Relatorio

Maxxi Lorenzzo Santos Rios, 472722 Alysson da Silva Moura, 400660
Leonardo Gomes Prado, 472920 Luis Fernando A. Brito, 418824

05/02/2022

Incluindo os pacotes

Primeiro nós precisamos incluir o pacote `glmnet` que utilizaremos para estimar utilizando os métodos de *shrinkage*

Questões

Considere os dados sobre automóveis disponíveis no Sigaa UFC em formato csv intitulado "house_data_exer_tnkc_wobasement.

Primeiro vamos carregar e guardar os nossos dados na variável `dados`:

```
dados <- read.csv("house_data_exer_tnkc_wobasement.csv")

head(dados)
```

```
##   ï..price bedrooms bathrooms sqft_living sqft_lot floors waterfront view
## 1   364000        4      1.75        2010     8625      1          0    0
## 2   875000        5      3.25        4230    21455      2          0    0
## 3  1406890        5      2.25        3580    16789      2          0    0
## 4   550000        3      1.75        1410     5000      1          0    0
## 5  1240000        5      4.00        4410    14380      2          0    0
## 6   326100        2      1.00         880     7683      1          0    0
##   condition grade sqft_above yr_built
## 1         4     7       1340     1957
## 2         3    10       2720     1990
## 3         5     9       3580     1966
## 4         4     7        810     1923
## 5         3    11       4410     2006
## 6         3     6        880     1942
```

Para facilitar nossa vida, vamos renomear a primeira coluna, para `price`

```
names(dados)[1] <- "price"
```

Pra garantir que o nosso *number generator* seja igual ao da questão, vamos utilizar o comando `base::set.seed()`

```
set.seed(123)
```

Agora vamos separar nossa base em *treino* e *teste*

```
## separamos os dados entre treino e teste para evitar overfitting
index <- sample(nrow(dados),nrow(dados)*0.80)
dados.train <- dados[index,]
```

```r
dados.test <- dados[-index,]


index_y <- which(colnames(dados)=="price")

#Como o pacote `gmlnet` não aceita dataframes, precisamos converter
#os dados para matrix
X.train <- as.matrix(dados.train[,-index_y])
Y.train <- as.matrix(dados.train[,index_y])

X.test <- as.matrix(dados.test[,-index_y])
Y.test <- as.matrix(dados.test[,index_y])
```

Primeira Questão

Estime modelo de previsão usando MQO

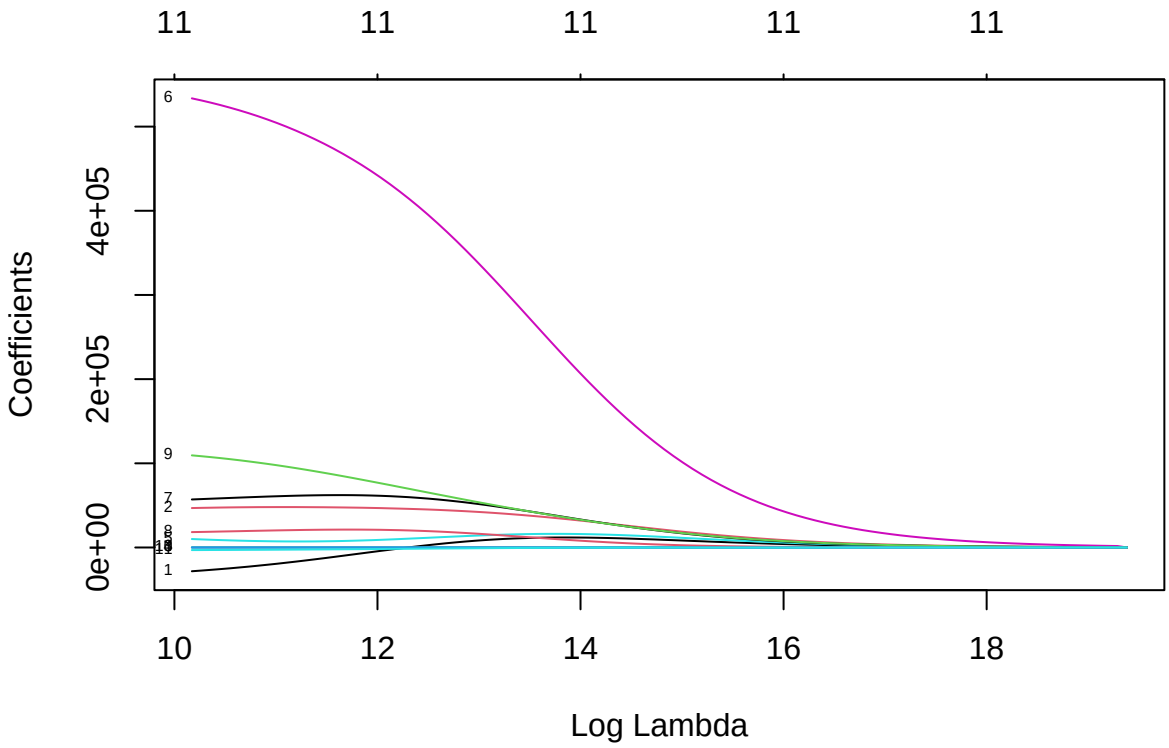
Para estimar o modelo de Mínimo Quadrados Ordinários (MQO), precisamos utilizar a função `lm`

Segunda Questão

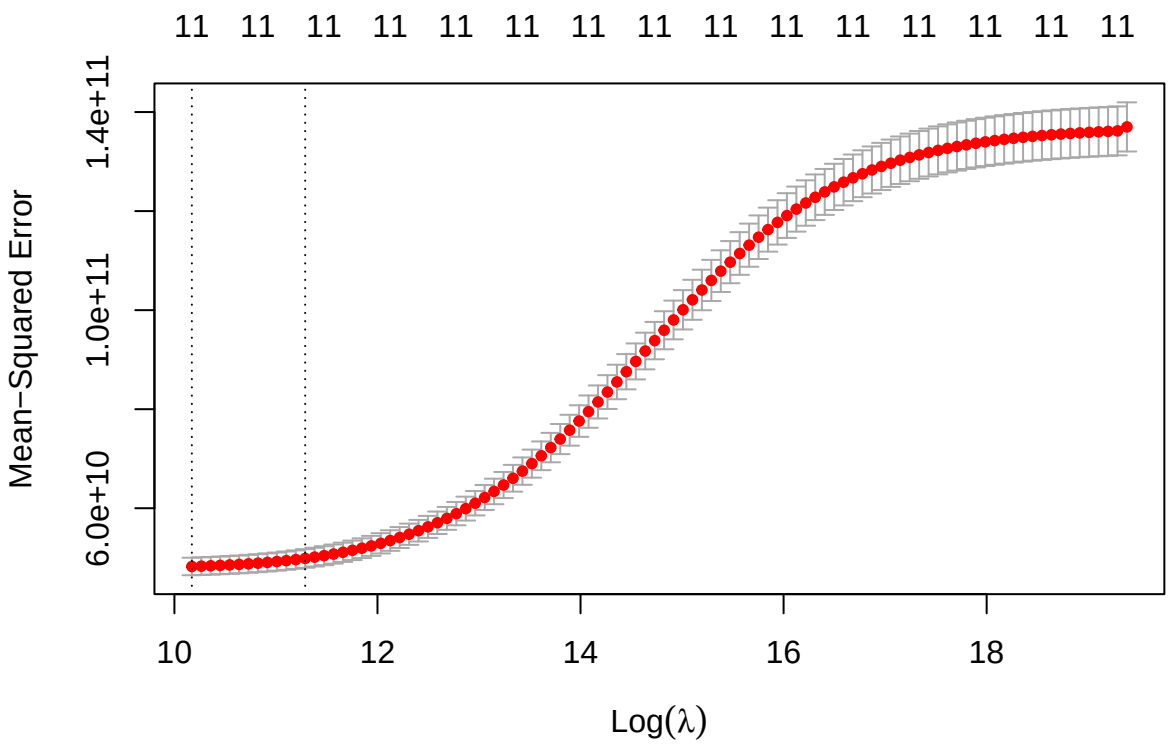
Estime modelo de previsão usando Ridge

Para o modelo Ridge temos que minimizar a seguinte função: β

```r
ridge.fit <- glmnet(x= X.train, y=Y.train, family="gaussian", alpha=0)
plot(ridge.fit, xvar="lambda", label=TRUE)
```




```
cv.ridge <-  cv.glmnet(x=X.train, y=Y.train, family="gaussian", alpha=0, nfolds=10)
plot(cv.ridge)
```




```
coef(cv.ridge, s=cv.ridge$lambda.min)
```

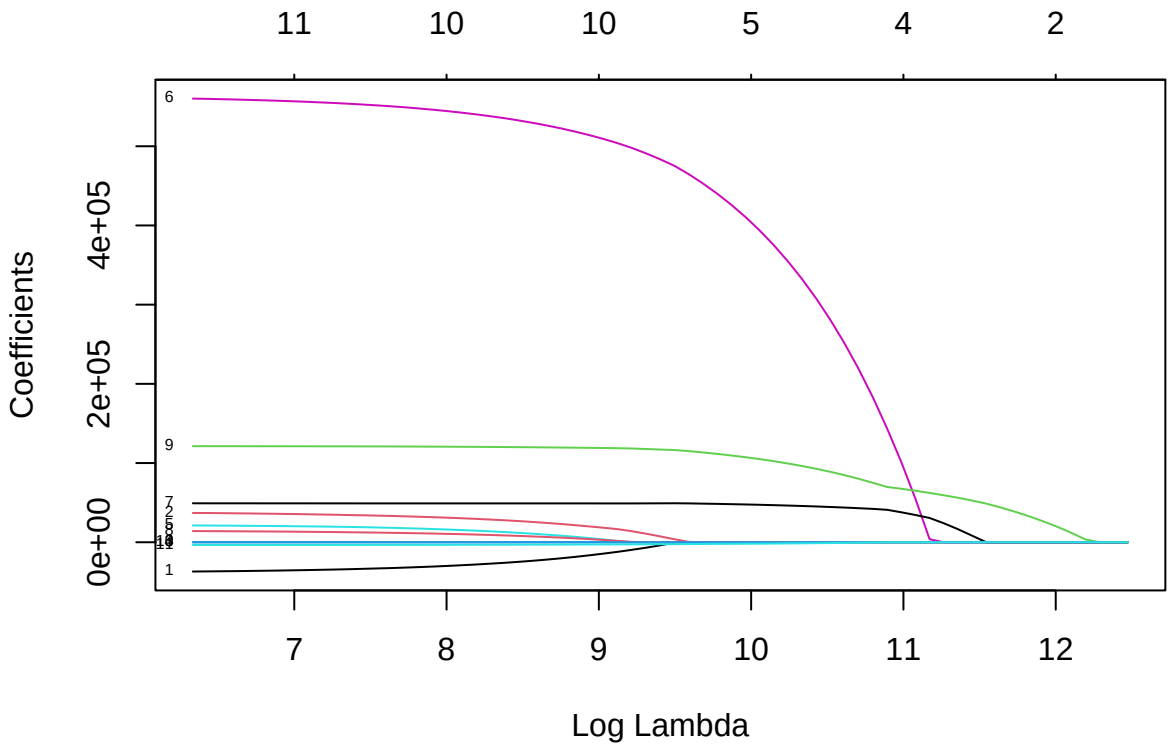
```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)  5.280961e+06
## bedrooms    -2.820557e+04
## bathrooms    4.685924e+04
## sqft_living  1.354207e+02
## sqft_lot    -2.332355e-01
## floors       1.003572e+04
## waterfront   5.335569e+05
## view         5.707396e+04
## condition    1.827411e+04
## grade        1.094731e+05
## sqft_above   3.983643e+01
## yr_built    -3.057854e+03
```

```
Y_pred.ridge <- predict(ridge.fit, newx = X.test, s=cv.ridge$lambda.min)

MSE_ridge=mean((Y.test-Y_pred.ridge)^2)
```

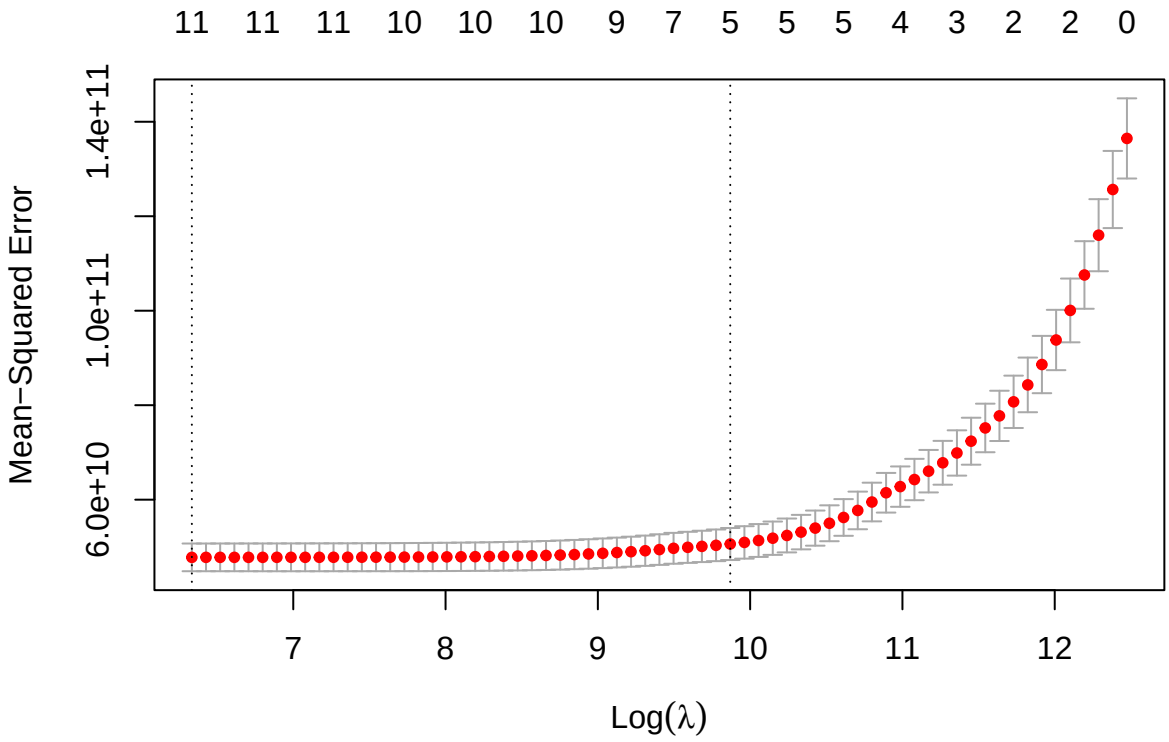
Terceira Questão

Estime modelo de previsão usando Lasso

```
lasso.fit <- glmnet(x=X.train, y=Y.train, family="gaussian", alpha=1)

plot(lasso.fit,xvar="lambda", label=TRUE)
```




```
cv.lasso <- cv.glmnet(x=X.train, y=Y.train,family="gaussian", alpha=1, nfolds=10)

plot(cv.lasso)
```




```
coef(cv.lasso, s=cv.lasso$lambda.min)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)  5.899854e+06
## bedrooms    -3.694220e+04
## bathrooms    3.701211e+04
## sqft_living  1.771408e+02
## sqft_lot    -2.491870e-01
## floors       2.121299e+04
## waterfront   5.601319e+05
## view         4.920484e+04
## condition    1.413335e+04
## grade        1.213249e+05
## sqft_above   1.598994e+00
## yr_built    -3.402046e+03
```

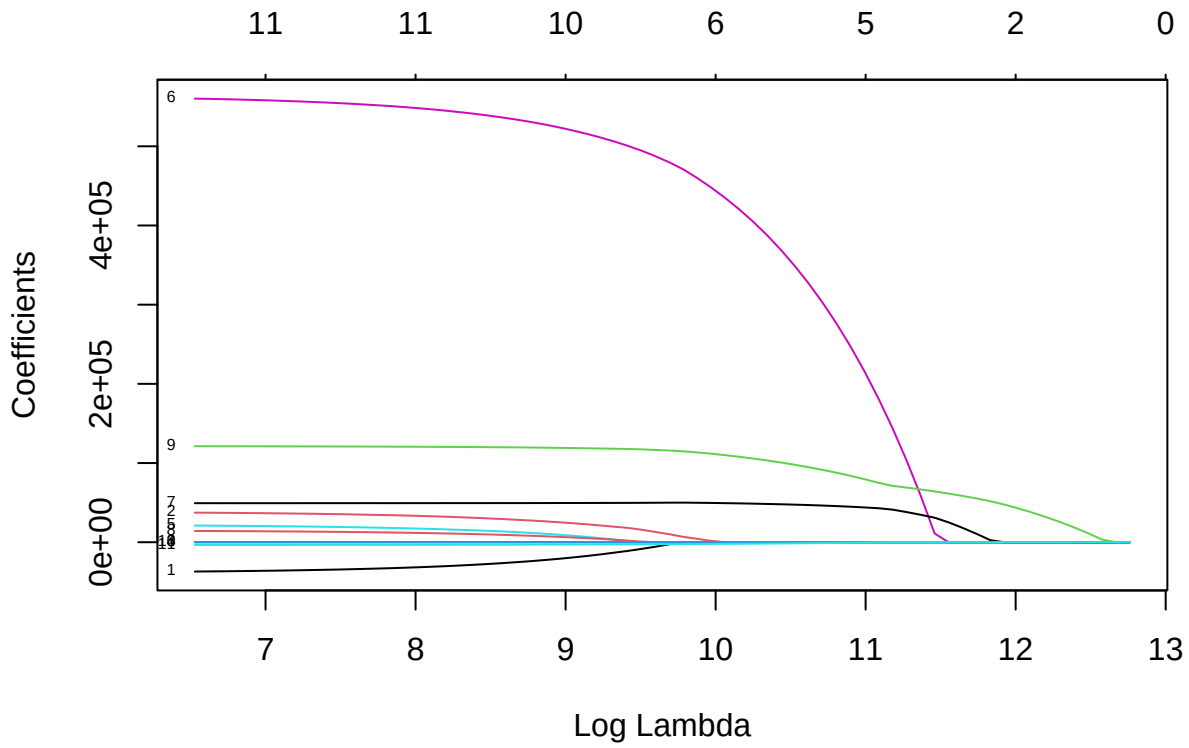
```
Y_pred.lasso <- predict(lasso.fit, newx = X.test, s=cv.lasso$lambda.min)

MSE_lasso <- mean((Y.test-Y_pred.lasso)^2)
```

Qual o melhor modelo? Justifique.

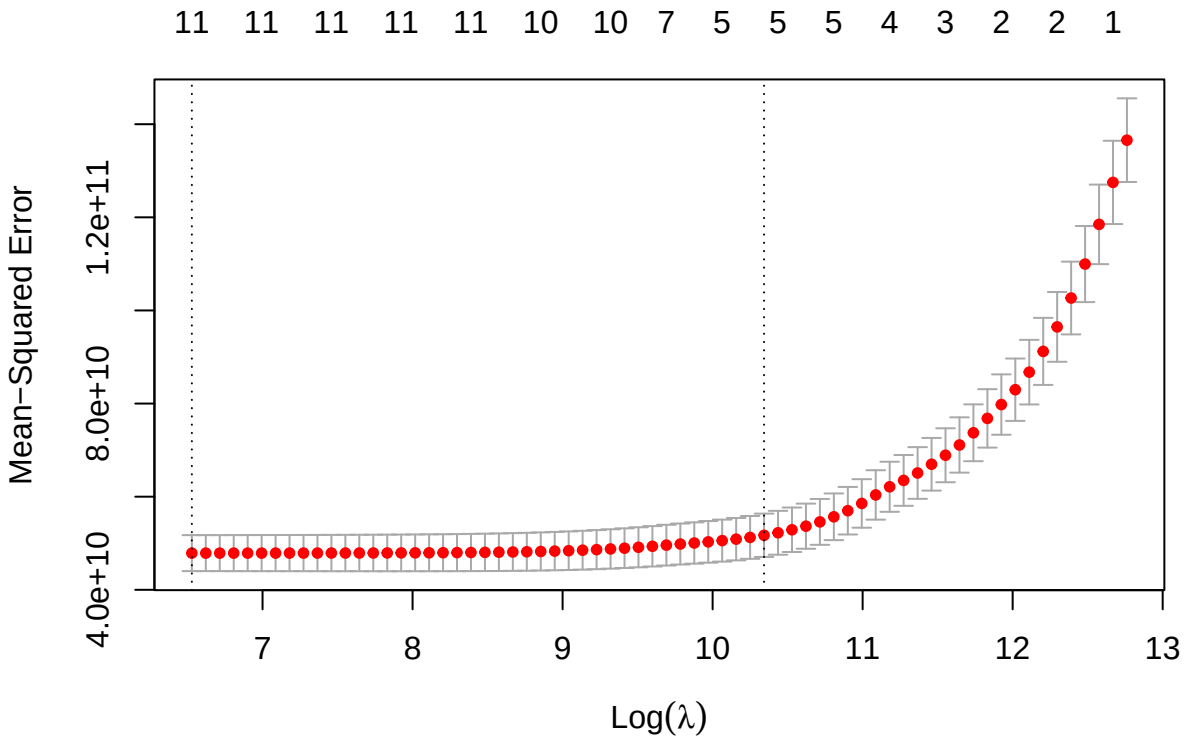
Questão Extra

Ponto extra: Estime modelo de previsão usando Elastic-net (com alpha=$\{$ 0.25;0.5;0.75$\}$). Compare com modelos anteriores.

```
elastic.fit <- glmnet(x=X.train, y=Y.train,family="gaussian",alpha=0.75)

plot(elastic.fit,xvar="lambda", label=TRUE)
```




```
cv.elastic <- cv.glmnet(x=X.train, y=Y.train, family="gaussian", alpha=0.75, nfolds=10)

plot(cv.elastic)
```




```
coef(cv.elastic, s=cv.elastic$lambda.min)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)  5.904087e+06
## bedrooms    -3.698330e+04
## bathrooms    3.734970e+04
## sqft_living  1.764818e+02
## sqft_lot    -2.507869e-01
## floors       2.106140e+04
## waterfront   5.601952e+05
## view         4.931320e+04
## condition    1.423917e+04
## grade        1.212345e+05
## sqft_above   2.303345e+00
## yr_built    -3.404147e+03
```

```
Y_pred.elastic<- predict(elastic.fit, newx = X.test, s=cv.elastic$lambda.min)

MSE_elastic=mean((Y.test-Y_pred.elastic)^2)

cbind(MSE_lasso,MSE_elastic)
```

```
##        MSE_lasso MSE_elastic
## [1,] 53078876009 53075757947
```