|  |
| --- |
| **PROTOCOL CODING IN “R”** |

A blue and grey logo

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| **PROJECT ID** | **18 Bwanya** |
| **Title:** | **Gene Expression Analysis and Visualization of Lung Cancer Pathways Using R and Bioinformatics Pipelines** |
| **Version:** | **01** |
| **Author(s)**  **(+ initials)** | **Silke Römkens, i6325117**  **Bart Bruijnen, i6330097**  **Jarne Paar, i6318995**  **Cecilia Mantovani, i6323828**  **Sabya Sherpa, i6326426**  **Anne Fleur van Groningen, i6315029** |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision / History** | | **Review** | |
| **Version** | **Change description** | **Date** | **Initials** |
| 01 | First issue | 18/02/2025 |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1. INTRODUCTION 2](#_Toc119389311)

[2. EQUIPEMENT / MATERIAL / SOFTWARE / DATA / SAMPLES (select what is applicable) 2](#_Toc119389312)

[3. HEALTH AND SAFETY (if applicable) 2](#_Toc119389313)

[4. SPECIFIC RECOMMENDATIONS / WARNING (if applicable) 2](#_Toc119389314)

[5. PROCEDURE TO FOLLOW 2](#_Toc119389315)

[6. DATA ANALYSIS AND STATISTICS (if applicable) 3](#_Toc119389316)

[7. LITERATURE 4](#_Toc119389317)

[8. APPENDIX (if required) 4](#_Toc119389318)

|  |
| --- |
| 1. INTRODUCTION |

Lung cancer is one of the leading cause of cancer-related deaths worldwide, with two main types: small cell mung cancer (SCLC) and non-small cell lung cancer(NSCLC). NSCLC accounts for approximately 85% of all lung cancer cases and includes adenocarcinoma, squamous cell carcinoma, and large cell carcinoma (Herbst et al., 2018). Lung cancer is strongly associated with risk factors such as smoking, environmental pollutants, and genetic predisposition (Siegel et al., 2021). Despite advancements in treatment strategies, including targeted therapies and immunotherapy, lung cancer remains a major public health challenge due to late-stage diagnosis and resistance to treatment. Identifying differentially expressed genes (DEGs) can provide insight into the molecular mechanisms of lung cancer, potentially leading to novel therapeutic targets and improved diagnostic tools.

This protocol describes the steps involved in analyzing differentially expressed genes (DEGs) in lung cancer using RNA-seq data. In the project, we focus on analyzing the publicly available dataset GSE1089 to identify significant DEGs between tumor and healthy tissues. By investigating gene expression differences, we aim to gain more insight into biological pathways and genes relevant to lung cancer. Our objectives include identifying differentially expressed genes (DEGs) that show notable expression variations between tumor and normal lung tissues, investigating enriched pathways and molecular functions linked to the identified DEGs, and evaluating genetic and environmental risk factors affecting gene expression alterations in NSCLC. Moreover, we hope to assist future interventions by suggesting possible biomarkers or therapeutic targets for the treatment and management of lung cancer, while linking our discoveries with current literature to confirm results and uncover new insights.

|  |
| --- |
| 2. SOFTWARE / DATA |

Various programs and databanks were employed to work in and aid the project’s progress:

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Version / (Main) Developer** |
| *Computer* | A computer with at least 16 GB RAM available. This is recommended so running the code is a smooth and efficient endeavour. | - |
| *Github* | GitHub is an online and desktop application. The program is employed to create a global location to share and work on files collectively. It is primarily used to work on the code, but can also be used to work on other files of interest which are non-code related. | Version 3.4.16 (x64) |
| *R* | R is a programming language primarily employed for statistical computing and data visualization. To use this programming language, the “R” package first must be fetched online. | 4.2.2. |
| *Rstudio* | Rstudio is a program that uses the R programming language to develop a working code. | 4.2.2. |
| *NCBI Gene Expression Omnibus (GEO)* | The GSE81089 dataset comprises RNA sequencing (RNA-seq) data from 199 non-small cell lung cancer (NSCLC) tissue samples, along with paired normal lung tissues from 19 patients. Each sample includes 63129 gene expression readings recorded by Ensembl IDs. | National Center for Biotechnology Information |
| *Rstudio packages* | **Biocmanager**: The BiocManager package, as the modern successor package to BiocInstaller, allows users to install and manage packages from the Bioconductor project. Bioconductor focuses on the statistical analysis and comprehension of high-throughput genomic data. | 1.30.25 / Martin Morgan |
|  | **DESeq2**: Package from the Bioconductor project. Allows estimate variance-mean dependence in count data from high-throughput sequencing assays and tests for differential expression based on a model using the negative binomial distribution. | 3.20 / Michael Love |
|  | **ggplot2**: A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". ggplot2 helps with mapping variables by offering freedom in aesthetics, graphical primitives, and other details. | 3.5.1 / Hadley Wickham |
|  | **dplyr**: A fast, consistent tool for working with data frame-like objects, both in and out of memory. | 1.1.4 / Hadley Wickam |
|  | **pheatmap**: Allows implementation of heatmaps and offers more control over dimensions and appearance. | 1.0.12 / Raivo Kolde |
|  | **clusterProfiler**: Package from the Bioconductor project. This package supports functional characteristics of both coding and non-coding genomics data for thousands of species with up-to-date gene annotation. It provides a universal interface for gene functional annotation from various sources. It provides a tidy interface to access, manipulate, and visualize enrichment results to aid in efficient data interpretation. | 3.20 / Guanchuang Yu |
|  | **org.Hs.eg.db**: Package from the Bioconductor project. Helps with genome-wide annotation for Humans, primarily based on mapping using Entrez Gene identifiers. | 3.20 / Marc Carlson |
|  | **GEOquery**: Package from the Bioconductor project. The NCBI Gene Expression Omnibus (GEO) is a public repository of microarray data. GEOquery is the bridge between GEO and BioConductor and thus integrates GEO into Rstudio. | 3.20 / Sean Davis |

