Project
Transplant kidney rejection
High Dimensional Data Analysis

Jan Alexander* Annabel Vaessens† Steven Wallaert‡

8/4/2020

1 Introduction

The data is loaded as the raw dataset, the centered dataset and the standardized dataset.

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

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

```
dim(X_GSE21374)
```

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

```
GeneExpression <- t(X_GSE21374)
GeneExpression_C <- scale(t(X_GSE21374),scale = F) # centered
GeneExpression_S <- scale(t(X_GSE21374),scale = T) # scaled

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
```

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

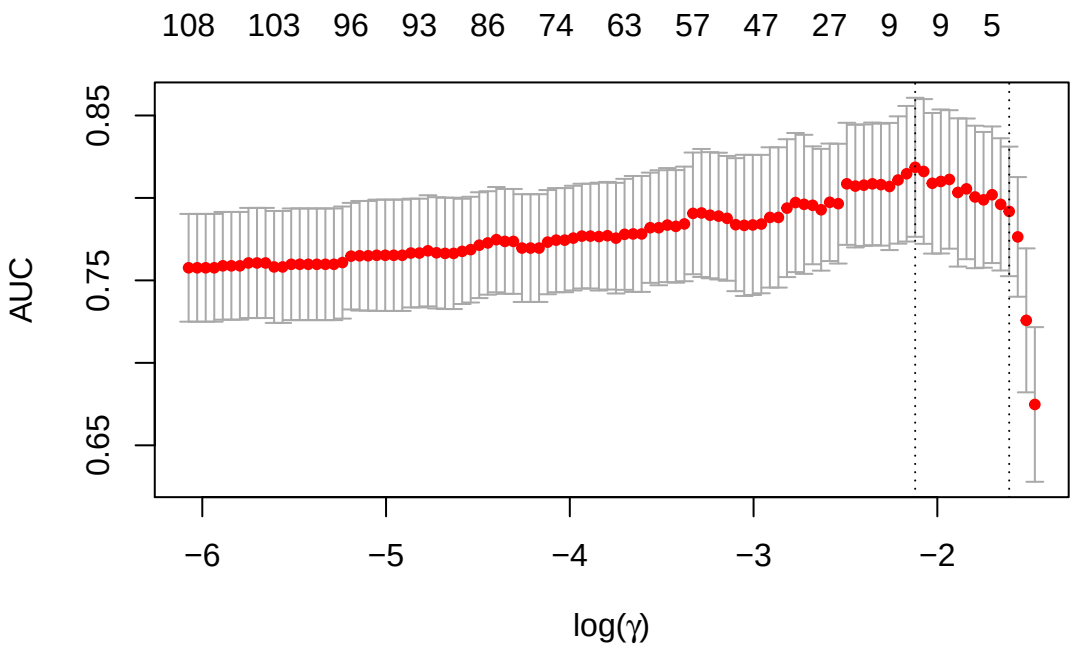
2 Prediction of kidney transplant rejection

