Penalized Cox Regression

Simulated data

```r
library(survival)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.1
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```

Data generation

```r
set.seed(1234)

N <- 1000
p <- 30  # total num. features
nzc <- p/3 # num. 'true' predictors, one third of them B are not 0

X <- matrix(rnorm(N * p), nrow = N, ncol = p)
beta <- rnorm(nzc)
y <- local({
  linear_predictor <- X[, seq_len(nzc)] %*% beta
  hazard <- exp(linear_predictor)
  y_time_event <- rexp(N, rate = hazard)
  y_time_censoring <- rexp(N, rate = hazard * 0.5)
  y_time <- pmin(y_time_event, y_time_censoring)
  y_event <- y_time_event <= y_time_censoring
  Surv(y_time, y_event)
})
```

We should have about 1/3 of the data points censored:

```r
table(y[,2])
```

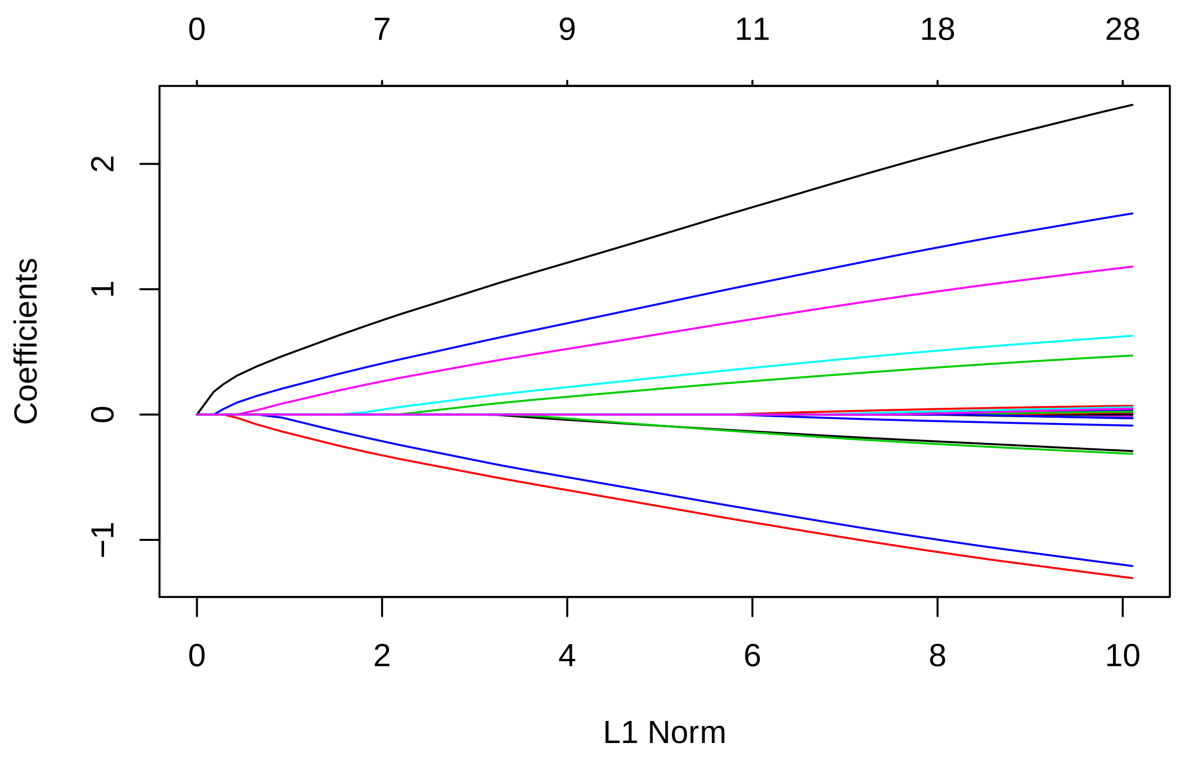
```
##
##   0   1
## 346 654
```

Fit the model

$\alpha \sum |\beta_i| + (1-\alpha) \sum \beta^2 \leq c$

how do we choose 1?

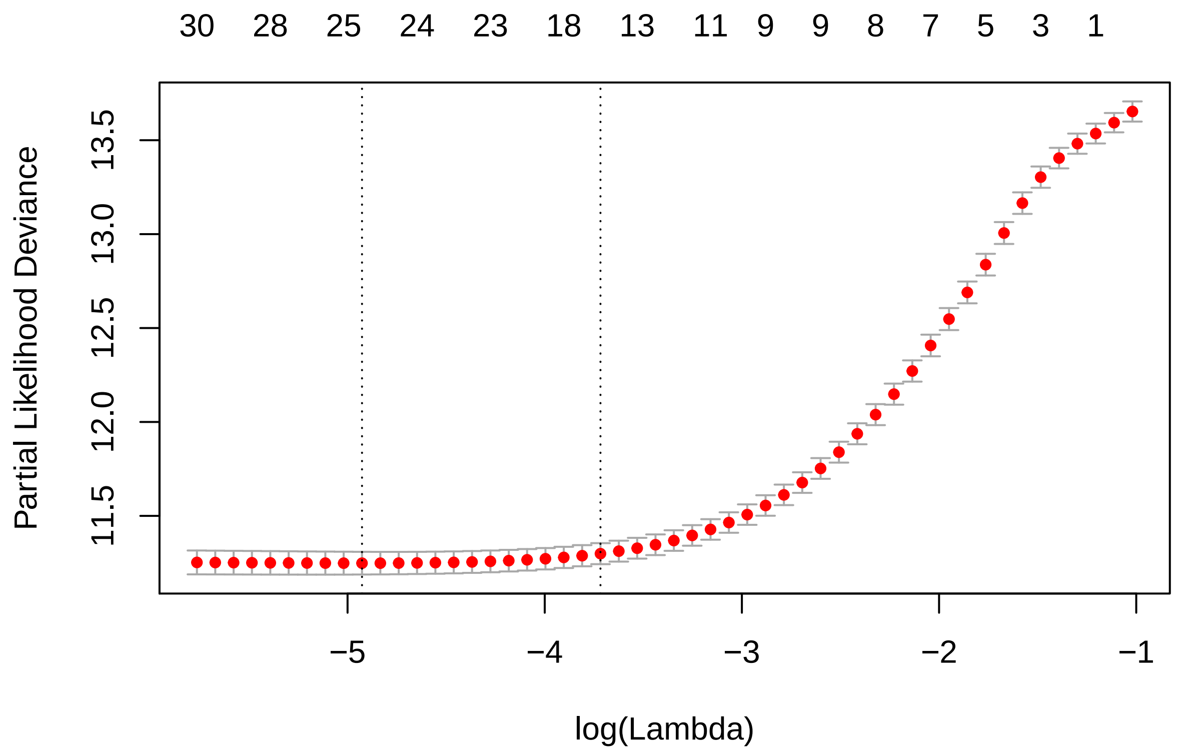
```r
fit <- glmnet(X, y, family = "cox")
plot(fit)
```



Selecting a threshold through cross-validation

the lower the better, 25 coefficient are not 0, but it's better take a more conservative one (15,16)

