EDA_GSE5847

Raman Butta

2025-09-11

Contents

Introduction to the GSE5847 Dataset **1**

Samples & Experimental Design 2

Platform & Technical Details 2

Installations & Data pre-processing **2**

EDA through Principal Component Analysis (PCA) **5**

Differential Gene Expression Analysis (a hypothesis testing) **8**

Our objective : 8

Explanation 9

Key Insights 10

Volcano Plot **11**

Explanation of the Volcano Plot Code 12

MA Plot **13**

Conclusion **16**

Introduction to the GSE5847 Dataset

- This project analyses the **GSE5847_series_matrix.txt.gz** which is a series matrix file downloaded from the GEO (Gene Expression Omnibus). GSE5847 is the GEO Accession no.
- This dataset examines **gene expression in tumor epithelium and surrounding stromal cells** of human breast cancer, captured via **laser capture microdissection (LCM)**.
- **Objective**: To contrast gene expression patterns between epithelial (tumor) and stromal (microenvironment) compartments.

Samples & Experimental Design

- **Total Samples**: 95 microarray samples.
- **Patient Representation**: Derived from **50 unique patients**, including **15 IBC** and **35 non-IBC** cases.
- **Sample Type Breakdown**: Almost each patient contributed **matched epithelial (tumor) and stromal samples**, meaning a pair from each individual.

Platform & Technical Details

- **Array Platform**: Affymetrix Human Genome U133A Array (**GPL96**).
- **Probe Coverage**: Approximately **22,000 probe sets** per sample.

  N.B. GPL96 is the GEO code for the HG-U133A chip which is one of the first microarray chips developed by Affymetrix in the early 2000s. The microarray doesn't measure gene expression directly. Instead it has short DNA oligonucleotides ("probes") printed on it. Each probe is designed to hybridize an mRNA sequence from a specific gene. Each mRNA strand is marked with a flouroscent dye, so the amount of it sticking in the probes gives an indication of which gene is expressed more. The Hybridization intensity → log expression value is what you'll see in the dataset.

  Each GPL96 chip has 22,283 probe sets printed on it. So each sample produces a vector of ~22k expression values. With 95 samples and a row for Probe ID, the full expression matrix is about 22,283 rows × 96 columns.

  These 22,283 probe sets represented ~13,000 well characterised genes of the human genome in the early 2000s. That's because some genes like TP53, GAPDH, etc. have multiple transcript variants due to alternative splicing and therefore separate probes.

Installations & Data pre-processing

This series matrix file is like a giant spreadsheet which contains a huge matrix of expression values (genes × samples). But it contains other metadata as well. So the best way to read it is to use the `GEOquery` package in R.

```
# Install and load required packages

install.packages("factoextra")

if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("limma", update = FALSE)
BiocManager::install("GEOquery", update = FALSE)

# For Affymetrix HG-U133A (GPL96), Bioconductor provides an annotation package:
# Install required annotation packages
BiocManager::install("org.Hs.eg.db", update = FALSE)
BiocManager::install("hgu133a.db", update = FALSE)
```

Now let's load these packages.

```r
library(limma)
library(GEOquery)
library(hgu133a.db)
library(factoextra)
print("All required packages are installed and loaded.")
```

Now let's load the dataset using GEOquery.

```r
file_path <- "GSE5847_series_matrix.txt.gz"  # Adjust to your path
gse <- getGEO(filename = file_path, destdir = getwd())   # gse is an ExpressionSet object created by ge
dim(gse)
```

```
## Features  Samples
##    22283       95
```

```r
# Extract expression data and metadata
# Expression data is in the expression slot
exprs_data <- exprs(gse)
dim(exprs_data)

# Extract phenotype data for group assignment
pheno_data <- pData(gse)
head(pheno_data) # Each row corresponds to one of your 95 samples. These fields are what you'll use to

# Extract feature data
feature_data <- fData(gse)
head(feature_data)
```

The functions **exprs**, **pData**, and **fData** are provided by the package Biobase which is a dependency of GEOquery. These functions are commonly used to extract components from an ExpressionSet object like gse.

- `exprs_data` → the **expression matrix** (rows = probe sets, columns = samples). Each entry is the **expression intensity** for each probe set in each sample.
- `pheno_data` → the **phenotype metadata** (information about those samples). Each row = one sample (aligned with exprs_data columns).
- `f_data` → the **feature data**. Each row = one probe set (aligned with exprs rows).

Quick sanity check for grouping

Now let's use the extracted `pheno_data` to check how many samples belong to each group :

```r
table(pheno_data$`source_name_ch1`)
```

```
##
##           human breast cancer stroma human breast cancer tumor epithelium
##                                   47                                    48
```

```r
table(pheno_data$`characteristics_ch1`)
```

```
##
##     diagnosis: IBC diagnosis: non-IBC
##                 26                 69
```

From the above it is evident that there are nearly equal number of tumor and stroma samples (47 vs 48). But there are 15 IBC and 80 non-IBC samples. Hence I will first focus on comparing tumor vs stroma samples.

