RNASeq

m2407447

```r
# Load the required libraries
library("DESeq2")
library("ggplot2")

# Load the count data
counts <- read.csv("C:/Users/Asus/OneDrive/Desktop/RStudio assessment/RNAseq-assessment/exercise1_counts
                   row.names = 1)

# Load the sample description
sample_description <- read.table("C:/Users/Asus/OneDrive/Desktop/RStudio assessment/RNAseq-assessment/ex
                                 header = TRUE, sep = "\t", row.names = 1)

# Set rownames of sample_description to match colnames of counts
rownames(sample_description) <- colnames(counts)

# Check alignment between colnames and rownames
all(colnames(counts) == rownames(sample_description))
```

```
## [1] TRUE
```

```r
# Fix missing 'condition' values
sample_description[c("control_FFa1.bam", "control_FFa2.bam", "control_FFa3.bam"), "condition"] <- "conti
sample_description[c("mutant_KOa1.bam", "mutant_KOa2.bam", "mutant_KOa3.bam"), "condition"] <- "KOa"
sample_description[c("mutant_KOb1.bam", "mutant_KOb2.bam", "mutant_KOb3.bam"), "condition"] <- "KOb"

# Ensure 'condition' column is a factor
sample_description$condition <- factor(sample_description$condition, levels = c("control", "KOa", "KOb"

# Construct DESeqDataSet
dds <- DESeqDataSetFromMatrix(countData = counts,
                              colData = sample_description,
                              design = ~ condition)

# Inspect the DESeqDataSet
dds
```

```
## class: DESeqDataSet
## dim: 26301 9
## metadata(1): version
## assays(1): counts
## rownames(26301): 497097 100503874 ... 100040384 100040400
## rowData names(0):
## colnames(9): control_FFa1.bam control_FFa2.bam ... mutant_KOb2.bam
```

