# Project
# Transplant kidney rejection
# High Dimensional Data Analysis

Jan Alexander* Annabel Vaessens† Steven Wallaert‡

8/4/2020

```
load('RejectionStatus.rda')
load('X_GSE21374.rda')
dim(RejectionStatus)
```

```
## [1] 282   2
```

```
dim(X_GSE21374)
```

```
## [1] 54675   282
```

```
GeneExpression <- scale(t(X_GSE21374))

GeneExpression <-
  GeneExpression[order(as.numeric(row.names(GeneExpression))), ]
```

```
## Warning in order(as.numeric(row.names(GeneExpression))): NAs introduced by
## coercion
```

```
RejectionStatus <-
  RejectionStatus[order(as.numeric(RejectionStatus$Patient_ID)), ]

all.equal(row.names(GeneExpression), as.character(RejectionStatus$Patient_ID))
```

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

## 1 Introduction

The data is loaded as presented in the assignment.

Three research questions are defined:

- How do the 54675 genes vary in terms of their gene expression levels? Is the variability associated with kidney rejection? (only to be answered in a data explorative manner).
- Which genes are differentially expressed between the two kidney rejection groups? You must control the False Discovery Rate at 10%.
- Can the kidney rejection be predicted from the gene expressions? What genes are most important in predicting the kidney transplant rejection? How well does the prediction model perform in terms of predicting rejection status?

*jan.alexander@ugent.be
†annabel.vaessens@vub.be
‡steven.wallaert@ugent.be

# 2 Data exploration

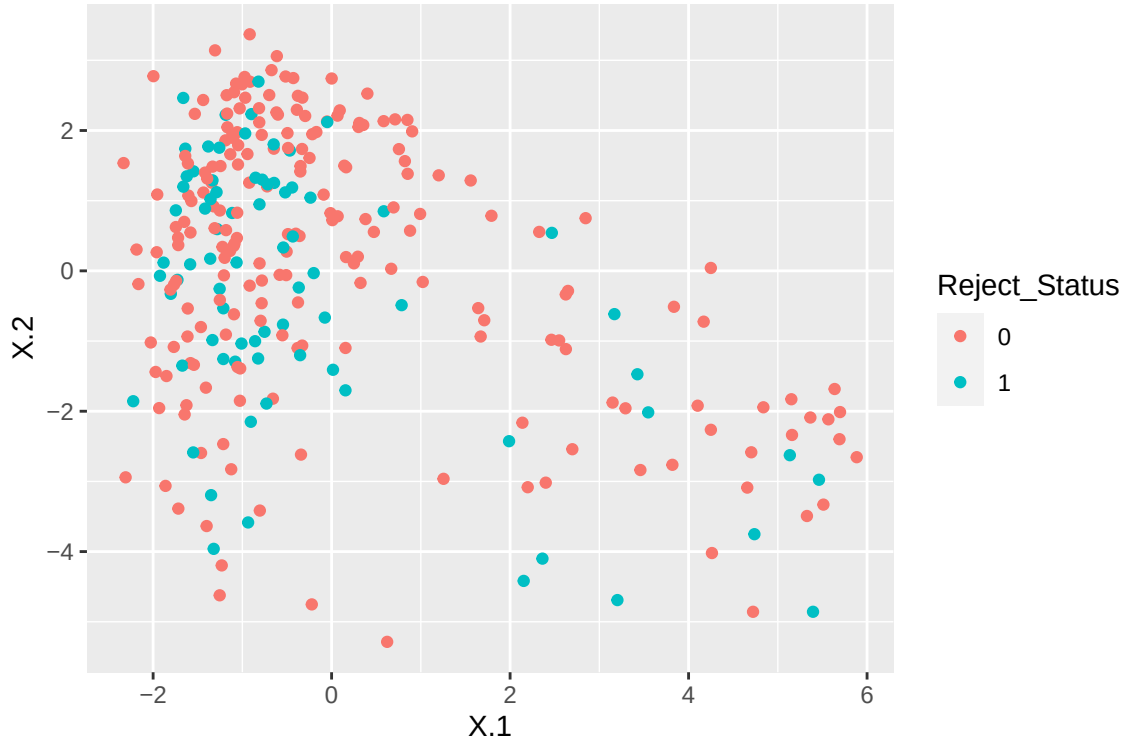
In the complete dataset, 27 % of the transplanted kidnesy were rejected.

## 2.1 Sparse principle components analysis

```
Gen_spc <- PMA::SPC(GeneExpression, K = 2, sumabsv = 2)
```

```
## 123456789101112131415161718192O
## 123456789101112131415161718192O
```

```
Uk <- Gen_spc$u ; Dk <- diag(Gen_spc$d)
Zk <- data.frame(X = Uk %*% Dk, Patient_ID = row.names(GeneExpression))
Zk <- merge(Zk, RejectionStatus, by = 'Patient_ID') %>%
  mutate(Reject_Status = as.factor(Reject_Status))
ggplot(data = Zk, aes(x = X.1, y = X.2, col = Reject_Status)) +
  geom_point()
```




```
rm(Zk, Dk, Uk, X_GSE21374, Gen_spc)
```

