Deseq2_GSE29001

```r
# Convert matrix to a dataframe
df_GSE29001 <- as.data.frame(GSE29001)



# Extract only the count matrix
count_matrix <- df_GSE29001[, 2:ncol(df_GSE29001)]

# Sample information
sample_names <- colnames(count_matrix)
conditions = c("normal", "normal", "tumor", "normal", "normal", "tumor", "tumor", "normal", "normal", "
                            "normal", "normal", "tumor", "tumor", "normal", "normal", "tumor", "tumor",
                            "normal", "normal", "tumor", "tumor", "normal", "normal", "tumor", "tumor",
                            "tumor", "normal", "normal", "tumor", "tumor", "normal", "normal", "tumor",

sample_info <- data.frame(Sample = sample_names, condition = conditions)

#### SKIRMISH FOR DESEQ2 DATASET ###########
count_matrix <- as.matrix(sapply(count_matrix, as.numeric))
count_matrix <- count_matrix[complete.cases(count_matrix), ]
count_matrix <- round(count_matrix)

#get rownames of previous data
row_names_GSE29001 <- rownames(GSE29001)

# Set gene_id to rownames again
rownames(count_matrix) <- row_names_GSE29001

# Create the DESeqDataSet using the updated sample_info
dds <- DESeqDataSetFromMatrix(countData = count_matrix,
                              colData = sample_info,
                              design = ~ condition)
```

```
## converting counts to integer mode

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

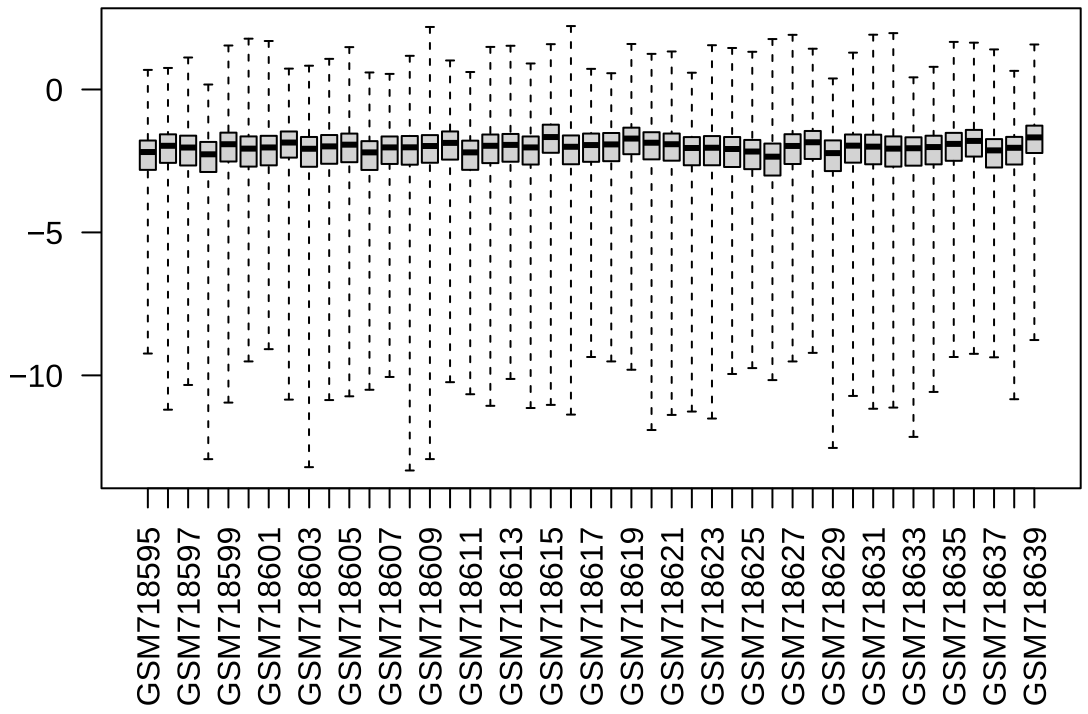
```r
# Perform differential gene expression analysis
dds <- DESeq(dds)
```

