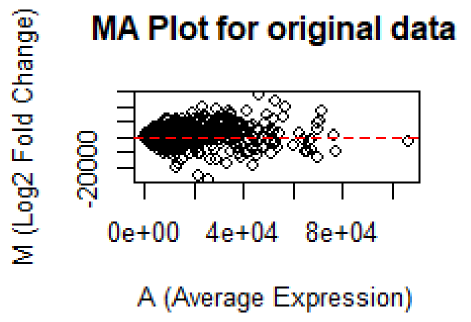
Final R project

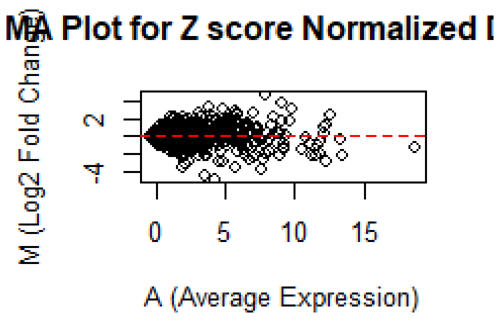
Anusha Jain

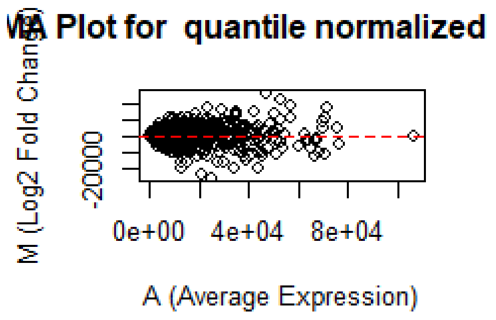
2023-12-06

```
library(R.utils)

#reading in the data of diabetic and nondiabetic renal tissue in mice
data_project <-
read.table("C:\\Users\\anush\\Downloads\\GSE642_series_matrix.txt", sep =
"",header = TRUE, row.names = 1)


#changing column names of samples to show which are diabetic samples and
which arent
column_names<-colnames(data_project)
new_column_names <- character(length(column_names))
for (i in 1:length(column_names)) {
  gsm_id <-column_names[i]
  if (i <=6) {
    new_column_names[i] <- paste(gsm_id, "Non_diabetic", sep = "_")
  } else {
    new_column_names[i] <- paste(gsm_id, "Diabetic", sep = "_")
  }
}
colnames(data_project) <- new_column_names
print(colnames(data_project))

##  [1] "GSM9920_Non_diabetic" "GSM9921_Non_diabetic" "GSM9922_Non_diabetic"
##  [4] "GSM9923_Non_diabetic" "GSM9924_Non_diabetic" "GSM9925_Non_diabetic"
##  [7] "GSM9926_Diabetic"     "GSM9927_Diabetic"     "GSM9928_Diabetic"
## [10] "GSM9929_Diabetic"     "GSM9930_Diabetic"     "GSM9931_Diabetic"

View(data_project)
dim(data_project)

## [1] 12488    12

#original data
original_data <- data_project[, grepl("Non_diabetic|Diabetic",
colnames(data_project))]

# Z-score normalization
normalized_data2<-scale((data_project))

#quantile normalization
library(preprocessCore)
```

```r
expression_data <- (data_project)
quant <- normalize.quantiles(as.matrix(expression_data))