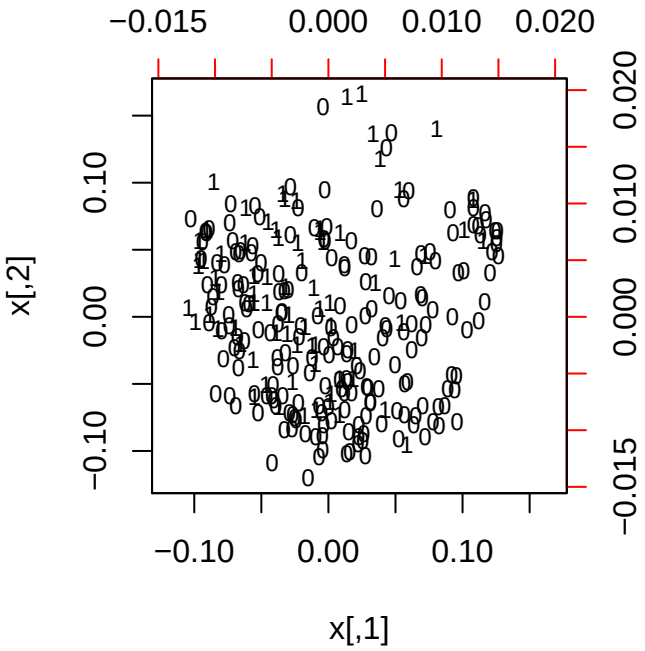
Unfortunately, naive sparce principle component analysis does not seem to work well.

## 2.2 Partial least squares:

```
GeneExpression_comb <-
  list(genes = as.matrix(GeneExpression), Rejection = as.matrix(RejectionStatus$Reject_Status))
pls_model <- pls::plsr(genes ~ Rejection, data = GeneExpression_comb, validation = "CV")
```

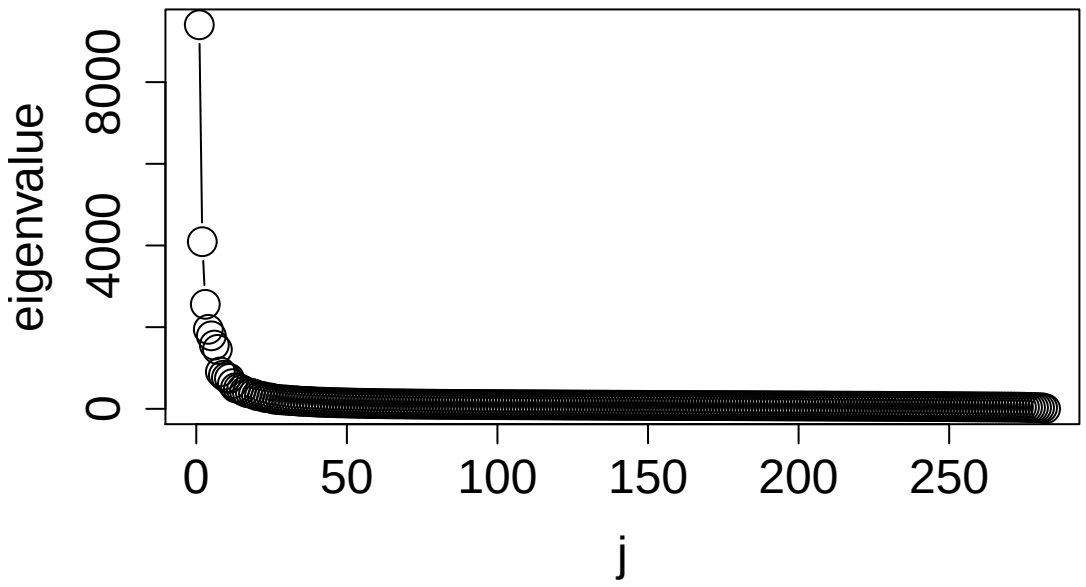
### 2.3 Multi-dimensional scaling:

```
GeneExpression.svd <- svd(GeneExpression)

k <-2
Uk <- GeneExpression.svd$u[,1:k]; Dk <- diag(GeneExpression.svd$d[1:k])
Vk <-GeneExpression.svd$v[,1:k]
Xk <- Uk %*% Dk %*% t(Vk)
Zk <- Uk %*% Dk

rownames(Zk) <- RejectionStatus[[2]]
rownames(Vk) <- colnames(GeneExpression)
ColnamesNull <- colnames(GeneExpression)
ColnamesNull[]<- ""

biplot(Uk, Vk, xlabs=RejectionStatus[[2]], var.axes=FALSE, ylabs=ColnamesNull, cex=0.75)
```

