Gene Expression Analysis in Cancer using RNA-Seq

Pranava Upparlapalli

2025-06-06

Table of Contents

## 🔬 Project Overview

This project explores gene expression variation across five cancer types (BRCA, KIRC, LUAD, COAD, PRAD) using RNA-Seq count data. Differential expression was assessed using DESeq2, followed by PCA, clustering, and visualization to identify cancer-specific gene signatures.

### STEP - 1: Loading Libraries

library(tidyverse)  
library(tidyr)  
library(DESeq2)  
library(ggplot2)  
library(VennDiagram)  
library(UpSetR)  
library(RColorBrewer)  
library(pheatmap)  
library(reshape2)  
library(pROC)  
library(caret)  
library(parallel)  
library(mltools)  
library(MASS)  
library(ggfortify)  
library(stats)  
library(data.table)

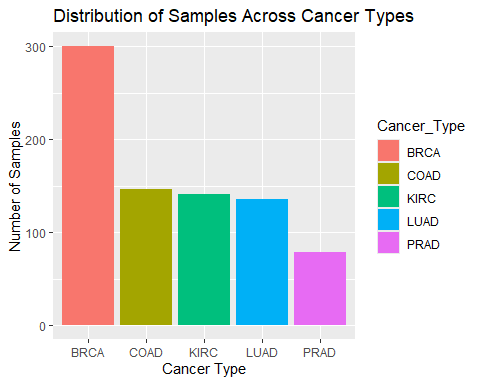
### STEP - 2: Loading Data + Normalization:

# Set the paths for data and labels  
data\_path <- "data/data.csv"  
labels\_path <- "data/labels.csv"  
  
data <- read.csv(data\_path, header = TRUE)  
labels <- read.csv(labels\_path, header = TRUE)  
  
merged\_data <- merge(data, labels, by = "X", all.x = TRUE)  
merged\_data <- na.omit(merged\_data)  
merged\_data$Class <- as.factor(merged\_data$Class)  
gc()

used (Mb) gc trigger (Mb) max used (Mb)  
 Ncells 6868332 366.9 16011544 855.2 11791454 629.8  
 Vcells 45934884 350.5 75869196 578.9 75868631 578.9

### STEP -3: Data division visualisation:

# Count samples for each cancer type  
class\_data <- as.data.frame(table(merged\_data$Class))  
colnames(class\_data) <- c("Cancer\_Type", "Count")  
class\_data$Count <- sort(class\_data$Count, decreasing = TRUE)  
  
# Create bar plot  
data\_division <- ggplot(class\_data, aes(x = Cancer\_Type, y = Count, fill = Cancer\_Type)) +  
 geom\_bar(stat = "identity") +  
 theme\_gray() +  
 labs(title = "Distribution of Samples Across Cancer Types",  
 x = "Cancer Type",  
 y = "Number of Samples")  
print(data\_division)



ggsave("plots/sample\_distribution.png", plot = data\_division, width = 8, height = 5)  
gc()

used (Mb) gc trigger (Mb) max used (Mb)  
 Ncells 7044196 376.3 16011544 855.2 11791454 629.8  
 Vcells 46278554 353.1 75869196 578.9 75868631 578.9

### STEP - 4: DESeq2 Analysis

# 1. Prepare data  
# This intermediate variable holds the numeric data with samples as rows  
gene\_data <- merged\_data[, !names(merged\_data) %in% c("X", "Class")]  
rownames(gene\_data) <- merged\_data$X  
  
# Transpose to get the final matrix: genes as rows, samples as columns  
gene\_expr\_matrix <- t(gene\_data)  
gene\_expr\_matrix <- round(gene\_expr\_matrix) # Ensure integer counts  
  
# Prepare sample information  
sample\_info <- data.frame(Class = merged\_data$Class)  
rownames(sample\_info) <- merged\_data$X  
  