quant <- (quant)
rownames(quant)<-rownames(data_project)
colnames(quant)<-new_column_names

#comparison

par(mfrow = c(2, 2))

M <- rowMeans(original_data[, 1:6]) - rowMeans(original_data[, 7:12])
A <- rowMeans(cbind(original_data[, 1:6],original_data[, 7:12]))
plot(A,M, main = "MA Plot for original data",
     xlab = "A (Average Expression)", ylab = "M (Log2 Fold Change)")
abline(h = 0, col ="red",lty=2)


M <- rowMeans(normalized_data2[, 1:6]) - rowMeans(normalized_data2[, 7:12])
A <- rowMeans(cbind(normalized_data2[, 1:6], normalized_data2[, 7:12]))
plot(A,M, main = "MA Plot for Z score Normalized Data",
     xlab = "A (Average Expression)", ylab = "M (Log2 Fold Change)")
abline(h = 0, col ="red",lty=2)


M <- rowMeans(quant[, 1:6]) - rowMeans(quant[, 7:12])
A <- rowMeans(cbind(quant[, 1:6],quant[, 7:12]))
plot(A,M, main = "MA Plot for  quantile normalized data",
     xlab = "A (Average Expression)", ylab = "M (Log2 Fold Change)")
abline(h = 0, col ="red",lty=2)

par(mfrow = c(1, 1))
```

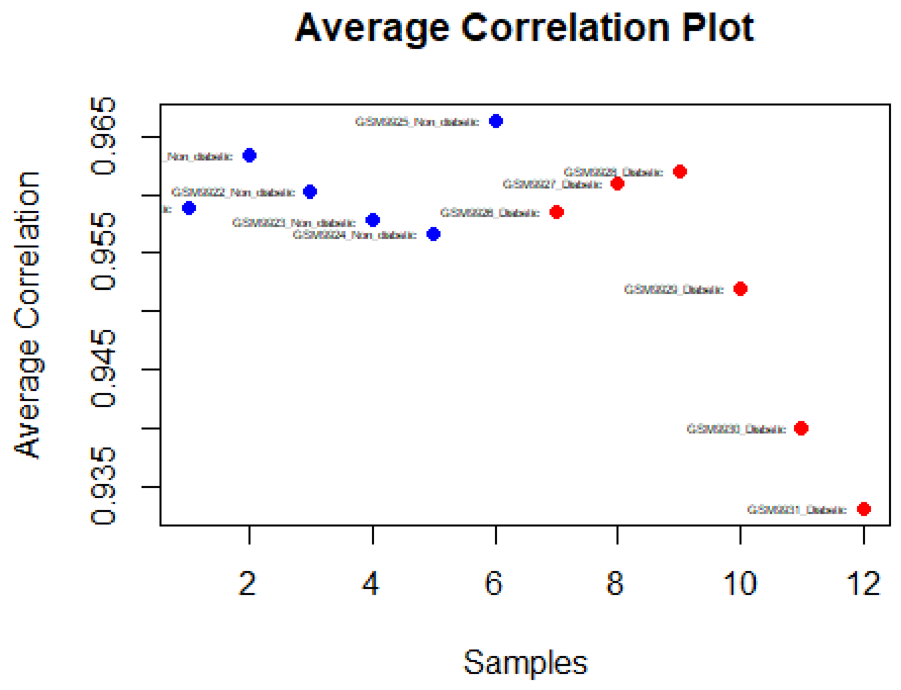




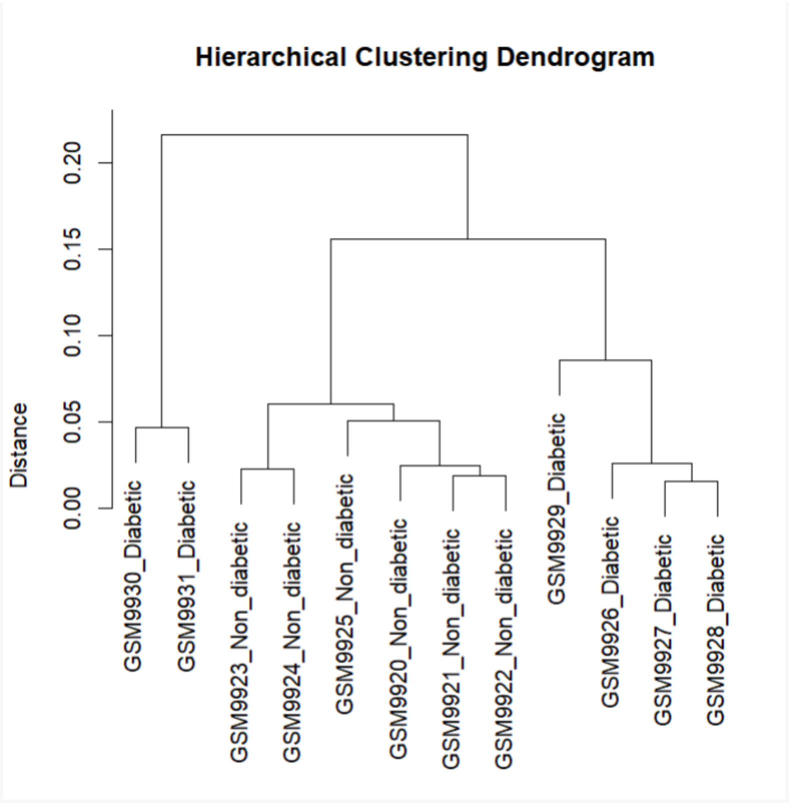


```
# to identify the outliers I plotted an avg correlation plot and dendrogram
labels <- c(rep("Non_diabetic", 6), rep("Diabetic", 6))
sample_info <- data.frame(GSM_ID = colnames(normalized_data2), Condition =
labels)
print(sample_info)