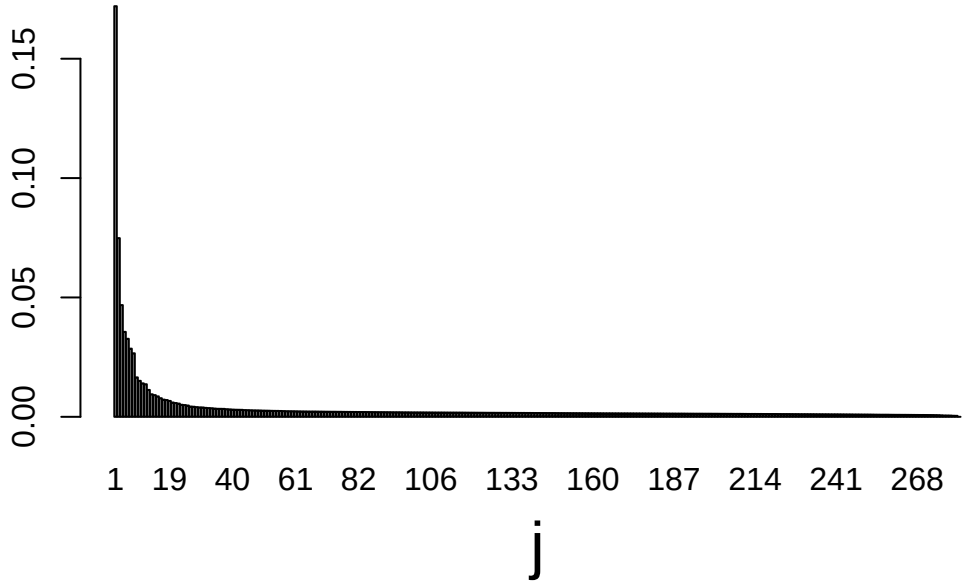



In the biplot of the two first dimensions of the svd, no distinction can be made between rejected and accepted kidneys.

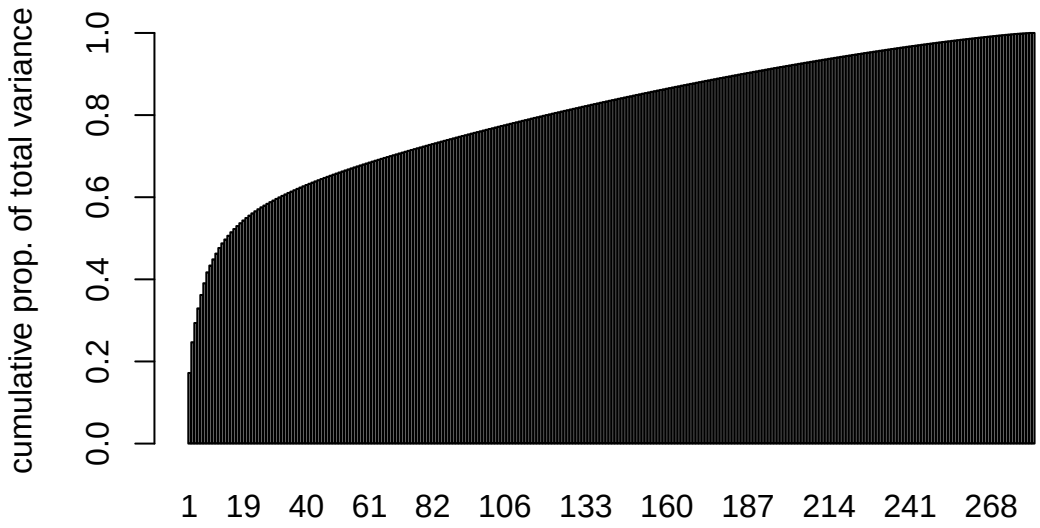
```
totvar <- sum(GeneExpression.svd$d^2)/(nrow(GeneExpression)-1)
plot(GeneExpression.svd$d^2/(nrow(GeneExpression)-1), type="b",
     ylab="eigenvalue",xlab="j",cex=2, cex.axis=1.5, cex.lab=1.5)
```




```
barplot(GeneExpression.svd$d^2/(nrow(GeneExpression)-1)/totvar,
        names.arg=1:nrow(GeneExpression), xlab='j', cex.lab=2)
```



```
barplot(cumsum(GeneExpression.svd$d^2/(nrow(GeneExpression)-1)/totvar), names.arg=1:nrow(GeneExpression)
```




In the scree plots above it can be seen that the two first dimensions account for only 26% of the total variance in the dataset.