```r
set.seed(1234)
fit.cv10 <- cv.glmnet(X, y, family = "cox")
plot(fit.cv10)
```



```
#we can also see the numeric value
str(fit.cv10)
```

```
## List of 10
##  $ lambda    : num [1:52] 0.361 0.329 0.3 0.273 0.249 ...
##  $ cvm       : num [1:52] 13.7 13.6 13.5 13.5 13.4 ...
##  $ cvsd      : num [1:52] 0.0537 0.0515 0.0528 0.0538 0.0546 ...
##  $ cvup      : num [1:52] 13.7 13.6 13.6 13.5 13.5 ...
##  $ cvlo      : num [1:52] 13.6 13.5 13.5 13.4 13.4 ...
##  $ nzero     : Named int [1:52] 0 1 1 1 2 3 4 5 5 5 ...
##   ..- attr(*, "names")= chr [1:52] "s0" "s1" "s2" "s3" ...
##  $ name      : Named chr "Partial Likelihood Deviance"
##   ..- attr(*, "names")= chr "deviance"
##  $ glmnet.fit:List of 12
##   ..$ a0       : NULL
##   ..$ beta     :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. .. ..@ i       : int [1:757] 0 0 0 0 3 0 3 7 0 3 ...
##   .. .. ..@ p       : int [1:53] 0 0 1 2 3 5 8 12 17 22 ...
##   .. .. ..@ Dim     : int [1:2] 30 52
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : chr [1:30] "V1" "V2" "V3" "V4" ...
##   .. .. .. ..$ : chr [1:52] "s0" "s1" "s2" "s3" ...
##   .. .. ..@ x       : num [1:757] 0.065 0.1252 0.181 0.243 0.0447 ...
##   .. .. ..@ factors : list()
##   ..$ df       : int [1:52] 0 1 1 1 2 3 4 5 5 5 ...
##   ..$ dim      : int [1:2] 30 52
##   ..$ lambda   : num [1:52] 0.361 0.329 0.3 0.273 0.249 ...
##   ..$ dev.ratio: num [1:52] 0 0.00581 0.0107 0.01484 0.02202 ...
##   ..$ nulldev  : num 7698
##   ..$ npasses  : int 1845
##   ..$ jerr     : int 0
##   ..$ offset   : logi FALSE
##   ..$ call     : language glmnet(x = X, y = y, family = "cox")
##   ..$ nobs     : int 1000
##   ..- attr(*, "class")= chr [1:2] "coxnet" "glmnet"
##  $ lambda.min: num 0.00725
##  $ lambda.1se: num 0.0243
##  - attr(*, "class")= chr "cv.glmnet"
```

Estimated coefficients:

```
# we can extract one of the threshold
coef(fit.cv10, s = "lambda.1se") #lambda.min global minimum, lambda.1se more concervative one
```

```
## 30 x 1 sparse Matrix of class "dgCMatrix"
##                1
## V1   2.013506525
## V2   0.009425558
## V3   0.358727691
## V4   1.286244321
## V5   0.488818732
## V6   0.948241038
## V7  -0.202498677
## V8  -1.059839099
## V9  -0.221302386
## V10 -0.961357414
## V11  .
## V12  .
## V13  .
## V14  0.038965359
## V15  .
## V16  .
## V17  .
## V18  .
## V19  0.012206575
## V20  .
## V21  .
## V22 -0.045944015
## V23  .
## V24  .
## V25  .
## V26  .
## V27  .
## V28  .
## V29  0.019593248
## V30  0.002423260
```

Compare estimated values with ‘true’ values:

```r
beta.true <- c(round(beta, 2), rep(0, ncol(X) - length(beta)))
beta.est <- round(coef(fit.cv10, s = "lambda.1se"), 2)
cbind(beta.true, beta.est)
```

```
## 30 x 2 sparse Matrix of class "dgCMatrix"
##     beta.true     1
## V1       2.49  2.01
## V2       0.05  0.01
## V3       0.46  0.36
## V4       1.58  1.29
## V5       0.62  0.49
## V6       1.19  0.95
## V7      -0.28 -0.20
## V8      -1.35 -1.06
## V9      -0.29 -0.22
## V10     -1.18 -0.96
## V11      .     .
## V12      .     .
## V13      .     .
## V14      .     0.04
## V15      .     .
## V16      .     .
## V17      .     .
## V18      .     .
## V19      .     0.01
## V20      .     .
## V21      .     .
## V22      .    -0.05
## V23      .     .
## V24      .     .
## V25      .     .
## V26      .     .
## V27      .     .
## V28      .     .
## V29      .     0.02
## V30      .     0.00
```

Make predictions

Using built-in functions:

```r
predict(fit.cv10, newx = X[1:5, ], s = "lambda.1se")
```

```
##               1
## [1,] -5.1652836
## [2,]  0.6997955
## [3,]  2.9578906
## [4,] -1.4213800
## [5,]  2.0290144
```

‘Manually’:

```r
b <- coef(fit.cv10, s = "lambda.1se")
b.i <- which(b != 0)
bnz <- b[b.i]
y0 <- X[1:5, b.i, drop = FALSE] %*% bnz
print(y0)
```

```
##            [,1]
## [1,] -5.1652836
## [2,]  0.6997955
## [3,]  2.9578906
## [4,] -1.4213800
## [5,]  2.0290144
```

Case study nr. 1

Load data and packages

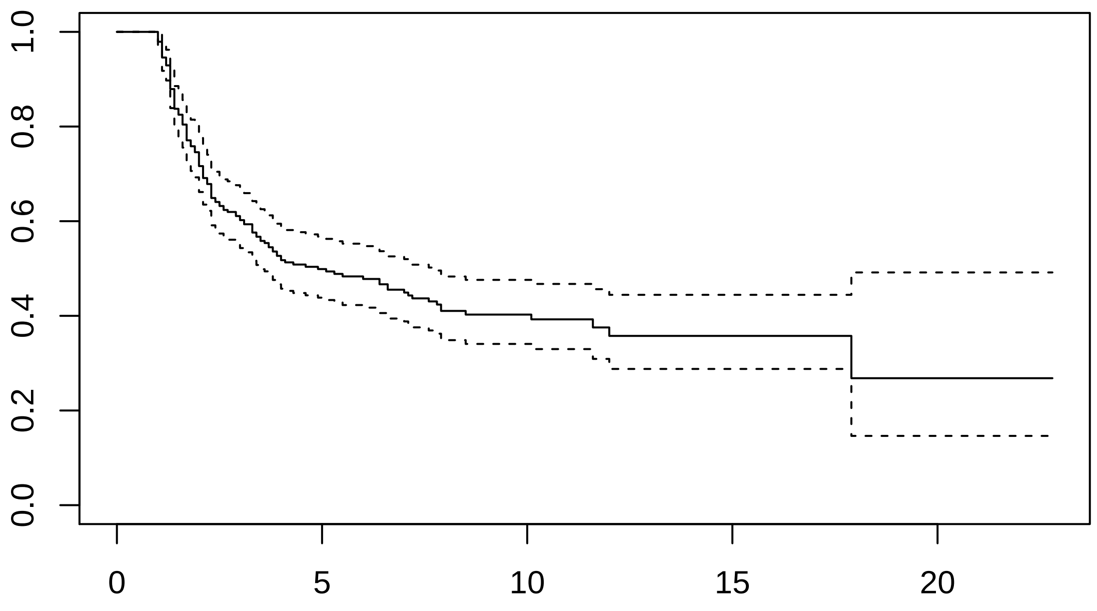
```r
library(survival)
library(glmnet)

