Trabajo 3: Programación

Victor Manuel Cerrato Molina y Gema Correa Fernández

27 de Mayo de 2017

Ajuste de Modelos Lineales

Antes de nada, debemos cargar las siguientes librerías que usaremos a lo largo de la práctica:

```r
# install.packages("e1071")
library(e1071)   # Para usar skewness

# install.packages("readr")
library(readr)   # Parar usar read.csv

# install.packages("caret")
library(caret)   # Para usar BoxCoxTrans
```

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

```
## Loading required package: ggplot2
```

Problema de Regresión

Apartado 1

Comprender el problema a resolver.

Para el **Problema de Regresión**, hemos escogido el data.frame Los Angeles Ozone. Estos datos registran el nivel de concentración atmosférica de ozono de ocho mediciones meteorológicas diarias, realizadas en la cuenca de Los Ángeles en 1976. En concreto, tenemos 330 casos completos. Nuestro objetivo es predecir la variable **ozone** a partir del resto de variables. Para ello tendremos que buscar el mejor modelo y encontrar que factores meteorológicos son los más influyentes en las mediciones de ozono.

Los atributos que tenemos en este data.frame son:

1. **ozone**: Máximo ozono diario.
2. **vh**: Altura (*Vandenberg 500 mb Height*).
3. **wind**: Velocidad del viento (mph).
4. **humidity**: Humedad (%).
5. **temp**: Temperatura (*Sandburg AFB Temperature*).
6. **ibh**: Altura de la base de inversión (*Inversion Base Height*).
7. **dpg**: Gradiente de presión de Daggot (*Daggot Pressure Gradient*).
8. **ibt**: Temperatura de la base de inversión (*Inversion Base Temperature*).
9. **vis**: Visibilidad (en millas).
10. **doy**: Día del año en que fue tomada la medición.

Ninguna variable podrá tomar valores *NA*. Además, nuestra variable de respuesta será **ozone**, como se nos dice en el enlace de la base de datos.

A continuación, vamos a leer nuestro data.frame:

```r
# Guardamos el data frame en una varible
LAozone <- read.csv("datos/LAozone.data")

# Visualizamos la cabecera del data frame
head(LAozone)
```

```
##   ozone   vh wind humidity temp  ibh dpg ibt vis doy
## 1     3 5710    4       28   40 2693 -25  87 250   3
## 2     5 5700    3       37   45  590 -24 128 100   4
## 3     5 5760    3       51   54 1450  25 139  60   5
## 4     6 5720    4       69   35 1568  15 121  60   6
## 5     4 5790    6       19   45 2631 -33 123 100   7
## 6     4 5790    3       25   55  554 -28 182 250   8
```

```r
# Mostramos la dimensión (330 filas y 10 columnas)
dim(LAozone)
```

```
## [1] 330  10
```

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 2

Los conjuntos de training, validación y test usados en su caso.

Como en la base de datos elegida solo se nos proporciona un archivo, somos nosotros quiénes tenemos que hacer la partición de los datos. Para ello, vamos a usar el mismo procedimiento usado en el pdf proporcionado por la profesora. Por tanto, partimos el data.frame en un 70% para el training y en un 30% para el test.

```r
# Establecemos una semilla por defecto para hacer el mismo tipo de partición
set.seed(1)

# Nos quedamos con los indices para el training
train = sample (nrow(LAozone), round(nrow(LAozone)*0.7))

# Reservamos por separado training y test
LAozone.train = LAozone[train,]
LAozone.test = LAozone[-train,]
```

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 3

Preprocesado de los datos: Falta de datos, categorización, normalización, reducción de dimensionalidad, etc.

Para el preprocesado de datos lo primero que hacemos, es borrar la última columna de nuestro data.frame, en concreto, la variable **doy**. Este atributo, guarda el día del año en que fue tomada la medición, por lo que la podemos considerar como un identificador, puesto que entre una fila y otra existe una diferencia de una unidad.

```r
# Borramos la última columna (doy) del training
LAozone.train = LAozone.train[-ncol(LAozone.train)]
```

```r
# Borramos la última columna (doy) del test
LAozone.test = LAozone.test[-ncol(LAozone.test)]
```

Ahora, vamos hacer una transformación con los atributos asimétricos, ya que son necesarios para la aplicación de algunos métodos de aprendizaje sensibles a distancias. Se consideran asimétricos cuando el valor *skewness* se aleja de 0.

$$skewness = \frac{\sum (x_i - mean(x))^3}{(n-1)v^3/2}$$

Primero, obtenemos el valor de asimetría de cada atributo:

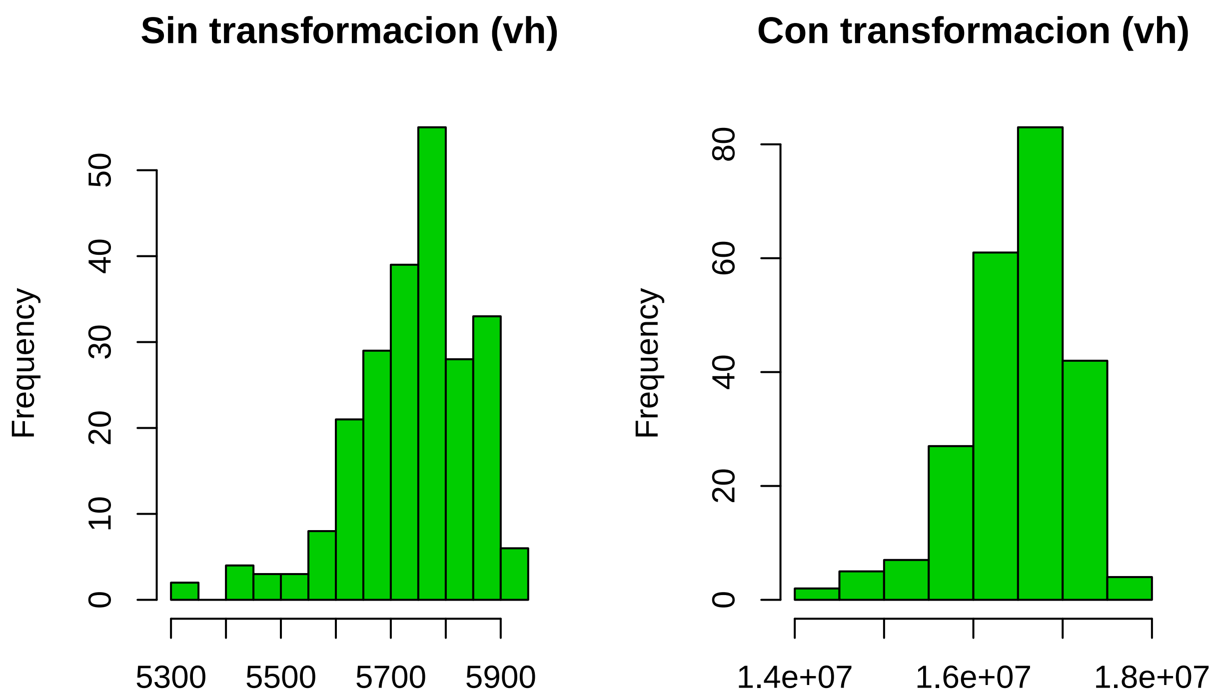
```r
# Obtenemos el valor de asimetría de los datos training
v_asimetria = apply(LAozone.train, 2, skewness)

# Ordenamos de mayor a menor los valores obtenidos
sort(abs(v_asimetria), decreasing = T)
```

```
##         vh   humidity      ozone        vis        dpg        ibh
## 0.99189851 0.87041616 0.86283756 0.81909512 0.34404777 0.20842831
##        ibt       wind       temp
## 0.18713411 0.09115408 0.07762523
```

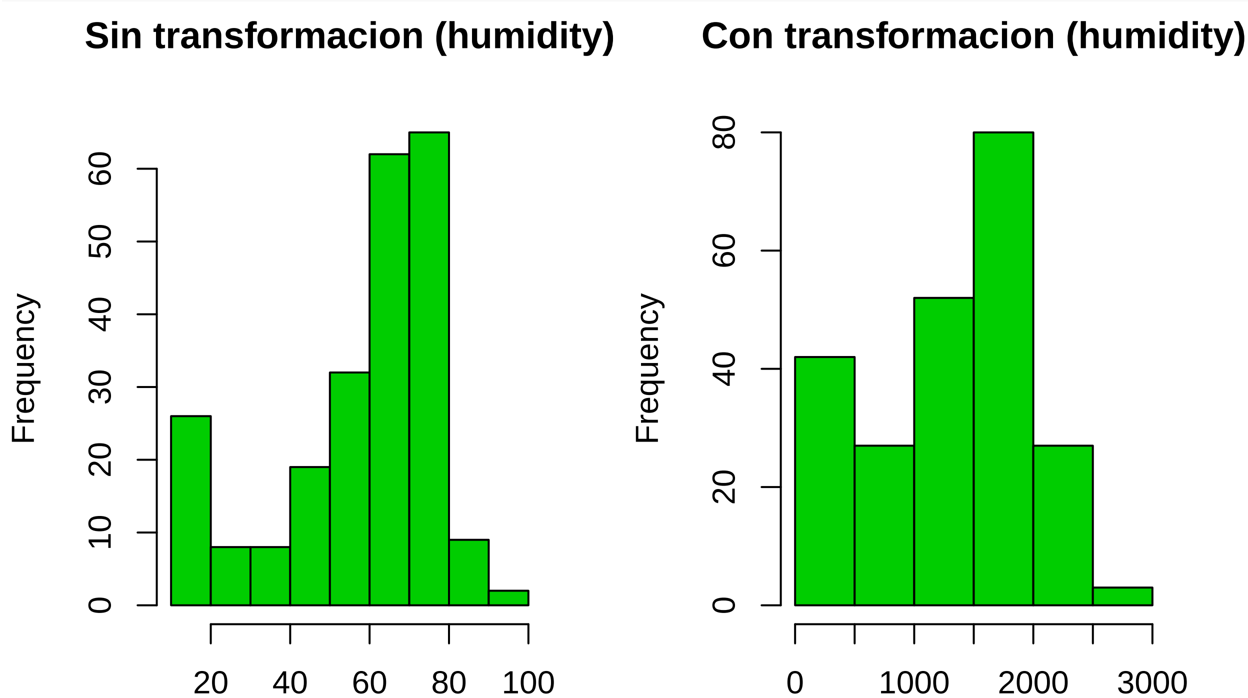
Segundo, transformaremos los datos a partir de un umbral de 0.8. Elegimos este umbral, ya que los valores más cercanos a 0 serán los más simétricos. Por lo tanto, realizaremos transformaciones para *vh*, *humidity* y *vis*.