|  |
| --- |
| 3. HEALTH AND SAFETY |

No physical laboratory work is involved, so the health and safety risks are minimal. Since this project primarily involves extensive computer work, it is essential to maintain proper ergonomics to ensure health and safety. Sitting in an appropriate posture with a good chair and desk setup can help prevent back and neck pain. Screen brightness should be adjusted to a comfortable level to reduce eye strain, and regular breaks should be taken to prevent skeletal discomfort and fatigue. Additionally, proper positioning of your laptop is crucial to avoid strain injuries. To promote circulation and overall well-being, stretching exercises and short walks are recommended throughout the workday.   
Ensuring the secure handling of computational data is vital to prevent accidental loss or corruption.

|  |
| --- |
| 4. SPECIFIC RECOMMENDATIONS / WARNING |

We are using GitHub, so we have to exercise caution when working on our code together. If we work individually, we must push our code separately; otherwise, GitHub will not accept multiple edits at once, and some of the code will be lost.

|  |
| --- |
| 5. PROCEDURE TO FOLLOW |

These are the procedure to follow to get results

1. Download the lung cancer data from GEO database​ to perform the analysis of the data
2. Install and require libraries in R, the libraries we will be using are DESeq2, ggplot2, dplyr, pheatmap, clusterProfiler, org.HS.eg.db and GEOquery.
3. Load the count matrix, it has the data from step 1
4. Load metadata, this can be required from the R library GEOquery
5. Create a DESeq2 dataset in R to create a statistical model
6. Perform Quality control
7. Get results and sort by pValue
8. Filter significant genes with adjusted p-value < 0.05
9. Visualize results such as volcano plot, heatmap, MA plot
10. Functional enrichment analysis using GO, KEGG

|  |
| --- |
| 6. DATA ANALYSIS AND STATISTICS |

For the analysis of the normalized gene expression data (GSE81089\_FPKM\_cufflinks.tsv), we used R and the DESeq2 package. DESeq2 applies a statistical model to assess differences in gene expression between two or more sample groups. In our case, it assessed the difference between the people with NSCLC and normal lung tissue. To identify differentially expressed genes (DEGs), DESeq2 calculated a p-value for each gene. We used a threshold of p< 0.05 to determine statistically significant DEGs. To visualize the results, we used R to generate volcano plots to highlight significant DEGs.

|  |
| --- |
| 7. LITERATURE |

1. Herbst, R. S., Morgensztern, D., & Boshoff, C. (2018). The biology and management of non-small cell lung cancer. *Nature*, 553(7689), 446-454.
2. Siegel, R. L., Miller, K. D., & Jemal, A. (2021). Cancer statistics, 2021. *CA: A Cancer Journal for Clinicians*, 71(1), 7-33.

|  |
| --- |
| 8. APPENDIX |

Project.Main.R  
  
# Welcome to our project. For the code to work, a set of packages have to be

# installed. Follow the following commands to acquire the proper packages.

# Check if BiocManager is installed; install it if not

if (!requireNamespace("BiocManager")) {

install.packages("BiocManager")

} else {

message("BiocManager is already installed")

}

# Create vector containing all the packages

packages <- c("DESeq2", "ggplot2", "dplyr", "pheatmap",

"clusterProfiler", "org.Hs.eg.db", "GEOquery")

# Check if required packages is installed; install it if not

if (!requireNamespace(packages)) {

BiocManager::install(packages)

} else {

message("Packages are already installed")

}

# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

# Step 1. Preparatory Analysis

# Source files and data

source("Project.Functions.R")

require(DESeq2)

require(ggplot2)

require(dplyr)

require(pheatmap)

require(clusterProfiler)

require(org.Hs.eg.db)

require(GEOquery)

#load raw counts

counts <- read.delim("FPKM\_cufflinks.tsv", header=TRUE,

row.names=1, sep="\t", check.names=FALSE)

# Looking at the head counts to see the type of data inside

head(counts)

#load metadata using the getGEO function

gse <- getGEO(GEO = 'GSE81089', GSEMatrix = TRUE)

