EST171 - Aprendizado de Máquina
Departamento de Estatística
Universidade Federal de Minas Gerais

Lista 1

Henrique Aparecido Laureano Arthur Tarso Rego

Setembro de 2016

Sumário

Exercício I

Os dados `worldDevelopmentIndicators.csv` contém os dados do PIB per capita (X) e a expectativa de vida (Y) de diversos países. O objetivo é criar preditores de Y com base em X. Em aula vimos como isso pode ser feito através de polinômios. Aqui, faremos isso via expansões de Fourier.

```
# <code r> ========================================================================= #
path <- "C:/Users/henri/Dropbox/Scripts/aprendizado de maquina/"

data1 <- read.csv(paste0(path, "worldDevelopmentIndicators.csv"))

summary(data1)
# </code r> ======================================================================== #
```

```
            CountryName  LifeExpectancy   GDPercapita
 Afghanistan      :  1   Min.   :45.33   Min.   :   251
 Albania          :  1   1st Qu.:64.06   1st Qu.:  1682
 Algeria          :  1   Median :72.49   Median :  5786
 Angola           :  1   Mean   :70.30   Mean   : 14150
 AntiguaandBarbuda:  1   3rd Qu.:76.75   3rd Qu.: 16863
 ArabWorld        :  1   Max.   :83.48   Max.   :103858
 (Other)          :205
```

a) Normalize a covariável de modo que $x \in (0, 1)$. Para isso, faça $x = (x - x_{\min})/(x_{\max} - x_{\min})$, em que $x_{\min}$ e $x_{\max}$ são os valores mínimos e máximos de x segundo a amostra usada.

```
# <code r> ========================================================================= #
names(data1)[3] <- "X"

data1$Xn <- with(data1, ( X - min(X) ) / ( max(X) - min(X) ) )

summary(data1$Xn)
# </code r> ======================================================================== #
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.01381 0.05342 0.13420 0.16030 1.00000
```

b) Usando o método dos mínimos quadrados e a validação cruzada do tipo *leave-one-out*, estime o erro quadrático médio das regressões

$$g(x) = \hat{\beta}_0 + \hat{\beta}_1 \sin(2\pi x) + \hat{\beta}_2 \cos(2\pi x) + \hat{\beta}_3 \sin(2\pi 2x) + \hat{\beta}_4 \cos(2\pi 2x) + \ldots + \hat{\beta}_{2p-1} \sin(2\pi p x) + \hat{\beta}_{2p} \cos(2\pi p x)$$

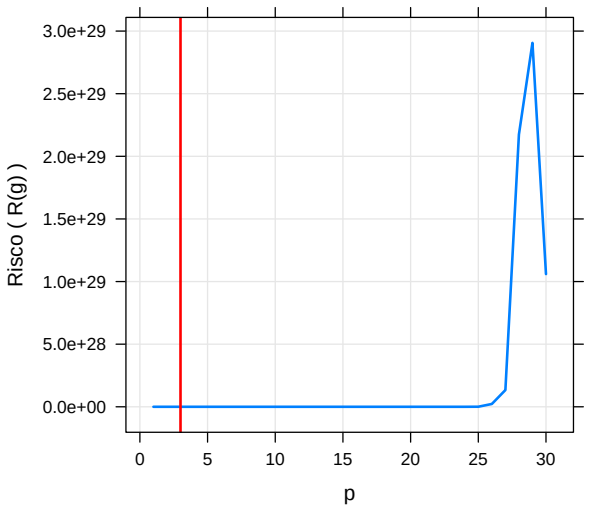
para $p = 1, ..., 30$**.**

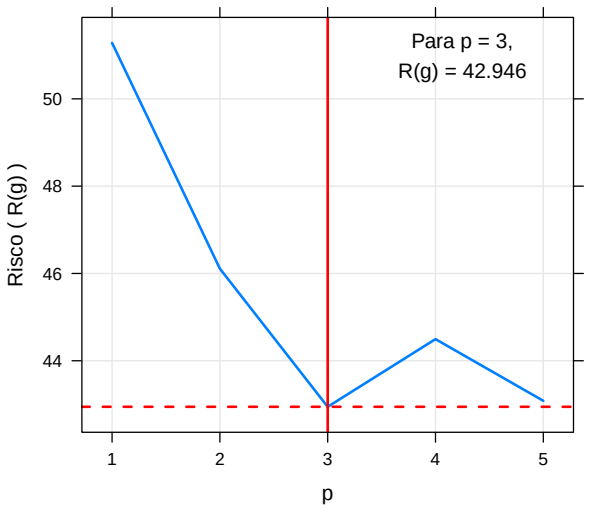
```r
# <code r> ========================================================================== #
library(cvTools)

Y <- data1[ , 2] ; X <- data1[ , 3] ; Xn <- data1[ , 4]

mse <- 0

for (p in 1:30){

  if (p == 1){

    seno <- sin(2 * pi * Xn) ; coss <- cos(2 * pi * Xn)

    XLM <- cbind(seno, coss)

    fit <- lm(Y ~ XLM)

    rmse <- repCV(fit, K = length(Xn)) ; mse[p] <- rmse$cv[[1]] ** 2}

  if (p > 1){

    seno <- sin(2 * pi * Xn) ; coss <- cos(2 * pi * Xn)

    XLM <- cbind(seno, coss)

    for (i in 2:p){

      seno <- sin(2 * pi * i * Xn) ; coss <- cos(2 * pi * i * Xn)

      XLM <- cbind(XLM, cbind(seno, coss))}

    fit <- lm(Y ~ XLM)

    rmse <- repCV(fit, K = length(Xn)) ; mse[p] <- rmse$cv[[1]] ** 2}}

mse
# </code r> ========================================================================= #
```

```
 [1] 5.128216e+01 4.610982e+01 4.294558e+01 4.449556e+01 4.308229e+01
 [6] 5.476830e+01 4.468426e+02 1.496953e+02 2.398555e+03 5.781719e+03
[11] 3.577478e+02 5.836964e+06 3.178243e+07 2.382372e+09 1.304575e+11
[16] 2.054332e+12 5.665055e+13 1.752371e+12 4.845864e+16 5.272471e+17
[21] 1.690995e+19 1.001170e+20 3.224824e+23 8.684489e+23 5.385807e+25
[26] 2.250809e+27 1.336144e+28 2.174205e+29 2.905797e+29 1.059644e+29
```

c) Plote o gráfico do risco estimado vs p. Qual o valor de p escolhido? Denotaremos ele por p_{esc}

```r
# <code r> ========================================================================= #
library(latticeExtra)

print(xyplot(mse ~ 1:30
             , type = c("l", "g")
             , lwd = 2
             , xlab = "p"
             , ylab = "Risco ( R(g) )"
             , panel = function(...){
               panel.xyplot(...)
               panel.abline(v = which.min(mse), col = 2, lwd = 2)})
      , position = c(0, 0, .5, 1)
      , more= TRUE)

print(xyplot(mse[1:5] ~ 1:5
             , type = c("l", "g")
             , lwd = 2
             , xlab = "p"
             , ylab = "Risco ( R(g) )"
             , panel = function(...){
               panel.xyplot(...)
               panel.abline(v = which.min(mse), col = 2, lwd = 2)
               panel.abline(h = min(mse), col = 2, lwd = 2, lty = 2)
               panel.text(4.25, 51, labels = paste(
                 "Para p = 3,\nR(g) =", round(min(mse), 3)))})
      , position = c(.5, 0, 1, 1))
# </code r> ========================================================================= #
```