The dataset is split into a training and test dataset.

```
ind_train <-
  sample(seq_len(nrow(RejectionStatus)), size = floor(nrow(RejectionStatus) * 0.70))

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

2.1 Lasso regression

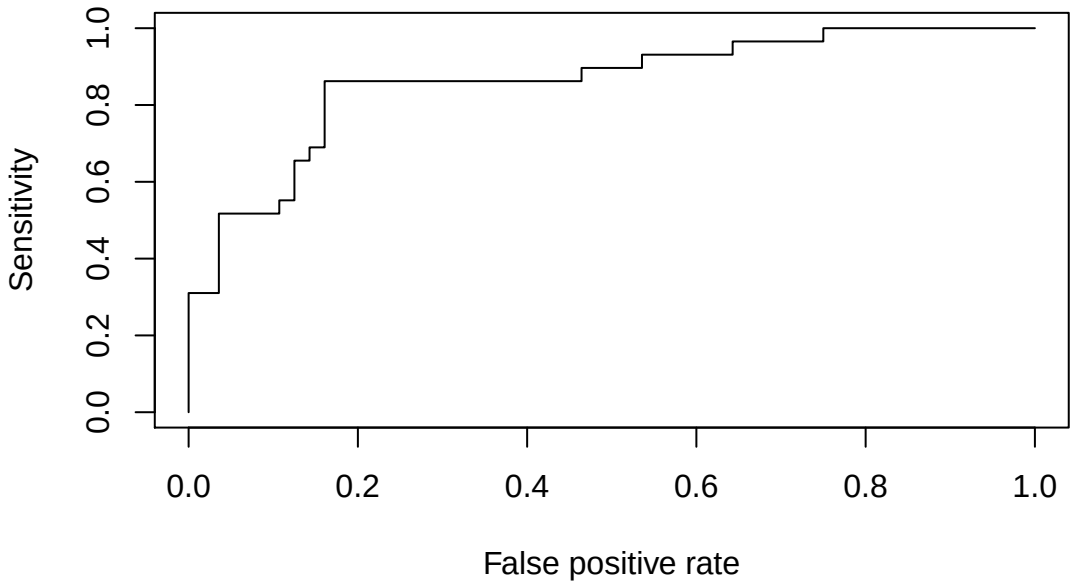
```
m.cv <-
  cv.glmnet(
    x = X_train,
    y = Y_train,
    alpha = 1,
    family = 'binomial',
    type.measure = "auc"
  )
m <- glmnet(
  x = X_train,
  y = Y_train,
  alpha = 1,
  family = 'binomial',
  lambda = m.cv$lambda.min
)
pred_m <-
  prediction(predict(
    m,