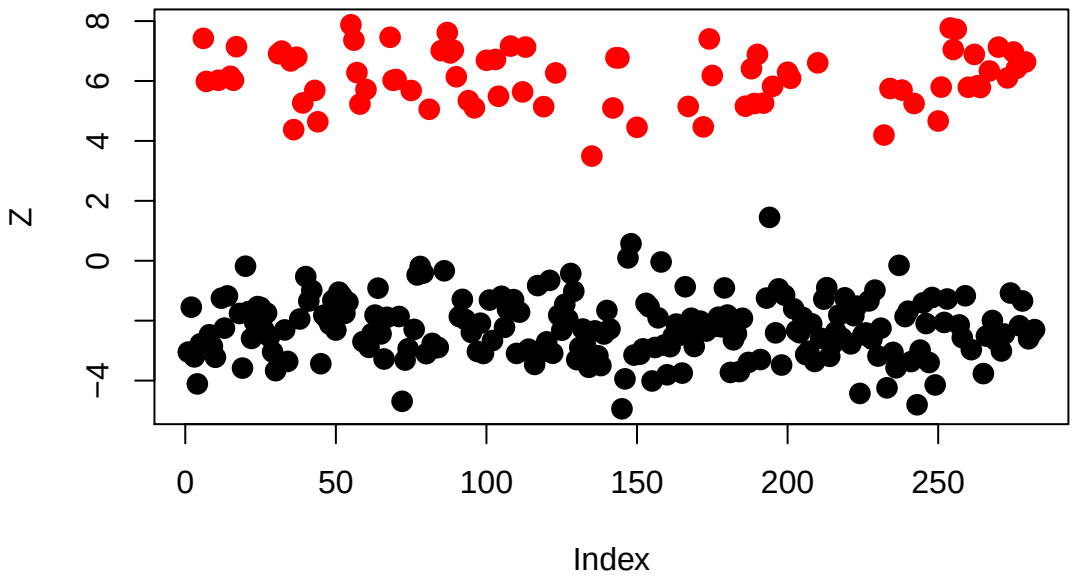
## 2.4 LDA

In an exploratory fashion, we look at the results of an LDA, performed with the first 300 gene expressions. We chose to do this because LDA performed using all gene expressions did not work.

```
GeneExpression.lda <- lda(GeneExpression[,1:300], grouping = RejectionStatus$Reject_Status)
```

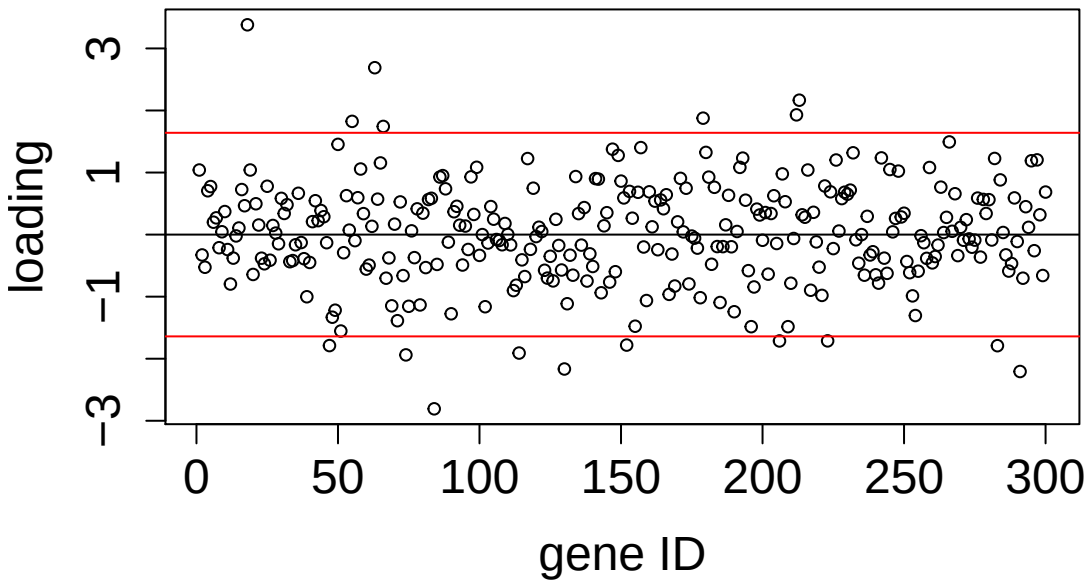
```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
V <- GeneExpression.lda$scaling
Z <- GeneExpression[,1:300] %*% V

plot(Z, col =  RejectionStatus$Reject_Status+1, pch= 16, cex = 1.5)
```




We can see a clear distinction between the 2 rejection status groups.

```
plot(V, cex = 0.8, cex.axis = 1.5, cex.lab = 1.5, xlab = "gene ID",
     ylab = "loading")

abline(h=0)
abline(h=c(2,-2)*sd(V), col = 2)
```




```
id <- which(abs(V) > 2*sd(V))
```

The most contributing gene expressions are:

```
colnames(GeneExpression[,id])
```

```
##  [1] "1552263_at"   "1552307_a_at" "1552318_at"   "1552327_at"   "1552332_at"
##  [6] "1552347_at"   "1552367_a_at" "1552411_at"   "1552436_a_at" "1552473_at"
## [11] "1552508_at"   "1552548_at"   "1552557_a_at" "1552558_a_at" "1552573_s_at"
## [16] "1552658_a_at" "1552667_a_at"
```

We repeated this for 243 folds of 225 gene expressions because

```
243 * 225 == dim(GeneExpression)[2]
```

