Senescence Gene Expression Pattern in Melanoma: A Comparison Between Cell Model and Clinical Data

Abbreviations

CDK	Cyclin Dependent Kinase
DDR	DNA Damage Response
DNA	Deoxyribonucleic Acid
MC1R	Melanocortin 1 Receptor
NGS	Next Generation Sequencing
RNA	Ribonucleic Acid
SASP	Senescence Associated Secretory Phenotype
SAPA	Senescence Associated Proliferation Arrest
TCGA	The Cancer Genome Atlas Program
UV	Ultraviolet
WHO	World Health Organization
QC	Quality Control

ABSTRACT

Melanoma, a formidable challenge in the realm of oncology, has continually perplexed researchers due to its complex molecular roots. In this research, the focus is on understanding how senescence-related genes behave in melanoma, specifically comparing findings from cell models and clinical data. To discern gene expression differences, the study employed differential expression analysis, a method for identifying genes that are differentially expressed in a given condition. To enhance the interpretation of the results, visualization tools were utilized. These tools allow researchers to graphically represent complex data, facilitating a clearer understanding of the patterns and relationships within the gene expression data. A noteworthy aspect of the study involves exploring intersections between senescent genes identified in clinical samples and those found in the cell model data. This comparison aims to reveal commonalities or differences in the presence of senescent genes between the cell model data with a specific mutation and the clinical data. By conducting this comprehensive analysis, the research aims to contribute valuable insights into the senescence gene expression patterns in melanoma.

Table of Contents

1. Introduction

1.1 Understanding Cancer

Cancer is a complex group of diseases in which some cells in the body grow uncontrollably and abnormally due to genetic mutations or alterations and spread to other parts of the body. Cancer affects millions of people worldwide. According to the report by the American Cancer Society, cancer is the cause for 1 in every 6 deaths worldwide, which is more than AIDS (acquired immunodeficiency syndrome), tuberculosis and malaria combined. Cancer has become the second leading cause of death after cardiovascular diseases globally in countries with high Human Development Index (HDI) (*Global cancer facts and figures, 4th edition. Atlanta: American Cancer Society, Inc. 2022*).

Human body is made up of trillions of cells and cancer can potentially originate in any part of the body. In normal circumstances, human cells grow and multiply according to the requirements of the body through a process called cell division. When the cells become old or damaged, they die, and new cells replaces the old cells. Occasionally, this meticulously regulated process breaks down, leading to the proliferation of abnormal cells beyond their boundaries and forming clusters of tissue known as tumors. Based on the potential to spread, tumors are classified into two, Benign tumors and Malignant tumors. Benign tumors are non-carcinogenic which typically grow slowly and are localized. The cells also tend to have well defined borders which prevents it from spreading to neighboring cells. On the other hand, Malignant tumors are characterized by the uncontrolled cell growth, invasion into nearby tissues and potential to spread to other parts of the body (Metastasis). Malignant tumors can be life threatening and can interfere with functions of vital organ in the body as they grow and spread. Most primary tumors detected in humans are benign tumors, which are harmless in most cases except when the growth press on vital organs or tissues. Certain benign tumors produce high levels of hormones which causes physiological imbalance in body. For instance, hyperthyroidism is detected in patients with Thyroid adenomas, caused by excessive release of thyroid hormone into the circulation. Nonetheless, the death caused by benign tumors are relatively few compared to malignant tumors, which is responsible for 90% of death in cancer (*Robert A. Weinberg, The biology of cancer, 2ed edition, 2013*).

Cancer cells exhibit a range of distinctions from normal cells. For example, Normal cells grow only when they receive the signals to grow but cancerous cells grow in the absence of signals as well. Cancer cells disregard the usual signals that prompt cells to cease division or undergo programmed cell death known as apoptosis. Normal cells halt their growth when they encounter neighboring cells, but cancer cells penetrate neighboring regions and spread to distant parts of body. Compared to normal cells, cancer cells relay on different kinds of nutrients, evade detection by immune system and accumulate a multitude of modifications in their chromosomes. Cancer usually occurs due to changes in genes that control cell division and growth. Factors that contribute to the risk of cancer include exposure to chemicals or various substances, alongside specific behavioral patterns. The most studied or suspected risk factors for cancer include, age, alcohol, cancer-causing substances, chronic inflammation, diet, hormones, immunosuppression, infectious agents, obesity, radiation, sunlight, tobacco (*Risk factors for cancer, National Cancer Institute, December 23, 2015, cancer.gov*).

There are predominantly three categories of genes affected in genetic changes that contribute to cancer: proto-oncogenes, tumor suppressor genes, and DNA repair genes. Proto-oncogenes typically regulate cell growth, division, and differentiation. When mutated, these genes can become oncogenes and can cause uncontrolled cell growth. Tumor suppressor genes, as the name indicate involved in controlling cell growth and division. Alterations in these genes will cause uncontrolled cell growth. The damaged DNA is taken care by the DNA repair genes, mutations of these genes will cause additional mutations in other genes and duplications and deletions of chromosomal parts. These mutations can make the cell cancerous. There are changes in tissue which is not cancer, but they should be carefully monitored since, if left untreated, those tissues has the potential to become cancer tissue in future. The examples of such kinds of tissues include Hyperplasia, Dysplasia and Carcinoma in situ. Hyperplasia occurs when the extra cells build up when the cells within a tissue multiply faster than the normal cells. Hyperplasic cells look normal under microscope therefore not considered as cancer cells. Dysplasia is an advanced condition of Hyperplasia since, the cell growth is like hyperplasia, but the cells look abnormal in microscopic conditions. These types of cells should be carefully monitored and treated if needed. Carcinoma in situ is much more advanced in terms of cell growth and it is even considered as the stage zero of cancer. It is not considered as cancer because in carcinoma in situ even though the cells look abnormal, they still do not invade the nearby tissue the way that cancer cells do (*National Cancer Institute, Cancer.gov*).

1.2 Melanoma Overview

Melanoma is defined as a type of skin cancer which originate from the cells called melanocytes. It is also known as the deadliest form of skin cancer. New reports shows that the malignant melanoma is on a rise in Europe especially in the north and northwestern countries like United Kingdom (UK) and Ireland. Compared to other malignant tumors, melanoma ranked as the sixth most common tumor in men and women in Europe (*Paolo A. Ascierto, et al. 2021*).

Melanocytes are found in the basal layer of epidermis, and it produces the ultraviolet (UV) absorbing pigment called melanin. Melanin is synthesized by melanocytes when α-melanocyte stimulating hormone (α-MSH), produced by keratinocytes binds to melanocortin 1 receptor (MC1R) on melanocytes. The produced melanin is then transported to keratinocytes which then uses the melanin as a shield for the nuclei from the mutagenic effects of the ultraviolet (UV) radiation. The keratinocytes then eventually mature and die, but the outer layer of skin is protected by both melanin in keratinocytes and the dead layer of keratinocytes, which act as a protective barrier to the cells beneath. The number of melanocytes is same in all skin types and it is not the number of melanocytes but the ratio of two different types of melanin in our body determines the skin color. Melanocytes produces two types of melanin; one is a black/brown pigment and second is a red/yellow pigment. The former is called eumelanin, and the latter is pheomelanin. People with darker skin has more eumelanin in their skin and are better UV protected (*Davis LE, et al. 2019*).

The risk factors for melanoma includes, sun exposure, indoor tanning, immunosuppression, moles, family history and obesity. The UV exposure is considered as the main risk for melanoma. The UV light induce the production of thymidine-dimers which is a DNA photoproduct, and this will also induce mutations in signaling pathways and finally carcinogenesis. The commonly mutated proteins include, NRAS and BRAF which are the members in mitogen-activated protein kinase (MAP-K). in young patients with moles and sun exposure, BRAF mutation is more common. Coming to the second risk factor, even though the international agency for research on cancer has identified tanning bed as a carcinogen due to the higher levels of UVA (wavelength; 320-400) and UVB (wavelength; 280-320) radiation, an estimated 7.8 million women and 1.9 million men uses tanning beds annually. Immunosuppressed patients also shown to have an increased risk of melanoma. Moles are considered as a benign growth of melanocytes. Family history also plays as a potential risk

factor for developing melanoma since, 10% of the patients with melanoma had a family history associated to the disease. Finally, obesity can also induce melanoma. Studies have shown that people with a Body Mass Index (BMI) > 30 has more risk to develop melanoma since, excess body fat will induce BRAF V600E oncogene activity (*Saginala K, et al. 2021*).

Melanoma is classified into 4 different groups including (i) superficial spreading, (ii) nodular, (iii) lentigo maligna and (iv) acral lentiginous melanomas. Among these four categories superficial spreading melanoma is the most common and it accounts for about 70% of total melanomas. Nodular form only represents 15 to 30 % of all melanomas. The last two types represent less than 10% of cases. The melanoma is also staged using four different methods, (i) the Clark scale, (ii) the Breslow scale, (iii) TNM staging and (iv) Number stages. Each system ususes different properties of melanoma to categories into different stages. For instance, the depth of lesion affecting various skin layers is measured in the Clark scale and in Breslow scale the thickness of melanoma in the skin is measured (*Liu Y, et al. 2014*).

Diagnosing melanoma involves conducting biopsy of the lesion by a health care provider, regardless of its detection method. It is usually done by a well-trained pathologist after careful examination of variety of histopathological features. There are also several histological mimics of melanoma, and this makes the diagnosis more challenging. Therefore, researchers also use molecular biomarkers to help in recognition of melanoma and immunohistochemistry (IHC) to interpret in tough cases. The most used biomarkers for the diagnosis of melanoma are melanocytic markers and proliferative markers. Immunohistochemistry is also used to determine the stage of melanoma. The treatment for melanoma includes, surgical resection, chemotherapy, and targeted therapies (*Davis LE, et al. 2019*).

1.3 Somatic and familial melanoma

Somatic melanoma is defined as the melanoma that arise from the somatic mutation of the skin cells. These kinds of mutations are not inherited and will not pass down to offspring. On the other hand, familial melanoma refers to a rare form of melanoma that occurs due to genetic predisposition.

In melanoma, the somatic mutations are more common than germline mutations. The v-Raf murine sarcoma viral oncogene homolog B proto-oncogene (BRAF) mutations accounts for

33-65% of cutaneous malignant melanomas. BRAF V600E, is a mutation that involves substitution of valine by glutamic acid at amino acid position 600 is the most recurrent mutation. This specific mutation is involved in melanocyte nevi. The less common BRAF mutations includes, BRAF V600K, BRAF V600D, and BRAF V600R. Some observations made on the relationship between intermittent sun exposure and BRAF mutation have found out that melanoma arising from skin which was unexposed to sun rarely showed BRAF mutation whereas the other group showed BRAF mutation in all cases. Neuroblastoma RAS viral oncogene homolog (NRAS) is a gene that encodes a protein in RAS family. Mutation of this gene accounts for the second most frequent somatic mutation in melanoma. NRAS mutation activates the RAS protein and there by activation of Mitogen-Activated Protein Kinase (MAPK) pathway and Phosphatidyl inositol 3 – kinase (PI3K) pathways. Activation of these 2 pathways will lead to cell proliferation. Next gene is a tumour suppressor gene called Neurofibromatosis type 1 (NF1), that involved in the down regulation of RAS proteins. NF1 mutations were found in 12-30% of melanoma. The Rac family small GTPase 1 gene (RAC1) is a gene that involves in the MAPK downstream signaling and studies have identified mutation of this gene also contribute to melanoma. The phosphate and tensin homolog gene (PTEN) is also a tumour suppressor gene and PTEN abnormalities have been reported in 28 to 43% of melanomas. The loss of PTEN will result in the activation of PI3K signaling pathway. The lower frequency mutations in melanoma includes the mutation in V-Akt murine thymoma viral oncogene homologs 1 and 3 (AKT1 and AKT3) and mutations in MAP2K1 and MAP2K2 (*Motwani J, et al. 2021*).

In case of familial melanomas, mutation in Cyclin-dependent kinase 2a (CDKN2A) gene has the highest risk in developing melanoma. This gene encodes 2 proteins namely P16 INK4A and P14 ARF. These proteins are associated with the regulation of apoptosis and cell cycle. The germline mutation of CDKN2A will result in melanoma by various process involves in cell cycle promotion and cell proliferation. Another gene called telomerase reverse transcriptase (TERT) also found to have somatic mutation in high rate in melanoma and it usually has the mutation on promoter region. The Genome-Wide Association Studies (GWAS) has identified 15 low risk melanoma susceptibility genes. That include, melanocortin type 1 receptor (MC1R), solute carrier family 45 member 2 (SLC45A2), oculocutaneous albinism II (OCA2) and many more (*Motwani J, et al. 2021*).

1.4 Cellular senescence and melanoma

“Cellular senescence is stress-inducible state of terminal proliferative arrest accompanied by a hypersecretory phenotype referred to us senescence-associated secretory phenotype (SASP) " (*Schmitt CA. et al. 2022*).

Senescent cells assist adult tissue in its process of tissue regeneration and remodelling. It is vividly evident from an experiment carried out on a mouse model designed to pinpoint senescent cells as eliminating these cells during the wound healing process culminated in retarding tissue repair. On the contrary, there found to have an accelerated restoration of cellular tissues in another mouse model that has specifically induced senescence from oncogenes and local transportation of keratinocytes briefly exposed to Senescence Associated Secretory Phenotype (SASP) factors. Apart from that, the development of fibrosis in the process of skin wound healing has been found to have restriction in senescent human cells grown in culture. The antifibriotic effects in organs like kidneys, liver and lungs possessed by senescent cells are also evident through mouse models. Senescent cells are also hypermetabolic which means cells exhibit increased oxygen consumption and utilize glycolysis, fatty acid oxidation and oxidative phosphorylation in an abnormal manner to optimize energy production (*Schmitt CA, et al. 2022)*.

