Tema6_Ejercicio

Fran Camacho

2025-02-14

Tema 6 - Ejercicio

En la librería MASS podemos encontrar el dataset Boston, el cual incluye 506 observaciones de 14 variables relacionadas con el mercado de la vivienda de dicha ciudad estadounidense. Puede encontrarse información detallada sobre el contenido de dichas variables en el siguiente enlace:

https://www.rdocumentation.org/packages/MASS/versions/7.3-54/topics/Boston

This data frame contains the following columns:

crim per capita crime rate by town. zn proportion of residential land zoned for lots over 25,000 sq.ft. indus proportion of non-retail business acres per town. chas Charles River dummy variable (= 1 if tract bounds river; 0 otherwise). nox nitrogen oxides concentration (parts per 10 million). rm average number of rooms per dwelling. age proportion of owner-occupied units built prior to 1940. dis weighted mean of distances to five Boston employment centres. rad index of accessibility to radial highways. tax full-value property-tax rate per $10,000. ptratio pupil-teacher ratio by town. black (a strange formula) where is the proportion of blacks by town. lstat lower status of the population (percent). medv median value of owner-occupied homes in $1000s.

Paso 1: Carga de los datos

```
#Load data from CRAN package MASS
#install.packages("MASS")
library(MASS)
```

Paso 2: Explorar y preparar los datos

Carga de paquetes que son necesarios para diversas funciones.

```
if (!require(GGally)) install.packages('GGally', dependencies = T)
```

```
## Cargando paquete requerido: GGally
```

```
## Cargando paquete requerido: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(GGally)

if (!require(gridExtra)) install.packages('gridExtra', dependencies = T)
```

```
## Cargando paquete requerido: gridExtra
```

```
library(gridExtra)

if (!require(lmtest)) install.packages('lmtest', dependencies = T)
```

```
## Cargando paquete requerido: lmtest

## Cargando paquete requerido: zoo

##
## Adjuntando el paquete: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(lmtest)

if (!require(car)) install.packages('car', dependencies = T)
```

```
## Cargando paquete requerido: car

## Cargando paquete requerido: carData
```

```
library(car)
```

Examinamos la estructura y el aspecto del dataset importado:

```
#Structure
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
#Summary
summary(Boston)
```

```
##       crim                 zn               indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm              age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

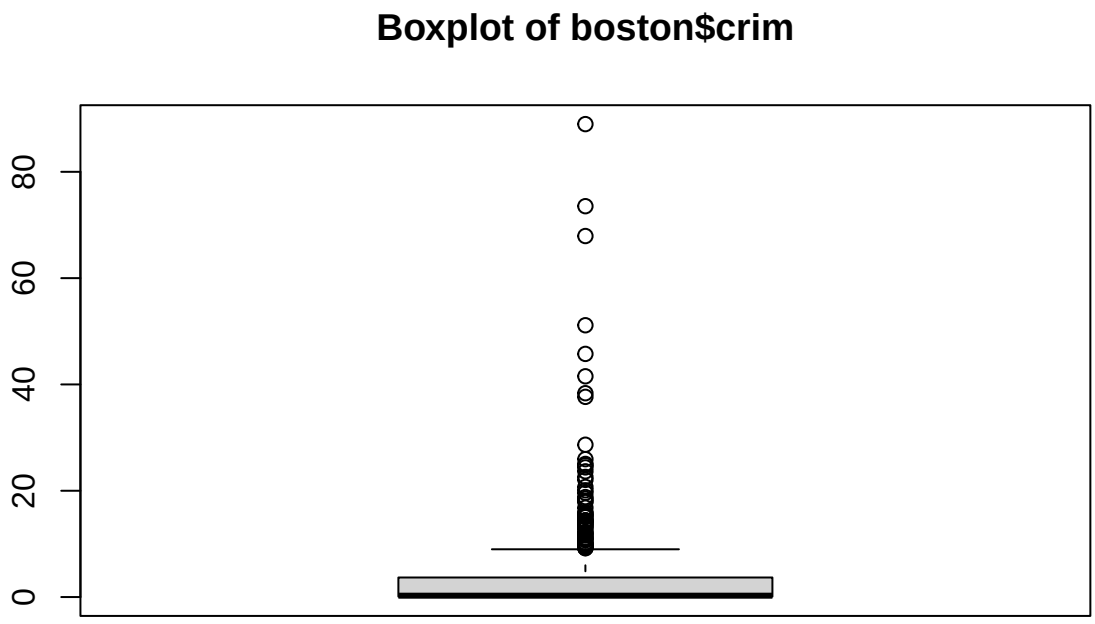
La variable dependiente en este estudio es medv (valor medio de la vivienda en 1000 de $). Al ser numérica, está incluida en el resumen estadístico.

```
#check there are no nulls

boston <- Boston
write.csv(boston,file='Chapter06/boston.csv') #<- export data to be used lately with Python & scikit-le

#sum(is.na(boston)) -> 0
#count total missing values in each column of data frame
#sapply(boston, function(x) sum(is.na(x)))
colSums(is.na(boston))
```

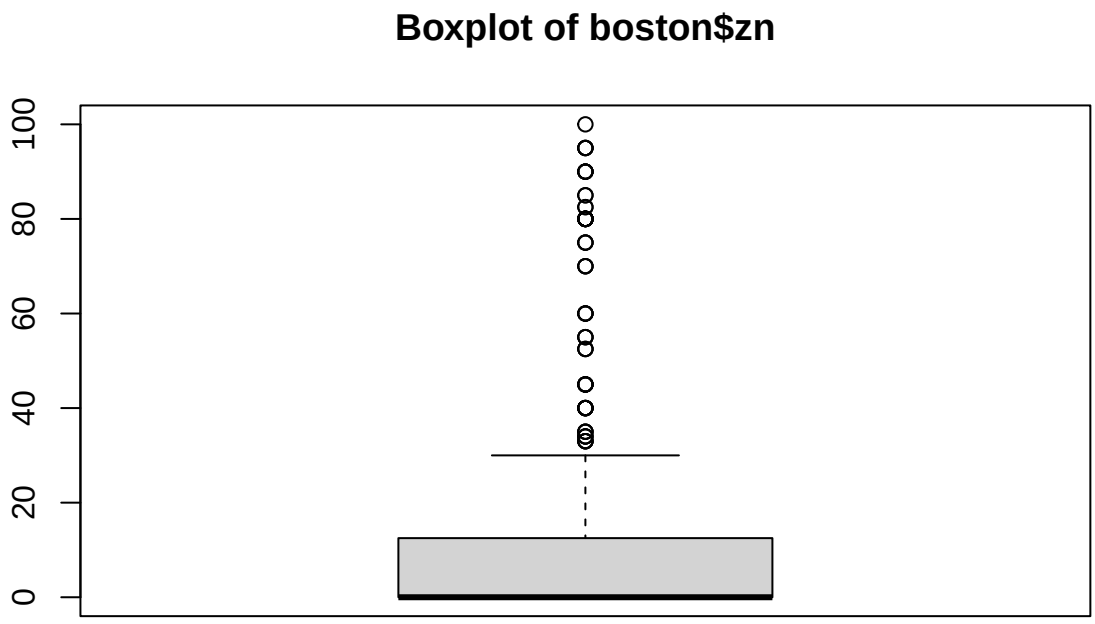
```
##    crim      zn   indus    chas     nox      rm     age     dis     rad     tax 
##       0       0       0       0       0       0       0       0       0       0 
## ptratio   black   lstat    medv 
##       0       0       0       0
```

En el sumario estadístico, se puede ver que algunas variables tienen una media y mediana bastante separadas: crim, zn. Las examinamos con más detalle mediante diagramas de caja.

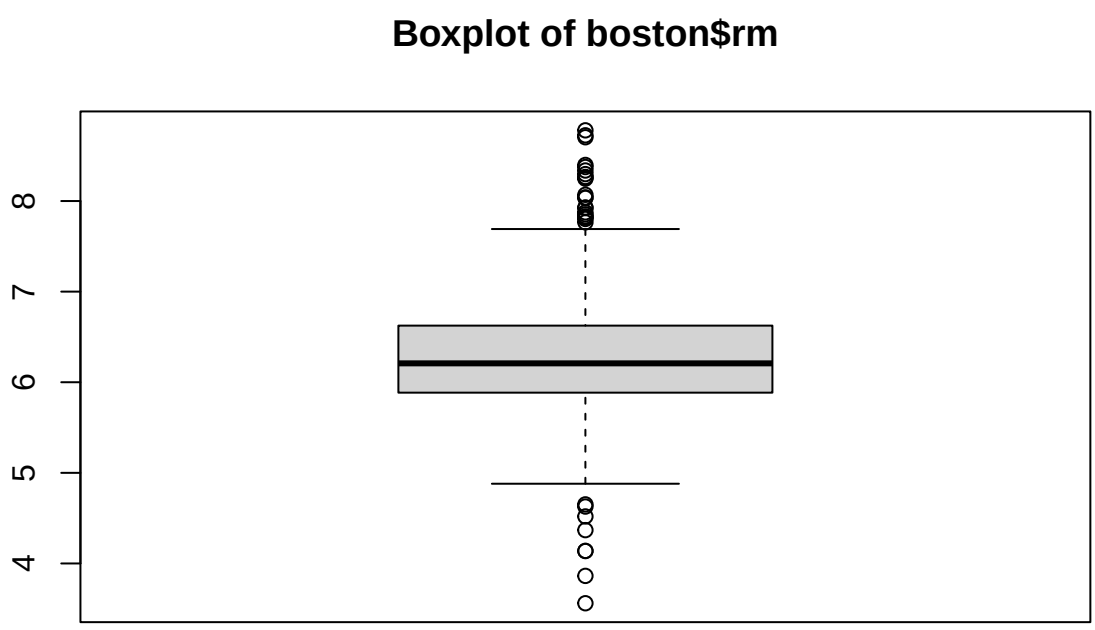
```r
boxplot(x = boston$crim, main = "Boxplot of boston$crim")
```



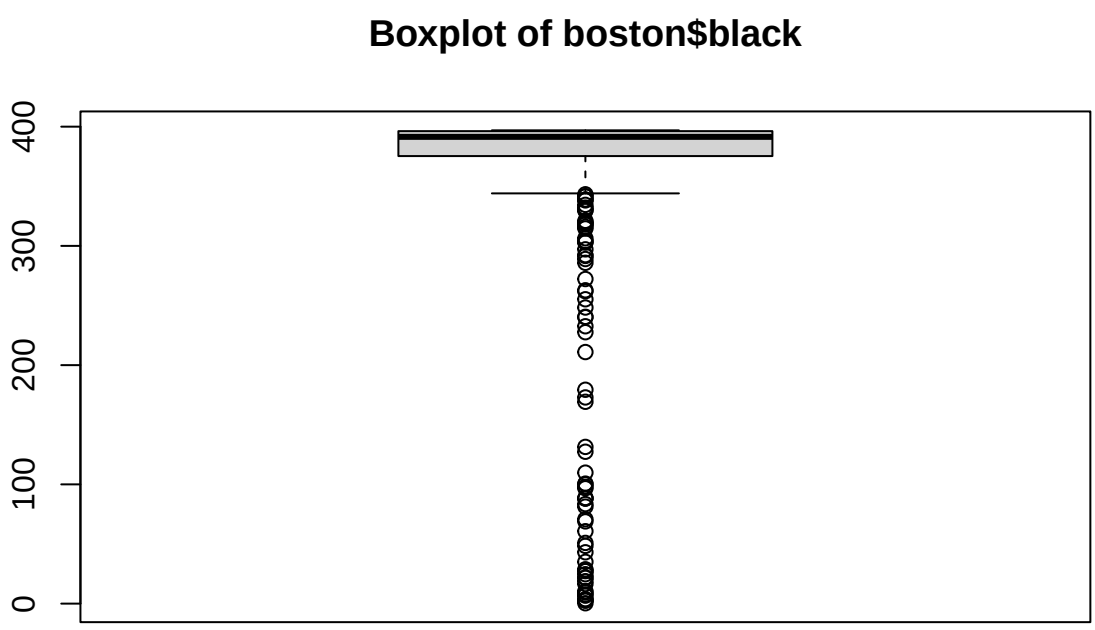

