (Untitled)

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

```
Coefficient:  groupnormal
         logFC    logCPM        F       PValue          FDR
4539  -5.707072 16.435074 78.94146 6.500654e-19 5.914295e-15
14083 -8.065606 12.149374 56.98807 4.421582e-14 2.011378e-10
6203  12.223611  7.423858 49.62893 1.869468e-12 5.669472e-09
11627  8.362842  8.186717 45.88457 1.261198e-11 2.868594e-08
8033  -4.232252 15.793719 44.96936 2.011996e-11 3.661028e-08
15717 11.746845  6.953027 41.30711 1.306771e-10 1.981500e-07
>
```

```
# ========genes with FDR < 0.1 and absolute value of Log2FoldChange > 1.5 ==============================
dge_selected_genes <- as.integer(top_genes$table$FDR<0.1 & abs(top_genes$table$logFC>1.5))
names(dge_selected_genes) = rownames(top_genes$table)
head(dge_selected_genes)#
length(dge_selected_genes) #10415
table(dge_selected_genes)
```

81:1 genes with FDR < 0.1 and absolute value of Log2FoldChange > 1.5

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

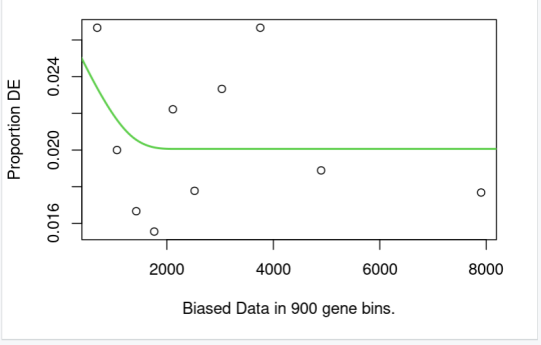
```
> head(dge_selected_genes)#
 4539 14083  6203 11627  8033 15717
    0     0     1     1     0     1
> length(dge_selected_genes) #10415
[1] 9098
> table(dge_selected_genes)
dge_selected_genes
   0    1
8908  190
>
```

Fitting the Probability Weighting Function (PWF)

We rst need to obtain a weighting for each gene, depending on its length, given by the PWF

```
head(dge_selected_genes)#
length(dge_selected_genes) #10415
table(dge_selected_genes)
# ====== Fitting the Probability Weighting Function (PWF) ==========
pwf = nullp(dge_selected_genes, 'hg19', 'ensGene')
head(null_sgenes)
```

```
> pwf = nullp(dge_selected_genes, 'hg19', 'ensGene')
Loading hg19 length data...
Warning message:
In library() :
  libraries '/usr/local/lib/R/site-library', '/usr/lib/R/site-library' contain no packages
> head(null_sgenes)
                DEgenes bias.data       pwf
ENSG00000169429       0    1179.0 0.02122129
ENSG00000124102       0     556.0 0.02411311
ENSG00000170323       1     842.0 0.02262012
ENSG00000038945       1    3581.0 0.02006331
ENSG00000188536       0     492.5 0.02446896
ENSG00000138207       1     934.0 0.02219187
>
```



Using the Wallenius approximation

we use the default method, to calculate the over and under expressed GO categories among DE genes.

```
#==========  Using the Wallenius approximation ===============
GO.wall <- goseq(pwf, "hg19", "ensGene")
# Sort GO terms by p-value
GO.wall <- GO.wall[order(GO.wall$over_represented_pvalue), ]
# Print top 10 most significant GO terms
top_GO_terms <- head(GO.wall, 10)
print(top_GO_terms)
unique(GO.wall$ontology)# "BP" "CC" "MF"
```

```
> print(top_GO_terms)
       category over_represented_pvalue under_represented_pvalue numDEInCat numInCat                                        term ontology
1378  GO:0003341           3.324061e-13                        1         18       97                            cilium movement       BP
2438  GO:0005929           7.142577e-13                        1         31      346                                    cilium       CC
2439  GO:0005930           7.815240e-13                        1         17       88                                  axoneme       CC
15207 GO:0097014           9.452874e-13                        1         17       89                             ciliary plasm       CC
10232 GO:0044782           1.616351e-11                        1         24      232                       cilium organization       BP
12721 GO:0060271           3.141912e-10                        1         22      222                           cilium assembly       BP
8418  GO:0035082           4.344472e-10                        1         12       55                          axoneme assembly       BP
7622  GO:0032838           4.356693e-10                        1         18      146 plasma membrane bounded cell projection cytoplasm       CC
3100  GO:0007018           4.415091e-10                        1         22      225                 microtubule-based movement       BP
430   GO:0001578           9.558442e-10                        1         13       72               microtubule bundle formation       BP
> unique(GO.wall$ontology)# "BP" "CC" "MF"
[1] "BP" "CC" "MF"
>
```

```
84  #===========  Using the Wallenius approximation ===============
85  GO.wall <- goseq(pwf, "hg19", "ensGene")
86  # Sort GO terms by p-value
87  GO.wall <- GO.wall[order(GO.wall$over_represented_pvalue), ]
88  # Print top 10 most significant GO terms
89  top_GO_terms <- head(GO.wall, 10)
90  print(top_GO_terms)
91  unique(GO.wall$ontology)# "BP" "CC" "MF"
84:41  Using the Wallenius approximation
```