***Nota**: No modificaremos ozone, ya que es nuestra variable de respuesta.*

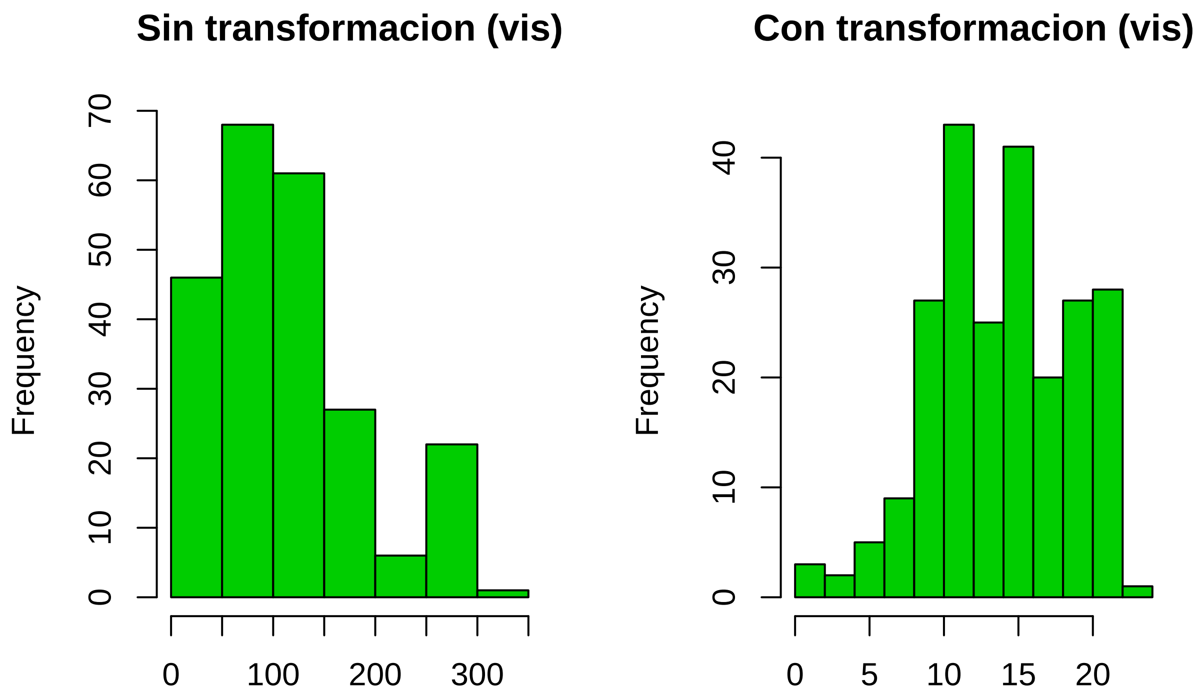
```r
# Transformación para la variable vh
# ---------------------------------------------------------------------------------
par(mfrow = c(1:2)) # Dividimos la región en dos partes
vh_trans = BoxCoxTrans(LAozone.train$vh)  # Obtenemos la transformación
# Comparamos los histogramas con transformación y sin transformación
hist(LAozone.train$vh, main = "Sin transformacion (vh)", xlab = "", col = 3)
hist(predict(vh_trans,LAozone.train$vh), main = "Con transformacion (vh)", xlab = "",
     col = 3)
```




```r
# Transformación para la variable humidity
# ---------------------------------------------------------------------------------
par(mfrow=c(1:2)) # Dividimos la región en dos partes
humidity_trans = BoxCoxTrans(LAozone.train$humidity) # Obtenemos la transformación
# Comparamos los histogramas con transformación y sin transformación
hist(LAozone.train$humidity, main = "Sin transformacion (humidity)", xlab = "",
     col = 3)
hist(predict(humidity_trans,LAozone.train$humidity),
     main = "Con transformacion (humidity)", xlab = "", col = 3)
```



```r
# Transformación para la variable vis
# ---------------------------------------------------------------------------------
par(mfrow=c(1:2)) # Dividimos la región en dos partes
vis_trans = BoxCoxTrans(LAozone.train$vis) # Obtenemos la transformación
# Comparamos los histogramas con transformación y sin transformación
hist(LAozone.train$vis, main = "Sin transformacion (vis)", xlab = "", col = 3)
hist(predict(vis_trans,LAozone.train$vis), main = "Con transformacion (vis)",
     xlab = "", col = 3)
```