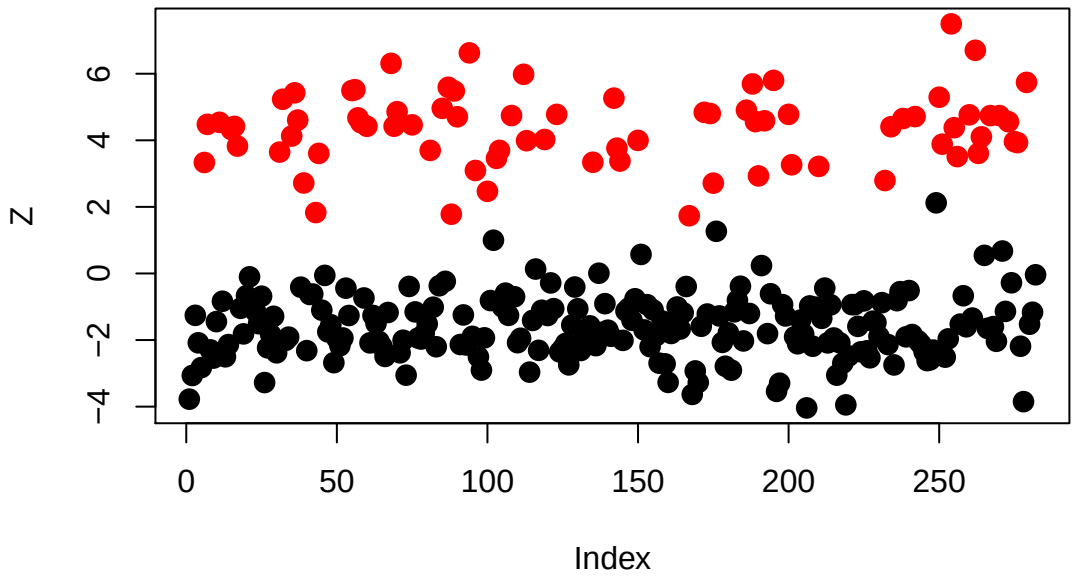
```
## [1] TRUE
```

```
id.all <- vector("numeric")

for (i in 1:(dim(GeneExpression)[2]/225)){

  start <- 1 + (i-1)*225
  stop <- start + 224
  gene.lda <- lda(GeneExpression[,start:stop], grouping = RejectionStatus$Reject_Status)

  V <- gene.lda$scaling
  Z <- GeneExpression[,start:stop] %*% V


  id.all <- append(id.all, which(abs(V) > 3*sd(V)) - 1 + start) # -1 to start at 0

}
```

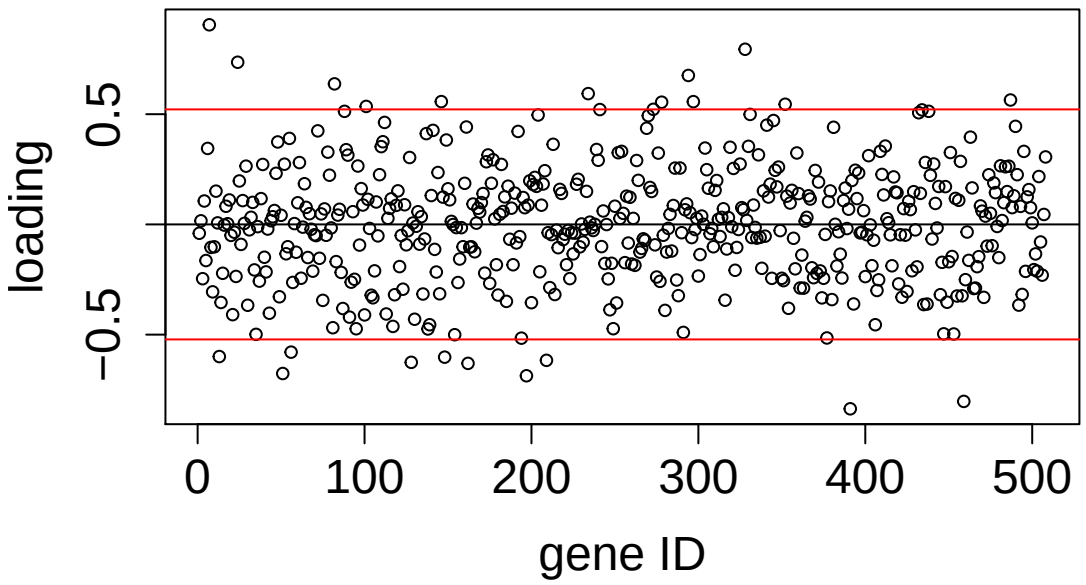
```
gene.lda <- lda(GeneExpression[,id.all], grouping = RejectionStatus$Reject_Status)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
V <- gene.lda$scaling
Z <- GeneExpression[,id.all] %*% V
plot(Z, col =  RejectionStatus$Reject_Status+1, pch= 16, cex = 1.5)
```




```
plot(V, cex = 0.8, cex.axis = 1.5, cex.lab = 1.5, xlab = "gene ID",
     ylab = "loading")

abline(h=0)
abline(h=c(2,-2)*sd(V), col = 2)
```




```
id <- which(abs(V) > 2*sd(V))

colnames(GeneExpression[,id.all][,id])
```

```
##  [1] "1553102_a_at" "1553584_at"   "1555495_a_at" "1565703_at"   "1568513_x_at"
##  [6] "201745_at"    "203253_s_at"  "204959_at"    "206313_at"    "206445_s_at"
## [11] "207614_s_at"  "210970_s_at"  "212036_s_at"  "213853_at"    "218757_s_at"
## [16] "219121_s_at"  "220485_s_at"  "220554_at"    "223922_x_at"  "225956_at"
## [21] "229603_at"    "238461_at"    "242669_at"
```

Capping off at +- 3SD, this resulted in 508 gene expressions.