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

```
> print(top_GO_terms)
       category over_represented_pvalue under_represented_pvalue numDEInCat numInCat                                        term ontology
1378  GO:0003341            3.324061e-13                       1         18       97                            cilium movement       BP
2438  GO:0005929            7.142577e-13                       1         31      346                                    cilium       CC
2439  GO:0005930            7.815240e-13                       1         17       88                                  axoneme       CC
15207 GO:0097014            9.452874e-13                       1         17       89                             ciliary plasm       CC
10232 GO:0044782            1.616351e-11                       1         24      232                         cilium organization       BP
12721 GO:0060271            3.141912e-10                       1         22      222                             cilium assembly       BP
8418  GO:0035082            4.344472e-10                       1         12       55                           axoneme assembly       BP
7622  GO:0032838            4.356693e-10                       1         18      146 plasma membrane bounded cell projection cytoplasm       CC
3100  GO:0007018            4.415091e-10                       1         22      225                      microtubule-based movement       BP
430   GO:0001578            9.558442e-10                       1         13       72                    microtubule bundle formation       BP
> unique(GO.wall$ontology)# "BP" "CC" "MF"
[1] "BP" "CC" "MF"
>
```


```
84  #===========  Using the Wallenius approximation ===============
85  GO.wall <- goseq(pwf, "hg19", "ensGene")
86  # Sort GO terms by p-value
87  GO.wall <- GO.wall[order(GO.wall$over_represented_pvalue), ]
88  # Print top 10 most significant GO terms
89  top_GO_terms <- head(GO.wall, 10)
90  print(top_GO_terms)
91  unique(GO.wall$ontology)# "BP" "CC" "MF"
84:41  Using the Wallenius approximation
```

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

```
> print(top_GO_terms)
       category over_represented_pvalue under_represented_pvalue numDEInCat numInCat                                        term ontology
1378  GO:0003341            3.324061e-13                       1         18       97                            cilium movement       BP
2438  GO:0005929            7.142577e-13                       1         31      346                                    cilium       CC
2439  GO:0005930            7.815240e-13                       1         17       88                                  axoneme       CC
15207 GO:0097014            9.452874e-13                       1         17       89                             ciliary plasm       CC
10232 GO:0044782            1.616351e-11                       1         24      232                         cilium organization       BP
12721 GO:0060271            3.141912e-10                       1         22      222                             cilium assembly       BP
8418  GO:0035082            4.344472e-10                       1         12       55                           axoneme assembly       BP
7622  GO:0032838            4.356693e-10                       1         18      146 plasma membrane bounded cell projection cytoplasm       CC
3100  GO:0007018            4.415091e-10                       1         22      225                      microtubule-based movement       BP
430   GO:0001578            9.558442e-10                       1         13       72                    microtubule bundle formation       BP
> unique(GO.wall$ontology)# "BP" "CC" "MF"
[1] "BP" "CC" "MF"
>
```


```
> unique(GO.wall$ontology)
[1] "BP" "CC" "MF"
```