```r
boxplot(x = boston$zn, main = "Boxplot of boston$zn")
```




```
#boxplot(x = boston$indus, main = "Boxplot of boston$indus")
#boxplot(x = boston$nox, main = "Boxplot of boston$nox")
boxplot(x = boston$rm, main = "Boxplot of boston$rm")
```




```
#boxplot(x = boston$age, main = "Boxplot of boston$age")
#boxplot(x = boston$dis, main = "Boxplot of boston$dis")
#boxplot(x = boston$tax, main = "Boxplot of boston$tax")
#boxplot(x = boston$ptratio, main = "Boxplot of boston$ptratio")
boxplot(x = boston$black, main = "Boxplot of boston$black")
```




```
#boxplot(x = boston$lstat, main = "Boxplot of boston$lstat")
```

Al hacer los diagramas de caja de todos los predictores (por si acaso), he visto que también hay muchos valores atípicos para black y rm. (rm no creo que afecte tanto, porque el rango es muy pequeño -de hecho la mediana y la media de rm no son muy diferentes-). Entiendo que estas variables pueden provocar efectos indeseados en el resultado final.

Análisis de correlación

Para poder establecer un modelo de regresión lineal múltiple, lo primero es estudiar la relación que existe entre las variables independientes. Para ello, comenzamos obteniendo la matriz de correlación entre todas las variables disponibles

```
corr_matrix <- round(cor(x = boston, method = "pearson"), 3)

corr_matrix
```

```
##           crim     zn  indus   chas    nox     rm    age    dis    rad    tax
## crim     1.000 -0.200  0.407 -0.056  0.421 -0.219  0.353 -0.380  0.626  0.583
## zn      -0.200  1.000 -0.534 -0.043 -0.517  0.312 -0.570  0.664 -0.312 -0.315
## indus    0.407 -0.534  1.000  0.063  0.764 -0.392  0.645 -0.708  0.595  0.721
## chas    -0.056 -0.043  0.063  1.000  0.091  0.091  0.087 -0.099 -0.007 -0.036
## nox      0.421 -0.517  0.764  0.091  1.000 -0.302  0.731 -0.769  0.611  0.668
## rm      -0.219  0.312 -0.392  0.091 -0.302  1.000 -0.240  0.205 -0.210 -0.292
```

```
## age      0.353 -0.570  0.645  0.087  0.731 -0.240  1.000 -0.748  0.456  0.506
## dis     -0.380  0.664 -0.708 -0.099 -0.769  0.205 -0.748  1.000 -0.495 -0.534
## rad      0.626 -0.312  0.595 -0.007  0.611 -0.210  0.456 -0.495  1.000  0.910
## tax      0.583 -0.315  0.721 -0.036  0.668 -0.292  0.506 -0.534  0.910  1.000
## ptratio  0.290 -0.392  0.383 -0.122  0.189 -0.356  0.262 -0.232  0.465  0.461
## black   -0.385  0.176 -0.357  0.049 -0.380  0.128 -0.274  0.292 -0.444 -0.442
## lstat    0.456 -0.413  0.604 -0.054  0.591 -0.614  0.602 -0.497  0.489  0.544
## medv    -0.388  0.360 -0.484  0.175 -0.427  0.695 -0.377  0.250 -0.382 -0.469
##         ptratio  black  lstat   medv
## crim      0.290 -0.385  0.456 -0.388
## zn       -0.392  0.176 -0.413  0.360
## indus     0.383 -0.357  0.604 -0.484
## chas     -0.122  0.049 -0.054  0.175
## nox       0.189 -0.380  0.591 -0.427
## rm       -0.356  0.128 -0.614  0.695
## age       0.262 -0.274  0.602 -0.377
## dis      -0.232  0.292 -0.497  0.250
## rad       0.465 -0.444  0.489 -0.382
## tax       0.461 -0.442  0.544 -0.469
## ptratio   1.000 -0.177  0.374 -0.508
## black    -0.177  1.000 -0.366  0.333
## lstat     0.374 -0.366  1.000 -0.738
## medv     -0.508  0.333 -0.738  1.000
```

```
#corr_matrix[,14]
corr_matrix[,14][order(abs(corr_matrix[,14]))]
```

```
##    chas     dis   black      zn     age     rad    crim     nox     tax   indus
##   0.175   0.250   0.333   0.360  -0.377  -0.382  -0.388  -0.427  -0.469  -0.484
## ptratio      rm   lstat    medv
##  -0.508   0.695  -0.738   1.000
```

```
#sort(abs(corr_matrix[,14]))
```

Los valores que parecen más relacionados con el precio de la vivienda son 'lstat' (correlación negativa) y 'rm' (positiva). (Parece lógico -y un poco clasista también, pero bueno-: cuantas más habitaciones, mayor precio, cuanto más porcentaje de 'clase baja' en la zona, menor precio). (Por no hablar de la variable 'black'). Las variables que parecen menos relacionados con el precio de la vivienda son 'chas', 'dis', 'black' y 'zn'.

También se ve que están muy relacionadas entre sí 'indus' y 'nox', 'tax' y 'rad'. (Investigando en internet he visto que la multicolinealidad puede ser un problema).

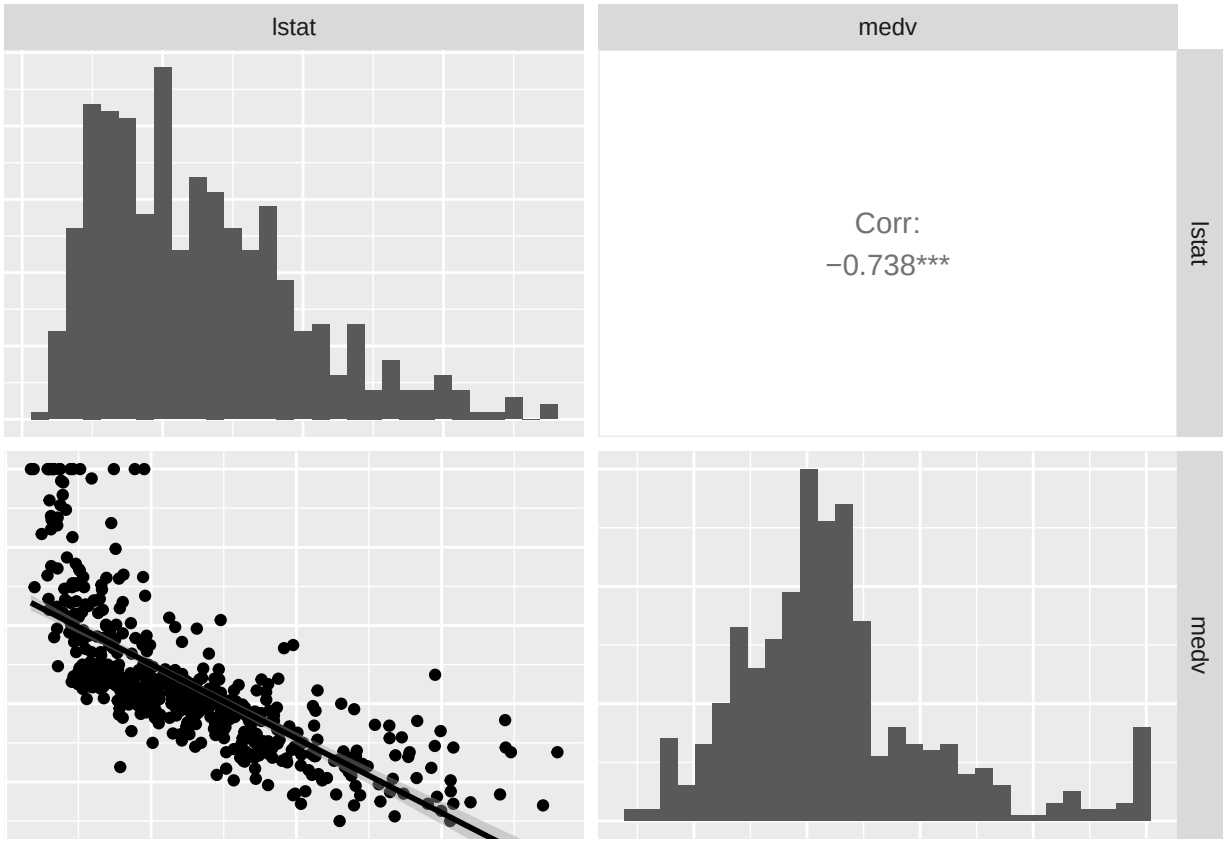
Como se indica en el material complementario de este tema, también se puede utilizar la función "ggpairs" de la libreria **GGally**. Con esta función, además de los valores de correlación para cada par de variables, también se obtiene los diagramas de dispersión y la distribución de cada una de las variables:

```
#ggpairs(boston, lower = list(continuous = "smooth"), diag = list(continuous = "barDiag"), axisLabels =
```

No puedo ver bien aquí el resultado poniendo todas las variables juntas, así que elijo algunas de ellas: (En python, como en la pantalla del navegador donde se ejecutan los notebooks the JupyterLab hay más espacio, se ve algo mejor).

```
ggpairs(boston[,c(13,14)], lower = list(continuous = "smooth"), diag = list(continuous = "barDiag"), axi
```

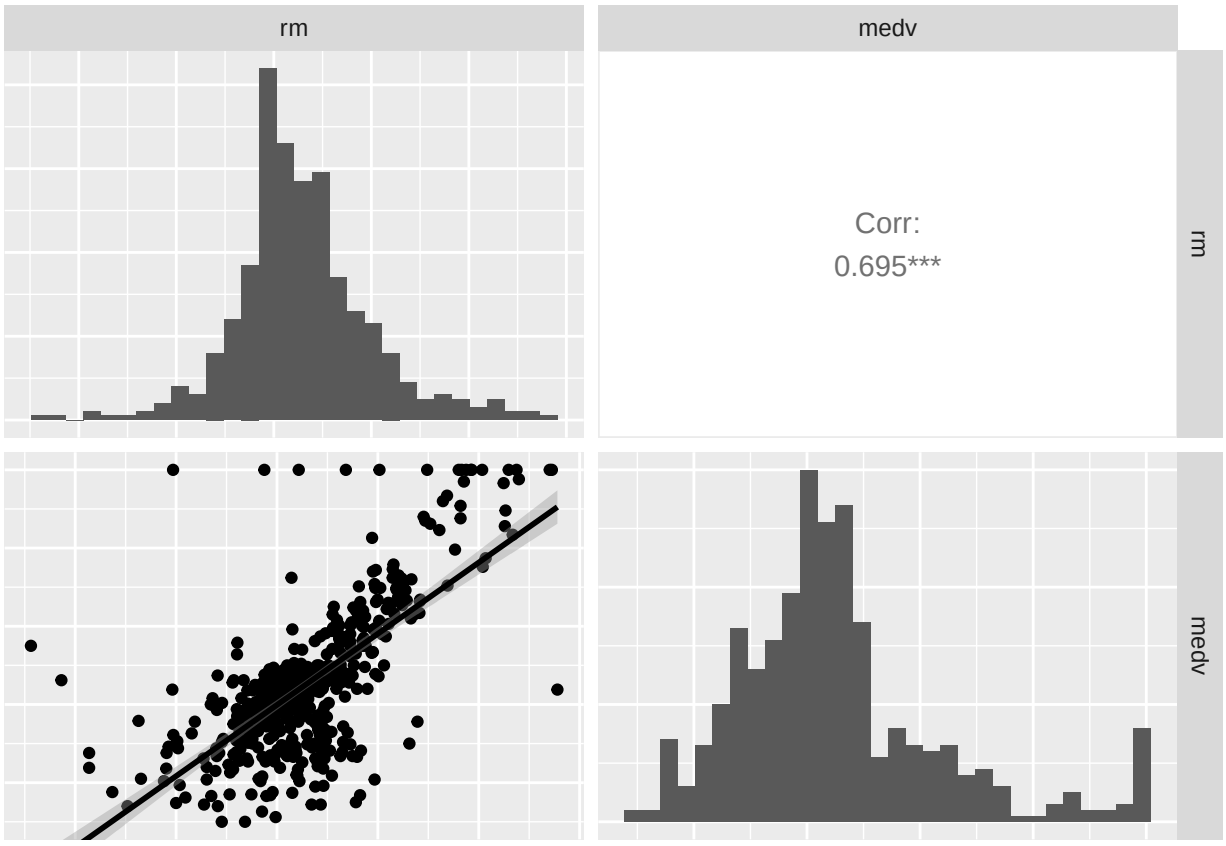
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



La relación entre lstat y medv puede que no sea exactamente lineal. Quizá pueda ser una mejora del modelo.

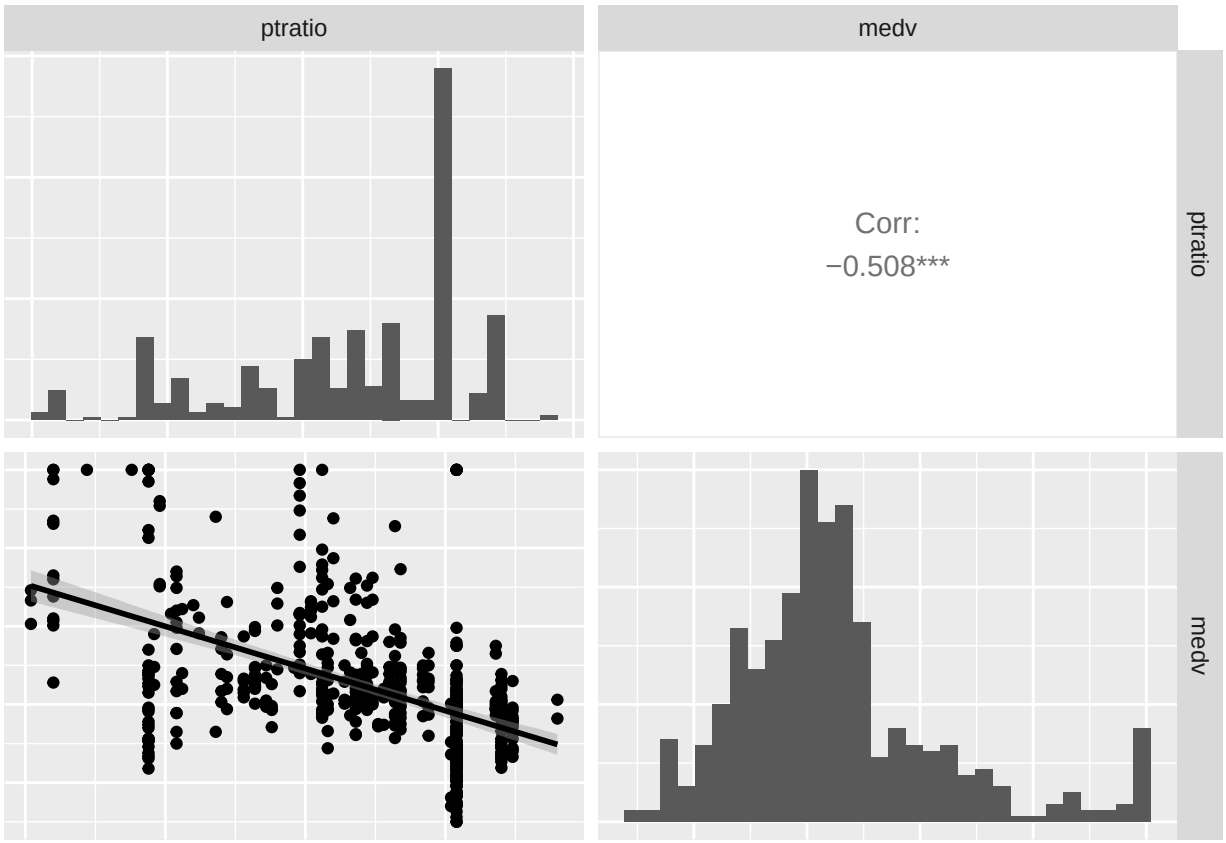
```
ggpairs(boston[,c(6,14)], lower = list(continuous = "smooth"), diag = list(continuous = "barDiag"), axis
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



