RNA-Seq Exploratory Data Analysis

Jacob Hoover

Table of contents

1 EDA

Explore stomach cancer cell lines in CCLE. How many biologically distinct groups can you reasonably define? What do the gene expression profiles of each group tell us about the biology of each group?

2 Load Libraries

```
# Data wrangling
library(tidyr)
library(dplyr)

# Bioconductor
```

```
library(DESeq2)
library(ComplexHeatmap)
library(biomaRt)

# Data visualization
library(ggrepel)
library(ggplot2)
library(scales)
library(patchwork)
library(EnhancedVolcano)
```

3 Prepare Data

First, the counts data, `dat1`, and annotation data, `ann1`, are read in with the first 5 rows displayed as a sanity check.

```
# read in counts and metadata

dat1 <- read.table("data/CCLE_RNAseq_genes_counts_20180929.gct.gz",
                   header=TRUE,
                   skip=2)

ann1 <- read.csv("data/Cell_lines_annotations_20181226.txt",
                   header=TRUE,
                   sep="\t")

# Sanity Check
head(dat1, 5)
head(ann1, 5)
```

The row names of `dat1` are set to the `Description` column which contains the hgnc_symbols of each gene. The `Name` and `Description` columns of `dat1` are removed so the dataframe only holds count data.

```
# Set Description to rownames and sum duplicates before removing
gene_names <- dat1$Description
dat1 <- dat1[, !(colnames(dat1) %in% c("Name","Description"))]
dat1 <- rowsum(dat1, group = gene_names)

cat("Number of genes in counts data:", dim(dat1)[1], "\n", "Number of samples in counts data
```

```
Number of genes in counts data: 54271
 Number of samples in counts data: 1019
```

```
cat("Number of cell lines in metadata:", dim(ann1)[1], "\n", "Number of features in metadata
```

```
Number of cell lines in metadata: 1461
 Number of features in metadata: 33
```

Below are the 33 features of the cell lines stored in the metadata. This is reduced to ten columns that may be impactful on the RNA-seq count data.

```
print(colnames(ann1))
```

```
 [1] "CCLE_ID"                     "depMapID"
 [3] "Name"                        "Pathology"
 [5] "Site_Primary"                "Site_Subtype1"
 [7] "Site_Subtype2"               "Site_Subtype3"
 [9] "Histology"                   "Hist_Subtype1"
[11] "Hist_Subtype2"               "Hist_Subtype3"
[13] "Gender"                      "Life_Stage"
[15] "Age"                         "Race"
[17] "Geo_Loc"                     "inferred_ethnicity"
[19] "Site_Of_Finding"             "Disease"
[21] "Annotation_Source"           "Original.Source.of.Cell.Line"
[23] "Characteristics"             "Growth.Medium"
[25] "Supplements"                 "Freezing.Medium"
[27] "Doubling.Time.from.Vendor"   "Doubling.Time.Calculated.hrs"
[29] "type"                        "type_refined"
[31] "PATHOLOGIST_ANNOTATION"      "mutRate"
[33] "tcga_code"
```

```
# Keep important metadata columns
ann2 <- ann1[ , c("CCLE_ID", "depMapID", "Name", "Pathology", "Site_Primary", "Histology", "C
```

Preprocessing continues by identifying NA, NaN, and Null values within `ann2`. Each column is looped through and all identified values are replaced with `missing info`.

```
# Check NA/NaN/Null values
cat("Total NA values:", sum(is.na(ann2)))
```

```
Total NA values: 3323
```

```
cat("Total NaN values:", sum(is.nan(as.matrix(ann2))))
```

```
Total NaN values: 0
```

```
cat("Total Null values:", sum(sapply(ann2, is.null)))
```

```
Total Null values: 0
```

```
for (col in colnames(ann2)) {
  na_count <- sum(is.na(ann2[[col]]))
  cat(col, ":", na_count, "\n")
}
```

```
CCLE_ID : 0
depMapID : 4
Name : 400
Pathology : 400
Site_Primary : 400
Histology : 400
Gender : 400
Age : 662
Site_Of_Finding : 413
tcga_code : 244
```

```
# Replace NA values
ann2[is.na(ann2)] <- "missing info"

# Sanity check
sum(is.na(ann2))
```

```
[1] 0
```

The cell lines are stored as rows in `ann2` and as columns in `dat1`. This requires the ordering and merging of the two dataframes by their `CCLE_ID` so that the expression counts and annotation features align for each cell line.

```
print(sum(ann2$CCLE_ID %in% colnames(dat1)))
```

```
[1] 1004
```

```
# Order counts with metadata by CCLE_ID
order_counts <- colnames(dat1) %in% ann2$CCLE_ID
dat1 <- dat1[ ,order_counts]

# Order metadata with counts
order_meta  <- ann2$CCLE_ID %in% colnames(dat1)
ann2 <- ann2[order_meta, ]

# Merge ordered datasets
dat_counts <- dat1[ ,ann2$CCLE_ID]
```

The `Site_Primary` feature is used to identify the different cancer cell lines stored in the dataframes. This column is used as indices to extract `stomach` cancer cell lines and subset the count and annotation dataframes so that they only contain these samples. There are now 54,271 genes and 36 samples.

```
# subset by cell line tissue
print(unique(ann2$Site_Primary))
```

```
 [1] "lung"                                "large_intestine"
 [3] "haematopoietic_and_lymphoid_tissue" "urinary_tract"
 [5] "bone"                                "skin"
 [7] "breast"                              "liver"
 [9] "pleura"                              "ovary"
[11] "oesophagus"                          "endometrium"
[13] "central_nervous_system"              "soft_tissue"
[15] "pancreas"                            "autonomic_ganglia"
[17] "stomach"                             "kidney"
[19] "upper_aerodigestive_tract"           "salivary_gland"
[21] "biliary_tract"                       "thyroid"
[23] "prostate"                            "small_intestine"
[25] "missing info"
```

```
# Matrix of stomach cell lines
index_stomach <- ann2[ann2$Site_Primary == "stomach", ]
dim(index_stomach)
```

```
[1] 36 10
```

```
# Look at values for other features
print(index_stomach$Pathology)
```

```
 [1] "metastasis" "metastasis" "metastasis" "metastasis" "metastasis"
 [6] "primary"    "metastasis" "metastasis" "metastasis" "metastasis"
[11] "primary"    "primary"    "primary"    "metastasis" "metastasis"
[16] "metastasis" "metastasis" "primary"    "metastasis" "metastasis"
[21] "metastasis" "primary"    "metastasis" "metastasis" "metastasis"
[26] "primary"    "primary"    "metastasis" "primary"    "metastasis"
[31] "primary"    "metastasis" "primary"    "primary"    "metastasis"
[36] "metastasis"
```

```
print(index_stomach$Site_Of_Finding)
```

```
 [1] "lymph_node" "muscle"     "lymph_node" "liver"      "lymph_node"
 [6] ""           "liver"      "ascites"    "NS"         "ascites"
[11] ""           ""           ""           "lymph_node" "liver"
[16] "lymph_node" "lymph_node" ""           "lymph_node" "ascites"
[21] "liver"      ""           "ascites"    "ascites"    "ascites"
[26] ""           ""           "lymph_node" ""           "ascites"
[31] ""           "pleura"     ""           ""           "lymph_node"
[36] "liver"
```

```
# subset by stomach IDs
dat_subset <- dat_counts[ ,index_stomach$CCLE_ID]
ann_subset <- ann2[ann2$CCLE_ID %in% index_stomach$CCLE_ID, ]

# Class variable
print(unique(ann_subset$Pathology))
```

```
[1] "metastasis" "primary"
```

```
# Sanity check
all(index_stomach$CCLE_ID %in% colnames(dat_subset))
```

```
[1] TRUE
```

```r
dim(dat_subset)
```

```
[1] 54271    36
```

```r
head(dat_subset[,1:4])
```

```
          RERFGC1B_STOMACH HS746T_STOMACH LMSU_STOMACH ECC10_STOMACH
5S_rRNA                  4              8           30              2
7SK                    113             18           11              9
A1BG                   304             12          330           1231
A1BG-AS1               393            100          203            609
A1CF                    88             25           11             27
A2M                     79              1           37            100
```

Now, filter the rows of `dat_subset` so that it only contains `protein_coding` and `lncRNA` genes. I am particularly interested in long noncoding RNAs (lncRNAs) and want to analyze there differential expression in stomach cancer cell lines. This retrieves a vector of 59,758 genes with either of the two biotypes. Of these, 19,306 genes were also in the `dat_subset` counts matrix.

```r
# Use Ensembl database and "hsapiens_gene_ensembl" dataset
ensembl <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")

biotype_vals <- c("protein_coding", "lncRNA")

# Retrieve protein coding and lncRNA genes w/ attributes
coding_and_lncrna <- getBM(
  attributes = c("ensembl_gene_id", "hgnc_symbol", "gene_biotype"),
  filters = "biotype",
  values = biotype_vals,
  mart = ensembl
)

# Extract the "hgnc_symbol" column from the list of coding genes
coding_gene_ids <- coding_and_lncrna$hgnc_symbol
length(coding_gene_ids)
```

```
[1] 59758
```

```r
# Filter the "ccle_counts" data frame to keep genes from list
dat_subset <- dat_subset[rownames(dat_subset) %in% coding_gene_ids, ]
dim(dat_subset)
```

```
[1] 19306    36
```

```r
# Convert the filtered data frame to a standard data frame
dat_subset <- data.frame(dat_subset)

cat("There are now", dim(dat_subset)[1], "genes.")
```

```
There are now 19306 genes.
```

The last step of preprocessing uses a low count threshold to identify those genes with less than 100 counts across all 36 samples. The 2,129 genes falling below this threshold are removed to prevent the influence of outliers. All values in the filtered counts matrix are checked for missing or duplicated data to ensure the final matrix is ready for further analysis.

```r
# Number of genes with low counts
low_counts <- sum(rowSums(dat_subset) < 100)
cat("Number of genes with low counts:", low_counts)
```

```
Number of genes with low counts: 2129
```

```r
# filter out low count genes
dat_filtered <- dat_subset[rowSums(dat_subset) >= 100, ]

# Check NA/NaN/Null values
cat("Total NA values:", sum(is.na(dat_filtered)))
```

```
Total NA values: 0
```

```r
cat("Total NaN values:", sum(is.nan(as.matrix(dat_filtered))))
```

```
Total NaN values: 0
```

```r
cat("Total Null values:", sum(sapply(dat_filtered, is.null)))
```

```
Total Null values: 0
```

```r
# Check duplicate samples and genes
cat("Number of duplicate samples:", sum(duplicated(colnames(dat_filtered))))
```

```
Number of duplicate samples: 0
```

```r
cat("Number of duplicate genes:", sum(duplicated(rownames(dat_filtered))))
```

```
Number of duplicate genes: 0
```

```r
cat("Analysis will be performed on", nrow(dat_filtered), "genes and", ncol(dat_filtered), uni
```

