Project Network-based Data Analysis

Annalisa Xamin

Contents

Data selection

The analysis will use the dataset GSE20437 obtained from GEO.The dataset is generated from Affymetrix HU133A microarrays and contains 42 tissue samples.

In detail, the data includes:

- 18 reduction mammoplasty (RM) breast epithelium samples,
- 18 histologically normal (HN) epithelial samples from breast cancer patients (9 ER+ and 9 ER-), and
- 6 histologically normal epithelial samples from prophylactic mastectomy patients.

Note that sample numbers correspond to individual patient samples.

```
# download the GSE20437 expression data series
#gse <- getGEO("GSE20437", destdir= './data/', getGPL = F)

# load the local copy of the data
gse <- getGEO(file = "./data/GSE20437_series_matrix.txt.gz", getGPL = FALSE)
```

```
# getGEO returns a list of expression objects, but...
length(gse)
```

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

```
# shows us there is only one object in it.
# We assign it to the same variable.
#gse <- gse[[1]] # run only if you download data and
# if you don't use the local copy
```

```
# extract metadata
metadata <- data.frame(gse@phenoData@data)
```

```
expr(gse[1])
```

```
## gse[1]
```

Annotation of probesets from a GEO dataset to gene symbols

For the later analysis, it is useful to annotate the probe sets of the GEO data set to gene symbol. To do that, first we extract the name of the probes and then, we perform the annotation using biomaRt.

```
id_to_annotate <- rownames(gse@assayData[["exprs"]])
# extract id to annotate

# annotation
mart <- useMart("ENSEMBL_MART_ENSEMBL")
mart <- useDataset("hsapiens_gene_ensembl", mart)
annotLookup <- getBM(
  mart = mart,
  attributes = c(
    "affy_hg_u133_plus_2",
    "ensembl_gene_id",
    "gene_biotype",
    "external_gene_name",
    "description"),
  filter = "affy_hg_u133_plus_2",
  values = id_to_annotate,
  uniqueRows=TRUE)
```

```
head(annotLookup)
```

```
##   affy_hg_u133_plus_2 ensembl_gene_id   gene_biotype external_gene_name
## 1           203957_at ENSG00000278573     pseudogene
## 2         201104_x_at ENSG00000271254 protein_coding
## 3           203957_at ENSG00000278066     pseudogene
## 4           204673_at ENSG00000293532 protein_coding              MUC2
## 5           202801_at ENSG00000288516 protein_coding             PRKACA
## 6           203488_at ENSG00000288324 protein_coding             ADGRL1
##                                                                                  description
## 1
## 2
## 3
## 4                 mucin 2, oligomeric mucus/gel-forming [Source:HGNC Symbol;Acc:HGNC:7512]
## 5 protein kinase cAMP-activated catalytic subunit alpha [Source:HGNC Symbol;Acc:HGNC:9380]
## 6              adhesion G protein-coupled receptor L1 [Source:HGNC Symbol;Acc:HGNC:20973]
```

```
indicesLookup <- match(rownames(gse@assayData[["exprs"]]), annotLookup$affy_hg_u133_plus_2)
#head(annotLookup[indicesLookup, "external_gene_name"])
```

```
dftmp <- data.frame(rownames(gse@assayData[["exprs"]]),
                    annotLookup[indicesLookup,
                                c("affy_hg_u133_plus_2", "external_gene_name")])
head(dftmp, 20)
```

```
##      rownames.gse.assayData...exprs.... affy_hg_u133_plus_2 external_gene_name
```

```
## 2697                          1007_s_at          1007_s_at              DDR1
## 2072                            1053_at            1053_at              RFC2
## 2748                             117_at             117_at             HSPA6
## 5260                             121_at             121_at              PAX8
## 3793                          1255_g_at          1255_g_at  GUCA1ANB-GUCA1A
## 4330                            1294_at            1294_at              UBA7
## 5181                            1316_at            1316_at              THRA
## 295                             1320_at            1320_at            PTPN21
## 1843                          1405_i_at          1405_i_at              CCL5
## 1896                            1431_at            1431_at            CYP2E1
## 5223                            1438_at            1438_at             EPHB3
## 357                             1487_at            1487_at           ESRRAP1
## 742                           1494_f_at          1494_f_at            CYP2A6
## 107                           1598_g_at          1598_g_at              GAS6
## 1753                        160020_at          160020_at              MMP14
## 1918                            1729_at            1729_at             TRADD
## 2040                            1773_at            1773_at       CHURC1-FNTB
## 5218                             177_at             177_at              PLD1
## 1405                             179_at             179_at           UPK3BP1
## 2176                            1861_at            1861_at               BAD
```

```
length(rownames(gse@assayData[["exprs"]]))
```

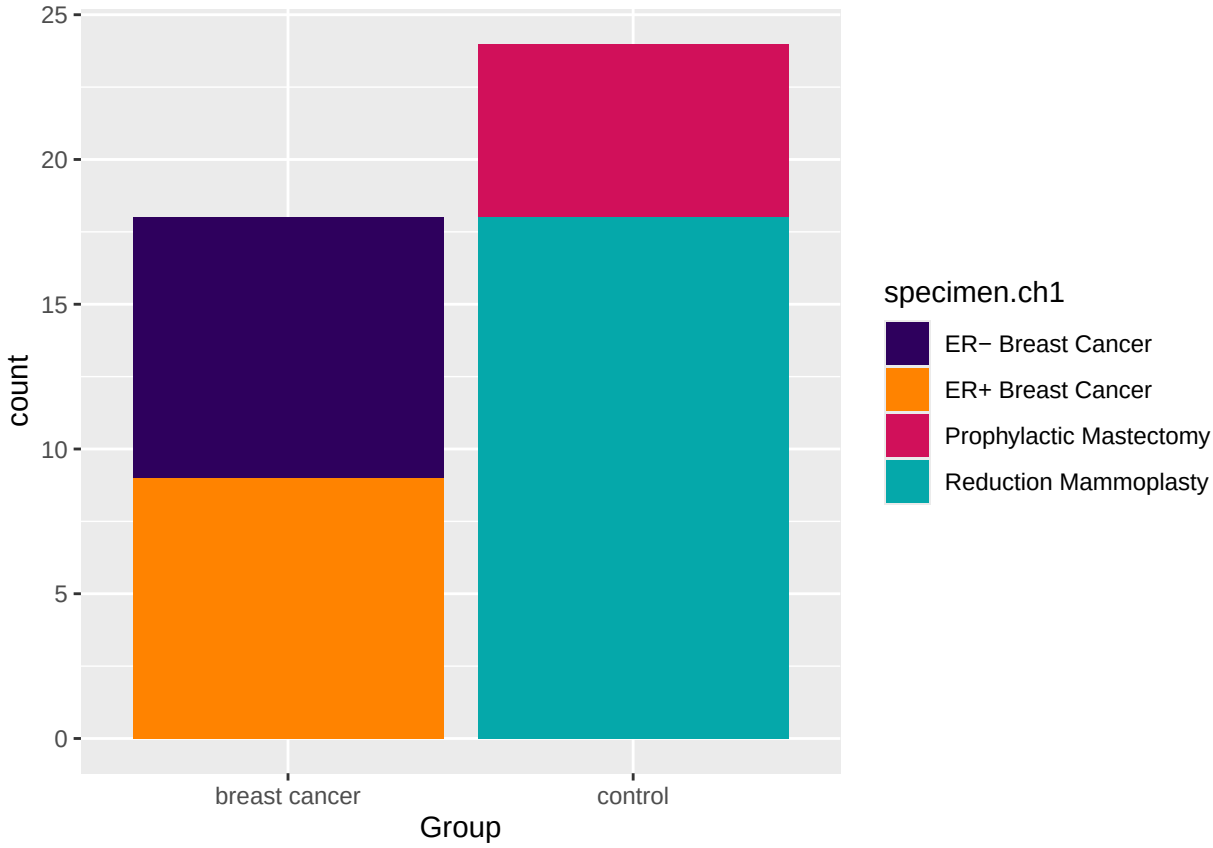
```
## [1] 22283
```

```
table(dftmp[,1] == dftmp[,2])
```

```
##
##  TRUE
## 20259
```

Explore the groups in the dataset

```
p <- ggplot(metadata, aes(x=disease.state.ch1, fill=specimen.ch1))+
  geom_bar()+
  scale_fill_manual(values = my_colors[c(1,4,6,7)])
p + labs(x = "Group")
```



Exploratory analysis

```
# show what we have:
show(gse)
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 22283 features, 42 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: GSM512539 GSM512540 ... GSM512580 (42 total)
##   varLabels: title geo_accession ... tissue:ch1 (38 total)
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
##   pubMedIds: 20197764
## Annotation: GPL96
```

The actual expression data are accessible in the `exprs` section of `gse`, an Expression Set and the generic data class that BioConductor uses for expression data.

```
head(exprs(gse))
```

```
##           GSM512539 GSM512540 GSM512541 GSM512542 GSM512543 GSM512544 GSM512545 GSM512546 GSM512547 G
## 1007_s_at    2461.4    3435.7    1932.5    2377.7    3055.3    2978.1    2348.5    2963.9    2776.9
## 1053_at        26.7     159.0      31.2     140.7      69.9      98.5      37.0      59.9      86.7
## 117_at         82.6     243.4     150.2      95.1     209.3     103.4      91.2     168.4     162.7
```

```
## 121_at        942.3     897.5     840.8     870.9     685.4     791.8     886.5     954.2     843.1
## 1255_g_at      71.8      87.9      75.4      58.1      31.8      40.3      70.5      43.3      51.6
## 1294_at       630.2     571.4     346.3     679.9    1289.3     421.1     417.6     811.6     778.1
##           GSM512550 GSM512551 GSM512552 GSM512553 GSM512554 GSM512555 GSM512556 GSM512557 GSM512558 G
## 1007_s_at    3037.1    3545.8    3322.6    1963.7    3609.6    2078.9    4138.6    4260.7    2453.6
## 1053_at        82.9      97.7      69.7      82.0      45.6      84.5      31.7      37.4      82.4
## 117_at        113.5      80.0     186.4     106.6     145.6     144.4     133.6     278.6     173.0
## 121_at        912.2     911.6     862.4     705.0     984.6     853.8     846.8    1273.0     833.6
## 1255_g_at      53.7      30.5      15.2      42.5      76.6      88.2      90.6      65.8      25.8
## 1294_at       987.7     938.5     924.6     480.8    1054.1     632.0     448.0    1345.2    1248.9
##           GSM512561 GSM512562 GSM512563 GSM512564 GSM512565 GSM512566 GSM512567 GSM512568 GSM512569 G
## 1007_s_at    4340.1    3155.3    2390.3    2738.8    3233.1    2836.6    2915.4    3457.5    2798.7
## 1053_at        76.7     100.3     115.4      14.1      47.6      77.1      47.1      47.0      83.2
## 117_at        168.0      95.2      73.6     122.7     107.6     120.9     143.4      92.5      72.1
## 121_at        827.0     629.4     709.2     305.6     877.4     425.7     643.8     771.3     681.1
## 1255_g_at      87.9      44.6      59.3      12.0      82.1      59.2      62.2      28.3      97.6
## 1294_at      2218.1    1321.1     606.7    1709.9     980.8    1268.4     955.8    1157.5     888.6
##           GSM512572 GSM512573 GSM512574 GSM512575 GSM512576 GSM512577 GSM512578 GSM512579 GSM512580
## 1007_s_at    3669.5    3310.1    3942.2    4520.4    3596.1    2989.0    3164.5    2764.3    4258.5
## 1053_at        24.1       8.8      44.6      54.7      56.7      89.9      63.4      57.0      59.5
## 117_at        165.8     141.6      97.1     132.7     124.3     210.5     131.4      89.6     123.3
## 121_at        746.9    1090.3    1008.7     718.6     988.4     295.9     957.3     630.8     869.2
## 1255_g_at      53.0      39.9      11.0      50.2      60.0      34.3      33.5      61.7      50.4
## 1294_at      1138.5     483.0    1326.5    1179.4     668.3     863.2    1055.5    1287.6    1127.8
```

To conveniently access the data rows and columns present in `exprs(gse)`, this matrix is assigned to its own variable `ex`.