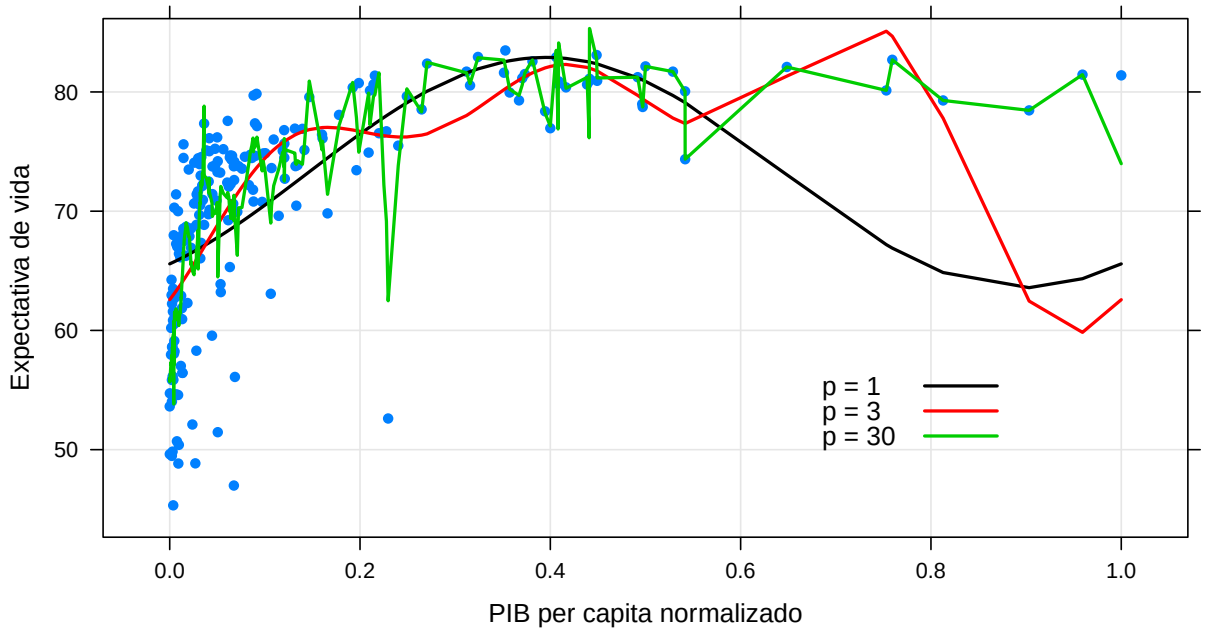




d) Plote as curvas ajustadas para $p = 1$, $p = p_{esc}$ e $p = 30$ sob o gráfico de dispersão de X por Y. Qual curva parece mais razoável? Use um grid de valores entre 0 e 1 para isso. Como estes ajustes se comparam com o visto em aula via polinômios? Discuta.

```r
# <code r> ========================================================================== #
# p = 1 ------------------------------------------------------------------------
seno <- sin(2 * pi * Xn) ; coss <- cos(2 * pi * Xn)

XLM <- cbind(seno, coss)

p1 <- lm(Y ~ XLM)

data.p1 <- data.frame(X = Xn, Y = fitted(p1))
data.p1 <- data.p1[order(data.p1$X), ]

# p = 3 ------------------------------------------------------------------------
seno <- sin(2 * pi * Xn) ; coss <- cos(2 * pi * Xn)

XLM <- cbind(seno, coss)

for (i in 2:3){

  seno <- sin(2 * pi * i * Xn) ; coss <- cos(2 * pi * i * Xn)

  XLM <- cbind(XLM, cbind(seno,coss))}

p3 <- lm(Y ~ XLM)

data.p3 <- data.frame(X = Xn, Y = fitted(p3))
data.p3 <- data.p3[order(data.p3$X), ]

# p = 30 -----------------------------------------------------------------------
seno <- sin(2 * pi * Xn) ; coss <- cos(2 * pi * Xn)

XLM <- cbind(seno, coss)

for (i in 2:30){

  seno <- sin(2 * pi * i * Xn) ; coss <- cos(2 * pi * i * Xn)

  XLM <- cbind(XLM, cbind(seno, coss))}

p30 <- lm(Y ~ XLM)

data.p30 <- data.frame(X = Xn, Y = fitted(p30))
```

```r
data.p30 <- data.p30[order(data.p30$X), ]

xyplot(Y ~ Xn, data1
       , type = c("p", "g")
       , pch = 16
       , xlab = "PIB per capita normalizado"
       , ylab = "Expectativa de vida"
       , key = list(corner = c(.8, .2)
                    , text = list(c("p = 1", "p = 3", "p = 30"))
                    , lines = list(lwd = 2, col = 1:3))) +
  as.layer(xyplot(data.p1$Y ~ data.p1$X, col = 1, type = "l", lwd = 2)) +
  as.layer(xyplot(data.p3$Y ~ data.p3$X, col = 2, type = "l", lwd = 2)) +
  as.layer(xyplot(data.p30$Y ~ data.p30$X, col = 3, type = "l", lwd = 2))
# </code r> ========================================================================== #
```



A curva que parece ser mais razoável é a para um $p = 3$. A curva para $p = 1$ é muito suave, apresentando um comportamento um tanto ingênuo. Já a curva para $p = 30$ apresenta um *overfitting*, correndo muito atrás dos dados.

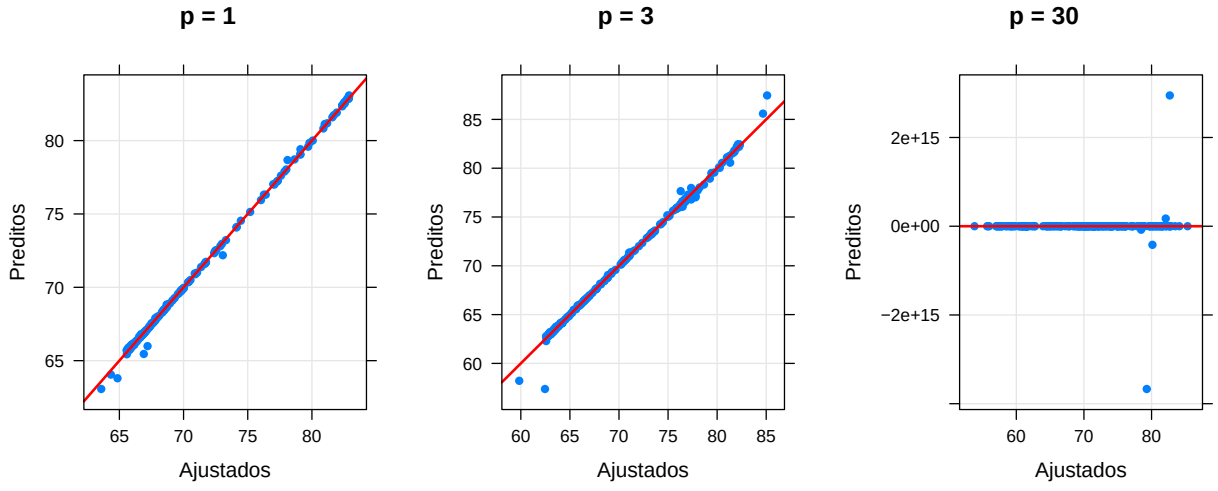
Em comparação com os polinômios, esses ajustes apresentam uma menor variabilidade nas regiões de maior incerteza, i.e., nas regiões com menos observações, assumindo comportamentos mais constantes e menos suscetíveis as poucas observações existentes.

e) Plote o gráfico de valores preditos vs ajustados para $p = 1$, $p = p_{esc}$ e $p = 30$ (não se esqueça de usar o *leave-one-out* para calcular os valores preditos! Caso contrário você terá problemas de overfitting novamente). Qual p parece ser mais razoável?

```r
# <code r> ========================================================================== #
# p = 1 ------------------------------------------------------------------------
pred.p1 <- 0

for (i in 1:length(Xn)){

  Xn.fit <- Xn[-i]
  Xn.pred <- Xn[i]

  Y.fit <- Y[-i]
  Y.pred <- Y[i]

  seno <- sin(2 * pi * Xn.fit) ; coss <- cos(2 * pi * Xn.fit)
  XLM <- data.frame(seno, coss)

  p1.cv <- lm(Y.fit ~ seno + coss, XLM)

  seno <- sin(2 * pi * Xn.pred) ; coss <- cos(2 * pi * Xn.pred)
  XLM.pred <- data.frame(seno, coss)

  pred.p1[i] <- predict(p1.cv, XLM.pred)}

# p = 3 ------------------------------------------------------------------------
pred.p3 <- 0

for (i in 1:length(Xn)){

  Xn.fit <- Xn[-i]
  Xn.pred <- Xn[i]

  Y.fit <- Y[-i]
  Y.pred <- Y[i]

  seno <- sin(2 * pi * 1 *Xn.fit) ; coss <- cos(2 * pi * 1 * Xn.fit)
  seno.coss <- cbind(seno, coss)

  for (j in 2:3){

    seno <- sin(2 * pi * j * Xn.fit) ; coss <- cos(2 * pi * j * Xn.fit)
    seno.coss <- cbind(seno.coss, seno, coss)}

  XLM <- data.frame(seno.coss) ; names(XLM) <- as.character(1:6)

  p3.cv <- lm(Y.fit ~ ., XLM)

  seno <- sin(2 * pi * 1 * Xn.pred) ; coss <- cos(2 * pi * 1 * Xn.pred)
```

```
  seno.coss <- cbind(seno, coss)

  for (j in 2:3){

    seno <- sin(2 * pi * j * Xn.pred) ; coss <- cos(2 * pi * j * Xn.pred)
    seno.coss <- cbind(seno.coss, seno, coss)}

  XLM.pred <- data.frame(seno.coss) ; names(XLM.pred) <- as.character(1:6)

  pred.p3[i] <- predict(p3.cv, XLM.pred)}

# p = 30 ------------------------------------------------------------------
pred.p30 <- 0

for (i in 1:length(Xn)){

  Xn.fit <- Xn[-i]
  Xn.pred <- Xn[i]

  Y.fit <- Y[-i]
  Y.pred <- Y[i]

  seno <- sin(2 * pi * 1 * Xn.fit) ; coss <- cos(2 * pi * 1 * Xn.fit)
  seno.coss <- cbind(seno, coss)

  for (j in 2:30){

    seno <- sin(2 * pi * j * Xn.fit) ; coss <- cos(2 * pi * j * Xn.fit)
    seno.coss <- cbind(seno.coss, seno, coss)}

  XLM <- data.frame(seno.coss) ; names(XLM) <- as.character(1:60)

  p30.cv <- lm(Y.fit ~ ., XLM)

  seno <- sin(2 * pi * 1 * Xn.pred) ; coss <- cos(2 * pi * 1 * Xn.pred)
  seno.coss <- cbind(seno, coss)

  for (j in 2:30){
    seno <- sin(2 * pi * j * Xn.pred) ; coss <- cos(2 * pi * j * Xn.pred)
    seno.coss <- cbind(seno.coss, seno, coss)}

  XLM.pred <- data.frame(seno.coss) ; names(XLM.pred) <- as.character(1:60)

  pred.p30[i] <- predict(p30.cv, XLM.pred)}

print(xyplot(pred.p1 ~ fitted(p1)
             , type = c("p", "g")
```

```r
              , pch = 16
              , xlab = "Ajustados"
              , ylab = "Preditos"
              , main = "p = 1"
              , panel = function(...){
                panel.xyplot(...)
                panel.abline(0, 1, col = 2, lwd = 2)})
       , position = c(0, 0, 1/3, 1)
       , more = TRUE)

print(xyplot(pred.p3 ~ fitted(p3)
              , type = c("p", "g")
              , pch = 16
              , xlab = "Ajustados"
              , ylab = "Preditos"
              , main = "p = 3"
              , panel = function(...){
                panel.xyplot(...)
                panel.abline(0, 1, col = 2, lwd = 2)})
       , position = c(1/3, 0, 2/3, 1)
       , more = TRUE)

print(xyplot(pred.p30 ~ fitted(p30)
              , type = c("p", "g")
              , pch = 16
              , xlab = "Ajustados"
              , ylab = "Preditos"
              , main = "p = 30"
              , panel = function(...){
                panel.xyplot(...)
                panel.abline(0, 1, col = 2, lwd = 2)})
       , position = c(2/3, 0, 1, 1))
# </code r> ========================================================================== #
```



Tanto o $p = 1$ quanto o $p = 3$ se apresentam razoáveis, entretanto, para $p = 3$ as observações mais extremas tiveram predições não muito boas. Para $p = 30$ duas observações resultaram em predições extremamentes divergentes.