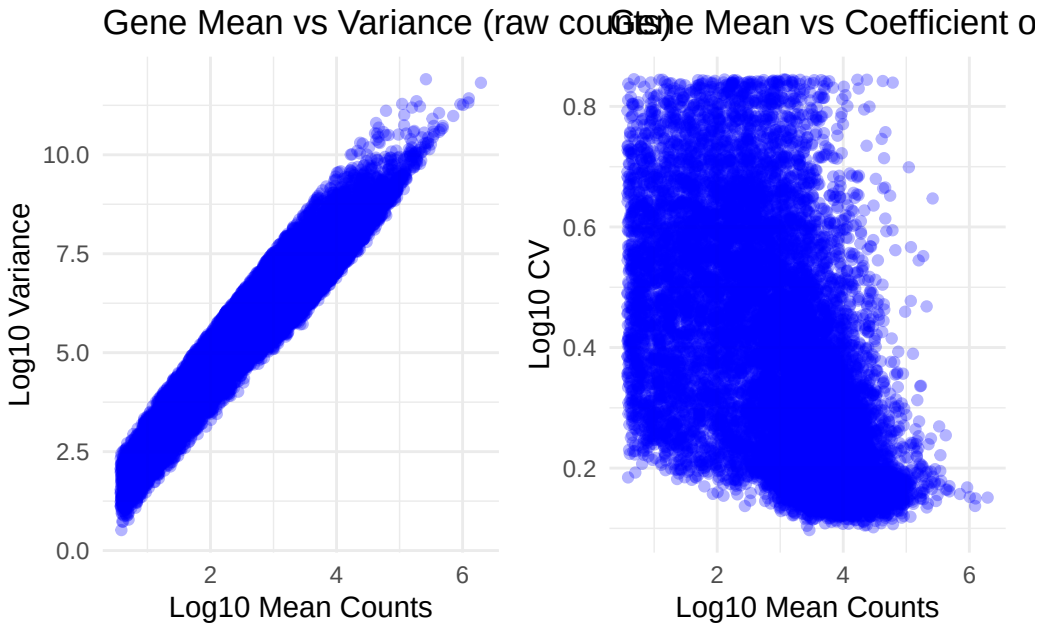
```
Analysis will be performed on 17177 genes and 36 stomach samples.
```

DESeq2 is used to create a dataset object from `dat_filtered` and `ann_subset`, with the design formula focused on the `Pathology` feature. Distribution statistics are performed on the raw data counts from the DESeq2 object. To analyze the data distribution, gene expression mean vs. variance is plotted in `p1` and displays heteroscedasticity, or the increase in variance with the increase in mean. In `p2`, mean vs. coefficient of variation (CV) shows that lowly expressed genes have higher relative variability (CV), making them less stable and more prone to technical noise, while highly expressed genes exhibit more stable relative expression patterns. Together, these plots provide quality control insights and justify the need for variance-stabilizing transformation (vst).

```r
dds <- DESeqDataSetFromMatrix(countData = dat_filtered,
                              colData = ann_subset,
                              design = ~ Pathology)
```

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

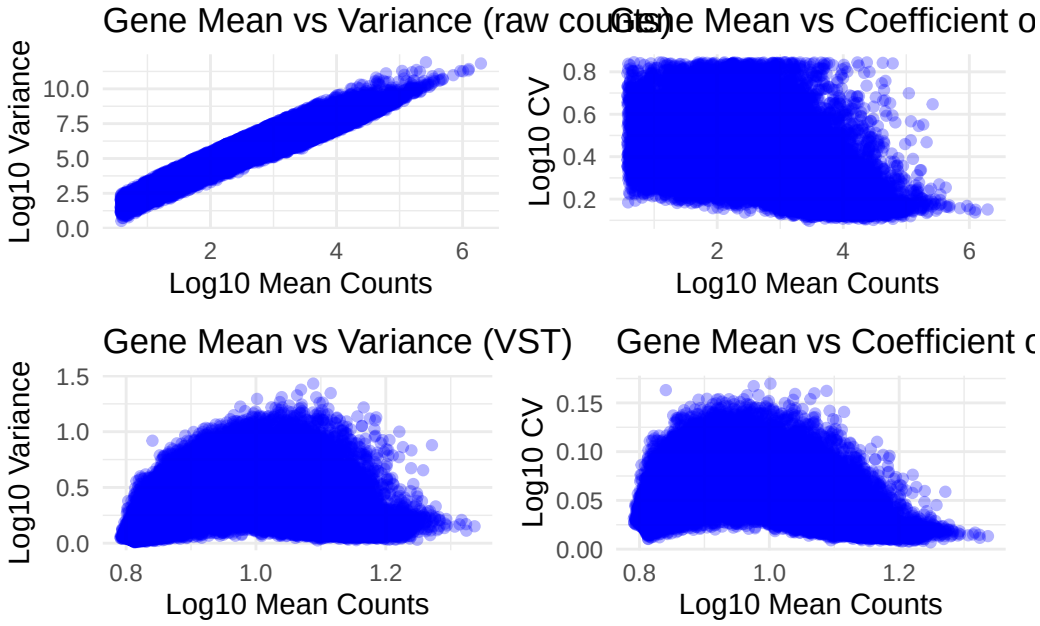
```r
vsd <- vst(dds, blind=FALSE)

dat_means <- rowMeans(counts(dds))
dat_sd <- apply(counts(dds), 1, sd)
dat_var <- apply(counts(dds), 1, var)
dat_cv <- dat_sd / dat_means

df_raw <- data.frame(
  mean = dat_means,
  sd = dat_sd,
  var = dat_var,
  cv = dat_cv
)

p1 <- ggplot(df_raw, aes(x = log10(mean + 1), y = log10(var + 1))) +
  geom_point(alpha = 0.3, color = "blue") +
  labs(
    x = "Log10 Mean Counts",
    y = "Log10 Variance",
    title = "Gene Mean vs Variance (raw counts)"
  ) +
  theme_minimal()

p2 <- ggplot(df_raw, aes(x = log10(mean + 1), y = log10(cv + 1))) +
  geom_point(alpha = 0.3, color = "blue") +
  labs(
    x = "Log10 Mean Counts",
    y = "Log10 CV",
    title = "Gene Mean vs Coefficient of Variation (raw counts)"
  ) +
  theme_minimal()

p1 + p2
```



The same distribution statistics are now performed on the variance-stabilizing transform counts to visualize the data after normalization. The purpose is to reduce variance occurring from the extent of the average gene expression so that truly differential genes, regardless of large or small mean expression, can be identified.

```
vst_means <- rowMeans(assay(vsd))
vst_sd <- apply(assay(vsd), 1, sd)
vst_var <- apply(assay(vsd), 1, var)
vst_cv <- vst_sd / vst_means

df_vst <- data.frame(
  mean = vst_means,
  sd = vst_sd,
  var = vst_var,
  cv = vst_cv
)

p3 <- ggplot(df_vst, aes(x = log10(mean + 1), y = log10(var + 1))) +
  geom_point(alpha = 0.3, color = "blue") +
  labs(
    x = "Log10 Mean Counts",
    y = "Log10 Variance",
    title = "Gene Mean vs Variance (VST)"
  ) +
  theme_minimal()
```

```
p4 <- ggplot(df_vst, aes(x = log10(mean + 1), y = log10(cv + 1))) +
  geom_point(alpha = 0.3, color = "blue") +
  labs(
    x = "Log10 Mean Counts",
    y = "Log10 CV",
    title = "Gene Mean vs Coefficient of Variation (VST)"
  ) +
  theme_minimal()

(p1 + p2) / (p3 + p4)
```



The code below demonstrates another way of visualizing the effect of normalizing raw count data. Genes with average expression localized in the 20th and 80th percentile are selected to show how this normalization affects genes with both low and high expression. The subsequent plots show reduction in the technical noise (counts) by the variance-stabilizing transformation and its effective achievement of homoscedasticity.

```
percentile_a <- quantile(dat_means, 0.799)
percentile_b <- quantile(dat_means, 0.801)
percentile_c <- quantile(dat_means, 0.199)
percentile_d <- quantile(dat_means, 0.201)

low_genes <- names(dat_means[dat_means >= percentile_c & dat_means < percentile_d])
```

```
print(low_genes)
```

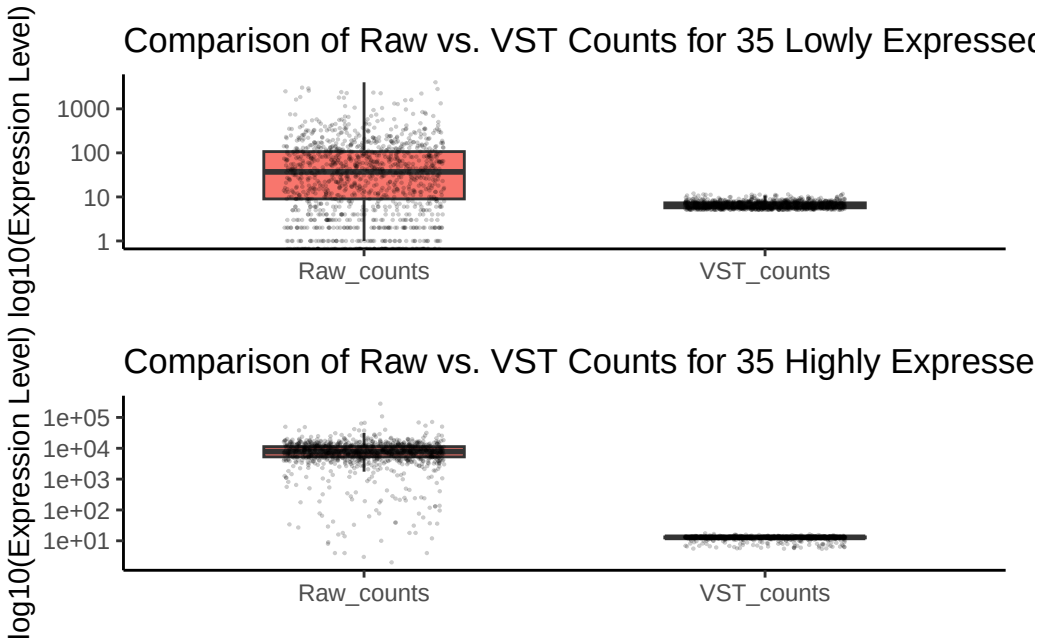
```
 [1] "AURKC"       "B3GAT2"      "CCDC148"     "CDH15"       "CIRBP-AS1"
 [6] "CLCN1"       "DCST2"       "FOXG1"       "GLB1L3"      "GLT1D1"
[11] "GPR65"       "HAPLN4"      "HCG15"       "IGFL1"       "KRTAP5-AS1"
[16] "MMP25"       "MORF4L2-AS1" "MYT1L"       "NAALADL1"    "NT5DC4"
[21] "NUTM2G"      "OTOGL"       "OVCH2"       "PCDHGA9"     "RAP2C-AS1"
[26] "RGCC"        "SLC23A1"     "SLC45A2"     "TMEM179"     "TNFRSF13C"
[31] "WAS"         "XKR5"        "ZMAT1"       "ZNF280A"
```

```
high_genes <- names(dat_means[dat_means >= percentile_a & dat_means < percentile_b])

print(high_genes)
```

```
 [1] "ANAPC11"  "BRMS1"    "BTF3L4"   "C12orf75" "CNOT11"   "COPS5"
 [7] "DBNDD2"   "DNAJC2"   "EPHX1"    "GTF2A1"   "GTF2H3"   "HPRT1"
[13] "ILK"      "KCTD10"   "KPNA1"    "MKLN1"    "MORC2"    "NUS1"
[19] "PCF11"    "PHF20"    "PREPL"    "R3HDM2"   "RPE"      "S100A9"
[25] "SEC61B"   "SGK1"     "SLC25A23" "SPNS2"    "STRBP"    "STRN"
[31] "TOM1L2"   "UBR3"     "WFS1"
```

```
df_low_sig <- data.frame(
  Gene = rep(low_genes, each = ncol(counts(dds))),
  Raw_counts = as.vector(counts(dds)[low_genes, ]),
  VST_counts = as.vector(assay(vsd)[low_genes, ])
)

df_high_sig <- data.frame(
  Gene = rep(high_genes, each = ncol(counts(dds))),
  Raw_counts = as.vector(counts(dds)[high_genes, ]),
  VST_counts = as.vector(assay(vsd)[high_genes, ])
)

df_long_low <- df_low_sig %>% pivot_longer(cols = c(Raw_counts, VST_counts),
               names_to = "Count_Type", values_to = "Count_Value")

df_long_high <- df_high_sig %>% pivot_longer(cols = c(Raw_counts, VST_counts),
               names_to = "Count_Type", values_to = "Count_Value")

p5 <- ggplot(df_long_low, aes(x = Count_Type, y = Count_Value, fill = Count_Type)) +
```

```
  geom_boxplot(outlier.shape = NA, width = 0.5) +
  geom_jitter(width = 0.2, alpha = 0.2, size = 0.1) +
  scale_y_log10() +
  labs(title = "Comparison of Raw vs. VST Counts for 35 Lowly Expressed Genes",
       x = "",
       y = "log10(Expression Level)") +
  theme_classic() +
  theme(legend.position = "none")

p6 <- ggplot(df_long_high, aes(x = Count_Type, y = Count_Value, fill = Count_Type)) +
  geom_boxplot(outlier.shape = NA, width = 0.5) +
  geom_jitter(width = 0.2, alpha = 0.2, size = 0.1) +
  scale_y_log10() +
  labs(title = "Comparison of Raw vs. VST Counts for 35 Highly Expressed Genes",
       x = "",
       y = "log10(Expression Level)") +
  theme_classic() +
  theme(legend.position = "none")

p5/p6
```