#Extract metadata using pData function

metadata <- pData(phenoData(gse[[1]]))

# Look at the data inside. Head gives you the first 6

head(metadata)

# I did colnames to see the different colomns

colnames(metadata)

# Create subset

metadata.subset <- metadata[, c(1, 8, 48, 49, 50, 51, 52, 53, 54, 56)]

# Look at the different names

colnames(counts)

# Renaming the colnames to the appropriate names to make it more readable

metadata.subset <- setNames(metadata.subset, c(

"Title", "Source", "Age", "Life\_Status", "Sex", "Histology", "Performance",

"Smoking\_Status", "Tumor\_stage", "Sample"

)[match(names(metadata.subset), c(

"title", "source\_name\_ch1", "age:ch1", "dead:ch1", "gender:ch1", "histology:ch1", "ps who:ch1",

"smoking:ch1", "stage tnm:ch1", "tumor (t) or normal (n):ch1"

))])

# Set column 'Sample' in metadata.subset as row names in metadata.subset (\*to be able to match it later for deseq2 to column names of counts)

rownames(metadata.subset) <- metadata.subset$Sample

# Extract the deaths

dead <- metadata.subset[, 3]

paste(dead)

# Remove the last row from Data

head(counts)

dim(counts)

counts <- counts[-nrow(counts), ]

# Check if the last row is removed

dim(counts) # Check new dimensions

# Ensure the output directory exists

if (!dir.exists("Output")) {

dir.create("Output")

}

# Select our the genes of interest

interest.genes <- c("ENSG00000157764", "ENSG00000133703")

# Subset our genes of interest into new df by filtering on columns

express <- counts[rownames(counts) %in% interest.genes, , drop = FALSE]

# Convert the data into data frame

express <- as.data.frame(express)

# Reshape the expression data to better fit the dataframe

express$Gene <- rownames(express)

express<- reshape2::melt(express, id.vars = "Gene", variable.name = "Sample",

value.name = "Expression")

# Merge with metadata

expression <- merge(express, metadata.subset, by = "Sample", all.x = TRUE)

# Plot expression levels of selected genes

save.pdf(function(){

ggplot(expression, aes(x = Sample, y = Expression, fill = Gene)) +

geom\_col(position = "dodge") +

facet\_wrap(~ Gene, scales = "free\_y") +

coord\_flip() + # Flip x and y axes

theme\_minimal() +

labs(title = "Gene Expression Levels Across Samples",

x = "Expression Level",

y = "Sample") # Swap x and y labels accordingly

}, "Sample Gene Expression Levels")

# Plot expression levels of selected genes

save.pdf(function(){

ggplot(expression, aes(x = Sample, y = Expression, fill = Gene)) +

geom\_bar(stat = "identity", position = "dodge") +

facet\_wrap(~ Gene, scales = "free\_y") + # Separate plots per gene

theme\_minimal() +

labs(title = "Gene Expression Levels Across Samples",

x = "Sample",

y = "Expression Level") +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))# Rotate sample labels

}, "Sample Gene Expression Levels")

# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

# Step 2. Differential Gene Expression Analysis

# Making sure the row names in metadata.subset matches to column names in counts

all(colnames(counts) %in% rownames(metadata.subset))

# Find Columns in counts That Are Not in metadata.subset

setdiff(colnames(counts), rownames(metadata.subset))

# Remove everything after the underscore in counts

colnames(counts) <- sub("\_.\*", "", colnames(counts))

colnames(counts) #check if it is removed

# Check again if row names in metadata.subset matches to column names in counts

all(colnames(counts) %in% rownames(metadata.subset))

# Check if they are in the same order

all(colnames(counts) == rownames(metadata.subset))

# Reorder metadata.subset rows to match the column order in counts

metadata.subset <- metadata.subset[match(colnames(counts), rownames(metadata.subset)), , drop = FALSE]

# Check if they now match

all(colnames(counts) == rownames(metadata.subset))

# Check the values in the counts

summary(counts)

# Convert all data values to Absolute values. (Non-negative)

Data <- abs(Data)

# Round values to integers

Data <- round(Data)

# Construct a DESeqDataSet object

dds <- DESeqDataSetFromMatrix(countData = Data,

colData = metadata,

design = ~ Source)

dds

# Quality control

# Remove genes with low counts (choose one)

keep <- rowSums(counts(dds)) >= 10

dds <- dds[keep,]

keep2 <- rowMeans(counts(dds)) >=10

dds <- dds[keep2,]

dds

# Set the factor level

Project.Functions.R  
  
# Function to save a plot to PDF.

# plot\_function must be a defined plotting function &

# filename will be the designated file name for the pdf

save.pdf <- function(plot\_function, filename) {

# Define the output PDF file path

pdf\_file <- file.path("Output", paste0(filename, ".pdf"))

# Open the PDF device to save the plot

pdf(pdf\_file)

# Call the plot function to create the plot

plot\_function()

# Close the PDF device (this finalizes and writes the plot to the file)

dev.off()

# Inform the user that the plot has been saved

message("Plot has been saved to: ", pdf\_file)

}