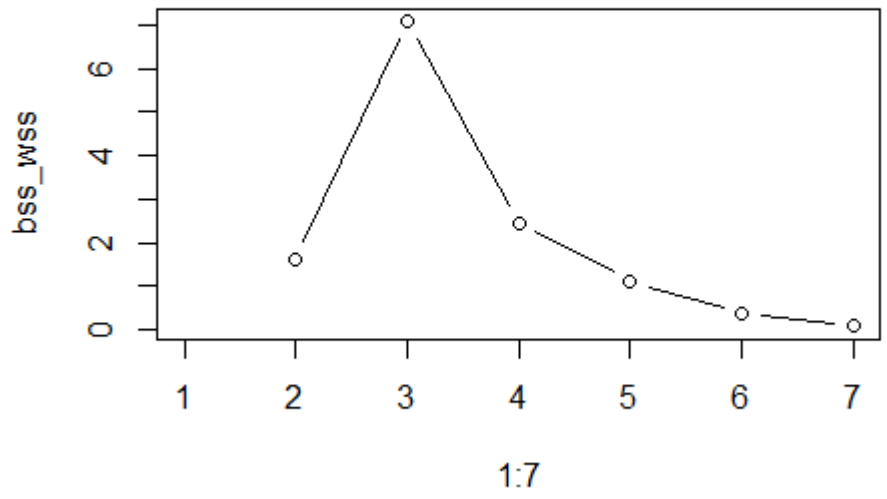
Using all these, and only these gene expressions, an LDA delivers 23 genes.

Why at 3SD? PAS OP UITVOEREN DUURT LANG, in plaats daarvan, zie plot in de map

```
id.all <- vector("numeric")
bss_wss <- vector("numeric")
p <- vector("numeric")

for(j in 2:8){
  id.all <- vector("numeric")
for (i in 1:(dim(gene)[2]/225)){

  start <- 1 + (i-1)*225
  stop <- start + 224
  gene.lda <- lda(gene[,start:stop], grouping = RejectionStatus$Reject_Status)

  V <- gene.lda$scaling
  Z <- gene[,start:stop] %*% V
```

```
  id.all <- append(id.all, which(abs(V) > j*sd(V)) - 1 + start) # -1 to start at 0

}

gene.lda <- lda(gene[,id.all], grouping = RejectionStatus$Reject_Status)

V <- gene.lda$scaling
Z <- gene[,id.all] %*% V

a <- sum((Z[RejectionStatus$Reject_Status==1] - mean(Z[RejectionStatus$Reject_Status==1]))**2)
b <- sum((Z[RejectionStatus$Reject_Status==0] - mean(Z[RejectionStatus$Reject_Status==0]))**2)

within <- a+b


a <- sum((mean(Z[RejectionStatus$Reject_Status==0])-mean(Z))**2)
b <- sum((mean(Z[RejectionStatus$Reject_Status==1])-mean(Z))**2)
between <- a*sum(RejectionStatus$Reject_Status==0)+b*sum(RejectionStatus$Reject_Status==1)

bss_wss[j] <- between/within
p[j] <- length(id.all)
}

plot(1:7, bss_wss, "b")
```

```
knitr::include_graphics("hoeveel_sd.png")
```



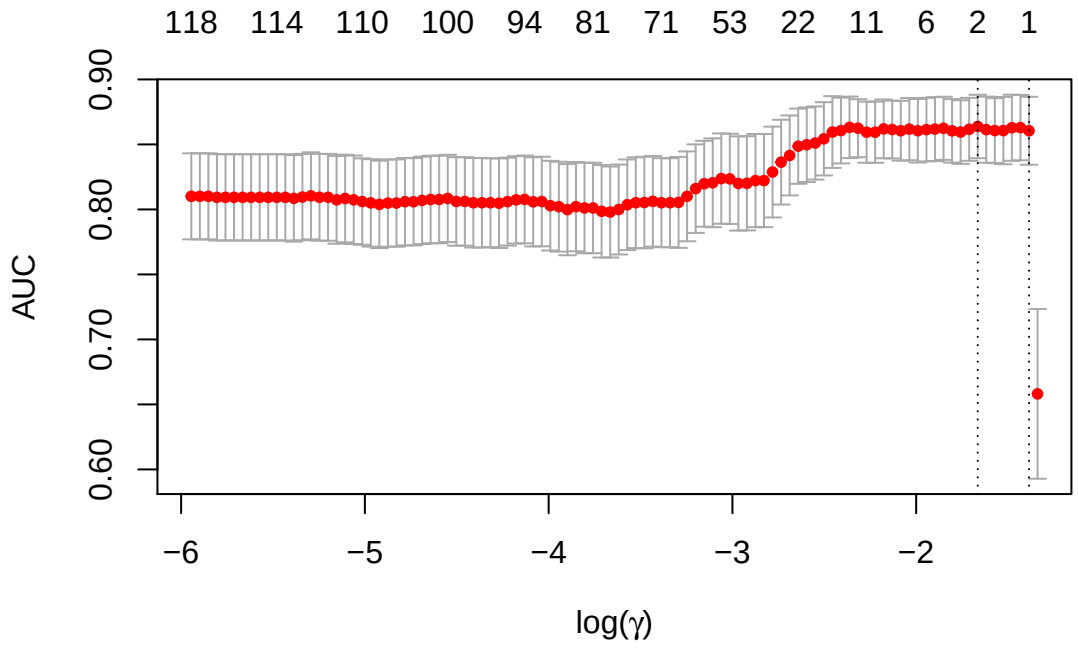
# 3 Differentiating genes between kidney acceptance and rejection