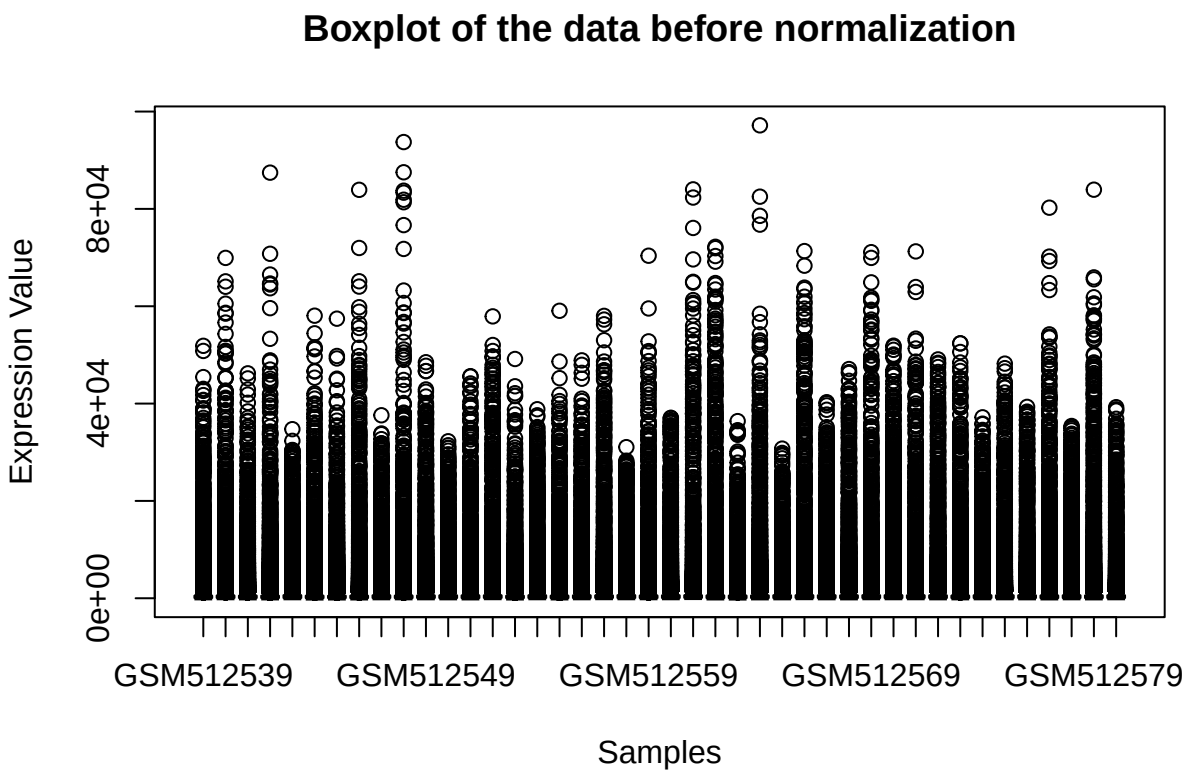
```
# exprs (gse) is a matrix that we can assign to its own variable, to
# conveniently access the data rows and columns
ex <- exprs(gse)
dim(ex) # 42 sample, 22283 genes
```

```
## [1] 22283    42
```

The dataset contains gene expression data of 22283 genes (rows) from 42 patients (columns).

Pre-processing

```
# Analyze value distributions
boxplot(ex, main = 'Boxplot of the data before normalization',
        xlab = "Samples",
        ylab = "Expression Value",
        varwidth = TRUE
        )
```



The boxplot shows that scaling is necessary. So, in this case, I try to apply a log transformation to the data.

```
ex2<-log(ex)
ex2 <- na.omit(as.matrix(ex2))
#dim(ex2) # 22283    42 same as before
boxplot(ex2, main = 'Boxplot of the data after applying a logarithmic transformation',
        xlab = "Samples",
        ylab = "Expression Value"
        )
```

From the boxplot after the log transformation, I can see that there is some variation in the median of the samples. So, one of the simplest normalization strategies is to align the log values so that all channels have the same median. A convenient choice is zero so that positive or negative values reflect signals above or below the median for a particular channel.

```
normalized.log.ex=scale(log(ex))

# boxplot post median normalization on ex
boxplot(normalized.log.ex,
        main = 'Boxplot of the data after median normalization',
        xlab = "Samples",
        ylab = "Expression Value")
```

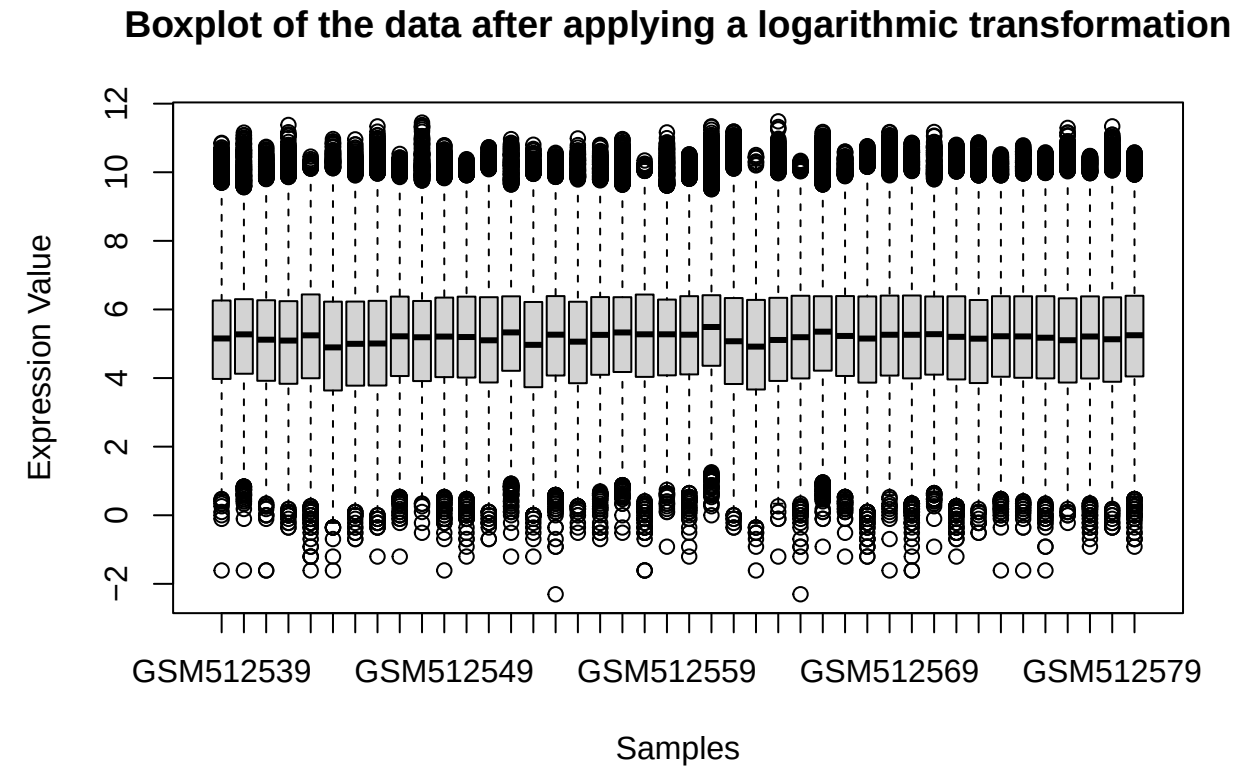
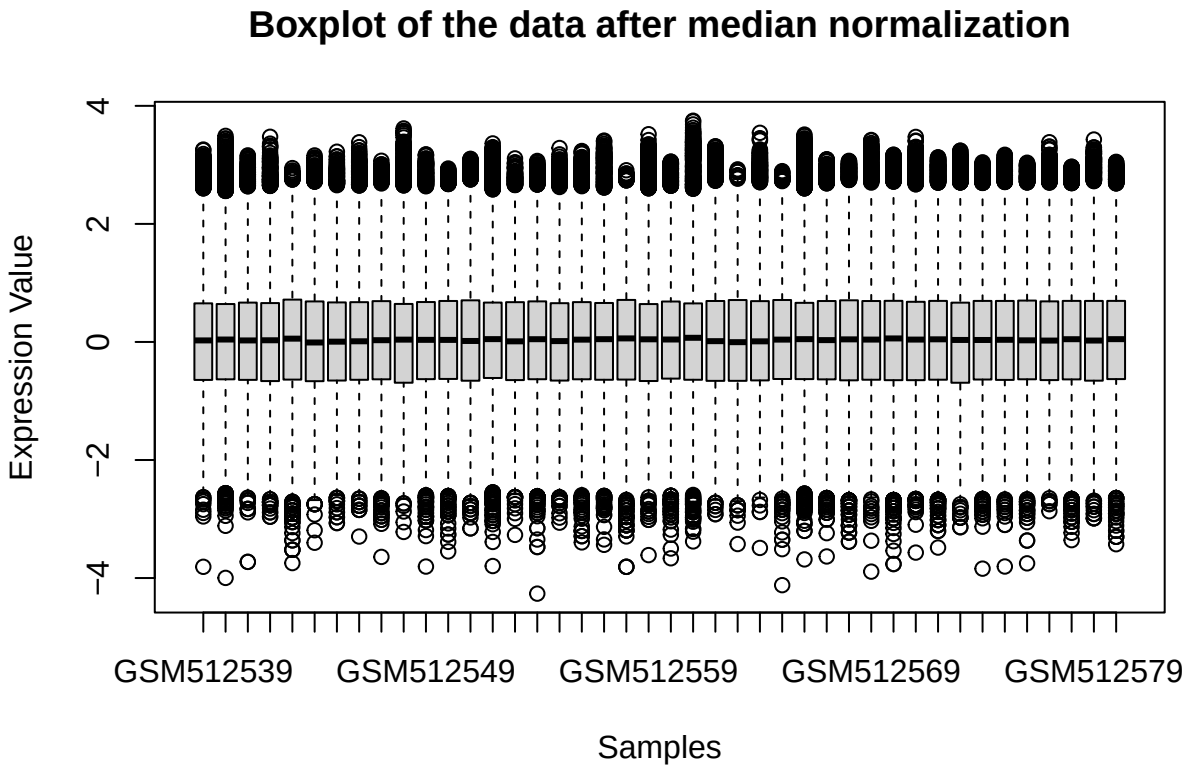