Now let's map Affymetrix probe IDs (like 202431_at) to gene symbols (like TP53) using the hgu133a.db annotation package for the HG-U133A (GPL96) platform. I'll use the `mapIds` function, provided by the AnnotationDbi package (a dependency of hgu133a.db) to query the hgu133a.db database to retrieve annotations.

```
# Map probe IDs to symbols

probe_ids <- rownames(exprs_data)        # all 22,283 probes
gene_symbols <- mapIds(
  hgu133a.db,
  keys = probe_ids,
  column = "SYMBOL",
  keytype = "PROBEID",
  multiVals = "first"
)
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
# Add gene symbols as a new column to feature data
feature_data <- data.frame(ProbeID = probe_ids, GeneSymbol = gene_symbols)
head(feature_data)
```

```
##             ProbeID GeneSymbol
## 1007_s_at 1007_s_at       DDR1
## 1053_at     1053_at       RFC2
## 117_at       117_at      HSPA6
## 121_at       121_at       PAX8
## 1255_g_at 1255_g_at     GUCA1A
## 1294_at     1294_at       UBA7
```

Now let's collapse multiple probe sets into one expression value per gene (so you go from ~22k probes → ~13k genes)

```
#The chunk reduces the dataset from 22,283 probes to 13,039 genes by averaging expression values for pr

# 1. Attach gene symbols to the expression data
exprs_df <- as.data.frame(exprs_data)
exprs_df$GeneSymbol <- gene_symbols

# 2. Remove probes that didn't map to a gene (NA)
exprs_df <- exprs_df[!is.na(exprs_df$GeneSymbol), ]

# 3. Collapse multiple probe sets into one row per gene (by mean here)
library(dplyr)
library(tibble) # Add this line to load the tibble package
gene_exprs <- exprs_df %>%
```

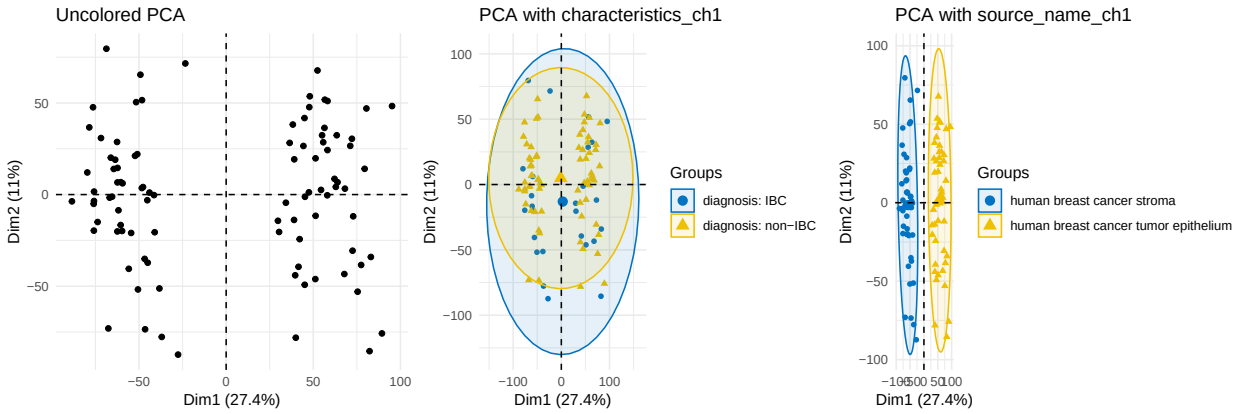
```
  group_by(GeneSymbol) %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE))

# Convert back to matrix with GeneSymbol as rownames
gene_exprs_mat <- as.matrix(column_to_rownames(gene_exprs, "GeneSymbol"))

# Check new dimensions
dim(gene_exprs_mat)
```

```
## [1] 13039    95
```

Now `gene_exprs_mat` is our clean **gene** $\times$ **sample** expression matrix. With this matrix, we can ask next: "Can we cluster these samples into distinct groups based on their gene expression profiles?" We know the answer to this question is YES, because the experimenter of this study has mentioned groups like Stroma & Tumor (based on tissue) or IBC & non-IBC (based on cancer subtype). However it's impossible to understand any clustering in a huge 13,039 dimensional space. Hence we will reduce it to 2 principal component axes using PCA.

EDA through Principal Component Analysis (PCA)

```
library(factoextra)
library(ggplot2)

# Perform PCA
pca_result <- prcomp(t(gene_exprs_mat), scale. = TRUE)

# Plot 1: Uncolored
p1 <- fviz_pca_ind(pca_result, label = "none", title = "Uncolored PCA") + theme_minimal()

# Plot 2: Colored by characteristics_ch1
p2 <- fviz_pca_ind(pca_result, label = "none", habillage = pheno_data$characteristics_ch1,
                   palette = "jco", addEllipses = TRUE, title = "PCA with characteristics_ch1") + theme

# Plot 3: Colored by source_name_ch1
p3 <- fviz_pca_ind(pca_result, label = "none", habillage = pheno_data$source_name_ch1,
                   palette = "jco", addEllipses = TRUE, title = "PCA with source_name_ch1") + theme_mini

# Arrange plots side by side
library(gridExtra)
grid.arrange(p1, p2, p3, ncol = 3)
```



Explanation

It is evident that the variation along x-axis is greater and that's because it represents the first principal component, explaining 27.4% of the total variance. Further this variance in PC1 (x-axis) is able to single-handedly separate the tumour and stroma samples, but not really the IBC and non-IBC samples.