# 4 Prediction of kidney transplant rejection

```
ind_train <-
  sample(seq_len(nrow(RejectionStatus)), size = floor(nrow(RejectionStatus) * 0.80))

Y_train <- as.matrix(RejectionStatus[ind_train, 'Reject_Status'])
X_train <- as.matrix(GeneExpression[ind_train,])
Y_test <- as.matrix(RejectionStatus[-ind_train, 'Reject_Status'])
X_test <- as.matrix(GeneExpression[-ind_train,])
```

## 4.1 Lasso regression

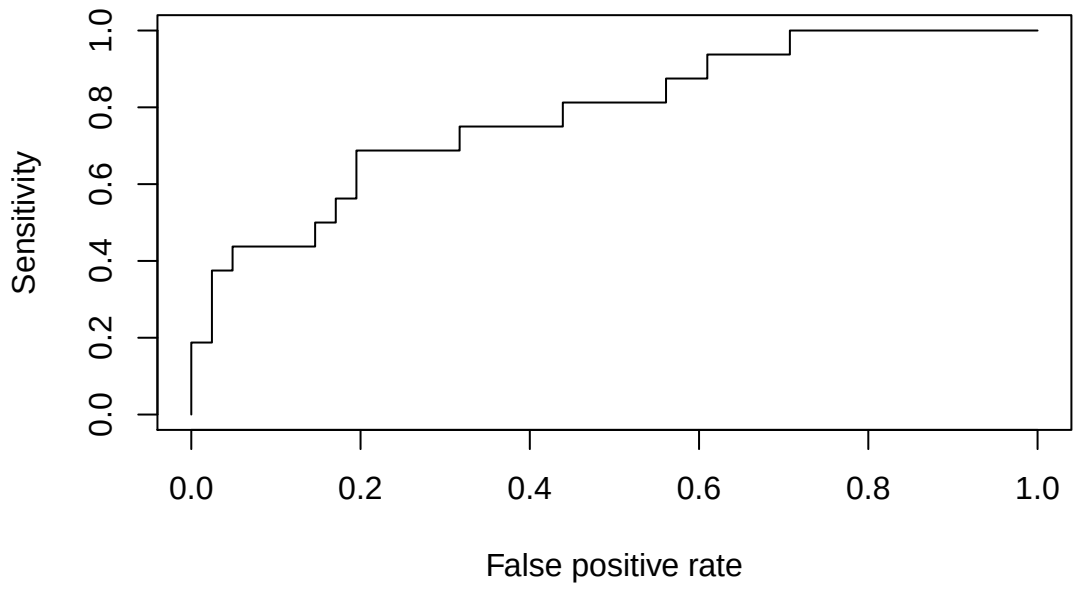
```
m.cv <-
  cv.glmnet(
    x = X_train,
    y = Y_train,
    alpha = 1,
    family = 'binomial',
    type.measure = "auc"
  )
plot(m.cv, xlab = TeX(" $ log(\\gamma ) $ "))
```



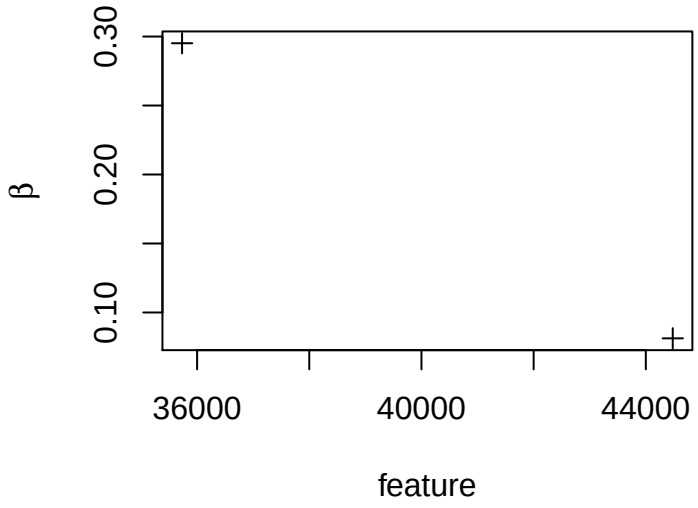

In the figure above, one can see that for $\gamma$ equal to 0.189167 , the area under the curve ( $AUC$ ) is maximal for the train dataset based on a 10-fold cross-validation over the train dataset.