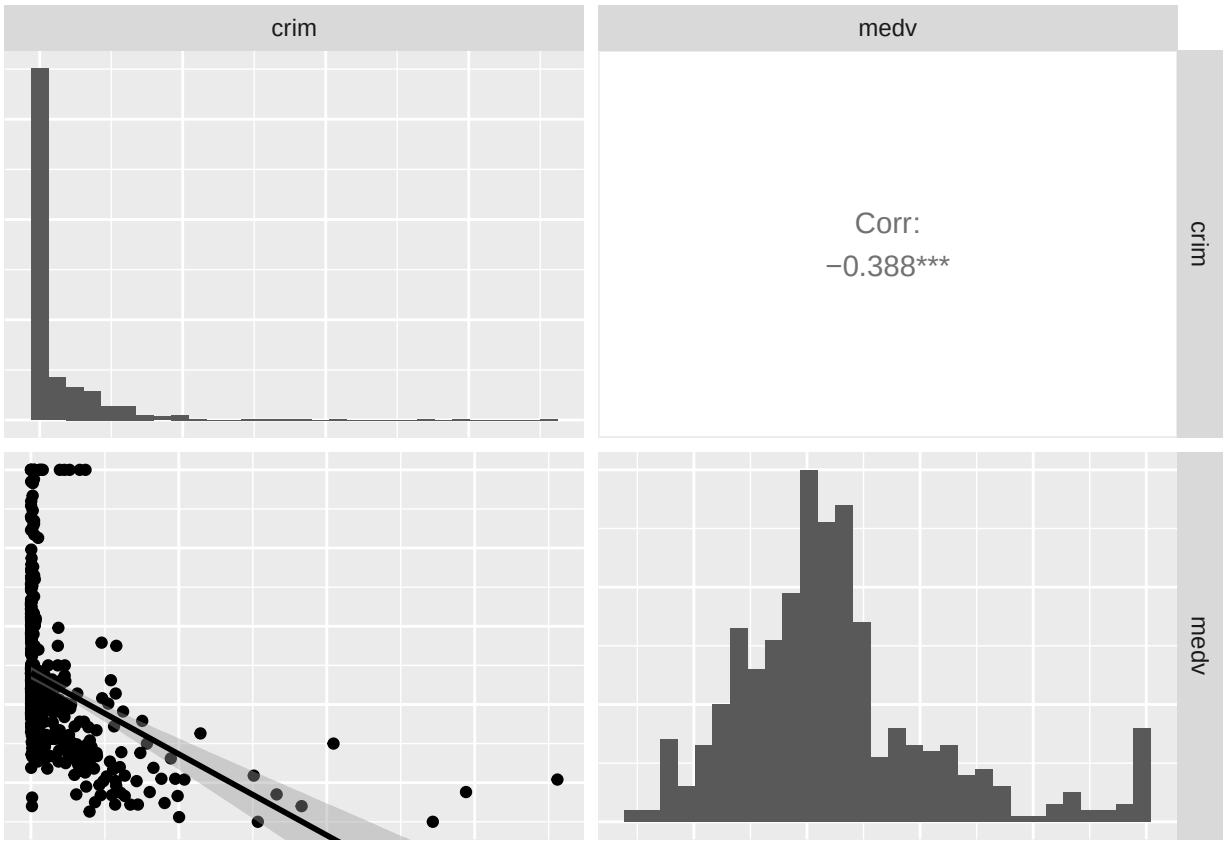

```
ggpairs(boston[,c(11,14)], lower = list(continuous = "smooth"), diag = list(continuous = "barDiag"), ax
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggpairs(boston[,c(1,14)], lower = list(continuous = "smooth"), diag = list(continuous = "barDiag"), axis
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



No veo que entre estas dos variables haya alguna relación "razonable" la verdad. El diagrama de dispersion me parece muy extraño.

Una vez visto todo esto (- variables que son cualitativas como chas y rad - variables que están muy muy relacionadas entre sí, como tax y rad - variables que tienen muchos valores atípicos como crim, zn y black) estimamos el modelo con todas ellas, y luego utilizaremos el criterio AIC para ver si es conveniente eliminar alguna.

```
model <- lm(medv ~ ., data = boston)

summary(model)
```

```
##
## Call:
## lm(formula = medv ~ ., data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
```

```
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

El modelo con todos los predictores tiene un $\mathbf{R^2}$ de **0.7406**, lo que indica que es capaz de explicar un 74% de la variabilidad del precio de las viviendas. El hecho de que el **valor p** del estadístico F sea despreciable ($<$ **2.2e-16**), y también los valores p de todas las variables menos 2 ('indus' y 'age'), indican que es probable que el modelo sea correcto. (Así que con todo lo extrañas que me parecen esas otras variables, parece que las que no aportan son indus y age).

Vamos a comprobar si efectivamente esas dos variables 'indus' y 'age' no aportan nada al modelo. Para ello utilizaremos el criterio de Akaike (AIC - Akaike Information Criterion):

```
#?step -> Choose a model by AIC in a Stepwise Algorithm
step(object = model, direction = "both", trace = 1)
```

```
## Start:  AIC=1589.64
## medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad +
##     tax + ptratio + black + lstat
##
##           Df Sum of Sq   RSS    AIC
## - age      1      0.06 11079 1587.7
## - indus    1      2.52 11081 1587.8
## <none>                 11079 1589.6
## - chas     1    218.97 11298 1597.5
## - tax      1    242.26 11321 1598.6
## - crim     1    243.22 11322 1598.6
## - zn       1    257.49 11336 1599.3
## - black    1    270.63 11349 1599.8
## - rad      1    479.15 11558 1609.1
## - nox      1    487.16 11566 1609.4
## - ptratio  1   1194.23 12273 1639.4
## - dis      1   1232.41 12311 1641.0
## - rm       1   1871.32 12950 1666.6
## - lstat    1   2410.84 13490 1687.3
##
## Step:  AIC=1587.65
## medv ~ crim + zn + indus + chas + nox + rm + dis + rad + tax +
##     ptratio + black + lstat
##
##           Df Sum of Sq   RSS    AIC
```

```
## - indus    1      2.52 11081 1585.8
## <none>                 11079 1587.7
## + age      1      0.06 11079 1589.6
## - chas     1    219.91 11299 1595.6
## - tax      1    242.24 11321 1596.6
## - crim     1    243.20 11322 1596.6
## - zn       1    260.32 11339 1597.4
## - black    1    272.26 11351 1597.9
## - rad      1    481.09 11560 1607.2
## - nox      1    520.87 11600 1608.9
## - ptratio  1   1200.23 12279 1637.7
## - dis      1   1352.26 12431 1643.9
## - rm       1   1959.55 13038 1668.0
## - lstat    1   2718.88 13798 1696.7
##
## Step:  AIC=1585.76
## medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio +
##     black + lstat
##
##           Df Sum of Sq   RSS    AIC
## <none>                 11081 1585.8
## + indus    1      2.52 11079 1587.7
## + age      1      0.06 11081 1587.8
## - chas     1    227.21 11309 1594.0
## - crim     1    245.37 11327 1594.8
## - zn       1    257.82 11339 1595.4
## - black    1    270.82 11352 1596.0
## - tax      1    273.62 11355 1596.1
## - rad      1    500.92 11582 1606.1
## - nox      1    541.91 11623 1607.9
## - ptratio  1   1206.45 12288 1636.0
## - dis      1   1448.94 12530 1645.9
## - rm       1   1963.66 13045 1666.3
## - lstat    1   2723.48 13805 1695.0

##
## Call:
## lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad +
##     tax + ptratio + black + lstat, data = boston)
##
## Coefficients:
## (Intercept)         crim           zn         chas          nox           rm
##   36.341145    -0.108413     0.045845     2.718716   -17.376023     3.801579
##         dis          rad          tax      ptratio        black        lstat
##   -1.492711     0.299608    -0.011778    -0.946525     0.009291    -0.522553
```

Pues según este criterio (es mejor un valor AIC menor, y que el modelo tenga menos variables), el mejor modelo es

medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio + black + lstat

es decir, el que efectivamente no contiene ni indus ni age.

```r
model <- lm(medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio + black + lstat, data = bost

summary(model)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad +
##     tax + ptratio + black + lstat, data = boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.5984  -2.7386  -0.5046   1.7273  26.2373
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.341145   5.067492   7.171 2.73e-12 ***
## crim         -0.108413   0.032779  -3.307 0.001010 **
## zn            0.045845   0.013523   3.390 0.000754 ***
## chas          2.718716   0.854240   3.183 0.001551 **
## nox         -17.376023   3.535243  -4.915 1.21e-06 ***
## rm            3.801579   0.406316   9.356  < 2e-16 ***
## dis          -1.492711   0.185731  -8.037 6.84e-15 ***
## rad           0.299608   0.063402   4.726 3.00e-06 ***
## tax          -0.011778   0.003372  -3.493 0.000521 ***
## ptratio      -0.946525   0.129066  -7.334 9.24e-13 ***
## black         0.009291   0.002674   3.475 0.000557 ***
## lstat        -0.522553   0.047424 -11.019  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.736 on 494 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7348
## F-statistic: 128.2 on 11 and 494 DF,  p-value: < 2.2e-16
```

Se comprueba que el valor R^2 es igual que el del modelo completo, pero el valor estadístico F es mejor.

[

PREGUNTA 1

Si eliminamos chas:

```r
model_no_chas <- lm(medv ~ crim + zn + nox + rm + dis + rad + tax + ptratio + black + lstat, data = bos

summary(model_no_chas)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + nox + rm + dis + rad + tax +
##     ptratio + black + lstat, data = boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.3716  -2.7943  -0.5508   1.8942  26.3982
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.620311   5.113241   7.162 2.90e-12 ***
## crim         -0.114056   0.033032  -3.453 0.000602 ***
## zn            0.045742   0.013647   3.352 0.000864 ***
## nox         -16.469153   3.556086  -4.631 4.65e-06 ***
## rm            3.844639   0.409818   9.381  < 2e-16 ***
## dis          -1.526099   0.187136  -8.155 2.89e-15 ***
## rad           0.315531   0.063785   4.947 1.04e-06 ***
## tax          -0.012674   0.003391  -3.737 0.000208 ***
## ptratio      -0.978442   0.129857  -7.535 2.34e-13 ***
## black         0.009730   0.002695   3.611 0.000337 ***
## lstat        -0.528103   0.047827 -11.042  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.78 on 495 degrees of freedom
## Multiple R-squared:  0.7353, Adjusted R-squared:  0.7299
## F-statistic: 137.5 on 10 and 495 DF,  p-value: < 2.2e-16
```

El valor R^2 baja levemente, y el valor estadístico F en cambio sube.

Merece la pena mantener una variable como chas en el modelo? Yo diría que no, pero claro, me gustaría saber tu opinión.