- **PCA**: `prcomp()` computes PCA once, reused for all plots.

  `fviz_pca_ind()` function from the `factoextra` package helps to visualize the results of PCA through a scatter plot over the first 2 PC axes by default. Data points can be colored based on various factors, such as group membership (`habillage`), quality of representation (`cos2`), or contribution to the principal components (`contrib`).

  Here, I use `habillage` which clusters data points based on labeled class data available in the dataset, thus dispensing with a k-means clustering step required in absence of labeled class data.

  I also added confidence ellipses around groups of data points to show their spread.

- **Plots**:
  - `p1`: Uncolored scatter plot using `fviz_pca_ind` with no `habillage`.
  - `p2`: Colored by `characteristics_ch1` with ellipses.
  - `p3`: Colored by `source_name_ch1` with ellipses.

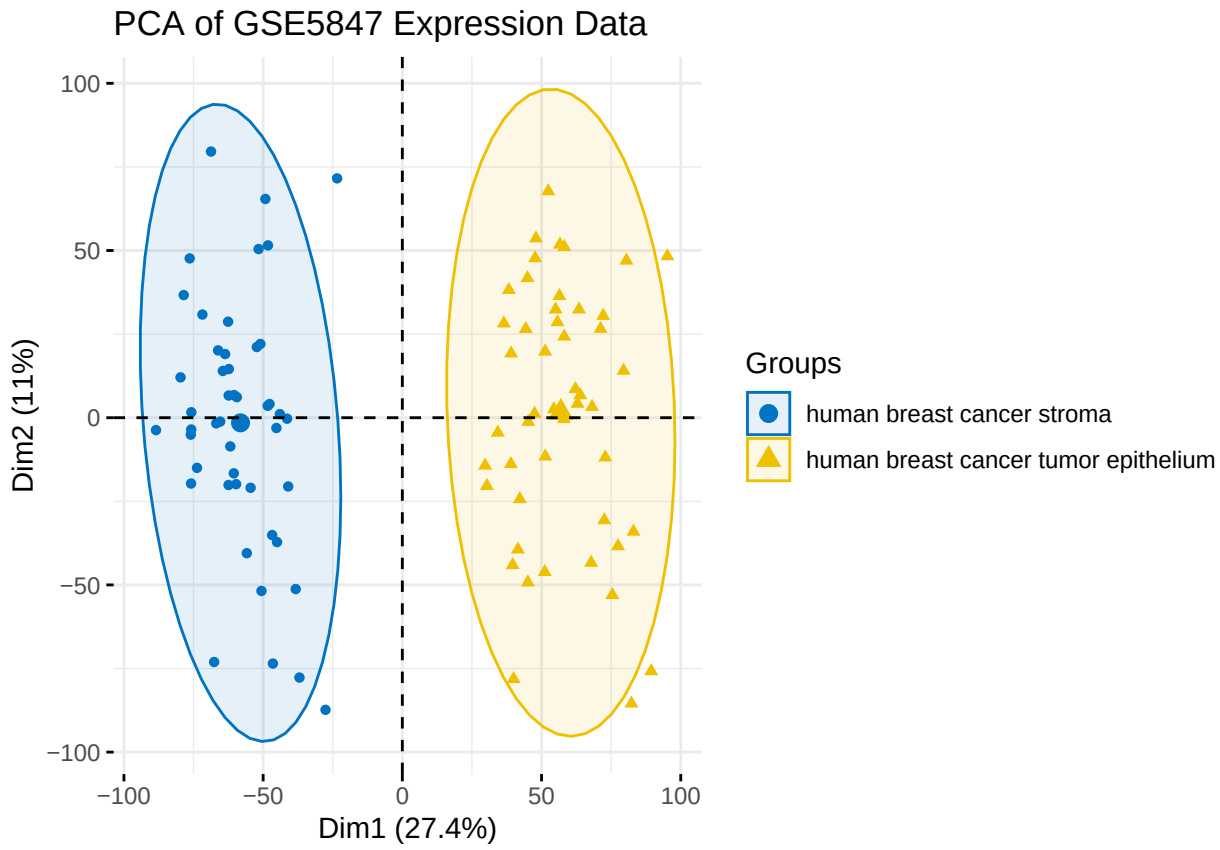
- **Layout**: `grid.arrange` displays the three plots horizontally (`ncol = 3`), with `fig.width=12` and `fig.height=4` setting the overall figure size.

- **Options**: `warning=FALSE, message=FALSE` suppresses messages, and `include=TRUE` shows the code.