```
## estimating size factors

## estimating dispersions
## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 278 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

```
resultsNames(dds)
```

```
## [1] "Intercept"                 "condition_tumor_vs_normal"
```

```
par(mar=c(8,5,2,2))
boxplot(log10(assays(dds)[["cooks"]]), range=0, las=2)
```



```
res <- results(dds, name="condition_tumor_vs_normal")
summary(res)
```

```
##
## out of 22277 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 5321, 24%
## LFC < 0 (down)     : 4591, 21%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Box-and-Whisker Plot:depicts the distribution of gene expression values across distinct sample groups (here, normal and cancer).The boxes show the interquartile range (IQR), while the centre line inside the box reflects the median expression value. The whiskers extend from the margins of the boxes and represent the variability of the data. Outliers are points that are not within the whiskers. The figure allows us to examine the expression distributions of the two groups and discover any variations in their central tendencies and spread

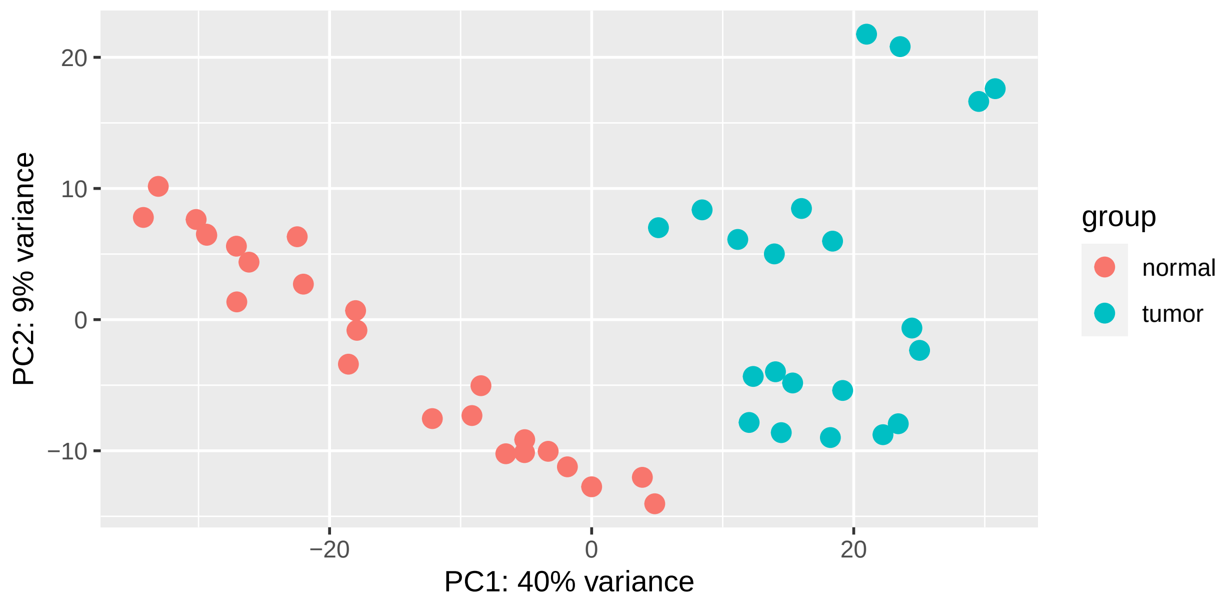
```
# Adjustment to overcome NA-error --> exclude rows containing NA's
filtered_indices <- which(res$log2FoldChange > 1 & res$padj < 0.05 & !is.na(res$padj))
filtered_downregulated_genes <- res[filtered_indices, ]
gene_names <- rownames(filtered_downregulated_genes)
head(gene_names,10)
```

```
##  [1] "117_at"      "1405_i_at"   "200050_at"   "200052_s_at" "200593_s_at"
##  [6] "200600_at"   "200616_s_at" "200628_s_at" "200629_at"   "200644_at"
```

```
rld <- rlog(dds, blind=TRUE)
```

```
## rlog() may take a few minutes with 30 or more samples,
## vst() is a much faster transformation
```

```
PCA1 <- plotPCA(rld, intgroup = "condition")
PCA1
```



Each point on the PCA plot represents a sample, and the color of the points indicates which experimental groups the samples(normal and cancer) belong to.

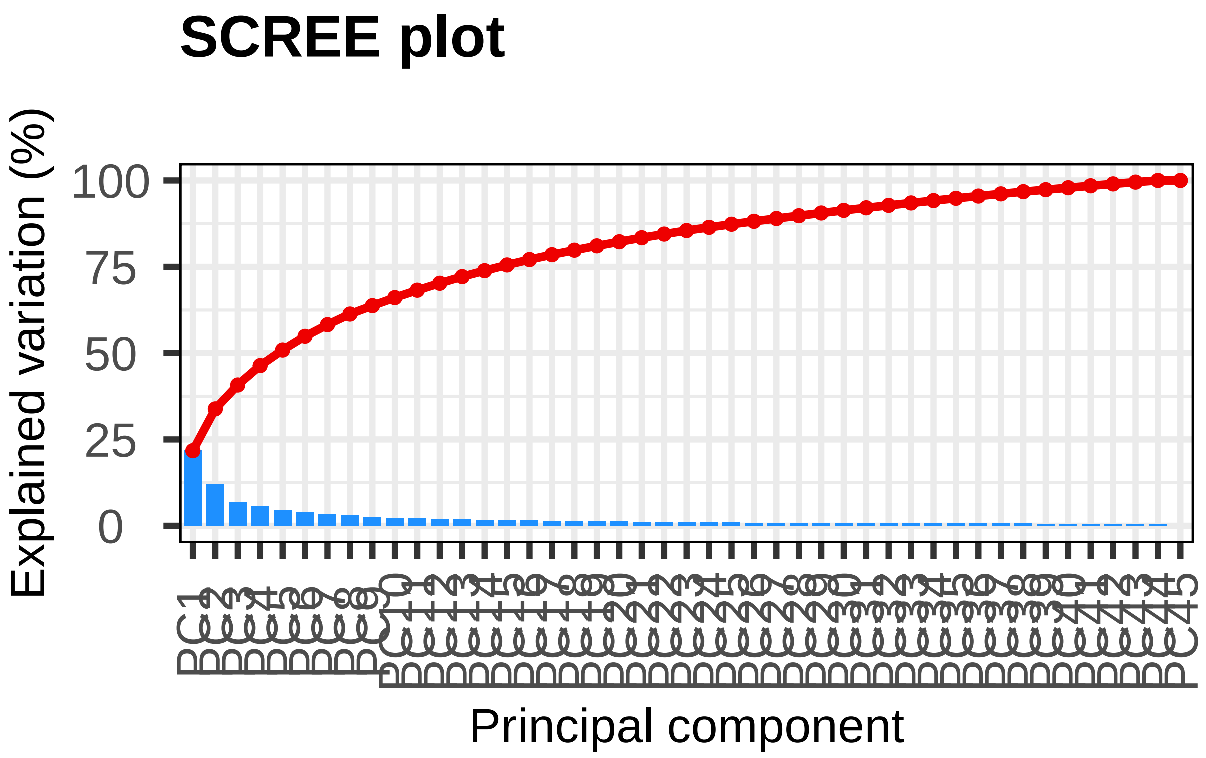
```
library(DESeq2)
library(PCAtools)
```