```
##   mutant_KOb3.bam
## colData names(3): sample condition batch
```

```
# Run DESeq2 differential expression analysis
dds <- DESeq(dds)
```

```
# Get results
res <- results(dds)

# View top results
head(res)
```

```
## log2 fold change (MLE): condition KOb vs control
## Wald test p-value: condition KOb vs control
## DataFrame with 6 rows and 6 columns
##            baseMean log2FoldChange     lfcSE       stat      pvalue       padj
##           <numeric>      <numeric> <numeric>  <numeric>   <numeric>  <numeric>
## 497097     15.14479     -1.7865361  1.486567 -1.2017862 2.29446e-01         NA
## 100503874  11.85732     -2.3836573  2.053297 -1.1608925 2.45686e-01         NA
## 100038431   1.21545     -2.6540514  4.050501 -0.6552402 5.12313e-01         NA
## 19888      22.92281      0.0184976  1.375022  0.0134526 9.89267e-01         NA
## 20671      20.48147     -0.0243392  1.164116 -0.0209079 9.83319e-01         NA
## 27395     725.65285      1.0206448  0.259405  3.9345560 8.33506e-05 0.00103535
```

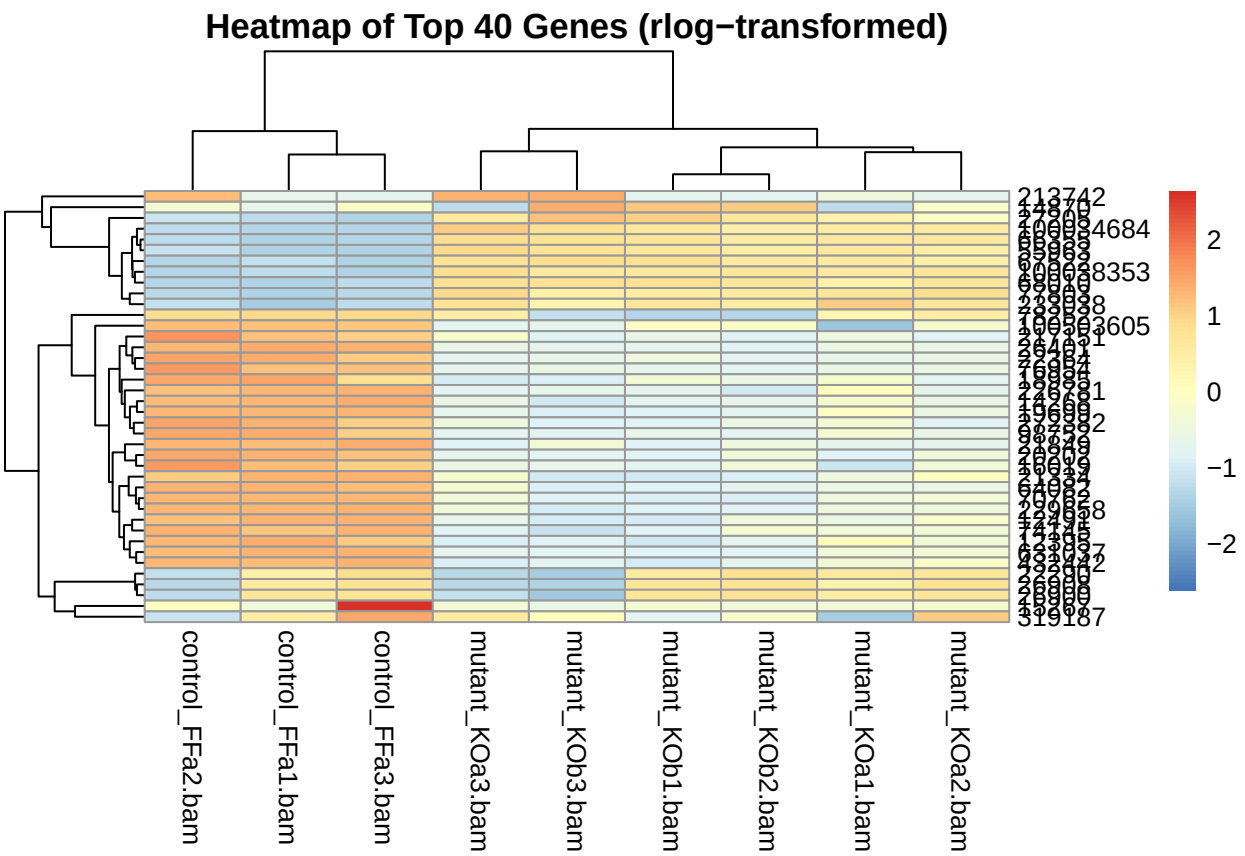
```
# Perform rlog transformation
rlog_dds <- rlog(dds, blind = TRUE)

# Perform variance stabilizing transformation (VST)
vst_dds <- vst(dds, blind = TRUE)

# Check the transformed data (optional)
head(assay(rlog_dds))
```

```
##           control_FFa1.bam control_FFa2.bam control_FFa3.bam mutant_KOa1.bam
## 497097           3.8561447        3.9907973        3.5322792        3.923806
## 100503874        3.2895831        2.8326071        2.8201166        3.834407
## 100038431        0.1543662        0.1551115        0.1903975        0.154316
## 19888            3.9103723        3.6262932        3.8451124        5.046242
## 20671            4.2033204        4.3750856        3.9098102        4.550450
## 27395            9.2090107        8.7918710        9.2758269        9.106490
##           mutant_KOa2.bam mutant_KOa3.bam mutant_KOb1.bam mutant_KOb2.bam
## 497097          3.8489878       3.9980385       3.7603473       3.5253500
## 100503874       2.8226210       2.9321205       2.8973928       2.8131612
## 100038431       0.1543294       0.3042526       0.1537077       0.1536189
## 19888           3.6068923       4.1745338       3.8179714       3.6830152
## 20671           4.0867366       4.4566664       4.2622207       4.0354695
## 27395           9.5797251       9.3512412       9.6722305       9.8827325
##           mutant_KOb3.bam
## 497097          3.5590907
## 100503874       2.8554608
## 100038431       0.1538331
## 19888           3.8816181
## 20671           4.2166975
## 27395           9.9615722
```

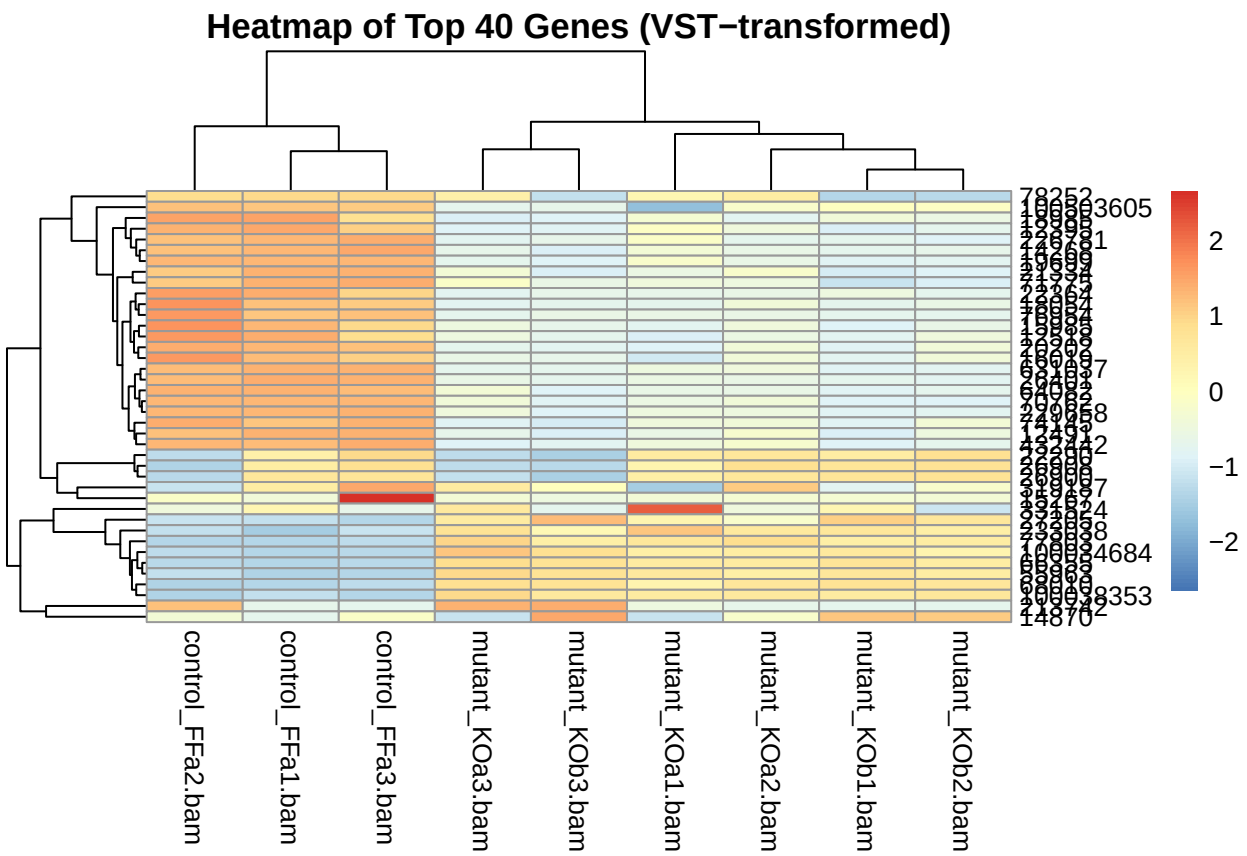
```r
head(assay(vst_dds))
```

```
##           control_FFa1.bam control_FFa2.bam control_FFa3.bam mutant_KOa1.bam
## 497097            7.139487         7.303577         6.476658        7.220155
## 100503874         7.216136         6.476658         6.476658        7.747532
## 100038431         6.476658         6.476658         6.692303        6.476658
## 19888             7.027737         6.476658         6.957081        