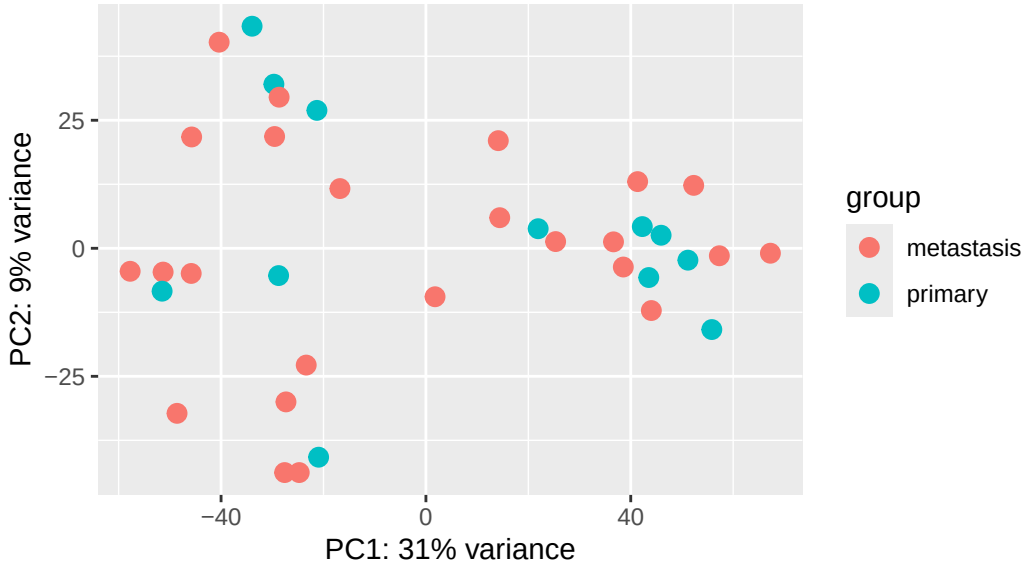


4 Principal Component Analysis

To kick off principal component analysis (PCA), a quick PCA plot is generated using the normalized counts object, the `Pathology` feature, and the top 500 genes.

```
plotPCA(vsd, intgroup = "Pathology", ntop = 500)
```

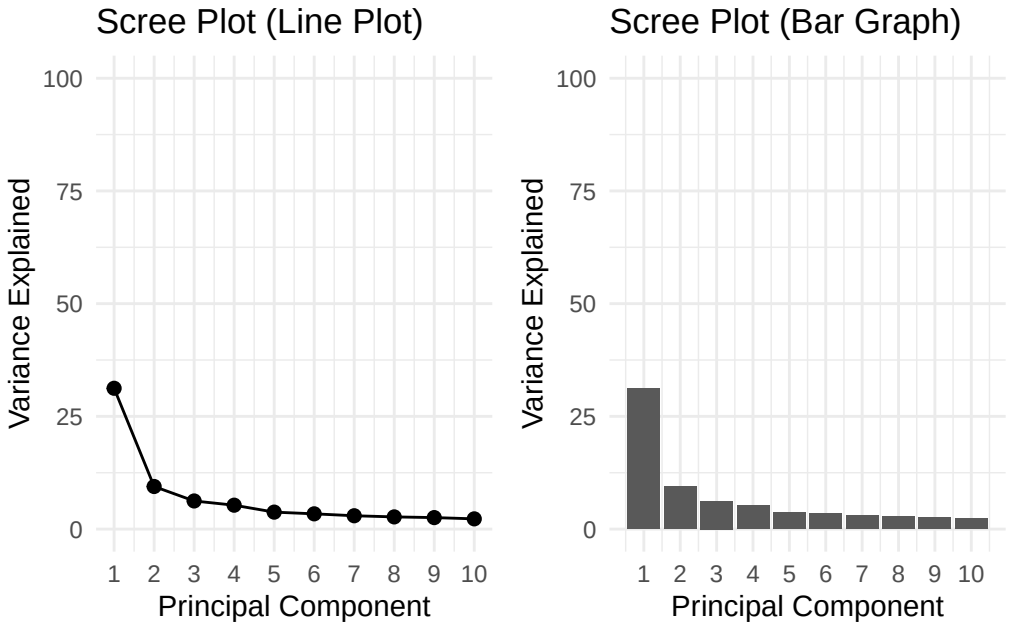
```
using ntop=500 top features by variance
```



After visualizing the top two principal components, `PC1` and `PC2`, there does not appear to be any clear separation explained by `Pathology`. This could mean pathology differences are not driving variation in the data, but higher PCs may better explain this variance. Currently, `PC1` is found to explain 31% of the variance and `PC2` explains 9%. Below is a scree plot to identify the amount of variance explained by higher PCs, and visualize where the "elbow" occurs.

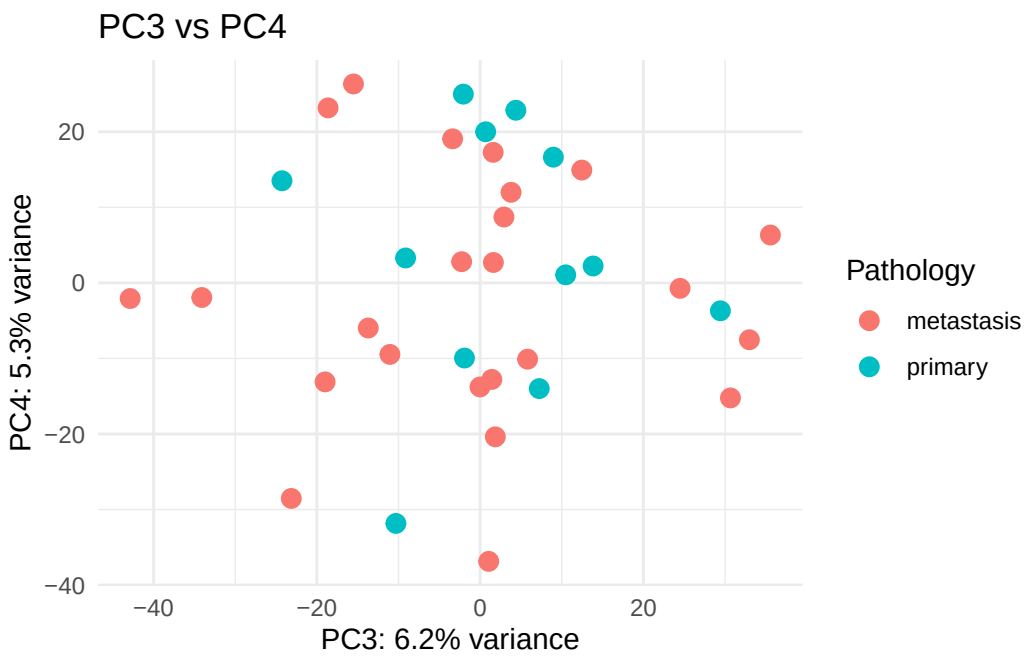
```
top_genes_var <- order(vst_var, decreasing=TRUE)[1:500]

var_mat <- assay(vsd)[top_genes_var, ]

pca_res <- prcomp(t(var_mat), center=TRUE, scale=FALSE)

explained_pca_var <- pca_res$sdev^2 / sum(pca_res$sdev^2) * 100

df_variance <- data.frame(PC=seq_len(10), var = explained_pca_var[1:10])
```

```r
p7 <- ggplot(df_variance, aes(x=PC, y=var)) +
  geom_line() +
  geom_point(size=2)+
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot (Line Plot)") +
  ylim(0, 100) +
  scale_x_continuous(breaks = 1:10) +
  theme_minimal()

p8 <- ggplot(df_variance, aes(x=PC, y=var)) +
  geom_col() +
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot (Bar Graph)") +
  ylim(0, 100) +
  scale_x_continuous(breaks = 1:10) +
  theme_minimal()

p7 + p8
```



Now, let's try plotting `PC3` and `PC4` to see if these components better explain `Pathology` impact on variance.

```r
pca_df <- data.frame(PC3 = pca_res$x[, 3], PC4 = pca_res$x[, 4], ann_subset)

p9 <- ggplot(pca_df, aes(x = PC3, y = PC4, color = Pathology)) +
  geom_point(size = 3) +
  labs(title = "PC3 vs PC4",
       x = paste0("PC3: ", round(explained_pca_var[3], 1), "% variance"),
       y = paste0("PC4: ", round(explained_pca_var[4], 1), "% variance")) +
  theme_minimal()

p9
```

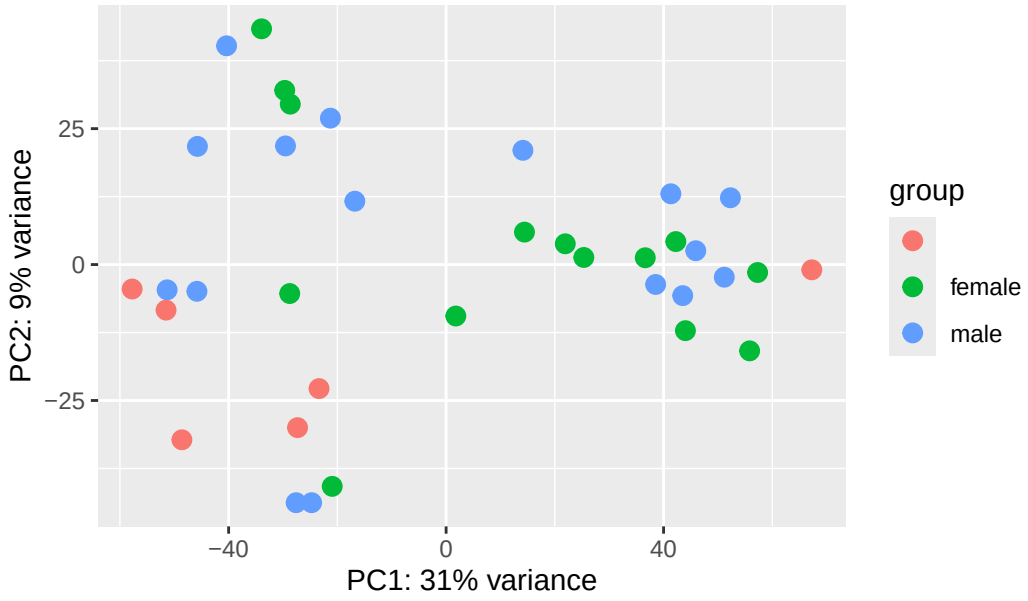


These do not show separation explained by `Pathology` either. Let's try a few other features like `Gender` and `Site_Of_Finding`.

```r
p10 <- plotPCA(vsd, intgroup = "Gender", ntop = 500)
```

```
using ntop=500 top features by variance
```

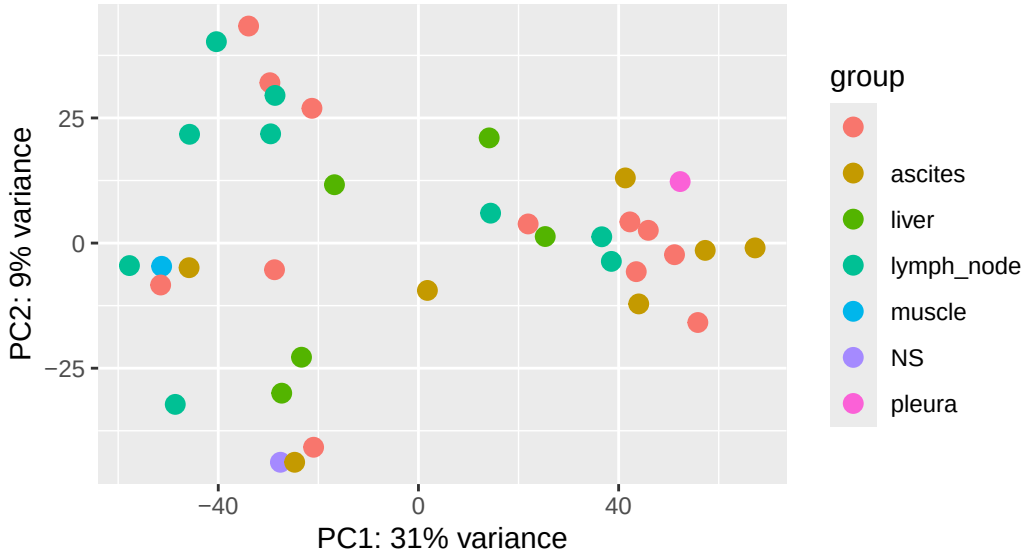
```r
p10
```




```
p11 <- plotPCA(vsd, intgroup = "Site_Of_Finding", ntop = 500)
```

```
using ntop=500 top features by variance
```

```
p11
```



The use of PCA is to capture the largest sources of variation. It does not appear that the largest sources of variation are clearly separated or explained by one of the feature variables in

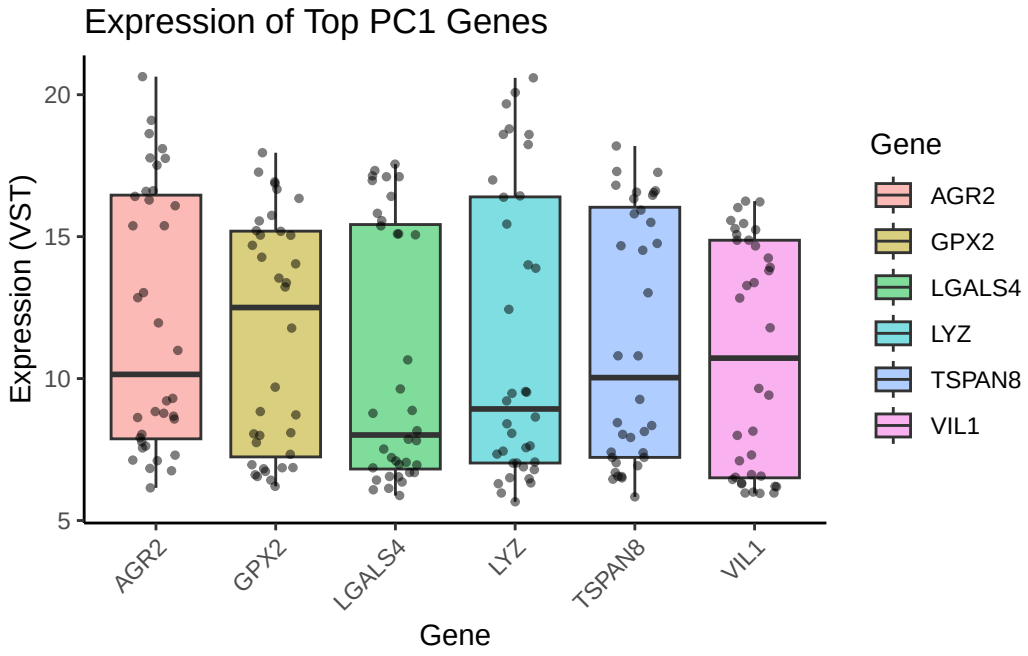
the metadata. This could be due to the variance being explained by features not captured in our subset data like unknown subtypes within pathology, batch effects, sample dates, timing, or other technical noise. Despite this, we can continue investigating the differential expression of the genes driving these principal components.

```
as.data.frame(pca_res$rotation)
```

```
pc1_genes <- pca_res$rotation[,1]
top_pc1_genes <- names(sort(pc1_genes, decreasing=TRUE)[1:6])
print(top_pc1_genes)
```

```
[1] "LYZ"    "AGR2"   "LGALS4" "TSPAN8" "GPX2"   "VIL1"
```

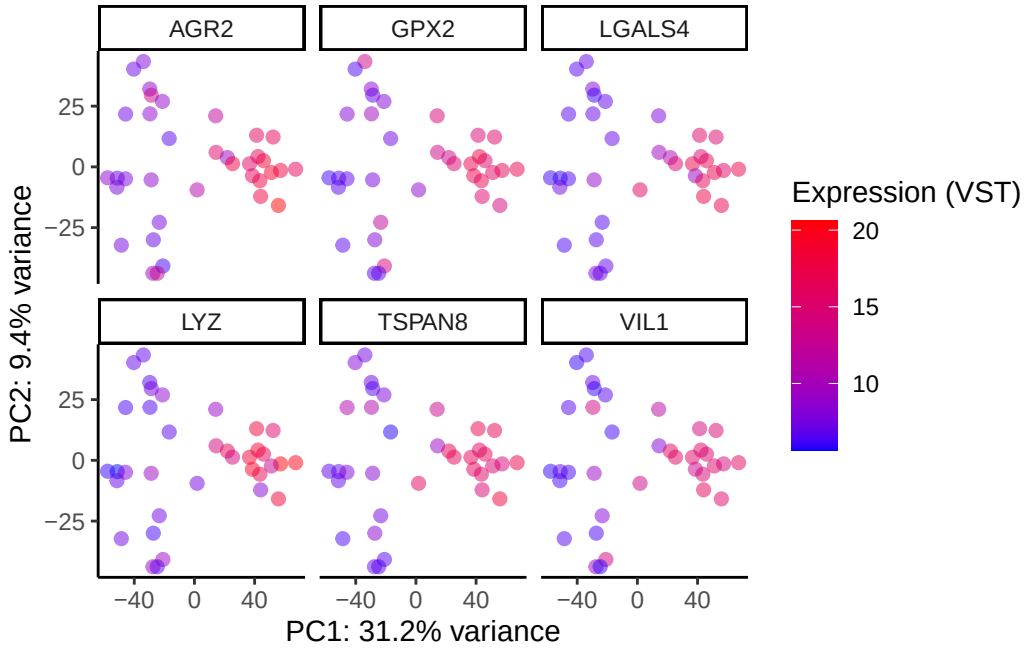
```
df_pc1_genes <- data.frame(t(assay(vsd)[top_pc1_genes, ]))

df_pc1_long <- df_pc1_genes %>%
  pivot_longer(
    cols = everything(),
    names_to = "Gene",
    values_to = "Expression"
  )

p12 <- ggplot(df_pc1_long, aes(x = Gene, y = Expression)) +
  geom_boxplot(aes(fill = Gene), alpha = 0.5) +
  geom_jitter(width = 0.2, size = 1, alpha = 0.5) +
  labs(title = "Expression of Top PC1 Genes",
       x = "Gene",
       y = "Expression (VST)") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

p12
```



The top 6 genes driving PC1 show differential expression above. Let's now see how these appear in relation to the separation shown between `PC1` and `PC2` using `pcaPlot`.

```
df_pc <- data.frame(PC1 = pca_res$x[, 1], PC2 = pca_res$x[, 2], ann_subset)

df_long_top <- data.frame(
  df_pc1_genes,
  df_pc[,c("PC1", "PC2", "Pathology")]
)

df_long_genes <- pivot_longer(
  df_long_top,
  cols = all_of(top_pc1_genes),
  names_to = "Gene",
  values_to = "Expression"
  )

p13 <- ggplot(df_long_genes, aes(x = PC1, y = PC2, color = Expression)) +
  geom_point(size = 2, alpha = 0.5) +
  scale_color_gradient(low = "blue", high = "red") +
  theme_classic() +
  labs(x = paste0("PC1: ", round(explained_pca_var[1], 1), "% variance"),
       y = paste0("PC2: ", round(explained_pca_var[2], 1), "% variance"),
       color = "Expression (VST)") +
```

```
  facet_wrap(~ Gene)

p13
```



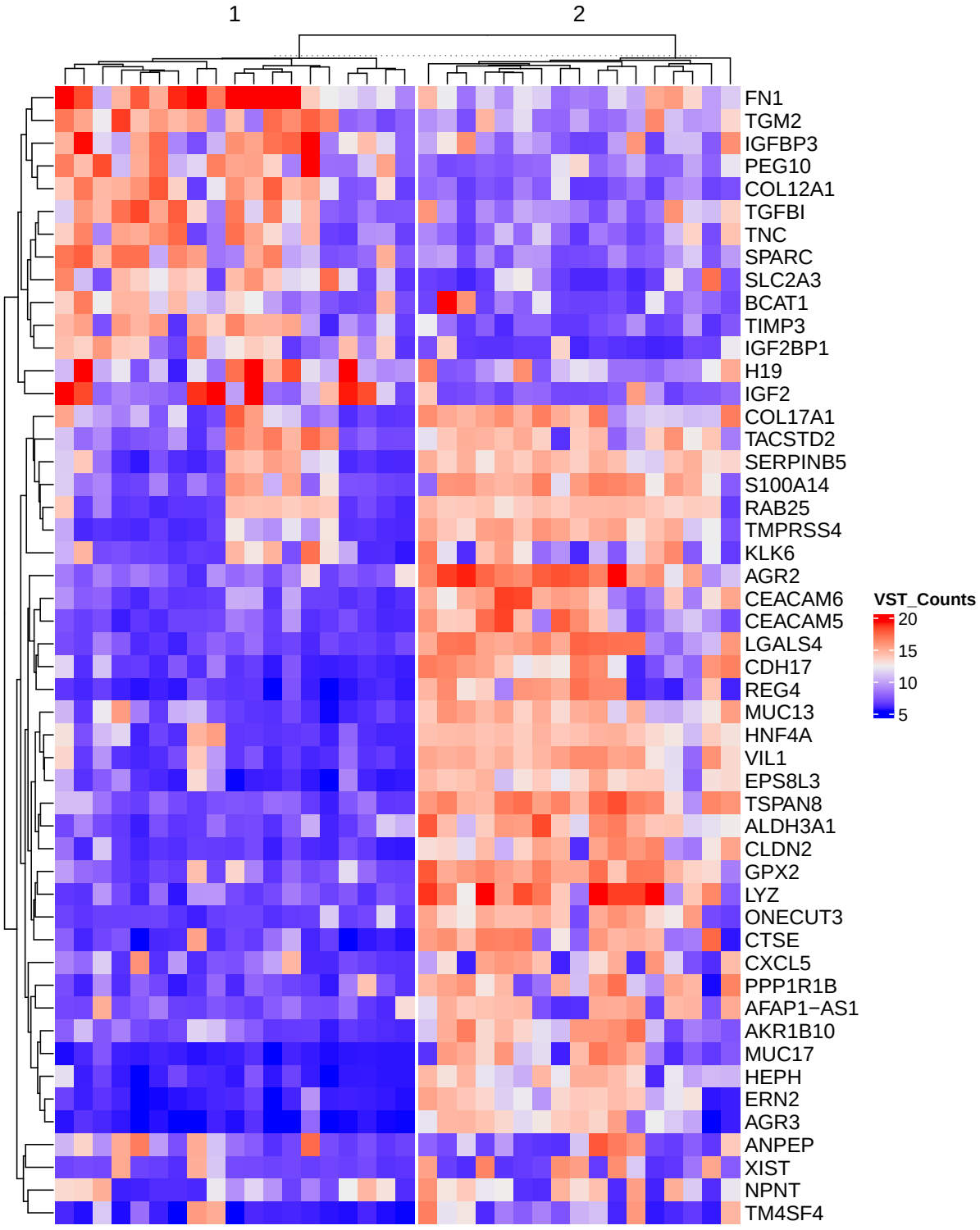
Analyzing the 6 genes plotted on `PC1` and `PC2`, it is clear that differential expression aligns with sample grouping. The sample group on the left appears to have lower expression values than the second group appearing on the right. This suggests these genes are major contributors to a biological difference between samples groups, but `Pathology`, `Gender`, and `Site_Of_Finding` do not appear to explain the variance.

5 Hierarchical Clustering

Let's continuing exploring the data and the separation seen between the two sample groupings in the genes with the most `PC1` variance. Below hierarchical clustering is used to cluster the RNA-seq samples with similar gene expression profiles.

```
top50_var <- head(order(rowVars(assay(vsd)), decreasing = TRUE), 50)
top50_var_mat <- assay(vsd)[top50_var, ]

hc <- hclust(dist(t(top50_var_mat)), method = "complete")
clusters <- cutree(hc, k = 2)
```

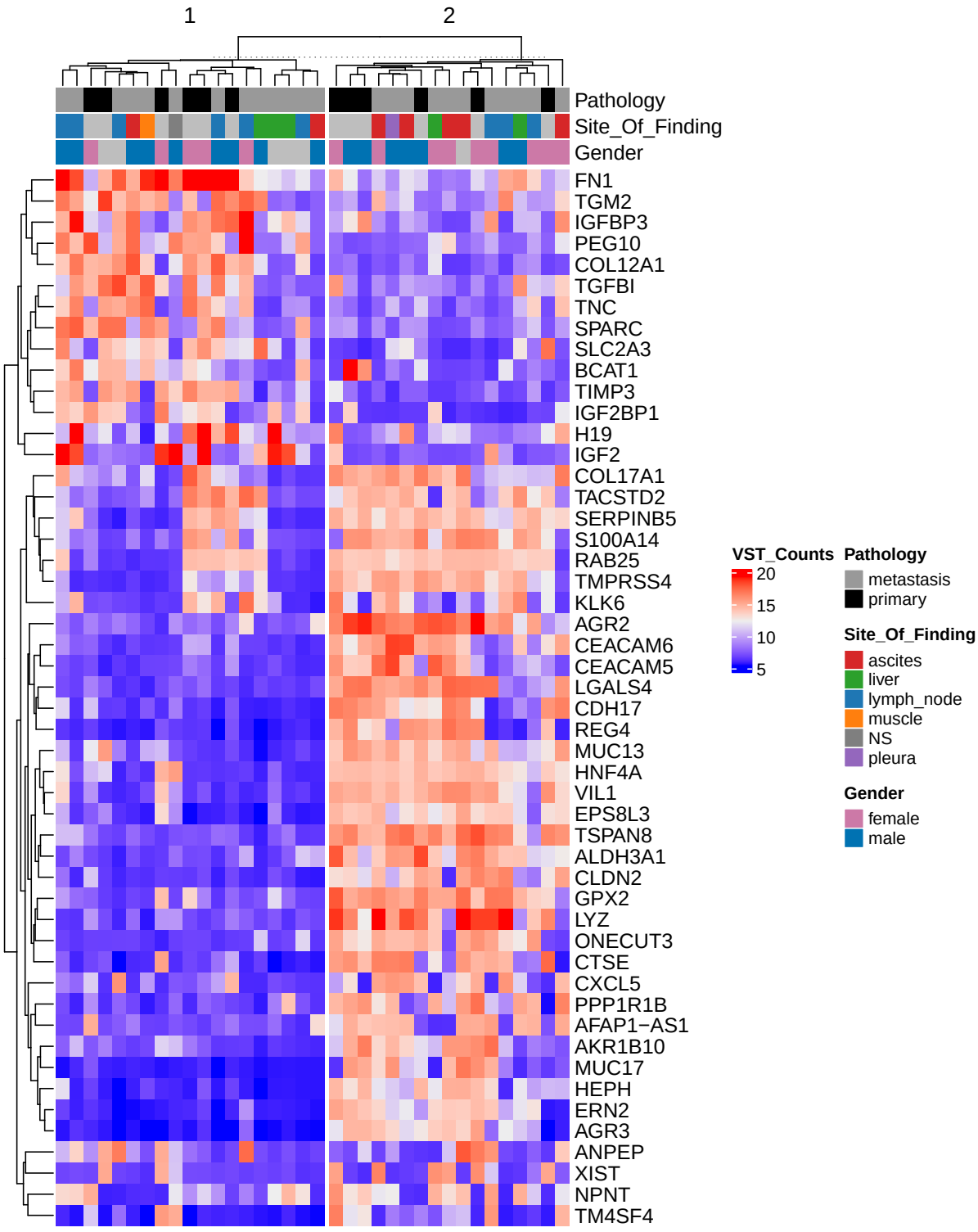
```r
p14 <- Heatmap(top50_var_mat,
        color = colorRamp2(c(min(top50_var_mat),
                             median(top50_var_mat),
                             max(top50_var_mat)),
                           c("blue", "white", "red")),
        clustering_distance_columns = "euclidean",
        clustering_method_columns = "complete",
        clustering_distance_rows = "euclidean",
        clustering_method_rows = "complete",
        show_row_names = TRUE,
        show_column_names = FALSE,
        name = "VST_Counts",
        column_split = clusters)

p14
```



The cluster above clearly shows two clusters with near opposite gene expression profiles. Let's

confirm this isn't being explained by one of the features from PCA by annotating the dendrogram.

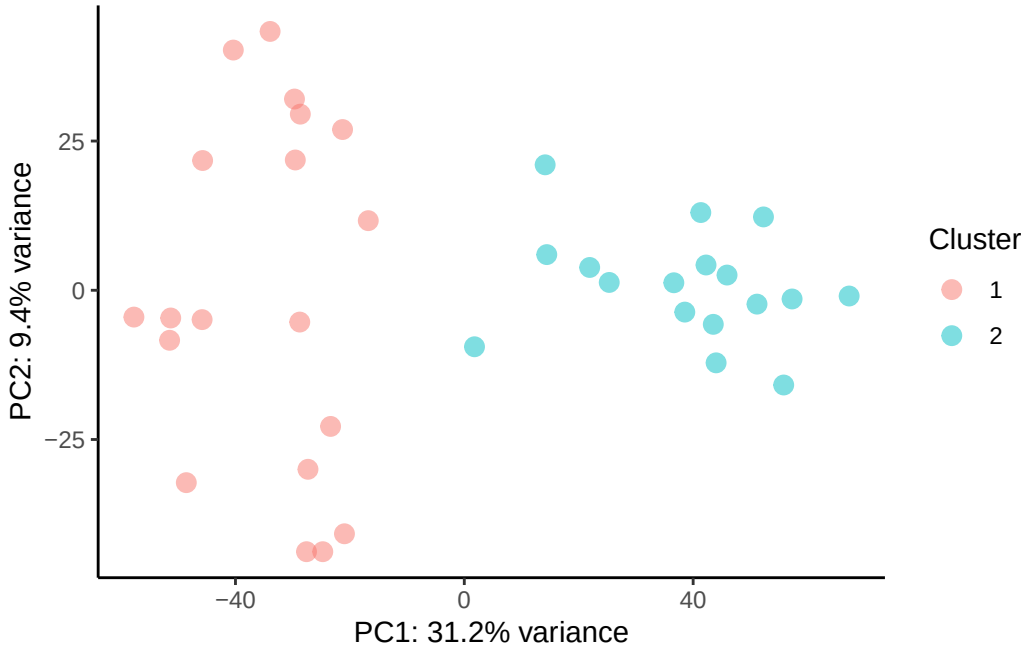
```
ann_subset_df <- ann_subset
ann_subset_df <- ann_subset_df %>% dplyr::select("Pathology", "Site_Of_Finding", "Gender")

sample_annotation_colors <- list(
  Pathology = c("primary" = "black", "metastasis" = "grey60"),
  Site_Of_Finding = c("lymph_node" = "#1F77B4",
                      "muscle" = "#FF7F0E",
                      "liver" = "#2CA02C",
                      "ascites" = "#D62728",
                      "pleura" = "#9467BD",
                      "unknown" = "#D3D3D3",
                      "NS" = "#7F7F7F"),
  Gender = c("male" = "#0072B2",
             "female" = "#CC79A7",
             "missing_info" = "#999999")
)

col_annotation <- HeatmapAnnotation(
  df = ann_subset_df,
  col = sample_annotation_colors
  )

p15 <- Heatmap(top50_var_mat,
        color = colorRamp2(c(min(top50_var_mat),
                             median(top50_var_mat),
                             max(top50_var_mat)),
                           c("blue", "white", "red")),
        clustering_distance_columns = "euclidean",
        clustering_method_columns = "complete",
        clustering_distance_rows = "euclidean",
        clustering_method_rows = "complete",
        show_row_names = TRUE,
        show_column_names = FALSE,
        name = "VST_Counts",
        column_split = clusters,
        top_annotation = col_annotation)

p15
```



There still does not appear to be a relationship between annotated cell line features selected

and the sample clusters. We need to figure out if the clusters found in hierarchical clustering relate to the sample separation in PCA. Below, genes are assigned and colored by their cluster and plotted with `PC1` and `PC2`.

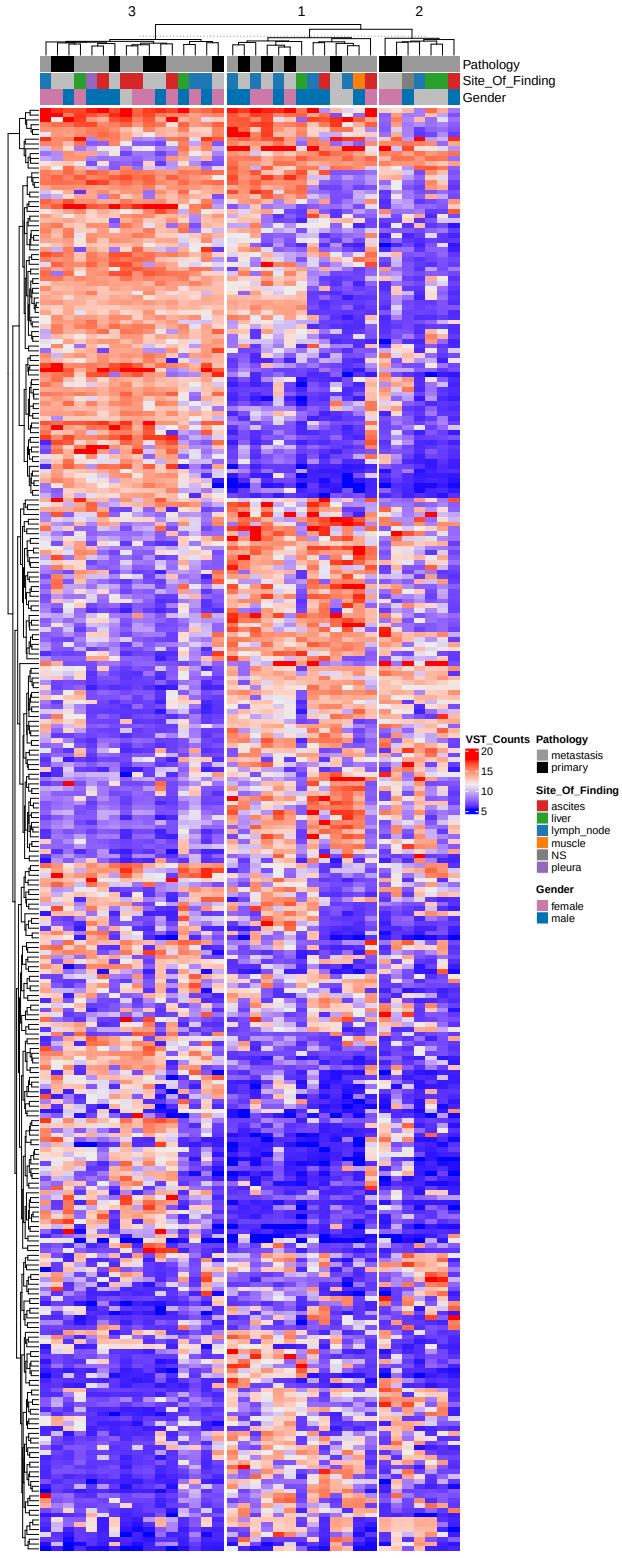
```
df_pc$Cluster2 <- as.factor(clusters)

p16 <- ggplot(df_pc, aes(x = PC1, y=PC2, color=Cluster2)) +
  geom_point(size = 3, alpha = 0.5) +
  scale_color_manual(values = hue_pal()(length(unique(clusters)))) +
  theme_classic() +
  labs(x = paste0("PC1: ", round(explained_pca_var[1], 1), "% variance"),
       y = paste0("PC2: ", round(explained_pca_var[2], 1), "% variance"),
       color = "Cluster")

p16
```



The PCA plot shows the clusters perfectly match the separation previously seen. This confirms the biological effect is not method-specific and there is a hidden variable that is biologically meaningful. But, before moving on let's confirm there aren't additional clusters that may better align with the variables we have viewed so far (`Pathology`, `Gender`, `Site_Of_Finding`).