Como se puede comprobar los valores obtenidos son más simétricos. Entonces, ya solo nos queda guardar esas transformaciones para el train.

```r
# Transformación vh (train)
LAozone.train$vh = predict(vh_trans,LAozone.train$vh)

# Transformación humidity (train)
LAozone.train$humidity = predict(humidity_trans,LAozone.train$humidity)

# Transformación vis (train)
LAozone.train$vis = predict(vis_trans,LAozone.train$vis)
```

También guardamos esas mismas transformaciones para el test.

```r
# Transformación vh (test)
LAozone.test$vh = predict(vh_trans,LAozone.test$vh)

# Transformación humidity (test)
LAozone.test$humidity = predict(humidity_trans,LAozone.test$humidity)

# Transformación vis (test)
LAozone.test$vis = predict(vis_trans,LAozone.test$vis)
```

Además, para el preprocesado de datos, podemos eliminar las variables con varianza 0 o muy próximas.

```r
nearZeroVar(LAozone.train)
```

```
## integer(0)
```

Se comprueba que no existe ninguna, entonces no se hace realiza ningún cambio.

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 4

Selección de clases de funciones a usar.

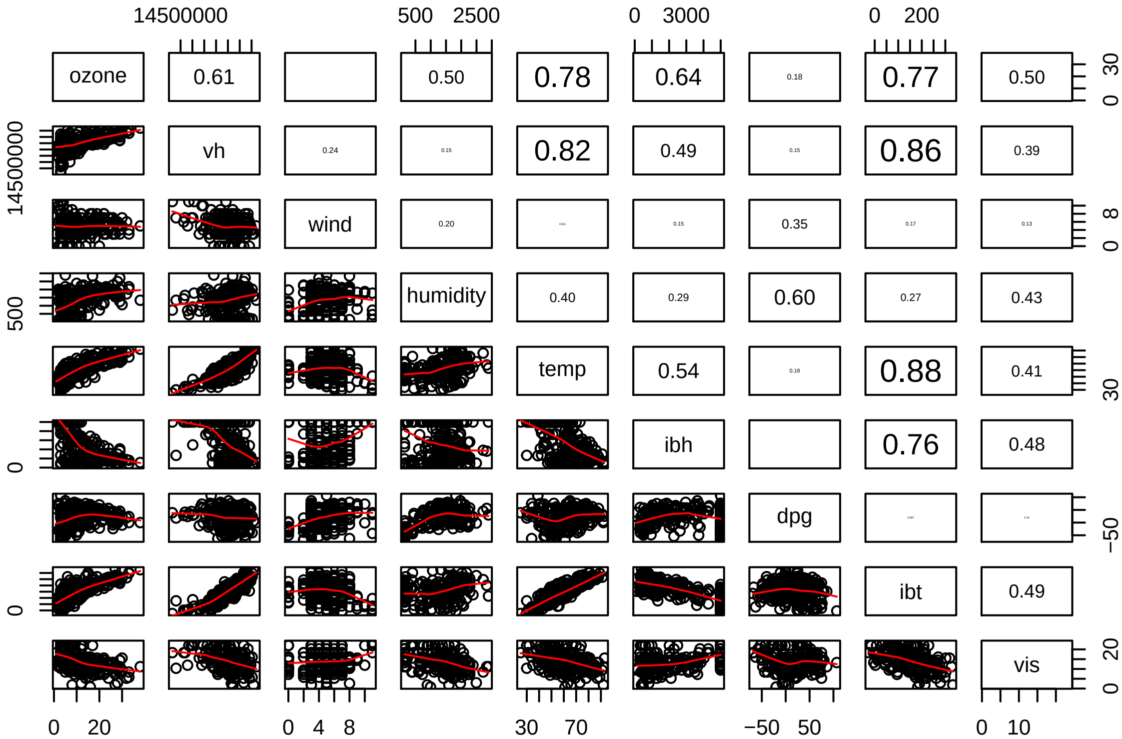
```r
attach(LAozone.train) # Para simplificar y prescindir del prefijo LAozone.train
```

Antes de nada, implementamos una función que tiene el mismo funcionamiento que *pairs*, sin embargo en el triángulo superior mostramos los coeficientes de correlación lineal, que nos dice como de relacionadas linealmente están las variables. Estos coeficientes se imprimen con un tamaño proporcional a su valor.

```r
panel.cor <- function(x, y, digits=2, prefix="", cex.cor) {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits=digits)[1]
  txt <- paste(prefix, txt, sep="")
  if(missing(cex.cor)) cex <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex * r)
}
```

Hacemos uso de la función y mostramos gráficamente todos con todos.

```r
pairs(LAozone.train, lower.panel=panel.smooth,upper.panel=panel.cor)
```