Figure 1: Boxplot of the data after applying a logarithmic transformation



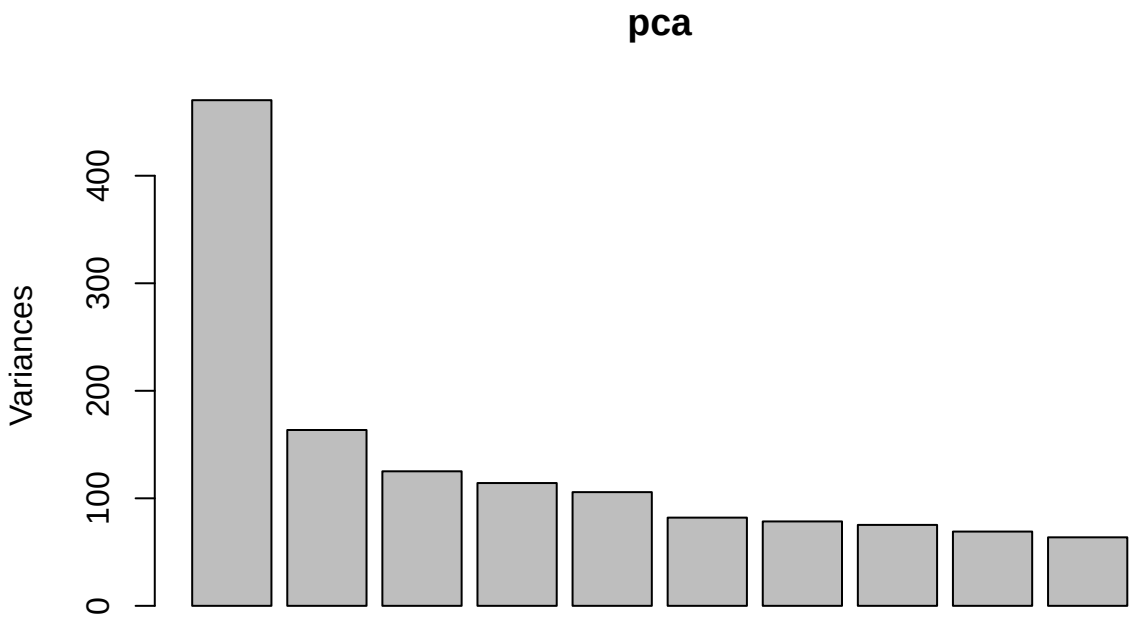
PCA

PCA is a dimensionality reduction technique that allows to condense thousands of dimensions into just two or three.For the dataset's samples, the PCA scores display the coordinates in relation to these additional dimensions.

```
pca <- prcomp(t(normalized.log.ex))

summary(pca)
```

```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5     PC6     PC7     PC8     PC9
## Standard deviation     21.6856 12.78817 11.18542 10.68629 10.28318 9.05796 8.85994 8.67632 8.31063 7
## Proportion of Variance  0.1798  0.06253  0.04783  0.04366  0.04043 0.03137 0.03001 0.02878 0.02641 0
## Cumulative Proportion   0.1798  0.24232  0.29016  0.33382  0.37425 0.40561 0.43563 0.46441 0.49081 0
##                            PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19    PC20   PC2
## Standard deviation     7.64884 7.42525 7.35447 7.25191 7.12698 7.03899 6.96990 6.86700 6.81828 6.726
## Proportion of Variance 0.02237 0.02108 0.02068 0.02011 0.01942 0.01894 0.01857 0.01803 0.01777 0.017
## Cumulative Proportion  0.56081 0.58189 0.60257 0.62267 0.64209 0.66104 0.67961 0.69764 0.71541 0.732
##                            PC24    PC25    PC26    PC27    PC28    PC29    PC30    PC31    PC32    PC
## Standard deviation     6.51084 6.39868 6.39565 6.35069 6.16713 6.14788 6.07658 6.01840 5.86441 5.806
## Proportion of Variance 0.01621 0.01565 0.01564 0.01542 0.01454 0.01445 0.01412 0.01385 0.01315 0.012
## Cumulative Proportion  0.78278 0.79843 0.81407 0.82949 0.84403 0.85848 0.87260 0.88645 0.89960 0.912
##                            PC36    PC37    PC38    PC39    PC40    PC41      PC42
## Standard deviation     5.46227 5.35021 5.28516 5.20027 5.10709 5.01262 4.456e-14
## Proportion of Variance 0.01141 0.01094 0.01068 0.01034 0.00997 0.00961 0.000e+00
## Cumulative Proportion  0.94846 0.95940 0.97008 0.98042 0.99039 1.00000 1.000e+00
```

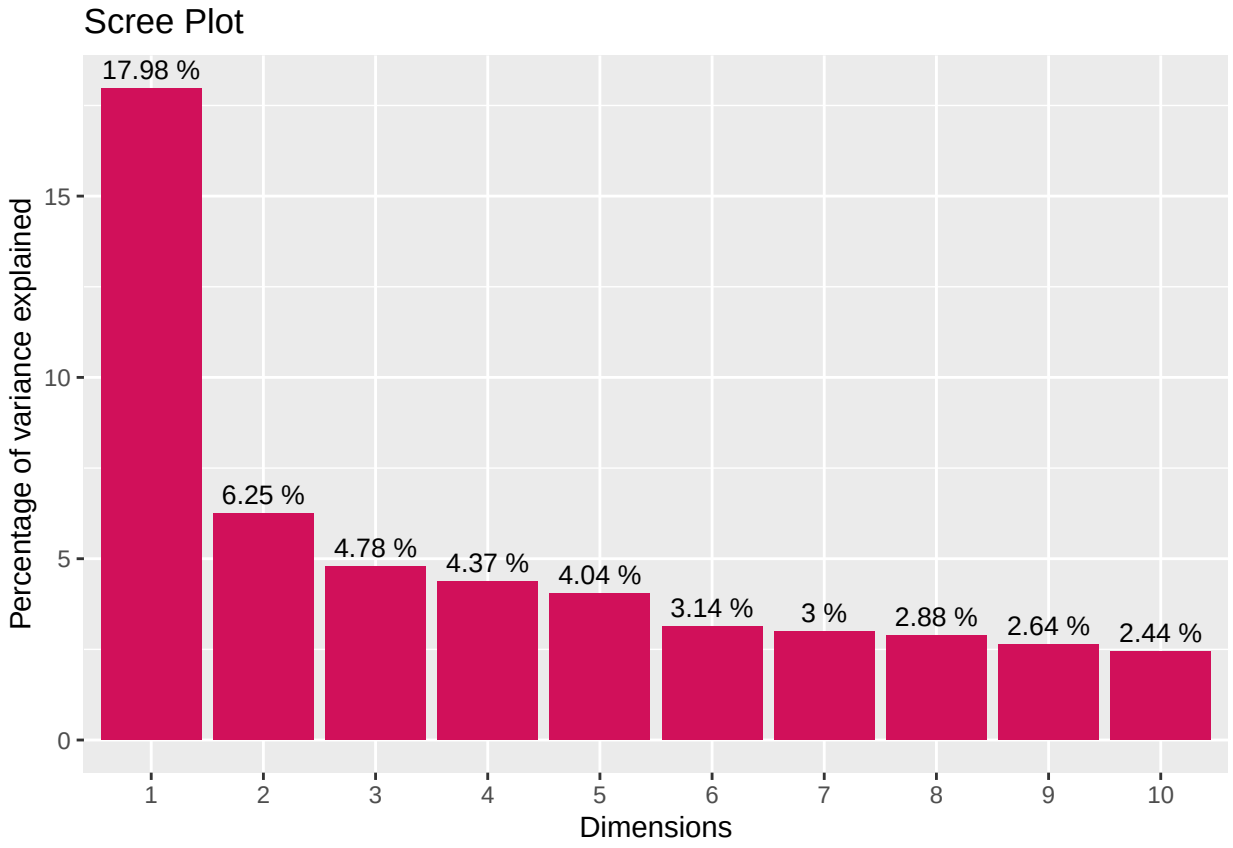
```
screeplot(pca)
```



To get the summary of the PCA and the plot showing the variance explained by the first 10 components, it is possible to use the functions commented in the chunks above.

However, using `ggplot2` and `factoextra` packages is possible to get a more concise and informative plot reporting the same information.

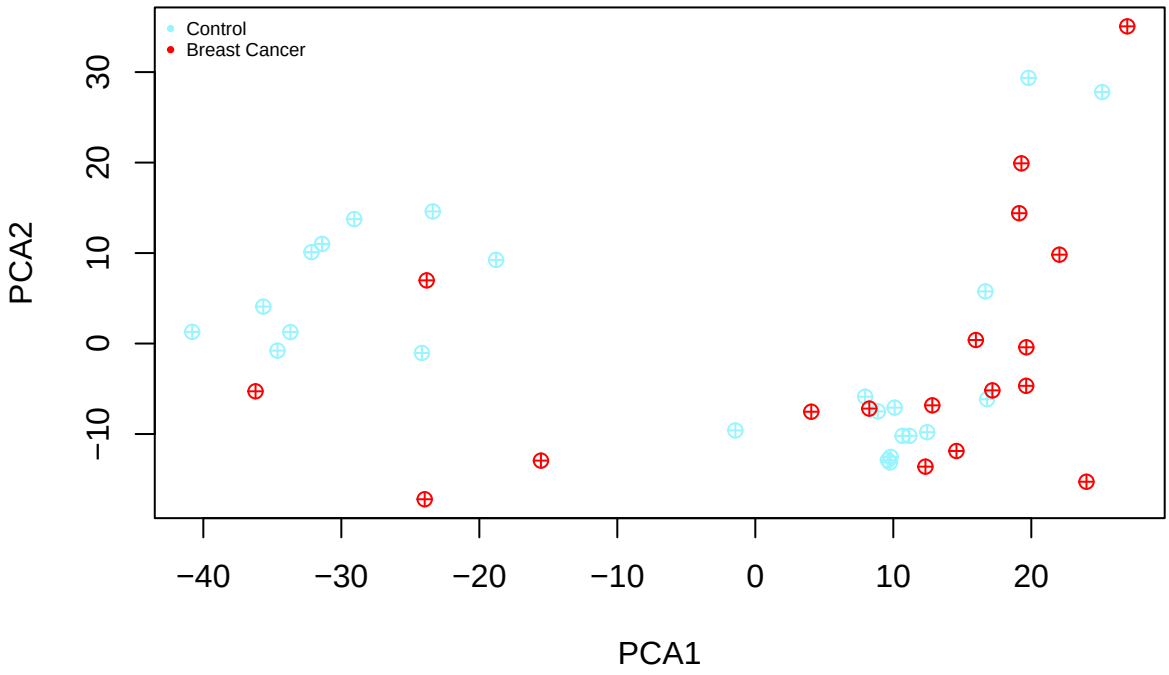
```
pcaVar <- get_eig(pca)
pcaVar <- pcaVar$variance.percent[1:10]
screeDf <- data.frame("Dimensions" = as.factor(seq(1,10)),
                      "Percentages" = pcaVar,
                      "Labels" = paste(round(pcaVar, 2), "%"))

p <- ggplot(data = screeDf, aes(x=Dimensions, y=Percentages))+
  geom_bar(stat = "identity", fill = "#d1105a")+
  geom_text(aes(label=Labels), vjust=-0.5, color="black", size=3.6)+
  ggtitle("Scree Plot")+
  ylab("Percentage of variance explained")+
  scale_x_discrete(labels = as.factor(seq(1,10)))
p
```



The scree plot shows that the first dimensions on the left are the more important because the percentage of variance explained by them is higher. The remaining principal components account for a very small proportion of the variability and are probably unimportant.

Let's try to plot the PCA, looking if we can see a separation between Control and Breast Cancer groups.

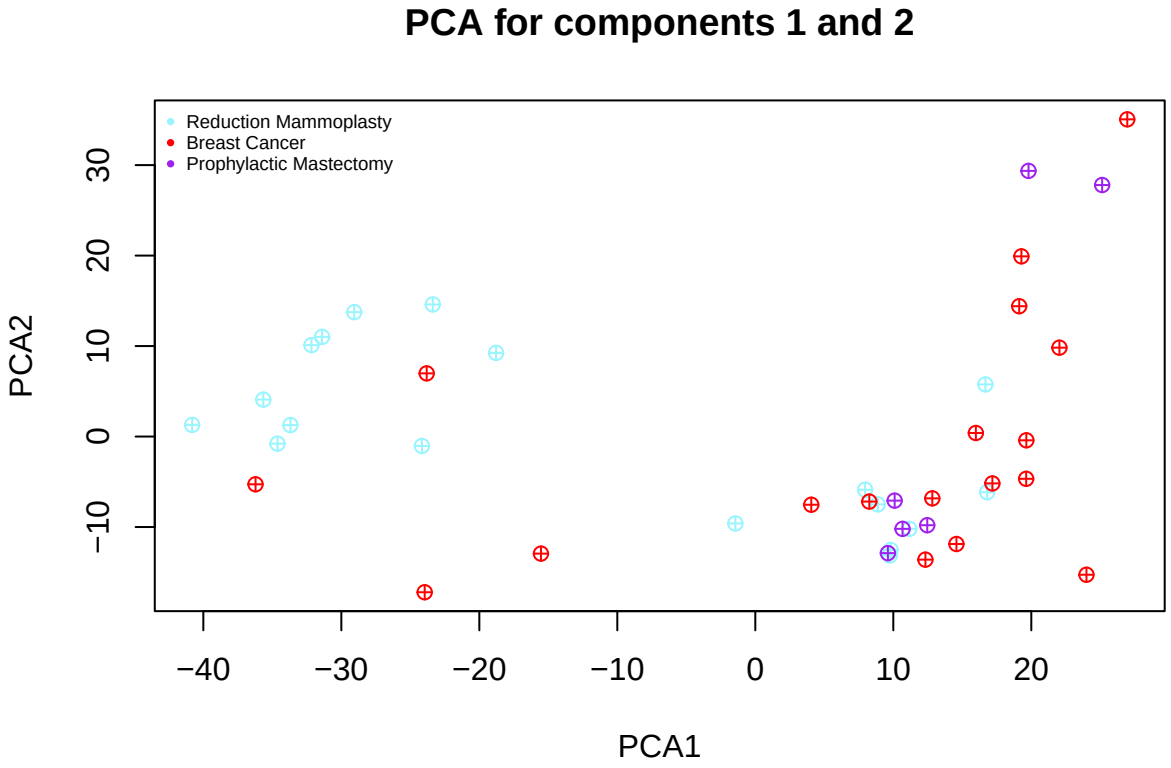
```
# draw PCA plot control VS breast cancer
group <- c(rep("cadetblue1",18), rep("red",18), rep("cadetblue1",6) )
plot(pca$x[,1], pca$x[,2], xlab="PCA1", ylab="PCA2", main="PCA for components 1 and 2", type="p", pch=1
text(pca$x[,1], pca$x[,2], rownames(pca$data), cex=0.75)
legend("topleft", col=c("cadetblue1","red"), legend = c("Control", "Breast Cancer"),
    pch = 20, bty='n', cex=.55)
```