The separation is better in case of `source_name_ch1` as cf. `characteristics_ch1`. So now let's separately plot the PCA plot colored by `source_name_ch1` which indicates whether the sample is from tumor epithelium or stroma.

```
# Load required packages
library(factoextra)
library(ggplot2)

# Perform PCA on the expression matrix
pca_result <- prcomp(t(gene_exprs_mat), scale. = TRUE)

# Visualize PCA results
```

```r
fviz_pca_ind(pca_result,
             label = "none", # Remove sample labels for clarity
             habillage = pheno_data$source_name_ch1, # Color by sample type if available in pheno_data
             palette = "jco", # Use a clear color scheme
             addEllipses = TRUE) + # Add confidence ellipses
  ggtitle("PCA of GSE5847 Expression Data")
```




```r
# Check variance explained by principal components
summary(pca_result)$importance[, 1:3]
```

```
##                             PC1      PC2      PC3
## Standard deviation     59.79083 37.81570 27.63147
## Proportion of Variance  0.27417  0.10967  0.05855
## Cumulative Proportion   0.27417  0.38385  0.44240
```

The above EDA/PCA evidence of group separation along PC1 motivates us to see whether **gene expression differs** between stroma and tumor epithelium. In other words, PCA provided us a hypothesis that we'll quantitatively test next. So let's now perform differential expression analysis using the `limma` package.

Differential Gene Expression Analysis (a hypothesis testing)

Our objective :

- Let's assume our null hypothesis for each gene is that its expression level does not differ significantly between stroma and tumor samples—meaning any observed difference is due to random chance, not true biological variation.
- A low p-value (e.g., <0.05) suggests that such a difference is unlikely to occur by random chance, so the null is rejected for that gene—implying the gene is differentially expressed
- Results will finally be ranked by ascending p-value to prioritize genes offering the strongest evidence against the null hypothesis

The `limma` package is designed for analyzing microarray and RNA-seq data, particularly for identifying differentially expressed genes between conditions. It uses linear models to assess the significance of differences in gene expression across groups, making it suitable for this dataset where we want to compare tumor epithelium vs. stroma samples.

```
# Load required packages
library(limma)
library(statmod)

# Define groups based on pheno_data$source_name_ch1
group <- factor(pheno_data$source_name_ch1, levels = c("human breast cancer stroma", "human breast canc

# Create design matrix with valid names
design <- model.matrix(~ 0 + group)
colnames(design) <- c("stroma", "tumor_epithelium") # Replace spaces with valid names
print(head(design)) # Show design matrix structure
```

```
##   stroma tumor_epithelium
## 1      1                0
## 2      1                0
## 3      1                0
## 4      1                0
## 5      1                0
## 6      1                0
```

```
# Fit linear model
fit <- lmFit(gene_exprs_mat, design)

# Define contrast (tumor vs. stroma)
contrast.matrix <- makeContrasts("tumor_epithelium - stroma", levels = design)
fit2 <- contrasts.fit(fit, contrast.matrix)
fit2 <- eBayes(fit2)

# Extract top differentially expressed genes
results <- topTable(fit2, number = 10, adjust.method = "BH")
print(results) # Show top 10 genes with logFC, p-value, and adjusted p-value
```

```
##                logFC  AveExpr         t      P.Value    adj.P.Val        B
## MYCT1     -0.8140965 4.755817 -27.78317 1.715576e-47 2.236940e-43 97.63931
## HNRNPCL1   0.6942216 5.604739  24.88383 1.772994e-43 1.155903e-39 88.55250
```

```
## KCNIP2     0.6242360 4.989503  24.41926 8.397175e-43 3.649692e-39 87.01962
## LINC00939  0.6282673 4.455900  23.92877 4.440695e-42 1.169028e-38 85.37690
## POU1F1     0.3693322 3.511073  23.92601 4.482813e-42 1.169028e-38 85.36758
## POTEKP     0.7862386 5.223100  23.68905 1.011265e-41 2.197648e-38 84.56475
## MTHFSD     0.9287392 5.642040  23.37523 2.996763e-41 5.582113e-38 83.49230
## PRRG2      1.3050584 5.660269  22.83177 2.015398e-40 3.284846e-37 81.60967
## PGK2       0.5502890 5.130680  22.79254 2.315438e-40 3.354556e-37 81.47252
## TSHB       0.6783768 6.252285  21.25441 6.109021e-38 7.965552e-35 75.95734
```

Explanation

Loading Packages

- `library(limma)`: Loads the `limma` package, which provides tools for linear modeling of gene expression data.
- `library(statmod)`: A dependency for `limma`, offering statistical functions like empirical Bayes moderation.

Defining Groups

- `group <- factor(pheno_data$source_name_ch1, ...)`:
  - `pheno_data$source_name_ch1` contains metadata (e.g., "human breast cancer stroma" or "human breast cancer tumor epithelium") for your 95 samples.
  - `factor()` converts this into a categorical variable, with `levels` specifying the order (stroma first, tumor second).
  - This defines the experimental groups we're comparing, based on biological context from our EDA.

Creating the Design Matrix

- `design <- model.matrix(~ 0 + group)`:
  - `model.matrix` creates a matrix encoding the group structure. The `~ 0 + group` formula means no intercept (avoids a reference level) and includes a column for each group.
  - Result: A 95 × 2 matrix (one row per sample, one column per group) with 0s and 1s (e.g., 1 for stroma, 0 for tumor).
- `colnames(design) <- c("stroma", "tumor_epithelium")`:
  - Renames columns to valid R names (replacing spaces with underscores) because `makeContrasts` requires syntactically correct names.