8.128730
## 20671             7.097347         7.303577         6.476658        7.493025
## 27395             9.498159         9.038154         9.573177        9.384314
##           mutant_KOa2.bam mutant_KOa3.bam mutant_KOb1.bam mutant_KOb2.bam
## 497097           7.130573        7.305229        7.017147        6.476658
## 100503874        6.476658        6.799304        6.748085        6.476658
## 100038431        6.476658        6.932007        6.476658        6.476658
## 19888            6.476658        7.290141        6.929650        6.756214
## 20671            6.941007        7.389990        7.168352        6.871396
## 27395            9.918705        9.658047       10.024914       10.269679
##           mutant_KOb3.bam
## 497097           6.662121
## 100503874        6.662121
## 100038431        6.476658
## 19888            6.998734
## 20671            7.114359
## 27395           10.362465
```

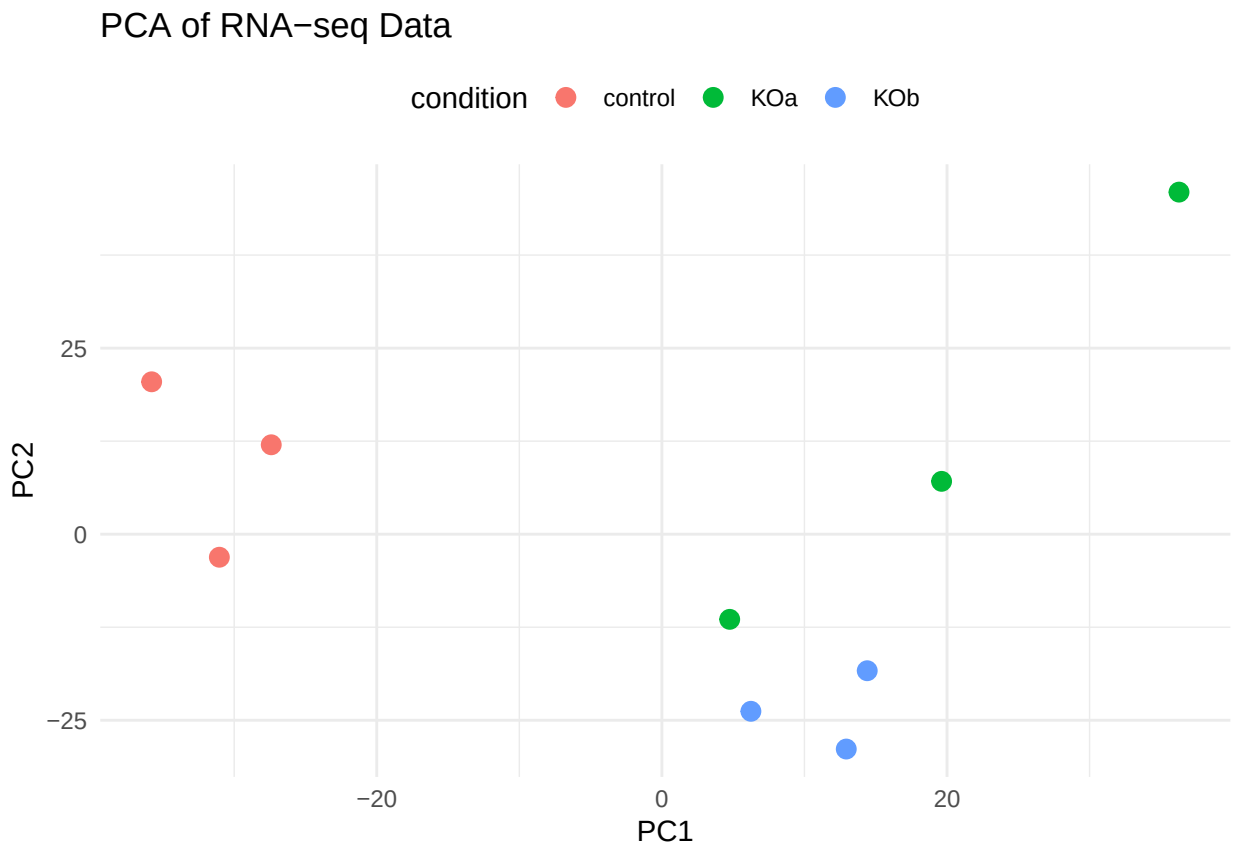
```r
# Load pheatmap library (if not already loaded)
library(pheatmap)

# Get the top 40 most variable genes using rlog-transformed data
rlog_data <- assay(rlog_dds)  # Access rlog-transformed data
vst_data <- assay(vst_dds)    # Access vst-transformed data

# Calculate the variance of each gene across samples
rlog_var_genes <- apply(rlog_data, 1, var)
vst_var_genes <- apply(vst_data, 1, var)

# Get the indices of the top 40 genes by variance
top_40_rlog_genes <- order(rlog_var_genes, decreasing = TRUE)[1:40]
top_40_vst_genes <- order(vst_var_genes, decreasing = TRUE)[1:40]

# Create heatmap for rlog-transformed data
pheatmap(rlog_data[top_40_rlog_genes, ],