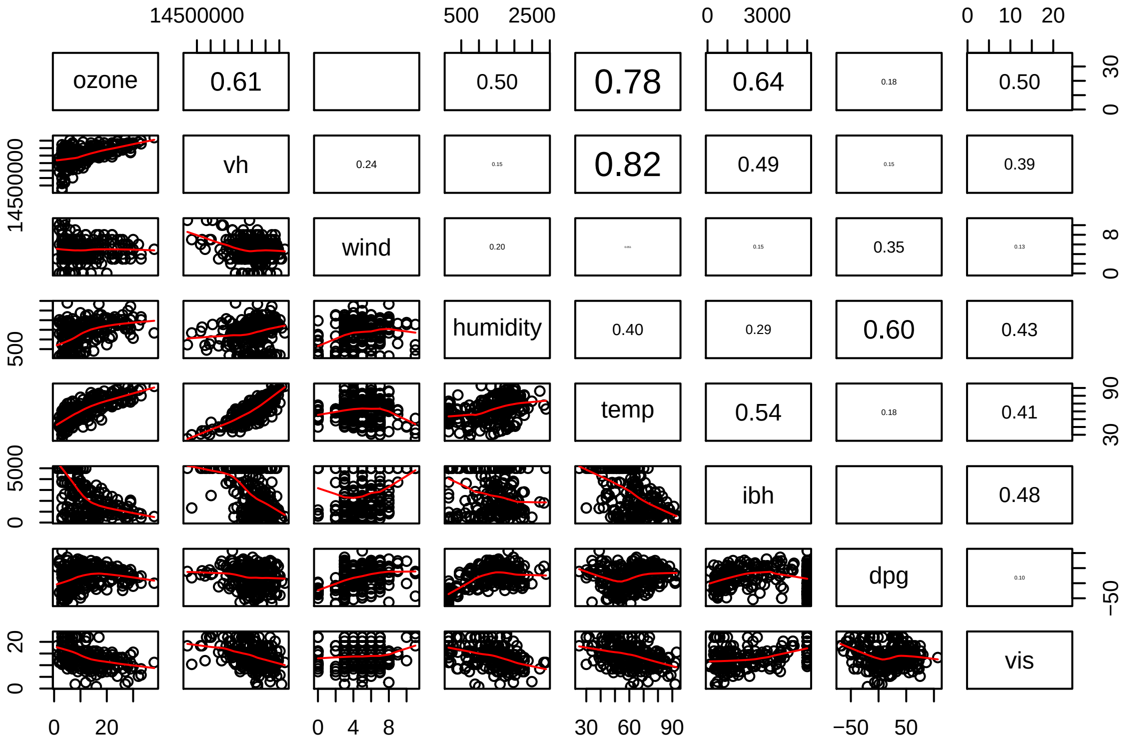


Observamos que *ibt* está muy relacionada linealemte con 3 variables, por consiguiente la quitamos.

```
# Quitamos ibt tanto para train como para test
LAozone.train = LAozone.train[,-8]
LAozone.test = LAozone.test[,-8]
```

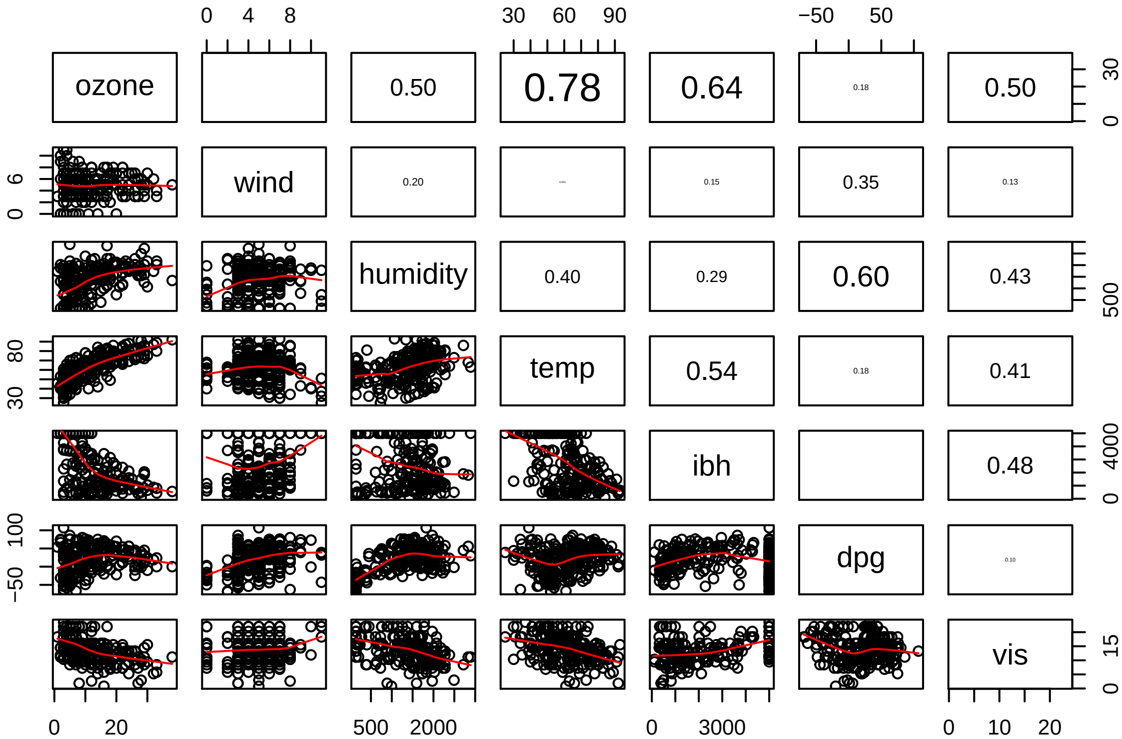
Volvemos a mostrar todos con todos, pero esta vez sin *ibt*:

```
pairs(LAozone.train, lower.panel=panel.smooth,upper.panel=panel.cor)
```