# 2. Create DESeq2 object  
# Check for matching sample names  
stopifnot(all(colnames(gene\_expr\_matrix) == rownames(sample\_info)))  
dds <- DESeqDataSetFromMatrix(countData = gene\_expr\_matrix,  
 colData = sample\_info,  
 design = ~ Class)

converting counts to integer mode

# 3. Run analysis  
dds <- DESeq(dds)

estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

-- note: fitType='parametric', but the dispersion trend was not well captured by the  
 function: y = a/x + b, and a local regression fit was automatically substituted.  
 specify fitType='local' or 'mean' to avoid this message next time.

final dispersion estimates

fitting model and testing

res <- results(dds) # Get full results for plotting  
  
# 4. Get significant genes  
res\_sig <- subset(res, padj < 0.05 & !is.na(padj)) # Filter for significance  
print(summary(res\_sig))

out of 7639 with nonzero total read count  
 adjusted p-value < 0.1  
 LFC > 0 (up) : 3920, 51%  
 LFC < 0 (down) : 3719, 49%  
 outliers [1] : 0, 0%  
 low counts [2] : 0, 0%  
 (mean count < 0)  
 [1] see 'cooksCutoff' argument of ?results  
 [2] see 'independentFiltering' argument of ?results  
   
 NULL

sig\_gene\_names <- rownames(res\_sig)  
  
# Extract normalized counts for visualization  
sig\_genes\_data <- as.data.frame(counts(dds, normalized = TRUE)[sig\_gene\_names, ])  
sig\_genes\_data$Gene <- rownames(sig\_genes\_data)

# --- MORE ROBUST Cleaning, Clustering, and Reshaping ---  
  
# 1. Create a clean numeric matrix from sig\_genes\_data  
gene\_names <- sig\_genes\_data$Gene  
numeric\_data <- sig\_genes\_data[, -which(names(sig\_genes\_data) == "Gene")]  
rownames(numeric\_data) <- gene\_names  
  
# 2. Find which genes (rows) are valid for analysis  
is\_row\_valid <- apply(numeric\_data, 1, function(row) {  
 all(is.finite(row)) && var(row, na.rm = TRUE) > 0  
})  
  
# 3. Create the final, clean data matrix  
gene\_matrix\_for\_clustering <- numeric\_data[is\_row\_valid, , drop = FALSE] # drop=FALSE is safer  
  
# 4. === ADDED SAFETY CHECK ===  
# Check if we have enough genes to perform clustering  
if (nrow(gene\_matrix\_for\_clustering) >= 2) {  
  
 # Perform clustering on the GENES  
 gene\_hclust <- hclust(dist(gene\_matrix\_for\_clustering))  
 ordered\_gene\_names <- rownames(gene\_matrix\_for\_clustering)[gene\_hclust$order]  
  
 # Filter the original data frame to keep only the clean genes  
 sig\_genes\_data\_clean <- sig\_genes\_data[is\_row\_valid, ]  
  
 # Reshape the CLEAN data into a long format for plotting  
 sig\_genes\_data\_long <- pivot\_longer(  
 sig\_genes\_data\_clean,  
 cols = -Gene,  
 names\_to = "Sample",  
 values\_to = "Expression"  
 )  
  
 # Set the Gene factor levels based on the clustering order  
 sig\_genes\_data\_long$Gene <- factor(sig\_genes\_data\_long$Gene,  
 levels = ordered\_gene\_names)  
   
 print(paste("Successfully clustered", length(ordered\_gene\_names), "genes."))  
  