load('./LymphomaData.rda')

str(patient.data)
```

```
## List of 3
##  $ x     : num [1:7399, 1:240] -0.221 -0.1786 -0.0503 -0.1922 -0.2944 ...
##  $ time  : num [1:240] 5 5.9 6.6 13.1 1.6 1.3 1.4 2.2 3.4 2.2 ...
##  $ status: num [1:240] 0 0 0 0 1 1 1 1 1 1 ...
```

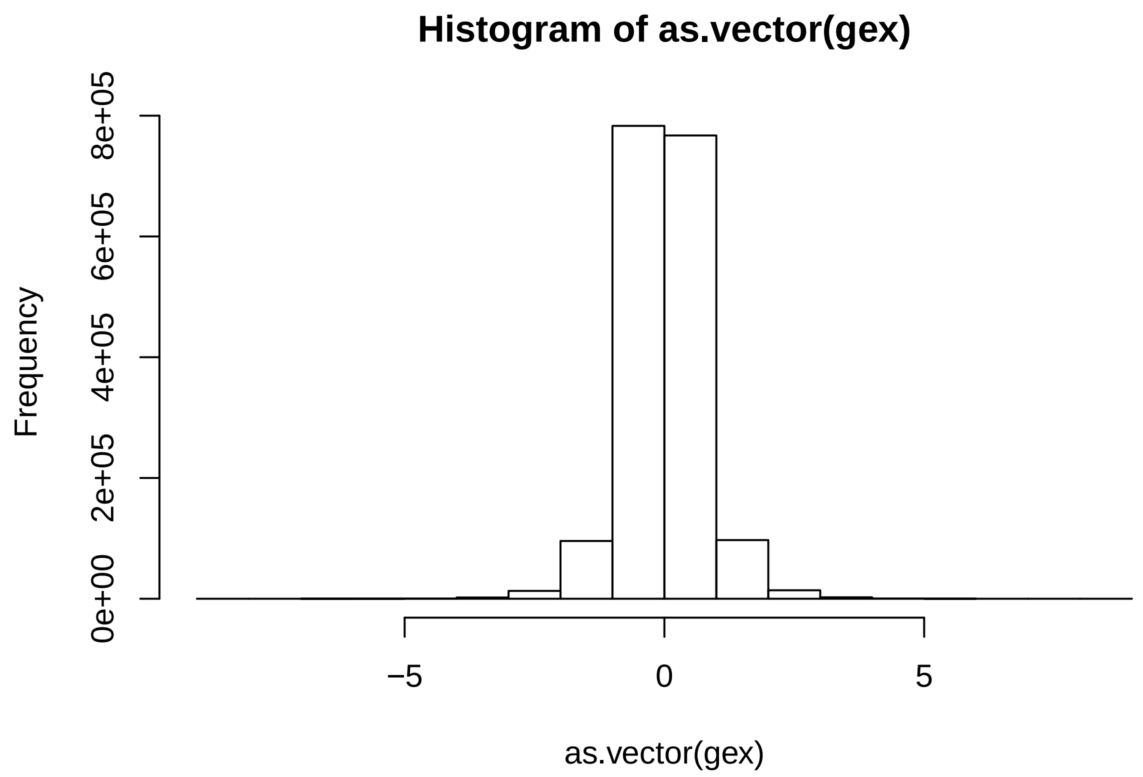
```r
gex <- t(patient.data$x)
y <- Surv(patient.data$time, patient.data$status)
```

Exploratory analysis

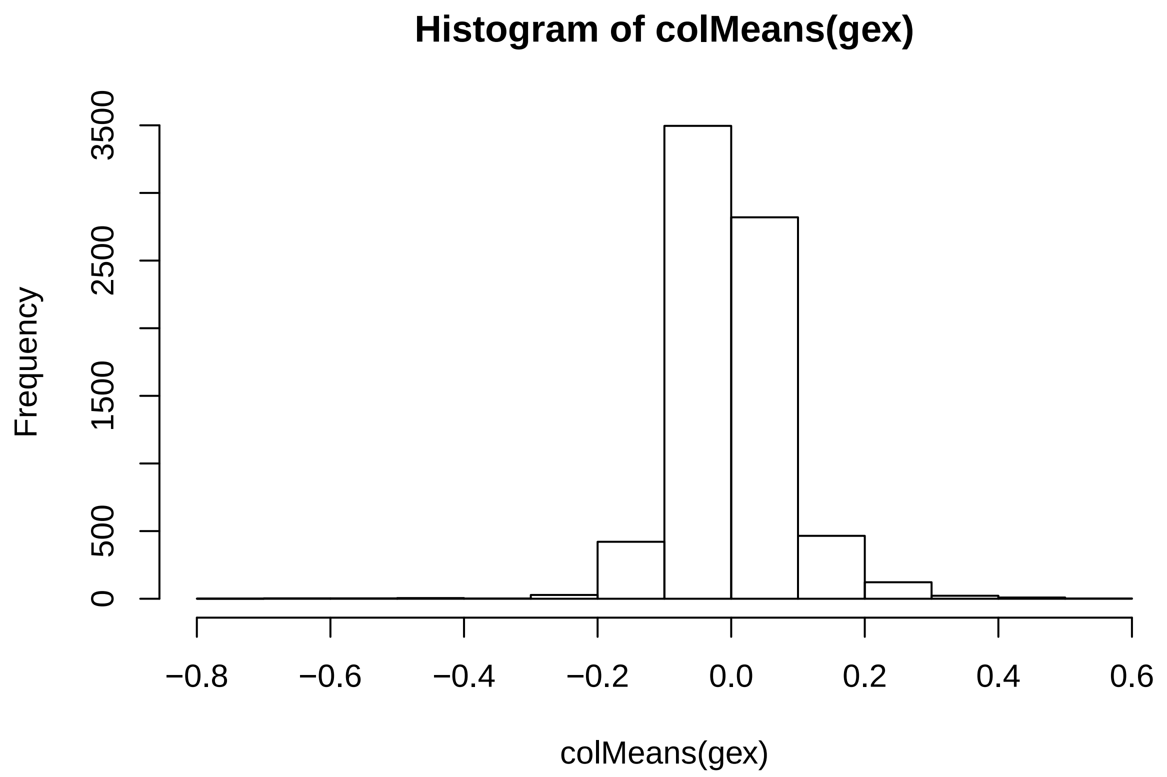
```r
plot(survfit(y ~ 1))
```



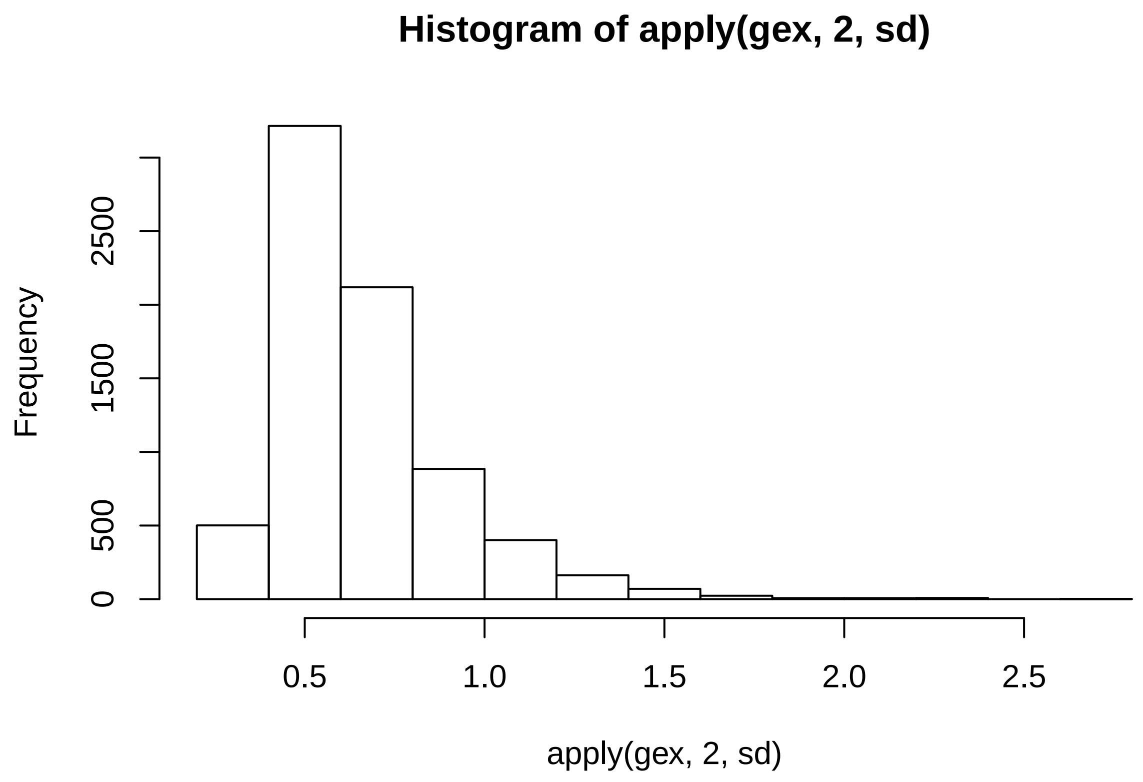
```r
hist(as.vector(gex))
```