    newx = X_test,
    type = 'response'
  ),
  Y_test)
plot(m.cv, xlab = TeX(" $ log(\\gamma ) $ "))
```



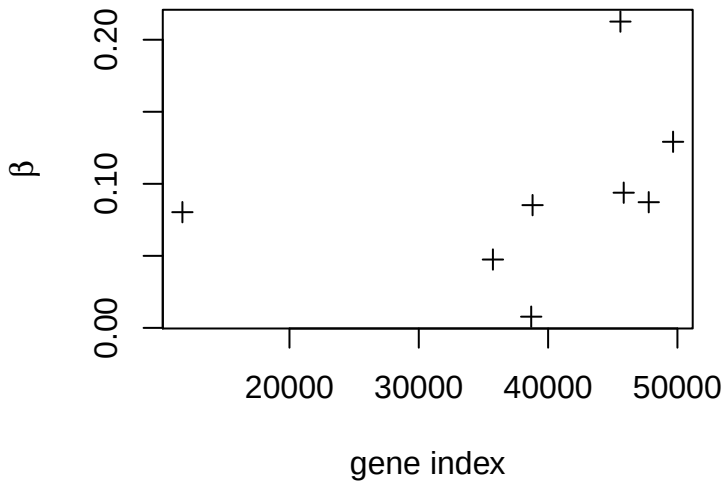
In the figure above, one can see that for γ equal to 0.1199672 , the area under the curve (AUC) is maximal (0.860837438423645) for the train dataset based on a 10-fold cross-validation over the train dataset.

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

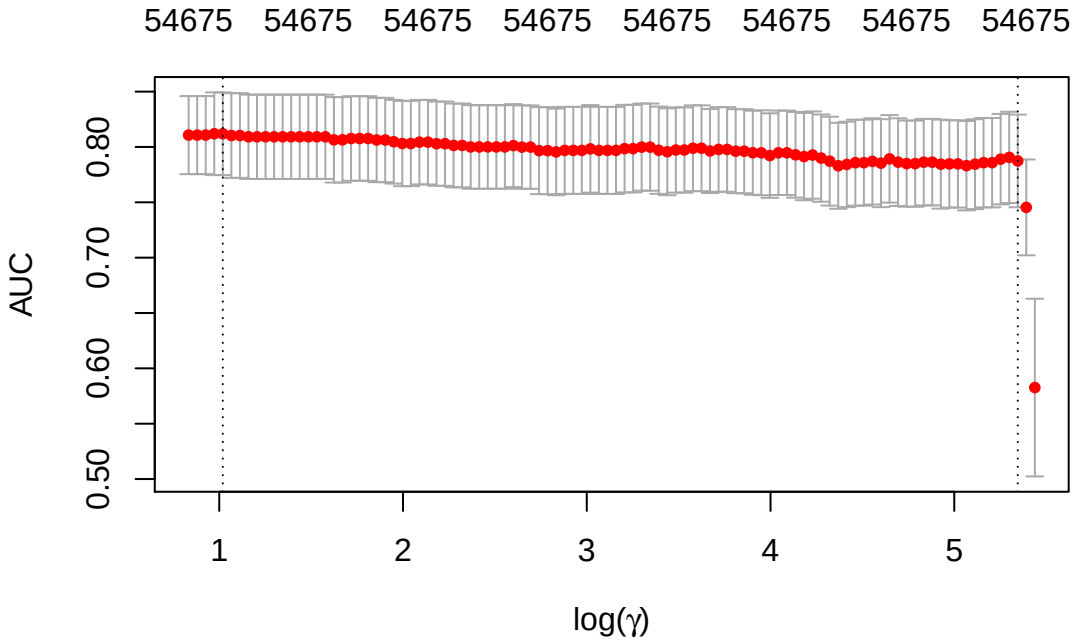
```
perf <- performance(pred_m, 'sens', 'fpr')
plot(perf)
```



This model only uses 8 of the genes. This is a considerable dimensional reduction. This is illustrated below. This figure shows the loadings of the 8 selected values.



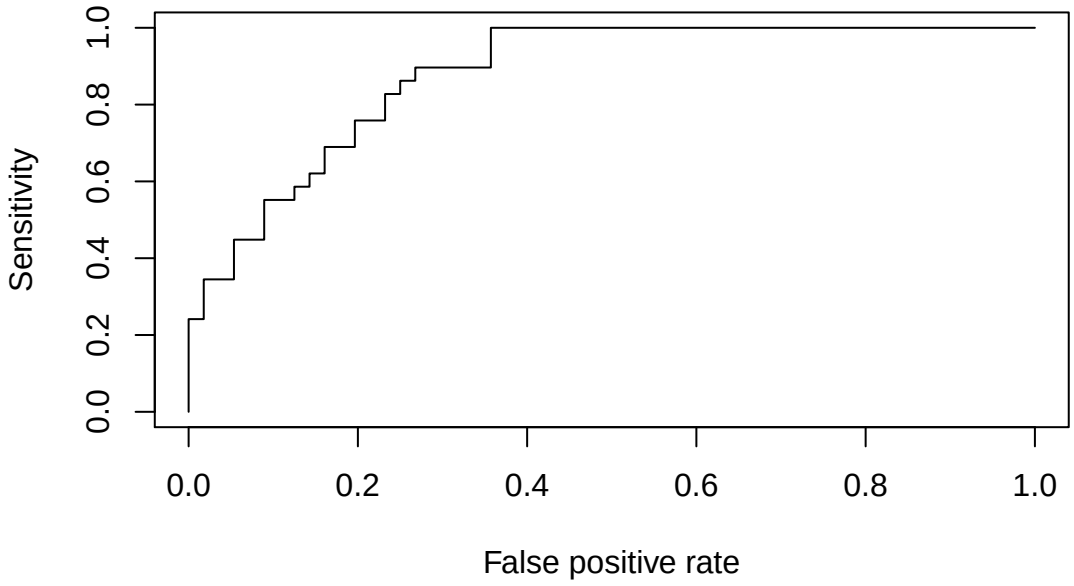
2.2 Ridge regression



Likewise as for the Lasso regression, for γ equal to 2.771399, the optimal AUC (0.878694581280788) is obtained.

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

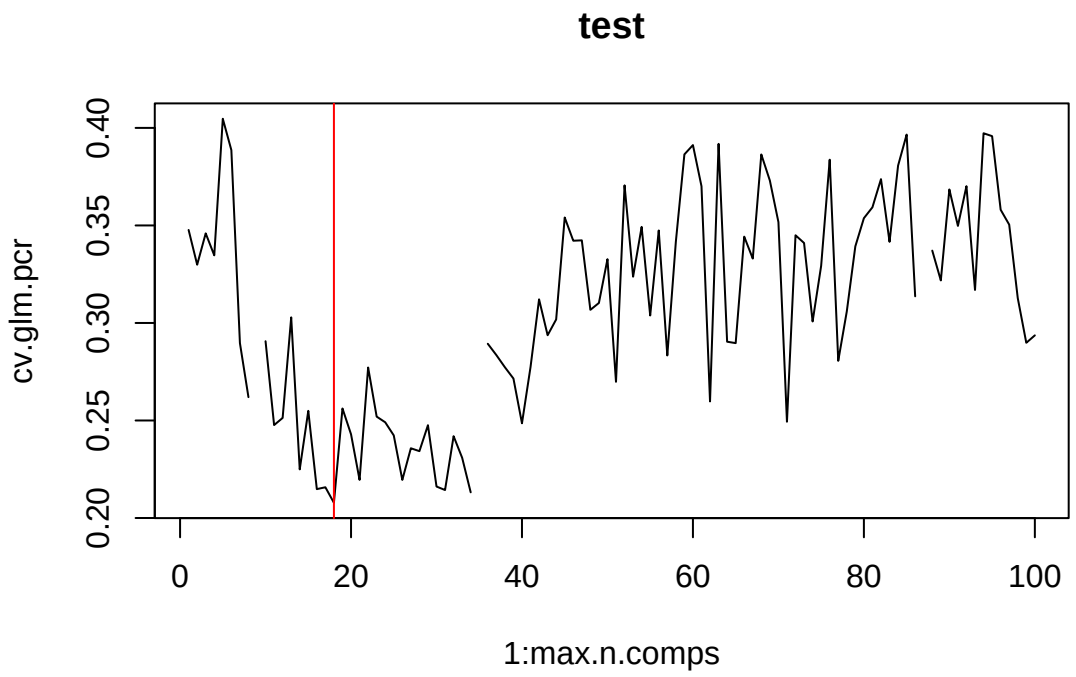
```
perf <- performance(pred_m, 'sens', 'fpr')
plot(perf)
```



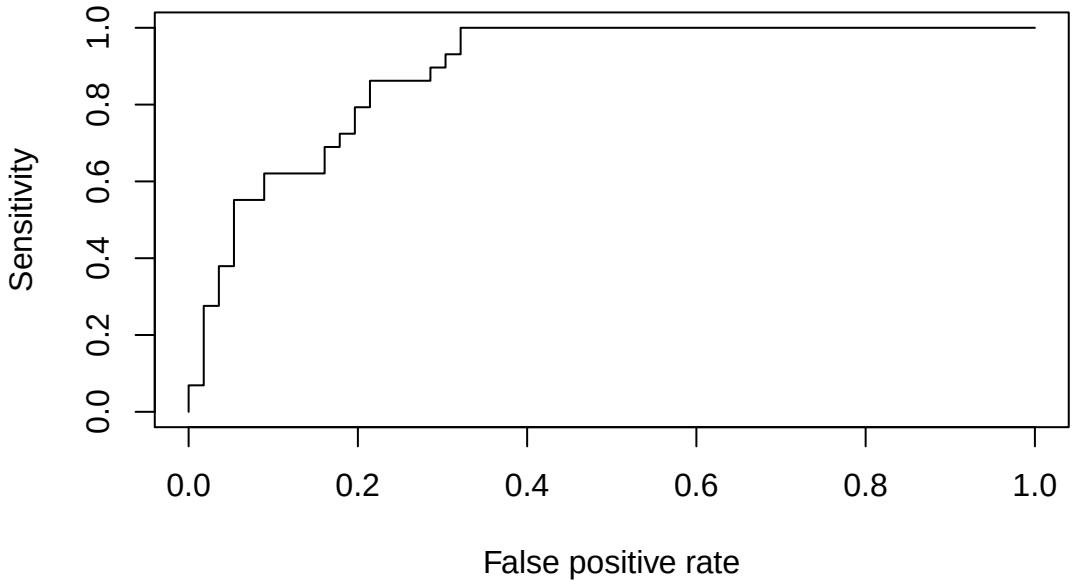
2.3 Principal component regression

```
trapezoid_integration <- function(x, y) {
  elements <- (y[-1] + y[-length(y)]) * abs(diff(x))
  sum(elements/2)
}
# cost function for CV
AUC_est <- function(obs, pred){
  obs <- factor(obs, levels = c(0,1))
  # AUC is estimated by calculating the sensitivity and the FPR for different values of cutoff C
  intervals <- 500
  sensitivity <- rep(NA, intervals+1)
  specificity <- rep(NA, intervals+1)
  for(i in seq(0, intervals)){
    cutoff <- i/intervals
    ypred <- factor(as.numeric(pred > cutoff), levels = c(0, 1))
    tab <- table(obs, ypred)
    TN <- tab[1]
    FN <- tab[2]
    FP <- tab[3]
    TP <- tab[4]
    sensitivity[i+1] <- TP / (TP + FN)
    specificity[i+1] <- TN / (FP + TN)
  }
  #plot(1-specificity, sensitivity)
  AUC <- trapezoid_integration(1-specificity, sensitivity)
  #cat(paste0('AUC estimate : ', AUC, '\n'))
  return(1-AUC)
}

max.n.comps <- 100 #random nr

cv.glm.pcr <- rep(NA, max.n.comps)
```

```r
X_train.svd <- svd(X_train)

U <- X_train.svd$u
D <- diag(X_train.svd$d)
Z_train <- U %*% D

for (i in 1:max.n.comps) {
  fitdata <- data.frame(Y_train, Z_train[, 1:i])

  mod <- glm(Y_train ~ ., data = fitdata, family = "binomial")

  cv.glm.pcr[i] <-
    cv.glm(fitdata, mod, cost = AUC_est, K = 10)$delta[1]
}

plot(1:max.n.comps, cv.glm.pcr, type = "l", main = 'test')
npc.min <- which.min(cv.glm.pcr)
npc.val <- cv.glm.pcr[npc.min]
abline(v = npc.min, col = 2)
```




```r
V <- X_train.svd$v
Z_test <- X_test %*% V
fitdata <- data.frame(Y_train, Z_train[,1:npc.min])
mod <- glm(Y_train ~ ., data = fitdata)
preddata <- data.frame(Z_test[,1:npc.min])
pred_mod <- prediction(predict(mod, newdata = preddata), Y_test)
perf_mod <- performance(pred_mod, "sens", "fpr")
plot(perf_mod)
```



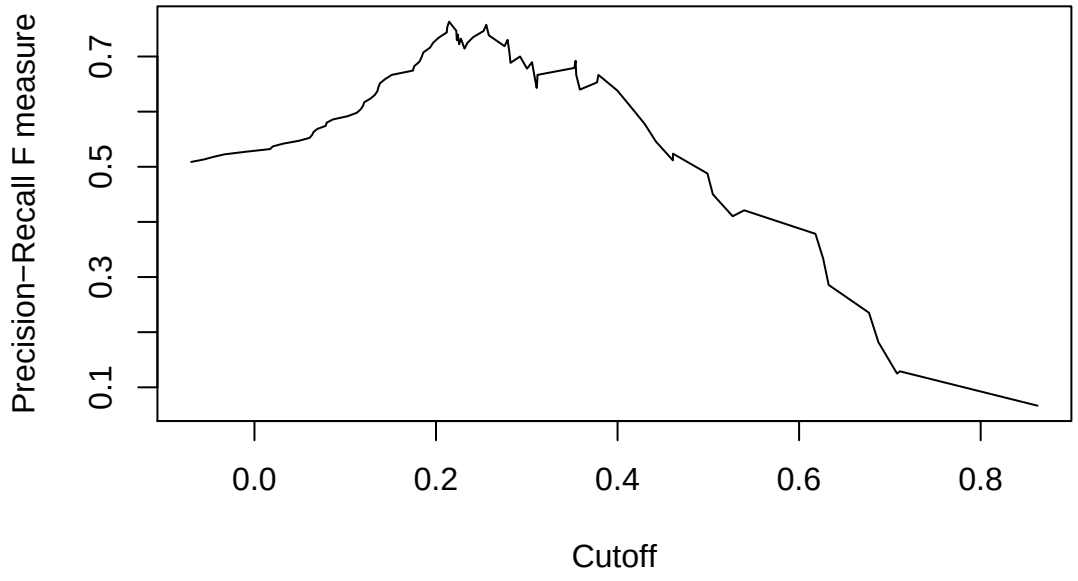
AUC is 0.889162561576355.

2.4 Final model evaluation

The PCR model is performing best, based on the AUC criterium. By chosing cutoff c = 0.21, we can achieve a F1-score of over 0.7. This is represented on the following graph. Below, the F1 graph, the confusion matrix for the PCR model with cutoff c = 0.21 is shown.

The confusion matrix shows no false negatives at all, but the number of false positives is high.

```
perf_f1 <- performance(pred_mod, "f")
plot(perf_f1)
```




```
pred <- predict(mod, newdata = preddata)
cutoff <- .21
ypred <- factor(as.numeric(pred > cutoff), levels = c(0, 1))
```

```
tab <- table(Y_test, ypred)
knitr::kable(tab, digits = 0, align = 'l')
```

	0	1
0	36	20
1	0	29