         scale = "row",
         clustering_distance_rows = "euclidean",
         clustering_distance_cols = "euclidean",
         clustering_method = "complete",
         show_rownames = TRUE,
         show_colnames = TRUE,
         main = "Heatmap of Top 40 Genes (rlog-transformed)")
```




```
# Create heatmap for vst-transformed data
pheatmap(vst_data[top_40_vst_genes, ],
         scale = "row",
         clustering_distance_rows = "euclidean",
         clustering_distance_cols = "euclidean",
         clustering_method = "complete",
         show_rownames = TRUE,
         show_colnames = TRUE,
         main = "Heatmap of Top 40 Genes (VST-transformed)")
```




```r
# Load the required libraries
library(DESeq2)
library(ggplot2)

# Perform PCA on the rlog-transformed data (or VST-transformed if preferred)
rlog_data <- assay(rlog_dds)  # Access rlog-transformed data

# Perform PCA
pca_result <- prcomp(t(rlog_data))  # Transpose data so that samples are on rows

# Plot PCA
pca_df <- data.frame(PC1 = pca_result$x[, 1], PC2 = pca_result$x[, 2],
                     condition = sample_description$condition)

# Generate PCA plot
ggplot(pca_df, aes(x = PC1, y = PC2, color = condition)) +
  geom_point(size = 3) +
  labs(title = "PCA of RNA-seq Data", x = "PC1", y = "PC2") +
  theme_minimal() +
  theme(legend.position = "top")
```

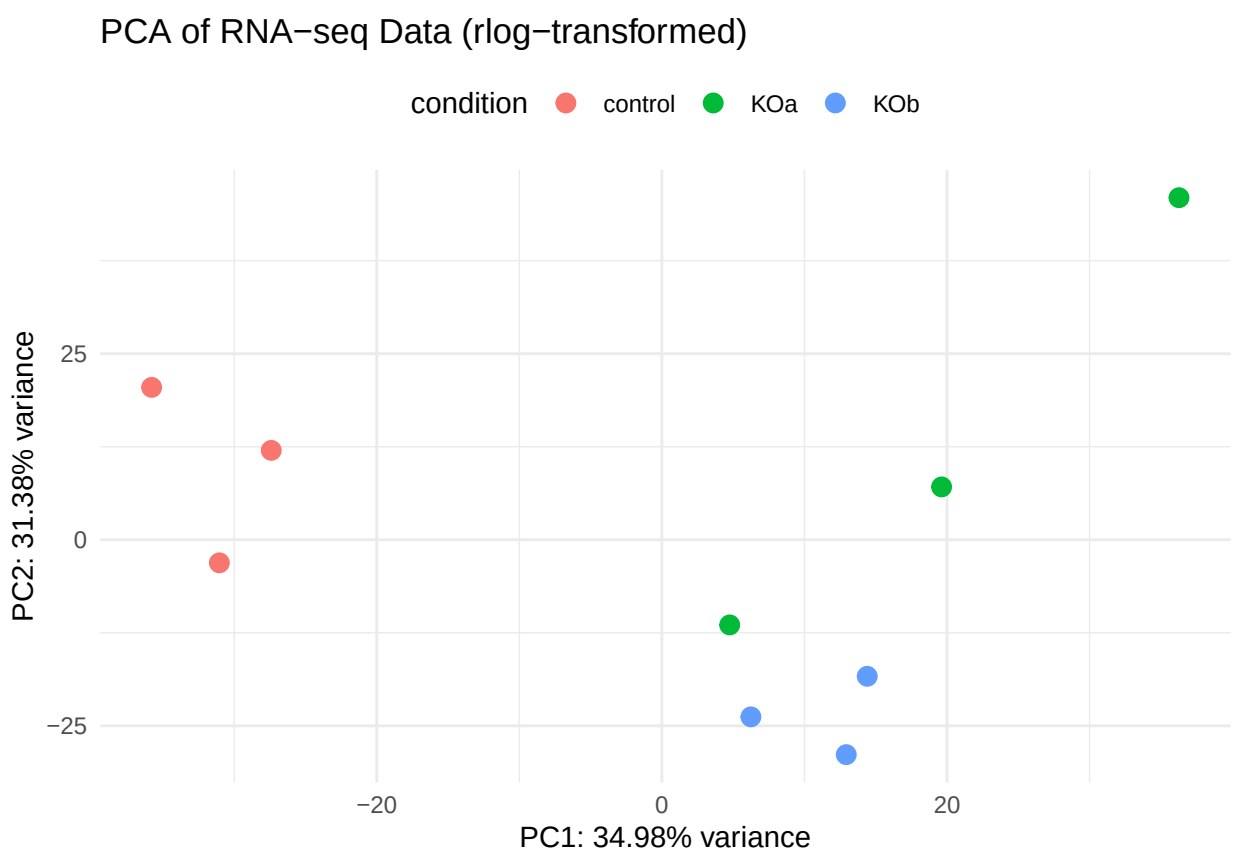



```
# Load the required libraries
library(DESeq2)
library(ggplot2)

# Perform PCA on the rlog-transformed data (already rlog-transformed as rlog_dds)
rlog_data <- assay(rlog_dds)  # Access rlog-transformed data