##                   GSM_ID    Condition
## 1  GSM9920_Non_diabetic Non_diabetic
## 2  GSM9921_Non_diabetic Non_diabetic
## 3  GSM9922_Non_diabetic Non_diabetic
## 4  GSM9923_Non_diabetic Non_diabetic
## 5  GSM9924_Non_diabetic Non_diabetic
## 6  GSM9925_Non_diabetic Non_diabetic
## 7      GSM9926_Diabetic     Diabetic
## 8      GSM9927_Diabetic     Diabetic
## 9      GSM9928_Diabetic     Diabetic
## 10     GSM9929_Diabetic     Diabetic
## 11     GSM9930_Diabetic     Diabetic
## 12     GSM9931_Diabetic     Diabetic

colors <- ifelse(sample_info$Condition == "Diabetic", "red", "blue")


library(gplots)

matrix <- cor(normalized_data2, use = "pairwise.complete.obs")
avg_correlation <- apply(matrix, 1, mean)
```

```
plot(avg_correlation,
     main = "Average Correlation Plot",
     xlab = "Samples",
     ylab = "Average Correlation",
     pch = 19, col = colors)
sample_names<- colnames(normalized_data2)
text(x = 1:length(sample_names), y = avg_correlation, labels =
sample_names,cex=0.4, pos = 2, offset = 0.4)
```

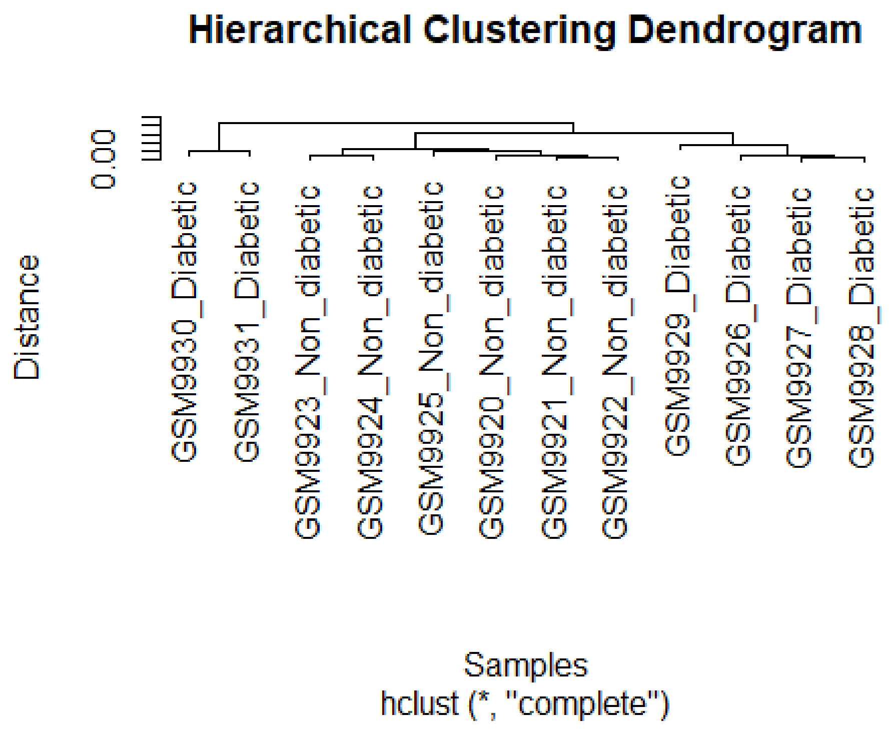


```
dendrogram <- hclust(dist(matrix))
plot(dendrogram,
     main = "Hierarchical Clustering Dendrogram",
     xlab = "Samples",
     ylab = "Distance")
```

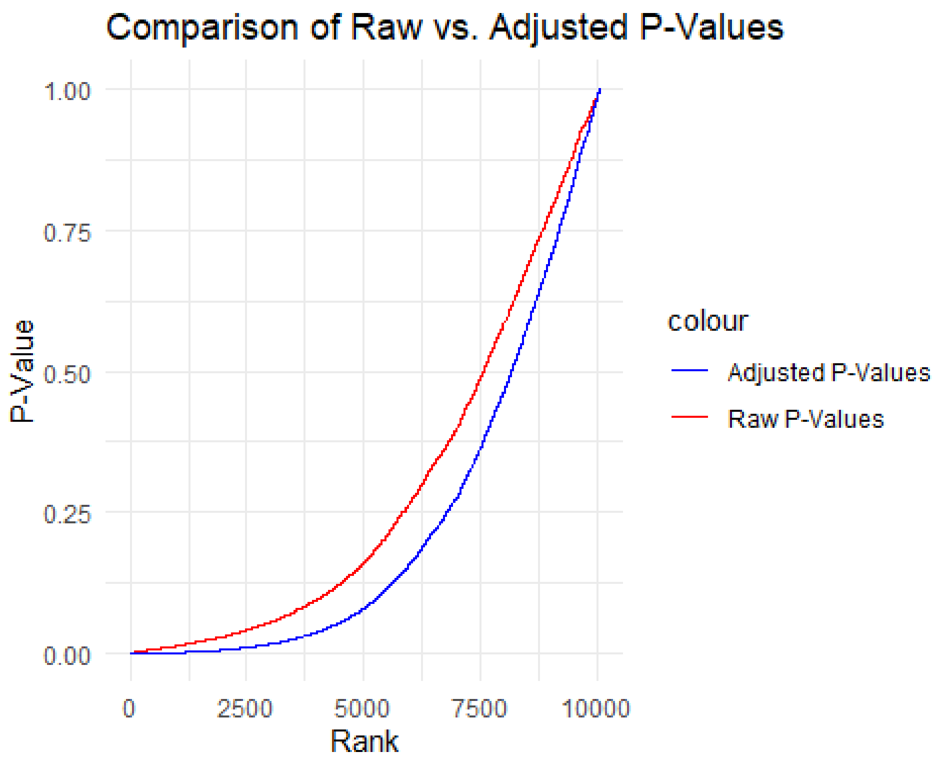


```
#the outliers are GSM9930_Diabetic and GSM9931_Diabetic

#removing the outliers
library(impute)
columns_to_remove <- c(11, 12)
newdata_project <- data_project[, -columns_to_remove]


#normalize data after removing outliers
normalized_data1<-scale(newdata_project)


#removing genes with expression values lower than 25th percentile
log_expression_data <- log2(normalized_data1 + 1)