} else {  
   
 print("Skipping clustering: Fewer than 2 genes remained after filtering.")  
 # Create an empty data frame so the script doesn't fail later  
 sig\_genes\_data\_long <- data.frame()   
   
}

## [1] "Successfully clustered 7639 genes."

gc()

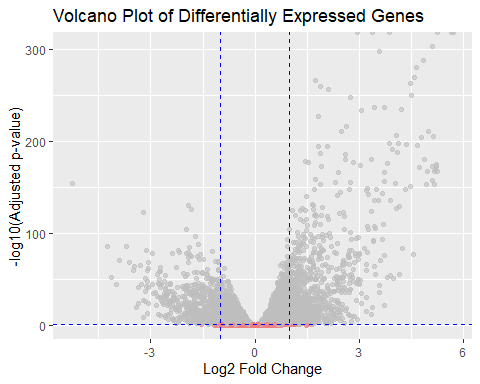
## used (Mb) gc trigger (Mb) max used (Mb)  
## Ncells 7144506 381.6 16011544 855.2 16011544 855.2  
## Vcells 163354885 1246.3 397073465 3029.5 397073446 3029.5

### STEP - 5: DDSEQ2 - Visualisation:

# Create a volcano plot  
volcano\_plot <- ggplot(res, aes(x = log2FoldChange, y = -log10(padj))) +  
 geom\_point(aes(color = ifelse(padj < 0.05, "Significant", "Not Significant")), alpha = 0.6) +  
 scale\_color\_manual(values = c("lightcoral", "gray"), guide = FALSE) + # Remove color legend  
 geom\_vline(xintercept = c(-1, 1), linetype = "dashed", color = "blue") +  
 geom\_hline(yintercept = -log10(0.05), linetype = "dashed", color = "blue") + # Added significance line  
 labs(title = "Volcano Plot of Differentially Expressed Genes",  
 x = "Log2 Fold Change",  
 y = "-log10(Adjusted p-value)") +  
 theme\_gray()  
  
print(volcano\_plot)

Warning: The `guide` argument in `scale\_\*()` cannot be `FALSE`. This was deprecated in  
 ggplot2 3.3.4.  
 ℹ Please use "none" instead.  
 This warning is displayed once every 8 hours.  
 Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
 generated.

Warning: Removed 1461 rows containing missing values or values outside the scale range  
 (`geom\_point()`).

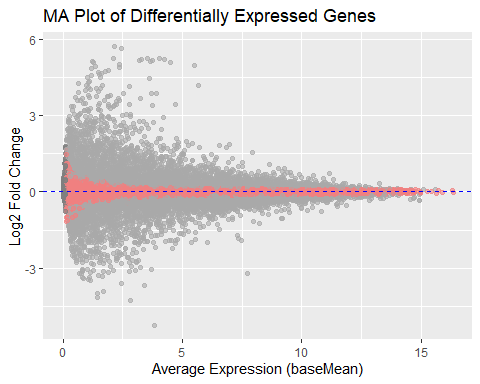


ggsave("plots/volcano\_plot.png", plot = volcano\_plot, width = 8, height = 6)

Warning: Removed 1461 rows containing missing values or values outside the scale range  
 (`geom\_point()`).

# Create an MA plot  
ma\_plot <- ggplot(res, aes(x = baseMean, y = log2FoldChange)) +  
 geom\_point(aes(color = ifelse(padj < 0.05, "Significant", "Not Significant")), alpha = 0.6) +  
 scale\_color\_manual(values = c("lightcoral", "darkgray"), guide = FALSE) + # Remove color legend  
 geom\_hline(yintercept = 0, linetype = "dashed", color = "blue") +  
 labs(title = "MA Plot of Differentially Expressed Genes",  
 x = "Average Expression (baseMean)",  
 y = "Log2 Fold Change") +  
 theme\_gray()  
  
print(ma\_plot)

Warning: Removed 284 rows containing missing values or values outside the scale range  
 (`geom\_point()`).



ggsave("plots/ma\_plot.png", plot = ma\_plot, width = 8, height = 6)

Warning: Removed 284 rows containing missing values or values outside the scale range  
 (`geom\_point()`).