Let's try to add the control subtypes. The vector group used in the PCA plot is based on the data. The samples corresponding to the colors are the following:

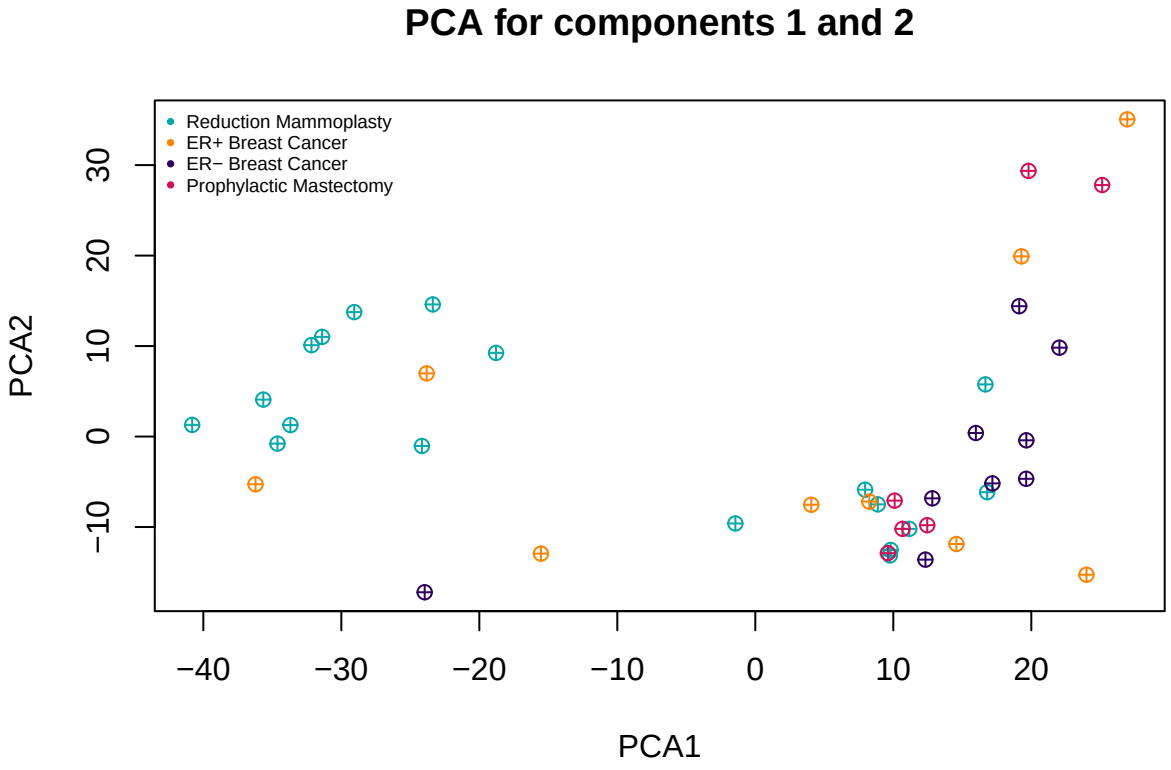
- **Light blue**: reduction mammoplasty (RM) breast epithelium samples
- **Red**: histologically normal (HN) epithelial samples from breast cancer patient
- **Purple**: histologically normal breast epithelium (NlEpi) from prophylactic mastectomy patient samples

```
# draw PCA plot
group <- c(rep("cadetblue1",18), rep("red",18), rep("purple",6) ) # vector of colors based on the order
plot(pca$x[,1], pca$x[,2], xlab="PCA1", ylab="PCA2", main="PCA for components 1 and 2", type="p", pch=1
text(pca$x[,1], pca$x[,2], rownames(pca$data), cex=0.75)
legend("topleft", col=c("cadetblue1","red","purple"), legend = c("Reduction Mammoplasty", "Breast Cancer
    pch = 20, bty='n', cex=.55)
```



Then, I try to see if there is a separation also inside different types of Breast Cancer.

```
# draw PCA plot with all subtypes
group <- c(rep(my_colors[7],18), rep(my_colors[4],9), rep(my_colors[1],9), rep(my_colors[6],6) ) # vect
plot(pca$x[,1], pca$x[,2], xlab="PCA1", ylab="PCA2", main="PCA for components 1 and 2", type="p", pch=1
text(pca$x[,1], pca$x[,2], rownames(pca$data), cex=0.75)
legend("topleft", col=c(my_colors[7],my_colors[4],my_colors[1],my_colors[6]), legend = c("Reduction Mamm
    pch = 20, bty='n', cex=.55)
```



Interactive PCA plot

Let's try to explore an interactive PCA plot.

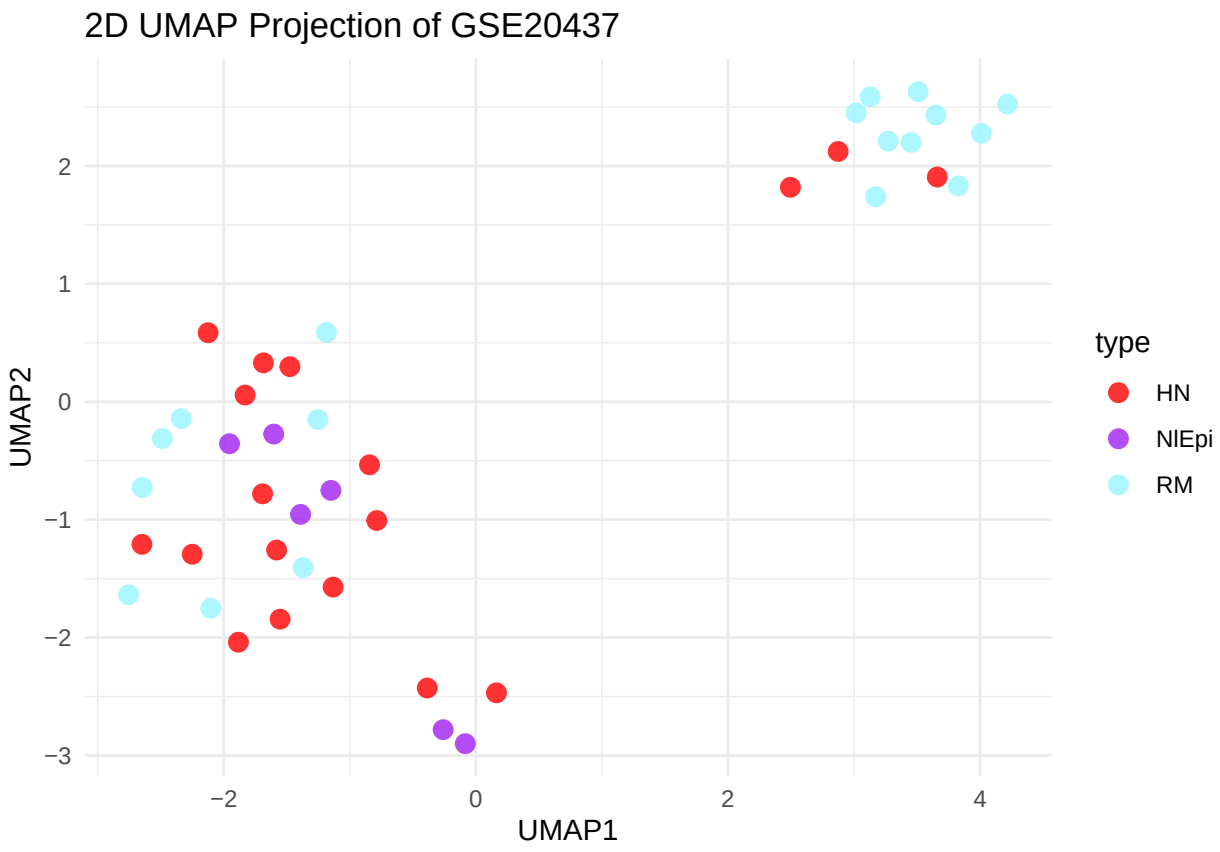
```
components<-pca[["x"]]
components<-data.frame(components)
type<-c(rep("RM", 18), rep("HN",18), rep("NlEpi",6))
components<-cbind(components, type )

fig <- plot_ly(components, x=~PC1, y=~PC2,
               color=type,colors=c('cadetblue1', 'red','purple'),
               type='scatter',mode='markers')
fig
```

```
fig2 <- plot_ly(components, x=~PC1, y=~PC2, z=~PC3,
                color=type, colors=c('cadetblue1', 'red','purple'),
                mode='markers', marker = list(size = 4))
fig2
```

```
fig3 <- plot_ly(components, x=~PC1, y=~PC3,
                color=type, colors=c('cadetblue1', 'red','purple'),
                type='scatter',mode='markers')
fig3
```

```
set.seed(123)
umap_result <- umap(t(normalized.log.ex))

umap_df <- as.data.frame(umap_result$layout)
colnames(umap_df) <- c("UMAP1", "UMAP2")

# Add sample types to the UMAP data frame
```

```r
umap_df$type <- c(rep("RM", 18), rep("HN",18), rep("NlEpi",6))

ggplot(umap_df, aes(x = UMAP1, y = UMAP2, color = type)) +
  geom_point(size = 3, alpha = 0.8) +
  scale_color_manual(values = c("RM" = "cadetblue1",
                                "HN" = "red",
                                "NlEpi" = "purple")) +
  theme_minimal() +
  labs(title = "2D UMAP Projection of GSE20437", x = "UMAP1", y = "UMAP2")
```