]

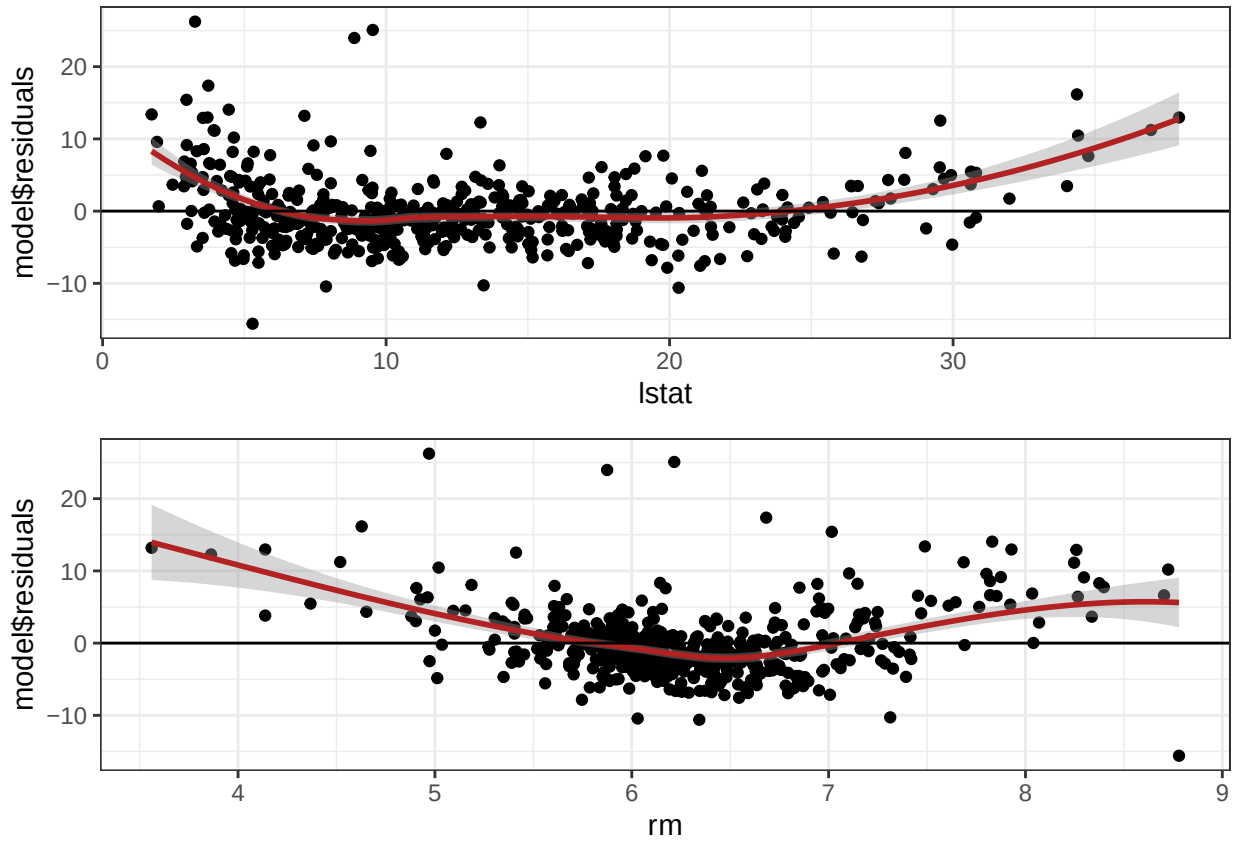
Validez del modelo

Pasamos a validar el modelo. Debemos asegurarnos de que cumple los supuestos de un modelo de regresión lineal:

Si para los predictores, los residuos se distribuyen aleatoriamente a lo largo del eje x en un diagrama de dispersión con esos predictores, entonces la relación es lineal.

```r
library(ggplot2)
library(gridExtra)

#lstat
plot_lstat <- ggplot(data = boston, aes(lstat, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()


#rm
plot_rm <- ggplot(data = boston, aes(rm, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()

grid.arrange(plot_lstat, plot_rm)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



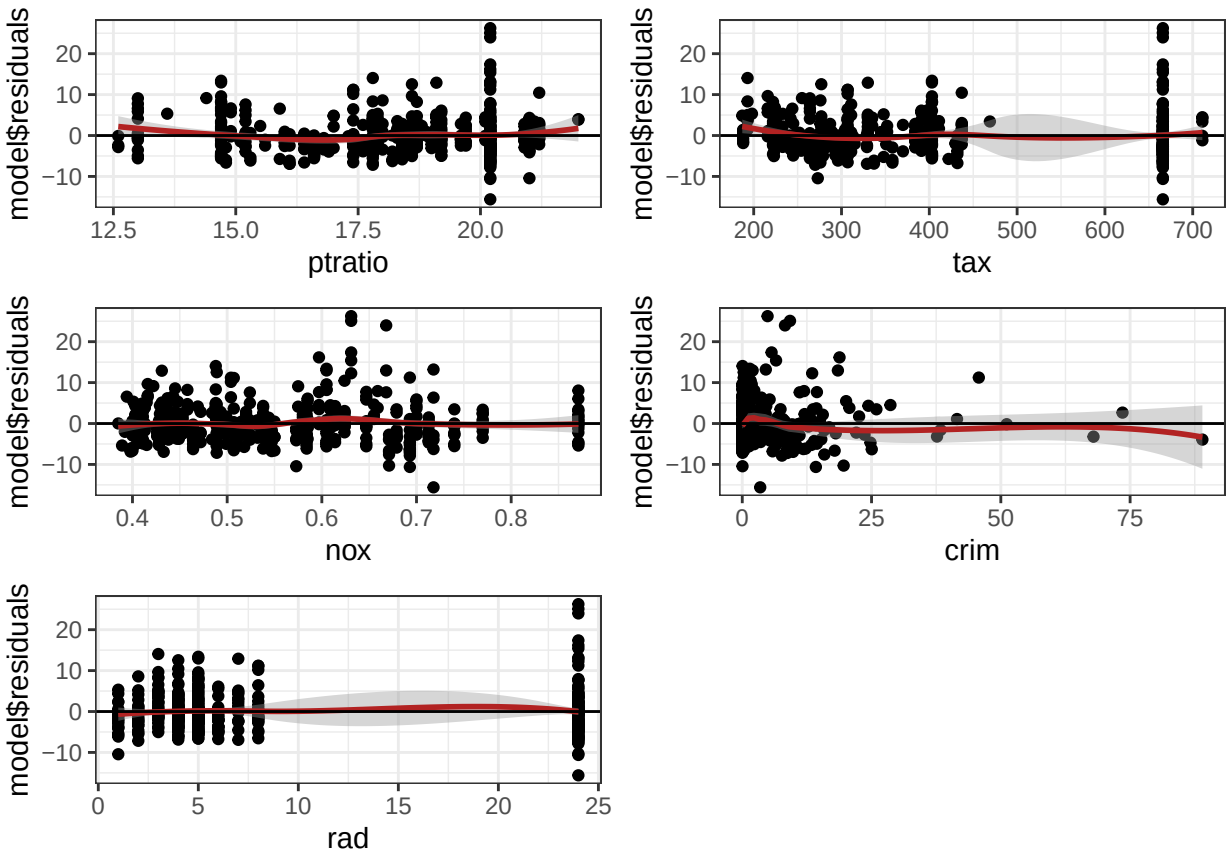

```
# model_no_chas almost identical
```

Hay la misma proporción de puntos arriba y abajo del 0. Pero para lstat los puntos se acumulan en la parte izquierda del eje x, y para rm en la parte central. No se reparten a lo largo de todo el eje x de manera aleatoria.

```
# seven intermediate

#ptratio
plot_ptratio <- ggplot(data = boston, aes(ptratio, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()

#indus
#plot_indus <- ggplot(data = boston, aes(indus, model$residuals)) +
#geom_point() + geom_smooth(color = "firebrick") +
##geom_hline(yintercept = 0) +
#theme_bw()

#tax
plot_tax <- ggplot(data = boston, aes(tax, model$residuals)) +
```

```
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()

#nox
plot_nox <- ggplot(data = boston, aes(nox, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()

#crim
plot_crim <- ggplot(data = boston, aes(crim, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()

#rad
plot_rad <- ggplot(data = boston, aes(rad, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()

#age
#plot_age <- ggplot(data = boston, aes(age, model$residuals)) +
#geom_point() + geom_smooth(color = "firebrick") +
#geom_hline(yintercept = 0) +
#theme_bw()


#grid.arrange(plot_ptratio, plot_indus, plot_tax, plot_nox, plot_crim, plot_rad, plot_age)
grid.arrange(plot_ptratio, plot_tax, plot_nox, plot_crim, plot_rad)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