Cessation of proliferation coupled with an unpaired DNA Damage Response (DDR) is regarded as one among the primary characteristics defining senescent cells. The principal agents driving senescence-associated proliferation arrest (SAPA) are cyclin dependent kinase (CDK) inhibitors 1and 2A, typically referred to as p21 and p16INK4a which block the formation of CDK-cyclin complexes crucial for regulating cell cycle checkpoints during the G1-S phase transition. Also,senescent cells are commonly identified in premalignant lesions found in individuals with cancer which is a fact that makes cellular senescence a natural barrier to tumorigenesis.*(Schmitt CA, et al2022*). Moreover, a paradoxical effect on cancer treatment is indicated by cellular senescence but it is often overlooked as a plausible cancer treatment though cell senescence can be an antidote to proliferation of cancer cells (*Zeng S, et al. 2018*).

Additionally, distinctive features of senescent cells have been recognized within premalignant lesions, particularly in human naevi and these findings led to assess senescence in both neoplastic and malignant human tissues but due to the absence of a singular marker that reliably and definitively identifies this distinct state, accurately labelling senescent cells in

preclinical studies has posted significant challenges.There was only a consistent profile of numerous senescence indicators, among which the most prominent ones were, damaged DNA, activated DDR, halted cell cycle, expanded lysosomal compartment and histone modifications (*Schmitt CA, elat. 2022*). Hence, studies asserting the detection of senescence in tissue samples solely based on a single marker should be approached and interpreted cautiously. Complexity of senescence therefore requires a comprehensive evaluation across multiple markers for a more accurate understanding.

The exhibition of properties that promote tumour growth by senescent cells have been discovered through certain studies. In evidence, various components of the SASP are linked to pro-tumorigenic processes, such as chronic inflammation, mitogenic signalling, stemness, angiogenesis and invasion,genotoxicity and immune suppression.Also,studies conducted on patient-derived samples have provided correlational evidence supporting the pro-tumorigenic aspects of senescence. For example, research involving cultured human melanoma cells revealed that the SASP displayed by these cells exhibited pro-tumorigenic and metastasis in a xenograft mouse model (*Schmitt CA, et al. 2022*).

Furthermore, senescent cells can also halt their cell cycle indefinitely and can be eliminated by immune cells like macrophages, neutrophils or natural killer cells. However, they have the capacity to persist in a senescent state for years. The accumulation of senescent cells elevates the risk of potentially re-entering the cell cycle and promoting oncogenesis. Despite this, senescence also acts as a natural barrier against uncontrolled cell growth and the formation of malignancies. These contradictory characteristics of senescence make a considerable challenge in our attempts to target senescence-related pathways for cancer treatment (*Zeng S, et al. 2018*).

1.5 Significance of BRAF V600E mutation in relation to senescence

Human moles or naevi are benign tumours of melanocytes, the cells responsible for skin pigmentation and many of these moles accommodate mutations, notably the V600E (valine is substituted for glutamic acid and 600 represent the position of the amino acid on the protein) mutation in the BRAF gene, which is a protein kinase and downstream effector of Ras(family of proteins involved in cell signalling pathways). However, naevi usually stay in a growth-

arrested state for many years and only rarely progress into malignancy. This prompts the question of whether moles experience BRAFV600E triggered senescence because continued expression of BRAFV600E in human melanocytes induce cell cycle arrest. Accompanied by activation of both p16INK4a and senescence-associated acidic beta galactosidase (SA- β-gal) activity, a widely recognized marker for senescence. Confirming these findings in vivo congenital naevi consistently show positivity for SA- β-gal, indicating the existence of this classical marker associated with senescence in a largely growth arrested, neoplastic human lesion. Studies have also proven that factors other than p16INK4a contribute to the protection against BRAFV600E based proliferation. In laboratory settings as well as in living organisms, melanocytes expressing BRAFV600E mutation exhibit typical signs of senescence. This indicates that the oncogene induced senescence represents a legitimate protective physiological process. Melanocytic naevi presents an intriguing human setting wherein an active oncogene can exist alongside with long-term arrested cells. (Michaloglou C, et al.2005).

1.6 Aim of the study

The aim of the work presented in this master thesis was to find out the senescent gene expression pattern in melanoma by employing differential expression analysis. This study involves a comparative analysis between cell model and clinical datasets. Specifically, it aims to identify and characterize the common genes between significantly regulated genes obtained from differential expression analysis and the known set of senescent genes, elucidating potential correlations and distinctions in senescence-associated mechanisms between these contexts.

2. MATERIALS AND METHODS

2.1. RNA Sequencing analysis workflow of cell model data

The data received for cell model is in the form of fastq file format and it is paired end sequencing data. Data is named P6, P7 and P8. The experimental group comprises a cell model subjected to treatment involving BRAF mutation, while the control group underwent treatment involving melanoma mutation except BRAF.

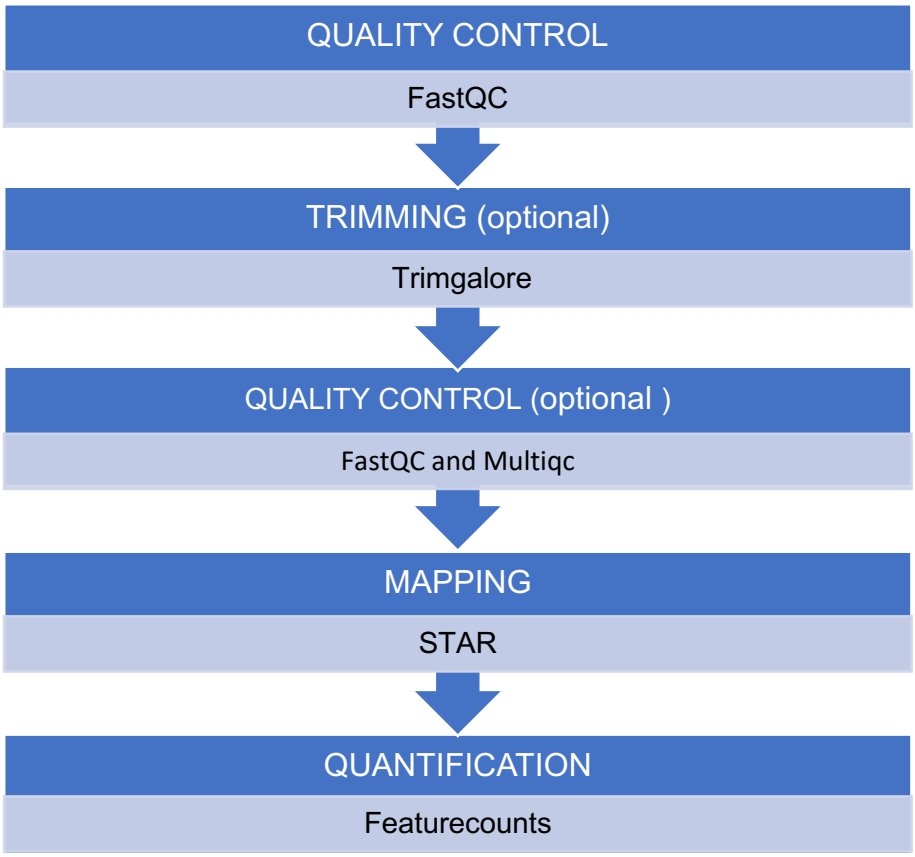

Figure 1: This diagram illustrates the sequences of steps in the pipeline designed for data preparation before conducting differential expression analysis.

- Computer setup

  Performing the RNA seq analysis can be computationally intensive, especially for larger datasets. For this project an external powerful server is used for fast computing.

  In general : 16 Gb of RAM minimum (better to have 96+ Gb), 500 Gb of disk space (better to have 10+ Tb), Fast CPU (at least 8 cores or more), External storage.

The sequencing data was supplied in a fastq file format. In the first step the quality of every data is assessed. For this purpose, Fastqc and Multiqc were used.

Conda, a package manager is installed since, it simplifies the installation and management of software packages, libraries and dependencies. The following command is used to install conda.

```
curl -OL https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
#check version
conda -V
# version 23.7.2
```

2.1.1. Quality control

The raw reads are subjected to quality control to check mainly the Phred quality score, Adapter content, Duplication rate, Guanine and Cytosine (GC) content. The quality control (QC) is important because Next Generation Sequencing (NGS) technologies produce vast amount of data and it is necessary to check for the quality since the issues can arise during library preparation and sequencing.

For quality control, FastQC is used. It analyses various aspect of the data and generate report that provide insights into different quality metrics.

```
#install fastqc
conda install -c bioconda fastqc
# version v0.12.1
```

To perform FastQC, run the following command,

```
fastqc "file name" -o "path to output directory"
# to perform fastqc for multiple files at same time
fastqc "specify the directory"/*
```

In the context of paired-end sequencing data, FastQC, is designed to generate distinct analytical reports for each set of paired reads, commonly denoted as read1(R1) and read2 (R2). This enables a comprehensive evaluation of the quality metrics to individual read sets within a paired-end dataset. These individual reports will contain quality metrics, charts and analysis specific to each set of reads, allowing you to assess the quality of both datasets separately. This will help in understanding the quality of each read in paired-end dataset.

Upon analysing the FastQC reports for each sample it is found out that one sample contain adapter contamination. The read 1 is shown in figure 2 and read 2 in figure 3.

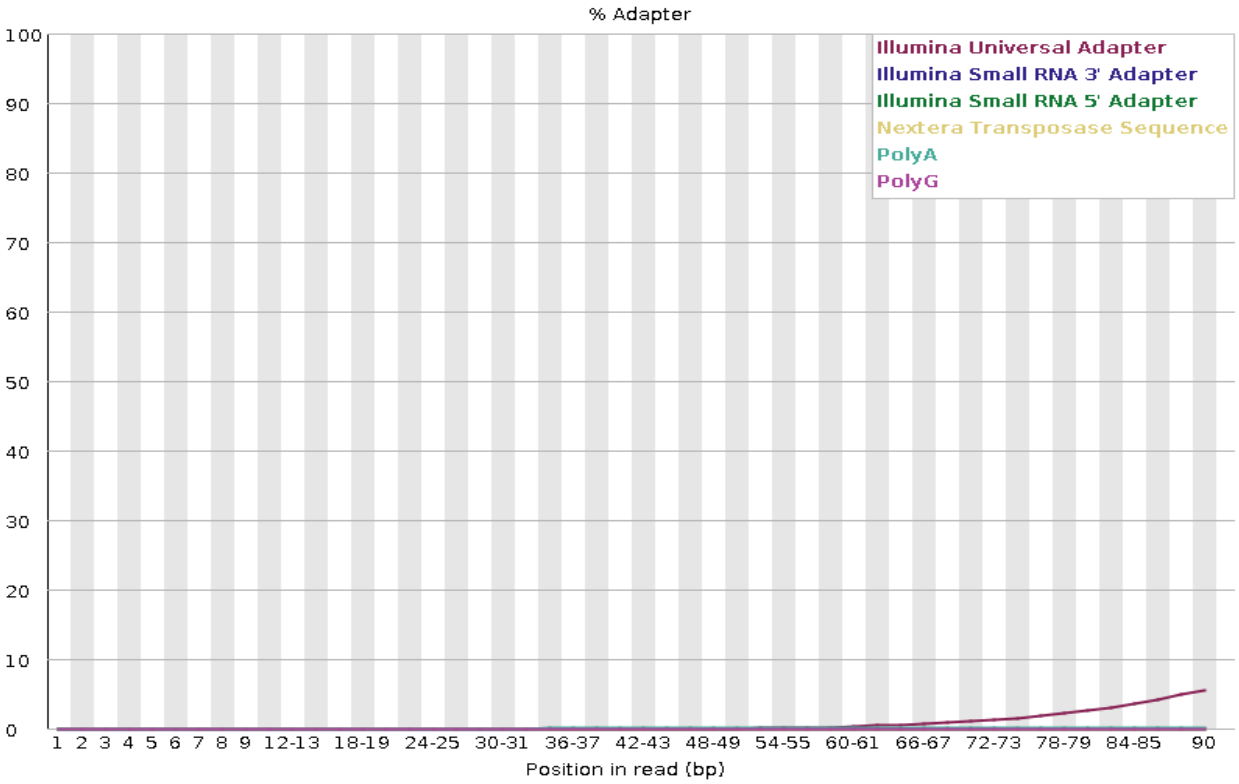

Figure 2 : adapter contamination in FastQC report of read1, P7 control sample

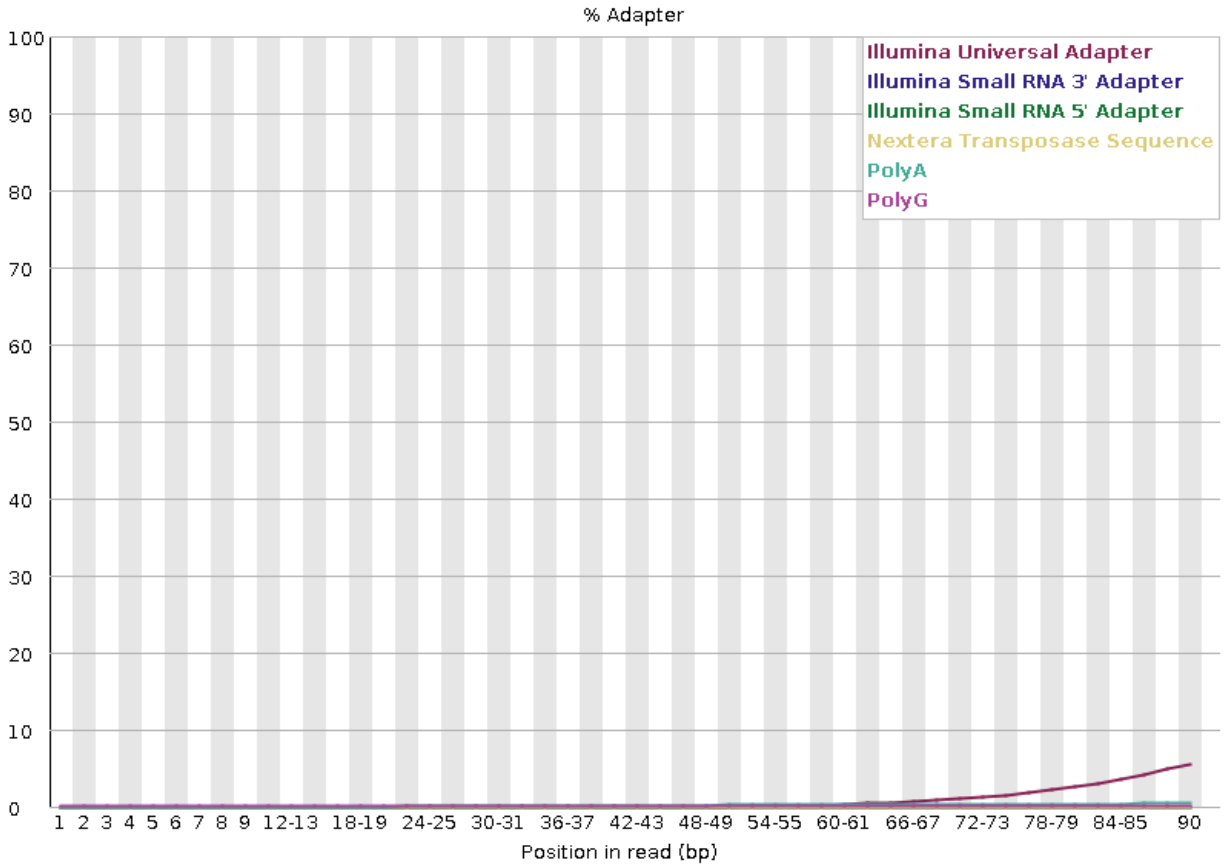

Figure 3 : adapter contamination in p7 control sample read 2

FastQC generates reports for each input file separately and that's where MultiQC is used to generate single report from multiple samples. Compiling data within a single report offers a rapid method to swiftly review statistics with ease (Philip Ewels, et, al. 2016).

```
#install multiqc

pip install multiqc

#version 1.14
```

MultiQc is performed by running the MultiQC command and below displayed adapter content (figure 4) and mean quality score (figure 5) results from the MultiQC report.