Con la anterior salida, tomamos la decisión de quitar *vh*, ya que es una variable muy relacionada linealmente con otras variables.

```
# Quitamos vh tanto para train como para test
LAozone.train = LAozone.train[,-2]
LAozone.test = LAozone.test[,-2]

# Y volvemos a mostrar el gráfico
pairs(LAozone.train, lower.panel=panel.smooth,upper.panel=panel.cor)
```



A partir de ahora, trabajaremos con los atributos que aparecen en el gráfico anterior.

Una vez arreglados nuestros datos, ya podemos pasar a la selección del modelo. Para empezar a seleccionar el mejor modelo, nos basaremos en E_{test}, debido a que éste se aproxima al error fuera de la muestra, E_{out}, mejor que E_{in}. Por ello creamos una función que nos calcule los errores E_{in} y E_{test}.

```
# Función que calcula los errores E_in y E_test
errores <- function(m){

  train = predict(m) # usa training
  test = predict(m, LAozone.test, type= "response") # usa test

  # Calculamos los errores
  etr = mean((train - LAozone.train[,1])^2)
  etst = mean((test - LAozone.test[,1])^2)

  list(Error_Train = etr, Error_Test = etst)
}
```

Una vez implementada esa función, comenzamos a elegir el mejor modelo. Primero usaremos **Regresión Lineal** sobre las variables.

Modelo 1: Predecimos *ozone* con todas las variables mediante Regresión Lineal.

```
# Creamos el modelo
m1 = lm(ozone ~ wind + humidity + temp + ibh + dpg + vis, data = LAozone.train)
summary(m1) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ wind + humidity + temp + ibh + dpg + vis,
```

```
##     data = LAozone.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9179 -2.8846 -0.1982  3.1803 13.6217
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.3822234  2.3476541  -2.293 0.022799 *
## wind         0.1981550  0.1425829   1.390 0.165984
## humidity     0.0023593  0.0006335   3.724 0.000248 ***
## temp         0.3049269  0.0252785  12.063  < 2e-16 ***
## ibh         -0.0010986  0.0002052  -5.354 2.12e-07 ***
## dpg         -0.0134387  0.0107581  -1.249 0.212907
## vis         -0.1636910  0.0768552  -2.130 0.034274 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.346 on 224 degrees of freedom
## Multiple R-squared:  0.7138, Adjusted R-squared:  0.7061
## F-statistic: 93.11 on 6 and 224 DF,  p-value: < 2.2e-16
```

```r
errores(m1) # Obtenemos los errores
```

```
## $Error_Train
## [1] 18.31556
##
## $Error_Test
## [1] 26.12091
```

Como se puede ver en la salida del *summary()*, los coeficientes más importantes son *humidity*, *temp* y *vh*, ya que tienen 3 estrellas. La primera comparación que haremos será de *ozone* con las tres mejores, sin niguna transformación.

Modelo 2: Predecimos *ozone* con la variable *temp* mediante Regresión Lineal.

```r
# Creamos el modelo
m2 = lm(ozone ~ temp, data=LAozone.train)
summary(m2) # Realizamos un análisis del modelo
```