Fitting the Linear Model

- `fit <- lmFit(gene_exprs_mat, design)`:
  - `lmFit` fits a linear model for each of the 13,039 genes in `gene_exprs_mat` against the design matrix.
  - Mathematically, for each gene g, it fits: $Y_g = \beta_{g1}X_1 + \beta_{g2}X_2 + \epsilon_g$,
    where Y_g is the 95 × 1 expression vector for gene g,X_1 and X_2 are the 95 × 1 **group indicator** columns from the design matrix, and β_{g1}, β_{g2} are the **coefficients** estimating group effects.
  - For each of the 13,039 rows (genes) in `gene_exprs_mat`, `lmFit` fits a linear model using the 95 group indicators from design matrix, producing an output `fit` with 13,039 × 2 coefficients
  - Thus for each gene, it estimates expression as a function of group membership (stroma or tumor).
  - This is like running a separate regression for every gene, capturing how group affects expression.

Defining and Applying the Contrast

- After we get `fit` i.e. the matrix of expression in the 2 groups (tumor_epithelium and stroma) for each gene, we now have a hypothesis test : Is the **expression difference** between these 2 groups significant for each gene?
- So now we want a new `fit2` matrix which merges the 2 columns of the `fit` matrix by finding their difference, thus being a 13,039 $\times$ 1 matrix of **contrast estimates** (e.g., [1.3, -0.1, 1.4, ...] for all genes). In other words, we want to find genes where $[\beta_{tumor} - \beta_{stroma}]$ is significantly different from zero. This is where contrasts come in.
- Thus we want a weight vector: $[-1, 1]$ (called `contrast.matrix`) applied to the coefficients $[\beta_{stroma}, \beta_{tumor}]$ to compute the difference. We find this `contrast.matrix` using `makeContrasts()` function of R.
- We then use this `contrast.matrix` to compute the new `fit2` matrix using `contrasts.fit()` function of R.
- Finally we use `eBayes()` function to compute **moderated**(i.e. reduced variances) t-statistics and p-values for each gene.

Extracting Results

- `results <- topTable(fit2, number = 10, adjust.method = "BH")`:
  - This ranks genes by significance
  - `topTable` extracts the top 10 genes with the smallest adjusted p-values (< 0.05 indicates significance) and sorts accordingly (not by logFC).That means the most significant genes are at the top of the table.
  - `adjust.method = "BH"`: Applies the Benjamini-Hochberg method to control the false discovery rate (FDR), correcting for multiple testing across 13,039 genes.
  - Output includes: log fold change (logFC), average expression (AveExpr), t-statistic, p-value, and adjusted p-value (adj.P.Val).
  - **Results Table**: Top 10 genes with columns like:
  - `logFC`: Difference in expression (tumor - stroma). (A positive logFC means higher expression in tumor; negative means higher in stroma.)
  - `adj.P.Val`: Adjusted p-value (e.g., < 0.05 indicates significance).
  - `logFC` $\rightarrow$ log fold change between groups.
  - `AveExpr` $\rightarrow$ average log expression across all samples.
  - `t` $\rightarrow$ moderated t-statistic.
  - `P.Value` $\rightarrow$ raw p-value.
  - `B` $\rightarrow$ log-odds of being DE

Key Insights

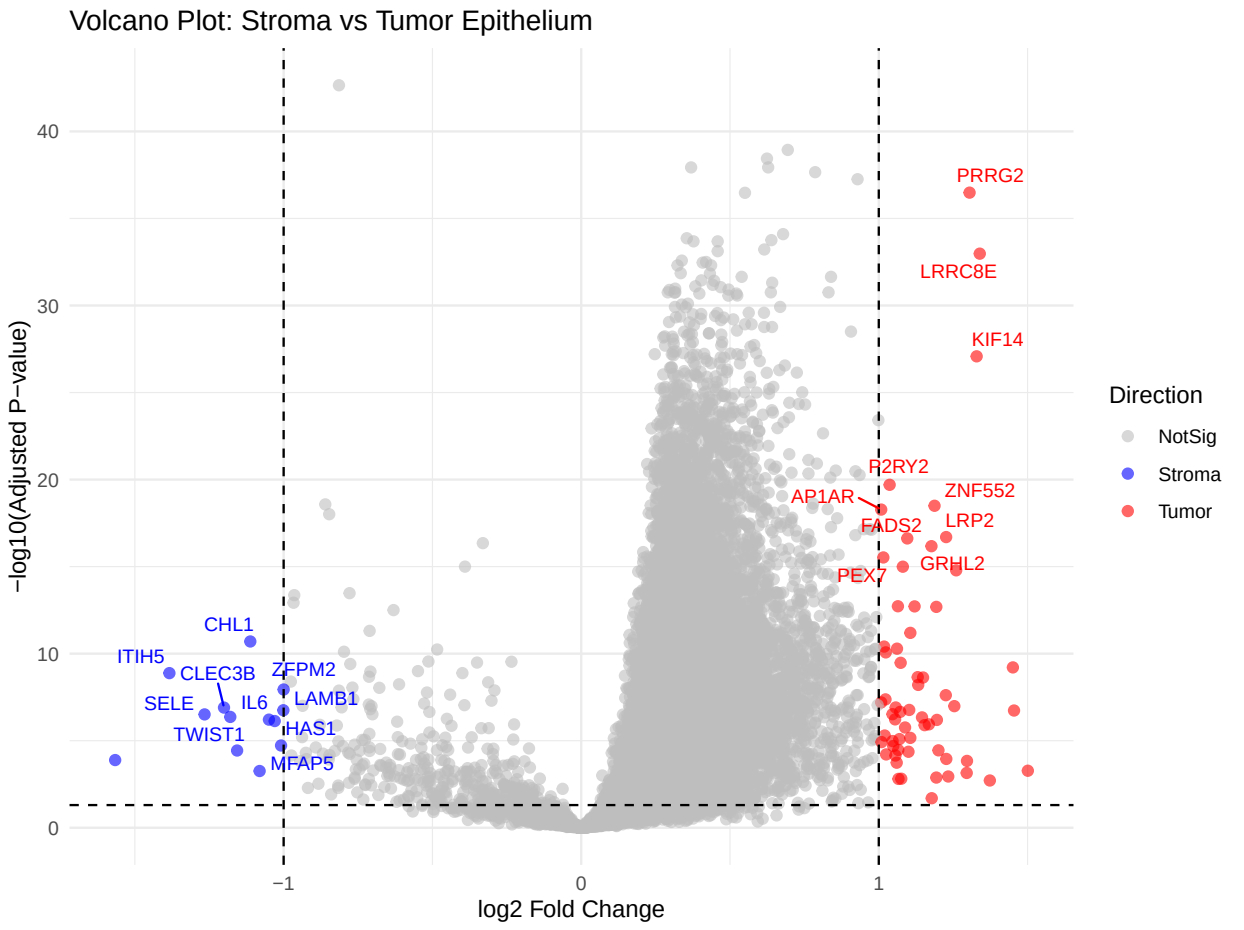
- The code tests whether gene expression differs between stroma and tumor epithelium, building on your EDA where PCA showed separation.
- Significant genes (low adj.P.Val) are candidates for further biological investigation (e.g., tumor-specific markers).

Now the resulting `topTable` output, with metrics like log fold change (logFC) and adjusted p-values (adj.P.Val), provides a ranked list of differentially expressed genes.

Volcano Plot

The top 10 genes represent only a subset of the 13,039 genes. To get a wider visual idea of the differential expression of all genes at large, the next logical step is to visualize the full dataset using a volcano plot. For this we'll only need 2 columns from the `fit2` matrix : **logFC** & **adj.P.Val**. This plot will map logFC against -log10(adj.P.Val) for all genes, highlighting those with both statistical significance (e.g., adj.P.Val $<$ 0.05) and biological relevance (e.g., |logFC| $>$ 1).

```
library(ggplot2)
library(ggrepel)

# 1. Get all results
all_results <- topTable(fit2, number = Inf, adjust.method = "BH")

# 2. Classify ALL genes by p-value + direction
all_results$diffexpressed <- "NotSig"
all_results$diffexpressed[all_results$logFC > 1 & all_results$adj.P.Val < 0.05] <- "Tumor"
all_results$diffexpressed[all_results$logFC < -1 & all_results$adj.P.Val < 0.05] <- "Stroma"

# 3. Pick top 10 per side for labeling
top_up <- all_results[all_results$diffexpressed == "Tumor", ][1:10, ]
top_down <- all_results[all_results$diffexpressed == "Stroma", ][1:10, ]
top_genes <- rbind(top_up, top_down)

# 4. Volcano plot
ggplot(all_results, aes(x = logFC, y = -log10(adj.P.Val))) +
  geom_point(aes(color = diffexpressed), alpha = 0.6, size = 2) +
  scale_color_manual(values = c("Tumor" = "red", "Stroma" = "blue", "NotSig" = "grey")) +
  geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "black") +
  geom_vline(xintercept = c(-1, 1), linetype = "dashed", color = "black") +
  geom_text_repel(
    data = top_genes,
    aes(label = rownames(top_genes), color = diffexpressed),
    size = 3, max.overlaps = 20, show.legend = FALSE
  ) +
  labs(title = "Volcano Plot: Stroma vs Tumor Epithelium",
       x = "log2 Fold Change",
       y = "-log10(Adjusted P-value)",
       color = "Direction") +
  theme_minimal()
```



Explanation of the Volcano Plot Code

Data Preparation

- `all_results` contains the full differential expression results for all 13,039 genes, including logFC and adj.P.Val.
- `diffexpressed` is a new column I created to classify genes into groups (Tumor / Stroma / NotSig). This is used in the ggplot to color the points.

In the code `top_up <- all_results[all_results$diffexpressed == "Tumor", ][1:10, ]` I select the top 10 genes with highest logFC (most upregulated in tumor)

And in `top_down <- all_results[all_results$diffexpressed == "Stroma", ][1:10, ]` I select the top 10 genes with lowest logFC (most upregulated in stroma). These 20 genes are stored in `top_genes` for labeling on the ggplot.

The plot

```
ggplot(all_results, aes(x = logFC, y = -log10(adj.P.Val)))
```

- Each dot = one gene.
- `x = logFC`: how much higher in epithelium (positive) or stroma (negative).
- `y = -log10(adj.P.Val)`: higher up means stronger statistical significance.
  - Example: adj.P.Val = 0.05 → –log10(0.05) 1.3. So **top-left genes** = significantly higher in stroma, **top-right genes** = significantly higher in tumor epithelium.