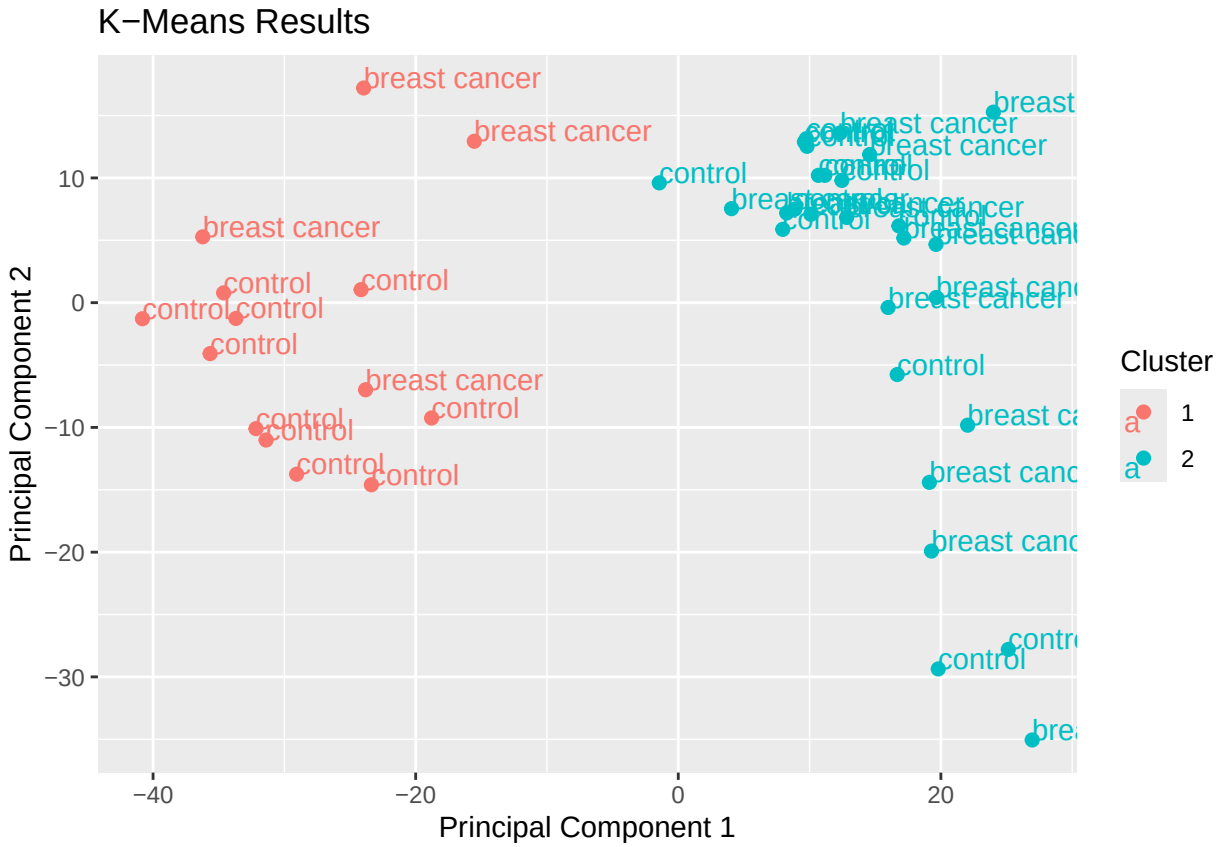
Clustering

K-means

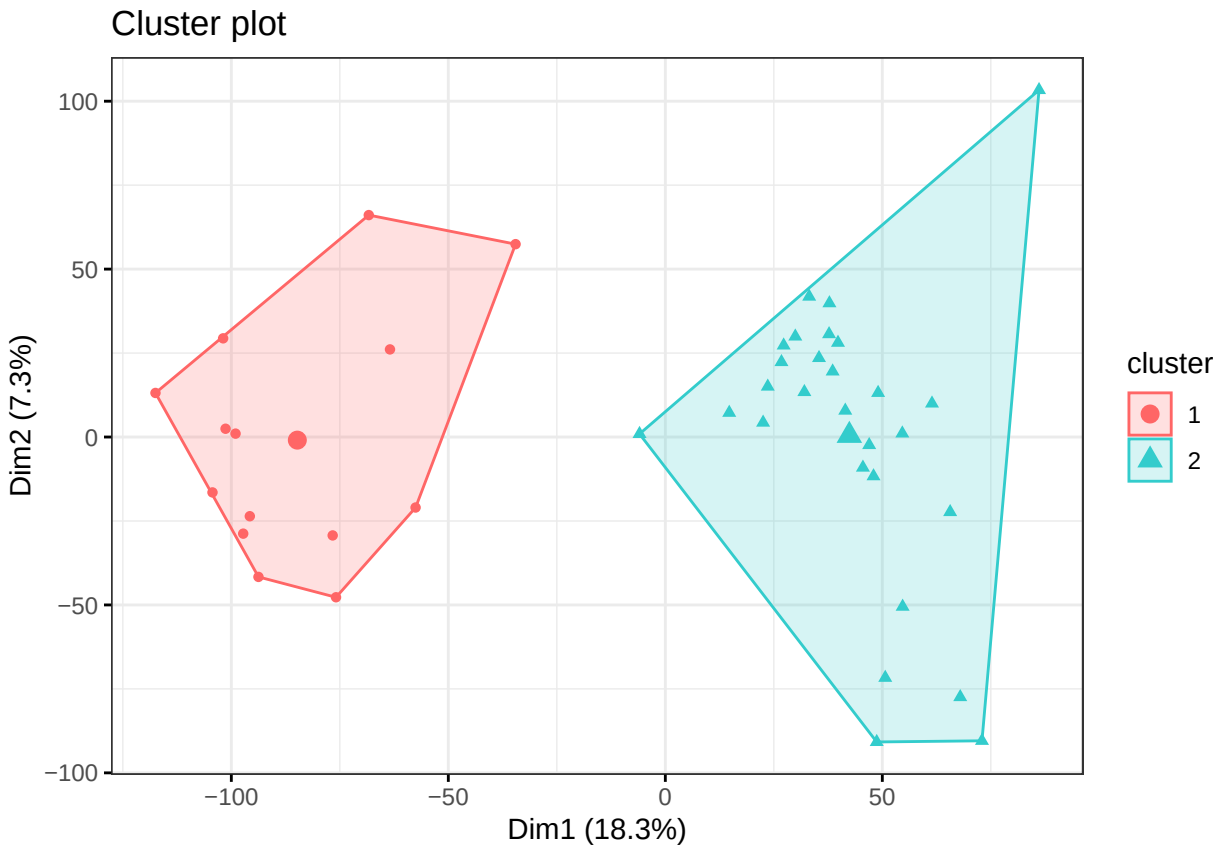
```r
set.seed(1)
k <- 2 # number of clusters

kmeans_result <- kmeans(t(normalized.log.ex),k)
table(kmeans_result$cluster) # tells how many samples were assigned to each cluster
```

```
##
##  1  2
## 14 28
```

```r
plot(kmeans_result, data=t(normalized.log.ex)) + geom_text(aes(label=metadata$disease.state.ch1),hjust=
```




```
fviz_cluster(kmeans_result, data = t(normalized.log.ex),
             palette = c("#FF6666", "#33cccc"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
             )
```

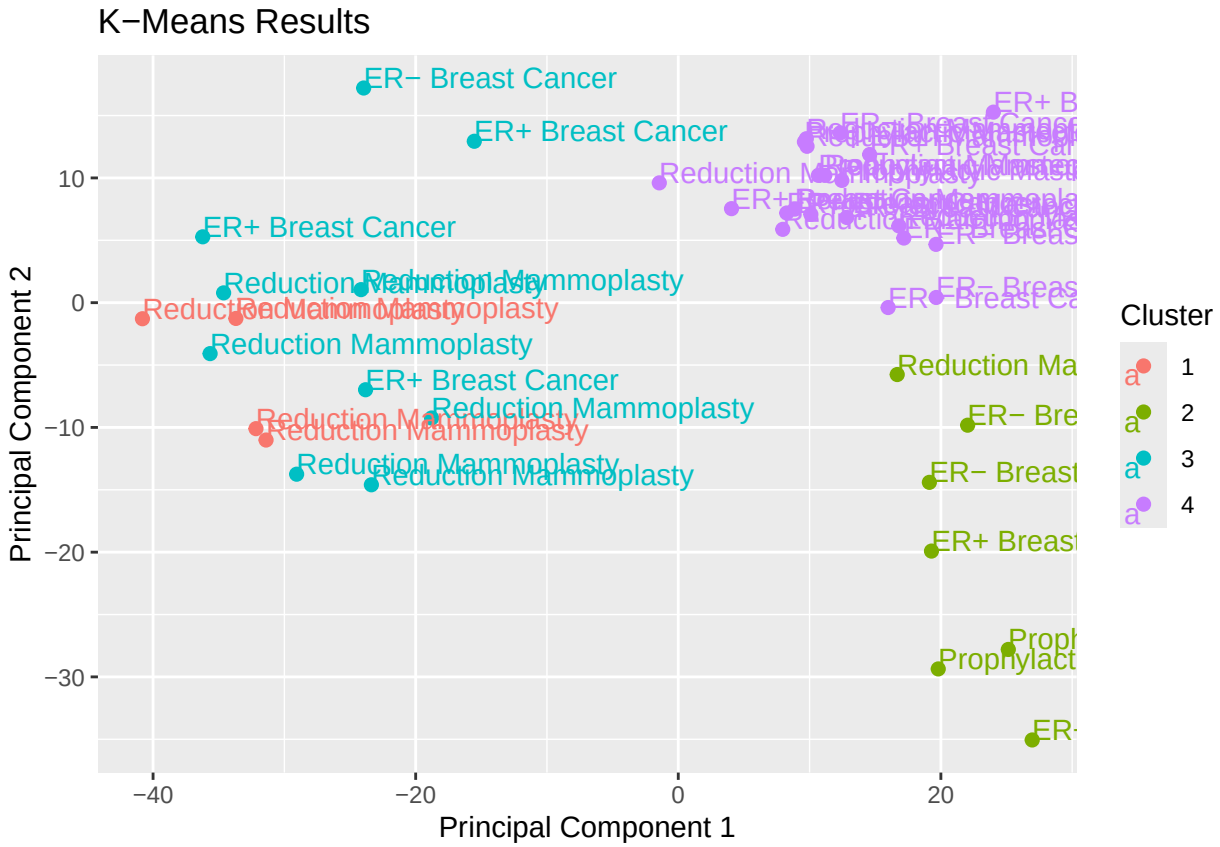


Let's try increasing the number of clusters.

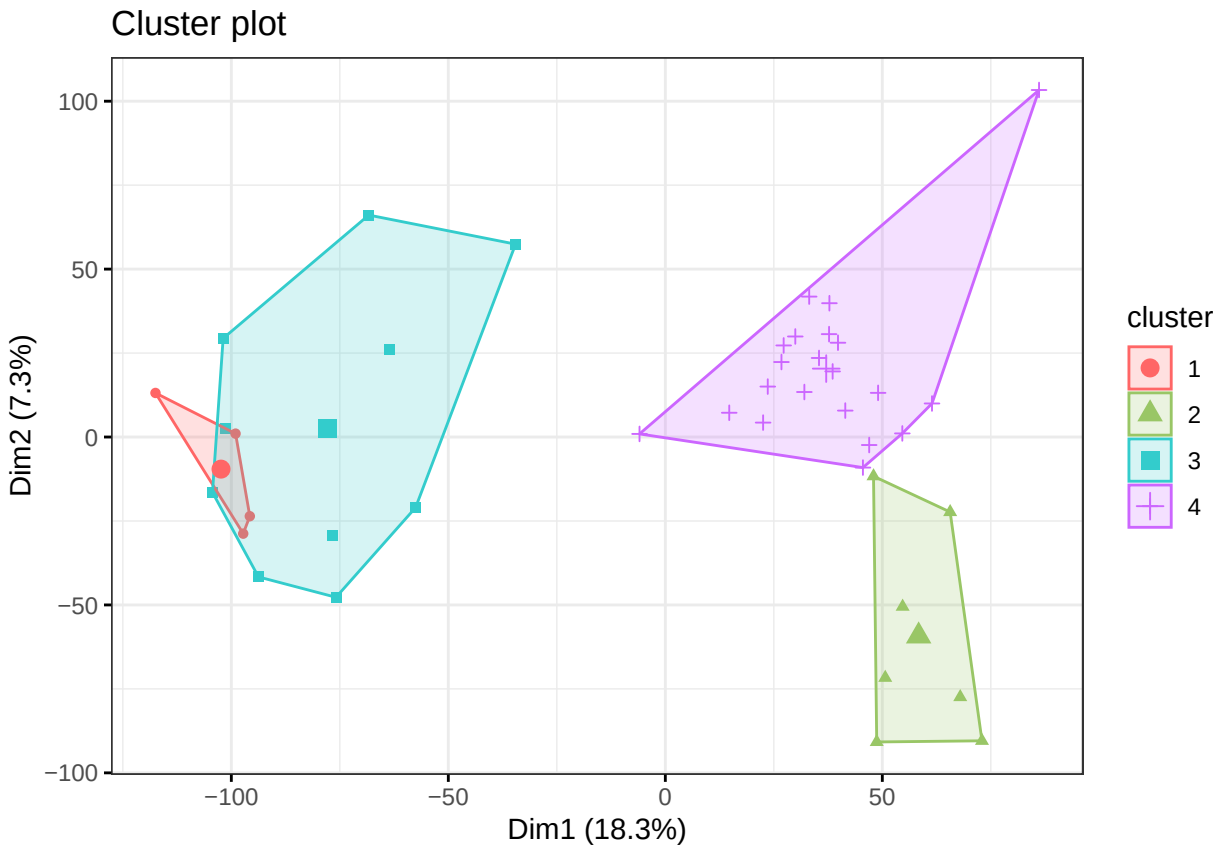
```
set.seed(1)
k <- 4 # number of clusters

kmeans_result <- kmeans(t(normalized.log.ex),k)
table(kmeans_result$cluster) # tells how many samples were assigned to each cluster
```

```
##
##  1  2  3  4
##  4  7 10 21
```

```
plot(kmeans_result, data=t(normalized.log.ex)) + geom_text(aes(label=metadata$specimen.ch1),hjust=0,vjus
```