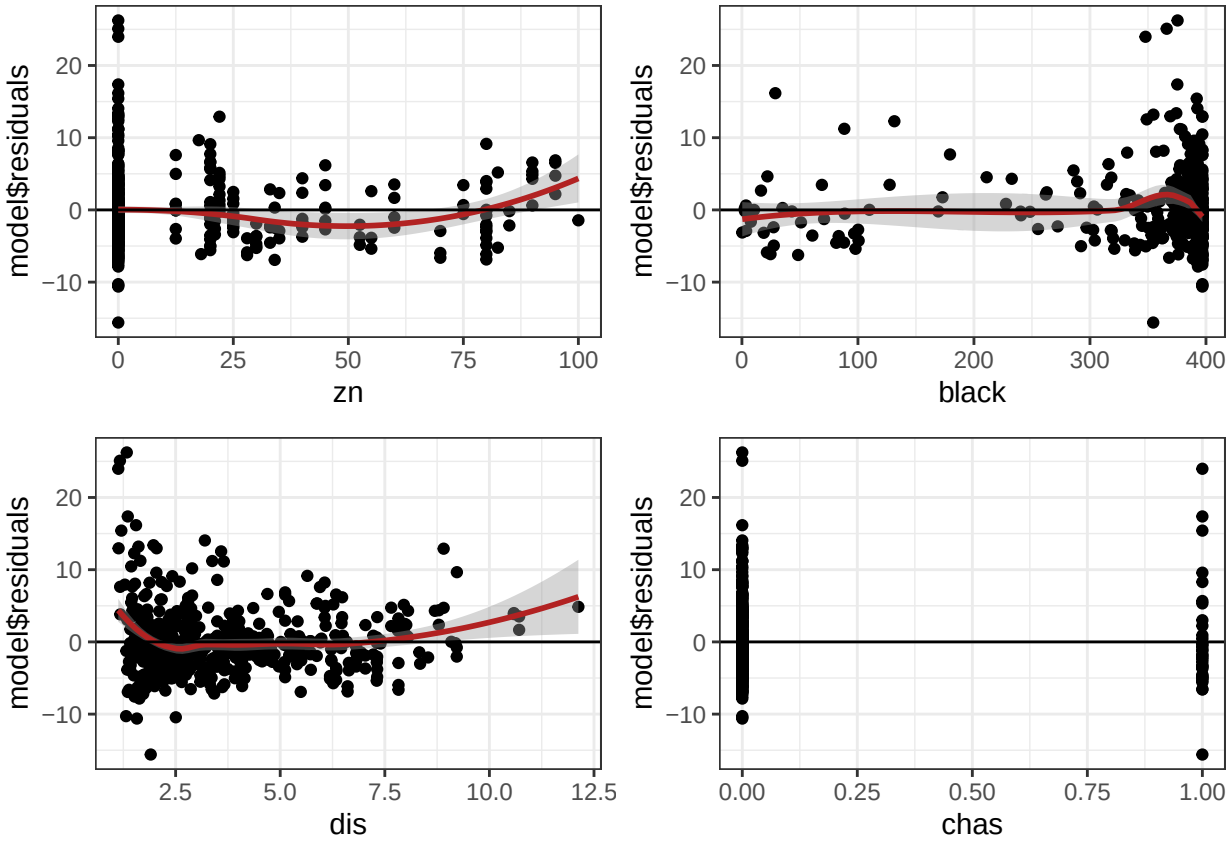
Para tax, crim y rad, los puntos no se reparten a lo largo del eje x.

```
#last four

#zn
plot_zn <- ggplot(data = boston, aes(zn, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()

#black
plot_black <- ggplot(data = boston, aes(black, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()

#dis
plot_dis <- ggplot(data = boston, aes(dis, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
theme_bw()

#chas
plot_chas <- ggplot(data = boston, aes(chas, model$residuals)) +
geom_point() + geom_smooth(color = "firebrick") +
geom_hline(yintercept = 0) +
```

```
theme_bw()


grid.arrange(plot_zn, plot_black, plot_dis, plot_chas)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at -0.5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 13

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 0

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 156.25

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x
## else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : pseudoinverse used at
## -0.5

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x
## else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : neighborhood radius 13

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x
## else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : reciprocal condition
## number 0

## Warning in predLoess(object$y, object$x, newx = if (is.null(newdata)) object$x
## else if (is.data.frame(newdata))
## as.matrix(model.frame(delete.response(terms(object)), : There are other near
## singularities as well. 156.25

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : radius 2.5e-05

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : all data on boundary of neighborhood. make span bigger
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 1.01

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : zero-width neighborhood. make span bigger

## Warning: Failed to fit group -1.
## Caused by error in `predLoess()`:
## ! NA/NaN/Inf en llamada a una función externa (arg 5)
```



zn podría pasar, black y dis no tienen los puntos repartidos a lo largo del eje x. (chas es un caso especial, claro).

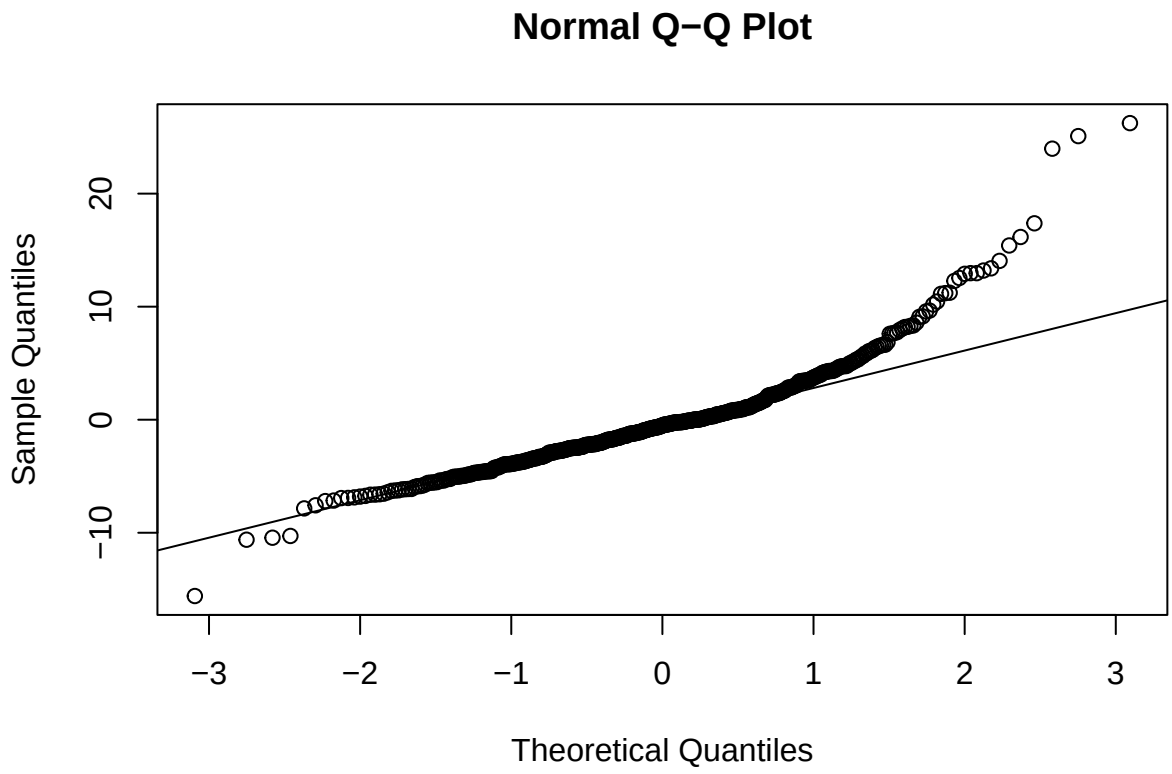
Vamos a examinar ahora los siguientes 3 criterios para los residuos: normalidad, homocedasticidad, y autocorrelación.

Normalidad:

Veamos a continuación si los residuos de la regresión se distribuyen según una Normal.

Para ello, realizaremos un gráfico Q-Q y un test Shapiro-Wilk:

```
#Q-Q graph
qqnorm(model$residuals)
qqline(model$residuals)
```



La línea de puntos se separa de la línea recta .. (lo que indica no normalidad).

Shapiro-Wilk

```
shapiro.test(model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.90131, p-value < 2.2e-16
```

Según el test de Shapiro-Wilk, se confirma que los residuos no siguen una distribución normal. (El valor p es significativo, por lo que el valor W que es superior a 0.05 indica la no normalidad).

Otra manera/librería :

```r
if (!require(olsrr)) install.packages('olsrr', dependencies = T)
```

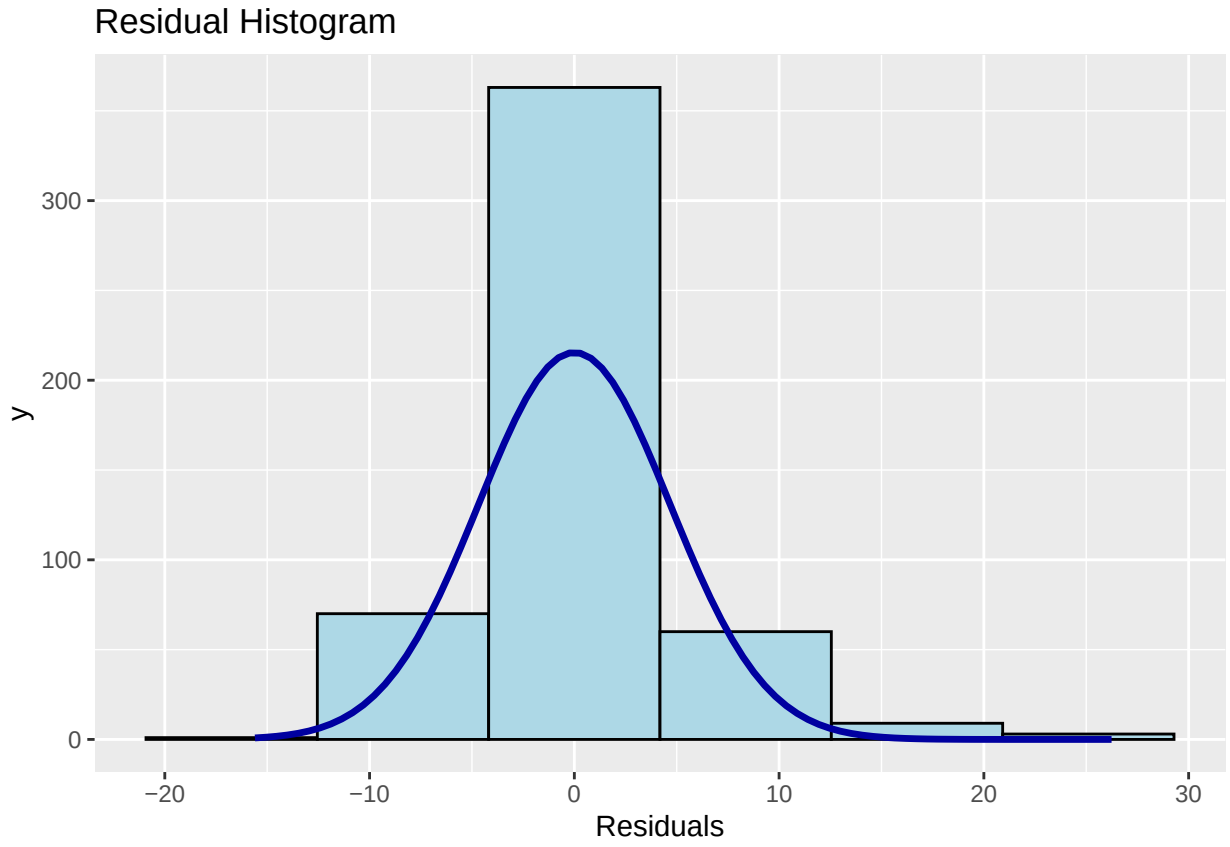
```
## Cargando paquete requerido: olsrr
```

```
##
## Adjuntando el paquete: 'olsrr'
```