```
## Loading required package: ggplot2

## Loading required package: ggrepel

##
## Attaching package: 'PCAtools'

## The following objects are masked from 'package:stats':
##
##     biplot, screeplot
```

```
vst <- assay(vst(dds))
p <- pca(vst, removeVar = 0.1)
```

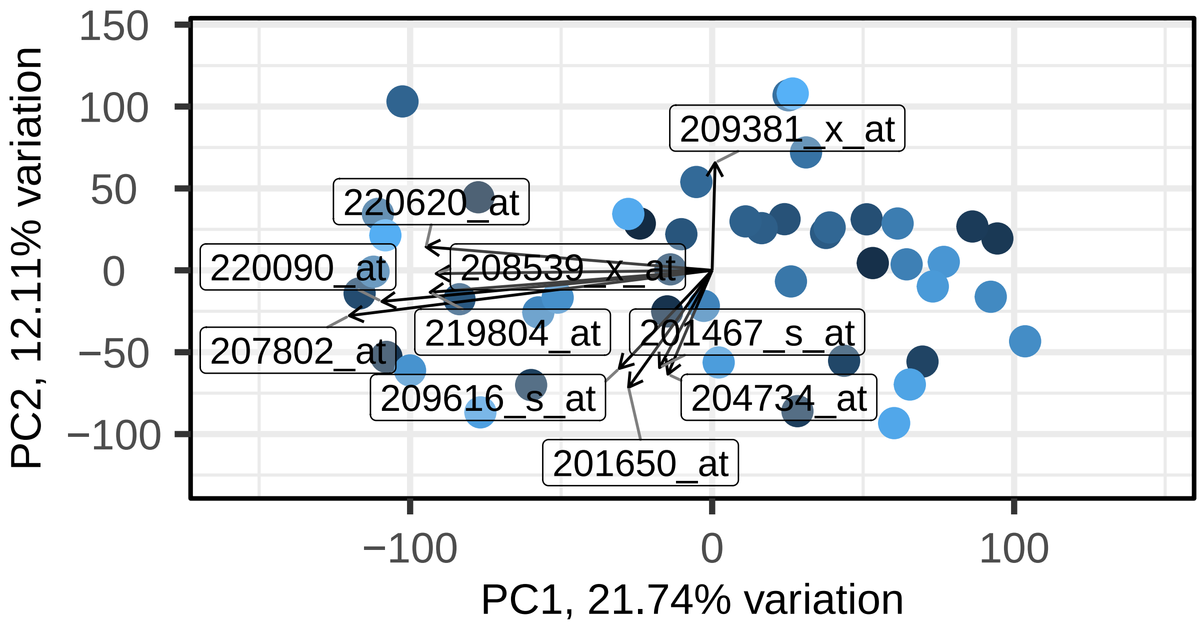
```
## Warning: useNames = NA is deprecated. Instead, specify either useNames = TRUE
## or useNames = TRUE.

## -- removing the lower 10% of variables based on variance
```

```
screeplot(p, axisLabSize = 18, titleLabSize = 22)
```



The scree plot depicts the proportion of variation explained by each principle component, allowing us to assess the importance of each PC in capturing the overall variability in the dataset. This figure is useful for evaluating the dimensionality of the data and estimating how many principal components are required to represent a substantial percentage of the dataset's variability.

```
biplot(p, showLoadings = TRUE,
       labSize = 5, pointSize = 5, sizeLoadingsNames = 5)
```



A biplot is a type of scatter plot that combines the sample scores (positions of the samples in the PCA space) and the variable loadings (contributions of the variables to each principal component) on the same plot.The distance between the sample points on the biplot illustrates how similar or distinct the gene expression patterns of the samples are.