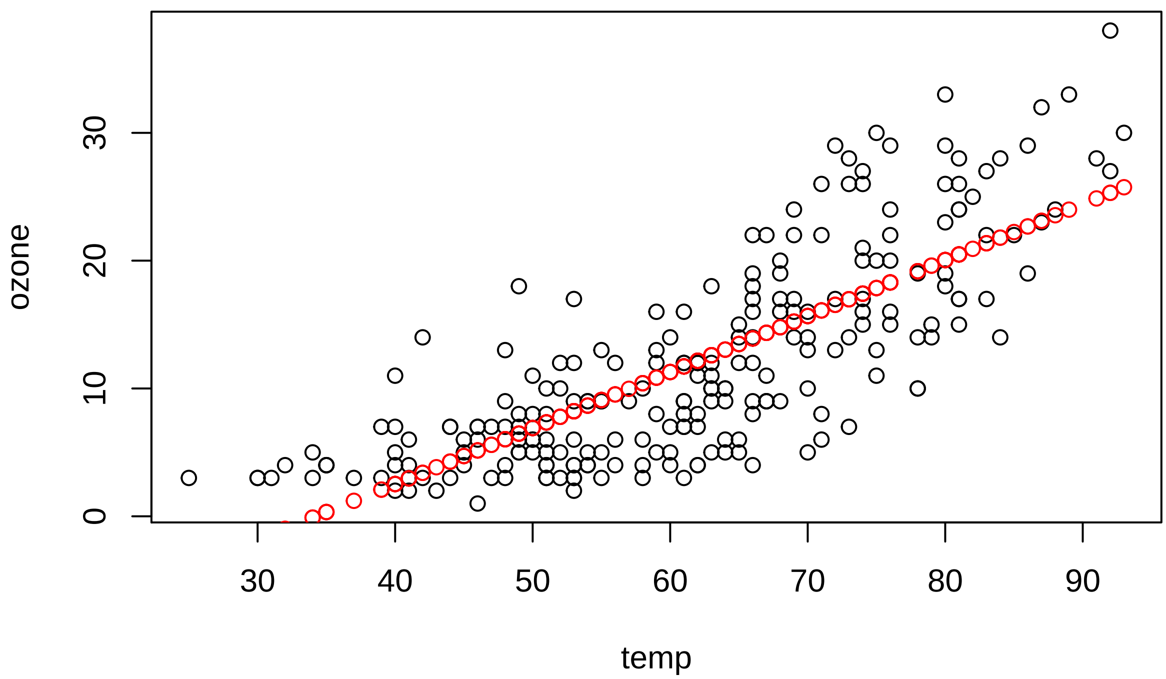
```
##
## Call:
## lm(formula = ozone ~ temp, data = LAozone.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.6682  -3.6682  -0.1634   3.3222  12.9508
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14.9987     1.4727  -10.18   <2e-16 ***
## temp          0.4381     0.0233   18.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.037 on 229 degrees of freedom
```

```
## Multiple R-squared:  0.607,  Adjusted R-squared:  0.6053
## F-statistic: 353.7 on 1 and 229 DF,  p-value: < 2.2e-16
```

```r
errores(m2) # Obtenemos los errores
```

```
## $Error_Train
## [1] 25.15115
##
## $Error_Test
## [1] 24.88593
```

```r
# Vamos a mostrar gráficamente el modelo
plot(ozone ~ temp, data = LAozone.train)
w = m2$coefficients
x = matrix(rep(1, length(temp)), nrow = length(temp))
x = cbind (x, temp)
y = apply(x, 1, function(vec) w %*% vec)
points(temp, y, col=2)
```



Como vemos el modelo se ajusta bastante bien, por lo que no haría falta el uso de una transformación, pero vamos a intentar mejorarlo.

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Modelo 3: Predecimos *ozone* con la variable *ibh* mediante Regresión Lineal.

```r
# Creamos el modelo
m3 = lm(ozone ~ ibh, data=LAozone.train)
# Realizamos un análisis del modelo
summary(m3)
```