The ROC curve, estimated with the cross-validation dataset, is shown below:

```r
m <- glmnet(
  x = X_train,
  y = Y_train,
  alpha = 1,
  family = 'binomial',
  lambda = m.cv$lambda.min
)
pred_m <-
  prediction(predict(
    m,
    newx = X_test,
    type = 'response'
  ),
  Y_test)
perf <- performance(pred_m, 'sens', 'fpr')
plot(perf)
```

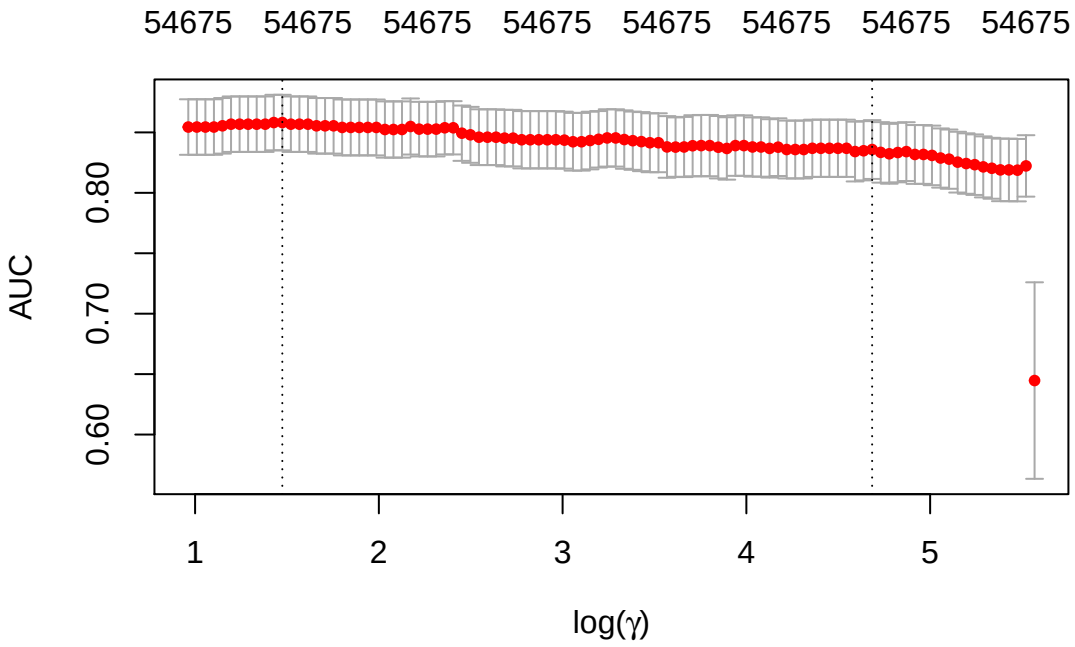



This model uses 2 of the features. This is a considerable dimensional reduction. This is illustrated below. This figure shows the loadings of the 2 selected values.




### 4.2 Ridge regression

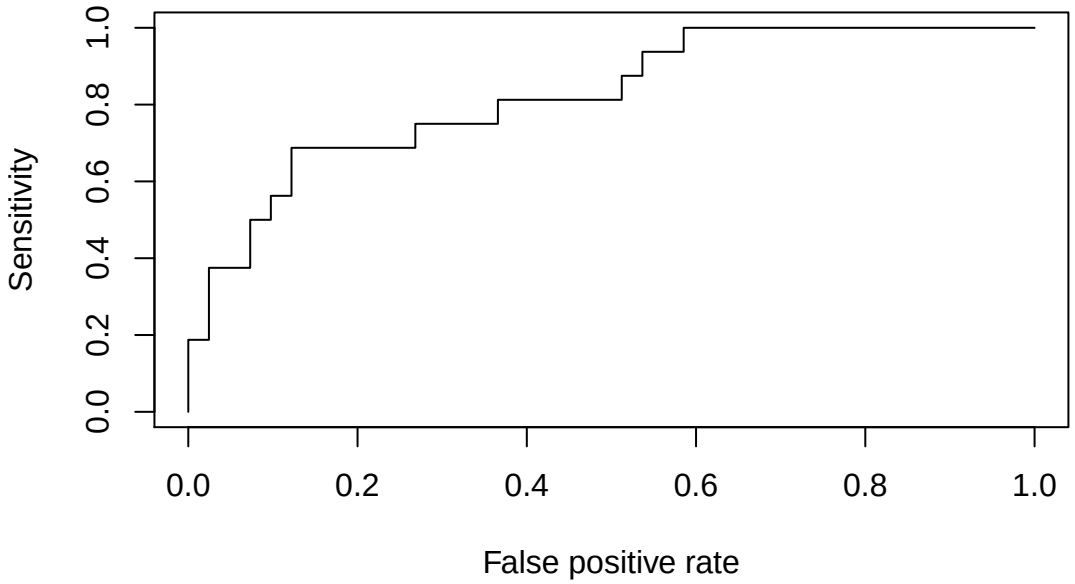
```
m.cv <-
  cv.glmnet(
    x = X_train,
    y = Y_train,
    alpha = 0,
    family = 'binomial',
    type.measure = "auc"
  )
plot(m.cv, xlab = TeX(" $ log(\\gamma ) $ "))
```




In the figure above, one can see that for $\gamma$ equal to 4.3700033 , the area under the curve ( $AUC$ ) is maximal for the train dataset based on a 10-fold cross-validation over the train dataset.

The ROC curve, estimated with the cross-validation dataset, is shown below:

```
m <- glmnet(
  x = X_train,
  y = Y_train,
  alpha = 0,
  family = 'binomial',
  lambda = m.cv$lambda.min
)
pred_m <-
  prediction(predict(
    m,
    newx = X_test,
    type = 'response'
  ),
  Y_test)
perf <- performance(pred_m, 'sens', 'fpr')
plot(perf)
```


Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate