NBA II

Val Huerta

10/15/2019

```
library(rsample)
```

```
## Loading required package: tidyr
```

```
library(glmnet)
```

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

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

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

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

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
#Primero he procedido a cambiar los nombres de las variables para facilitarme el entendimiento
#de las mismas, asi como la eliminacion de los posibles na que hubiera en la base de datos.
NBAdata <- read.csv("~/Documents/CUNEF/Predicción/Clase 2 /nba.csv")
NBAdata <- na.omit(NBAdata)

names(NBAdata)[3] = "Country"
names(NBAdata)[4] = "Ranking"
names(NBAdata)[6] = "Team"
names(NBAdata)[7] = "Partidos"
names(NBAdata)[8] = "Minutos"
names(NBAdata)[9] = "Efficiency"
names(NBAdata)[10] = "Acierto"
names(NBAdata)[11] = "IntentoTriple"
names(NBAdata)[12] = "IntentoLibre"
names(NBAdata)[13] = "ReboteAtaque"
names(NBAdata)[14] = "ReboteDefensa"
```

```r
names(NBAdata)[15] = "RebotesTotal"
names(NBAdata)[16] = "Asistencia"
names(NBAdata)[17] = "Robo"
names(NBAdata)[18] = "Bloqueo"
names(NBAdata)[19] = "PerdidaDeBalon"
names(NBAdata)[20] = "Compañerismo"
names(NBAdata)[21] = "BuenAtaque"
names(NBAdata)[22] = "BuenaDefensa"
names(NBAdata)[23] = "BuenoTotal"
names(NBAdata)[24] = "Contribución"

dim(NBAdata)
```

```
## [1] 483  28
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(leaps)

#Ahora procedo a realizar el modelo de regresion para poder estirmar las relaciones
#entre las variables. La varibale Player y Country no las considero relevantes para mi
#estudio ya que considero mas fuerte la relacion que hay del salario con el equipo en el que
#se juegue. Además en funcion de quien sea el jugador las demas variables serán mejores.
#(si es un jugador "bueno" sus resultados tambien lo seran)
#Tambien considero que el compañerismo no es base para la determinacion del salario.
#Por último no cuento tampoco con que tenga un buen ataque o una buena defensa ya que en
#la base de datos hay una variable definida como bueno total que es un conjunto de ambas y con
#considerar esa es suficiente.


#Nuestro objetivo es identificar y seleccionar, de entre todos los predictores disponibles,
#aquellos que están más relacionados con la variable "Salary" y así crear el mejor modelo.

#Para ello vamos a utilizar el modelo BACWARD que contiene desde el inicio todos los posibles
#predictores, en cada repeticion se van a generar todos los modelos que se pueden crear
#eliminando un solo predictor a la vez y se va a seleccionar el que menor RSS o mayor R^2 tenga.
#Este proceso se repite hasta que se llega al modelo nulo sin predictores.

BACKWARDmodel <- regsubsets(Salary~. - (Player + Country + Compañerismo + BuenAtaque + BuenaDefensa), da

BACKWARDmodel
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ . - (Player + Country + Compañerismo +
##     BuenAtaque + BuenaDefensa), data = NBAdata, nvmax = 23, method = "backward")
## 51 Variables  (and intercept)
##                Forced in Forced out
## Ranking            FALSE      FALSE
## Age                FALSE      FALSE
```

```
## TeamBOS            FALSE      FALSE
## TeamBRK            FALSE      FALSE
## TeamCHI            FALSE      FALSE
## TeamCHO            FALSE      FALSE
## TeamCLE            FALSE      FALSE
## TeamDAL            FALSE      FALSE
## TeamDEN            FALSE      FALSE
## TeamDET            FALSE      FALSE
## TeamGSW            FALSE      FALSE
## TeamHOU            FALSE      FALSE
## TeamIND            FALSE      FALSE
## TeamLAC            FALSE      FALSE
## TeamLAL            FALSE      FALSE
## TeamMEM            FALSE      FALSE
## TeamMIA            FALSE      FALSE
## TeamMIL            FALSE      FALSE
## TeamMIN            FALSE      FALSE
## TeamNOP            FALSE      FALSE
## TeamNYK            FALSE      FALSE
## TeamOKC            FALSE      FALSE
## TeamORL            FALSE      FALSE
## TeamPHI            FALSE      FALSE
## TeamPHO            FALSE      FALSE
## TeamPOR            FALSE      FALSE
## TeamSAC            FALSE      FALSE
## TeamSAS            FALSE      FALSE
## TeamTOR            FALSE      FALSE
## TeamTOT            FALSE      FALSE
## TeamUTA            FALSE      FALSE
## TeamWAS            FALSE      FALSE
## Partidos           FALSE      FALSE
## Minutos            FALSE      FALSE
## Efficiency         FALSE      FALSE
## Acierto            FALSE      FALSE
## IntentoTriple      FALSE      FALSE
## IntentoLibre       FALSE      FALSE
## ReboteAtaque       FALSE      FALSE
## ReboteDefensa      FALSE      FALSE
## RebotesTotal       FALSE      FALSE
## Asistencia         FALSE      FALSE
## Robo               FALSE      FALSE
## Bloqueo            FALSE      FALSE
## PerdidaDeBalon     FALSE      FALSE
## BuenoTotal         FALSE      FALSE
## Contribución       FALSE      FALSE
## OBPM               FALSE      FALSE
## DBPM               FALSE      FALSE
## BPM                FALSE      FALSE
## VORP               FALSE      FALSE
## 1 subsets of each size up to 23
## Selection Algorithm: backward
```

```
summary(BACKWARDmodel)
```

```
## Subset selection object
```

```
## Call: regsubsets.formula(Salary ~ . - (Player + Country + Compañerismo +
##     BuenAtaque + BuenaDefensa), data = NBAdata, nvmax = 23, method = "backward")
## 51 Variables  (and intercept)
##                Forced in Forced out
## Ranking            FALSE      FALSE
## Age                FALSE      FALSE
## TeamBOS            FALSE      FALSE
## TeamBRK            FALSE      FALSE
## TeamCHI            FALSE      FALSE
## TeamCHO            FALSE      FALSE
## TeamCLE            FALSE      FALSE
## TeamDAL            FALSE      FALSE
## TeamDEN            FALSE      FALSE
## TeamDET            FALSE      FALSE
## TeamGSW            FALSE      FALSE
## TeamHOU            FALSE      FALSE
## TeamIND            FALSE      FALSE
## TeamLAC            FALSE      FALSE
## TeamLAL            FALSE      FALSE
## TeamMEM            FALSE      FALSE
## TeamMIA            FALSE      FALSE
## TeamMIL            FALSE      FALSE
## TeamMIN            FALSE      FALSE
## TeamNOP            FALSE      FALSE
## TeamNYK            FALSE      FALSE
## TeamOKC            FALSE      FALSE
## TeamORL            FALSE      FALSE
## TeamPHI            FALSE      FALSE
## TeamPHO            FALSE      FALSE
## TeamPOR            FALSE      FALSE
## TeamSAC            FALSE      FALSE
## TeamSAS            FALSE      FALSE
## TeamTOR            FALSE      FALSE
## TeamTOT            FALSE      FALSE
## TeamUTA            FALSE      FALSE
## TeamWAS            FALSE      FALSE
## Partidos           FALSE      FALSE
## Minutos            FALSE      FALSE
## Efficiency         FALSE      FALSE
## Acierto            FALSE      FALSE
## IntentoTriple      FALSE      FALSE
## IntentoLibre       FALSE      FALSE
## ReboteAtaque       FALSE      FALSE
## ReboteDefensa      FALSE      FALSE
## RebotesTotal       FALSE      FALSE
## Asistencia         FALSE      FALSE
## Robo               FALSE      FALSE
## Bloqueo            FALSE      FALSE
## PerdidaDeBalon     FALSE      FALSE
## BuenoTotal         FALSE      FALSE
## Contribución       FALSE      FALSE
## OBPM               FALSE      FALSE
## DBPM               FALSE      FALSE
## BPM                FALSE      FALSE
```