```r
hist(colMeans(gex))
```



```
hist(apply(gex, 2, sd))
```



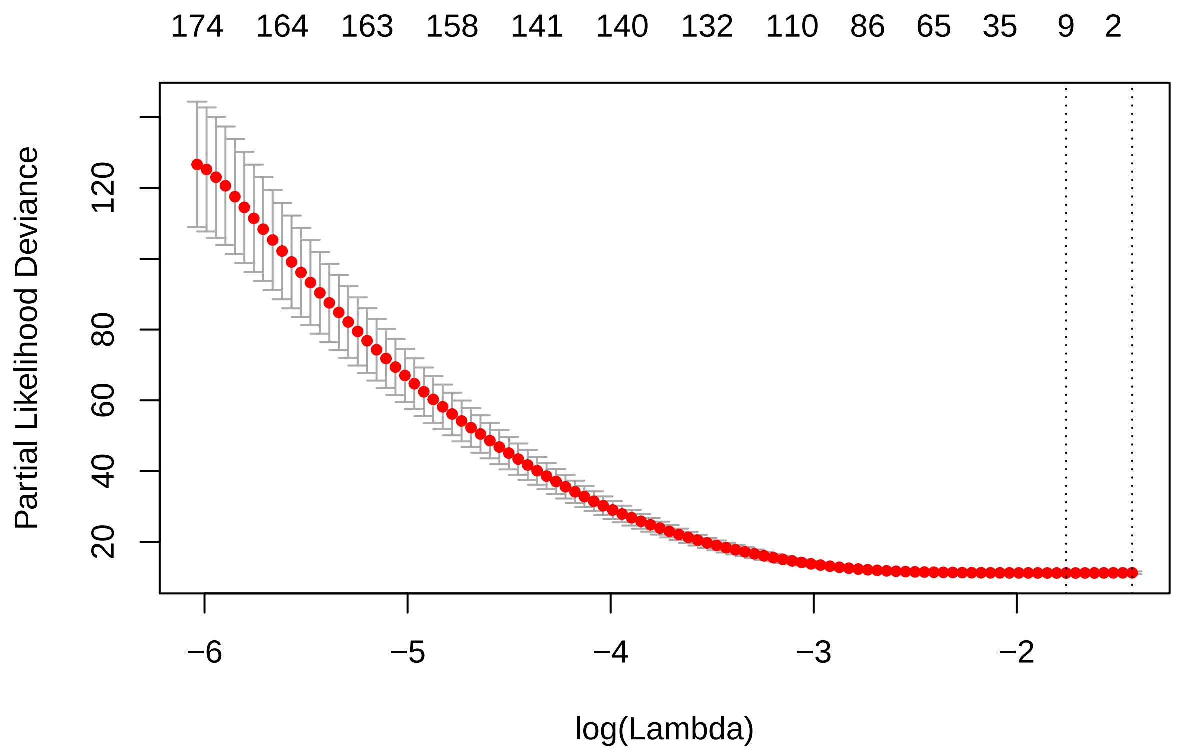
Split the data randomly into a training and a testing set

```r
i.training <- sample.int(nrow(gex), size = 160, replace = FALSE)
i.testing <- setdiff(seq_len(nrow(gex)), i.training)

gex.training <- gex[i.training,, drop = FALSE]
y.training <- y[i.training,, drop = FALSE]

gex.testing <- gex[i.testing,, drop = FALSE]
y.testing <- y[i.testing,, drop = FALSE]
```

Train the model