Count the number of enriched terms for each ontology (Biological Process, Molecular Function, and Cellular Component)

```r
#===========  Using the Wallenius approximation ===============
GO.wall <- goseq(pwf, "hg19", "ensGene")
# Sort GO terms by p-value
GO.wall <- GO.wall[order(GO.wall$over_represented_pvalue), ]
# Print top 10 most significant GO terms
top_GO_terms <- head(GO.wall, 10)
print(top_GO_terms)
unique(GO.wall$ontology)# "BP" "CC" "MF"
# ===== count the number of enriched terms for each ontology (Biological Process, Molecular Function, and Cellular Component)
number_CC <- nrow(GO.wall[GO.wall$ontology == "CC", ])
number_BP <- nrow(GO.wall[GO.wall$ontology == "BP", ])
number_MF <- nrow(GO.wall[GO.wall$ontology == "MF", ])

cat("Number of enriched Biological Process terms:", number_BP, "\n")
cat("Number of enriched Molecular Function terms:", number_MF, "\n")
cat("Number of enriched Cellular Component terms:", number_CC, "\n")
prop_BP <- number_BP / nrow(GO.wall)
cat("Proportion of enriched Biological Process terms:", prop_BP, "\n")
```

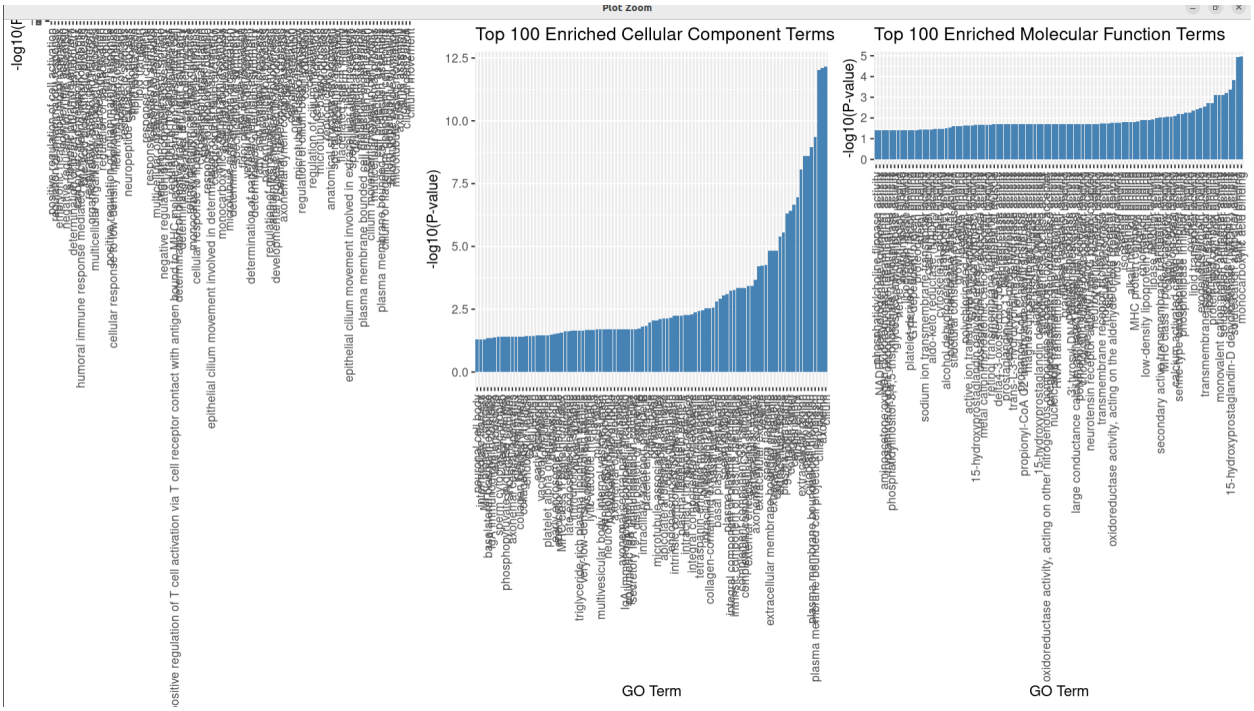
105:1 Using random sampling

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

```
> cat("Number of enriched Biological Process terms:", number_BP, "\n")
Number of enriched Biological Process terms: 13644
> cat("Number of enriched Molecular Function terms:", number_MF, "\n")
Number of enriched Molecular Function terms: 3972
> cat("Number of enriched Cellular Component terms:", number_CC, "\n")
Number of enriched Cellular Component terms: 1804
> prop_BP <- number_BP / nrow(GO.wall)
> cat("Proportion of enriched Biological Process terms:", prop_BP, "\n")
Proportion of enriched Biological Process terms: 0.7025747
>
```

```r
library(ggplot2)

# Subset the top 100 enriched terms for each ontology
top_BP <- head(GO.wall[GO.wall$ontology == "BP", ], 100)
top_CC <- head(GO.wall[GO.wall$ontology == "CC", ], 100)
top_MF <- head(GO.wall[GO.wall$ontology == "MF", ], 100)

# Create bar charts for each ontology
bp_plot <- ggplot(top_BP, aes(x = reorder(term, -over_represented_pvalue), y = -log10(over_represented_pvalue))) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "GO Term", y = "-log10(P-value)", title = "Top 100 Enriched Biological Process Terms") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))

cc_plot <- ggplot(top_CC, aes(x = reorder(term, -over_represented_pvalue), y = -log10(over_represented_pvalue))) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "GO Term", y = "-log10(P-value)", title = "Top 100 Enriched Cellular Component Terms") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))

mf_plot <- ggplot(top_MF, aes(x = reorder(term, -over_represented_pvalue), y = -log10(over_represented_pvalue))) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "GO Term", y = "-log10(P-value)", title = "Top 100 Enriched Molecular Function Terms") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))

bp_plot
# Combine the plots using the gridExtra package
library(gridExtra)
grid.arrange(bp_plot, cc_plot, mf_plot, ncol = 3)
```



Using random sampling

It may sometimes be desirable to use random sampling to generate the null distribution for category membership.

```
91 # ========== Using random sampling==================================================
92 GO.samp=goseq(pwf,"hg19","ensGene",method="Sampling",repcnt=1000)
93 head(GO.samp)
94
```

89:34 Using the Wallenius approximation

Console | Terminal | Background Jobs

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

```
> head(GO.samp)
      category over_represented_pvalue under_represented_pvalue numDEInCat numInCat                                                 term ontology
2   GO:0000003             0.000999001                        1         29      633                                       reproduction       BP
92  GO:0000226             0.000999001                        1         17      351                 microtubule cytoskeleton organization       BP
403 GO:0001539             0.000999001                        1         13       76               cilium or flagellum-dependent cell motility       BP
430 GO:0001578             0.000999001                        1         13       72                        microtubule bundle formation       BP
1378 GO:0003341            0.000999001                        1         18       97                                      cilium movement       BP
1384 GO:0003351            0.000999001                        1          7       30 epithelial cilium movement involved in extracellular fluid movement       BP
>
```

```
91  # ========== Using random sampling============================================
92  GO.samp=goseq(pwf,"hg19","ensGene",method="Sampling",repcnt=1000)
93  head(GO.samp)
94
95
96  plot(log10(GO.wall[,2]), log10(GO.samp[match(GO.wall[,1],GO.samp[,1]),2]),
97       xlab="log10(Wallenius p-values)",ylab="log10(Sampling p-values)",
98       xlim=c(-3,0))
99  abline(0,1,col=3,lty=2)
100
101
102
104:1   # Using random sampling                                   R Script
```

Console | Terminal | Background Jobs

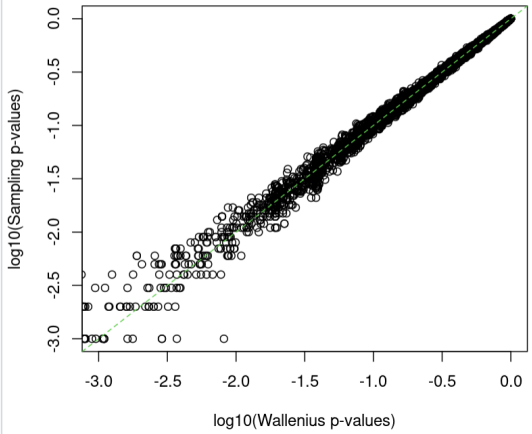
R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparatic

```
> plot(log10(GO.wall[,2]), log10(GO.samp[match(GO.wall[,1],GO.samp[,1]),2]),
+      xlab="log10(Wallenius p-values)",ylab="log10(Sampling p-values)",
+      xlim=c(-3,0))
> abline(0,1,col=3,lty=2)
>
```

R | Global Environment

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export | Publish



```r
#========= Number of enriched KEGG pathways ==================
library(topGO)
library(org.Hs.eg.db)
library(goseq)
# Convert ENSEMBL IDs to Entrez IDs
gene_ids <- names(dge_selected_genes)
gene_ids_entrez <- mapIds(org.Hs.eg.db, gene_ids, "ENTREZID", "ENSEMBL")
# Get KEGG pathways for Entrez IDs
en2eg <- as.list(org.Hs.egENSEMBL2EG)
eg2kegg <- as.list(org.Hs.egPATH)
grepKEGG <- function(id, mapkeys) { unique(unlist(mapkeys[id], use.names = FALSE)) }
kegg <- lapply(gene_ids_entrez, grepKEGG, eg2kegg)
# Perform KEGG pathway analysis using goseq
pwf <- nullp(gene_ids_entrez, "hg19", "ensGene")
KEGG <- goseq(pwf, gene2cat = kegg)
# Print the number of enriched pathways
cat("Number of enriched KEGG pathways:", nrow(KEGG), "\n")
# Print the top enriched pathways
head(KEGG)
```

113:1 # Number of enriched KEGG pathways

Console | **Terminal** | **Background Jobs**

R 4.2.2 · /media/mary/Data/PHD/Algorithm/__Projects__/3_AIB_proj2/proj2.RNA-SeqAnalysis/0_Data preparation/R__RNA_SEQ/

```
> cat("Number of enriched KEGG pathways:", nrow(KEGG), "\n")
Number of enriched KEGG pathways: 227
> # Print the top enriched pathways
> head(KEGG)
    category over_represented_pvalue under_represented_pvalue numDEInCat numInCat
200    05150            0.0003011350                0.9999758          5       36
136    04514            0.0006966218                0.9999104          6       65
140    04610            0.0017662371                0.9998470          4       31
219    05322            0.0022008996                0.9997964          4       33
155    04672            0.0051592501                0.9996221          3       21
7      00053            0.0054564954                0.9998529          2        7
>
```

```r
#E_GeneOntology.R

library(limma)
library(edgeR)
library(BiasedUrn)
library(geneLenDataBase)
library(goseq)
library(org.Hs.eg.db)
library(ggplot2)

########################## E ########################################################
# Read in the CSV files
merged_data <- read.csv("../GSE96/DATA/merged_all_counts__NB1_NB5_CB1_CB8.csv", header = TRUE, row.names = 1)
head(merged_data)
dim(merged_data)#22339    18
# ========== Uniqe ENSEMBL_GENE_ID =================
duplicates_ENSEMBL_GENE_ID <- duplicated(merged_data[10])#ENSEMBL_GENE_ID
# Subset to unique rows only
uniquegenes__ENSEMBL_GENE_ID <- merged_data[!duplicates_ENSEMBL_GENE_ID, ]
dim(uniquegenes__ENSEMBL_GENE_ID)
head(uniquegenes__ENSEMBL_GENE_ID)
# ========== set rownames uniquegenes__ENSEMBL_GENE_ID As "ENSEMBL_GENE_ID"# ==========
rownames(uniquegenes__ENSEMBL_GENE_ID) <- uniquegenes__ENSEMBL_GENE_ID[, 10]
head(uniquegenes__ENSEMBL_GENE_ID)
# ========== select NB and CB colomns ====================
merge_subset <- uniquegenes__ENSEMBL_GENE_ID[, c(2:6, 11:18)]
head(merge_subset)
```

```
dim(merge_subset)#19499    13

#===========Replace any NA values with 0 ==================
merge_subset[is.na(merge_subset)] <- 0
head(merge_subset)
dim(merge_subset)
# ======= REmove duplicates=================================
duplicates_m <- duplicated(merge_subset)
# Subset to unique rows only
uniquegenes__merge_subset <- merge_subset[!duplicates_m, ]
head(uniquegenes__merge_subset)
dim(uniquegenes__merge_subset) #15878    13
# ======= Define the experimental design and create the DGEList object ======================
group <- factor(c(rep("normal", ncol(uniquegenes__merge_subset[,c(1:5)])), rep("covid", ncol(uniquegenes__merge_subset[,c(6:13)]))))
head(group)
prep_deg <- DGEList(counts = uniquegenes__merge_subset, group = group)
head(prep_deg)
dim(prep_deg)
### cpm Filter normal ##############
# cpm/// Filter normal the genes that have low expression and/or low variability
# keep <- rowSums(cpm(y[, group == "normal"]) >= 1) >= 3
# head(keep)
# dim(keep)
# keep
# cpm/// Filter normal and covid
n_samples_normal <- ncol(merge_subset[,1:5])