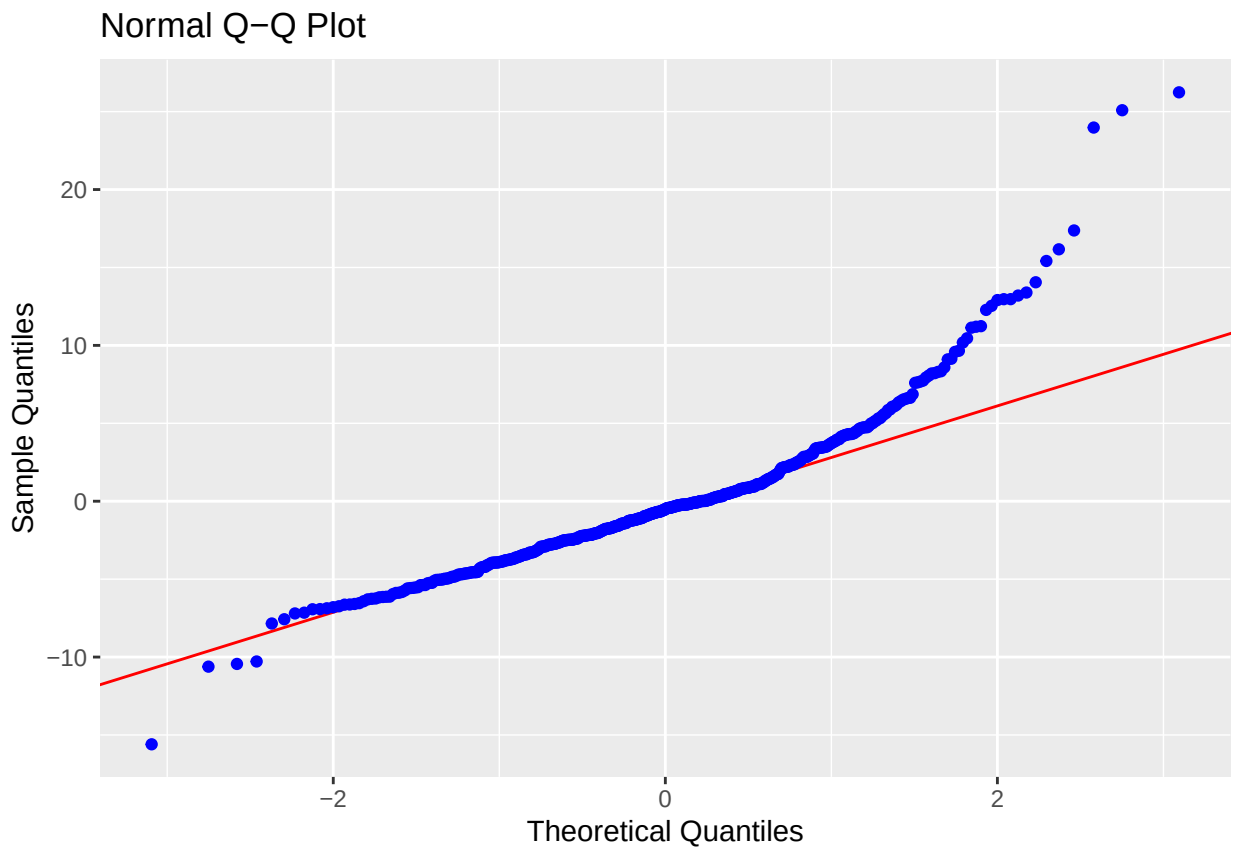
```
## The following object is masked from 'package:MASS':
##
##     cement
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

```r
library(olsrr)

ols_plot_resid_hist(model)
```




```r
ols_plot_resid_qq(model)
```



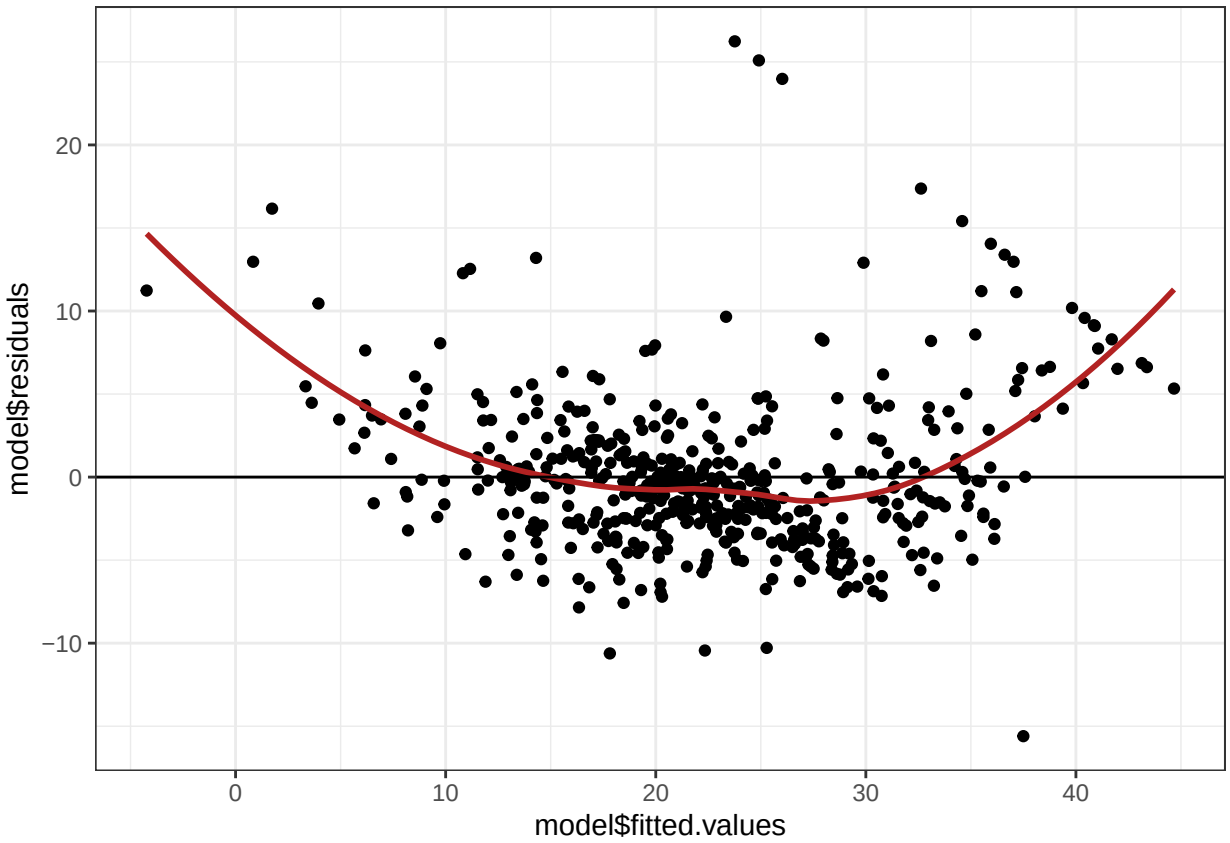
Homocedasticidad

Pasamos a revisar la homocedasticidad de los residuos. Para ello, representamos los residuos frente a los valores ajustados por el modelo. Confirmaremos la homocedasticidad si los primeros se distribuyen de forma aleatoria en torno a cero, manteniendo aproximadamente la misma variabilidad a lo largo del eje X.

Si, por el contrario, se observara algún patrón específico significaría que la variabilidad es dependiente del valor ajustado y por lo tanto se violaría el supuesto de homocedasticidad de los residuos:

```
ggplot(data = boston, aes(model$fitted.values, model$residuals)) +
geom_point() +
geom_smooth(color = "firebrick", se = FALSE) +
geom_hline(yintercept = 0) +
theme_bw()
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Los residuos se acumulan en la zona central, alrededor del 20. No parece que se distribuyan uniformemente a lo largo del eje x. (También me llama mucho la atención una línea recta que va desde las 12 en punto a las 3 en punto).

Breusch-Pagan

```
#library(lmtest)
bptest(model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model
## BP = 59.907, df = 11, p-value = 9.647e-09
```

El contraste de Breusch-Pagan devuelve un valor muy superior a 0, siendo p muy pequeño, lo que indica la falta de homocedasticidad (y la presencia de heterocedasticidad).

[https://en.wikipedia.org/wiki/Breusch%E2%80%93Pagan_test

. . . it tests whether the variance of the errors from a regression is dependent on the values of the independent variables. In that case, heteroskedasticity is present.]