# Perform PCA
pca_result <- prcomp(t(rlog_data))  # Transpose data so that samples are rows

# Extract the proportion of variance explained by each principal component
variance_explained <- (pca_result$sdev^2) / sum(pca_result$sdev^2)

# Print variance explained by first two PCs
variance_explained[1:2]
```

```
## [1] 0.3498241 0.3137770
```

```
# Create a data frame for the PCA results
pca_df <- data.frame(PC1 = pca_result$x[, 1], PC2 = pca_result$x[, 2],
                     condition = sample_description$condition)

# Plot PCA with variance explained on the axes
ggplot(pca_df, aes(x = PC1, y = PC2, color = condition)) +
  geom_point(size = 3) +
```

```
  labs(title = "PCA of RNA-seq Data (rlog-transformed)",
       x = paste("PC1: ", round(variance_explained[1] * 100, 2), "% variance", sep = ""),
       y = paste("PC2: ", round(variance_explained[2] * 100, 2), "% variance", sep = "")) +
  theme_minimal() +
  theme(legend.position = "top")
```




```
# Perform VST transformation
vst_dds <- varianceStabilizingTransformation(dds, blind = FALSE)

# Get the VST-transformed data
vst_data <- assay(vst_dds)

# Perform PCA on VST-transformed data
pca_vst_result <- prcomp(t(vst_data))  # Transpose to treat samples as rows

# Extract the proportion of variance explained by each principal component for VST
variance_vst_explained <- (pca_vst_result$sdev^2) / sum(pca_vst_result$sdev^2)

# Create a data frame for the VST PCA results
pca_vst_df <- data.frame(PC1 = pca_vst_result$x[, 1], PC2 = pca_vst_result$x[, 2],