### STEP - 6: PCA

# Load libraries if they aren't already  
library(matrixStats)  
library(cluster)

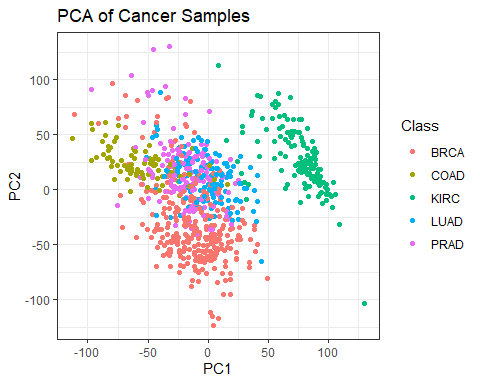
Warning: package 'cluster' was built under R version 4.3.3

library(factoextra)

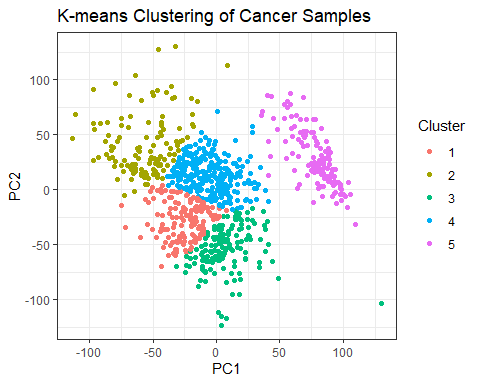
Warning: package 'factoextra' was built under R version 4.3.3

Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

# --- PCA Step ---  
# Use variance-stabilized data for best results  
vst\_data <- varianceStabilizingTransformation(dds, blind = FALSE)  
vst\_assay <- assay(vst\_data)  
  
# Filter out genes with zero variance to prevent errors  
vst\_assay\_filtered <- vst\_assay[rowVars(vst\_assay) > 0, ]  
  
# Run PCA on the samples  
pca\_results <- prcomp(t(vst\_assay\_filtered), center = TRUE, scale. = TRUE)  
  
# Extract the PC scores for plotting  
pca\_scores <- as.data.frame(pca\_results$x)  
pca\_scores$Class <- colData(dds)$Class  
  
# Print PCA plot (colored by true cancer type)  
pca\_plot <- ggplot(pca\_scores, aes(x = PC1, y = PC2, color = Class)) +  
 geom\_point(size = 1.5) +  
 labs(title = "PCA of Cancer Samples", x = "PC1", y = "PC2") +  
 theme\_bw()  
print(pca\_plot)

 ### Step - 7: K- clustering

# --- K-means Clustering Step ---  
# Perform K-means on the correct PCA scores  
kmeans\_result <- kmeans(pca\_scores[, c("PC1", "PC2")], centers = 5, nstart = 25)  
pca\_scores$K\_Cluster <- factor(kmeans\_result$cluster)  
  
# Print K-means plot (colored by algorithm-found clusters)  
cluster\_plot <- ggplot(pca\_scores, aes(x = PC1, y = PC2, color = K\_Cluster)) +  
 geom\_point(size = 1.5) +  
 labs(title = "K-means Clustering of Cancer Samples", x = "PC1", y = "PC2", color = "Cluster") +  
 theme\_bw()  
print(cluster\_plot)



# Assess cluster validity  
silhouette\_analysis <- silhouette(kmeans\_result$cluster, dist(pca\_scores[, c("PC1", "PC2")]))  
print(summary(silhouette\_analysis))

## Silhouette of 801 units in 5 clusters from silhouette.default(x = kmeans\_result$cluster, dist = dist(pca\_scores[, c("PC1", "PC2")])) :  
## Cluster sizes and average silhouette widths:  
## 142 120 149 253 137   
## 0.3192533 0.3221859 0.3396739 0.3692233 0.6006290   
## Individual silhouette widths:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.09723 0.25979 0.40966 0.38740 0.51866 0.72605