Autocorrelación

Durbin-Watson -> autocorrelación

El análisis de la posible presencia de autocorrelación en los residuos a través del contraste de Durbin-Watson

```
#library(car)
dwt(model, alternative = "two.sided")
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.4544515      1.077875       0
##  Alternative hypothesis: rho != 0
```

```
#library(lmtest)
dwtest(model)
```

```
##
##  Durbin-Watson test
##
## data:  model
## DW = 1.0779, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

Pues también se da la presencia de autocorrelación.

Es decir, no tenemos normalidad ni homecedasticidad, y sí autocorrelación ... Deduzco que los valores predichos por el modelo no serán de una gran valor.

Mejora del modelo

Vamos a ver qué pasa si tenemos en cuenta la posible relación no lineal entre lstat y medv:

```
model_lstat2 <- lm(medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio + black + lstat + I(ls

summary(model_lstat2)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad +
##     tax + ptratio + black + lstat + I(lstat^2), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.5603  -2.6709  -0.3071   1.9469  25.0085
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  42.124187   4.632559   9.093  < 2e-16 ***
## crim         -0.149072   0.030006  -4.968 9.34e-07 ***
## zn            0.021281   0.012500   1.703 0.089291 .
## chas          2.589821   0.775307   3.340 0.000900 ***
## nox         -13.534522   3.229619  -4.191 3.30e-05 ***
## rm            3.233174   0.372802   8.673  < 2e-16 ***
## dis          -1.357892   0.169051  -8.032 7.09e-15 ***
## rad           0.271744   0.057599   4.718 3.11e-06 ***
## tax          -0.009546   0.003068  -3.111 0.001970 **
## ptratio      -0.790820   0.118089  -6.697 5.82e-11 ***
```

```
## black         0.008174   0.002429   3.365 0.000824 ***
## lstat        -1.669567   0.119012 -14.029  < 2e-16 ***
## I(lstat^2)    0.033055   0.003198  10.337  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.298 on 493 degrees of freedom
## Multiple R-squared:  0.7868, Adjusted R-squared:  0.7816
## F-statistic: 151.6 on 12 and 493 DF,  p-value: < 2.2e-16
```

El R-squared pasa del 74% al 78.7%, y el estadístico F pasa de 128 a 151.6 Parece una mejora.

También, que zn ha dejado de ser significativo.

[

PREGUNTA 2

Si elimino zn, obtengo

```
#model_lstat2 <- lm(medv ~ crim + chas + nox + rm + dis + rad + tax + ptratio + black + lstat + I(lstat
#summary(model_lstat2)
```

Multiple R-squared: **0.7855**, Adjusted R-squared: 0.7808 F-statistic: **164.5** on 11 and 494 DF, p-value: < 2.2e-16

Es decir, el valor de R^2 dismuinuye ligeramente, pero el valor del estadístico F aumenta. ¿Es conveniente/merece la pena en este caso eliminar una variable como zn?

Mi intuición (o sentido común) dice que sí, pero no estoy seguro . . .

]

A ver qué dice el criterio de Akaike:

```
#?step -> Choose a model by AIC in a Stepwise Algorithm
step(object = model_lstat2, direction = "both", trace = 1)
```

```
## Start:  AIC=1488.49
## medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio +
##     black + lstat + I(lstat^2)
##
##              Df Sum of Sq     RSS    AIC
## <none>                    9107.3 1488.5
## - zn          1      53.5  9160.8 1489.5
## - tax         1     178.8  9286.1 1496.3
## - chas        1     206.1  9313.4 1497.8
## - black       1     209.2  9316.5 1498.0
## - nox         1     324.4  9431.7 1504.2
## - rad         1     411.2  9518.5 1508.8
## - crim        1     456.0  9563.2 1511.2
## - ptratio     1     828.5  9935.8 1530.5
## - dis         1    1191.9 10299.2 1548.7
## - rm          1    1389.5 10496.7 1558.3
## - I(lstat^2)  1    1974.1 11081.4 1585.8
## - lstat       1    3635.6 12742.8 1656.5
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad +
##     tax + ptratio + black + lstat + I(lstat^2), data = Boston)
##
## Coefficients:
## (Intercept)         crim           zn         chas          nox           rm
##   42.124187    -0.149072     0.021281     2.589821   -13.534522     3.233174
##         dis          rad          tax      ptratio        black        lstat
##   -1.357892     0.271744    -0.009546    -0.790820     0.008174    -1.669567
##  I(lstat^2)
##    0.033055
```

Se mantienen todas las variables, incluida zn.

[

PREGUNTA 3

El "valor AIC" de este modelo es inferior. ¿Tiene sentido comparar dos modelos según el AIC?

]

Dejamos como modelo final para hacer el último punto

medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio + black + lstat + I(lstat^2)

Correlación entre lo que predice el modelo, y los datos reales:

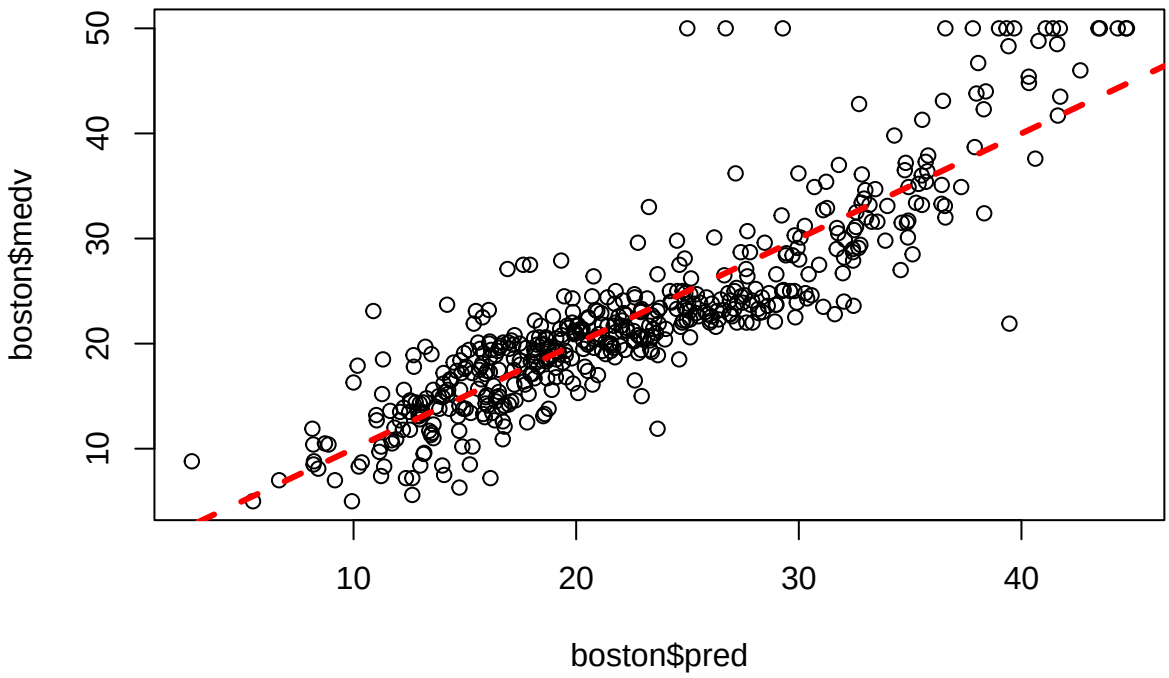
```
boston$pred <- predict(model_lstat2, boston)
```

```
cor(boston$pred, boston$medv)
```

```
## [1] 0.8870152
```

Es un valor bastante alto.

```
plot(boston$pred, boston$medv)
abline(a = 0, b = 1, col = "red", lwd = 3, lty = 2)
```



Los puntos están bastante cerca de la línea roja discontinua que marca donde las predicciones están muy cerca de los valores reales. Quizá el modelo no haga predicciones tan erróneas después de todo.

Predicción

En lugar de inventarme 5 registros, he pensado mejor combinar de manera aleatoria parejas de registros existentes. (Como si fueran viviendas que están entre otras 2 viviendas ya evaluadas).

```
boston1And10 <- (boston[1,]+boston[10,])/2
#boston1And10

boston15And40 <- (boston[15,]+boston[40,])/2
boston15And40$rad <- 3
#boston15And40

boston100And110 <- (boston[100,]+boston[110,])/2
boston100And110$rad <- 5
#boston100And110

boston200And220 <- (boston[200,]+boston[220,])/2
#boston200And220

boston300And500 <- (boston[300,]+boston[500,])/2
boston300And500$rad <- 5
#boston300And500
```

```
predict(model_lstat2, data.frame(crim=0.08818 , zn=15.25 , chas=0 , nox=0.531 , rm=6.2895 , dis=5.34105
```

```
##        1
## 22.98874
```

```
#predict(model_lstat2, data.frame(boston1And10))
```

```
predict(model_lstat2, data.frame(boston15And40))
```

```
##       15
## 25.51835
```

```
#predict(model_lstat2, data.frame(boston[100,]))  # -> 33.17257
#predict(model_lstat2, data.frame(boston[110,]))  # -> 18.12436
predict(model_lstat2, data.frame(boston100And110))
```

```
##     100
## 25.3321
```

```
#predict(model_lstat2, data.frame(boston[200,]))  # -> 30.69801
#predict(model_lstat2, data.frame(boston[220,]))  # -> 28.37799
predict(model_lstat2, data.frame(boston200And220))
```

```
##      200
## 29.24643
```

```
#predict(model_lstat2, data.frame(boston[300,]))  # -> 32.43072
#predict(model_lstat2, data.frame(boston[500,]))  # -> 17.2025
predict(model_lstat2, data.frame(boston300And500))
```

```
##      300
## 23.79381
```

Conclusión final:

Los resultados me parecen "razonables", a pesar de lo comentado (predictores con muchos valores atípicos, variables categóricas(?) .. predictor "racista*" ..).

*he encontrado críticas sobre este data set en internet. (Parece que este dataset también estaba en sckit-learn, y en la versión 1.2 lo quitaron. Al menos al hacer esta tarea con Python, me sale un mensaje avisando de esta circunstancia, como se puede ver en el repositorio de GitHub).