Threshold lines

```
geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "black")
```

- Horizontal line = adjusted p-value cutoff of 0.05.
- Genes above this line = "statistically significant."

```
geom_vline(xintercept = c(-1, 1), linetype = "dashed", color = "black")
```

- Vertical lines = logFC cutoff of ±1 (i.e., 2-fold change).
- Genes outside these lines = "biologically meaningful effect size."

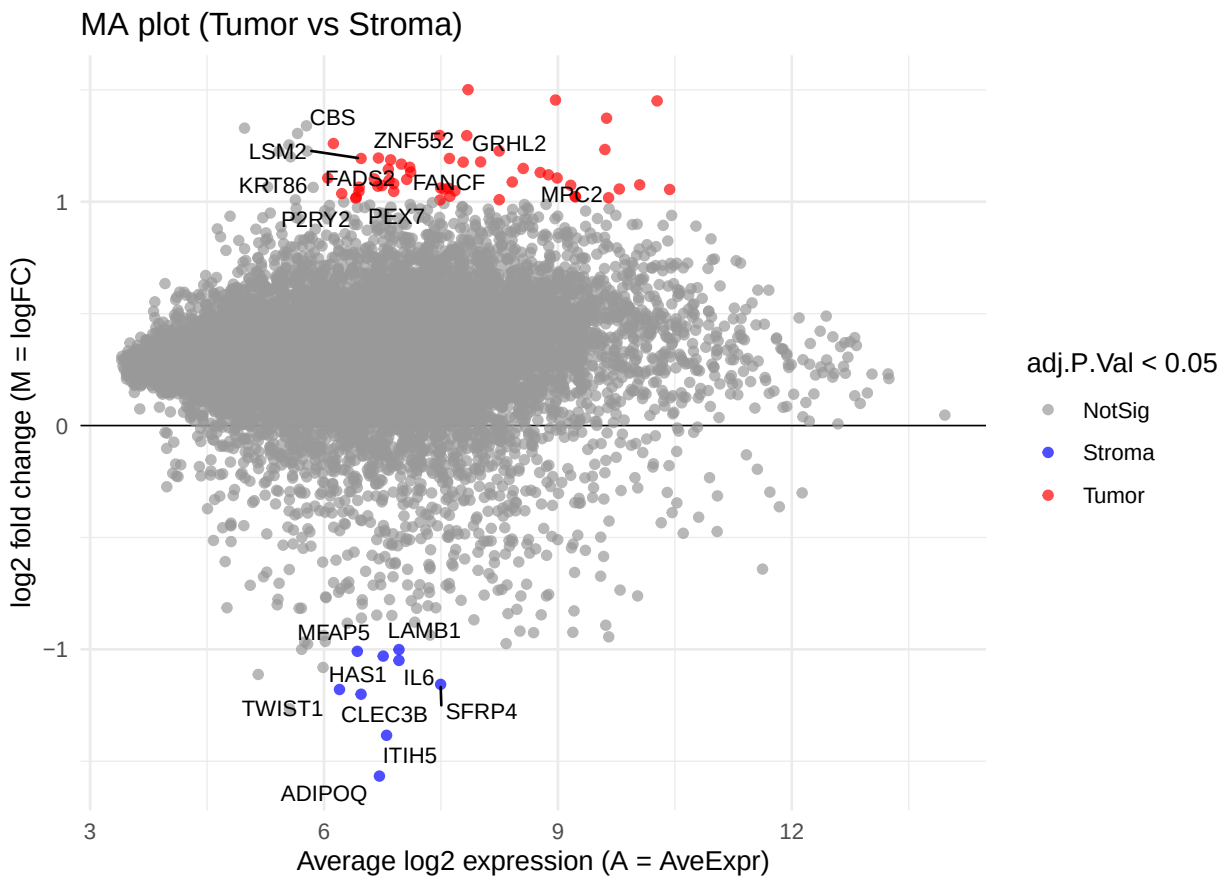
Now Volcano Plot answers the question : "Are the strongest fold changes statistically reliable?", but it still leaves the question : "*Do strong fold changes happen across well-expressed genes, or only in noisy low-expression ones?*". This will be answered by the MA Plot.