```
## VORP               FALSE      FALSE
## 1 subsets of each size up to 23
## Selection Algorithm: backward
##           Ranking Age TeamBOS TeamBRK TeamCHI TeamCHO TeamCLE TeamDAL
## 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 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 14  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 15  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 16  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 17  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 18  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 19  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 20  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 21  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 22  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
## 23  ( 1 ) "*"     "*" " "     " "     " "     "*"     " "     " "
##           TeamDEN TeamDET TeamGSW TeamHOU TeamIND TeamLAC TeamLAL TeamMEM
## 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 ) " "     " "     " "     "*"     " "     " "     " "     " "
## 14  ( 1 ) " "     " "     " "     "*"     " "     " "     " "     " "
## 15  ( 1 ) " "     " "     " "     "*"     " "     " "     " "     " "
## 16  ( 1 ) " "     " "     " "     "*"     " "     " "     "*"     " "
## 17  ( 1 ) " "     " "     " "     "*"     "*"     " "     "*"     " "
## 18  ( 1 ) " "     " "     "*"     "*"     "*"     " "     "*"     " "
## 19  ( 1 ) " "     " "     "*"     "*"     "*"     " "     "*"     " "
## 20  ( 1 ) " "     " "     "*"     "*"     "*"     " "     "*"     " "
## 21  ( 1 ) " "     " "     "*"     "*"     "*"     " "     "*"     " "
## 22  ( 1 ) " "     " "     "*"     "*"     "*"     " "     "*"     " "
## 23  ( 1 ) " "     " "     "*"     "*"     "*"     " "     "*"     " "
##           TeamMIA TeamMIL TeamMIN TeamNOP TeamNYK TeamOKC TeamORL TeamPHI
## 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 ) " "      " "      " "      "*"      " "      "*"      " "      " "
## 14  ( 1 ) " "      " "      " "      "*"      " "      "*"      " "      "*"
## 15  ( 1 ) " "      " "      " "      "*"      " "      "*"      " "      "*"
## 16  ( 1 ) " "      " "      " "      "*"      " "      "*"      " "      "*"
## 17  ( 1 ) " "      " "      " "      "*"      " "      "*"      " "      "*"
## 18  ( 1 ) " "      " "      " "      "*"      " "      "*"      " "      "*"
## 19  ( 1 ) " "      " "      " "      "*"      " "      "*"      " "      "*"
## 20  ( 1 ) " "      " "      " "      "*"      " "      "*"      " "      "*"
## 21  ( 1 ) " "      " "      " "      "*"      " "      "*"      " "      "*"
## 22  ( 1 ) " "      "*"      " "      "*"      " "      "*"      " "      "*"
## 23  ( 1 ) "*"      "*"      " "      "*"      " "      "*"      " "      "*"
##           TeamPHO TeamPOR TeamSAC TeamSAS TeamTOR TeamTOT TeamUTA TeamWAS
## 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 ) " "      " "      " "      " "      " "      " "      " "      " "
## 14  ( 1 ) " "      " "      " "      " "      " "      " "      " "      " "
## 15  ( 1 ) " "      " "      " "      " "      " "      " "      "*"      " "
## 16  ( 1 ) " "      " "      " "      " "      " "      " "      "*"      " "
## 17  ( 1 ) " "      " "      " "      " "      " "      " "      "*"      " "
## 18  ( 1 ) " "      " "      " "      " "      " "      " "      "*"      " "
## 19  ( 1 ) " "      " "      " "      " "      " "      " "      "*"      " "
## 20  ( 1 ) " "      " "      " "      " "      " "      " "      "*"      " "
## 21  ( 1 ) " "      " "      " "      " "      " "      " "      "*"      " "
## 22  ( 1 ) " "      " "      " "      " "      " "      " "      "*"      " "
## 23  ( 1 ) " "      " "      " "      " "      " "      " "      "*"      " "
##           Partidos Minutos Efficiency Acierto IntentoTriple IntentoLibre
## 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 ) "*"      "*"     " "         " "     " "           " "
## 14  ( 1 ) "*"      "*"     " "         " "     " "           " "
## 15  ( 1 ) "*"      "*"     " "         " "     " "           " "
## 16  ( 1 ) "*"      "*"     " "         " "     " "           " "
## 17  ( 1 ) "*"      "*"     " "         " "     " "           " "
## 18  ( 1 ) "*"      "*"     " "         " "     " "           " "
## 19  ( 1 ) "*"      "*"     " "         " "     "*"           " "
## 20  ( 1 ) "*"      "*"     " "         " "     "*"           " "
## 21  ( 1 ) "*"      "*"     "*"         " "     "*"           " "
## 22  ( 1 ) "*"      "*"     "*"         " "     "*"           " "
## 23  ( 1 ) "*"      "*"     "*"         " "     "*"           " "
##           ReboteAtaque ReboteDefensa RebotesTotal Asistencia Robo Bloqueo
## 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 ) "*"          "*"           "*"          " "         " "  " "
## 14  ( 1 ) "*"          "*"           "*"          " "         " "  " "
## 15  ( 1 ) "*"          "*"           "*"          " "         " "  " "
## 16  ( 1 ) "*"          "*"           "*"          " "         " "  " "
## 17  ( 1 ) "*"          "*"           "*"          " "         " "  " "
## 18  ( 1 ) "*"          "*"           "*"          " "         " "  " "
## 19  ( 1 ) "*"          "*"           "*"          " "         " "  " "
## 20  ( 1 ) "*"          "*"           "*"          " "         " "  " "
## 21  ( 1 ) "*"          "*"           "*"          " "         " "  " "
## 22  ( 1 ) "*"          "*"           "*"          " "         " "  " "
## 23  ( 1 ) "*"          "*"           "*"          " "         " "  " "
##           PerdidaDeBalon BuenoTotal Contribución OBPM DBPM BPM VORP
## 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 ) " "            "*"        " "          " "  " "  " " "*"
## 14  ( 1 ) " "            "*"        " "          " "  " "  " " "*"
```

```
## 15  ( 1 ) " "           "*"        " "          " "  " "  " " " "*"
## 16  ( 1 ) " "           "*"        " "          " "  " "  " " " "*"
## 17  ( 1 ) " "           "*"        " "          " "  " "  " " " "*"
## 18  ( 1 ) " "           "*"        " "          " "  " "  " " " "*"
## 19  ( 1 ) " "           "*"        " "          " "  " "  " " " "*"
## 20  ( 1 ) " "           "*"        " "          "*"  " "  " " " "*"
## 21  ( 1 ) " "           "*"        " "          "*"  " "  " " " "*"
## 22  ( 1 ) " "           "*"        " "          "*"  " "  " " " "*"
## 23  ( 1 ) " "           "*"        " "          "*"  " "  " " " "*"
```