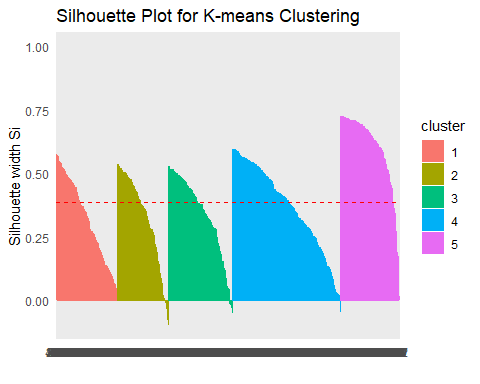
# Check the new cluster centroids  
print(kmeans\_result$centers)

## PC1 PC2  
## 1 -25.613343 -27.64696  
## 2 -56.601718 41.47128  
## 3 7.334241 -54.63707  
## 4 -5.426191 13.29876  
## 5 78.170256 27.19454

# Create and plot the silhouette using factoextra  
library(factoextra)  
sil\_plot <- fviz\_silhouette(silhouette\_analysis) +  
 theme\_minimal() +  
 labs(title = "Silhouette Plot for K-means Clustering")

## cluster size ave.sil.width  
## 1 1 142 0.32  
## 2 2 120 0.32  
## 3 3 149 0.34  
## 4 4 253 0.37  
## 5 5 137 0.60

print(sil\_plot)



ggsave("plots/silhouette\_plot.png", plot = sil\_plot, width = 8, height = 6)

### Corrected Gene Expression Heatmap ###  
  
# Ensure these libraries are loaded  
library(pheatmap)  
library(matrixStats)  
  
# 1. Use variance-stabilized data from the previous steps  
vst\_data <- varianceStabilizingTransformation(dds, blind = FALSE)  
  
# 2. Find the top 50 most variable genes to visualize  
top\_var\_genes <- head(order(rowVars(assay(vst\_data)), decreasing = TRUE), 50)  
heatmap\_matrix <- assay(vst\_data)[top\_var\_genes, ]  
  
# 3. Create an annotation bar for the sample cancer types  
annotation\_col <- data.frame(  
 CancerType = colData(dds)$Class  
)  
rownames(annotation\_col) <- colnames(heatmap\_matrix)  
  
# 4. Generate the heatmap and save it to a file  
# 'scale = "row"' is crucial for visualizing relative gene expression  
pheatmap(heatmap\_matrix,  
 scale = "row",  
 annotation\_col = annotation\_col,  
 show\_rownames = FALSE,  
 show\_colnames = FALSE,  
 main = "Heatmap of Top 50 Variable Genes",  
 filename = "plots/heatmap\_expression.png",  
 width = 8,  
 height = 6)

## ✅ Conclusion

This project demonstrates the ability of RNA-Seq analysis to uncover meaningful differences in gene expression across multiple cancer types. Techniques like DESeq2, PCA, and unsupervised clustering provided statistically and biologically relevant insights into transcriptomic profiles.

## 🔁 Session Info

sessionInfo()