percentile_threshold <- quantile(as.matrix(log_expression_data, 0.25))
low_expression_genes <- rownames(log_expression_data)[log_expression_data <
percentile_threshold[2]]
gene_logical_vector <- rownames(log_expression_data) %in%
low_expression_genes
newdata1_project<- normalized_data1[!gene_logical_vector, ]
dim(normalized_data1)

## [1] 12488    10

dim(newdata1_project)
```

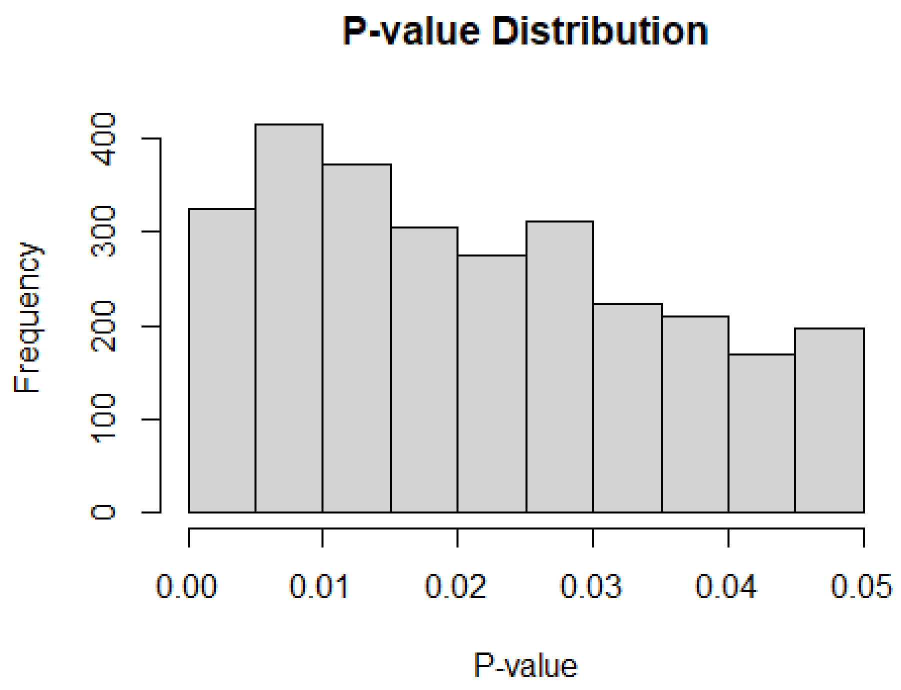
```
## [1] 10056    10

ann.dat2<-colnames(newdata1_project)
ann.dat2 <- ifelse(grepl("Non_diabetic", ann.dat2), 0, 1)

#Two sample test for diabetic and non diabetic
t.test.genes <- function(x,s1,s2) {
  x1 <- x[s1]
  x2 <- x[s2]
  x1 <- as.numeric(x1)
  x2 <- as.numeric(x2)
  t.out <- t.test(x1,x2, alternative="two.sided",var.equal=T)
  out <- as.numeric(t.out$p.value)
  return(out)

}
ttest<- apply(newdata1_project,1,t.test.genes,s1=ann.dat2==0,s2=ann.dat2==1)
ttest<-as.matrix(ttest)
View(ttest)
#adjusting for multiplicity using BH
adjusted <- p.adjust(ttest,method="BH")
p_value_df <- data.frame(Raw_P_Values = ttest, Adjusted_P_Values = adjusted)
sorted_p_values <- p_value_df[order(adjusted), ]
xaxis <- seq(1, nrow(sorted_p_values))
library(ggplot2)
```



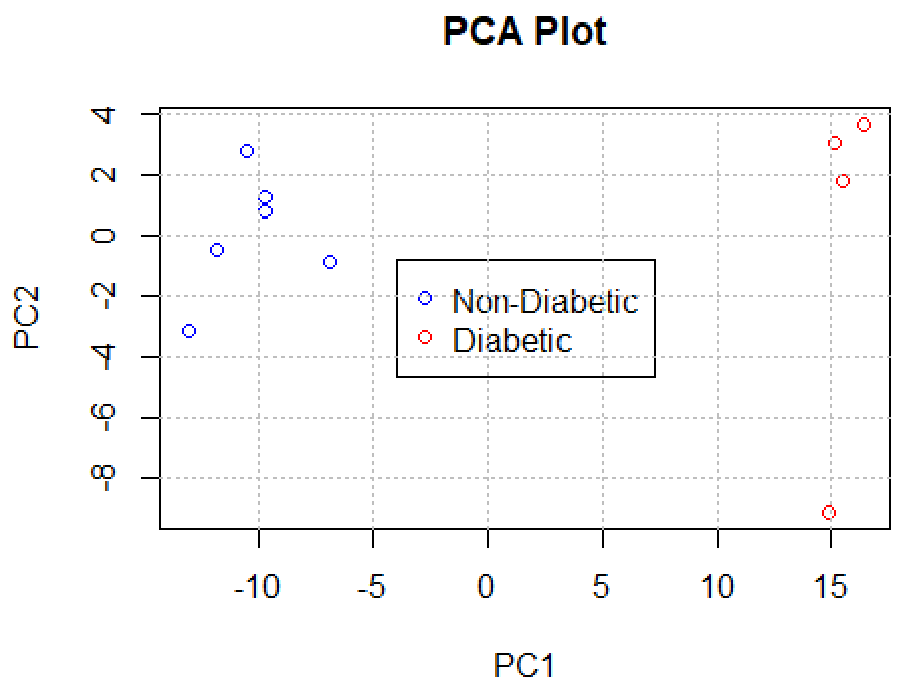
```r
ggplot(sorted_p_values, aes(x = xaxis, y = Adjusted_P_Values, color = "Raw P-
Values")) +
  geom_line() +
  geom_line(aes(y = Raw_P_Values, color = "Adjusted P-Values")) +
  labs(title = "Comparison of Raw vs. Adjusted P-Values", x = "Rank", y = "P-
Value") +