```
summary(BACKWARDmodel)$adjr2
```

```
##  [1] 0.2535726 0.3425206 0.4343876 0.4908340 0.5260436 0.5283861 0.5301497
##  [8] 0.5317862 0.5328823 0.5329839 0.5334534 0.5342143 0.5344041 0.5346559
## [15] 0.5345712 0.5345065 0.5343484 0.5340207 0.5335145 0.5331638 0.5329842
## [22] 0.5325231 0.5320958
```

```
which.max(summary(BACKWARDmodel)$adjr2)
```

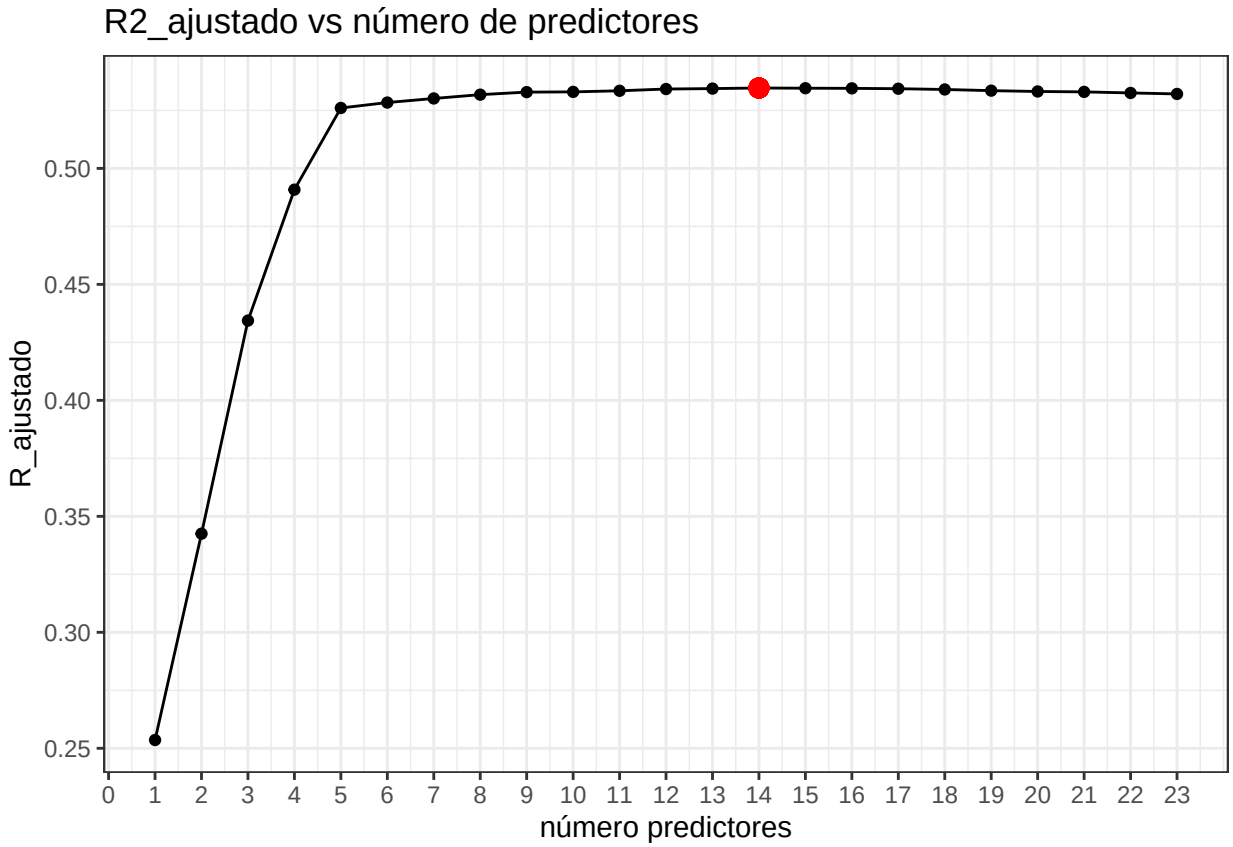
```
## [1] 14
```

```
#El que explica mejor la variable dependiente es el R^2

coef(object = BACKWARDmodel, 14)
```

```
##   (Intercept)       Ranking           Age       TeamCHO       TeamHOU
##  -6976313.547    -64933.445    527142.688   1883667.981  -2446823.204
##       TeamNOP       TeamOKC       TeamPHI      Partidos       Minutos
##  -1658977.783   1990123.431  -1482742.107   -163501.701      6059.103
##  ReboteAtaque ReboteDefensa  RebotesTotal    BuenoTotal          VORP
##  -1779445.511  -1651917.474   3503541.273    464630.338    774055.412
```

```
# Con esta funcion lo que encontramos son las variables conocidas como BETAS que son mas
#representativas para nuestro modelo y por tanto aquellas con las que trabajaremos.
```

