Selección de Modelos y Regularización, y Random Forests.

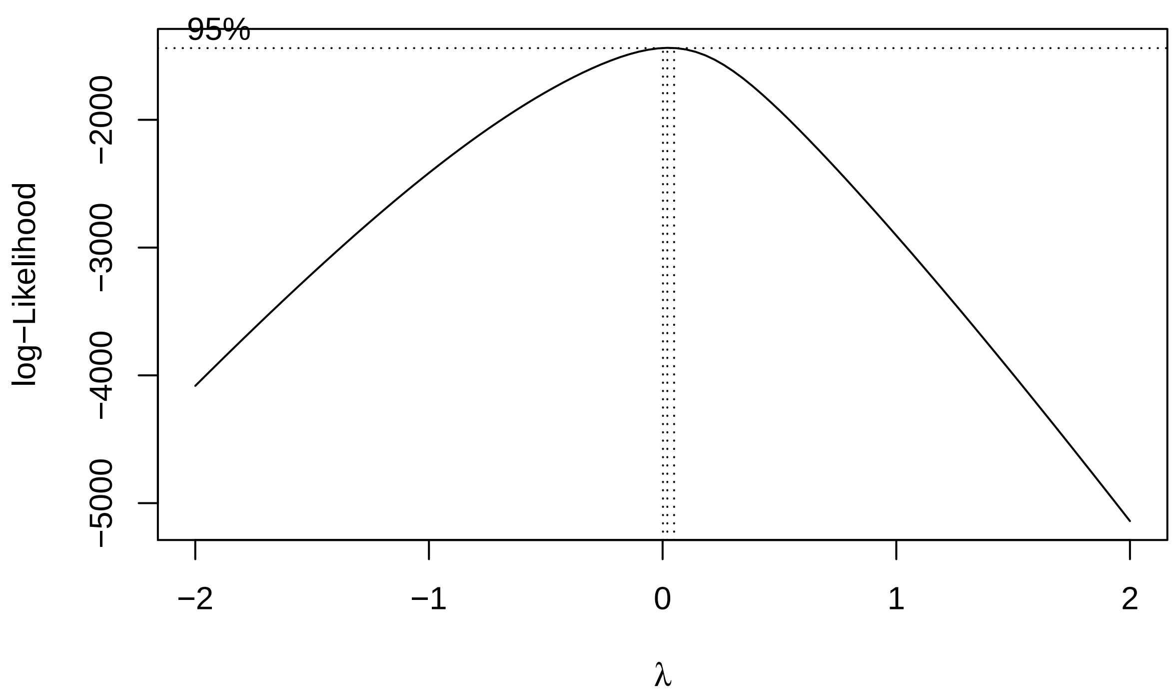
González Feria Juan Rosendo
González González Elvira
Nicolas Mata Jesús
Pacheco Martínez Mariana
Miranda Peñafiel Melissa Sofía

8 de mayo de 2020

1. Basado en el ejercicio 11 de ISL, p 264. Considere la base de datos `Boston` del paquete `MASS`. Considere modelos de regresión con la tasa de crimen per capita, crim como variable respuesta y las 13 restantes como variables predictoras. Realice lo siguiente.

1. Auxiliandose de la función `boxcox` inspeccione la posibilidad de aplicar una transformación a la variable respuesta. Una vez determinada la transformación a la variable respuesta, manténgala en todos los modelos.

Se tiene la siguiente gráfica para el coeficiente de la transformación:



Podemos ver que es bastante cercano a 0, el valor de λ real es:

```
## [1] 0.02020202
```

Dado que el valor de λ es cercano a 0 por definición de la función *boxcox* entonces vamos a aplicar la tranformación *logaritmo*.

2. Ajuste un modelo de regresión lineal multiple con efectos aditivos.

Se tiene para el modelo con todas la variables el siguiente `summary`

```
##
## Call:
## lm(formula = log(crim) ~ ., data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.56974 -0.55469 -0.03703  0.50211  2.62719
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.7328735  0.8687638  -4.297 2.09e-05 ***
## zn          -0.0116849  0.0022496  -5.194 3.02e-07 ***
## indus        0.0197559  0.0100155   1.973 0.049109 *
## chas        -0.0480924  0.1417115  -0.339 0.734477
## nox          3.8468127  0.6334840   6.072 2.52e-09 ***
## rm          -0.0489851  0.0735884  -0.666 0.505938
## age          0.0059883  0.0021524   2.782 0.005608 **
## dis         -0.0054376  0.0338405  -0.161 0.872409
## rad          0.1428556  0.0105729  13.511  < 2e-16 ***
## tax         -0.0001321  0.0006191  -0.213 0.831098
## ptratio     -0.0411372  0.0223889  -1.837 0.066755 .
## black       -0.0015015  0.0004411  -3.404 0.000718 ***
## lstat        0.0317048  0.0090930   3.487 0.000533 ***
## medv         0.0104694  0.0072667   1.441 0.150299
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7732 on 492 degrees of freedom
## Multiple R-squared:  0.8754, Adjusted R-squared:  0.8721
## F-statistic: 265.9 on 13 and 492 DF,  p-value: < 2.2e-16
```

Podemos ver que las variables `chas, rm, dis, tax` y `medv` **NO** son estadísticamente significativas, $\alpha = 0.05$, mientras que `ptratio` tampoco lo es. El resto si lo es.

3. Seleccione tres modelos, utilizando cada uno de los tres métodos: `regsubsets(..., method= 'exhaustive'), step(..., k=2), step(...,k=log(n))`

El primer modelo corresponde a uno que optimiza el valor de *AIC* mientras que el segundo modelo optimiza el valor de *BIC*

```
##
## Call:
## lm(formula = log(crim) ~ zn + indus + nox + age + rad + ptratio +
##     black + lstat + medv, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.56802 -0.56142 -0.03622  0.50036  2.68223
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -4.1008282  0.6426669  -6.381 4.04e-10 ***
## zn          -0.0120563  0.0019633  -6.141 1.68e-09 ***
## indus        0.0196185  0.0086757   2.261 0.024172 *  
## nox          3.8666940  0.5967848   6.479 2.23e-10 ***
## age          0.0057751  0.0020158   2.865 0.004348 ** 
## rad          0.1405972  0.0060069  23.406  < 2e-16 ***
## ptratio     -0.0405793  0.0222739  -1.822 0.069082 .  
## black       -0.0014633  0.0004333  -3.377 0.000791 ***
## lstat        0.0336943  0.0085958   3.920 0.000101 ***
## medv         0.0088723  0.0061687   1.438 0.150986    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.7706 on 496 degrees of freedom
## Multiple R-squared:  0.8752, Adjusted R-squared:  0.873 
## F-statistic: 386.6 on 9 and 496 DF,  p-value: < 2.2e-16
```

Este modelo tomó 9 variables de las 13, con dos variables no estadisticamente significativas, `medv` y `ptratio` considerando un nivel de signicancia de $\alpha = 5\%$

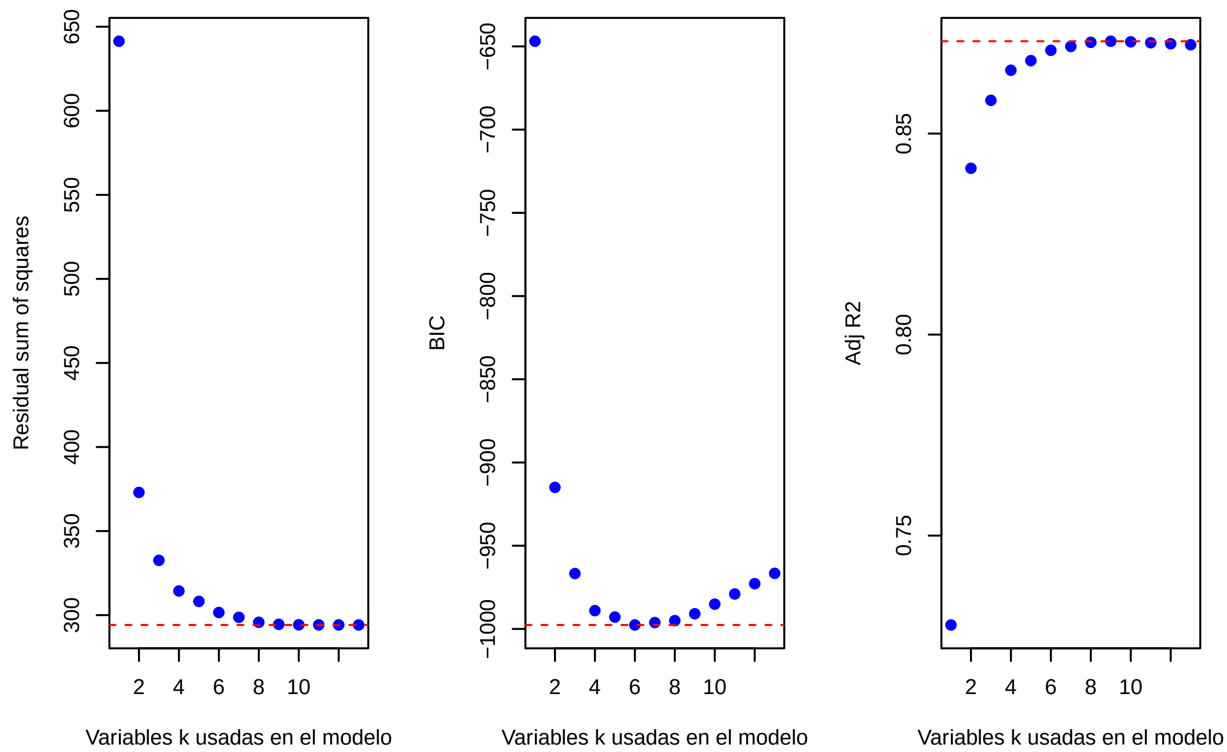
```
## 
## Call:
## lm(formula = log(crim) ~ zn + nox + age + rad + black + lstat, 
##     data = Boston)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -2.68072 -0.59355 -0.02178  0.49717  2.63190 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -4.8332339  0.3022006 -15.993  < 2e-16 ***
## zn          -0.0111057  0.0018389  -6.039 3.03e-09 ***
## nox          4.7142977  0.5078224   9.283  < 2e-16 ***
## age          0.0065785  0.0019892   3.307 0.001011 ** 
## rad          0.1377557  0.0053122  25.932  < 2e-16 ***
## black       -0.0014556  0.0004333  -3.359 0.000841 ***
## lstat        0.0250203  0.0065418   3.825 0.000148 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.7774 on 499 degrees of freedom
## Multiple R-squared:  0.8723, Adjusted R-squared:  0.8707 
## F-statistic: 567.9 on 6 and 499 DF,  p-value: < 2.2e-16
```

Todas las variables son estadisticamente significativas.

Para regsubset(. . . , method="exhaustive") se tiene que :

```
## Subset selection object
## Call: regsubsets.formula(log(crim) ~ ., data = Boston, nvmax = 13)
## 13 Variables  (and intercept)
##         Forced in Forced out
## zn          FALSE      FALSE
```

```
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## black       FALSE      FALSE
## lstat       FALSE      FALSE
## medv        FALSE      FALSE
## 1 subsets of each size up to 13
## Selection Algorithm: exhaustive
##           zn  indus chas nox rm  age dis rad tax ptratio black lstat medv
## 1  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     " "   " "   " " 
## 2  ( 1 )  " " " "   " "  "*" " " " " " " "*" " " " "     " "   " "   " " 
## 3  ( 1 )  "*" " "   " "  "*" " " " " " " "*" " " " "     " "   " "   " " 
## 4  ( 1 )  "*" " "   " "  "*" " " " " " " "*" " " " "     " "   "*"   " " 
## 5  ( 1 )  "*" " "   " "  "*" " " " " " " "*" " " " "     "*"   "*"   " " 
## 6  ( 1 )  "*" " "   " "  "*" " " "*" " " "*" " " " "     "*"   "*"   " " 
## 7  ( 1 )  "*" " "   " "  "*" " " "*" " " "*" " " "*"     "*"   "*"   " " 
## 8  ( 1 )  "*" "*"   " "  "*" " " "*" " " "*" " " "*"     "*"   "*"   " " 
## 9  ( 1 )  "*" "*"   " "  "*" " " "*" " " "*" " " "*"     "*"   "*"   "*" 
## 10  ( 1 ) "*" "*"   " "  "*" "*" "*" " " "*" " " "*"     "*"   "*"   "*" 
## 11  ( 1 ) "*" "*"   "*"  "*" "*" "*" " " "*" " " "*"     "*"   "*"   "*" 
## 12  ( 1 ) "*" "*"   "*"  "*" "*" "*" " " "*" "*" "*"     "*"   "*"   "*" 
## 13  ( 1 ) "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"   "*"
```



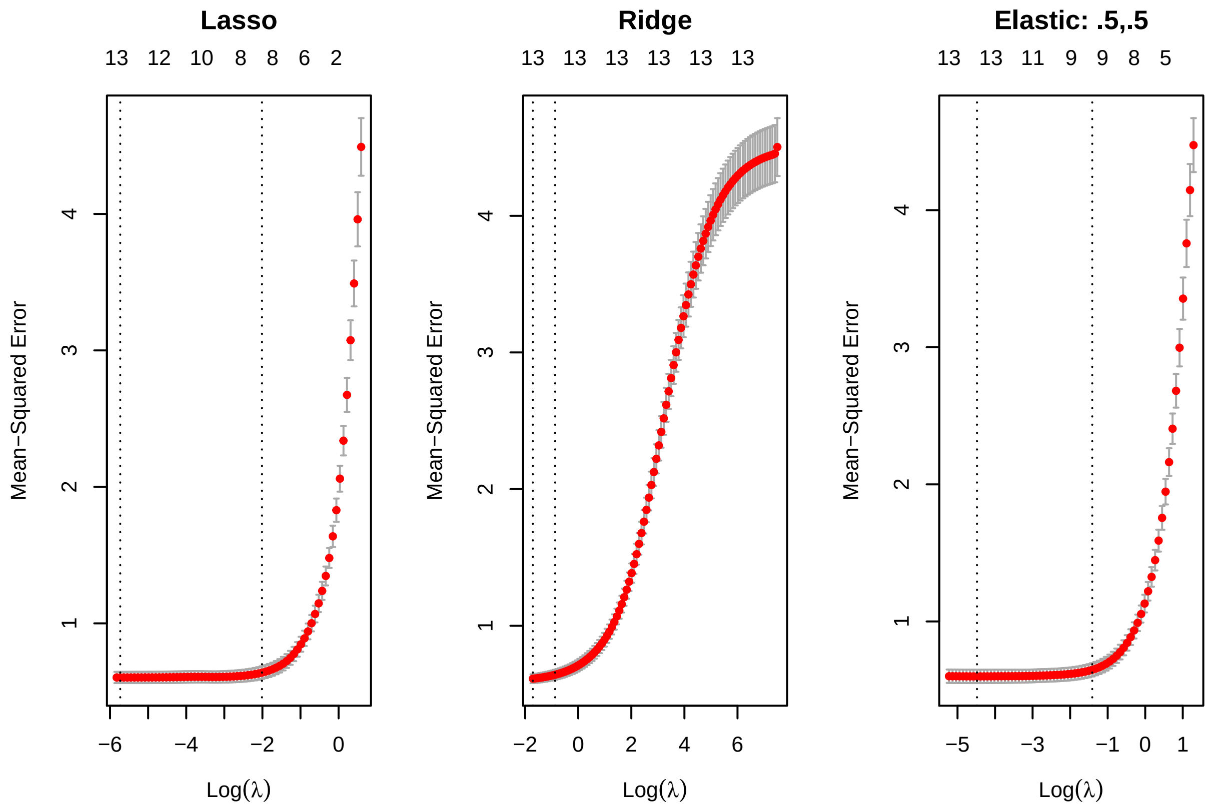
A partir de estos resultados eligimos el modelo con las variables `zn, indus, nox, age, rad, ptratio, black, lstat`.

La gráfica anterior muesta como se ve `regsubset(..., method="exhaustive")` respecto a la Suma al cuadrado de los residuales, BIC, R^2 ajustada respectivamente en función del número de variables del modelo.

Por lo tanto son 8 variables las que elegimos.

4. Seleccione tres modelos, con `lasso, ridge, elasticnet`, utilizando el paquete `glmnet`. Para el caso de elasticnet utilice `alpha=.5`.

Obteniendo las gráficas de los modelos se tiene :



A partir de estos resultados podemos ver que el número de coeficientes distintos de cero que se encuentran en nuestro modelo generado por *lasso* tomando en cuenta el parametro `lambda.min` son todas nuestras variables , lo mismo sucede para nuestros modelos *ridge y Elasticnet*.

Mientras que en el caso de usar como parametro `lambda.1se` consideramos para *lasso* 8 variables, 9 variables para *Elasticnet* y todas nuestras variables para *ridge*.

Debido a la poca cantidad de variables predictoras que tenemos en nuestro modelo decidimos usar el parametro `lambda.min` para obtener el MSE.

5. Para cada uno de los seis modelos seleccionados calcule la tasa de error de entrenamiento (*training error rate* o error aparente), así como la tasa de error de prueba (*test error rate*) también conocida como tasa de error de predicción validada o de prueba utilizando *Validation set i.e. Repeated training/test* o utilizando *cross-validation* con más de una repetición.

Nuestro conjunto de entrenamiento consiste en el 75% de todas nuestras observaciones , mientras que nuestro conjunto de prueba serán del 25% de las observaciones restantes.

El metodo usado fue *trainig/test* para la obtención de las tasas no aparentes con 500 repeticiones.

Nuestros errores resultantes fueron.

```
##                 Tr error (Ap) Tr/test 500 (NoAp)
## 1. lm               0.5813213          0.6220328
## 2. stepAIC          0.5820361          0.6098775
## 3. stepBIC          0.5959572          0.6136526
## 4. regsubsets       0.5844636          0.6090789
## 5. lasso            0.5828793          0.6194979
## 6. ridge            0.6036173          0.6345395
## 7. elastic_0.5      0.5819195          0.6198756
```

6. Unicamente para los dos modelos seleccionados por `step(...,k=2), step(...,k=log(n)`, calcule los errores de predicción validados de dos formas:

i) Incluya el proceso de selección `step(...,k=2), step(...,k=log(n)` dentro de cada repetición en el proceso de validación.

ii) Realice el proceso de selección `step(...,k=2), step(..., k=log(n))` una sola vez y déjelo fijo en cada repetición.

Para i) se tienen 500 modelos distintos para AIC y BIC que fueron variando con cada iteración y a partir de estos resultados se obtuvieron los MSE.

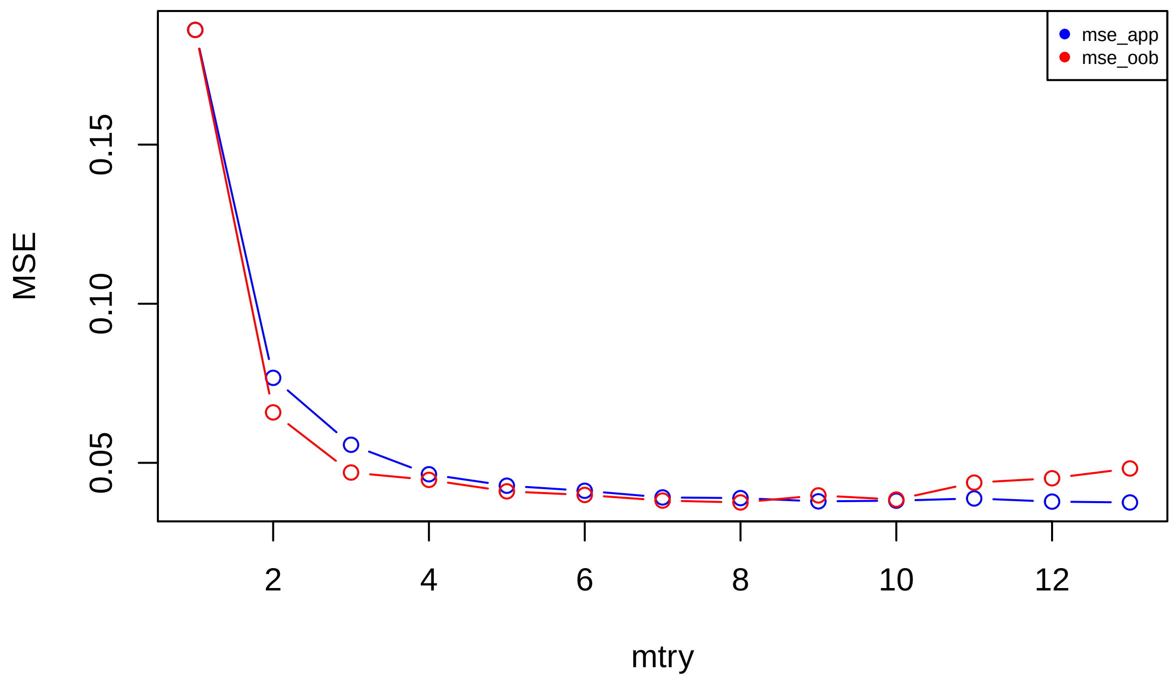
```
##   MSE AIC   MSE BIC
## 0.6195492 0.6351301
```

Para ii) se tienen 500 modelos fijos para AIC y BIC .

```
##   MSE AIC   MSE BIC
## 0.5909419 0.6014791
```

7. Utilizando el paquete `randomforest`, estime la tasa de error de predicción, en este caso el ECM, tanto la de entrenamiento con todo el conjunto de datos completo como la estimada de prueba (*training error and test error rate*). Esta última calculada por *Validation set i.e. Repeated training/test* o por *cross-validation* con más de una repetición.

Para `randomforest` se usaron 13 modelos donde cada uno difiere por el valor de *mtry*.



Se comprararon los MSE aparentes de cada uno de los modelos con los *MSE OOB* . Se escogió el modelo que corresponde al valor de $mtry = 7$.

Los errores correspondientes a este modelo son :

```
##      Tr error (Ap) Tr/test 500 (NoAp)
##         0.03913147         0.23549305
```

Resumiendo todo lo anterior tenemos las siguientes tablas, en ella se incluyen los parametros más importantes de cada uno de los modelos.

```
##                 Tr error (Ap) Tr/test 500 (NoAp) #Parms(df) lambda.min
## 1. lm                  0.5813             0.6220         14         NA
## 2. stepAIC             0.5820             0.6099         10         NA
## 3. stepBIC             0.5960             0.6137          7         NA
## 4. regsubsets          0.5845             0.6091          9         NA
## 5. lasso               0.5829             0.6195         12     0.0092
## 6. ridge               0.6036             0.6345         14     0.1843
## 7. elastic_0.5         0.5819             0.6199         13     0.0096
## 8. RandomForest        0.0391             0.2355         NA         NA
##                 lambda 1se mtry ntree
## 1. lm                   NA   NA    NA
## 2. stepAIC              NA   NA    NA
## 3. stepBIC              NA   NA    NA
## 4. regsubsets           NA   NA    NA
## 5. lasso            0.1495   NA    NA
## 6. ridge            0.5629   NA    NA
## 7. elastic_0.5      0.2061   NA    NA
## 8. RandomForest         NA    7   500
```

Para los modelos parametricos tenemos la siguiente tabla de coeficientes.

```
##                (Intercept)      zn  indus    chas    nox      rm    age     dis
## 1. lm              -3.7329 -0.0117 0.0198 -0.0481 3.8468 -0.0490 0.0060 -0.0054
## 2. stepAIC         -4.1008 -0.0121 0.0196      NA 3.8667      NA 0.0058      NA
## 3. stepBIC         -4.8332 -0.0111     NA      NA 4.7143      NA 0.0066      NA
## 4. regsubsets      -3.5954 -0.0120 0.0194      NA 3.7561      NA 0.0063      NA
## 5. lasso           -3.8382 -0.0115 0.0182 -0.0190 3.8513 -0.0317 0.0059 -0.0096
## 6. ridge           -4.0451 -0.0103 0.0144  0.0179 3.4574 -0.0201 0.0058 -0.0435
## 7. elastic_0.5     -3.8880 -0.0112 0.0182 -0.0043 3.8456 -0.0222 0.0058 -0.0132
##                   rad     tax ptratio   black  lstat   medv
## 1. lm          0.1429 -0.0001 -0.0411 -0.0015 0.0317 0.0105
## 2. stepAIC     0.1406      NA -0.0406 -0.0015 0.0337 0.0089
## 3. stepBIC     0.1378      NA      NA -0.0015 0.0250     NA
## 4. regsubsets  0.1420      NA -0.0528 -0.0014 0.0260     NA
## 5. lasso       0.1404      NA -0.0383 -0.0015 0.0305 0.0081
## 6. ridge       0.1025  0.0016 -0.0205 -0.0017 0.0297 0.0071
## 7. elastic_0.5 0.1393      NA -0.0354 -0.0014 0.0299 0.0067
```

Ejercicio 2:

Considere la información `(riboflavin$x,riboflavin$y)` disponible en `library(hdi)`. Este ejemplo corresponde a $n = 71$ observaciones en $p = 4088$ dimensiones. Realice lo siguiente.

1. Seleccione un modelo con `lasso, ridge, elasticnet` utilizando el paquete `glmnet`. Para el caso de `elasticnet` utilice `alpha=0.5`

2. Para cada uno de los tres modelos seleccionados calcule la tasa de error de entrenamiento (error aparente), así como la tasa de error validad utilizando *Repeated training/test* o por *cross-validation* con más de una repetición.

Recordemos que para un modelo elegido con ridge este mantendrá la cantidad de variables originales y sólo se buscará encontrar el valor de lambda que minimice el error cuadrático medio. Para lasso los modelos sí irán viendo una reducción en variables y se buscará la lambda que minimice el ECM como en el caso de ridge. Para elastic net se tendrá una idea análoga a lasso.

Ahora, para poder hacer la elección de modelos, al usar `cv.glmnet` se recurrió a k=5 folds debido al número limitado de observaciones con las que cuenta la base del problema. Otra consideración que se tuvo fue, al hacer repeated train/test, tomar el 80 % de los datos para el conjunto *train*.

A continuación se mostrarán las tasas de error de los modelos ajustados:

Tasas de error				
Modelo	Aparentes	No Aparentes	sd	Betas
Ridge	2.76	27.46	14.25	4089
Lasso	4.90	26.64	14.84	36
Elastic Net $\alpha = 0{,}5$	5.03	24.69	13.83	57

Cuadro 1: Tasas de error presentadas en porcentajes para los modelos seleccionados por *ridge, lasso* y *elastic net*. Tasas no aparentes fueron calculadas utilizando el método de *repeated train/test* con 500 iteraciones.

El criterio para escoger todos los modelos fue tomar `lambda.min`. Para *ridge* se utilizó porque era la que minimizaba el ECM, para *lasso* y *elastic net* se utilizó porque minimizaba el ECM y porque tomaba en cuenta más variables (β) que si se utilizaba `lambda.1se`. Buscamos que los últimos dos modelos tomaran en cuenta más variables ya que la base original cuenta con miles de ellas y a nivel de decenas no cambia mucho el costo computacional.

1. Anexo

A continuación se presenta la tabla con los valores obtenidos para `lambda.min` y `lambda.1se` con sus respectivos *df.max* que corresponde al número de variables seleccionadas para los modelos con esos lambdas.

Modelo	`lambda.min`	df.max(l.min)	`lambda.1se`	df.max(l.1se)
Ridge	5.9341	4088	27.5439	4088
Lasso	0.04813	35	0.1164	24
Elastic Net	0.0918	56	0.4071	20

Cuadro 2: Valores de *lambda.min* y *lambda,1se* con sus respectivos *df.max* para los modelos ajustados

Ahora mostramos las gráficas correspondientes a los modelos ajustados

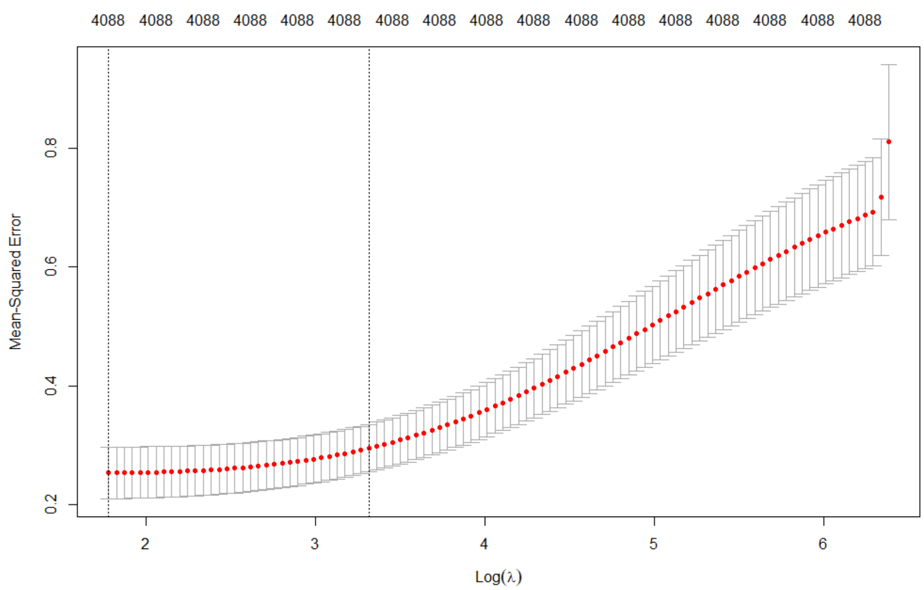


Figura 1: Ridge Model

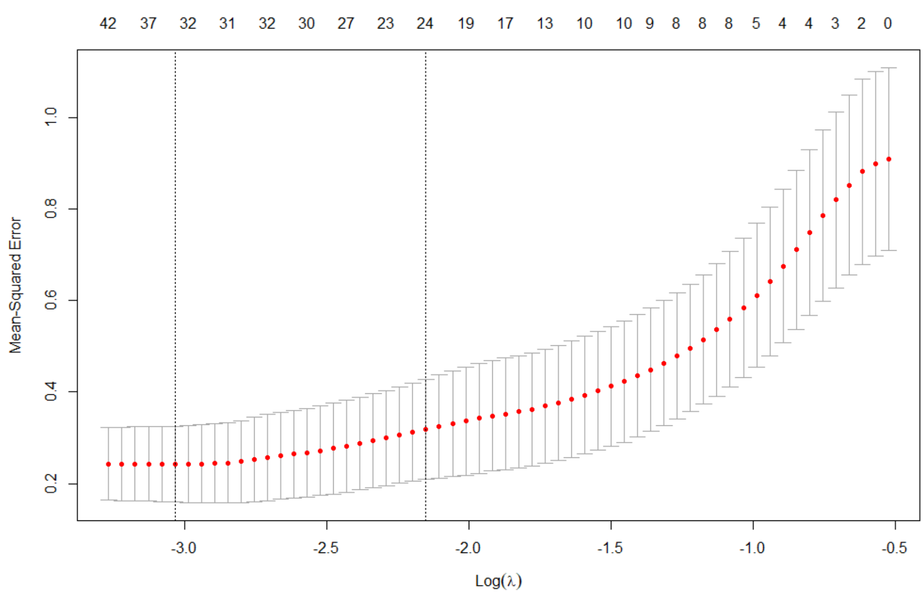


Figura 2: Lasso Model

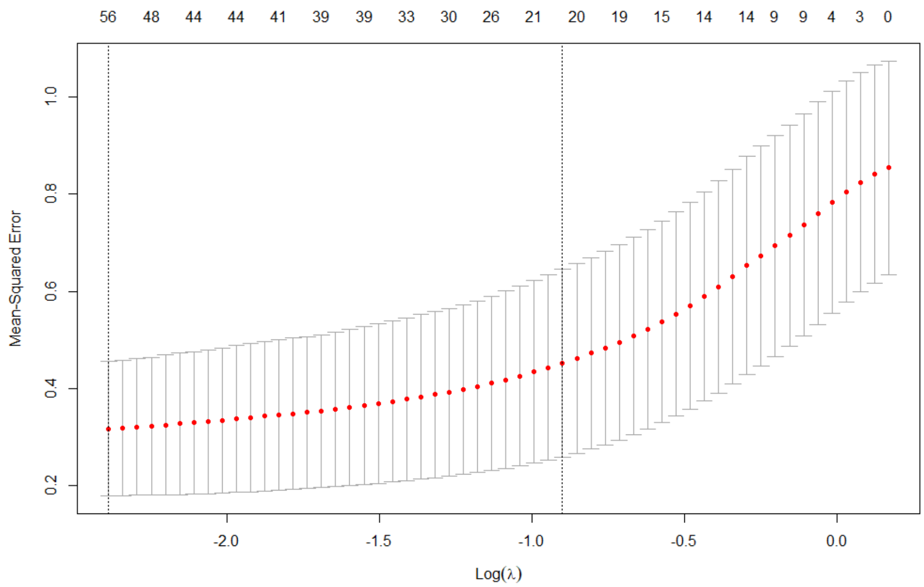


Figura 3: Elastic Net Model

Adicional a todo esto también obtuvimos otros datos interesantes de los modelos, como tasas de error con $Cross$ $Validation(k = 5, B = 1)$, el $Betas(df)$ que es el número final de variables seleccionadas por el modelo y el $Tiempo$ que tardó en correr cada modelo con 500 iteraciones.

Modelo	Error C.V(k=5,B=1)	Beta(df)	Tiempo(seg)	Tiempo(min)
Ridge	25.34	4089	650.47	10.84
Lasso	24.186	36	135.80	2.26
Elastic Net	31.71	57	155.87	2.6

Cuadro 3: Datos complementarios al Cuadro 2.

A excepción de *Elastic Net*, podemos ver que los errores dados por Cross Validation son menores a los obtenidos por train/test , sin embargo al tener una sola iteración, no es la mejor aproximación, por otro lado podemos ver que todas las $Beta(df)$ solo difieren en una unidad con las $df.max(l.min)$ del *Cuadro* 2, que fueron las variables máximas dadas por `lambda.min`, y esto se debe a que $Beta(df)$ también cuenta el intercepto, finalmente podemos ver que en cuestión de tiempo el *Ridge* tarda casi 5 veces más que los otros, y esto se debe a la cantidad de variables que entran al modelo, obviamente es más pesado computacionalmente procesar 4088 variables que solo 36 o 57.

Antes de obtener nuestros modelos finales, hicimos un primer intento con distintos conjuntos $Train/Test$ del 65 % de los datos para $Train$ y 35 % para $Test$; validamos las tasas para todos los modelos con $N = 100, N = 300$ y $N = 500$, las mejores tasas las obtuvimos con $N = 500$ y estos fueron los modelos obtenidos

Modelo	E.Aparente	E.NoAparente	sd	Error C.V(k=5,B=1)	Beta(df)
Ridge	2.7631	30.9624	12.6477	27.7055	4089
Lasso	3.8615	31.5440	11.7046	20.85702	43
Elastic Net	3.5520	29.9227	12.1172	27.2087	61

Cuadro 4: Tasas de error presentadas en porcentajes para los modelos ajustados con train/test al 65 %-35 %

Modelo	`lambda.min`	df.max(l.min)	`lambda.1se`	df.max(l.1se)
Ridge	5.9341	4088	34.7565	4088
Lasso	0.03814	42	0.0666	32
Elastic Net	0.0663	60	0.3079	27

Cuadro 5: Datos complementarios para el Cuadro 4.

Podemos ver que, aunque los errores aparentes son menores, los No Aparentes son mayores y el número de variables que toman los modelos también, y esto puede deberse a que la base tenía muy pocas observaciones y el 65 % de los datos no eran suficientes para entrenar el modelo, por lo que el error era más grande, es por eso que al final decidimos usar *repeated training/test*(80/20) ya que así se podía entrenar mejor el modelo y por lo tanto, obtener mejores resultados.