Feitas tais observações, o p que parece ser mais razoável é o $p = 3$.

f) Quais vantagens e desvantagens de se usar validação cruzada do tipo *leave-one-out* vs o *data-splitting*?

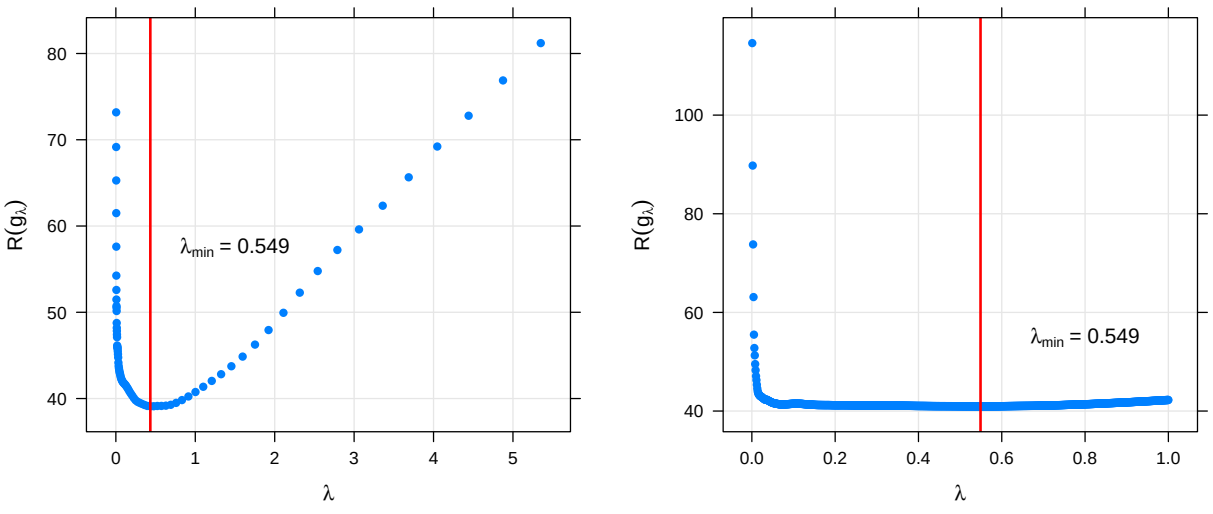
A validação cruzada do tipo *leave-one-out* é de maior custo computacional, já que consiste na retirada sistemática de cada observação da base de dados de treino, se tornando inviável no contexto de conjuntos de dados muito grandes ou de recursos de processamento limitados.

Contudo, ela se mostra mais vantajosa do que a validação cruzada do tipo *data-splitting* (dividir um conjunto de dados em base de treino, teste e de validação) por possibilitar a avaliação do impacto de cada observação.

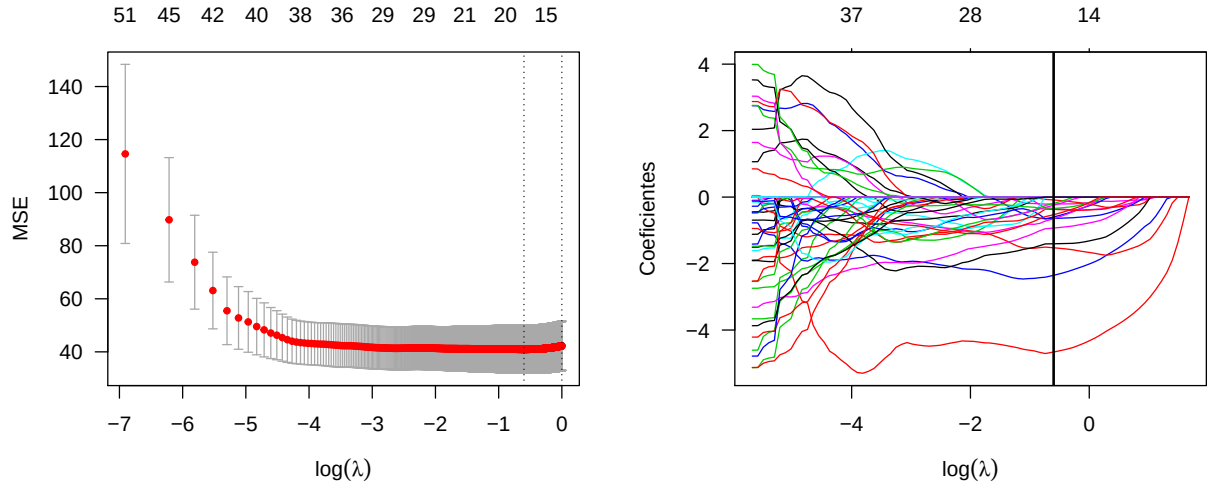
g) Ajuste a regressão Lasso (Frequentista e Bayesiana) e discuta os resultados encontrados.

```r
# <code r> ========================================================================= #
# Lasso Frequentista ------------------------------------------------------------
library(glmnet)

set.seed(22)

Xn.train <- Xn[1:150] ; Y.train <- Y[1:150]

Xn.test <- Xn[151:211] ; Y.test <- Y[151:211]

seno <- sin(2 * pi * Xn.train) ; coss <- cos(2 * pi * Xn.train)
XLM.train <- cbind(seno, coss)

for (i in 2:30){

  seno <- sin(2 * pi * i * Xn.train) ; coss <- cos(2 * pi * i * Xn.train)
  XLM.train <- cbind(XLM.train, cbind(seno, coss))}

seno <- sin(2 * pi * Xn.test) ; coss <- cos(2 * pi * Xn.test)
XLM.test <- cbind(seno, coss)

for (i in 2:30){

  seno <- sin(2 * pi * i * Xn.test) ; coss <- cos(2 * pi * i * Xn.test)
  XLM.test <- cbind(XLM.test, cbind(seno, coss))}
```

```r
lasso <- glmnet(cbind(1, XLM.train), Y.train, alpha = 1)

cv <- cv.glmnet(cbind(1, XLM.train), Y.train, alpha = 1)

print(xyplot(cv$cvm ~ cv$lambda
             , xlab = expression(lambda)
             , ylab = expression(R(g[lambda]))
             , type = c("p", "g")
             , pch = 16
             , panel = function(...){
               panel.xyplot(...)
               panel.abline(v = cv$lambda.min, col = 2, lwd = 2)
               panel.text(
                 1.5, 57.5, labels = expression(lambda[min]~"= 0.549"))})
      , position = c(0, 0, .5, 1)
      , more = TRUE)

cv <- cv.glmnet(cbind(1, XLM.train), Y.train
                , alpha = 1, lambda = seq(1, .001, length.out = 1000))

print(xyplot(cv$cvm ~ cv$lambda
             , xlab = expression(lambda)
             , ylab = expression(R(g[lambda]))
             , type = c("p", "g")
             , pch = 16
             , panel = function(...){
               panel.xyplot(...)
               panel.abline(v = cv$lambda.min, col = 2, lwd = 2)
               panel.text(.8, 55, labels = expression(lambda[min]~"= 0.549"))})
      , position = c(.5, 0, 1, 1))
# </code r> ========================================================================== #
```



```r
# <code r> ========================================================================== #
par(mfrow = c(1, 2))

plot(cv, las = 1, xlab = expression(log(lambda)), ylab = "MSE")

plot(lasso, xvar = "lambda"
     , las = 1, xlab = expression(log(lambda)), ylab = "Coeficientes")

abline(v = log(cv$lambda.min), lwd = 2)
# </code r> ========================================================================= #
```



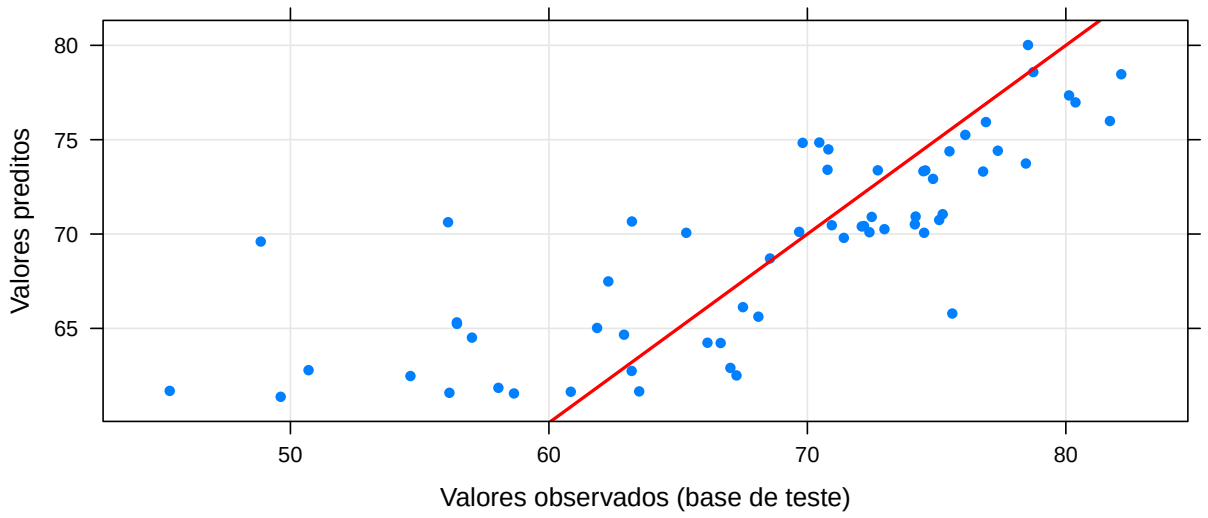

```r
# <code r> ========================================================================== #
i <- 0

for (i in 1:length(coef(lasso)[ , 26])){ # lambda mais próximo do obtido via cv

  if (coef(lasso)[i, 26] != 0){

    if (i == 1){

      cat("Intercepto =", coef(lasso)[i, 26], '\n')}

    else {

      cat("Beta[",i - 1,"] =", coef(lasso)[i, 26], '\n')}}}
# </code r> ========================================================================= #
```

```
Intercepto = 76.31641
Beta[ 3 ] = -4.678484
Beta[ 5 ] = -2.374822
Beta[ 7 ] = -0.2094418
```

```
Beta[ 9 ] = -1.518322
Beta[ 11 ] = -0.618371
Beta[ 13 ] = -0.3357034
Beta[ 15 ] = -0.08094573
Beta[ 17 ] = -0.646376
Beta[ 19 ] = -0.6773292
Beta[ 21 ] = -0.3690756
Beta[ 27 ] = -0.9395936
Beta[ 35 ] = -1.406763
Beta[ 37 ] = -0.3527128
Beta[ 44 ] = -0.2144417
Beta[ 53 ] = -0.2706215
Beta[ 61 ] = -0.5515885
```

```r
# <code r> ========================================================================== #
pred.lasso <- predict(lasso, s = cv$lambda.min, newx = cbind(1, XLM.test))

xyplot(pred.lasso ~ Y.test
       , pch = 16
       , type = c("p", "g")
       , xlab = "Valores observados (base de teste)"
       , ylab = "Valores preditos"
       , panel = function(...){
         panel.xyplot(...)
         panel.abline(0, 1, col = 2, lwd = 2)})

mean( (pred.lasso - Y.test)**2 )
# </code r> ========================================================================== #
```

```
[1] 34.18578
```

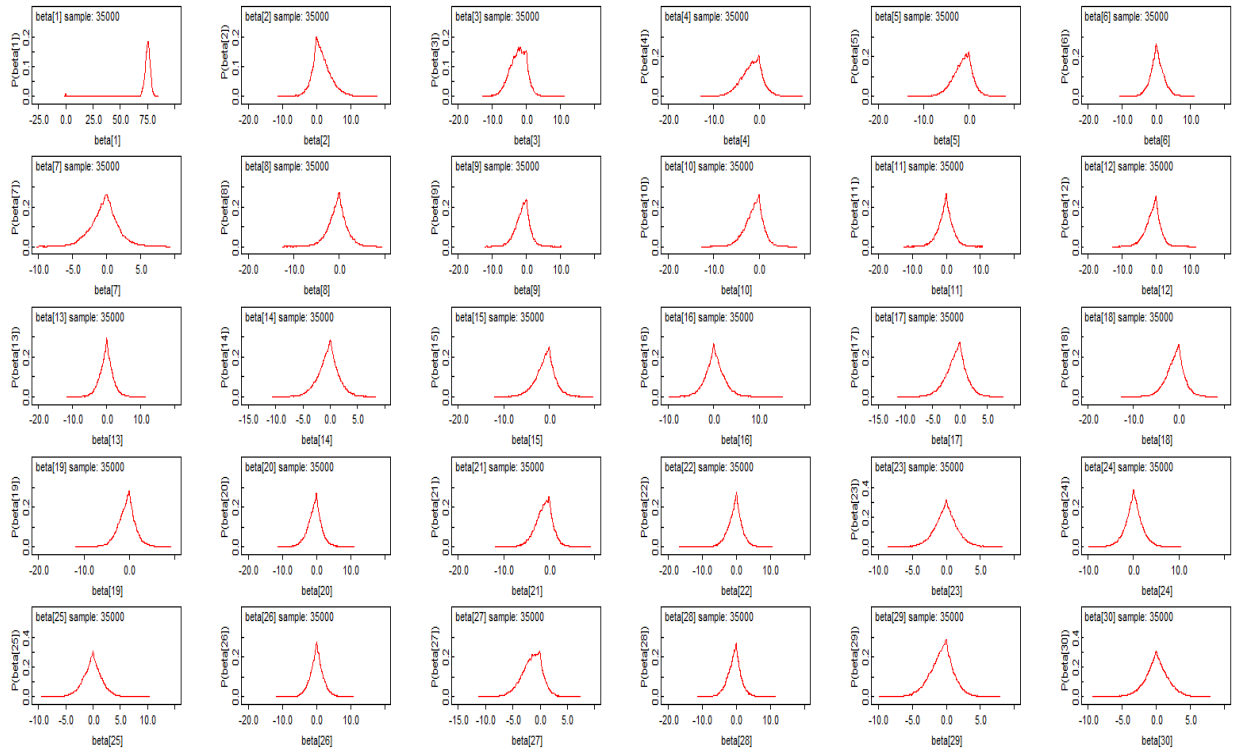


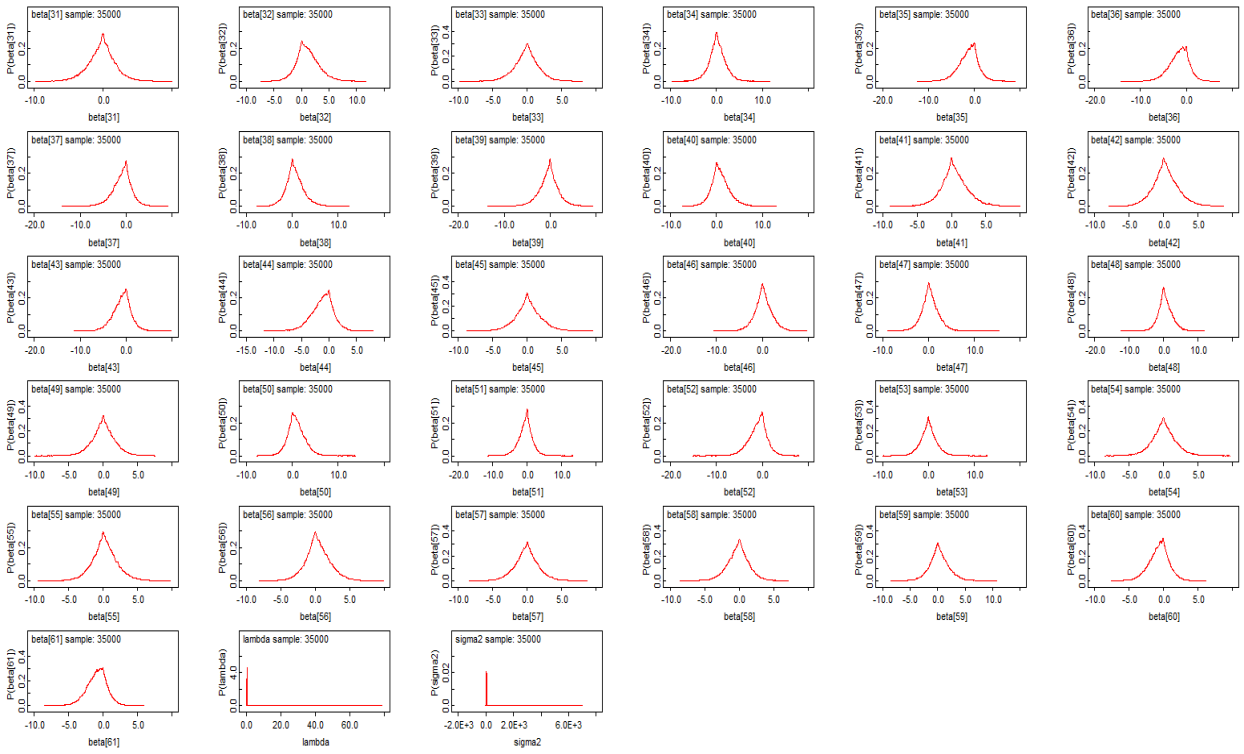
```r
# <code r> ========================================================================== #
# Lasso Bayesiana ----------------------------------------------------------------
library(rbugs)

M <- length(Y.train)

X <- XLM.train
X <- cbind(rep(1, M), X)

P <- ncol(X)

Y <- Y.train

N <- M

model.file <- file.path("model.txt")

data <- list ("N", "Y", "X", "P")

inits <- list(list(tau = 1, beta = rep(0, P), lambda = 1))

parameters <- c("tau", "beta", "sigma2", "sigma", "lambda")

model <- rbugs(data, inits, parameters, model.file
               , n.chains = 1
               ,  n.iter = 20000
               , n.burnin = 5000
               , n.thin = 10
               , bugs =
                 "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe"
               , bugsWorkingDir = "BUGS/")
# </code r> ========================================================================== #
```

Coef	Média	SD	Erro-MC	IC 2.5%	Mediana	IC 97.5%	Inicial	Amostra
beta[1]	74.09	9.40	0.82	69.53	75.28	79.70	1	15000
beta[2]	1.53	2.82	0.17	-3.37	1.12	8.01	1	15000
beta[3]	-2.41	2.49	0.12	-7.63	-2.25	1.90	1	15000
beta[4]	-1.73	2.50	0.10	-7.15	-1.49	2.88	1	15000
beta[5]	-1.41	2.16	0.08	-6.05	-1.16	2.52	1	15000
beta[6]	0.26	2.12	0.08	-4.01	0.16	4.64	1	15000
beta[7]	-0.37	2.06	0.06	-4.63	-0.24	3.79	1	15000
beta[8]	-0.36	2.11	0.06	-4.96	-0.26	3.89	1	15000
beta[9]	-1.15	2.09	0.07	-5.70	-0.94	2.69	1	15000
beta[10]	-1.04	1.97	0.06	-5.39	-0.82	2.52	1	15000
beta[11]	-0.20	2.21	0.08	-4.86	-0.11	4.24	1	15000

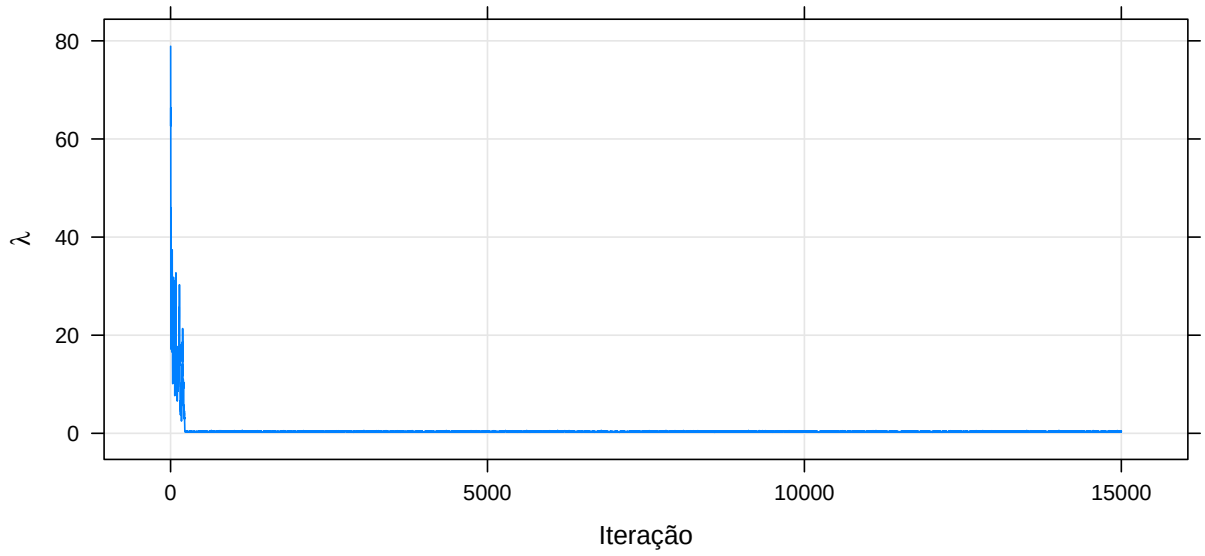
Coef	Média	SD	Erro-MC	IC 2.5%	Mediana	IC 97.5%	Inicial	Amostra
beta[12]	-1.03	2.27	0.08	-6.10	-0.77	3.14	1	15000
beta[13]	-0.05	1.94	0.06	-4.16	-0.00	3.86	1	15000
beta[14]	-0.38	1.93	0.06	-4.38	-0.28	3.54	1	15000
beta[15]	-0.83	2.27	0.09	-5.62	-0.65	3.53	1	15000
beta[16]	0.19	2.23	0.08	-4.37	0.12	4.86	1	15000
beta[17]	-0.78	1.91	0.06	-4.82	-0.63	2.92	1	15000
beta[18]	-0.75	1.98	0.05	-4.98	-0.57	3.03	1	15000
beta[19]	-0.64	2.02	0.07	-4.92	-0.49	3.38	1	15000
beta[20]	-0.56	2.14	0.07	-4.96	-0.45	3.90	1	15000
beta[21]	-1.09	1.89	0.06	-5.17	-0.90	2.39	1	15000
beta[22]	-0.39	2.16	0.06	-5.04	-0.25	3.87	1	15000
beta[23]	-0.03	1.75	0.05	-3.79	-0.02	3.56	1	15000
beta[24]	0.10	1.96	0.06	-3.88	0.06	4.23	1	15000
beta[25]	-0.09	1.85	0.05	-4.00	-0.05	3.67	1	15000
beta[26]	-0.03	2.09	0.06	-4.35	-0.00	4.26	1	15000
beta[27]	-1.48	1.96	0.07	-5.71	-1.29	2.02	1	15000
beta[28]	-0.59	2.08	0.07	-5.00	-0.47	3.60	1	15000
beta[29]	-0.71	1.85	0.06	-4.77	-0.54	2.78	1	15000
beta[30]	0.08	1.82	0.06	-3.73	0.05	3.87	1	15000
beta[31]	-0.14	1.94	0.06	-4.17	-0.09	3.78	1	15000
beta[32]	1.11	2.09	0.07	-2.74	0.91	5.71	1	15000
beta[33]	-0.16	1.81	0.06	-3.96	-0.09	3.40	1	15000
beta[34]	0.24	1.77	0.05	-3.35	0.18	3.94	1	15000
beta[35]	-1.20	2.13	0.07	-5.79	-0.95	2.77	1	15000
beta[36]	-1.63	2.20	0.09	-6.46	-1.39	2.20	1	15000
beta[37]	-0.86	2.11	0.07	-5.72	-0.62	2.95	1	15000
beta[38]	0.55	1.91	0.07	-3.08	0.39	4.83	1	15000
beta[39]	-0.34	2.15	0.08	-4.93	-0.23	3.98	1	15000
beta[40]	0.84	2.00	0.07	-2.85	0.64	5.27	1	15000
beta[41]	0.45	1.92	0.06	-3.33	0.31	4.59	1	15000
beta[42]	0.32	1.87	0.05	-3.41	0.23	4.34	1	15000
beta[43]	-0.88	1.95	0.06	-4.96	-0.70	2.80	1	15000
beta[44]	-1.18	1.99	0.07	-5.48	-0.99	2.46	1	15000
beta[45]	0.06	1.89	0.07	-3.85	0.04	4.00	1	15000
beta[46]	0.27	1.93	0.06	-3.67	0.19	4.37	1	15000
beta[47]	0.38	1.88	0.05	-3.38	0.27	4.39	1	15000
beta[48]	0.59	2.13	0.07	-3.35	0.38	5.29	1	15000
beta[49]	0.09	1.70	0.05	-3.38	0.06	3.61	1	15000
beta[50]	0.84	1.90	0.05	-2.78	0.69	4.88	1	15000
beta[51]	-0.53	2.07	0.07	-5.02	-0.37	3.51	1	15000
beta[52]	-1.02	2.09	0.07	-5.54	-0.77	2.75	1	15000
beta[53]	-0.02	1.80	0.05	-3.62	-0.03	3.69	1	15000
beta[54]	0.14	1.81	0.05	-3.41	0.04	4.16	1	15000

Coef	Média	SD	Erro-MC	IC 2.5%	Mediana	IC 97.5%	Inicial	Amostra
beta[55]	0.20	1.86	0.05	-3.60	0.14	4.14	1	15000
beta[56]	0.53	1.84	0.05	-3.06	0.37	4.46	1	15000
beta[57]	-0.02	1.75	0.05	-3.49	-0.03	3.71	1	15000
beta[58]	-0.21	1.60	0.05	-3.62	-0.12	2.89	1	15000
beta[59]	0.34	1.80	0.04	-3.19	0.23	4.15	1	15000
beta[60]	-0.69	1.44	0.03	-3.77	-0.57	2.05	1	15000
beta[61]	-0.87	1.46	0.04	-3.94	-0.77	1.88	1	15000
lambda	0.58	2.26	0.17	0.25	0.36	0.51	1	15000
sigma2	113.20	622.70	54.03	28.89	36.81	50.51	1	15000

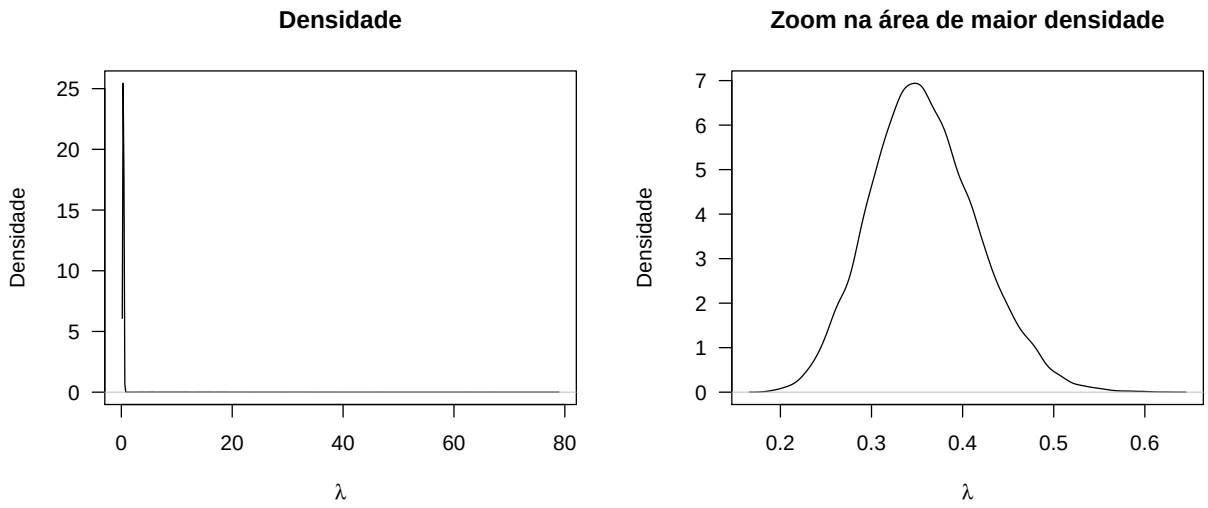





```
# <code r> ============================================================== #
lambda.iter <- read.table(paste0(path, "lambda-iter.txt"), header = TRUE)

xyplot(Lambda ~ Ite, lambda.iter
       , type = c("l", "g")
       , xlab = "Iteração"
       , ylab = expression(lambda))
# </code r> ============================================================== #
```



```r
# <code r> ========================================================================== #
par(mfrow = c(1, 2))

plot(density(lambda.iter$Lambda)
     , las = 1
     , xlab = expression(lambda)
     , ylab = "Densidade"
     , main = "Densidade")

plot(density(lambda.iter[250:15000, "Lambda"])
     , las = 1
     , xlab = expression(lambda)
     , ylab = "Densidade"
     , main = "Zoom na área de maior densidade")
# </code r> ========================================================================== #
```



Análise dos resultados

- λ: As diferentes abordagens resultaram em λ's muito próximos;
- $\beta's$: Pela abordagem Bayesiana, com exceção do intercepto os intervalos de credibilidade de todos os $\beta's$ comtemplaram o zero, contudo, cabe ressaltar que estes não foram intervalos do tipo HPD. Logo, a princípio não podemos dizer que esses coeficientes não foram significativos, mas por uma análise dos gráficos os $\beta's$ 1 (intercepto), 3, 4, 5, 10, 27, 35, 36, 44, 60 e 61 (onze $\beta's$) podem ser considerados como significativos, já que possuem áreas com grande densidade fora do ponto zero. Já na abordagem frequentista, dezesseis coeficientes não foram zerados, i.e., foram significativos. A saber: β 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 27, 35, 37, 44, 53 e 61. Os coeficientes que coincidiram foram os $\beta's$ 3, 5, 27, 35, 44 e 61 (seis coeficientes).

Exercício II

Neste exercício você irá implementar algumas técnicas vistas em aula para o banco de dados das faces. O objetivo é conseguir criar uma função que consiga predizer para onde uma pessoa está olhando com base em uma foto. Iremos aplicar KNN para esses dados, assim como uma regressão linear. Como não é possível usar o método dos mínimos quadrados quando o número de covariáveis é maior que o número de observações, para esta segunda etapa iremos usar o lasso.

a) Leia o banco *dadosFacesAltaResolucao.txt*. A primeira coluna deste banco contém a variável que indica a direção para a qual o indivíduo na imagem está olhando. As outras covariáveis contém os pixels relativos a essa imagem, que possui dimensão 64 por 64. Utilizando os comandos fornecidos, plot cinco imagens deste banco.

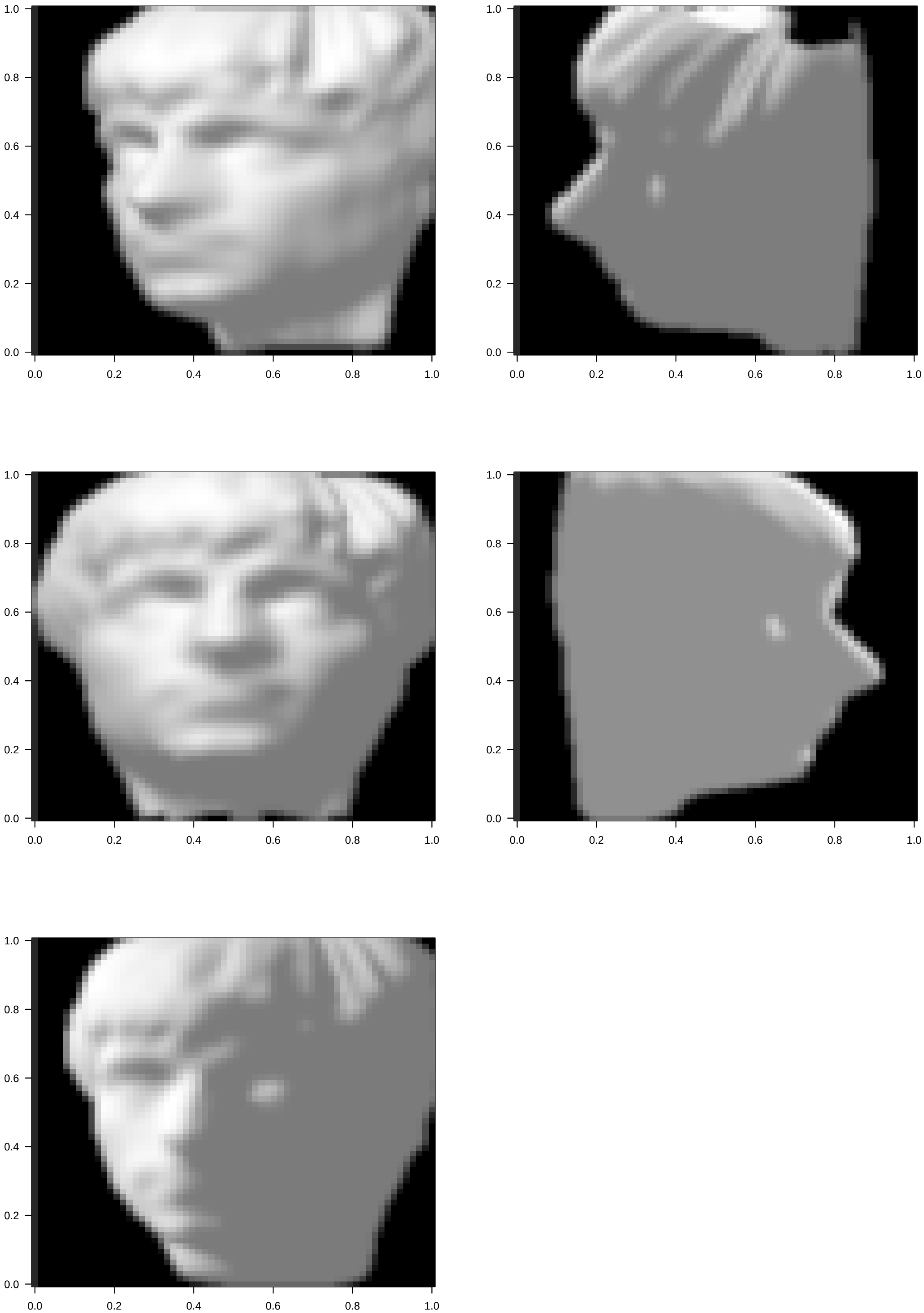
Divida o conjunto fornecido em treinamento (aproximadamente 60% das observações), validação (aproximadamente 20% das observações) e teste (aproximadamente 20% das observações). Utilizaremos o conjunto de treinamento e validação para ajustar os modelos. O conjunto de teste será utilizado para testar sua performanece

---

```r
# <code r> ========================================================================== #
data2 <- read.table(paste0(path, "dadosFacesAltaResolucao.txt"), header = TRUE)

data2_train <- data2[c(1:420), ] # 420/698

data2_val <- data2[c(421:559), ]

data2_test <- data2[c(560:698), ] # 560/698

library(jpeg)

count <- 1

M <- matrix(data = NA
            , nrow = 64
            , ncol = 64)

for (i in 1:64){

  for (j in 1:64){

    M[i, j] <- data2[1, 1 + count]
```

```
    count <- count + 1}}

par(mfrow = c(3, 2))

image(t(M)
      , las = 1
      , col = grey.colors(1000
                          , start = 0
                          , end = 1))

count <- 1

M <- matrix(data = NA
            , nrow = 64
            , ncol = 64)

for (i in 1:64){

  for (j in 1:64){

    M[i, j] <- data2[which.min(data2$y), 1 + count]

    count <- count + 1}}

image(t(M)
      , las = 1
      , col = grey.colors(1000
                          , start = 0
                          , end = 1))

# subset(data2, y > -1 & y < 1)$y

# row.names(data2[data2[ , "y"] == -0.39797, ])

count <- 1

M <- matrix(data = NA
            , nrow = 64
            , ncol = 64)

for (i in 1:64){

  for (j in 1:64){

    # as.numeric(row.names(data2[data2[ , "y"] == -0.39797, ]))

    M[i, j] <- data2[509, 1 + count]
```

```
    count <- count + 1}}

image(t(M)
      , las = 1
      , col = grey.colors(1000
                          , start = 0
                          , end = 1))

count <- 1

M <- matrix(data = NA
            , nrow = 64
            , ncol = 64)

for (i in 1:64){

  for (j in 1:64){

    M[i, j] <- data2[which.max(data2$y), 1 + count]

    count <- count + 1}}

image(t(M)
      , las = 1
      , col = grey.colors(1000
                          , start = 0
                          , end = 1))

count <- 1

M <- matrix(data = NA
            , nrow = 64
            , ncol = 64)

for (i in 1:64){

  for (j in 1:64){

    M[i, j] <- data2[698, 1 + count]

    count <- count + 1}}

image(t(M)
      , las = 1
      , col = grey.colors(1000
                          , start = 0, end = 1))
# </code r> ========================================================================== #
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.2 0.4 0.6 0.8 1.0

b) Qual o número de observações? Qual o número de covariáveis? O que representa cada covariável?

São 698 observações (420 para treino, 139 para validação e 139 para teste) e 4096 covariáveis, sendo que cada covariável representa a cor relativa ao pixel de sua posição na matriz da imagem.

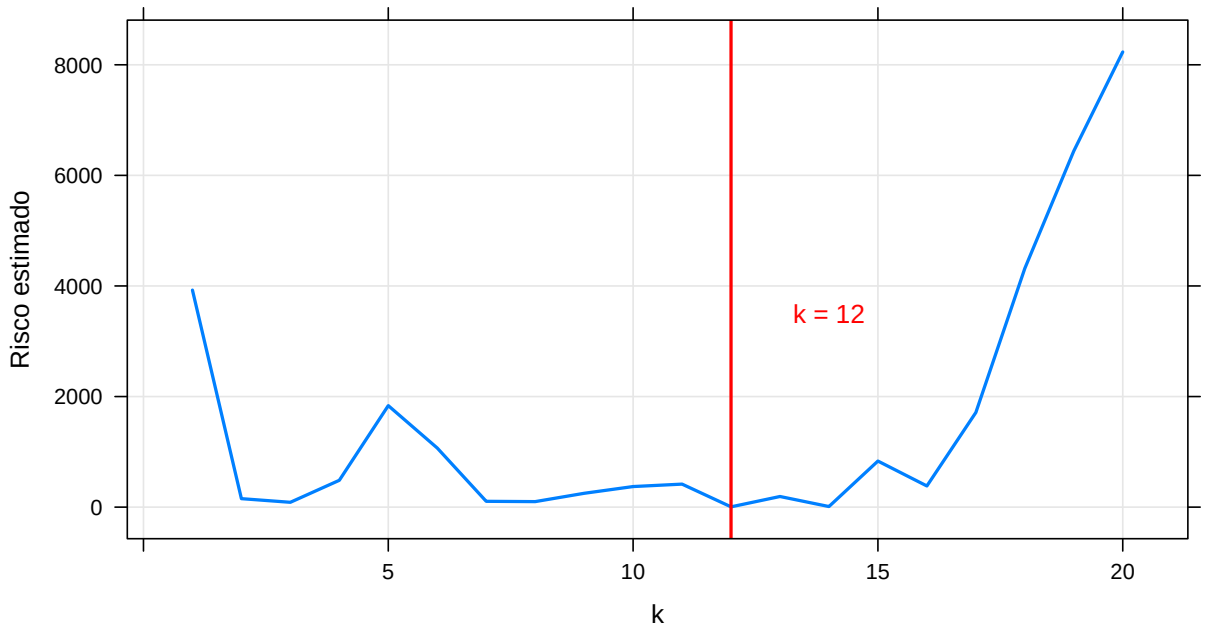
c) Para cada observação do conjunto de teste, calcule o estimador da função de regressão $r()$ dado pelo método dos k vizinhos mais próximos com $k = 5$. Você pode usar as funções vistas em aula.

```
# <code r> ============================================================== #
library(FNN)

knn.reg(data2_train[ , -1], data2_test[ , -1], data2_train[ , 1], k = 5)$pred
# </code r> ============================================================== #
```

```
  [1]  40.216000 -55.630400   9.459480 -26.780400  40.124800 -25.987400
  [7]  51.570000  69.595400 -46.860600 -55.244000 -21.676600 -35.421000
 [13] -63.752800  19.355800 -23.430000 -43.976600 -57.588400 -62.818800
 [19]  24.252200  16.298200  39.245200  68.085200  69.033600  32.896000
 [25] -47.912600 -42.637000  -1.590720  -1.366120  68.360600  -4.031388
 [31] -52.349200 -45.595000  63.367200  38.112000   4.744300 -42.796000
 [37] -68.313400  16.426400 -48.117200  60.517200 -69.483600  -7.989568
 [43] -43.888000   2.595472  43.399800 -43.728000 -52.848600 -15.506600
 [49] -34.000200  47.782200 -12.222280 -49.676800 -18.575800   7.388460
 [55]  37.182600  43.800800 -61.060600  49.560600   5.751260  16.426400
 [61]  62.915200 -55.244000 -17.795800 -15.453740 -47.257200   8.657500
 [67]  43.835000 -25.085800 -26.408800   7.825360 -47.776800   7.769960
 [73] -54.263600  40.641400   4.717560  26.608600  66.645400  32.804200
 [79]  10.616260 -20.130000  61.746600 -15.391540 -15.395940   6.475200
 [85]  61.714800  15.525600 -69.684800  32.583600 -39.760800 -33.132000
 [91] -67.595200 -20.069400 -23.994600 -14.607840  32.536400 -40.402400
 [97]  47.840600 -43.965000 -43.965000 -23.642000 -56.255600  37.877000
[103] -48.358800  64.943400 -52.750800 -11.205040 -19.598940  24.640800
[109]  25.518600   1.613820  56.504600 -67.595200  37.346600  62.946600
[115] -17.113800  61.067400  15.735000  40.875200 -67.964200 -18.000760
[121] -47.485000 -13.582240  -4.881200  -0.695040  -0.375960 -20.104000
[127]  51.564400 -12.022120  -4.653000 -34.831800  26.630800 -12.138300
[133]  16.057000  48.789600 -37.076800 -46.214400  61.248800  31.045600
[139] -13.399960
```

d) Utilize validação cruzada (*data splitting*) para escolher o melhor k. Plote k vs Risco estimado.

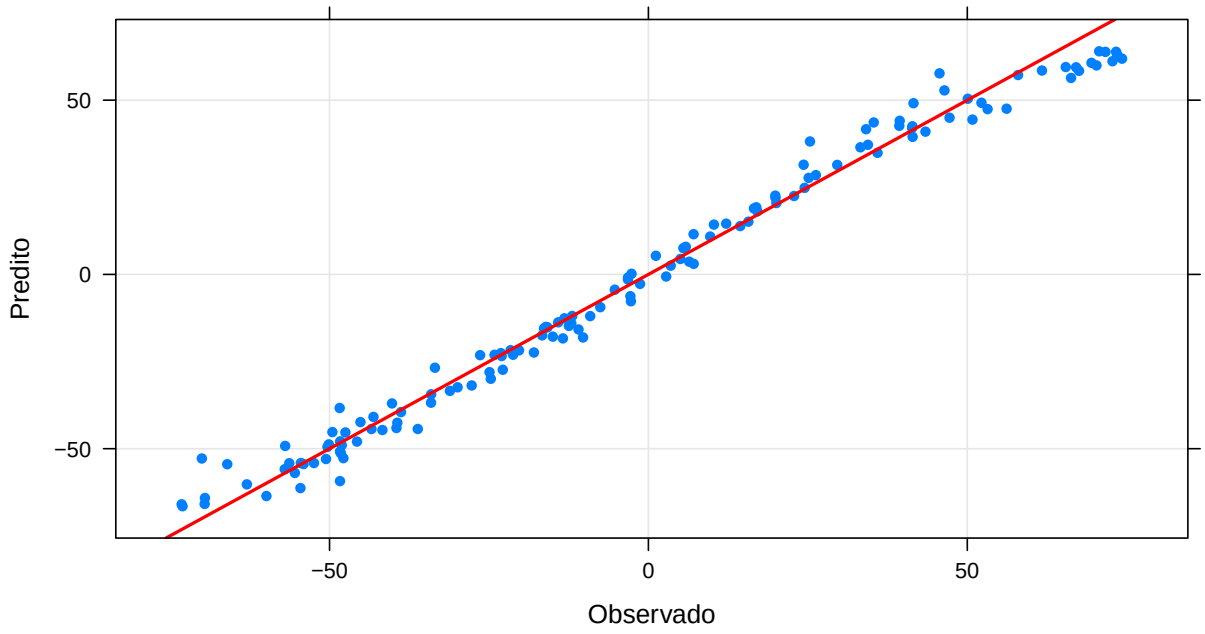
```r
# <code r> ================================================================== #
k <- c(1:20)

mse <- 0

for (i in 1:20){

  y_hat <-
    knn.reg(data2_train[ , -1], data2_val[ , -1], data2_train[ , 1], k = i)

  mse[i] <- sum(data2_val[, 1] - y_hat$pred) ** 2}

xyplot(mse ~ k
       , type = c("l", "g")
       , ylab = "Risco estimado"
       , lwd = 2
       , panel = function(...){
         panel.xyplot(...)
         panel.abline(v = which.min(mse), col = 2, lwd = 2)
         panel.text(14, 3500, labels = paste("k =", which.min(mse)), col = 2)})
# </code r> ================================================================== #
```



e) Utilize o conjunto de teste, estime o risco do KNN para o melhor k. Plote os valores preditos vs os valores observados para o conjunto de teste. Inclua a reta identidade.

```r
# <code r> ================================================================== #
model <-
  knn.reg(data2_train[ , -1], data2_test[ , -1], data2_train[ , 1], k = 12)

mean( (model$pred - data2_test[ , 1]) ** 2 )

xyplot(model$pred ~ data2_test[ , 1]
       , type = c("p", "g")
       , pch = 16
       , xlab = "Observado"
       , ylab = "Predito"
       , panel = function(...){
         panel.xyplot(...)
         panel.abline(0, 1, col = 2, lwd = 2)})
# </code r> ================================================================== #
```

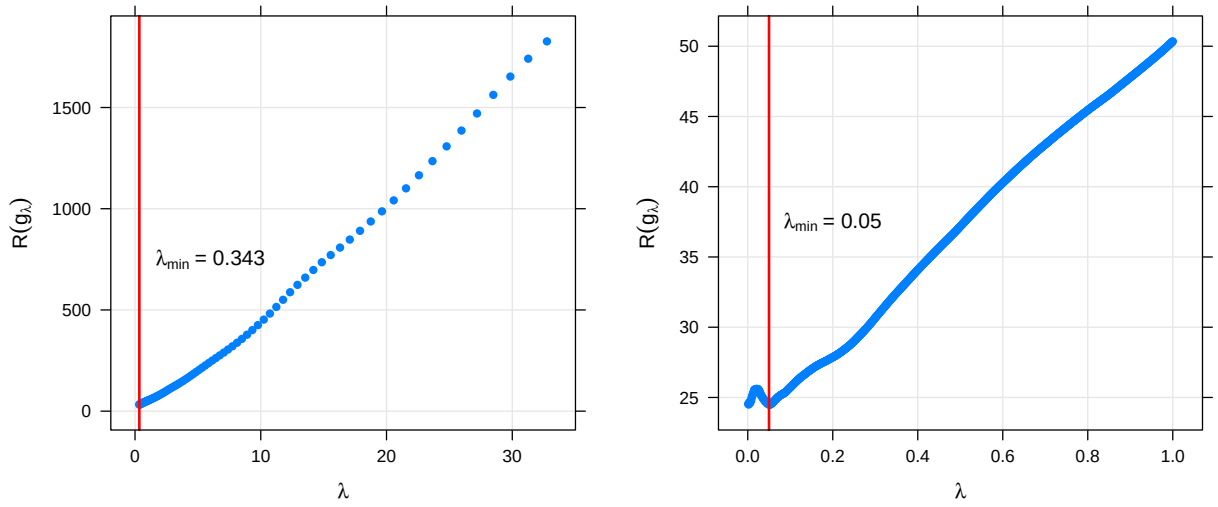
```
[1] 26.21408
```



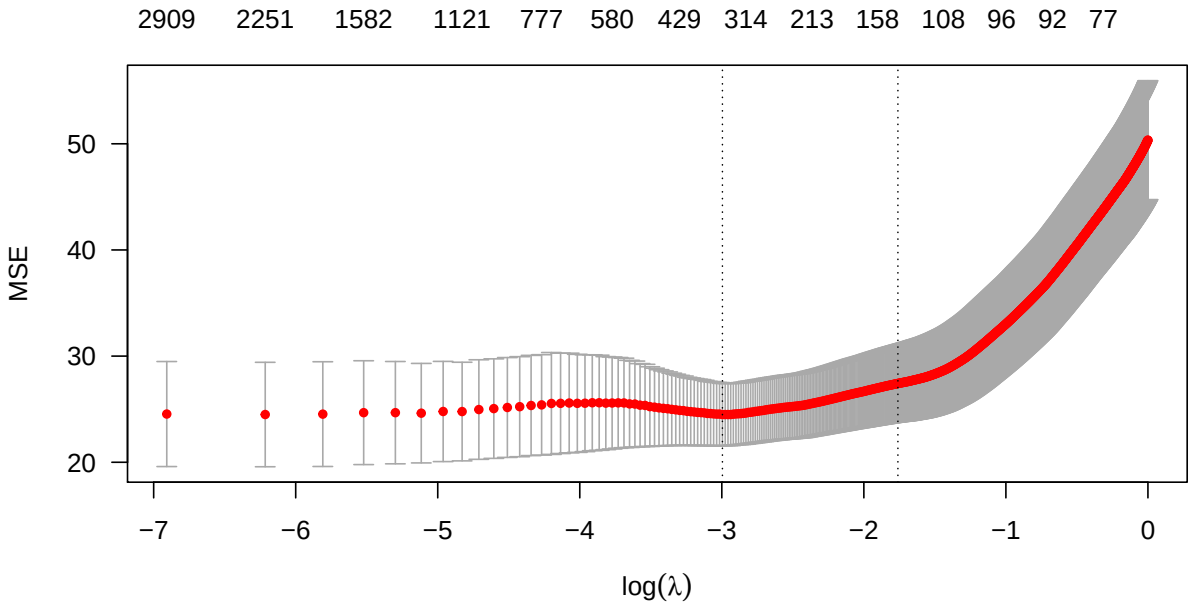
f) Ajuste uma regressão linear para os dados usando o conjunto de treinamento mais o de validação via lasso (lembre-se que a função que ajusta o lasso no R já faz validação cruzada automaticamente: ao contrário do KNN, neste caso não é necessário separar os dados em treinamento e validação). Qual o lambda escolhido? Plote lambda vs Risco estimado.

```r
# <code r> ================================================================== #
data2_trainval <- data2[1:559, ]
```

```r
set.seed(93)

lasso <- glmnet(
  as.matrix(data2_trainval[ , -1]), as.matrix(data2_trainval[ , 1])
  , alpha = 1)

cv <- cv.glmnet(
  as.matrix(data2_trainval[ , -1]), as.matrix(data2_trainval[ , 1])
  , alpha = 1)

print(
  xyplot(cv$cvm ~ cv$lambda
         , xlab = expression(lambda)
         , ylab = expression(R(g[lambda]))
         , type = c("p", "g")
         , pch = 16
         , panel = function(...){
           panel.xyplot(...)
           panel.abline(v = cv$lambda.min
                        , col = 2
                        , lwd = 2)
           panel.text(6, 750
                      , labels = expression(lambda[min]~"= 0.343"))})
  , position = c(0, 0, .5, 1)
  , more = TRUE)

cv <- cv.glmnet(
  as.matrix(data2_trainval[ , -1]),  as.matrix(data2_trainval[ , 1])
  , alpha = 1
  , lambda = seq(1, .001, length.out = 1000))

print(
  xyplot(cv$cvm ~ cv$lambda
         , xlab = expression(lambda)
         , ylab = expression(R(g[lambda]))
         , type = c("p", "g")
         , pch = 16
         , panel = function(...){
           panel.xyplot(...)
           panel.abline(v = cv$lambda.min
                        , col = 2
                        , lwd = 2)
           panel.text(.2, 37.5
                      , labels = expression(lambda[min]~"= 0.05"))})
  , position = c(.5, 0, 1, 1))
# </code r> ========================================================================== #
```




```
# <code r> ============================================================================ #
plot(cv, las = 1, xlab = expression(log(lambda)), ylab = "MSE")
# </code r> ============================================================================ #
```

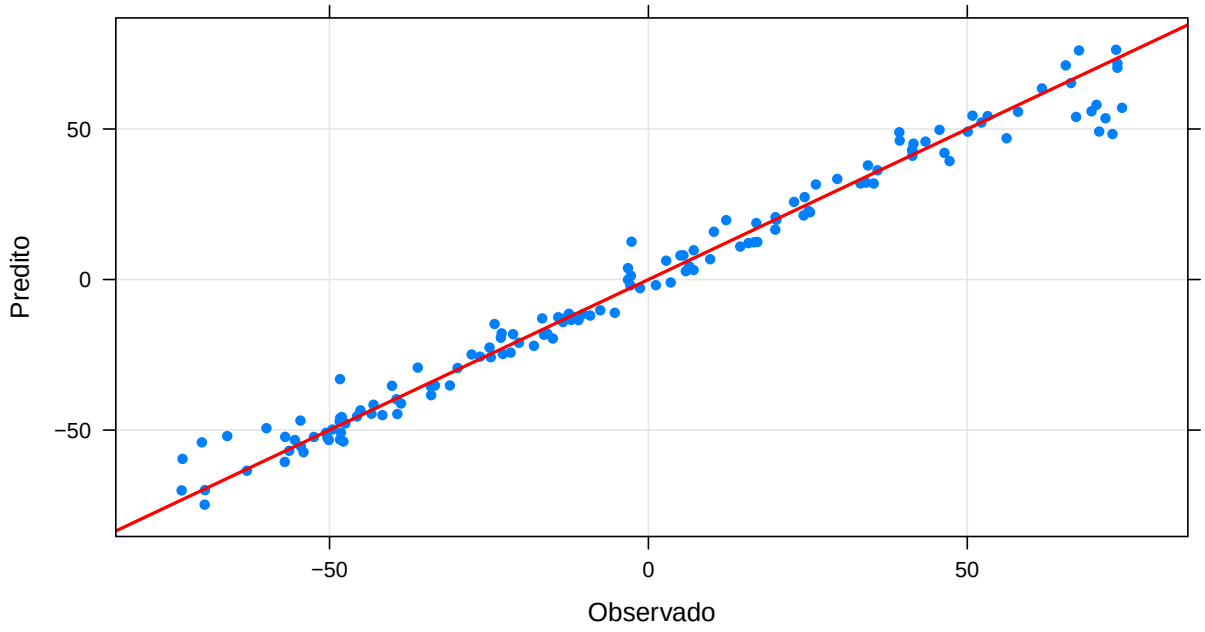


g) Utilizando o conjunto de teste, estime o risco do lasso para o melhor lambda. Plote os valores preditos vs os valores observados para o conjunto de teste. Inclua a reta identidade.

```
# <code r> ============================================================================ #
pred <- predict(lasso, s = cv$lambda.min, newx = as.matrix(data2_test[ , -1]))
```

```r
mean( (pred - data2_test[ , 1]) ** 2 )

xyplot(pred ~ data2_test[ , 1]
       , type = c("p", "g")
       , pch = 16
       , xlab = "Observado"
       , ylab = "Predito"
       , panel = function(...){
         panel.xyplot(...)
         panel.abline(0, 1, col = 2, lwd = 2)})
# </code r> ========================================================== #
```

```
[1] 36.31676
```



h) Quantos coeficientes foram estimados como sendo zero?

```r
# <code r> ========================================================== #
coef <- coefficients(lasso, s = cv$lambda.min)

count <- 0

for (i in 1:ncol(data2)){

  if (coef[i] == 0) count <- count + 1}

count
# </code r> ========================================================== #
```

```
[1] 3985
```

```
# <code r> ========================================================================== #
ncol(data2) - count
# </code r> ========================================================================== #
```

```
[1] 112
```

```
# <code r> ========================================================================== #
length(coef@x)
# </code r> ========================================================================== #
```

```
[1] 112
```

i) Qual modelo teve melhores resultados: regressão linear via lasso ou KNN?

Ambos apresentaram resultados muito similares, com uma ligeira vantagem para o KNN de $k = 12$, pois apresenta um risco estimado menor. A análise dos gráficos dos valores preditos vs valores ajustados evidencia um erro menor nos valores extremos para o KNN, o que pode justificar a vantagem deste modelo.