                         condition = sample_description$condition)

# Plot PCA for rlog and VST transformation
pca_rlog_df <- data.frame(PC1 = pca_result$x[, 1], PC2 = pca_result$x[, 2],
                          condition = sample_description$condition)
```

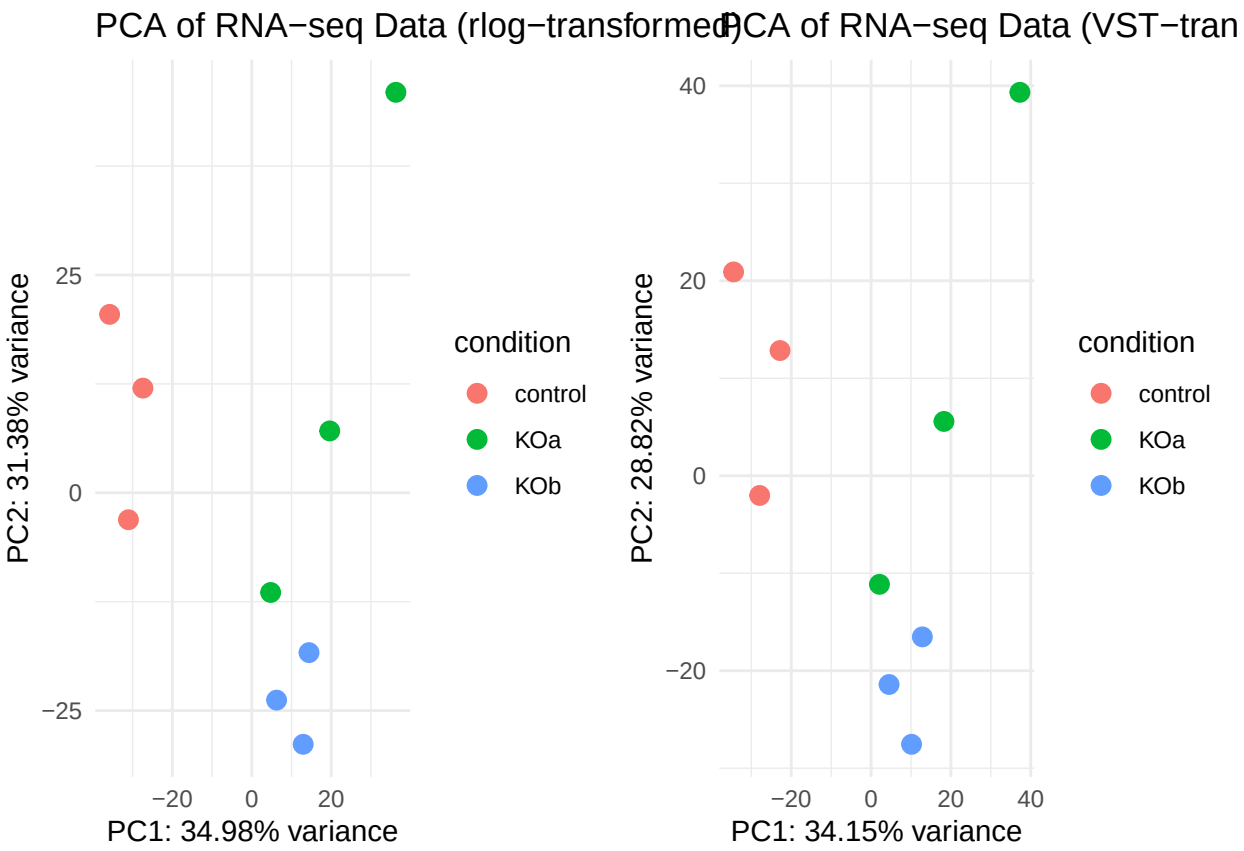
```r
# Create the plots side by side for comparison
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:Biobase':
##
##     combine
```

```
## The following object is masked from 'package:BiocGenerics':
##
##     combine
```

```r
p1 <- ggplot(pca_rlog_df, aes(x = PC1, y = PC2, color = condition)) +
  geom_point(size = 3) +
  labs(title = "PCA of RNA-seq Data (rlog-transformed)",
       x = paste("PC1: ", round(variance_explained[1] * 100, 2), "% variance", sep = ""),
       y = paste("PC2: ", round(variance_explained[2] * 100, 2), "% variance", sep = "")) +
  theme_minimal()

p2 <- ggplot(pca_vst_df, aes(x = PC1, y = PC2, color = condition)) +
  geom_point(size = 3) +
  labs(title = "PCA of RNA-seq Data (VST-transformed)",
       x = paste("PC1: ", round(variance_vst_explained[1] * 100, 2), "% variance", sep = ""),
       y = paste("PC2: ", round(variance_vst_explained[2] * 100, 2), "% variance", sep = "")) +
  theme_minimal()

# Arrange both plots side by side
grid.arrange(p1, p2, ncol = 2)
```


PCA of RNA−seq Data (rlog−transformed)
PC2: 31.38% variance
PC1: 34.98% variance
25
0
−25
−20
0
20
condition
control
KOa
KOb
PCA of RNA−seq Data (VST−tran
PC2: 28.82% variance
PC1: 34.15% variance
40
20
0
−20
−20
0
20
40
condition
control
KOa
KOb