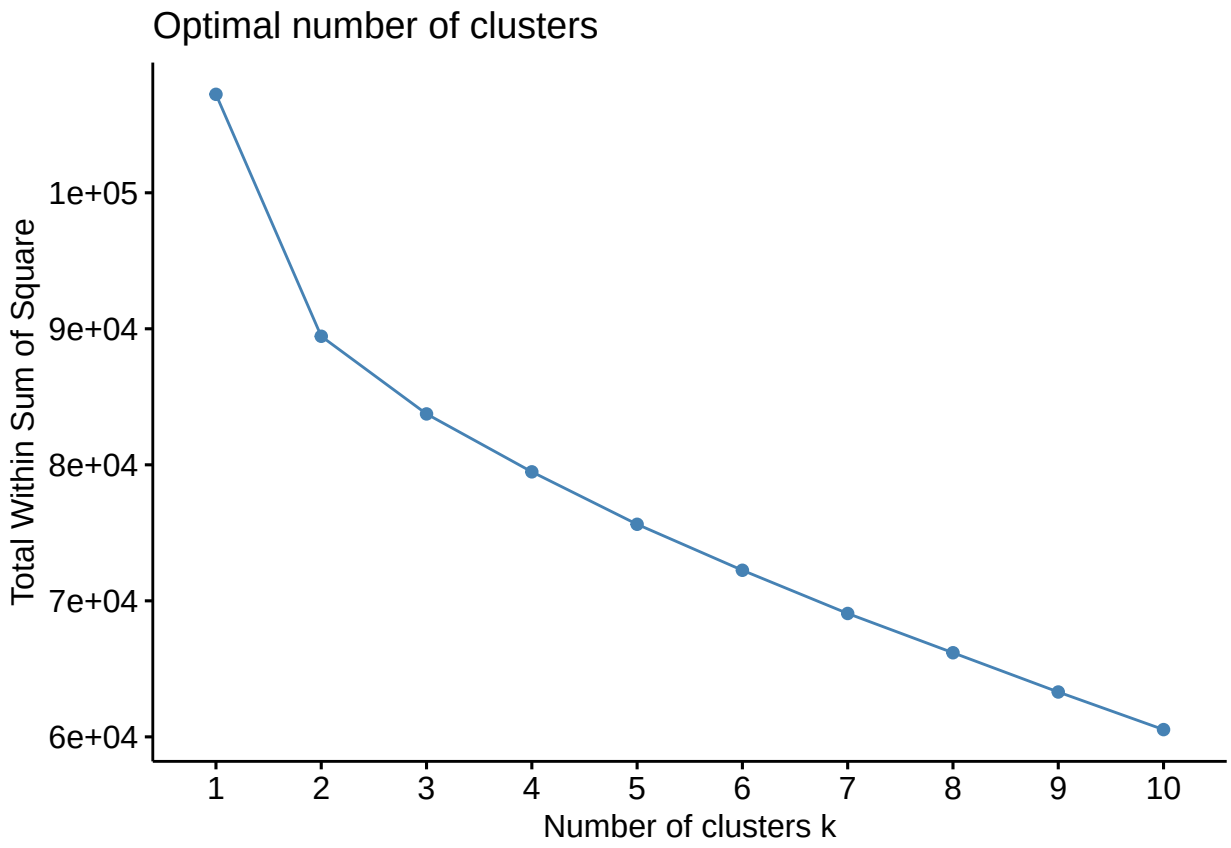



```
fviz_cluster(kmeans_result, data = t(normalized.log.ex),
             palette = c("#FF6666", "#99C666", "#33cccc", "#cc66ff"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
             )
```



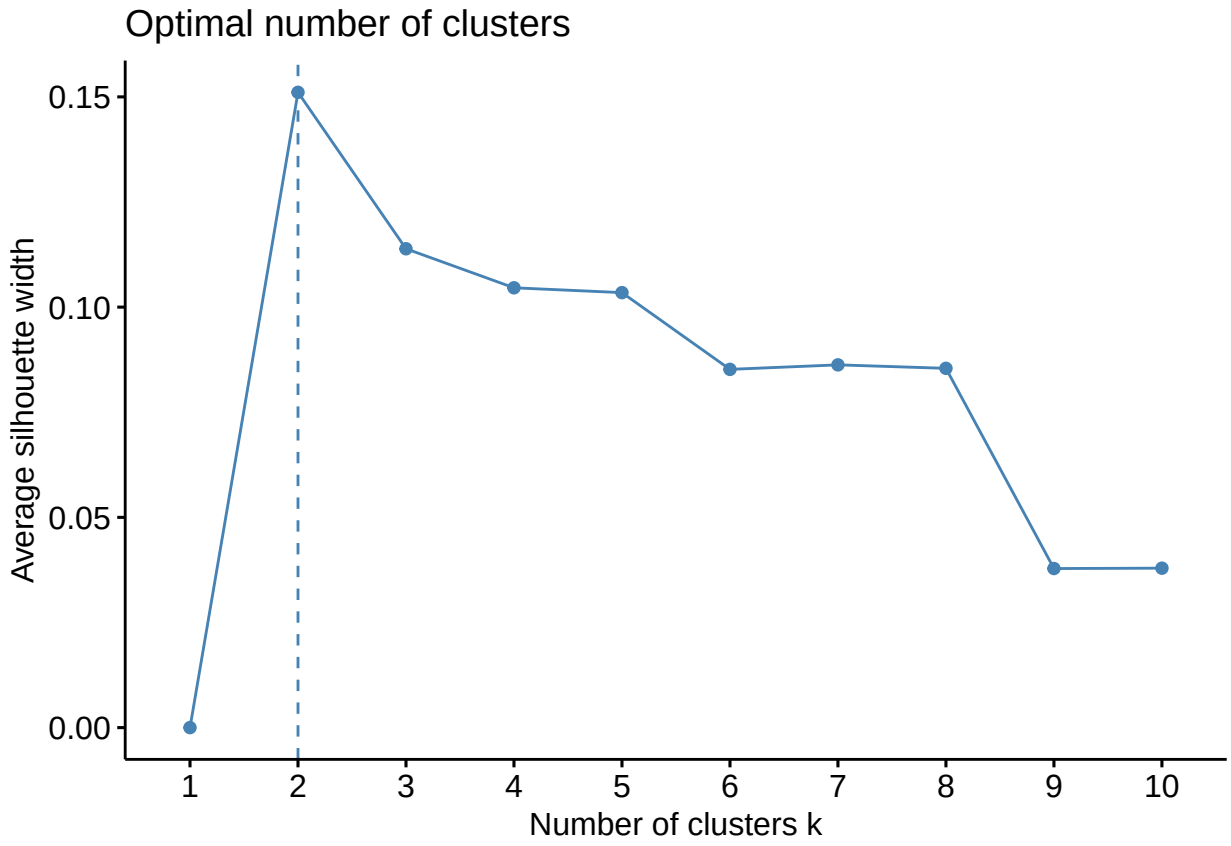
Hierarchical

```
# Elbow method
fviz_nbclust(t(normalized.log.ex), FUN = hcut, method = "wss")
```




```
## seems setting number of clusters equal to 2
```

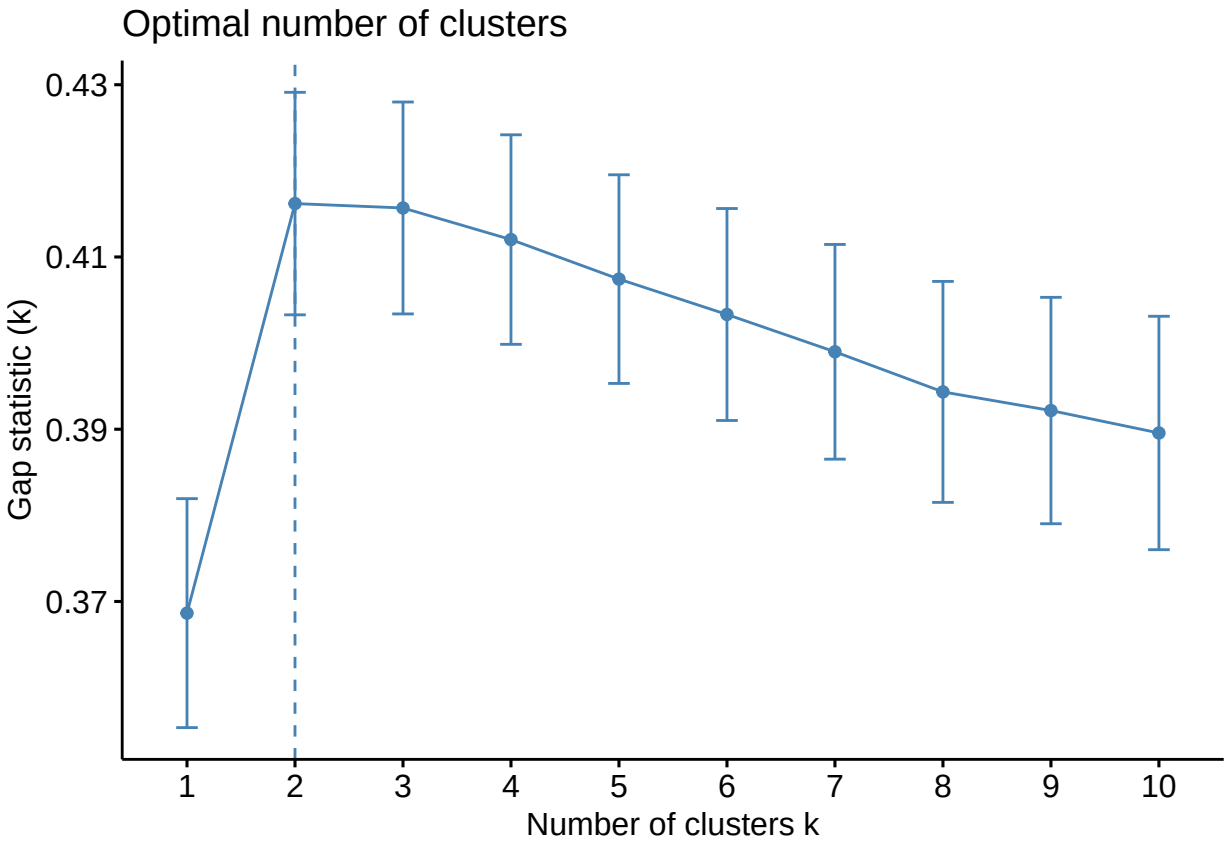
```
# Silhouette method
fviz_nbclust(t(normalized.log.ex), FUN = hcut, method = "silhouette")
```




```
## seems setting number of clusters equal to 2
```

We use the Gap statistic to calculate the goodness of clustering.

```
# Gap Statistic Method
gap_stat <- clusGap(t(normalized.log.ex), FUN = hcut, nstart = 25, K.max = 10, B = 50)
# K.max -> the maximum number of clusters to consider
# B -> number of Monte Carlo samples

fviz_gap_stat(gap_stat)
```




```
hc_result <- dist(t(normalized.log.ex)) %>% hclust(method = "ave")
hc_result2<- dist(t(normalized.log.ex), method="euclidean") %>% hclust( method = "complete")
hc_result3 <- dist(t(normalized.log.ex)) %>% hclust(method = 'single')

k_hc <- 2 # optimal number of clusters

groups <- cutree(hc_result, k=k_hc)
table(groups,type)
```

```
##       type
## groups HN NlEpi RM
##      1 17     6 18
##      2  1     0  0
```

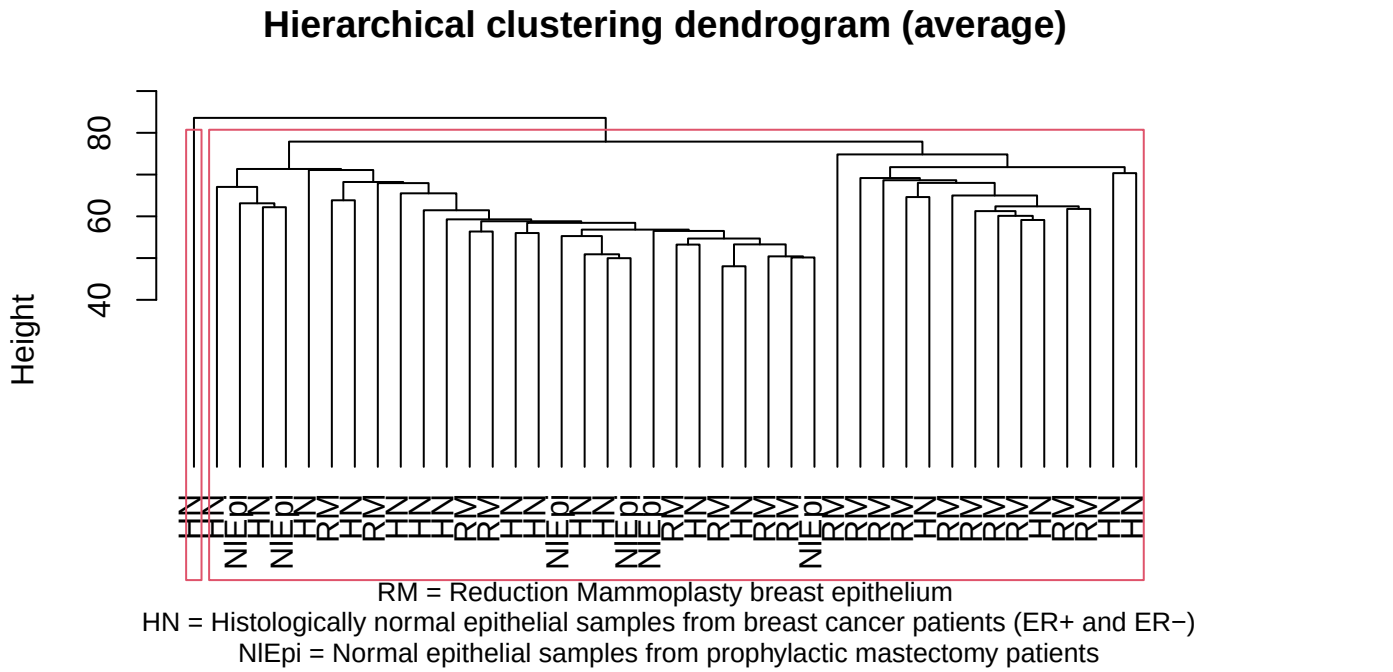
```
groups2<-cutree(hc_result2, k=k_hc)
table(groups2,type)
```

```
##        type
## groups2 HN NlEpi RM
##       1  4     0 10
##       2 14     6  8
```

```
groups3 <- cutree(hc_result3,k=k_hc)
table(groups3,type)
```