MA Plot

Here we will plot the effect size (fold change) vs average expression. So we will take these columns from the fit2 matrix : logFC (y-axis) & AveExpr (x-axis).

```
library(ggplot2)
library(ggrepel)   # optional for labeling

# Parameters you can change
p_cutoff <- 0.05    # significance threshold
label_n   <- 10     # top n genes per side to label (optional)

# 1) Get a results table (contains AveExpr)
all_results <- topTable(fit2, number = Inf, adjust.method = "BH")

# confirm columns
stopifnot(all(c("logFC", "AveExpr", "adj.P.Val") %in% colnames(all_results)))
```

```r
# Classify ALL genes by logFC + adj.P.Val + AveExpr
all_results$direction <- "NotSig"

all_results$direction[
  all_results$logFC > 1 &
  all_results$adj.P.Val < 0.05 &
  all_results$AveExpr > 6
] <- "Tumor"

all_results$direction[
  all_results$logFC < -1 &
  all_results$adj.P.Val < 0.05 &
  all_results$AveExpr > 6
] <- "Stroma"


# 3) Pick genes to label (top by adj.P.Val separately on each side)
top_up    <- subset(all_results, direction == "Tumor")
top_dn    <- subset(all_results, direction == "Stroma")
top_up_lab <- head(top_up[order(top_up$adj.P.Val), ], label_n)
top_dn_lab <- head(top_dn[order(top_dn$adj.P.Val), ], label_n)
labels_df  <- rbind(top_up_lab, top_dn_lab)

# 4) Make MA plot (AveExpr on x, logFC on y)
ggplot(all_results, aes(x = AveExpr, y = logFC)) +
  geom_hline(yintercept = 0, color = "black", linetype = "solid", size = 0.3) +
  geom_point(aes(color = direction), alpha = 0.7, size = 1.5) +
  scale_color_manual(values = c("Tumor" = "red", "Stroma" = "blue", "NotSig" = "grey60")) +
  geom_text_repel(data = labels_df, aes(label = rownames(labels_df)),
                  size = 3, max.overlaps = 20, show.legend = FALSE) +
  labs(title = "MA plot (Tumor vs Stroma)",
       x = "Average log2 expression (A = AveExpr)",
       y = "log2 fold change (M = logFC)",
       color = paste0("adj.P.Val < ", p_cutoff)) +
  theme_minimal()
```



`AveExpr` = the **average log expression** for each gene across all 95 samples. So the x-axis is basically "brightness of the probe on average."

By keeping a AveExpr > 6 criteria in the code, I have colored (Red or Blue) only those genes which have a high expression.

Finally I will list only those Tumor and Stroma expressed genes which are colored in this plot, which will then be a subject of gene expression biomarker analysis next.

```
library(dplyr)
library(tibble)

# 1) Add rownames as a proper column called "Gene"
all_results_tbl <- all_results %>%
  rownames_to_column(var = "Gene")

# 2) Keep only Tumor and Stroma genes
sig_genes <- all_results_tbl %>%
  filter(direction %in% c("Tumor", "Stroma")) %>%
  select(Gene, direction, adj.P.Val)

# 3) Split and sort by adjusted p-value
tumor_genes <- sig_genes %>%
  filter(direction == "Tumor") %>%
  arrange(adj.P.Val) %>%
  select(Gene, adj.P.Val)
```

```r
stroma_genes <- sig_genes %>%
  filter(direction == "Stroma") %>%
  arrange(adj.P.Val) %>%
  select(Gene, adj.P.Val)

# 4) Merge into one table with padding
max_len <- max(nrow(tumor_genes), nrow(stroma_genes))
summary_table <- data.frame(
  Tumor  = c(tumor_genes$Gene,  rep(NA, max_len - nrow(tumor_genes))),
  Stroma = c(stroma_genes$Gene, rep(NA, max_len - nrow(stroma_genes))),
  Tumor_adjP  = c(tumor_genes$adj.P.Val,  rep(NA, max_len - nrow(tumor_genes))),
  Stroma_adjP = c(stroma_genes$adj.P.Val, rep(NA, max_len - nrow(stroma_genes)))
)

# 5) Preview first rows
head(summary_table, 20)
```

```
##       Tumor Stroma   Tumor_adjP  Stroma_adjP
## 1     P2RY2  ITIH5 1.997369e-20 1.312656e-09
## 2    ZNF552 CLEC3B 3.198514e-19 1.254931e-07
## 3     FADS2  LAMB1 2.403227e-17 1.779328e-07
## 4     GRHL2 TWIST1 6.695859e-17 4.304647e-07
## 5      PEX7    IL6 2.975608e-16 6.177928e-07
## 6     FANCF   HAS1 1.010508e-15 7.431532e-07
## 7       CBS  MFAP5 1.633424e-15 1.885980e-05
## 8      MPC2  SFRP4 1.925939e-13 3.664215e-05
## 9      LSM2 ADIPOQ 2.057080e-13 1.296631e-04
## 10    KRT86   <NA> 6.395752e-12           NA
## 11   BCAP31   <NA> 3.918385e-11           NA
## 12    CLDN7   <NA> 5.163347e-11           NA
## 13    TMED3   <NA> 8.452987e-11           NA
## 14 HSP90AB1   <NA> 3.360842e-10           NA
## 15   SPINT2   <NA> 6.167168e-10           NA
## 16    VAMP8   <NA> 2.236943e-09           NA
## 17    IGSF3   <NA> 2.342912e-09           NA
## 18   INTS13   <NA> 6.231515e-09           NA
## 19  TMEM14A   <NA> 4.238146e-08           NA
## 20    ESRP2   <NA> 6.325230e-08           NA
```

Conclusion

So we've got our high-confidence DEGs, most of which are tumor-up genes. The top hits are consistent with biology. For example e.g. **IL6** is a known inflammatory cytokine in the stroma, and **FADS2, GRHL2** are tumor-associated regulators of metabolism/epithelial phenotype.

In the next stage of this project, I will explore the regulatory landscape of these genes through upstream transcription factor analysis, and assess their translational potential through downstream biomarker discovery pipelines.