n_samples_covid <- ncol(merge_subset[,6:13])
n_samples_per_group <- min(n_samples_normal, n_samples_covid)
keep <- rowSums(cpm(prep_deg) >= 1) >= n_samples_per_group
###################################
####### Perform data normalization and dispersion estimation
y <- prep_deg[keep, ]
y <- calcNormFactors(y)
head(y)
dim(y)#[1] 9098   13
design <- model.matrix(~group)
# estimateDisp => estimates the dispersion parameters using a negative binomial model and an empirical Bayes
# Input: prep_deg_norm, design
y <- estimateDisp(y, design)
head(y)
# Perform hypothesis testing for differential gene expression
# glmQLFit => fits a GLM to the data and estimates the dispersion parameters using a quasi-likelihood approach
fit <- glmQLFit(y, design)
obj_topTags <- glmQLFTest(fit, coef = 2)
obj_topTags
# extract the top differentially expressed genes from a glmQLFTest object
top_genes <- topTags(obj_topTags, n = nrow(obj_topTags$table))
head(top_genes)
# ========genes with FDR < 0.1 and absolute value of Log2FoldChange > 1.5 ==============================
dge_selected_genes <- as.integer(top_genes$table$FDR<0.1 & abs(top_genes$table$logFC>1.5))
names(dge_selected_genes) = rownames(top_genes$table)
head(dge_selected_genes)#
length(dge_selected_genes) #10415
table(dge_selected_genes)
# ====== Fitting the Probability Weighting Function (PWF) ===========
pwf = nullp(dge_selected_genes, 'hg19', 'ensGene')
head(null_sgenes)

#===========  Using the Wallenius approximation ===============
GO.wall <- goseq(pwf, "hg19", "ensGene")
# Sort GO terms by p-value
GO.wall <- GO.wall[order(GO.wall$over_represented_pvalue), ]
# Print top 10 most significant GO terms
top_GO_terms <- head(GO.wall, 10)
print(top_GO_terms)
unique(GO.wall$ontology)# "BP" "CC" "MF"
# ===== count the number of enriched terms for each ontology (Biological Process, Molecular Function, and Cellular Component)
number_CC <- nrow(GO.wall[GO.wall$ontology == "CC", ])
number_BP <- nrow(GO.wall[GO.wall$ontology == "BP", ])
number_MF <- nrow(GO.wall[GO.wall$ontology == "MF", ])

cat("Number of enriched Biological Process terms:", number_BP, "\n")
cat("Number of enriched Molecular Function terms:", number_MF, "\n")
cat("Number of enriched Cellular Component terms:", number_CC, "\n")
prop_BP <- number_BP / nrow(GO.wall)
cat("Proportion of enriched Biological Process terms:", prop_BP, "\n")
```

```
##############################################
#####plot the top enriched GO terms for each ontology
# (Biological Process, Molecular Function, and Cellular Component)
##############################################

library(ggplot2)

# Subset the top 100 enriched terms for each ontology
top_BP <- head(GO.wall[GO.wall$ontology == "BP", ], 100)
top_CC <- head(GO.wall[GO.wall$ontology == "CC", ], 100)
top_MF <- head(GO.wall[GO.wall$ontology == "MF", ], 100)

# Create bar charts for each ontology
bp_plot <- ggplot(top_BP, aes(x = reorder(term, -over_represented_pvalue), y = -log10(over_represented_pvalue))) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "GO Term", y = "-log10(P-value)", title = "Top 100 Enriched Biological Process Terms") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))

cc_plot <- ggplot(top_CC, aes(x = reorder(term, -over_represented_pvalue), y = -log10(over_represented_pvalue))) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "GO Term", y = "-log10(P-value)", title = "Top 100 Enriched Cellular Component Terms") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))

mf_plot <- ggplot(top_MF, aes(x = reorder(term, -over_represented_pvalue), y = -log10(over_represented_pvalue))) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(x = "GO Term", y = "-log10(P-value)", title = "Top 100 Enriched Molecular Function Terms") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))

bp_plot
# Combine the plots using the gridExtra package
library(gridExtra)
grid.arrange(bp_plot, cc_plot, mf_plot, ncol = 3)
```