```
#Mediante una representacion grafica comprobamos de una manera mas sencilla lo anteriormente
#realizado
p <- ggplot(data = data.frame(n_predictores = 1:23,
                              R_ajustado = summary(BACKWARDmodel)$adjr2),
            aes(x = n_predictores, y = R_ajustado)) +
    geom_line() +
    geom_point()

#Se identifica en rojo el máximo
p <- p + geom_point(aes(
                    x = n_predictores[which.max(summary(BACKWARDmodel)$adjr2)],
                    y = R_ajustado[which.max(summary(BACKWARDmodel)$adjr2)]),
                    colour = "red", size = 3)
p <- p +  scale_x_continuous(breaks = c(0:23)) +
          theme_bw() +
          labs(title = 'R2_ajustado vs número de predictores',
               x =  'número predictores')
p
```




```
#CROSS VALIDATION - VALIDATION SET

#Una vez realizado el modelo Backward, lo que quiero es estimar el test error de cada modelo
#y asi seleccionar el que menor error me propicie.
#Según la norma de one-standar-error, en resumen, es que seleccionemos el modelo mas simple de
#entre los cuales el test error sea semejante, es decir, que los modelos sean practicamente
#igual de buenos.

library(ISLR)
set.seed(1)
datos <- na.omit(NBAdata)

# Lo que estamos haciendo es:
# Emplear como training aproximadamente 2/3 de las observaciones, que en nuestro caso
#serian 322 Se seleccionan índices aleatorios que forman el training dataset

train <- sample(x = 1:483, size = 322, replace = FALSE)

# Los restantes forman el test dataset
```

```
BESTmodel <- regsubsets(Salary~.- (Player + Country + Compañerismo + BuenAtaque + BuenaDefensa), data =
BESTmodel
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ . - (Player + Country + Compañerismo +
##     BuenAtaque + BuenaDefensa), data = NBAdata[train, ], nvmax = 23,
##     method = "backward")
## 51 Variables  (and intercept)
```

```
##                Forced in Forced out
## Ranking            FALSE      FALSE
## Age                FALSE      FALSE
## TeamBOS            FALSE      FALSE
## TeamBRK            FALSE      FALSE
## TeamCHI            FALSE      FALSE
## TeamCHO            FALSE      FALSE
## TeamCLE            FALSE      FALSE
## TeamDAL            FALSE      FALSE
## TeamDEN            FALSE      FALSE
## TeamDET            FALSE      FALSE
## TeamGSW            FALSE      FALSE
## TeamHOU            FALSE      FALSE
## TeamIND            FALSE      FALSE
## TeamLAC            FALSE      FALSE
## TeamLAL            FALSE      FALSE
## TeamMEM            FALSE      FALSE
## TeamMIA            FALSE      FALSE
## TeamMIL            FALSE      FALSE
## TeamMIN            FALSE      FALSE
## TeamNOP            FALSE      FALSE
## TeamNYK            FALSE      FALSE
## TeamOKC            FALSE      FALSE
## TeamORL            FALSE      FALSE
## TeamPHI            FALSE      FALSE
## TeamPHO            FALSE      FALSE
## TeamPOR            FALSE      FALSE
## TeamSAC            FALSE      FALSE
## TeamSAS            FALSE      FALSE
## TeamTOR            FALSE      FALSE
## TeamTOT            FALSE      FALSE
## TeamUTA            FALSE      FALSE
## TeamWAS            FALSE      FALSE
## Partidos           FALSE      FALSE
## Minutos            FALSE      FALSE
## Efficiency         FALSE      FALSE
## Acierto            FALSE      FALSE
## IntentoTriple      FALSE      FALSE
## IntentoLibre       FALSE      FALSE
## ReboteAtaque       FALSE      FALSE
## ReboteDefensa      FALSE      FALSE
## RebotesTotal       FALSE      FALSE
## Asistencia         FALSE      FALSE
## Robo               FALSE      FALSE
## Bloqueo            FALSE      FALSE
## PerdidaDeBalon     FALSE      FALSE
## BuenoTotal         FALSE      FALSE
## Contribución       FALSE      FALSE
## OBPM               FALSE      FALSE
## DBPM               FALSE      FALSE
## BPM                FALSE      FALSE
## VORP               FALSE      FALSE
## 1 subsets of each size up to 23
## Selection Algorithm: backward
```

```r
#Como resultado obtenemos 23 modelos, el mejor para cada tamaño.
#En el siguiente paso lo que debemos hacer es compararlos mediante la estimacion del
#validation test error utilizando las observaciones que se han excluido del training
#y que se han designado como TEST.
```

```r
# Se genera un vector que almacenará el test-error de cada modelo, en nuestro caso son 23.

Error_validacion <- rep(NA, 23)

#Devuelve una matriz formada con los predictores indicados en la fórmula e introduce
#para todas las observaciones un intercept con valor 1, así al multiplicar por los
#coeficientes se obtiene el valor de la predicción (producto matricial).

test_matrix <- model.matrix(Salary~.-(Player + Country + Compañerismo + BuenAtaque + BuenaDefensa), data

#Para cada uno de los modelos almacenados en la variable mejores modelos
for (i in 1:23) {
# Se extraen los coeficientes del modelo
    coeficientes <- coef(object = BESTmodel, id = i)
# Se identifican los predictores que forman el modelo y se extraen de la
# matriz modelo
    predictores <- test_matrix[, names(coeficientes)]
# Se obtienen las predicciones mediante el producto matricial de los
# predictores extraídos y los coeficientes del modelo
    predicciones <- predictores %*% coeficientes
# Finalmente se calcula la estimación del test error como el promedio de
# los residuos al cuadrado (MSE)
    Error_validacion[i] <- mean((datos$Salary[-train] - predicciones)^2)
}

which.min(Error_validacion)
```

```
## [1] 5
```

```r
#El valor que minimiza mi error es el que esta en el puesto 5, por lo tanto el que utilizare
```

```r
sqrt(Error_validacion[5])
```

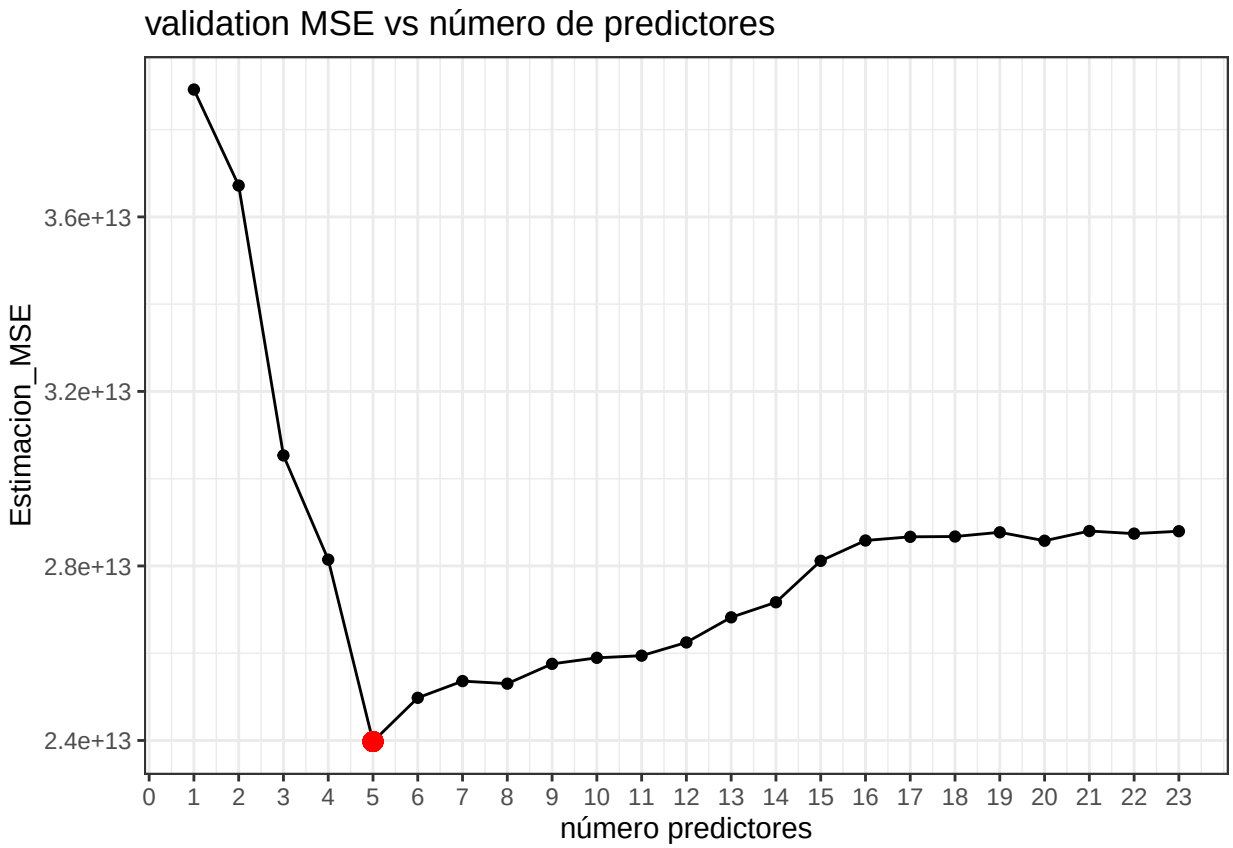
```
## [1] 4896481
```

```r
Error_validacion
```

```
##  [1] 3.891846e+13 3.671878e+13 3.053312e+13 2.814496e+13 2.397553e+13
##  [6] 2.497761e+13 2.536082e+13 2.530256e+13 2.575584e+13 2.589442e+13
## [11] 2.594343e+13 2.624688e+13 2.682247e+13 2.716753e+13 2.811705e+13
## [16] 2.858343e+13 2.866784e+13 2.867446e+13 2.877114e+13 2.857765e+13
## [21] 2.880046e+13 2.874082e+13 2.879600e+13
```

```r
#REPRESEANTACION GRAFICA.

p <- ggplot(data = data.frame(n_predictores = 1:23,
                              Estimacion_MSE = Error_validacion),
            aes(x = n_predictores, y = Estimacion_MSE)) +
    geom_line() +
    geom_point()
```

```
p <- p + geom_point(aes(x = n_predictores[which.min(Error_validacion)],
                        y = Error_validacion[which.min(Error_validacion)]),
                        colour = "red", size = 3)

p <- p +  scale_x_continuous(breaks = c(0:23)) +
          theme_bw() +
          labs(title = 'validation MSE vs número de predictores',
               x =  'número predictores')
p
```




```
#En ultimo lugar, al tener ya identificada la cantidad de predictores que debe contener
#nuestro modelo que en nuestro caso es 5, debemos volver a ajustar los posibles modelos
#con 5 predictores empleando tanto las variables de training como las de test (training + test)

BESTmodel <- regsubsets(Salary~.-(Player + Country + Compañerismo + BuenAtaque + BuenaDefensa), data = H

coef(object = BESTmodel, id = 5)
```

```
##  (Intercept)      Ranking          Age     Partidos      Minutos
## -5241479.325   -71763.309   505712.127  -157365.354     5323.941
##   BuenoTotal
##  1006514.726
```

```
#El problema de este modelo de validacion simple es que depende mucho de como se
#repartan las observaciones entre el train y el test.
```

```
#ELASTIC NET
#Es una combinacion de Ridge y Lasso, que se utilizan para minimizar el problema
```

```r
#entre el sesgo y la varianza proporcionando una disminucion del error de prediccion.

set.seed(123)
NBA_split <- initial_split(NBAdata, prop = .7, strata = "Salary")
NBA_train <- training(NBA_split)
NBA_test  <- testing(NBA_split)
```

```r
NBA_train_x <- model.matrix(Salary ~ ., NBA_train)[, -1]
NBA_train_y <- NBA_train$Salary
```

```r
NBA_test_x <- model.matrix(Salary ~ ., NBA_test)[, -1]
NBA_test_y <- NBA_test$Salary
```

```r
train_control <- trainControl(method = "cv", number = 10)

caret_mod <- train(
  x = NBA_train_x,
  y = NBA_train_y,
  method = "glmnet",
  preProc = c("center", "scale", "zv", "nzv"),
  trControl = train_control,
  tuneLength = 10
)

caret_mod
```

```
## glmnet
##
## 340 samples
## 579 predictors
##
## Pre-processing: centered (26), scaled (26), remove (553)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 306, 307, 306, 305, 307, 306, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda       RMSE     Rsquared   MAE
##   0.1       2224.515  5146814  0.5409626  3864783
##   0.1       5138.917  5145612  0.5411241  3864153
##   0.1      11871.566  5124532  0.5441487  3853863
##   0.1      27424.857  5091368  0.5489261  3835882
##   0.1      63354.979  5044686  0.5559698  3806902
##   0.1     146358.223  4998532  0.5618430  3768133
##   0.1     338106.486  4957839  0.5662415  3717012
##   0.1     781069.855  4943834  0.5666805  3673339
##   0.1    1804372.714  5018288  0.5557827  3710713
##   0.1    4168335.102  5252188  0.5310015  3974872
##   0.2       2224.515  5168913  0.5377707  3874178
##   0.2       5138.917  5153467  0.5399142  3868380
##   0.2      11871.566  5119243  0.5448659  3851923
##   0.2      27424.857  5076429  0.5513521  3829856
##   0.2      63354.979  5016212  0.5607240  3792275
##   0.2     146358.223  4962408  0.5674223  3741340
##   0.2     338106.486  4911439  0.5736247  3674802
```

```
##   0.2     781069.855  4932272  0.5681465  3644368
##   0.2    1804372.714  5122360  0.5385227  3796284
##   0.2    4168335.102  5402596  0.5240863  4179900
##   0.3       2224.515  5153466  0.5400965  3868373
##   0.3       5138.917  5143103  0.5414899  3863518
##   0.3      11871.566  5111935  0.5459383  3848661
##   0.3      27424.857  5060275  0.5540744  3823045
##   0.3      63354.979  4996139  0.5636456  3779905
##   0.3     146358.223  4931945  0.5721561  3715975
##   0.3     338106.486  4885009  0.5775149  3642105
##   0.3     781069.855  4961135  0.5623496  3648316
##   0.3    1804372.714  5183836  0.5331067  3881152
##   0.3    4168335.102  5618974  0.5045224  4422513
##   0.4       2224.515  5145125  0.5408690  3865277
##   0.4       5138.917  5136106  0.5422414  3859640
##   0.4      11871.566  5103974  0.5472055  3844981
##   0.4      27424.857  5042712  0.5570394  3814962
##   0.4      63354.979  4978548  0.5662482  3767878
##   0.4     146358.223  4910105  0.5754231  3696703
##   0.4     338106.486  4873648  0.5789564  3620657
##   0.4     781069.855  5017798  0.5519840  3688510
##   0.4    1804372.714  5246940  0.5295931  3974474
##   0.4    4168335.102  5858070  0.4790800  4666144
##   0.5       2224.515  5150449  0.5403004  3867317
##   0.5       5138.917  5130533  0.5431269  3858049
##   0.5      11871.566  5095916  0.5485018  3841920
##   0.5      27424.857  5025411  0.5599869  3805787
##   0.5      63354.979  4962164  0.5688602  3755249
##   0.5     146358.223  4890304  0.5784177  3677733
##   0.5     338106.486  4875116  0.5782093  3609753
##   0.5     781069.855  5086834  0.5393728  3746446
##   0.5    1804372.714  5332016  0.5225481  4082723
##   0.5    4168335.102  6125399  0.4377961  4914709
##   0.6       2224.515  5148866  0.5404869  3866562
##   0.6       5138.917  5126276  0.5437290  3856582
##   0.6      11871.566  5087871  0.5498225  3839725
##   0.6      27424.857  5011110  0.5623451  3797458
##   0.6      63354.979  4948963  0.5708465  3743821
##   0.6     146358.223  4876049  0.5805999  3661228
##   0.6     338106.486  4887672  0.5756581  3604757
##   0.6     781069.855  5132368  0.5321558  3794526
##   0.6    1804372.714  5432217  0.5123217  4201547
##   0.6    4168335.102  6299820  0.4296185  5062016
##   0.7       2224.515  5148556  0.5406692  3869072
##   0.7       5138.917  5122531  0.5443206  3855790
##   0.7      11871.566  5078611  0.5513313  3835903
##   0.7      27424.857  5000678  0.5638969  3791174
##   0.7      63354.979  4937246  0.5725513  3732977
##   0.7     146358.223  4865369  0.5822019  3646051
##   0.7     338106.486  4907523  0.5718312  3606232
##   0.7     781069.855  5154749  0.5308889  3826932
##   0.7    1804372.714  5537461  0.5016342  4320771
##   0.7    4168335.102  6477081  0.4259076  5198159
##   0.8       2224.515  5154047  0.5402117  3870670
```

```
##   0.8       5138.917  5120762  0.5447453  3855491
##   0.8      11871.566  5069042  0.5529216  3831884
##   0.8      27424.857  4991953  0.5651929  3785638
##   0.8      63354.979  4926738  0.5740544  3723442
##   0.8     146358.223  4857225  0.5833968  3633484
##   0.8     338106.486  4934816  0.5667018  3618330
##   0.8     781069.855  5184013  0.5287663  3865238
##   0.8    1804372.714  5656295  0.4881073  4449023
##   0.8    4168335.102  6659203  0.4241159  5333751
##   0.9       2224.515  5151444  0.5406035  3869388
##   0.9       5138.917  5116620  0.5453775  3852970
##   0.9      11871.566  5058998  0.5546510  3827448
##   0.9      27424.857  4983750  0.5664485  3780462
##   0.9      63354.979  4916771  0.5755708  3714773
##   0.9     146358.223  4852405  0.5839932  3624641
##   0.9     338106.486  4967184  0.5605800  3640910
##   0.9     781069.855  5221984  0.5253230  3912749
##   0.9    1804372.714  5789429  0.4702686  4576958
##   0.9    4168335.102  6841085  0.4287927  5466742
##   1.0       2224.515  5143021  0.5413283  3864545
##   1.0       5138.917  5114915  0.5455188  3851445
##   1.0      11871.566  5050807  0.5560279  3823852
##   1.0      27424.857  4976452  0.5675612  3775363
##   1.0      63354.979  4907219  0.5770188  3706173
##   1.0     146358.223  4850099  0.5841768  3617818
##   1.0     338106.486  5003114  0.5536886  3668352
##   1.0     781069.855  5273081  0.5199058  3968743
##   1.0    1804372.714  5911437  0.4505485  4684360
##   1.0    4168335.102  7040751  0.4377896  5609269
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 1 and lambda = 146358.2.
```

```
#El valor de alpha me ha dado 1, lo que determina que estamos ante un metodo Lasso.
#El método lasso fuerza a que las estimaciones de los coeficientes de los predictores
#tiendan a cero. La diferencia con Ridge es que lasso sí es capaz de fijar
#algunos de ellos exactamente a cero, lo que permite además de reducir la varianza,
#realizar selección de predictores.

library(glmnet)
# x e y son la matriz modelo y el vector respuesta creados anteriormente con
# los datos de NBAdata
cv_lasso <- cv.glmnet(x = NBA_train_x, y = NBA_train_y, alpha = 1)
min(cv_lasso$cvm)
```

```
## [1] 2.818485e+13
```

```
pred <- predict(cv_lasso,s=cv_lasso$lambda.min,NBA_test_x)
mean((NBA_test_y-  pred)^2)
```

```
## [1] 3.409145e+13
```

```
sqrt(3.341885e+13)
```

```
## [1] 5780904
```

```
#El modelo tiene un error de 5780904 millones de euros
```