## R version 4.3.2 (2023-10-31 ucrt)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 11 x64 (build 22631)  
##   
## Matrix products: default  
##   
##   
## locale:  
## [1] LC\_COLLATE=English\_India.utf8 LC\_CTYPE=English\_India.utf8   
## [3] LC\_MONETARY=English\_India.utf8 LC\_NUMERIC=C   
## [5] LC\_TIME=English\_India.utf8   
##   
## time zone: America/Chicago  
## tzcode source: internal  
##   
## attached base packages:  
## [1] parallel grid stats4 stats graphics grDevices utils   
## [8] datasets methods base   
##   
## other attached packages:  
## [1] factoextra\_1.0.7 cluster\_2.1.8   
## [3] data.table\_1.16.4 ggfortify\_0.4.17   
## [5] MASS\_7.3-60.0.1 mltools\_0.3.5   
## [7] caret\_7.0-1 lattice\_0.22-6   
## [9] pROC\_1.18.5 reshape2\_1.4.4   
## [11] pheatmap\_1.0.12 RColorBrewer\_1.1-3   
## [13] UpSetR\_1.4.0 VennDiagram\_1.7.3   
## [15] futile.logger\_1.4.3 DESeq2\_1.42.1   
## [17] SummarizedExperiment\_1.32.0 Biobase\_2.62.0   
## [19] MatrixGenerics\_1.14.0 matrixStats\_1.4.1   
## [21] GenomicRanges\_1.54.1 GenomeInfoDb\_1.38.8   
## [23] IRanges\_2.36.0 S4Vectors\_0.40.2   
## [25] BiocGenerics\_0.48.1 lubridate\_1.9.4   
## [27] forcats\_1.0.0 stringr\_1.5.1   
## [29] dplyr\_1.1.4 purrr\_1.0.2   
## [31] readr\_2.1.5 tidyr\_1.3.1   
## [33] tibble\_3.2.1 ggplot2\_3.5.1   
## [35] tidyverse\_2.0.0   
##   
## loaded via a namespace (and not attached):  
## [1] bitops\_1.0-9 gridExtra\_2.3 formatR\_1.14   
## [4] rlang\_1.1.4 magrittr\_2.0.3 compiler\_4.3.2   
## [7] systemfonts\_1.1.0 vctrs\_0.6.5 pkgconfig\_2.0.3   
## [10] crayon\_1.5.3 fastmap\_1.2.0 backports\_1.5.0   
## [13] XVector\_0.42.0 labeling\_0.4.3 utf8\_1.2.4   
## [16] rmarkdown\_2.29 prodlim\_2024.06.25 tzdb\_0.4.0   
## [19] ragg\_1.3.3 xfun\_0.49 zlibbioc\_1.48.2   
## [22] recipes\_1.1.0 DelayedArray\_0.28.0 BiocParallel\_1.36.0   
## [25] broom\_1.0.7 R6\_2.5.1 stringi\_1.8.4   
## [28] car\_3.1-3 parallelly\_1.40.1 rpart\_4.1.23   
## [31] Rcpp\_1.0.13-1 iterators\_1.0.14 knitr\_1.49   
## [34] future.apply\_1.11.3 Matrix\_1.6-5 splines\_4.3.2   
## [37] nnet\_7.3-19 timechange\_0.3.0 tidyselect\_1.2.1   
## [40] rstudioapi\_0.17.1 abind\_1.4-8 yaml\_2.3.10   
## [43] timeDate\_4041.110 codetools\_0.2-19 listenv\_0.9.1   
## [46] plyr\_1.8.9 withr\_3.0.2 evaluate\_1.0.1   
## [49] future\_1.34.0 lambda.r\_1.2.4 survival\_3.7-0   
## [52] ggpubr\_0.6.0 pillar\_1.9.0 carData\_3.0-5   
## [55] foreach\_1.5.2 generics\_0.1.3 RCurl\_1.98-1.16   
## [58] hms\_1.1.3 munsell\_0.5.1 scales\_1.3.0   
## [61] globals\_0.16.3 class\_7.3-22 glue\_1.8.0   
## [64] tools\_4.3.2 ggsignif\_0.6.4 ModelMetrics\_1.2.2.2   
## [67] gower\_1.0.1 locfit\_1.5-9.10 ipred\_0.9-15   
## [70] colorspace\_2.1-1 nlme\_3.1-166 GenomeInfoDbData\_1.2.11  
## [73] Formula\_1.2-5 cli\_3.6.3 textshaping\_0.4.1   
## [76] futile.options\_1.0.1 fansi\_1.0.6 S4Arrays\_1.2.1   
## [79] lava\_1.8.0 gtable\_0.3.6 rstatix\_0.7.2   
## [82] digest\_0.6.37 ggrepel\_0.9.6 SparseArray\_1.2.4   
## [85] farver\_2.1.2 htmltools\_0.5.8.1 lifecycle\_1.0.4   
## [88] hardhat\_1.4.0