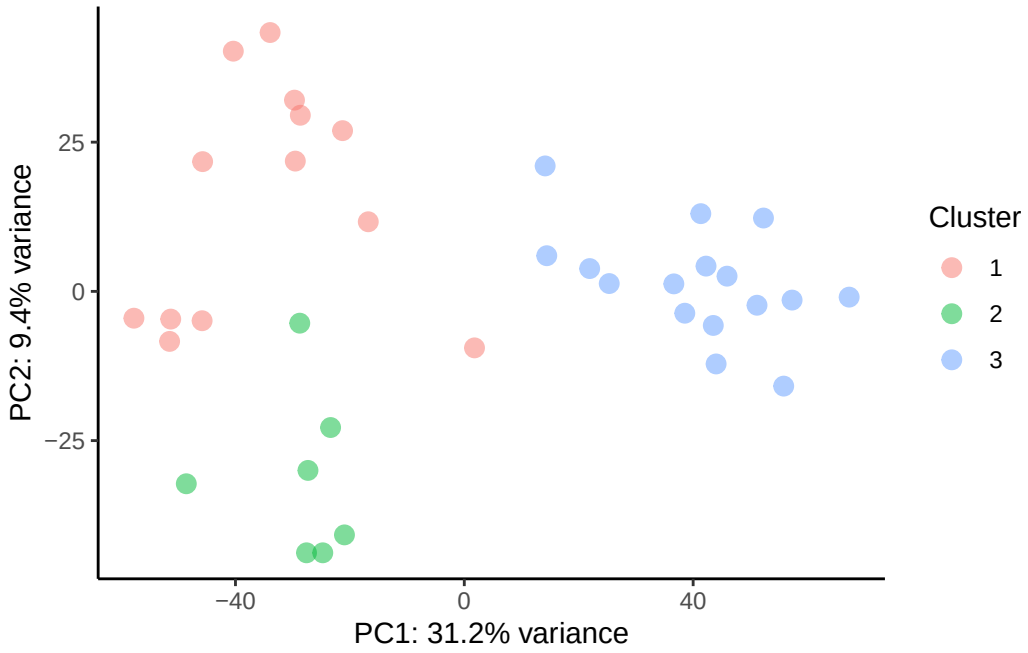
```
top300_var <- head(order(rowVars(assay(vsd)), decreasing = TRUE), 300)
top300_var_mat <- assay(vsd)[top300_var, ]
```

```r
hc <- hclust(dist(t(top300_var_mat)), method = "complete")
clusters_300 <- cutree(hc, k = 3)

p17 <- Heatmap(top300_var_mat,
        color = colorRamp2(c(min(top300_var_mat),
                             median(top300_var_mat),
                             max(top300_var_mat)),
                           c("blue", "white", "red")),
        clustering_distance_columns = "euclidean",
        clustering_method_columns = "complete",
        clustering_distance_rows = "euclidean",
        clustering_method_rows = "complete",
        show_row_names = FALSE,
        row_names_gp = gpar(fontsize = 8),
        show_column_names = FALSE,
        name = "VST_Counts",
        column_split = clusters_300,
        top_annotation = col_annotation)

p17
```


3
1
2
Pathology
Site_Of_Finding
Gender
VST_Counts
20
15
10
5
Pathology
metastasis
primary
Site_Of_Finding
ascites
liver
lymph_node
muscle
NS
pleura
Gender
female
male

The dendrogram above visualizes expression profile clusters for the top 300 genes. It is split by an additional cluster (k=3) to eliminate the possibility of another separating biological effect within the data. Let's check the PCA plot again now with three clusters on the top 300 genes.

```
df_pc$Cluster3 <- as.factor(clusters_300)

p18 <- ggplot(df_pc, aes(x = PC1, y=PC2, color=Cluster3)) +
  geom_point(size = 3, alpha = 0.5) +
  scale_color_manual(values = hue_pal()(length(unique(clusters_300)))) +
  theme_classic() +
  labs(x = paste0("PC1: ", round(explained_pca_var[1], 1), "% variance"),
       y = paste0("PC2: ", round(explained_pca_var[2], 1), "% variance"),
       color = "Cluster")

p18
```



It appears there may be an explainable variation between `Cluster 1` and `Cluster 2` by `PC2`. To finalize our analysis and uncover the biological explanation, the last steps will focus on differential expression analysis to identify how the top differentially expressed genes (DEGs) relate to the gene expression clusters separating samples.

6 Differential Expression Analysis

This uses the DESeq function to run the entire DESeq2 pipeline, with the design formula `Cluster`, so we can view the DEGs between the two clusters.

```
ann_subset$Cluster <- factor(clusters)

dds <- DESeqDataSetFromMatrix(countData = dat_filtered,
                              colData = ann_subset,
                              design = ~ Cluster)

dds$Cluster <- relevel(dds$Cluster, ref=2)

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 3460 genes
-- DESeq argument 'minReplicatesForReplace' = 7
-- original counts are preserved in counts(dds)

estimating dispersions

fitting model and testing
```

The quality control and dispersion estimation step verifies proper normalization, models biological variability, and ensures reliable statistical testing in RNA-seq data. This is similar to the steps performed to show plots `p1-p4`. It confirms normalization by comparing raw and scaled counts, estimates gene-specific dispersion to capture variability, and fits a mean–variance curve across all genes. Visual assessments show that most genes follow expected dispersion patterns, validating DESeq2's assumptions and ensuring accurate detection of differential expression.

```
# Examine size factors
print(sizeFactors(dds))
```

```
  RERFGC1B_STOMACH     HS746T_STOMACH       LMSU_STOMACH      ECC10_STOMACH
         1.1283944          1.2118026          0.7265959          1.1022305
 TGBC11TKB_STOMACH     SNU520_STOMACH        GSS_STOMACH     SNU620_STOMACH
         0.5599165          1.1096638          1.0397526          1.2907148
     ECC12_STOMACH        GSU_STOMACH       FU97_STOMACH       GCIY_STOMACH
         1.3423914          0.8137822          1.0246700          0.8626674
    SH10TC_STOMACH       MKN1_STOMACH      MKN74_STOMACH       KE39_STOMACH
         0.7394471          1.2084879          1.1506394          1.0488847
     HGC27_STOMACH      HUG1N_STOMACH      NUGC4_STOMACH      SNU16_STOMACH
         1.0307862          1.0865675          0.6995609          0.9999852
    NCIN87_STOMACH      OCUM1_STOMACH       SNU5_STOMACH     SNU601_STOMACH
         1.9752998          0.8714619          1.1871115          1.1031546
    SNU668_STOMACH NCCSTCK140_STOMACH     SNU719_STOMACH     SNU216_STOMACH
         1.0191102          1.4996831          0.8142842          1.1215716
     NUGC2_STOMACH       SNU1_STOMACH        AGS_STOMACH    KATOIII_STOMACH
         1.2361933          1.2490911          1.3141696          0.8588707
     NUGC3_STOMACH       IM95_STOMACH       MKN7_STOMACH      MKN45_STOMACH
         0.8291967          1.0021351          0.7267980          1.1471989
```

```
# Check raw versus normalized sample expression sums
print(colSums(counts(dds)))
```

```
  RERFGC1B_STOMACH     HS746T_STOMACH       LMSU_STOMACH      ECC10_STOMACH
         120307983          144552528          102211244          117505083
 TGBC11TKB_STOMACH     SNU520_STOMACH        GSS_STOMACH     SNU620_STOMACH
          59857475          142917351          122807432          158180003
     ECC12_STOMACH        GSU_STOMACH       FU97_STOMACH       GCIY_STOMACH
         154905130          117807717          142715890           91173431
    SH10TC_STOMACH       MKN1_STOMACH      MKN74_STOMACH       KE39_STOMACH
          98976670          141740971          127264584          125805442
     HGC27_STOMACH      HUG1N_STOMACH      NUGC4_STOMACH      SNU16_STOMACH
```

```
      129740410       139441903        93958630       115983065
 NCIN87_STOMACH    OCUM1_STOMACH     SNU5_STOMACH   SNU601_STOMACH
      223976799       115371268       153925786       140341699
 SNU668_STOMACH NCCSTCK140_STOMACH   SNU719_STOMACH   SNU216_STOMACH
      135916810       172946346        95291984       137527609
  NUGC2_STOMACH     SNU1_STOMACH      AGS_STOMACH  KATOIII_STOMACH
      147256904       145857876       163915452       109659822
  NUGC3_STOMACH     IM95_STOMACH     MKN7_STOMACH   MKN45_STOMACH
      112298141       118361777       100623536       131488621
```

```r
print(colSums(counts(dds, normalized=TRUE)))
```

```
  RERFGC1B_STOMACH     HS746T_STOMACH       LMSU_STOMACH      ECC10_STOMACH
         106618736          119287196          140671382          106606633
 TGBC11TKB_STOMACH     SNU520_STOMACH       GSS_STOMACH     SNU620_STOMACH
         106904282          128793381          118112163          122552254
     ECC12_STOMACH       GSU_STOMACH      FU97_STOMACH      GCIY_STOMACH
         115394903          144765661          139279850          105687815
    SH10TC_STOMACH      MKN1_STOMACH     MKN74_STOMACH      KE39_STOMACH
         133852262          117287865          110603358          119942111
     HGC27_STOMACH     HUG1N_STOMACH     NUGC4_STOMACH     SNU16_STOMACH
         125865485          128332478          134310865          115984786
    NCIN87_STOMACH     OCUM1_STOMACH     SNU5_STOMACH   SNU601_STOMACH
         113388761          132388200          129664133          127218520
    SNU668_STOMACH NCCSTCK140_STOMACH   SNU719_STOMACH   SNU216_STOMACH
         133368120          115321930          117025462          122620441
     NUGC2_STOMACH     SNU1_STOMACH      AGS_STOMACH  KATOIII_STOMACH
         119121260          116771211          124729297          127679078
     NUGC3_STOMACH     IM95_STOMACH     MKN7_STOMACH   MKN45_STOMACH
         135430039          118109605          138447727          114617111
```