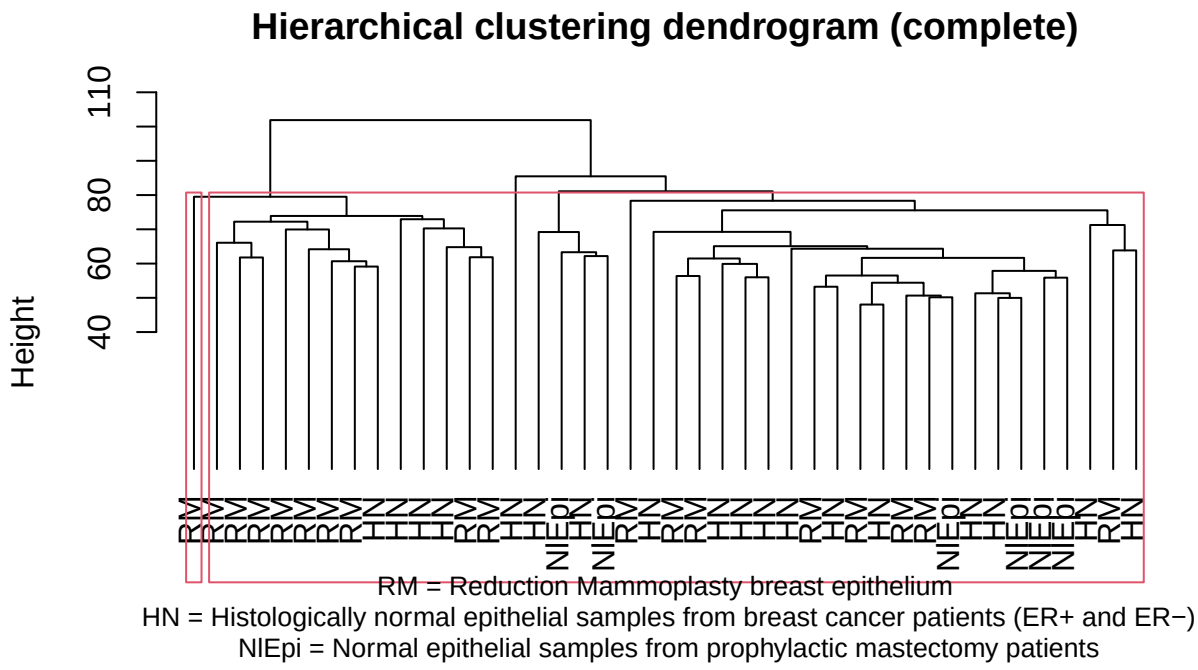
```
##        type
## groups3 HN NlEpi RM
##       1 17     6 18
##       2  1     0  0
```

```r
# Set up layout with extra space below the plot
par(mar = c(6, 4, 4, 2))   # increase bottom margin
par(xpd = TRUE)            # allow text outside plot region

# Plot dendrogram
plot(hc_result, hang = -1, labels = type,
     main = 'Hierarchical clustering dendrogram (average)')
rect.hclust(hc_result, k = 2, border = 2) # red boxes to show groups

# Add custom legend box below the plot
text(x = mean(par("usr")[1:2]), y = par("usr")[3] - 10,
     labels = paste(
       "RM = Reduction Mammoplasty breast epithelium\n",
       "HN = Histologically normal epithelial samples from breast cancer patients (ER+ and ER-)\n",
       "NlEpi = Normal epithelial samples from prophylactic mastectomy patients"
     ),
     cex = 0.8, adj = 0.5)
```

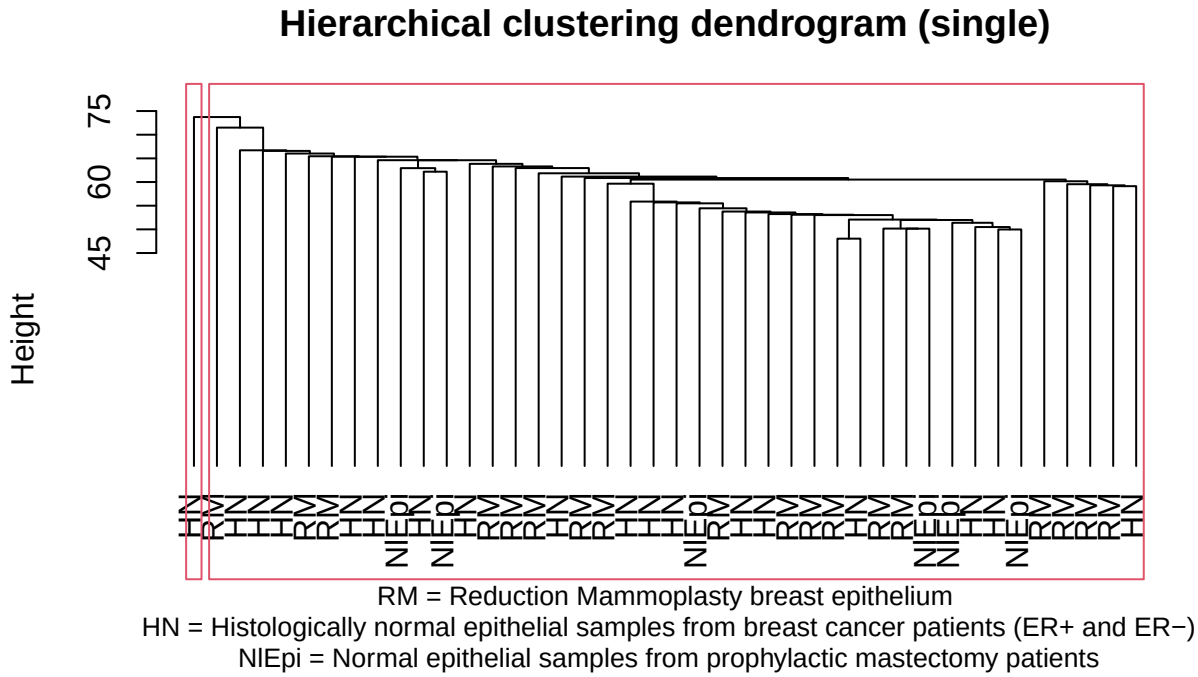



```r
# Set up layout with extra space below the plot
par(mar = c(6, 4, 4, 2))   # increase bottom margin
par(xpd = TRUE)            # allow text outside plot region

# Plot dendrogram
plot(hc_result2, hang = -1, labels = type,
     main = 'Hierarchical clustering dendrogram (complete)')
rect.hclust(hc_result, k = 2, border = 2) # red boxes to show groups

# Add custom legend box below the plot
text(x = mean(par("usr")[1:2]), y = par("usr")[3] - 10,
```

```
     labels = paste(
       "RM = Reduction Mammoplasty breast epithelium\n",
       "HN = Histologically normal epithelial samples from breast cancer patients (ER+ and ER-)\n",
       "NlEpi = Normal epithelial samples from prophylactic mastectomy patients"
     ),
     cex = 0.8, adj = 0.5)
```

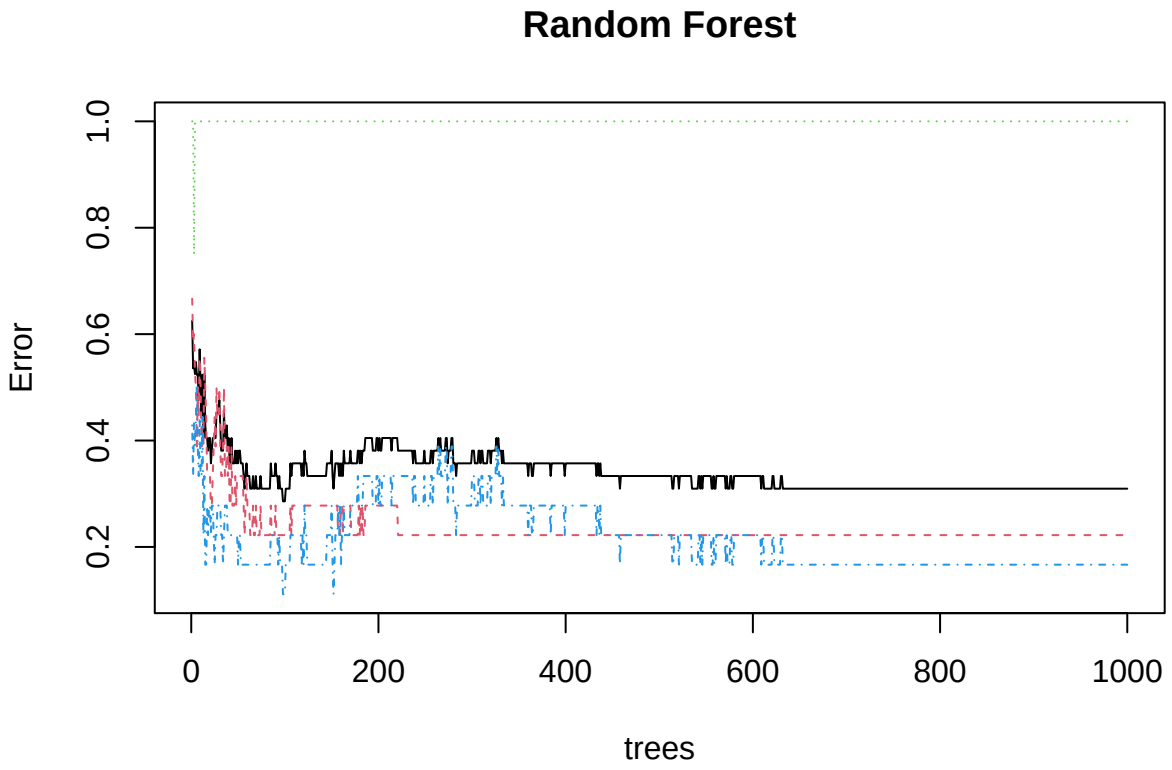



```
# Set up layout with extra space below the plot
par(mar = c(6, 4, 4, 2))   # increase bottom margin
par(xpd = TRUE)            # allow text outside plot region

# Plot dendrogram
plot(hc_result3, hang = -1, labels = type,
     main = 'Hierarchical clustering dendrogram (single)')
rect.hclust(hc_result, k = 2, border = 2) # red boxes to show groups

# Add custom legend box below the plot
text(x = mean(par("usr")[1:2]), y = par("usr")[3] - 10,
     labels = paste(
       "RM = Reduction Mammoplasty breast epithelium\n",
       "HN = Histologically normal epithelial samples from breast cancer patients (ER+ and ER-)\n",
       "NlEpi = Normal epithelial samples from prophylactic mastectomy patients"
     ),
     cex = 0.8, adj = 0.5)
```



Random forest

```
set.seed(1234)
rf <- randomForest(x=t(normalized.log.ex), y=as.factor(type), ntree=1000)
plot(rf, main = "Random Forest")
```