  scale_color_manual(values = c("Adjusted P-Values" = "blue", "Raw P-Values"
= "red")) +
  theme_minimal()
```



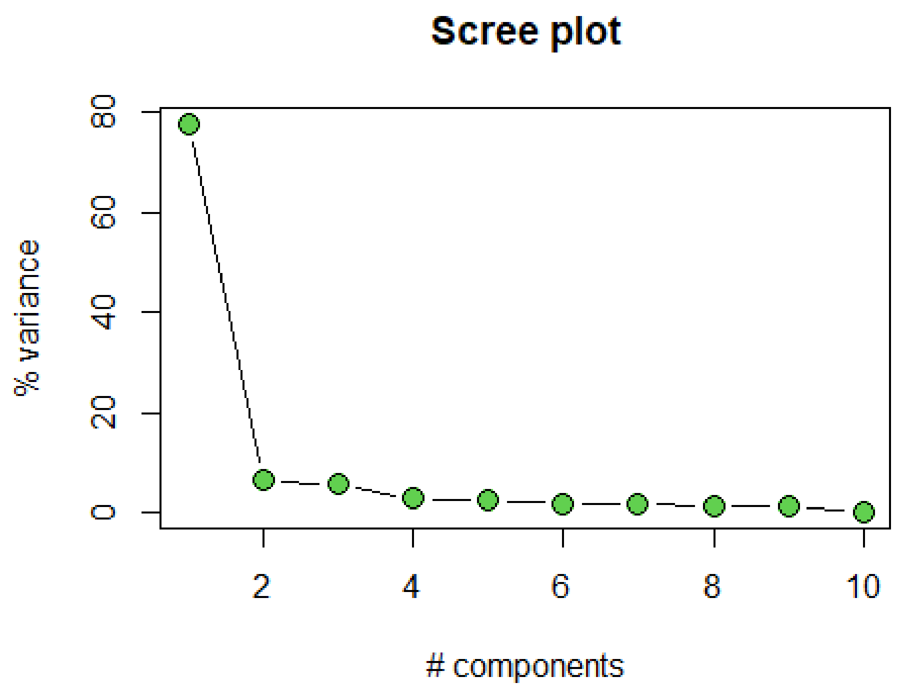
```r
#adjusting for multiplicity using holm
adjusted2<-p.adjust(ttest,method="holm")
#did not plot as all values have been adjusted to 1 using holm

#getting the genes of significance (pvalue less than 0.05)
significant_genes <- rownames(newdata1_project)[adjusted < 0.05]
length(significant_genes)

## [1] 2806

#plotting the scores of the significant genes in a histogram
hist(adjusted[adjusted<=0.05], main = "P-value Distribution", xlab = "P-
value")
```



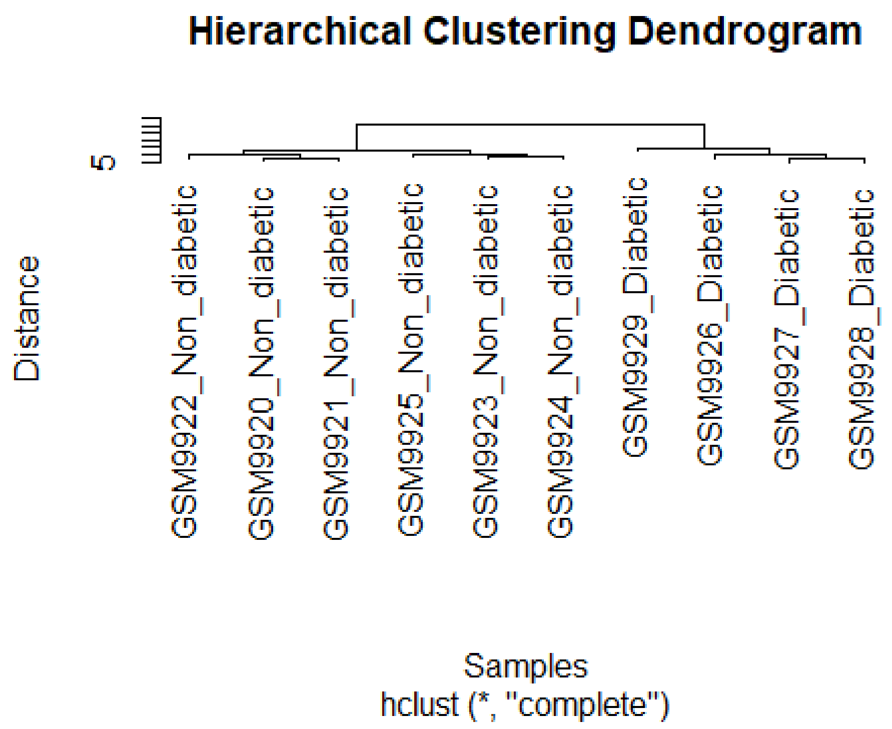
```
#subsetting data according to significant genes
subset<-newdata1_project[significant_genes,]
dim(subset)

## [1] 2806   10

#using pca and dendrogram as clustering methods
pca_result <- prcomp(t(subset))
pca_scores <- pca_result$x[, 1:2]  # Using the first two principal components
legend_colors <- c("blue", "red")
plot(pca_scores, col = legend_colors[ann.dat2+1], main = "PCA Plot")
legend("center", legend = c("Non-Diabetic", "Diabetic"), col = legend_colors,
pch = 1)
grid(col="grey")
```



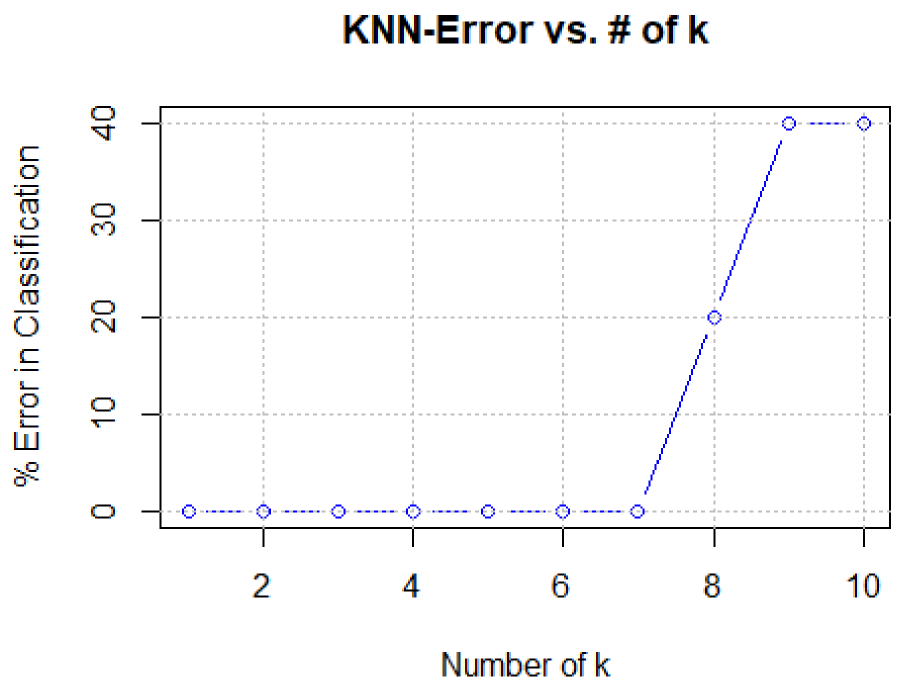
```
#why using pca  A scree plot, on the other hand, is a diagnostic tool to
check whether PCA
#works well on your data or not. Principal components are created in order of
the amount of
#variation they cover: PC1 captures the most variation, PC2 — the second most

dat.pca.var <- round(pca_result$sdev^2 / sum(pca_result$sdev^2)*100,2)
plot(1:length(dat.pca.var),dat.pca.var,type="b",xlab="# components",ylab="%
variance",pch=21,col=1,bg=3,cex=1.5,main="Scree plot")
```



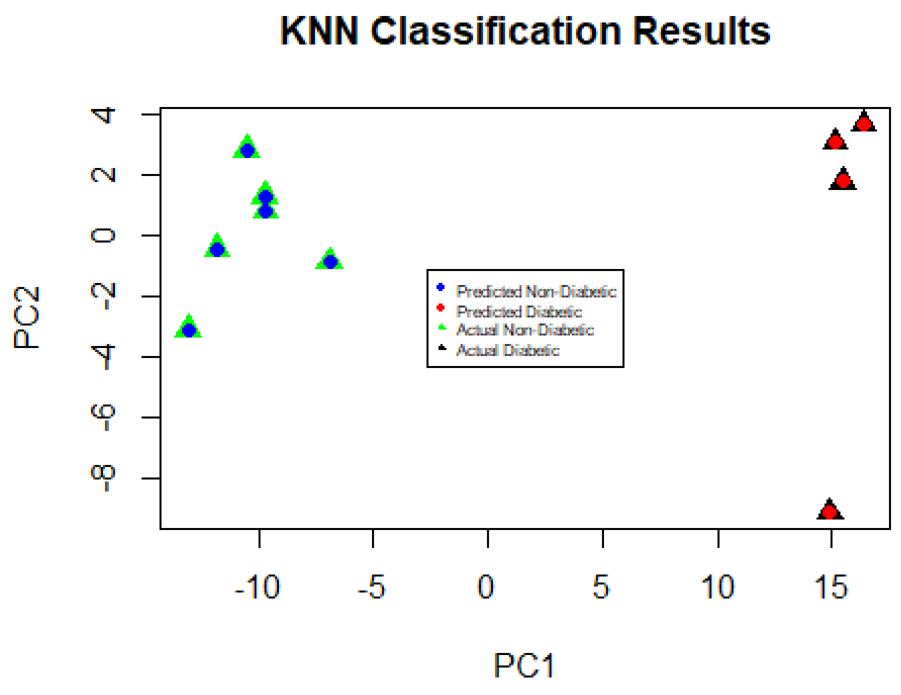
```
total_variability <- sum(dat.pca.var[1:2])
cat( round(total_variability , 2), "% of variability is explained by the
first two principal components.")

## 84.06 % of variability is explained by the first two principal components.

dendrogram2 <- hclust(dist(t(subset)))
plot(dendrogram2, main = "Hierarchical Clustering Dendrogram", xlab =
"Samples", ylab = "Distance")
```



```
library(class)

data_for_knn <- data.frame(pca1 = pca_scores[, 1], pca2 = pca_scores[, 2],
class = ann.dat2)
set.seed(100)
train_indices <- sample(1:nrow(data_for_knn), 0.6 * nrow(data_for_knn))
train_data <- data_for_knn[train_indices, ]
test_data <- data_for_knn[-train_indices, ]



# to determine value of k
sample_groups <- as.factor(data_for_knn$class)
error_list <- NULL
for (i in 1:10) {
  # Perform k-NN classification
  determine_k<- knn(data_for_knn[, c("pca1", "pca2")], data_for_knn[,
c("pca1", "pca2")], sample_groups, k = i, prob = TRUE)
  err1 <- sum(determine_k[sample_groups == 0] == 1)  # Number of incorrect
Non-Diabetic classifications
  err2 <- sum(determine_k[sample_groups == 1] == 0)  # Number of incorrect
Diabetic classifications
  totalerror <- sum(err1, err2) / nrow(data_for_knn) * 100
  error_list <- c(error_list, round(totalerror, 1))
}

plot(c(1:10), error_list, type = "b", col = "blue",
     xlab = "Number of k", ylab = "% Error in Classification",
     main = "KNN-Error vs. # of k")
grid(col = "grey")
```

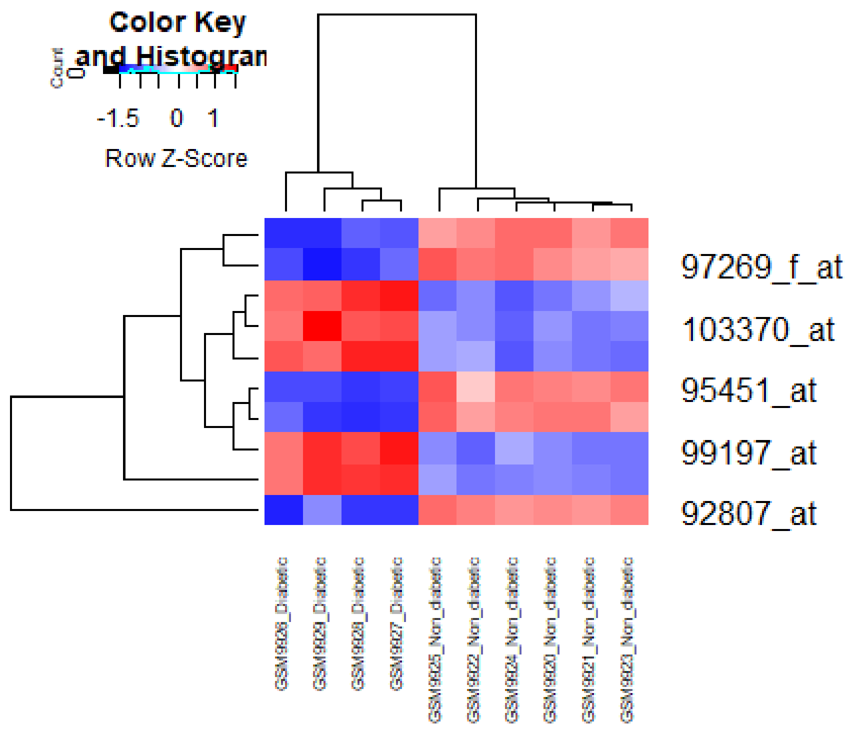


```
#knn classification
predicted_classes <- knn(train_data[, 1:2], test_data[, 1:2],
train_data$class, k=3)

#calculating accuracy
accuracy <- sum(predicted_classes == test_data$class) / nrow(test_data)
cat("Accuracy:", round(accuracy * 100, 2), "%\n")

## Accuracy: 100 %

#rearranging test data according to predicted classes
test_data$class <- predicted_classes
print(test_data)

##                            pca1       pca2 class
## GSM9923_Non_diabetic -13.09451 -3.0937353     0
## GSM9924_Non_diabetic -11.83001 -0.4436055     0
## GSM9927_Diabetic      15.45507  1.7970328     1
## GSM9928_Diabetic      15.12278  3.0818486     1

#creating a df using rearranged test_data and train_data
combined_data <- rbind(
  data.frame(pca1 = train_data$pca1, pca2 = train_data$pca2, class =
train_data$class),
  data.frame(pca1 = test_data$pca1, pca2 = test_data$pca2, class =
test_data$class)
)

# Create a scatter plot with predicted and actual class colors and symbols
colors2<-c("green", "black")
plot(pca_scores, col = legend_colors[predicted_classes + 1], pch = 1,
     main = "KNN Classification Results")

## Warning in Ops.factor(predicted_classes, 1): '+' not meaningful for
factors

points(pca_scores, col = colors2[ann.dat2+1],pch=17,cex=1.5)
points(combined_data[combined_data$class==0 , ], col = "blue", pch = 19)  #
Non-Diabetic (circle)
points(combined_data[combined_data$class==1, ], col = "red", pch = 19)
legend("center", legend = c("Predicted Non-Diabetic", "Predicted Diabetic",
"Actual Non-Diabetic", "Actual Diabetic"),
       col = c("blue", "red", "green", "black"), pch = c(19, 19, 17,
17),cex=0.5)
```



```
#Getting top 5 discriminant genes (positive and negative direction)

library(limma)

design_matrix <- model.matrix(~ann.dat2)
fit <- lmFit(subset, design_matrix)
```

```r
fit_ebayes <- eBayes(fit)
genes<- topTable(fit_ebayes, coef = "ann.dat2", sort.by = "t",15)
# Selecting top positive genes
top_positive_genes <- genes[genes$t > 0, ][1:5, ]
# Selecting top negative genes
top_negative_genes <- genes[genes$t < 0, ][1:5, ]

print(rownames(top_positive_genes))

## [1] "101872_at" "93268_at"  "99197_at"  "99147_at"  "103370_at"

print(rownames(top_negative_genes))

## [1] "99599_s_at" "104260_at"  "92807_at"   "97269_f_at" "95451_at"

subset_data <- subset[c(rownames(top_positive_genes),
rownames(top_negative_genes)), ]
subset_data <- as.matrix(subset_data)
par(mfrow = c(1, 1))
# Create a heatmap
heatmap.2(subset_data, scale = "row", trace = "none", col =
bluered(50),margins = c(8, 10),cexCol=0.6)
```


Color Key
and Histogram
Count
Row Z-Score
97269_f_at
103370_at
95451_at
99197_at
92807_at
GSM9926_Diabetic
GSM9929_Diabetic
GSM9928_Diabetic
GSM9927_Diabetic
GSM9925_Non_diabetic
GSM9922_Non_diabetic
GSM9924_Non_diabetic
GSM9920_Non_diabetic
GSM9921_Non_diabetic
GSM9923_Non_diabetic