```r
fit.cv10 <- cv.glmnet(gex.training, y.training, family = "cox")
```

```r
plot(fit.cv10)
```



```r
b <- coef(fit.cv10, s = "lambda.min")
sum(b != 0)
```

```
## [1] 9
```

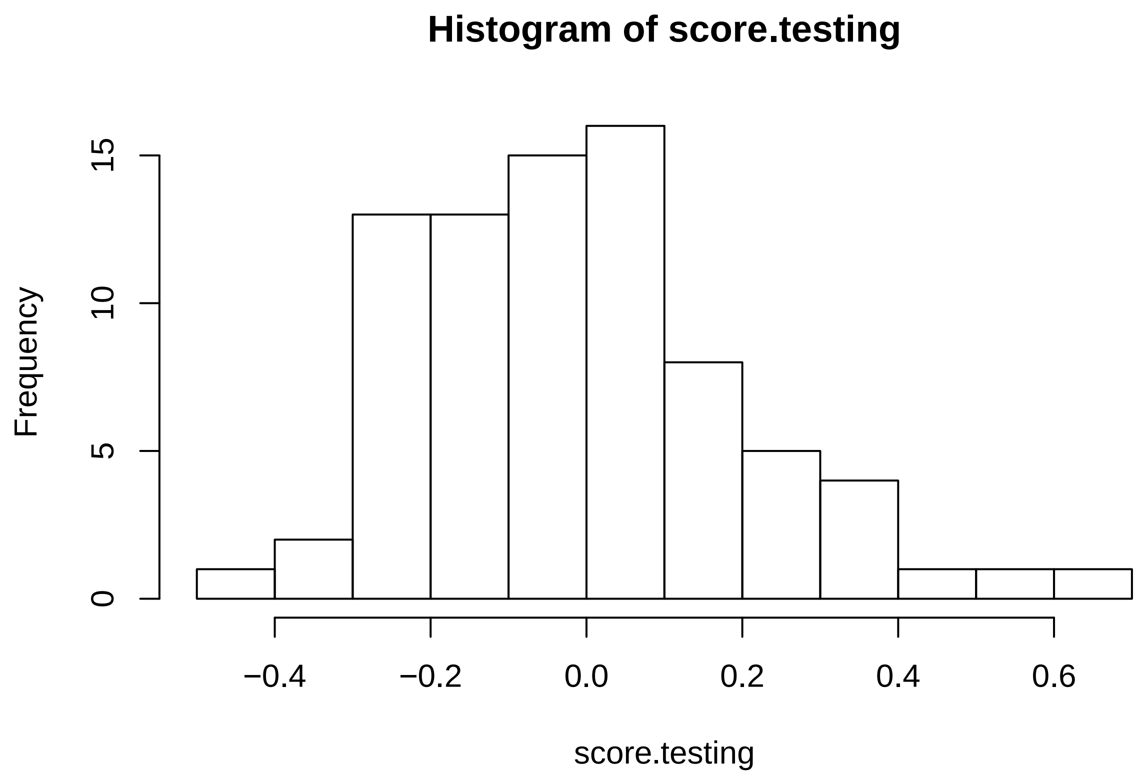
```r
round(b[b != 0], digits = 3)
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
## [1]  0.114  0.399  0.008 -0.078 -0.039 -0.016  0.008 -0.014  0.078
```

Test the model

```r
score.testing <- predict(fit.cv10, newx = gex.testing, s = "lambda.min")
hist(score.testing)
```



Question: how well is the score predicting survival?

A continuous predictor vs a right-censored time-to-failure outcome: Cox regression!

```r
summary(coxph(y.testing ~ score.testing))
```

```
## Call:
## coxph(formula = y.testing ~ score.testing)
##
##   n= 80, number of events= 47
##
##                   coef exp(coef) se(coef)     z Pr(>|z|)
## score.testing  2.5045   12.2369   0.7892 3.173  0.00151 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##               exp(coef) exp(-coef) lower .95 upper .95
## score.testing     12.24    0.08172     2.605     57.47
##
## Concordance= 0.624  (se = 0.046 )
## Likelihood ratio test= 9.6  on 1 df,   p=0.002
## Wald test            = 10.07  on 1 df,   p=0.002
## Score (logrank) test = 10.29  on 1 df,   p=0.001
```

```r
#higer score higher risk,
```

A more interpretable scale: by IQR variation:

```r
score_scaled.testing <- score.testing / IQR(score.testing)
summary(coxph(y.testing ~ score_scaled.testing))
```

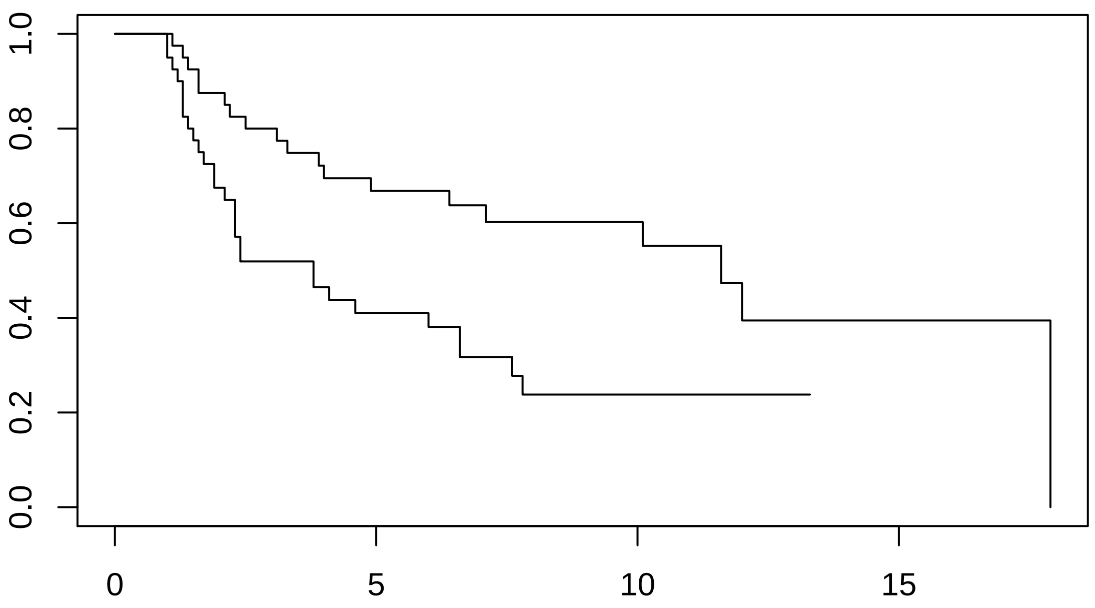
```
## Call:
## coxph(formula = y.testing ~ score_scaled.testing)
##
##   n= 80, number of events= 47
##
##                        coef exp(coef) se(coef)     z Pr(>|z|)
## score_scaled.testing 0.6758    1.9656   0.2130 3.173  0.00151 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                      exp(coef) exp(-coef) lower .95 upper .95
## score_scaled.testing     1.966     0.5087     1.295     2.984
##
## Concordance= 0.624  (se = 0.046 )
## Likelihood ratio test= 9.6  on 1 df,   p=0.002
## Wald test            = 10.07  on 1 df,   p=0.002
## Score (logrank) test = 10.29  on 1 df,   p=0.001
```

We can split the scores into 2 categories, and compare patients with ‘low’ vs ‘high’ score:

```r
gex_risk <- ifelse(score.testing <= median(score.testing), "low", "high")
table(gex_risk)
```

```
## gex_risk
## high  low
##   40   40
```

```r
fit.KM <- survfit(y.testing ~ gex_risk, conf.type = "log-log")
plot(fit.KM)
```



```
fit.KM
```

```
## Call: survfit(formula = y.testing ~ gex_risk, conf.type = "log-log")
##
##                 n events median 0.95LCL 0.95UCL
## gex_risk=high 40     28    3.8     2.1     6.6
## gex_risk=low  40     19   11.6     4.9      NA
```

are they significantly different? the p value is low so they are significantly different

```
survdiff(y.testing~gex_risk)
```

```
## Call:
## survdiff(formula = y.testing ~ gex_risk)
##
##                 N Observed Expected (O-E)^2/E (O-E)^2/V
## gex_risk=high 40       28     18.7      4.58      8.05
## gex_risk=low  40       19     28.3      3.04      8.05
##
##  Chisq= 8  on 1 degrees of freedom, p= 0.005
```

Question: how is the 6 months survival for patients classified as low risk, compared to patients classified as high risk?

```r
summary(fit.KM, time = 6)
```

```
## Call: survfit(formula = y.testing ~ gex_risk, conf.type = "log-log")
##
##                 gex_risk=high
##         time       n.risk      n.event     survival      std.err
##        6.000       14.000       24.000        0.381        0.079
## lower 95% CI upper 95% CI
##        0.229        0.531
##
##                 gex_risk=low
##         time       n.risk      n.event     survival      std.err
##       6.0000      22.0000      13.0000       0.6682       0.0755
## lower 95% CI upper 95% CI
##       0.4975       0.7923
```

AUC is high it's good

```r
library(survivalROC)
ROC <- survivalROC(Stime = y.testing[, 1],
                   status = y.testing[, 2],
                   marker = score.testing,
                   cut.values = quantile(score.testing, prob = 0:10/10),
                   predict.time = 10,
                   method = "KM")
ROC$AUC
```

```
## [1] 0.7307709
```