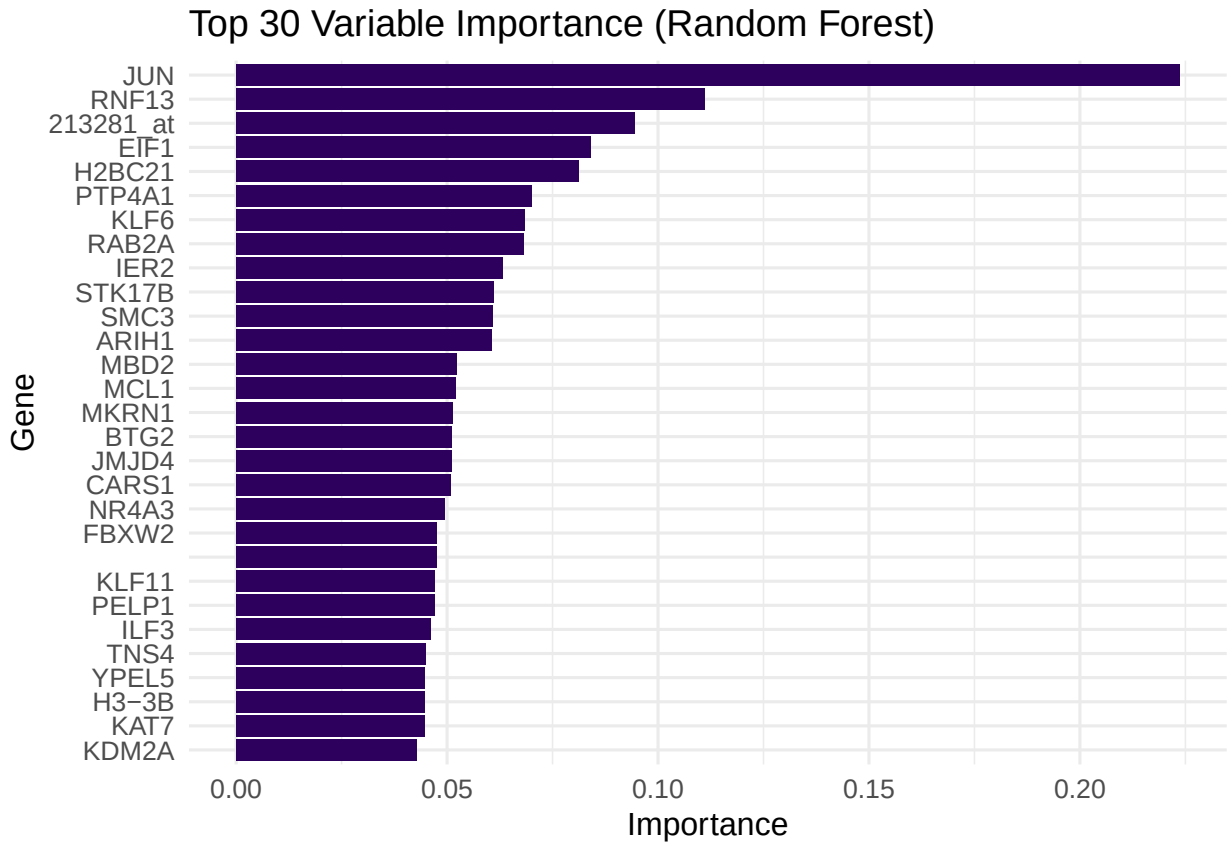
The plot above shows how the error rate changes according to the number of trees. This allows to track how model accuracy improves (or doesn't) as more trees are added. The black line represent the overall OOB (out-of-bag) error rate, while the other colored lines report the class specific error rates.

```
# a trivial test
predict(rf, t(normalized.log.ex[, 1:5]))
```

```
## GSM512539 GSM512540 GSM512541 GSM512542 GSM512543
##        RM        RM        RM        RM        RM
## Levels: HN NlEpi RM
```

```
# Get importance values
rf_importance <- importance(rf)
# Extract probe IDs in order of importance
ordered_probes <- rownames(rf_importance)[order(rf_importance[, 1], decreasing = TRUE)]
# Ensure correct mapping exists
probe_to_symbol <- annotLookup[, c("affy_hg_u133_plus_2", "external_gene_name")]
colnames(probe_to_symbol) <- c("probe", "symbol")
probe_symbols <- setNames(probe_to_symbol$symbol, probe_to_symbol$probe)

top_vars <- head(ordered_probes, 30)  # Show top 30 features
imp_df <- data.frame(
  probe = top_vars,
  importance = rf_importance[top_vars, 1],
  gene = ifelse(top_vars %in% names(probe_symbols), probe_symbols[top_vars], top_vars)
)

ggplot(imp_df, aes(x = reorder(gene, importance), y = importance)) +
  geom_bar(stat = "identity", fill = "#2e005d") +
  coord_flip() +
  labs(title = "Top 30 Variable Importance (Random Forest)", x = "Gene", y = "Importance") +
```

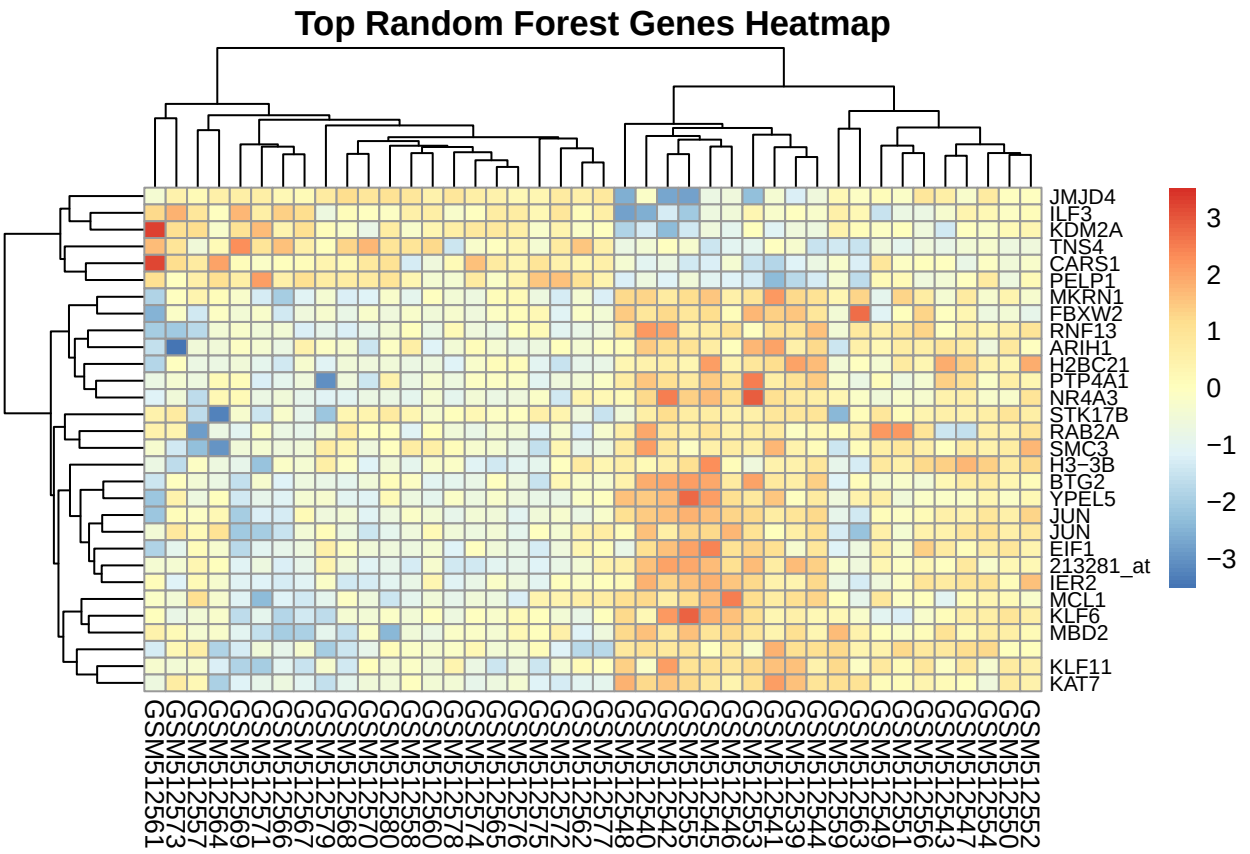
```
  theme_minimal(base_size = 12)
```



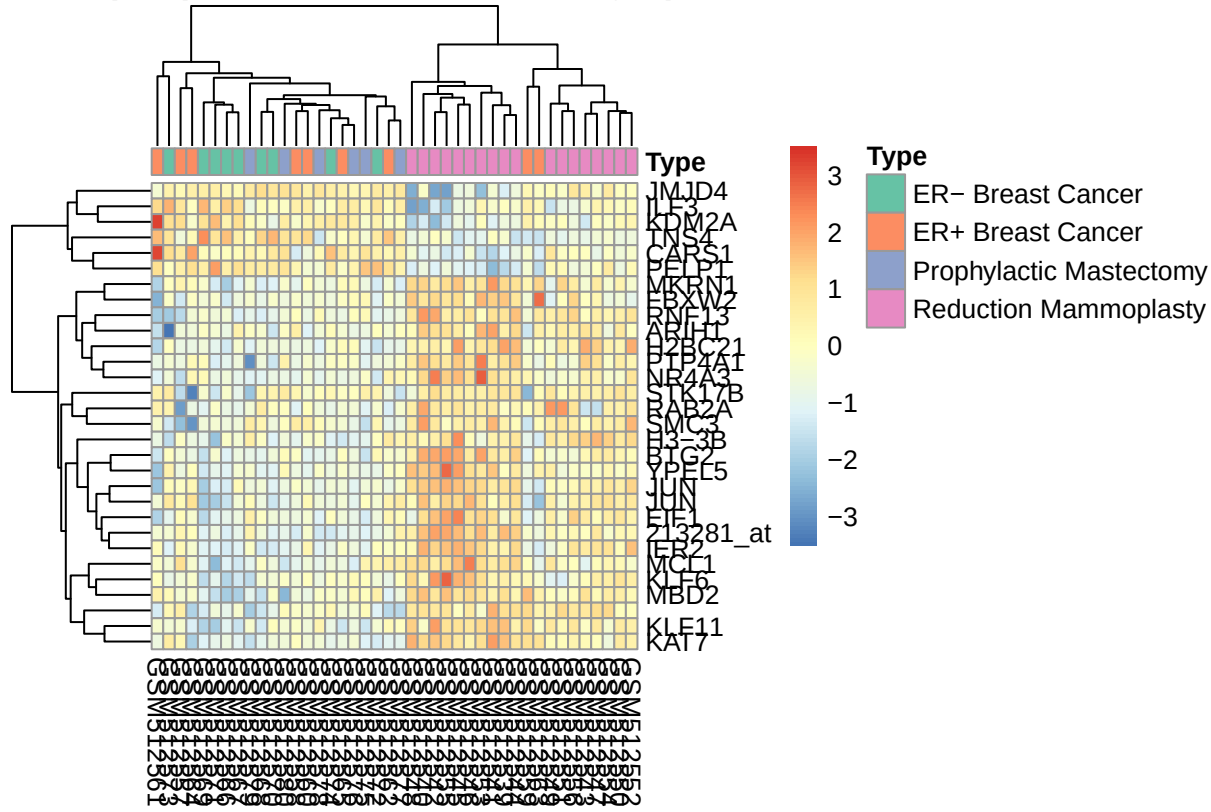
Heatmap

This is optional. Not suggested to include heatmap in the report, because at the end of the project there will be too many graphs and this is not a valuable one.

```
# Select top N important probes
top_n <- 30
top_probes <- imp_df$probe[1:top_n]

heatmap_data <- normalized.log.ex[top_probes, ]
# Add gene symbols
gene_symbols <- imp_df$gene[1:top_n]
rownames(heatmap_data) <- gene_symbols

pheatmap(heatmap_data,
         scale = "row",  # normalize expression within each gene
         clustering_distance_rows = "euclidean",
         clustering_distance_cols = "euclidean",
         clustering_method = "complete",
         show_rownames = TRUE,
         show_colnames = TRUE,
         fontsize_row = 8,
         main = "Top Random Forest Genes Heatmap")
```




```
# Ensure sample labels are aligned with expression data
sample_types <- metadata$type
names(sample_types) <- colnames(normalized.log.ex)

# Subset for current samples
sample_types <- sample_types[colnames(heatmap_data)]

# Set rownames to match sample IDs
rownames(metadata) <- metadata$geo_accession
# Create annotation dataframe
annotation_col <- data.frame(Type = metadata[colnames(heatmap_data), "specimen.ch1"])
rownames(annotation_col) <- colnames(heatmap_data)

# Convert to factor
annotation_col$Type <- factor(annotation_col$Type)
type_levels <- levels(annotation_col$Type)
palette_colors <- brewer.pal(n = length(type_levels), name = "Set2")  # or "Dark2", "Paired", etc.
names(palette_colors) <- type_levels

ann_colors <- list(Type = palette_colors)

pheatmap(heatmap_data,
         scale = "row",
         annotation_col = annotation_col,
         annotation_colors = ann_colors,
         show_colnames = TRUE,
         main = "Heatmap Top Random Forest Genes by Specimen")
```



Feature selection