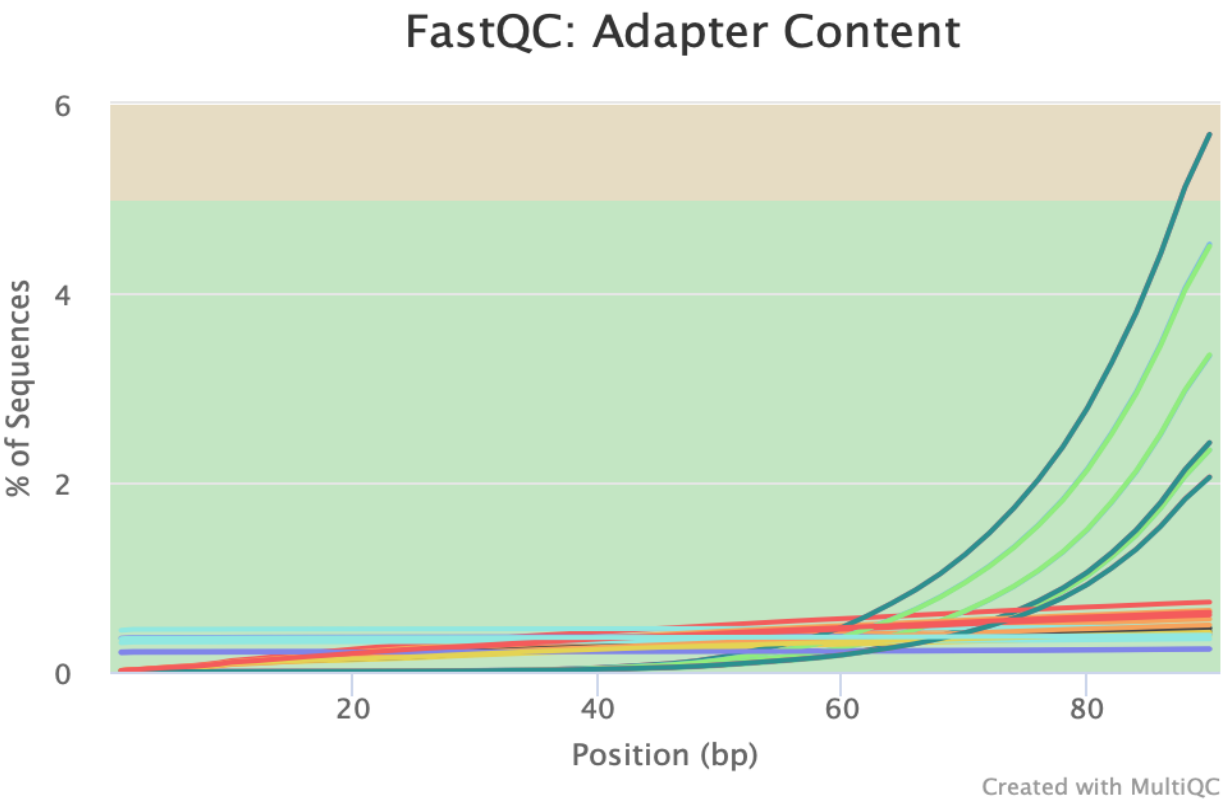

Figure 4: Adapter content in Multiqc report of the cell model data

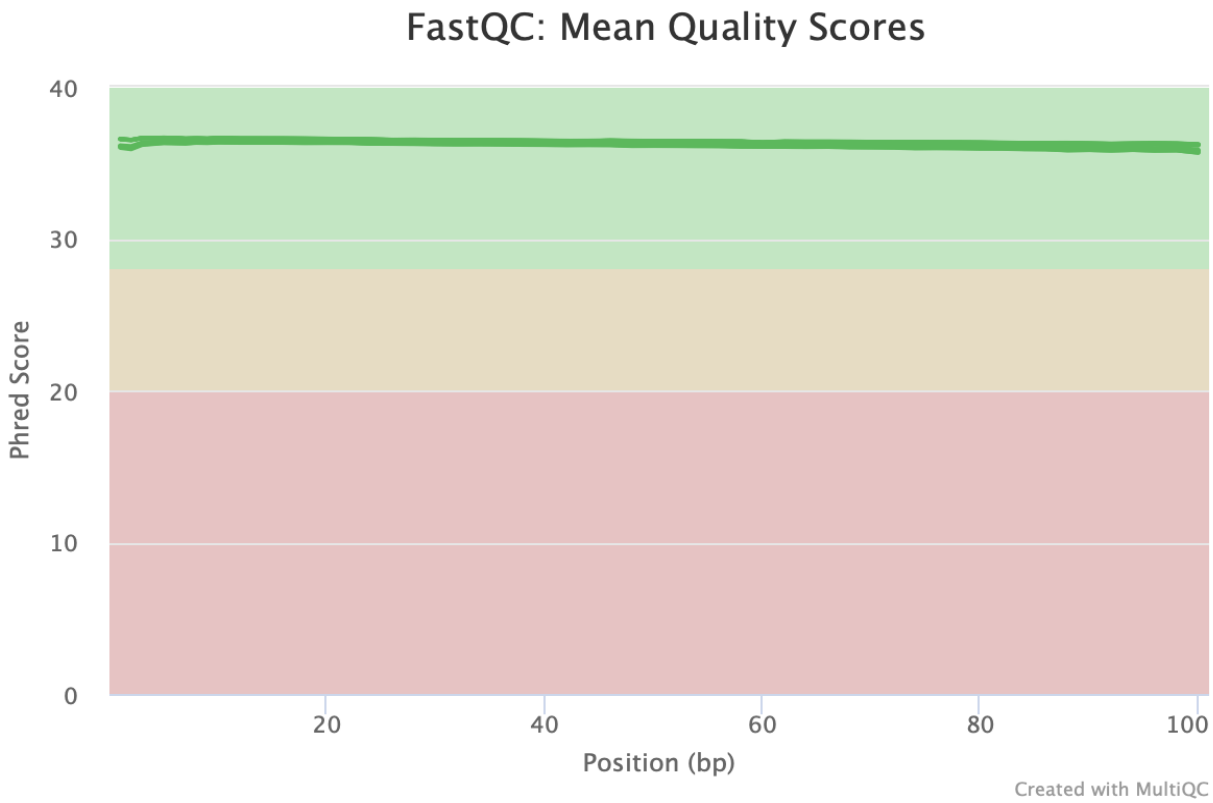

Figure 5 : Mean quality scores in multiqc report of the cell model data

From the FastQC and MultiQC reports it is evident that only one sequencing read contains the adapter contamination and it is minimal since, the adapter contamination is less than 5% in the base pairs 80 to 90. For these reasons trimming is omitted from the project. The figure 5 shows a good mean quality score throughout all sequencing reads, which is good for downstream analysis.

2.1.2 MAPPING

Following the quality control to ensure the data integrity, Spliced Transcripts Alignment to a Reference (STAR) is used to align RNA sequencing reads to reference genome. STAR is a splice-aware aligner and is faster than other aligners while maintaining high accuracy.

Run the following command to install STAR

```
conda install -C bioconda star
```

To perform mapping using STAR, it requires genome index and reads in FASTQ format. Including a gene annotation file is also recommended for better alignment since it provides information about gene locations, exons and introns. Hence, in this project gene annotation file have used in creating gene index.

Reference genome (version 36) is downloaded from the website genecode.com. The reference genome was a fasta file named genome sequence (GRCh38.p13). Then, GTF file named basic gene annotation (version 36) has also downloaded from the website genecode. Both of these files are then saved to a file named reference.

The following code is used to create star index,

```
 STAR –runThread 28 –runMode genomeGenerate –genomeDir starindex
–genomeFastaFiles
/home/stud01/NAS/star/reference/GRCh38.p13.genome.fa.gz       –
sjdbGTFfile/home/stud01/NAS/star/reference/genecode.v36.chr_pa
tch_hapl_scaff.basic.annotation.gtf.gz
```

Description of parameters:

--runThread	number of threads
--runMode	generate genome files and is set to genomeGenerate
--genomeDir	path to output directory
--genomeFastaFiles	path to genome fasta file
--sjdbGTFfile	path to annotation file

Now that the STAR index has been generated, the next step involves initiating STAR mapping using the following command.

```
STAR --runThreadN 28 —-genomeDir /GenomeDir/ --readFilesIn
data_recieved/Lonza_P6_BRAF_siCtrl_278_S95417_r1.fastq
data_recieved/Lonza_P6_BRAF_siCtrl_278_S95417_r2.fastq
--outFileNamePrefix /NAS/STAR_out/p6
```

Description of parameters:

--runThreadN	number of threads
--genomeDir	path to genome index directory
--readFilesIn	path to the reads
--outFileNamePrefix	output prefix name with path

The alignment result generated by STAR is stored in SAM (sequence alignment/map) format. There is also some other output files having the information about the alignment, splice junctions, log files on the run progress, read pairs, and mapping statistics.

2.1.3 QUANTIFICATION

Quantification is an important stage in RNA sequence analysis to measure the quantity of reads align to each gene or genomic region. FeatureCounts is a bioinformatic tool which is widely used for quantification and in this project the quantification is done with the aid of FeatureCounts.

```
#install featurecounts
conda install subread
```

To perform feature counts, the following line of code is used,

```
featureCounts -T 28 -t exon -a GenomeDir/genecode.v36.chr_patch
_haplscaff.basic.annotation.gtf -o p8BRAF.txt -p STAR_out/p8/
BRAF/p8BRAFAligned.out.sam
```

Description of Parameters:

-T	Nummer of threads
-t exon	Quantification at the level of exons
-a	Specifying annotation file
-o	Specifies output file name
-p	Specifies the input files are paired end reads.

2.1.4 Differential Expression Analysis of Cell model data

After completing the quantification step, the processed data is ready for subsequent differential gene expression analysis. R version 4.3.2 was employed for performing the differential expression analysis. It is used to identify the differences in gene expression across a cohort of samples. Frequently it is used to find out the differences between multiple biological conditions. There are multiple tools available for this analysis but in this project a popular Bioconductor package called DEseq2 has been used.

To get started first the package was installed and the library was then loaded,

```
BiocManager::install("DESeq2", force = TRUE)
library(DESeq2)
```

Several libraries have been pre-installed for future utilization. These libraries include, dplyr for data manipulation, pheatmap and ggplot2 for visualization, tidyverse and tibble for comprehensive data handling.

```
 suppressWarnings(library(dplyr))
 suppressWarnings(library(pheatmap))
 suppressWarnings(library(ggplot2))
 suppressWarnings(library(tidyverse))
 suppressWarnings(library(tibble))
```

The following lines of codes were used to initialize a data frame and subsequently import/read the associated data.

```
# Create an empty data frame to store the extracted data
 new_dataset <- data.frame()
# Read the dataset
 df <- read.table("/Users/methungeorge/Desktop/thesis/datasets/
p6BRAF.txt", header = TRUE, sep = "\t")
```

```
#extract the gene IDs
 new_dataset <-  df$Geneid
```

the extracted geneIDs is now saved in new_dataset.

The subsequent code sequence is designed to produce all necessary files essential for the analysis and concurrently extract the file names.

```
# Specify the directory containing your files
 directory <- "datasets/"


# List files with names containing "6," "7," or "8" (the files
are named P6,P7 and P8 )
 file_list   <-   list.files(directory,   pattern   =   "[678]",
full.names = TRUE)
# Initialize a vector to store the extracted strings
 file_names_extracted <- character(0)


# Initialize a counter for unique column names
 column_counter <- 1
```

the following set of codes will iterate over the “file_list”, which have all the required files and extract the first column that has the gene ids and the last column that has the expression rate. The extracted data will be append to a new data set with unique column names.

```
for (file_path in file_list) {
  # Read the dataset
  df <- read.table(file_path, header = TRUE, sep = "\t")
  # Extract the first and last columns as data frames
  id_column <- data.frame(df[, 1])
  expression_rate_column <- data.frame(df[, length(df)])
```

```
  # Generate a unique column name for expression_rate_column
  unique_column_name <-
    paste0("expression_rate_", column_counter)

  # Rename the expression_rate column
  colnames(expression_rate_column) <- unique_column_name

  # Append the data to the new dataset
  new_dataset <-
     cbind(new_dataset, id_column, expression_rate_column)

  file_name <- basename(file_path)
  extracted_string <- gsub(".*?([A-Z][^\\.]*)\\.[A-Za-z]+.*",
"\\1", file_name)

  # Append the extracted string to the vector
  file_names_extracted <-
    c(file_names_extracted, extracted_string)

  # Increment the counter
  column_counter <- column_counter + 1
}
```

The next lines of codes will rename the first column of the data frame to “GeneID”, removes the first column and create a sample metadata data frame by extracting the sample names from the data frame “df”.

```
df <- new_dataset
colnames(df)[1] <- "GeneID"
```

```
 rownames(df) <- df[, 1]
 df <- df[, -1]
# Get sample names from countData columns
 sample_names <- colnames(df)
# Create a sample metadata data frame
 colData <- data.frame(
   SampleName = sample_names,
   condition = file_names_extracted
)
 print (colData)
```

The coldata created is displayed bellow.

	SampleName	condition
1	expression_rate_1	BRAF
2	expression_rate_2	Mock
3	expression_rate_3	BRAF
4	expression_rate_4	Mock
5	expression_rate_5	BRAF
6	expression_rate_6	Mock

Table 1 : coldata of the cell model sample

The provided code segment establishes row names in a dataset (colData) based on a column named “SampleName”, creates a copy of the dataset, removes a “SampleName” from the colData , renames the remaining column and verifies the row names in “df” match the column names in “colData”.

```
# Set "SampleName" as row names
 rownames(colData) <- colData$SampleName
 row_names <- rownames(colData)
```

```
 colDataCopy <- colData


# Remove the "SampleName" column from the data frame
 colData <- colData[, -which(names(colData) == "SampleName")]
 colData = data.frame(colData)
 rownames(colData) <- row_names
 colnames(colData) <- 'condition'
# Verifying if the row names in count data match with the the
column names in colData.
 all(colnames(df) %in% rownames(colData))
# Verifying if they are in the same order
 all(colnames(df) == rownames(colData))
```

The above code to check the row names in count data matches the column names in colData returned TRUE. Upon establishing the “colData” and dataframe, the next step involves initializing the “DESeqDataSet”.

```
# Initilize the DESeqDataSet instance
 dds_cell <- DESeqDataSetFromMatrix(df, colData,
            design = ~ condition)
 dds_cell
```

Now to perform differential gene expression analysis, the following code is used.

```
#relevel(), is used to change the reference level for a
categorical
# variable.
# Subsequent analyses will compare other conditions to the "Mock"
dds_cell$condition <- relevel(dds_cell$condition, ref = 'Mock')
```

```
# Perform differential gene expression analysis

 dds_cell <- DESeq(dds_cell)
```

following the completion of differential expression analysis, the “results” function was employed to compute the statistical metrics for differential expression. The results then stored in a variable called “dds_res_cell” for further use.

```
 dds_res_cell = results(dds_cell)

 dds_res_cell
```

Presented below (table 2) are the outcomes derived from the executed codes.

```
log2 fold change (MLE): condition BRAF vs Mock
Wald test p-value: condition BRAF vs Mock
DataFrame with 67060 rows and 6 columns
                   baseMean log2FoldChange     lfcSE      stat    pvalue      padj
                  <numeric>      <numeric> <numeric> <numeric> <numeric> <numeric>
ENSG00000223972.5  0.489053       -2.32832  3.814719 -0.610353 0.5416282        NA
ENSG00000227232.5 10.935335       -1.82881  0.913805 -2.001313 0.0453587  0.127145
ENSG00000278267.1  0.342338        1.99522  4.040211  0.493841 0.6214184        NA
ENSG00000243485.5  0.852467       -0.61030  3.108784 -0.196315 0.8443638        NA
ENSG00000284332.1  0.000000             NA        NA        NA        NA        NA
...                     ...            ...       ...       ...       ...       ...
ENSG00000276017.1  0.000000             NA        NA        NA        NA        NA
ENSG00000278817.1  1.694011       -4.11246   2.19615 -1.872572 0.0611275        NA
ENSG00000277196.4  0.491311        1.10295   3.86419  0.285428 0.7753160        NA
ENSG00000278625.1  0.000000             NA        NA        NA        NA        NA
ENSG00000277374.1  0.000000             NA        NA        NA        NA        NA
```

Table 2: output summary of differential expression analysis of cell model data

The presented dataset represents the outcomes derived from differential expression analysis between the conditions denoted as “BRAF” and “Mock”. Comprising 67060 rows and 6 columns, each row signifies a unique gene identified by Ensembl Gene identifier (ENSG) and six columns encompass crucial statistical measures characterizing the genes differential expression behavior under the condition provided.

The “baseMean” represents mean expression level of a gene across the condition, “log2FoldChange” indicates the magnitude of change in gene expression levels between “BRAF and “Mock”. The “lfcSE” stands for the standard error associated with log2 fold change and the “stat” refers to a statistical metric calculated for hypothesis testing. The “pvalue” represent the significance of the expression change and “padj” represent the adjusted p value, considering the multiple testing corrections to evaluate the significance.

- **MA plot**

MA plot is used to visualize the results generated from DESeq2. In MA plot the M stands for log fold change and A represents average expression of data points across conditions. This plot is helpful for the statistical analysis, interpretation of the data and visually identify the genes exhibiting differential expression between the conditions provided.

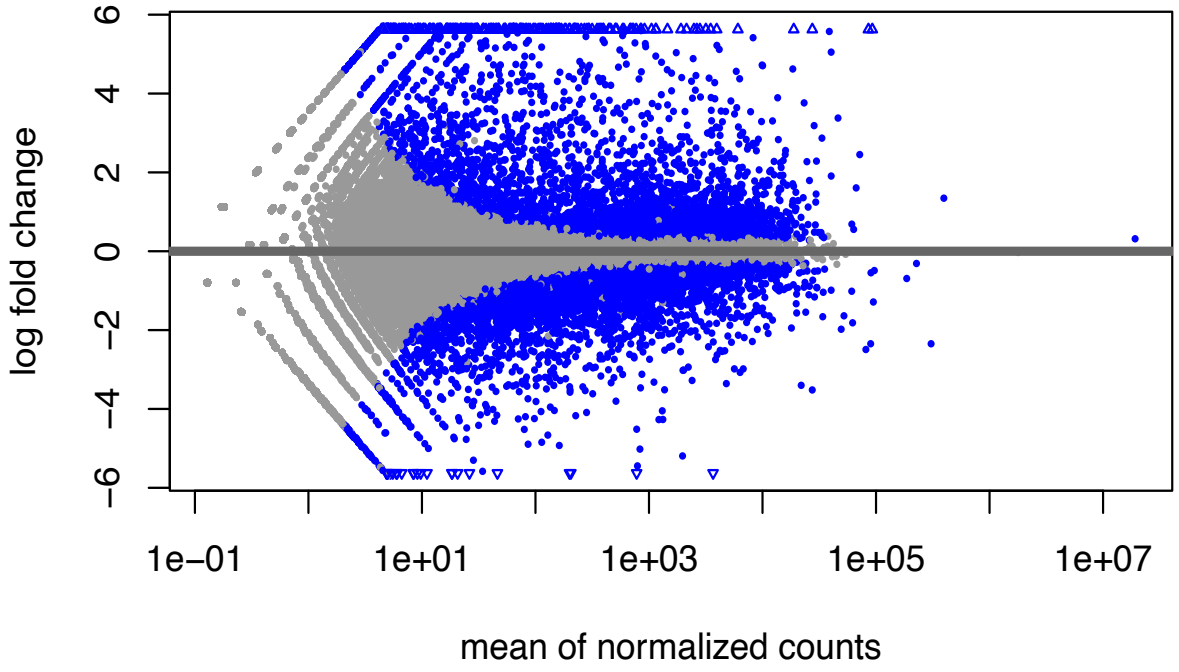

Figure 6: MA plot of the cell model data, here log fold change is plotted against mean of normalized counts.

The preceding MA plot (figure 6) provides an insight into the relationship between log fold change and average expression level of genes across the conditions. The “X” axis represents the average expression level of genes across the condition and the “Y” axis represents log fold change. The MA plot is a scatter plot and each points represents genes. the genes with substantial change in expression between conditions appear farther away from the centerline, reflecting a higher log2 fold change. The gray and blue colors represent differences between statistically significant and non-significant data in which blue represents significantly differentially expressed genes. The genes below the center line are downregulated and the genes above the center line are upregulated according to the conditions provided. In this analysis the reference is “Mock”, so the genes above the center line indicate the upregulation in the “BRAF” condition compared to the reference group (Mock).

DESeq2 automatically normalizes count data for differential expression analysis. However, in certain cases, visualizations may benefit from using normalized raw counts. To accomplish this, a helpful method involves utilizing the “vst()” function. This function conducts a variance stabilized transformation on the count data, accounting for variations in sample library sizes. The transformation process is achieved using the following code.

```
 vsd <- vst(dds_cell)
# calculating sample distances
 sample_distances <- assay(vsd) %>%
   t() %>%
   dist() %>%
   as.matrix()
```

- **Multi-Dimensional Scaling (MDS) plot**

MDS (multi-dimensional scaling) plot represents the relationships among the samples. In the MDS plot (figure 7), the closer the samples, more similarity in the count profiles. The plot is created using DESeq2 by, first calculating 'sample distances' and subsequently visualize them. The libraries were installed beforehand.

```
 mds_data <- data.frame(cmdscale(sample_distances))
```

```
mds <- cbind(mds_data, as.data.frame(colData(vsd)))
ggplot(mds, aes(X1, X2, shape = condition)) +
  geom_point(size = 3) +
  theme_minimal()
```

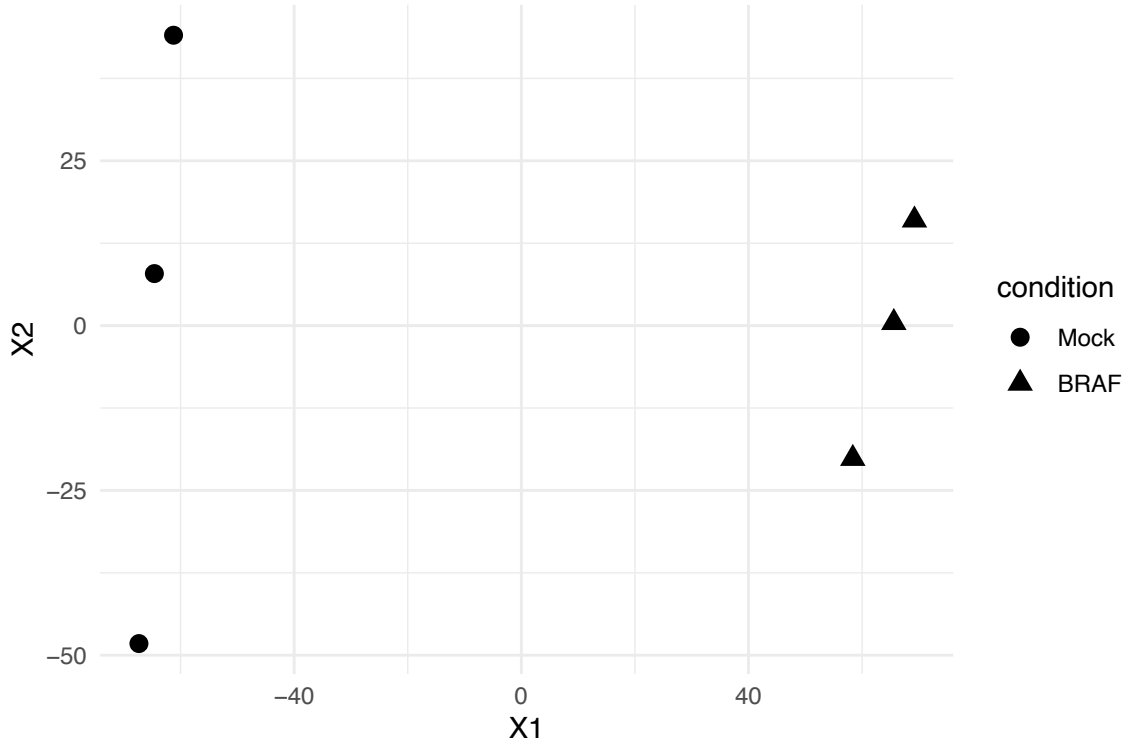

Figure 7: Multi-dimensional scaling plot of cell model data, this represents the relationship among samples.

In the above MDS plot the X1 and X2 represent arbitrary dimensional axes. The samples are positioned based on their gene expression profiles. The BRAF samples seems to have similar gene expression profiles and they cluster together. In case of Mock, the samples express a slight difference in their expression rate. The positioning of BRAF samples indicate that, it has relatively different gene expression profiles compared to Mock.

- **Heatmap**

Heat map is used in differentially expression analysis to visualize the gene's expression change between conditions. To generate the heat map, the data from differential expression analysis was sorted based on log2fold change in a descending order. Then selected the top

20 genes from the sorted list. A sample data frame also created from an existing data frame to include the details of the sample in heatmap. Pheatmap is an R package used to create heatmaps in R. The library for heatmap was called along with other libraries in the biggening. The following lines of codes were used,

```
#to create top20 genes
 top_20_genes <- order(dds_res$log2FoldChange,
                 decreasing=TRUE)[1:20]
#sample data frame
 annotCol <- as_tibble(colDataCopy) %>%
   column_to_rownames('SampleName') %>%
   select(condition) %>%
   as.data.frame()
#to plot heatmap

 pheatmap(assay(vsd)[top_20_genes, ], cluster_rows=TRUE,
  Show_rownames=TRUE, cluster_cols=FALSE,
  annotation_col=annotCol)
```

In the heatmap (figure 8), the conditions were given in different colors. Each row indicates each gene from the top 20 genes that was created with log2fold change. A color gradient is given to indicate the increasing numerical values associated with gene expression, with blue being the least and red being the highest. The higher intensity represents higher expression level of genes. The dendrogram on the left side of the heatmap represents the hierarchical clustering of rows (genes) based on their similarity in the dataset. The genes that are closer together on the dendrogram have similar characteristics.

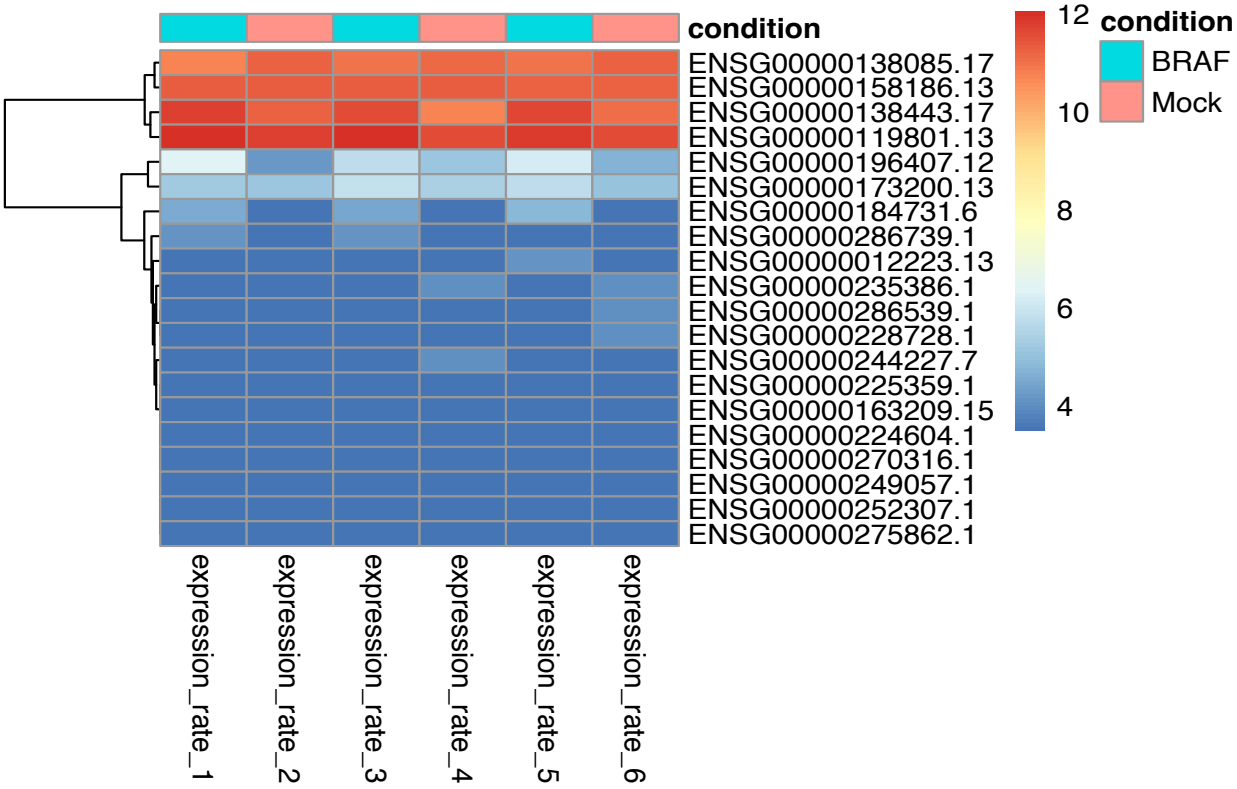

Figure 8 : Heatmap of the cell model data, depicting the gene expression change between conditions.

- **Principal Component Analysis (PCA) plot**

PCA stands for Principal Component Analysis. Libraries were called and the following sets of codes were used to create the PCA plot.

```
# Perform PCA on the vsd dataset, considering "condition" as a
grouping variable

 pca_data <- plotPCA(vsd, intgroup = c("condition"),

  returnData    = TRUE)


# Calculate the percentage of variance explained by each
principal component


 percentVar <- round(100 * attr(pca_data, "percentVar"))
```

```
# Create a scatter plot using ggplot2
  ggplot(pca_data, aes(x=PC1, y=PC2, color=condition)) +
  geom_point(size=3) +
  theme_minimal() +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance"))
```

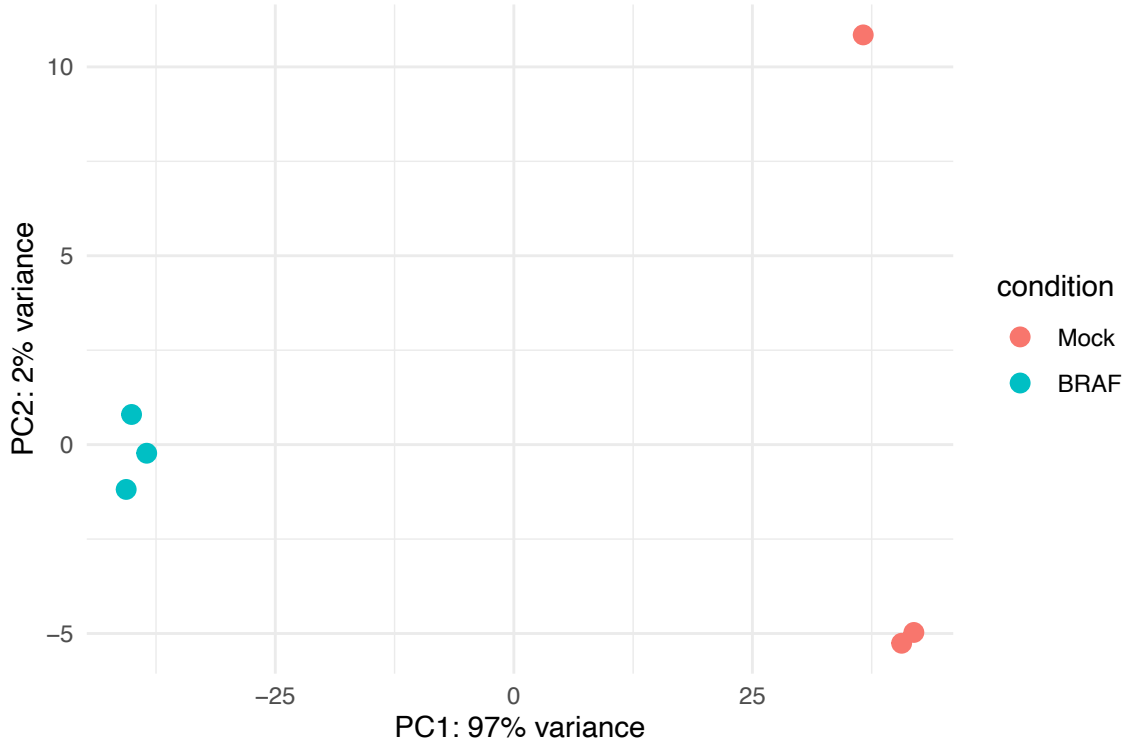

Figure 9: principal component analysis plot of cell model.

In the above PCA plot (figure 9) principal component 1 (PC1) captures 97%, which is the most significant variations present in the dataset. Principal component 2 (PC2) with much lower percentage of variance explained, captures a smaller amount of variability. In this PCA plot, it is noticeable that one sample belongs to the condition “Mock” is displaced from the group. Even though the sample is displaced far from the other two samples of Mock, the displacement

occurred along an axis with low variance (2%). For that reason, the variation it captures isn't significant compared to other components.

- **Volcano plot**

Volcano plot is used for visualization to assess the significance and fold change between different experimental conditions. The following sets of codes were used to create the volcano plot.

```
# Extracting required data from the results
 volcano_data <- as.data.frame(dds_res_cell)
 volcano_data$logPadj <- -log10(volcano_data$padj)



# Plotting the volcano plot
   ggplot(volcano_data, aes(x=log2FoldChange, y=logPadj)) +
   geom_point(aes(color = padj < 0.05), alpha = 0.6) +
   theme_minimal() +
   labs(title = "Volcano Plot", x = "Log2 Fold Change",
   y = "-  Log10 Adjusted p-value") +
   scale_color_manual(name = "Significance",
   values = c("gray", "red")) + theme(legend.position = "top")
```

The volcano plot (figure 10) illustrates the relationship between the "log2 fold change", representing the magnitude of change in gene expression between the conditions on the X axis, and the "-log10 adjusted p value", reflecting the statistical significance of these differences on Y axis. Genes distributed to the right of zero (positive values) on x axis, indicate upregulated genes compared to Mock (reference) and the ones to the left is downregulated.

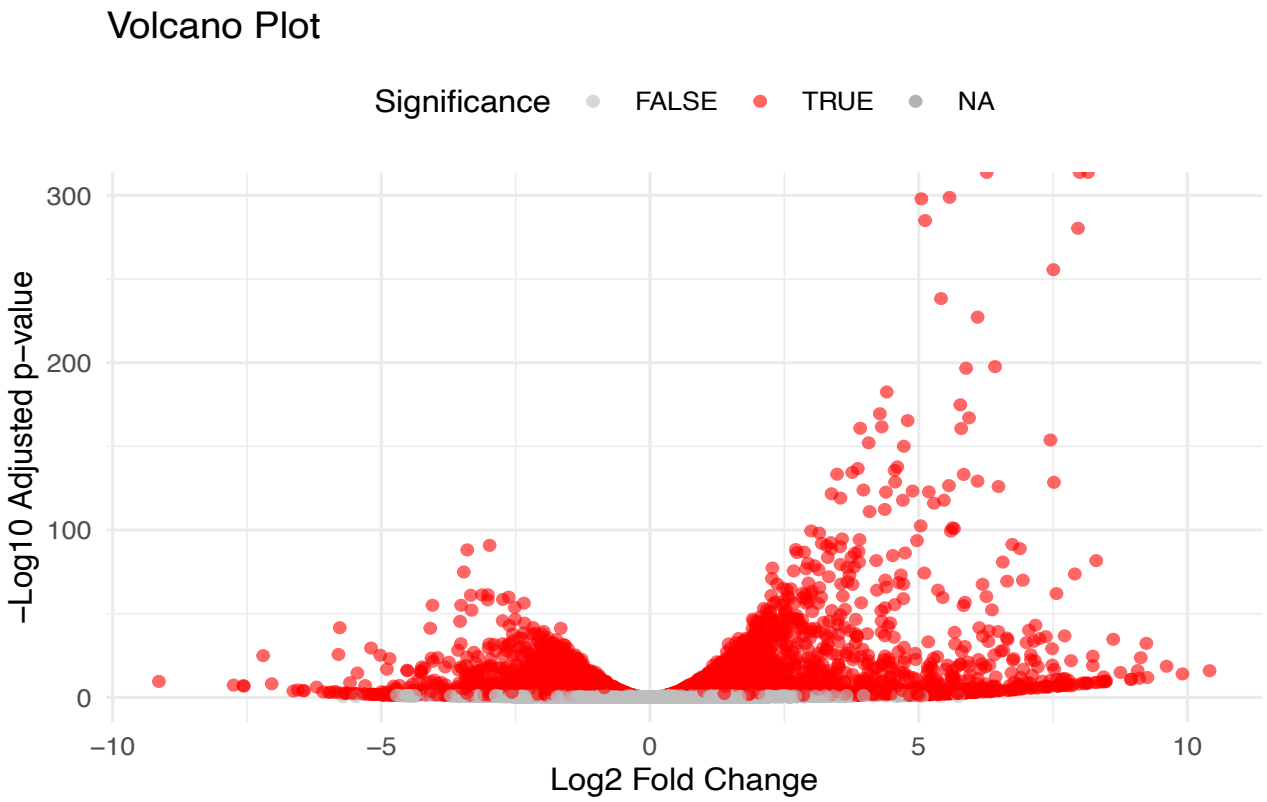

Figure 10: Volcano plot of cell model data showing the significance and fold change between two conditions.

2.2 Clinical Data Processing

2.2.1 Data acquisition from TCGA

The second analysis focused on comparing patient data characterized by the BRAF V600E mutation against a cohort of Skin Cutaneous Melanoma (SKCM) samples, specifically excluding those bearing the BRAF V600E mutation. The BRAF V600E mutation is a specific mutation found in BRAF gene where valine at position 600 is substituted by glutamic acid (V600E).

Initially, the Mutation Annotation Format (MAF) file encompassing BRAF V600E mutation file details was retrieved from FireBrowse, which is a website developed by Broad Institute and is a hub for exploring different cancer genomic datasets derived from projects such as The Cancer Genome Atlas (TCGA). Utilizing the tumour sample barcodes as identifiers, files

containing data specific to the BRAF V600E mutation were retrieved from the TCGA database and then stored into the file named “BRAF V600E”.

The retravel of Skin Cutaneous Melanoma (SKCM) files was executed with the help of programming language Python. The python used in this project is , version 3.9.6

```
# downloaded MAF of all the SKCM files in
   TCGA database (including BRAF mutated files)

import requests
import json

fields = [
    "file_name",
    "cases.submitter_id",
    "cases.samples.sample_type",
    "cases.disease_type",
    "cases.project.project_id"
    ]

fields = ",".join(fields)

files_endpt = "https://api.gdc.cancer.gov/files"

filters = {
    "op": "and",
    "content":[
        {
        "op": "in",
        "content":{
            "field": "cases.project.primary_site",
```

```
            "value": ["skin"]
            }
        },
        {
        "op": "in",
        "content":{
            "field": "files.experimental_strategy",
            "value": ["RNA-Seq"]
            }
        },
        {
        "op": "in",
        "content":{
            "field": "files.data_format",
            "value": ["tsv"]
            }
        }
    ]
}

params = {
    "filters": filters,
    "fields": fields,
    "format": "TSV",
    "size": "2000"
    }
```

```
try:
    response = requests.post(files_endpt,
    headers={"Content- Type": "application/json"}, json=params)

    if response.status_code == 200:
        r1 = response.content.decode("utf-8")
        print("Response received.")
        data_path
      = "/Users/methungeorge/Desktop/thesis/datasets2/data.tsv"

        # Save the response content to a file
        with open('data.tsv', 'w') as tsv_file:
            tsv_file.write(r1)

        print("Data saved as data.tsv")
    else:
        print(f"Error: {response.status_code}
        - {response.text}")

except requests.exceptions.RequestException as e:
    print(f"Request Exception: {e}")
```

The MAF files downloaded is then saved as “data.tsv”. then all the file names were extracted from the MAF file with the use of following codes,

```
import csv

input_file =   "/Users/methungeorge
               /Desktop/thesis/datasets2/data.tsv"
output_file = "/Users/methungeorge/Desktop/thesis

/datasets2/extracted_data.txt"
```

```
column_index = 4


extracted_values = []


with open(input_file, 'r', newline='') as file:
    reader = csv.reader(file, delimiter='\t')
    for row in reader:
        if len(row) > column_index:
            extracted_values.append(row[column_index])


# Save extracted values to a text file
with open(output_file, 'w') as txtfile:
    for value in extracted_values:
        txtfile.write(str(value) + '\n')
```

The extracted file names of all SKCM is saved in extracted_data.txt. the next step was to filter the files by checking if they are present on the BRAF V600E files, which has been downloaded and saved on the file named BRAF V600E. For that, the next lines of codes were employed.

```
import os


file1_path = "/Users/methungeorge
              /Desktop/thesis/datasets2/extracted_data.txt"
file2_directory = "/Users/methungeorge
              /Desktop/thesis/datasets2/BRAF_V600E"
file3_path = "/Users/methungeorge/Desktop
              /thesis/datasets2/new/new.txt"
# Read the list of names from file1
with open(file1_path, "r") as file1:
    file1_names = {line.strip() for line in file1}


# List the files in file2_directory
```

```
file2_files = set(os.listdir(file2_directory))


# Find names in file1 that do not correspond to files in
file2_directory
missing_names = file1_names - file2_files


# Save missing names to file3
with open(file3_path, "w") as file3:
    for missing_name in missing_names:
        file3.write(missing_name + "\n")


print(f"Missing names have been saved to {file3_path}")
```

The files were filtered and then saved as a text file named “new.txt”. the “new.txt” file contains the file names of SKCM data except the BRAF V600E files. One more filtration is carried out on the file “new.txt” to extract the “tsv” files.

```
input_file_path = '/Users/methungeorge
                  /Desktop/thesis/datasets2/new/new.txt'
output_file_path = '/Users/methungeorge
                 /Desktop/thesis/datasets2/new/tsv_files.txt'


with open(input_file_path, 'r') as file:
    files = file.read().splitlines()


# Filter out files that end with .gz or .rna_fusion.tsv
filtered_files = [file for file in files if not
file.endswith('.gz') and not file.endswith('.rna_fusion.tsv')]


# Write the filtered list of files to the output text file
with open(output_file_path, 'w') as file:
    for file_name in filtered_files:
        file.write(file_name + '\n')
```

The next script cross-references the file names listed in “tsv_files.txt” with the identifiers in the previously downloaded “data.tsv” file. When matches are found, it retrieves the corresponding IDs and store them in a new file named “new.txt” to facilitate the programmatic downloads from the TCGA database. The files with the IDs in new.txt is then downloaded with the aid of the following codes.

```
import os
import requests
import re

# Read the list of file IDs from the text file "new.txt"
with open("/Users/methungeorge/
          Desktop/thesis/datasets2/new.txt", "r") as id_file:
     file_ids = [line.strip() for line in id_file]

# Specify the download directory
download_dir = "/Users/methungeorge
                /Desktop/thesis/datasets2/new/trial"

# Loop through each file ID and download the corresponding files
for file_id in file_ids:
    data_endpoint = "https://api.gdc.cancer.gov/
                     data/{}".format(file_id)

    response = requests.get(data_endpoint,
                headers={"Content-Type": "application/json"})

    if response.status_code == 200:
        response_head_cd = response.headers
                           ["content-Disposition"]
        file_name = re.findall("filename=(.+)"
                    , response_head_cd)[0]
```

```
        # Specify the local path to save the downloaded file
        save_path = os.path.join(download_dir, file_name)

        with open(save_path, "wb") as output_file:
            output_file.write(response.content)
        print(f"Downloaded: {file_name}")
    else:
        print(f"Failed to download file with ID: {file_id}")
```

The datasets were then stored in a directory called “datasets2” and proceeded to differential expression analysis using DESeq2 computational tool. The following codes were executed in R programming interface.

2.2.2 Differential expression analysis of clinical data

The following codes will create a data frame and list all the files,

```
# Create an empty data frame to store the extracted data
new_dataset <- data.frame()

# Read the dataset
df <- read.table("/Users/methungeorge/Desktop/thesis
    /datasets2/data_for_second_analysis_2/datasets/BRAF_V600E
    /0bdec778-829d-4cba-8d0f-dcaa0d8803b1.rna_seq.augmented_
     star_gene_counts.tsv", header = TRUE, sep = "\t")
df <- df[-(1:4), ]
new_dataset <-  df$gene_id
# Recursively list all .tsv files
file_list <- list.files(path = '.',
     pattern = "\\.tsv$",  recursive = TRUE, full.names = FALSE)
```

The next set of codes will read multiple files in “file_list”, extract specific column that contain the counts, combines them into a new dataset, identifies and count specific patterns in the file names.

```
# Loop through the list of files, read each file, extract the first and last columns, and append them to the new dataset with unique column names

column_counter <- 0

file_names_extracted <- character(0)

for (file_path in file_list) {

  # Read the dataset

  df <- read.table(file_path, header = TRUE, sep = "\t")

  df <- df[-(1:4), ]

  # Extract the first and last columns as data frames

  id_column <- data.frame(df[, 1])

  expression_rate_column <- data.frame(df[, length(df)])


  # Generate a unique column name for expression_rate_column

  unique_column_name <- paste0("expression_rate_"

                        ,  column_counter)


  # Rename the expression_rate column

  colnames(expression_rate_column) <- unique_column_name


  # Append the data to the new dataset

  if(column_counter == 0) {

    new_dataset <- cbind(id_column, expression_rate_column)

  } else {

    new_dataset <- cbind(new_dataset, expression_rate_column)

  }
```

```
  # Regular expression to match either "BRAF V600E"
    or  "SKCM_except_BRAF_V_600E"
  pattern <- "(?i)BRAF_V600E|SKCM_except_BRAF_V_600E"

  # Extract the match
  extracted_string <- regmatches(file_path,
                      regexpr(pattern, file_path))

  # Append the extracted string to the vector
  file_names_extracted <- c(file_names_extracted
                          , extracted_string)

  # Increment the counter
  column_counter <- column_counter + 1
}

file_names_extracted <- lapply(file_names_extracted
            , function(x) { gsub("SKCM_except_BRAF_V_600E"
            , "SKCM_except_BRAF_V600E", x)
})

file_names_extracted <- unlist(file_names_extracted)

# Print the extracted strings
print(table(file_names_extracted))
```

the extracted file names are displayed below,

file_names_extracted

BRAF_V600E	SKCM_except_BRAF_V600E
123	371

The next set of codes will rename the first column of the data frame (df), sets row names according to the values on that column, then remove the first column from the data frame. After that ColData data frame will be created which contain information about the sample and condition and is necessary for the differential expression analysis.

```
df <- new_dataset
colnames(df)[1] <- "gene_id"
rownames(df) <- df[, 1]
df <- df[, -1]
# Get sample names from countData columns
sample_names <- colnames(df)


# Create a sample metadata data frame
colData <- data.frame(
  SampleName = sample_names,
  condition = file_names_extracted
)
```

The following codes will organize and renames the metadata columns, focusing on the “samplename” column, which becomes row names. The lines of code followed by this will check the alignment between column names in one dataset (df) and row names in metadata (colData). This is important to ensure the proper association between sample information and the actual data for an accurate analysis.

```
# Set "SampleName" as row names
rownames(colData) <- colData$SampleName
row_names <- rownames(colData)
colDataCopy <- colData
```

```
# Remove the "SampleName" column from the data frame

colData <- colData[, -which(names(colData) == "SampleName")]

colData = data.frame(colData)

rownames(colData) <- row_names

colnames(colData) <- 'condition'

# Verifying if the row names in count data match with the the
column

# names in colData.

all(colnames(df) %in% rownames(colData))

# Verifying if they are in the same order

all(colnames(df) == rownames(colData))

# round function is used to for better data presentation and
data integrity

df <- round(df)
```

Now the focus shifts to conducting differential expression analysis.

```
# Initilize the DESeqDataSet instance

dds <- DESeqDataSetFromMatrix(df, colData, design = ~ condition)

# relevel(), is used to change the reference level for a
categorical

# variable.

# Subsequent analyses will compare other conditions to the "BRAF
V600E"

dds$condition <- relevel(dds$condition, ref =
'SKCM_except_BRAF_V600E')



# Perform differential gene expression analysis

dds <- DESeq(dds)

# for testing p value 0.05, use results(dds, alpha = 0.05)

dds_res = results(dds)
```

The results of the differential expression analysis performed is now stored in the variable called “dds_res” and the summary of the dds_res is depicted in table 3.

```
log2 fold change (MLE): condition BRAF V600E vs SKCM except BRAF V600E
Wald test p-value: condition BRAF V600E vs SKCM except BRAF V600E
DataFrame with 60660 rows and 6 columns
                    baseMean log2FoldChange     lfcSE      stat    pvalue      padj
                   <numeric>      <numeric> <numeric> <numeric> <numeric> <numeric>
ENSG00000000003.15  9.410822      0.1187279 0.0850635  1.395756  0.162788  0.338607
ENSG00000000005.6   0.147467     -0.4772692 1.7724726 -0.269267  0.787724        NA
ENSG00000000419.13 27.642335      0.0841872 0.0599574  1.404117  0.160284  0.335104
ENSG00000000457.14  1.983343     -0.0175932 0.1068945 -0.164584  0.869271  0.932541
ENSG00000000460.17  1.913303      0.0135116 0.1078519  0.125280  0.900302  0.948137
...                      ...            ...       ...       ...       ...       ...
ENSG00000288669.1   0.000000             NA        NA        NA        NA        NA
ENSG00000288670.1   3.228899     -0.0461039 0.0840442 -0.548567  0.583303  0.746813
ENSG00000288671.1   0.000000             NA        NA        NA        NA        NA
ENSG00000288674.1   0.000000             NA        NA        NA        NA        NA
ENSG00000288675.1   0.254737     -0.1319051 0.2542045 -0.518894  0.603835  0.761228
```

Table 3 : information about the second differential expression analysis results.

The table (table 3) is a summary of the differential expression analysis performed using clinical data. The results describe the differences in gene expression between two conditions: "SKCM except BRAF V600E" and "BRAF V600E". The "SKCM except BRAF V600E" condition serves as the reference group and any reported fold change indicate differences relative to this reference.

To visualize the results and compare different parameters, different types of graphs were used. The libraries for the graphs were called beforehand.

- **MA Plot**

MA plot is used to visualize the results generated from DESeq2. In MA plot the M stands for log fold change and A represents average expression of data points across conditions.

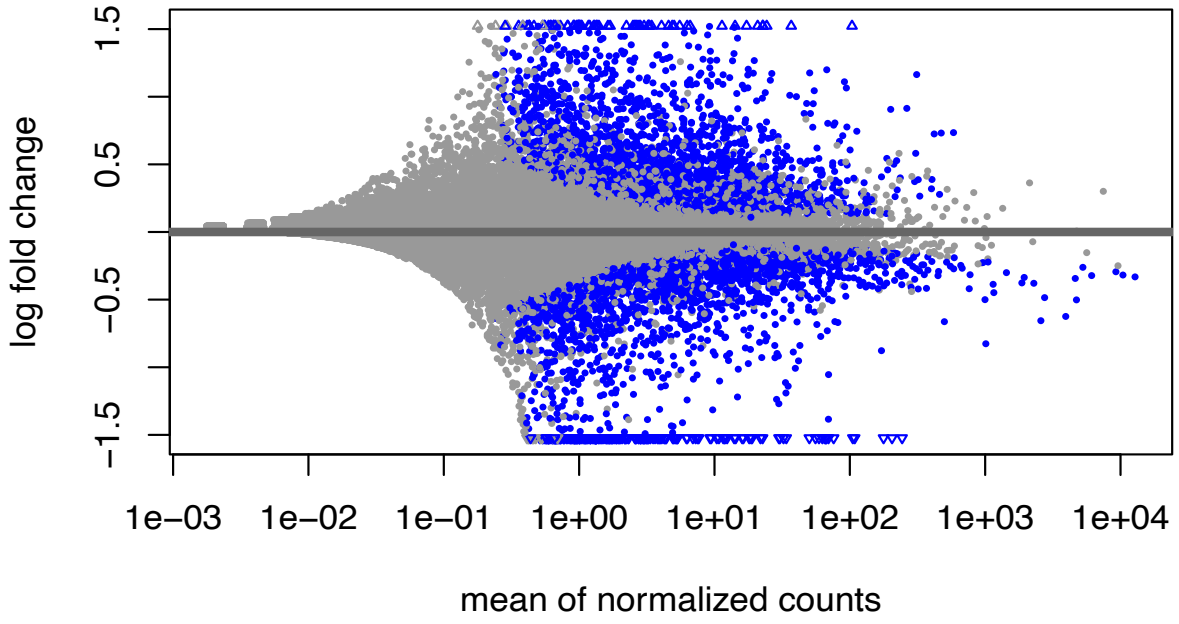

Figure 11: MA plot of the clinical data depicting the relation of log fold change and mean normalized counts.

The MA plot (figure 11) is a scatter plot and each points represents genes. the genes with substantial change in expression between conditions appear farther away from the centerline, reflecting a higher log2 fold change. The blue scatter points represent significantly differentially expressed genes. The triangles indicate higher fold changes, and the direction of the triangle indicates the direction of that fold change.

Visualizations may benefit from using normalized raw counts. To accomplish this, a helpful method involves utilizing the “vst()” function.

```
vsd <- vst(dds)
# calculating sample distances
sample_distances <- assay(vsd) %>%
  t() %>%
  dist() %>%
  as.matrix()
```

- **Multi-Dimensional Scaling (MDS) Plot**

The MDS plot gives information about the relationship among samples. The following codes were used to create an MDS plot.

```
mds_data <- data.frame(cmdscale(sample_distances))
mds <- cbind(mds_data, as.data.frame(colData(vsd)))
ggplot(mds, aes(X1, X2, shape = condition)) +
  geom_point(size = 3) +
  theme_minimal()
```

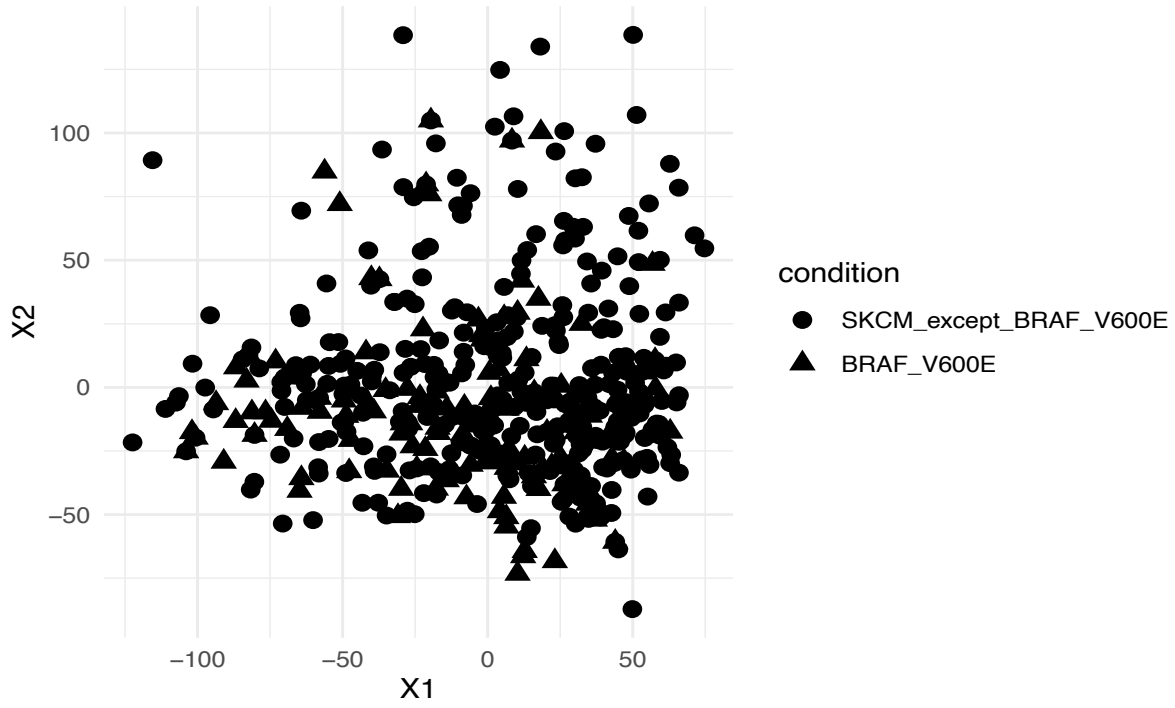

Figure12 : MDS plot of clinical data depicting the relationship among sample conditions.

The MDS plot (figure 12) includes data for two conditions: 'BRAF V600E' and 'SKCM_except_BRAF_V600E'. The conditions are represented by different shapes, circles for "SKCM_except_BRAF_V600E" and triangles for "BRAF_V600E". The MDS plot shows that the samples exhibit similar gene expression profiles. The overlap here suggests that the two conditions have some underlying similarities.

- **Heatmap**

To create the heatmap of the top 20 genes, the differential expression analysis data has been sorted based on log2fold change and a coldata has also created to give information about sample.

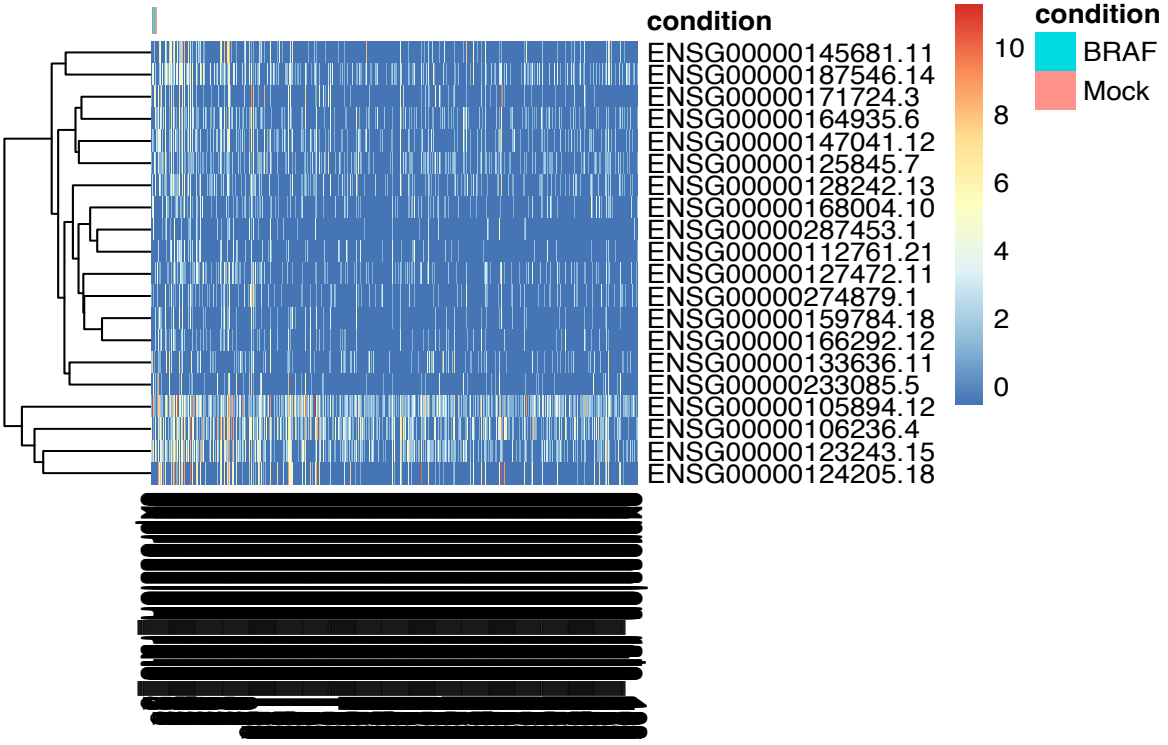

Figure 13: Heatmap of the top 20 genes in clinical data sorted by log2fold change.

The following codes were used to create the heatmap (figure 13)

```
top_20_genes <- order(dds_res$log2FoldChange
                , decreasing=TRUE)[1:20]
annotCol <- as_tibble(colDataCopy) %>%
  column_to_rownames('SampleName') %>%
  select(condition) %>%
  as.data.frame()
pheatmap(assay(vsd)[top_20_genes,],
         cluster_rows=TRUE,show_rownames=TRUE,
         cluster_cols=FALSE, annotation_col=annotCol)
```

In this heat map (figure 13), the rows are labelled with gene names. The columns are labelled with conditions, such as “BRAF V600E” and “SKCM_except_BRAF_V600E”. The cells in the grid are colored according to the expression rate of the genes in the different conditions. The expression rate is represented by a color scale, with red representing high expression and blue representing low expression.

- **Principal Component Analysis (PCA) Plot**

To create PCA plot, the following codes were used,

```
# Perform PCA on the vsd dataset, considering "condition" as a
grouping variable
pca_data <- plotPCA(vsd, intgroup = c("condition"), returnData
= TRUE)
# Calculate the percentage of variance explained by each
principal component
percentVar <- round(100 * attr(pca_data, "percentVar"))
# Create a scatter plot using ggplot2
ggplot(pca_data, aes(x=PC1, y=PC2, color=condition)) +
  geom_point(size=3) +             # Add points to the plot
  theme_minimal() +                # Apply a minimalistic theme
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance”"))
```

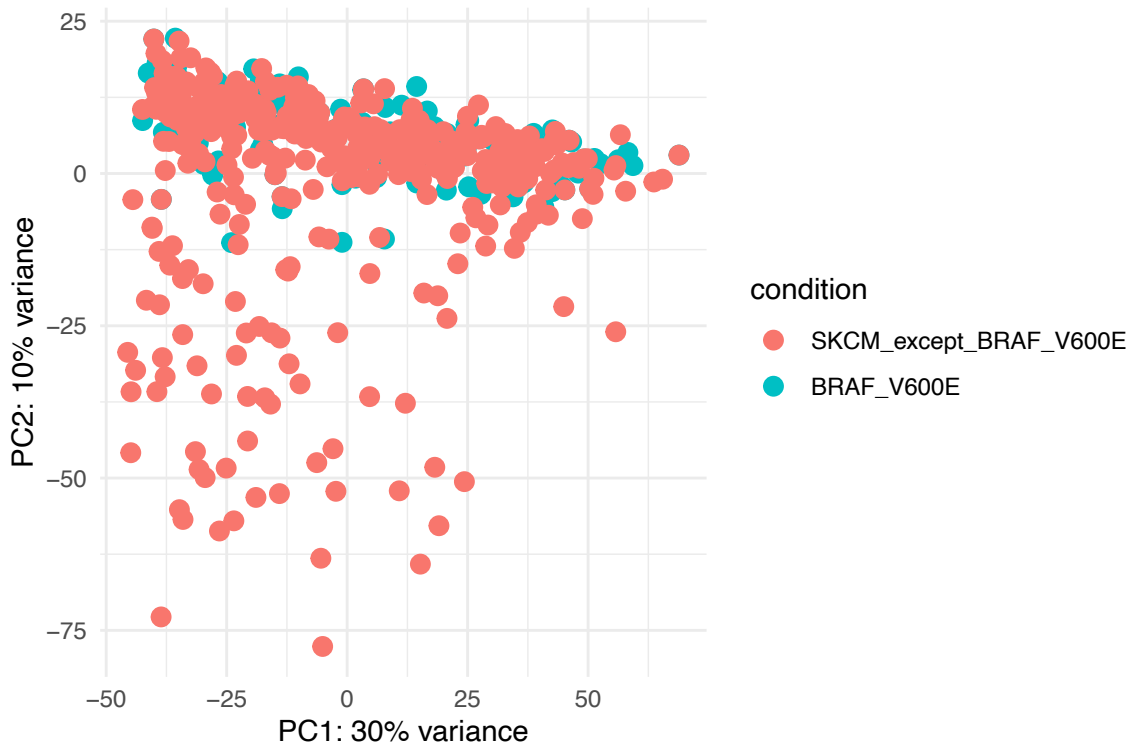

Figure 14: PCA plot of the clinical data

The PCA plot of clinical data (figure 14) shows 30% variance in principal component 1 and 10% in principal component 2. The distribution of samples indicates the similarities among them. Samples closer together share more similarities. PC1, which explains a higher percentage of variance has a stronger influence on the positioning of the samples.

- **Volcano Plot**

The subsequent code snippets were utilized to generate volcano plot.

```
# Extracting required data from the results
volcano_data <- as.data.frame(dds_res)
volcano_data$logPadj <- -log10(volcano_data$padj)
```

```
# Plotting the volcano plot
ggplot(volcano_data, aes(x=log2FoldChange, y=logPadj)) +
  geom_point(aes(color = padj < 0.05), alpha = 0.6) +
  theme_minimal() +
  labs(title = "Volcano Plot", x = "Log2 Fold Change",
  y = "-Log10 Adjusted p-value") +
  scale_color_manual(name = "Significance",
   values = c("gray", "red")) +
  theme(legend.position = "top")
```

Figure 15: Volcano plot of the clinical data

This volcano plot (Figure 15) shows the statistically significant changes in gene expression between “BRAF V600E” and “SKCM except BRAF V600E”. The genes scattered on the right of “0” on X axis shows a negative fold change indicating the down regulated genes and the scattered genes on the left shows an upregulation. Higher the genes appear in Y axis less significant the gene is.

2.3 Post-differential expression analysis

2.3.1 Cell Model Data

The “dds_res_cell” contains the differential expression analysis result of the cell model data. To filter out the significant genes in that data first the adjusted p-value (padj) is set to 0.05 and any gene with a higher p value considered not significant. Followed by this filtration, the up and down regulated genes were filtered from the significant genes by adjusting the log2fold change. The similar procedure is done on clinical data to filter the significantly regulated genes and in that up and down regulated genes. The following codes were used to get the results.

```
#filter the significantly regulated genes of cell model data
sig_cell <- subset(dds_res_cell, padj < 0.05 )
print(nrow(sig_cell))
#7024 significantly regulated genes
#To extract the row names
sig_cell_names <- rownames(sig_cell)
# Remove version information from all gene IDs in the vector
sig_cell_names <- gsub("\\..*", "",sig_cell_names)
```

```
#upregulated genes in cell model data
sig_cell_up <- subset(sig_cell, log2FoldChange > 0)
```

```
sig_cell_up <- rownames(sig_cell_up)
sig_cell_up <- gsub("\\..*", "",sig_cell_up)
#downregulated genes in cell model data
sig_cell_down <- subset(sig_cell, log2FoldChange < 0)
sig_cell_down <- rownames(sig_cell_down)
sig_cell_down <- gsub("\\..*", "",sig_cell_down)
```

There were 3718 upregulated genes and 3306 down regulated genes in cell model data. The genes saved in these variables are in “Ensembl gene ID “ format, in the next line of codes this is converted into gene names for better understanding.

```
library(biomaRt)
geneIDs_cell <- sig_cell_names
ensembl <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
gene_symbols_sig_cell <- getBM(attributes
                     = c("external_gene_name"),
                      filters = "ensembl_gene_id",
                      values =geneIDs_cell,
                      mart = ensembl)
```

The “gene_symbols_sig_cell” is a data frame with “external_gene_name” as the name of the column followed by gene names of significantly regulated genes in cell model data. The gene names then extracted from the data frame for future analysis.

```
gene_symbol_cell_vector <- gene_symbols_sig_cell$
                             external_gene_name
```

2.3.2 Clinical Data

The procedures done to clinical data is similar to the cell model data. To begin with, the significantly regulated genes of clinical data is first filtered out. Then separated the up and down regulated genes. The variable “dds_res” contains the differential expression result of clinical data.

```
sig_clinical <-  dds_res[!is.na(dds_res$padj) &
                 dds_res$padj < 0.05, ]
```

The differential expression result of clinical data had 60660 but after adjusting the p value to get the significantly regulated genes it was found to be 3820 genes.

```
#upregulated genes in clinical data
sig_clinical_up <- subset(sig_clinical, log2FoldChange > 0)
sig_clinical_up <- rownames(sig_clinical_up)
sig_clinical_up <- gsub("\\..*", "",sig_clinical_up)
#downregulated genes in clinical data
sig_clinical_down <- subset(sig_clinical, log2FoldChange < 0)
sig_clinical_down <- rownames(sig_clinical_down)
sig_clinical_down <- gsub("\\..*", "",sig_clinical_down)
```

There is 1966 upregulated and 1854 down regulated genes.the library called “biomaRt” is used to convert the gene ids to gene names for better understanding.

```
# extract the rownames
sig_clinical_names <- rownames(sig_clinical)
sig_clinical_names <- gsub("\\..*", "", sig_clinical_names )
#converting gene ids to gene names
library(biomaRt)
geneIDs_clinical <- sig_clinical_names
ensembl <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
gene_symbols_sig_clinical <- getBM(attributes
                        = c("external_gene_name"),
                      filters = "ensembl_gene_id",
                      values =geneIDs_clinical,
                      mart = ensembl)
```

```
#extract the gene names from the data frame
gene_symbol_clinical_vector <- gene_symbols_sig_clinical$
                                external_gene_name
```

The senescent genes file is downloaded from a database called Gene Set Enrichment Analysis (GSEA). The human collection of senescent genes is then selected. There were 35 different gene sets related to senescence in total. It was downloaded and saved as “genesets.v2023.2.Hs.tsv”. Then for better analysis a data frame with this 35 senescence gene set and its corresponding pathway.

```
# Read the TSV file
data <- read.table("/Users/methungeorge/Desktop
       /thesis/senescence_gene_list/genesets.v2023.2.Hs.tsv",
        header = FALSE, sep = "\t")
# Filter rows based on conditions in the first column
filtered_data <- data[data$V1 %in%
                 c("STANDARD_NAME", "GENE_SYMBOLS"), ]
sene_35_genes_in_dataframe <- data.frame(STANDARD_NAME
      = filtered_data$V2[filtered_data$V1 == "STANDARD_NAME"],
                     GENE_SYMBOLS = filtered_data$V2
                     [filtered_data$V1 == "GENE_SYMBOLS"])
```

To evaluate the shared genes with senescence-related genes, the following code segments were used.

```
# Read the TSV file
data <- read.table("/Users/methungeorge
       /Desktop/thesis/senescence_gene_list
       /genesets.v2023.2.Hs.tsv", header = TRUE, sep = "\t")
```

```
# Extract data from the second column based on the condition in
the first column

senescent_genes35 <- data[data[, 1] == "GENE_SYMBOLS", 2]



#senescent_genes35

senescent_genes35 <- paste(senescent_genes35, collapse = ",")

#senescent_genes35 is the single string containing items
separated by commas

gene_list <- unlist(strsplit(senescent_genes35, ","))

unique_sene_genes <- unique(gene_list)
```

the variable “unique_sene_genes” is a vector contains all the senescence genes.

The “sig_cell_up_genenames” is a data frame of gene names, which contain the significantly regulated genes in cell model which are upregulated. The data frame is created from the differential expression analysis result of cell model data, followed by setting p-adjusted value and log2fold change (the codes used are similar to the codes given earlier regarding changing the “padj2 value and log2fold change) the gene ids are then converted into gene names (the code snippet is given in the previous pages).

```
sig_cell_up_genenames

cell_up <- sig_cell_up_genenames$external_gene_name

cell_up_intersect_sene35 <-  intersect(unique_sene_genes
                            , cell_up)

print (length(cell_up_intersect_sene35))
```

The resulting “cell_up_intersect_sene35” will have the common genes found in both senescence gene list and upregulated genes from cell model data.

Similarly, the data containing different intersected results were created using the similar codes just by changing the variables accordingly. Multiple files were created with different intersections in relation to senescent gene set.

The gene lists created by checking the intersection of cell and clinical data in which both up and down regulated gene lists are given below.

“cell_up_intersect_sene35” is the gene list produced by checking the intersection of senescent genes with upregulated genes in cell model data, it is found out that there is 500 genes in common.

“cell_down_intersect_sene35” contains 520 genes and is the intersection of senescent gens with downregulated genes of cell model data.

“clinical_up_intersect_sene35” contains 334 genes. It is the intersection of upregulated genes in clinical sample with senescent genes

“clinical_down_intersect_sene35” has 163 genes and is the intersect of downregulated genes in clinical sample and senescent genes.

After this, each senescent gene sets were checked for intersection with significantly regulated genes of clinical and cell model data and is made into a dataframe.

3 RESULTS

STANDARD_NAME	GENE_SYMBOLS	SIG. GENES IN CELL MODEL VS SENESCENCE.	SIG. GENES IN CLINICAL VS SENESCENCE	BOTH IN CELL AND CLINICAL
BIOCARTA_TEL_PATHWAY	PRKCA,,TP53,,RB1,TP53,BCL2,BCL2,IGF1R,AKT1,AKT1,HSP90AA1,TP53,TP53,TP53,TP53,TP53,TP53,TERT,XRCC6,MYC,PPP2CA,PRKCA,TERF1,TNKS,KRAS,AKT1,EGFR,HSP90AA1,TEP1,TERF1,XRCC5,KRAS,TERT,EGFR,EGFR,EGFR	PRKCA,BCL2,AKT1,MYC	PRKCA,BCL2,IGF1R,AKT1,HSP90AA1,PPP2CA,TERF1	AKT1, PRKCA , BCL2
COURTOIS_SENESCENCE_TRIGGERS	E2F1,E2F3,NF1,PTEN	E2F3	E2F1, E2F3, NF1	E2F3
DEMAGALHAES_AGING_DN	COL1A1,COL3A1,COL4A5,FABP3,GHITM,UQCRQ,,ATP5MC3,NDUFB11,ACSS2,DIABLO,CX3CL1,TFRC,UQCRFS1,CA4,CALB1,NREP	COL1A1, COL3A1, FABP3, GHITM, UQCRQ, NDUFB11, ACSS2, TFRC	FABP3, ACSS2	FABP3, ACSS2
FRIDMAN_IMMORTALIZATION_DN	OPTN,CDKN1A,CDKN2A,HTATIP2,HPS5,FILIP1L,CLTB,CCN2,CYP1B1,AHR,ALDH1A3,,HSPA2,IFI16,IGFBP3,IGFBP4,IGFBP5,IGFBP6,IGFBP7,IRF5,IRF7,NDN,SERPINE1,MAP2K3,HTRA1,RB1,SPARC,TGFB1I1,TP53,MAP1LC3B,AOPEP,TSPYL5,CREG1,CXCL14	CDKN1A, CDKN2A, FILIP1L, CLTB, AHR, ALDH1A3, HSPA2, IFI16, IGFBP3, IGFBP4, IGFBP5, IGFBP7, IRF5, SERPINE1, HTRA1, SPARC, TGFB1I1, MAP1LC3B, AOPEP, CREG1	HTATIP2, HPS5, CLTB, CCN2, CYP1B1, AHR, ALDH1A3, IGFBP3, IGFBP6, SERPINE1, SPARC, TGFB1I1, CREG1, CXCL14	CLTB, AHR, ALDH1A3, IGFBP3, SERPINE1, SPARC, TGFB1I1, CREG1

FRIDMAN_SENESCENCE_DN	COL3A1,CKS1BP7, E2F4,EGR1,ALDH1A1,LAMA1,ID1,MARCKS,BMI1,CCN4, CCNB1,LDB2,CDC25B	COL3A1, LAMA1, ID1, BMI1, CCNB1, CDC25B	EGR1,ALDH1A1,LDB2	

Table 4 : common genes expressed in samples groups in association with senescence

The detailed version of the table presented (table 4) here can be retrived from the supplimentary materials.

4 DISCUSSION

The aim of this research is to findout the senescent gene expression pattern in melanoma by compareing a cell model data and clinical data. The data used in this research for the first analysis was the RNA sequencing data of a BRAF mutated cell, which is the cell model data. This data has been used for the first differential expression analysis and the referance for this analysis was data from a melanoma cell provided except BRAF mutation. For the second analysis, clinical data from the database The Cancer Genome Atlas program(TCGA) was used. A specific mutation, BRAF V600E containing patient data has been downloaded and which is then undergone differential expression analysis. The significantly regulated genes from both clinical and cell model data has then filtered from the large data.

Analyzing the clinical and cell model data from the table (table 4) in relation to senescence genes, it could be observed that, The presence of senescent genes in both clinical and cell model data are more evident. The cell model data displayes a greater abundance of senescent genes compared to the clinical data. This means that, more senescent generes are differentialy expressed and also more significant in BRAF mutated sample. From the table itr is evident that, in both clinical and cell model datasets, there is a shared expression of common senescent genes.

The findings that senescent genes are present in both clinical data with BRAF V600E mutation and cell model data with BRAF mutation has several future implications. Some of them are, Therapeutic implications, Biomarkers for BRAF mutations, Understanding tumour biology, prognostic implications and personalized medicine.

5 REFERENCES

1. Saginala K, Barsouk A, Aluru JS, Rawla P, Barsouk A. Epidemiology of Melanoma. Med Sci (Basel). 2021 Oct 20;9(4):63. doi: 10.3390/medsci9040063. PMID: 34698235; PMCID: PMC8544364.

2. Schmitt CA, Wang B, Demaria M. Senescence and cancer - role and therapeutic opportunities. Nat Rev Clin Oncol. 2022 Oct;19(10):619-636. doi: 10.1038/s41571-022-00668-4. Epub 2022 Aug 31. PMID: 36045302; PMCID: PMC9428886.

3. Liu Y, Sheikh MS. Melanoma: Molecular Pathogenesis and Therapeutic Management. Mol Cell Pharmacol. 2014;6(3):228. PMID: 25745537; PMCID: PMC4346328

4. Paolo A. Ascierto, Iwona Lugowska, Ruth Plummer, 2021. melanoma and other skin cancers : essentials for clinicians. ESMO.

5. Davis LE, Shalin SC, Tackett AJ. Current state of melanoma diagnosis and treatment. Cancer Biol Ther. 2019;20(11):1366-1379. doi: 10.1080/15384047.2019.1640032. Epub 2019 Aug 1. PMID: 31366280; PMCID: PMC6804807.

6. Conforti C, Zalaudek I. Epidemiology and Risk Factors of Melanoma: A Review. Dermatol Pract Concept. 2021 Jul 1;11(Suppl 1):e2021161S. doi: 10.5826/dpc.11S1a161S. PMID: 34447610; PMCID: PMC8366310.

7. Motwani J, Eccles MR. Genetic and Genomic Pathways of Melanoma Development, Invasion and Metastasis. Genes (Basel). 2021 Sep 28;12(10):1543. doi: 10.3390/genes12101543. PMID: 34680938; PMCID: PMC8535311.

8. Olbryt M. Molecular background of skin melanoma development and progression: therapeutic implications. Postepy Dermatol Alergol. 2019 Apr;36(2):129-138. doi:

10.5114/ada.2019.84590. Epub 2019 May 14. PMID: 31320844; PMCID: PMC6627250.

9. Zhao M, Chen L, Qu H. CSGene: a literature-based database for cell senescence genes and its application to identify critical cell aging pathways and associated diseases. Cell Death Dis. 2016 Jan 14;7(1):e2053. doi: 10.1038/cddis.2015.414. PMID: 26775705; PMCID: PMC4816187.

10. Zeng S, Shen WH, Liu L. Senescence and Cancer. Cancer Transl Med. 2018 May-Jun;4(3):70-74. doi: 10.4103/ctm.ctm_22_18. Epub 2018 Jun 29. PMID: 30766922; PMCID: PMC6372122.

11. van der Meer D, Barthorpe S, Yang W, Lightfoot H, Hall C, Gilbert J, Francies HE, Garnett MJ. Cell Model Passports-a hub for clinical, genetic and functional datasets of preclinical cancer models. Nucleic Acids Res. 2019 Jan 8;47(D1):D923-D929. doi: 10.1093/nar/gky872. PMID: 30260411; PMCID: PMC6324059.

12. Institute of Medicine (US) Roundtable on Value & Science-Driven Health Care. Clinical Data as the Basic Staple of Health Learning: Creating and Protecting a Public Good: Workshop Summary. Washington (DC): National Academies Press (US); 2010. PMID: 21595112.

13. Michaloglou C, Vredeveld LC, Soengas MS, Denoyelle C, Kuilman T, van der Horst CM, Majoor DM, Shay JW, Mooi WJ, Peeper DS. BRAFE600-associated senescence-like cell cycle arrest of human naevi. Nature. 2005 Aug 4;436(7051):720-4. doi: 10.1038/nature03890. PMID: 16079850.

14. Cooper GM. The Cell: A Molecular Approach. 2nd edition. Sunderland (MA): Sinauer Associates; 2000. The Development and Causes of Cancer. Available from: https://www.ncbi.nlm.nih.gov/books/NBK9963/

15. Pitot HC. The molecular biology of carcinogenesis. Cancer. 1993 Aug 1;72(3 Suppl):962-70. doi: 10.1002/1097-0142(19930801)72:3+<962::aid-cncr2820721303>3.0.co;2-h. PMID: 8334671.

16. Malignant vs benign tumors: what are the differences? | verywell health. 02may2023. https://www.verywellhealth.com/what-does-malignant-and-benign-mean-514240. Accessed 02.11.2023

17. Qin D. Next-generation sequencing and its clinical application. Cancer Biol Med. 2019 Feb;16(1):4-10. doi: 10.20892/j.issn.2095-3941.2018.0055. PMID: 31119042; PMCID: PMC6528456.

18. Kukurba KR, Montgomery SB. RNA Sequencing and Analysis. Cold Spring Harb Protoc. 2015 Apr 13;2015(11):951-69. doi: 10.1101/pdb.top084970. PMID: 25870306; PMCID: PMC4863231.

19. M Kappelmann-Fenzl, Next Generation Sequencing and Data Analysis. May 4 2021.

20. Babraham Bioinformatics, https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ Accessed 12.06.2023

21. Philip Ewels, Måns Magnusson, Sverker Lundin, Max Käller, MultiQC: summarize analysis results for multiple tools and samples in a single report, *Bioinformatics*, Volume 32, Issue 19, October 2016, Pages 3047–3048, https://doi.org/10.1093/bioinformatics/btw354

22. Nuno A. Fonseca, Johan Rung, Alvis Brazma, John C. Marioni, Tools for mapping high-throughput sequencing data, *Bioinformatics*, Volume 28, Issue 24, December