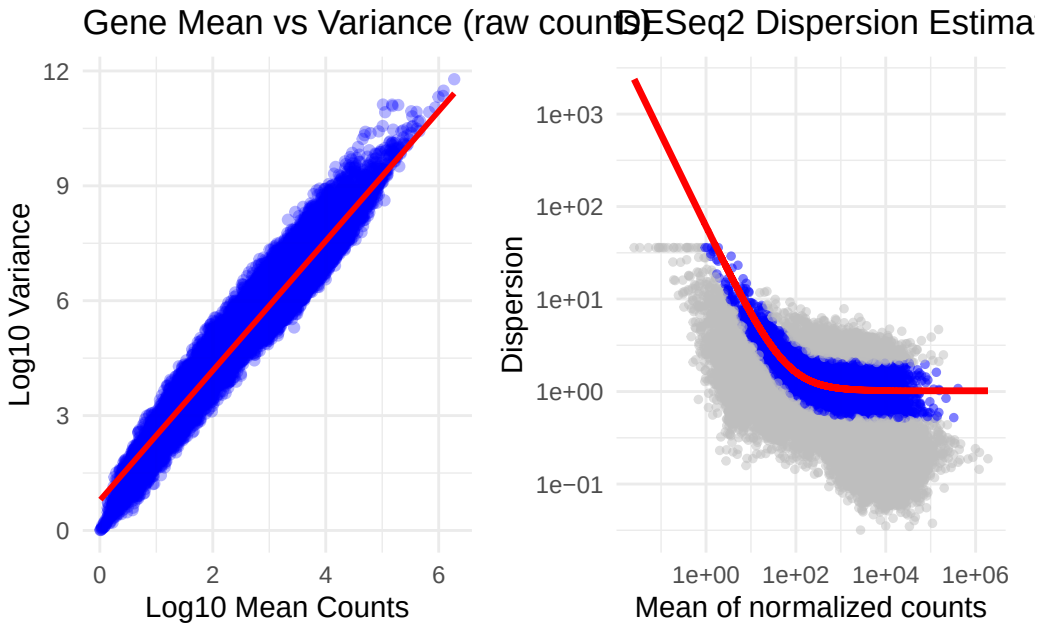
```r
mean_counts <- rowMeans(counts(dds))
variance_counts <- apply(counts(dds), 1, var)

# Visualize gene-wise dispersion for QC
df_dispersion <- data.frame(
  mean = mcols(dds)$baseMean,
  var = mcols(dds)$baseVar,
  disp = mcols(dds)$dispersion,
  fitted = mcols(dds)$dispFit
)
```

```r
df_dispersion <- df_dispersion[complete.cases(df_dispersion) & df_dispersion$mean > 0, ]

# Calculate reasonable dispersion estimates for visualziation
df_dispersion$ratio <- df_dispersion$disp / df_dispersion$fitted
df_dispersion$reasonable <- abs(log2(df_dispersion$ratio)) < 1

p19 <- ggplot(df_dispersion, aes(x = log10(mean + 1), y = log10(var + 1))) +
  geom_point(alpha = 0.3, color = "blue") +
  geom_smooth(method = "lm", se = TRUE, color = "red", fill = "pink") +
  labs(
    x = "Log10 Mean Counts",
    y = "Log10 Variance",
    title = "Gene Mean vs Variance (raw counts)"
  ) +
  theme_minimal()

# Expected value of dispersion for genes of given mean strength
p20 <- ggplot(df_dispersion, aes(x = mean, y = disp)) +
  geom_point(aes(color = reasonable), alpha = 0.5, size = 1) +
  geom_line(aes(y = fitted), color = "red", size = 1.2) +
  scale_x_log10() +
  scale_y_log10() +
  scale_color_manual(values = c("TRUE" = "blue", "FALSE" = "gray"),
                     name = "Reasonable dispersion",
                     labels = c("TRUE" = "Within 2-fold of expected", "FALSE" = "Outlier")) +
  labs(
    x = "Mean of normalized counts",
    y = "Dispersion",
    title = "DESeq2 Dispersion Estimates"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

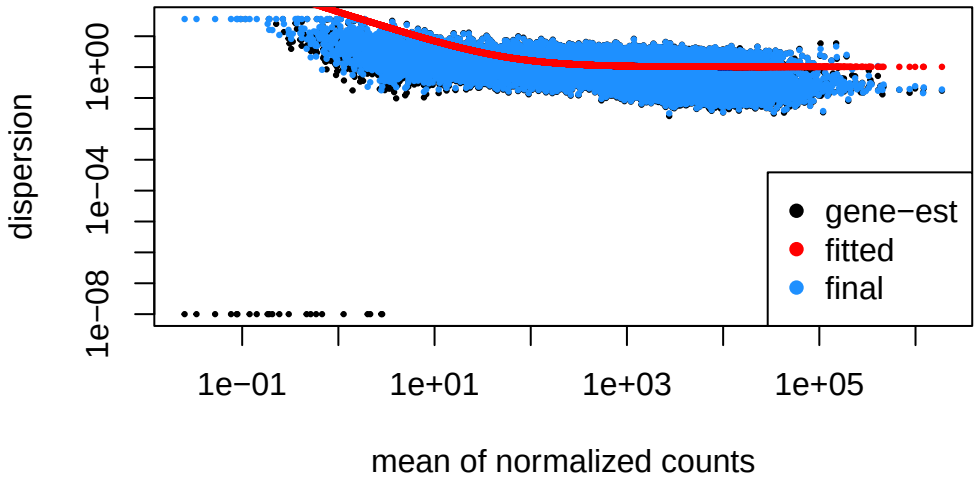
```r
p19 + p20
```

```
`geom_smooth()` using formula = 'y ~ x'
```



The `plotDispEsts` function shows the raw mean–variance relationship across genes, the fitted dispersion curve, and how individual gene estimates align with the expected trend. This plot confirms that DESeq2 has successfully modeled biological variability and that most genes fall within the expected range, supporting reliable differential expression analysis.

```
# Data dispersion estimates to assess model fit
p21 <- plotDispEsts(dds)
```



Now that we know the data has been reliably normalized, let's run the differential expression analysis.

```
res <- results(dds)
print(res)
```

```
log2 fold change (MLE): Cluster 1 vs 2
Wald test p-value: Cluster 1 vs 2
DataFrame with 17177 rows and 6 columns
          baseMean log2FoldChange     lfcSE      stat      pvalue        padj
         <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
A1BG      302.1398       4.664006  0.582440  8.007697 1.16876e-15 1.01660e-13
A1BG-AS1  116.5664       4.078728  0.515944  7.905371 2.67138e-15 2.14409e-13
A1CF     1488.2228      -0.302297  1.067049 -0.283302 7.76946e-01 8.65030e-01
A2M       240.1205       3.054514  0.742260  4.115151 3.86926e-05 3.22613e-04
A2M-AS1    32.5071       1.648145  0.715089  2.304812 2.11771e-02 6.50345e-02
...            ...            ...       ...       ...         ...         ...
ZYG11A     566.532      1.2887135  0.886849 1.4531378   0.1461855    0.280734
ZYG11B    5197.181      0.3500365  0.168769 2.0740535   0.0380743    0.102310
ZYX      13022.432      0.0152083  0.376295 0.0404161   0.9677614    0.982578
ZZEF1    11141.967      0.0309224  0.210134 0.1471558   0.8830091    0.933902
ZZZ3      6813.911      0.0163761  0.145248 0.1127457   0.9102321    0.949536
```

```
summary(res)
```

```
out of 17176 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 4332, 25%
LFC < 0 (down)     : 2022, 12%
outliers [1]       : 0, 0%
low counts [2]     : 1, 0.0058%
(mean count < 0)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

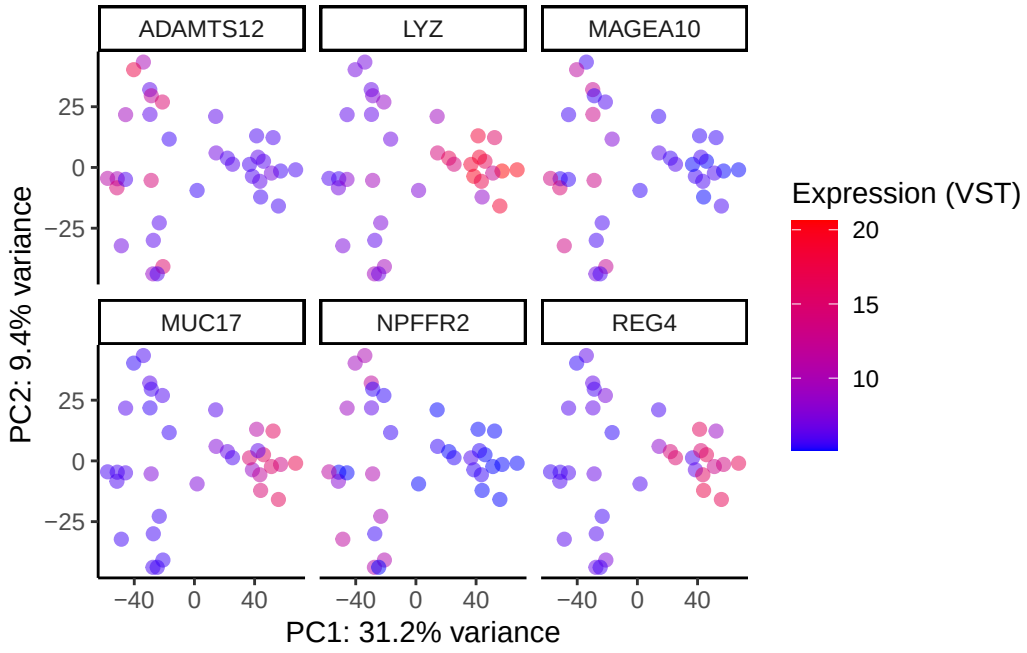
The results summary show that there is almost a 2:1 activation rate, 25% upregulated and 12% downregulated, among the significant differentially expressed genes. Quality indicators show data preprocessing was successful as there were 0% `outliers`, 0.0058% `low counts`, and 37% of genes were considered significant. This strongly suggests transcriptional reprogramming is occurring between the hidden condition. Let's investigate the genes with the largest expression change.

```r
res_ordered <- res %>%
  as.data.frame() %>%
  tibble::rownames_to_column(var = "gene_symbol") %>%
  filter(padj < 0.001) %>%
  arrange(desc(abs(log2FoldChange)))

(head(res_ordered, 10))
```

```
   gene_symbol    baseMean log2FoldChange     lfcSE       stat       pvalue
1         MUC17  11359.5498     -10.857572 0.7538793 -14.402268 5.007291e-47
2           LYZ 152113.0761     -10.840987 0.7007898 -15.469671 5.559069e-54
3          REG4  17033.3516     -10.529091 0.7399840 -14.228809 6.070693e-46
4         NPSR1    933.0308      -9.811104 0.9881525  -9.928734 3.121927e-23
5        LGALS4  34933.3461      -9.803315 0.5487254 -17.865611 2.185250e-71
6       MAGEA10   1995.8704       9.650567 0.9319899  10.354797 3.980286e-25
7      ADAMTS12   7777.6749       9.572806 0.8458044  11.317991 1.068938e-29
8          AGR3   5246.3881      -9.515338 0.8043432 -11.829948 2.733120e-32
9        NPFFR2    444.7470       9.484651 1.0712202   8.854063 8.438962e-19
10       MAGEA4   6057.0349       9.419117 0.8694758  10.833098 2.398977e-27
            padj
1  1.720105e-43
2  4.774128e-50
3  1.737837e-42
4  1.051416e-20
5  3.753386e-67
6  1.553759e-22
7  6.800027e-27
8  3.129605e-29
9  1.380453e-16
10 1.144578e-24
```

```r
degs <- c("MUC17", "LYZ", "REG4", "MAGEA10", "ADAMTS12", "NPFFR2")
degs_vsd <- data.frame(t(assay(vsd)[degs, ]))

df_degs_long <- data.frame(
  degs_vsd,
  df_pc[,c("PC1", "PC2")]
)

df_degs_pivot <- pivot_longer(
  df_degs_long,
```

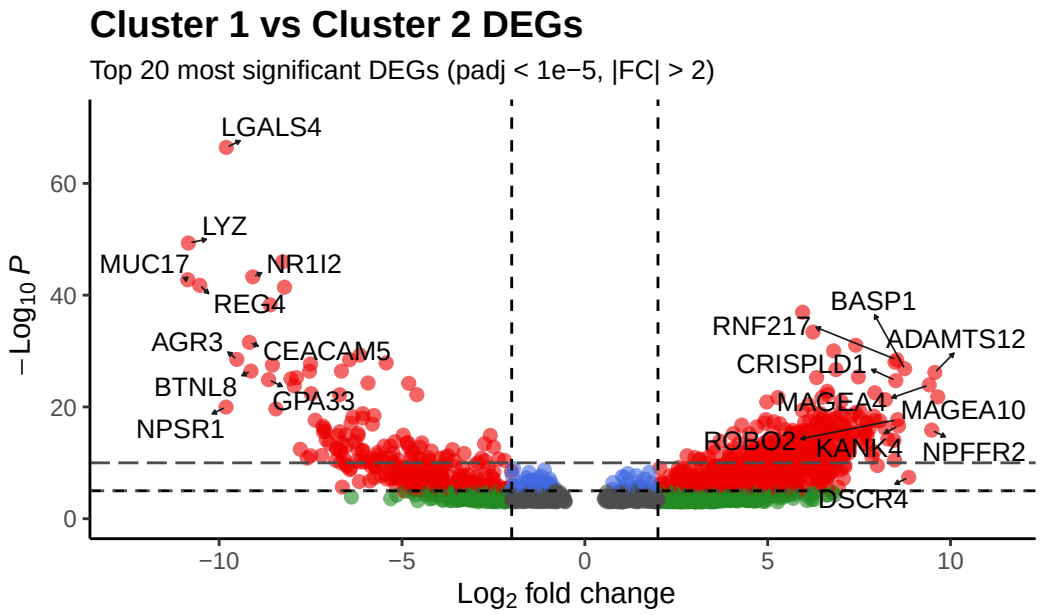
```
  cols = all_of(degs),
  names_to = "Gene",
  values_to = "Expression"
)

p22 <- ggplot(df_degs_pivot, aes(x = PC1, y = PC2, color = Expression)) +
  geom_point(size = 2, alpha = 0.5) +
  scale_color_gradient(low = "blue", high = "red") +
  theme_classic() +
  labs(x = paste0("PC1: ", round(explained_pca_var[1], 1), "% variance"),
       y = paste0("PC2: ", round(explained_pca_var[2], 1), "% variance"),
       color = "Expression (VST)") +
  facet_wrap(~ Gene)

p22
```



These top 6 genes perfectly align with the PC1 separation and the cluster separation shown from hierarchical clustering. This confirms these are major contributors to the biological variation between sample clusters. `LYZ`, `MUC17`, and `REG4` all are upregulated in cluster 2. `ADAMTS12`, `MAGEA10`, and `NPFFR2` all downregulate in cluster 2.

```
top_degs_up <- res_ordered %>%
  arrange(desc(log2FoldChange)) %>%
```

```r
  head(10)

top_degs_down <- res_ordered %>%
  arrange(log2FoldChange) %>%
  head(10)

label_genes <- c(top_degs_up$gene_symbol, top_degs_down$gene_symbol)

p23 <- EnhancedVolcano(res_ordered,
                x = "log2FoldChange",
                y = "padj",
                lab = res_ordered$gene_symbol,
                selectLab = label_genes,
                pCutoff = 0.00001,
                FCcutoff = 2,
                labSize = 3.5,
                title = "Cluster 1 vs Cluster 2 DEGs",
                subtitle = "Top 20 most significant DEGs (padj < 1e-5, |FC| > 2)",
                caption = "",
                pointSize = 2,
                colAlpha = 0.6,
                drawConnectors = TRUE,
                widthConnectors = 0.3,
                max.overlaps = 20,
                col = c("grey30", "forestgreen", "royalblue", "red2"),
                colCustom = NULL,
                cutoffLineType = "dashed",
                cutoffLineCol = "black",
                cutoffLineWidth = 0.5,
                hline = c(1e-5, 1e-10),
                hlineCol = c("grey30", "grey30"),
                hlineType = c("dotted", "longdash"),
                hlineWidth = c(0.5, 0.5)
                ) +
  theme_classic() +
  theme(legend.position = "none",
        plot.title = element_text(face = "bold", size = 14),
        plot.subtitle = element_text(size = 10))

p23
```



The volcano plot is an excellent visualization for identifying highly significant genes. The points in grey are considered non-significant genes. Green points are those genes with only significant `log2FoldChange`. Blue points are those with only significant `p-value`. Red signals genes that are highly significant in both metrics, and those furthest from the dashed lines are the most extreme up- and downregulated genes.

```
print(label_genes)
```

```
 [1] "MAGEA10"   "ADAMTS12" "NPFFR2"   "MAGEA4"   "DSCR4"    "BASP1"
 [7] "KANK4"     "ROBO2"    "RNF217"   "CRISPLD1" "MUC17"    "LYZ"
[13] "REG4"      "NPSR1"    "LGALS4"   "AGR3"     "CEACAM5"  "BTNL8"
[19] "NR1I2"     "GPA33"
```

As mentioned previously, I'm interested in investigating lncRNAs with significant differences in expression between sample clusters to identify their biological importance in stomach cancer cell lines. Let's view all lncRNA genes in a volcano plot and label the top 10 DEGs.

```
res_annotated <- res_ordered %>%
  left_join(coding_and_lncrna, by = c("gene_symbol" = "hgnc_symbol"))

lnc_RNA_res <- res_annotated %>%
  filter(gene_biotype == "lncRNA") %>%
  arrange(desc(abs(log2FoldChange)))
```

```r
lnc_RNA_counts <- sum(res_annotated$gene_biotype == "lncRNA")
print(lnc_RNA_counts)
```

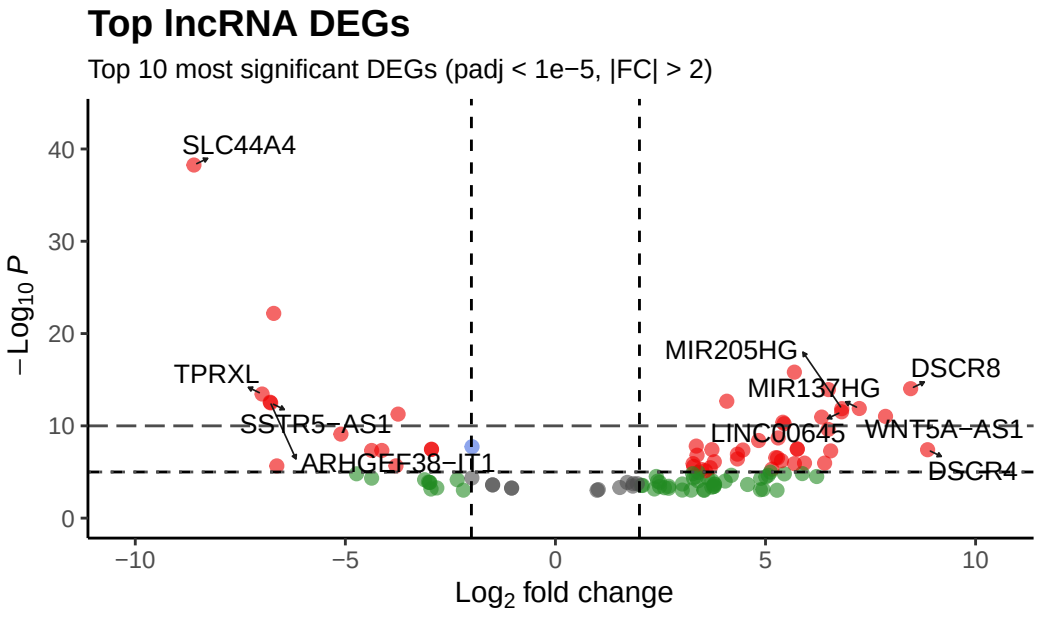
```
[1] 113
```

```r
top_lncRNA_degs <- head(lnc_RNA_res, 10)
print(top_lncRNA_degs)
```

```
    gene_symbol   baseMean log2FoldChange     lfcSE       stat       pvalue
1          DSCR4   83.83795       8.857234 1.4677331   6.034635 1.593219e-09
2       SLC44A4 6613.78826      -8.603695 0.6310943 -13.632979 2.549573e-42
3          DSCR8  226.18287       8.454194 1.0158517   8.322272 8.629290e-17
4     WNT5A-AS1   51.65016       7.856106 1.0661060   7.368973 1.719470e-13
5       MIR137HG  442.02009       7.231794 0.9451259   7.651672 1.983824e-14
6          TPRXL  600.50266      -6.981318 0.8568924  -8.147251 3.722889e-16
7       MIR205HG 1503.57478       6.816108 0.8915813   7.644965 2.090021e-14
8     LINC00645   12.13154       6.809924 0.9040913   7.532341 4.983875e-14
9  ARHGEF38-IT1   15.66870      -6.788310 0.8625677  -7.869887 3.549607e-15
10    SSTR5-AS1  877.46914      -6.777205 0.8637307  -7.846433 4.280374e-15
           padj ensembl_gene_id gene_biotype
1  3.870599e-08 ENSG00000184029       lncRNA
2  5.473934e-39 ENSG00000203463       lncRNA
3  9.380803e-15 ENSG00000198054       lncRNA
4  9.405609e-12 ENSG00000244586       lncRNA
5  1.320704e-12 ENSG00000225206       lncRNA
6  3.572309e-14 ENSG00000292995       lncRNA
7  1.375410e-12 ENSG00000230937       lncRNA
8  3.024842e-12 ENSG00000258548       lncRNA
9  2.771275e-13 ENSG00000249885       lncRNA
10 3.267542e-13 ENSG00000261713       lncRNA
```

```r
p24 <- EnhancedVolcano(lnc_RNA_res,
                x = "log2FoldChange",
                y = "padj",
                lab = lnc_RNA_res$gene_symbol,
                selectLab = top_lncRNA_degs$gene_symbol,
                pCutoff = 0.00001,
                FCcutoff = 2,
                labSize = 3.5,
                title = "Top lncRNA DEGs",
```

```
                subtitle = "Top 10 most significant DEGs (padj < 1e-5, |FC| > 2)",
                caption = "",
                pointSize = 2,
                colAlpha = 0.6,
                drawConnectors = TRUE,
                widthConnectors = 0.3,
                max.overlaps = 20,
                col = c("grey30", "forestgreen", "royalblue", "red2"),
                colCustom = NULL,
                cutoffLineType = "dashed",
                cutoffLineCol = "black",
                cutoffLineWidth = 0.5,
                hline = c(1e-5, 1e-10),
                hlineCol = c("grey30", "grey30"),
                hlineType = c("dotted", "longdash"),
                hlineWidth = c(0.5, 0.5)
                ) +
  theme_classic() +
  theme(legend.position = "none",
        plot.title = element_text(face = "bold", size = 14),
        plot.subtitle = element_text(size = 10))

p24
```



7 Interpretation

Comprehensive differential expression and clustering analyses identify two biologically distinct subtypes of gastric cancer cell lines.

The first represents a differentiated epithelial–intestinal program, marked by elevated expression of secretory mucins (MUC17, MUC5AC, REG4), brush-border genes, and adhesion molecules. This suggests maintenance of intestinal differentiation and epithelial function.

The second exhibits a dedifferentiated, mesenchymal-like state, enriched for cancer-testis antigens (MAGEA4/10), extracellular matrix remodelers (ADAMTS12), WNT-associated long noncoding RNAs (WNT5A-AS1, LINC00645), and neural guidance factors, consistent with epithelial–mesenchymal transition, immune evasion, and invasive potential.

This provides a biologically sound framework for therapeutic exploration because while epithelial-differentiated tumors may benefit from epithelial-targeted approaches, dedifferentiated tumors may be more applicable to immunotherapies directed at cancer-testis antigens or inhibitors of WNT/EMT signaling.

Therapeutic exploration may based on these subtype-defining DEGs and lncRNAs include drug discovery, prognostic biomarkers, and proteomics. They can guide subtype-specific therapeutic targeting (e.g., WNT/EMT or CT antigens), inform predictive biomarker panels and liquid biopsy assays, and support integrative proteogenomics to uncover protein-level mechanisms. They also provide opportunities for immune profiling, regulatory network modeling, and single-cell validation to refine cell-of-origin hypotheses.