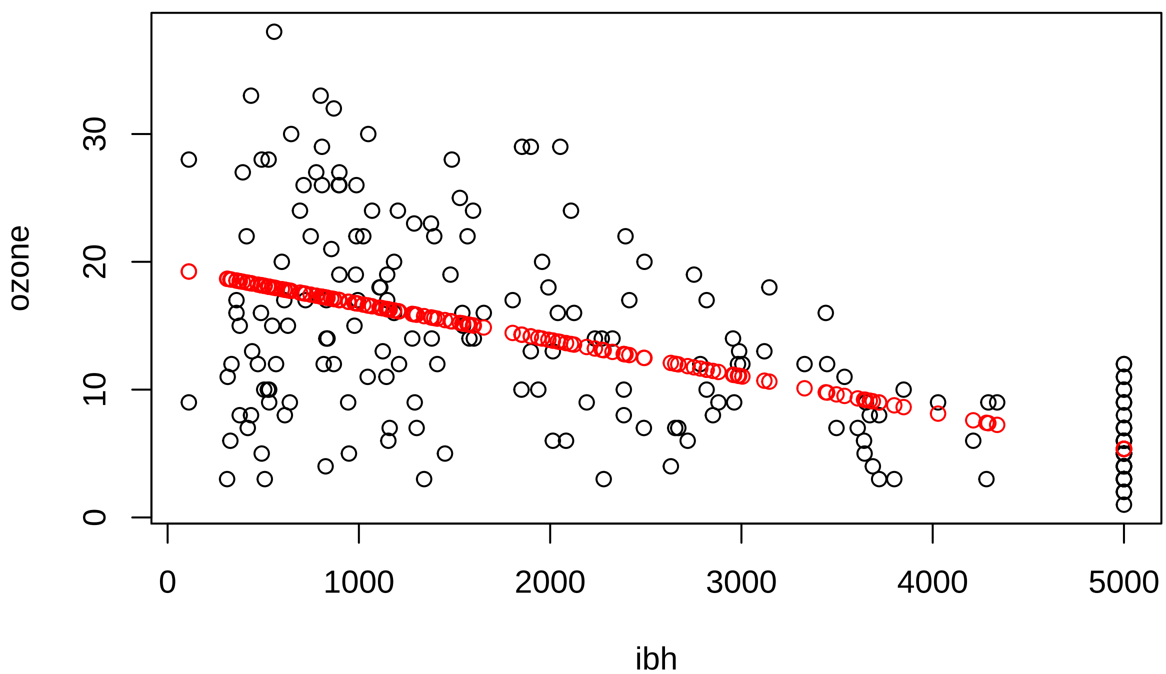
```
##
## Call:
```

```
## lm(formula = ozone ~ ibh, data = LAozone.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6744  -3.3627  -0.3627   3.2249  20.0240
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.5572652  0.7298592    26.8   <2e-16 ***
## ibh         -0.0028389  0.0002271   -12.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.194 on 229 degrees of freedom
## Multiple R-squared:  0.4057, Adjusted R-squared:  0.4031
## F-statistic: 156.3 on 1 and 229 DF,  p-value: < 2.2e-16
```

```r
# Obtenemos los errores
errores(m3)
```

```
## $Error_Train
## [1] 38.0332
##
## $Error_Test
## [1] 51.70113
```

```r
# Vamos a mostrar gráficamente el modelo
plot(ozone ~ ibh, data=LAozone.train)
w = m3$coefficients
x = matrix(rep(1, length(ibh)), nrow= length(ibh))
x = cbind (x, ibh)
y = apply(x, 1, function(vec) w %*% vec)
points(ibh, y, col=2)
```



```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Modelo 4: Predecimos *ozone* con *humidity* mediante Regresión Lineal.

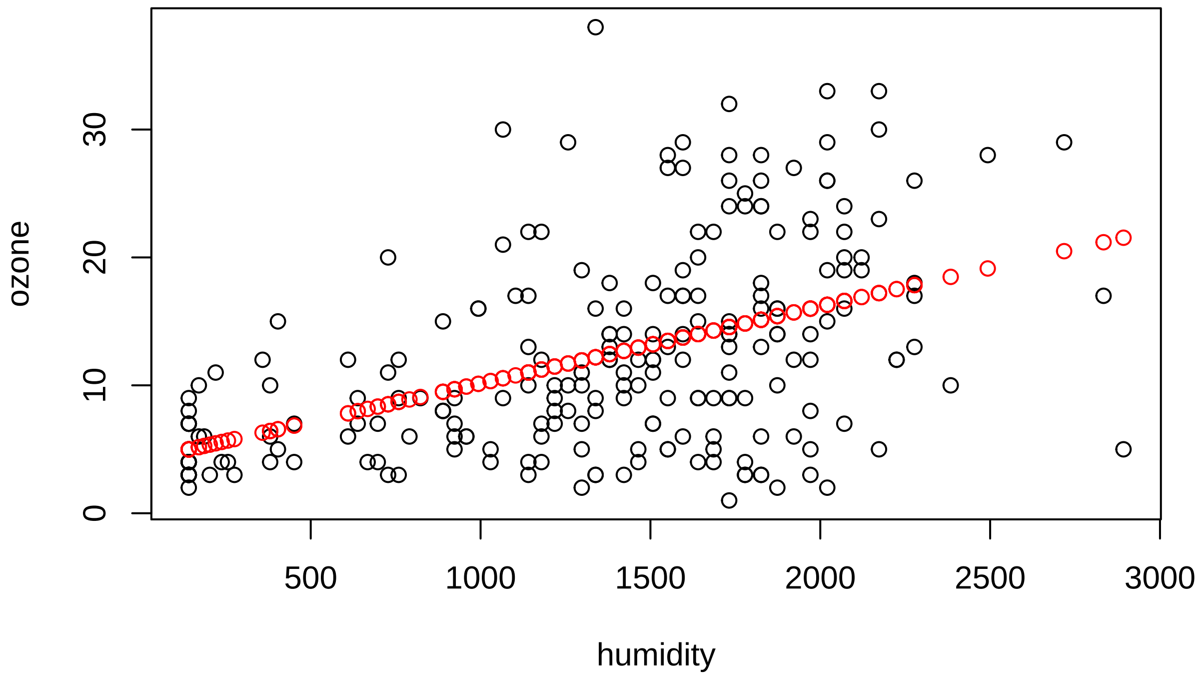
```r
# Creamos el modelo
m4 = lm(ozone ~ humidity, data=LAozone.train)
summary(m4) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ humidity, data = LAozone.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.541  -4.190  -0.990   3.466  25.807
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.1417992  1.0114407   4.095 5.86e-05 ***
## humidity    0.0060150  0.0006913   8.701 6.54e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.965 on 229 degrees of freedom
## Multiple R-squared:  0.2485, Adjusted R-squared:  0.2452
## F-statistic: 75.71 on 1 and 229 DF,  p-value: 6.536e-16
```

```r
errores(m4) # Obtenemos los errores
```

```
## $Error_Train
## [1] 48.09558
##
## $Error_Test
## [1] 59.62795
```

```r
# Vamos a mostrar gráficamente el modelo
plot(ozone ~ humidity, data=LAozone.train)
w = m4$coefficients
x = matrix(rep(1, length(humidity)), nrow = length(humidity))
x = cbind (x, humidity)
y = apply(x, 1, function(vec) w %*% vec)
points(humidity, y, col=2)
```



El atributo que mejor explica por sí sola la variable *ozone* es *temp* (Modelo 2) y con amplia diferencia atendiendo a los errores que produce la regresión lineal.

```
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

A continuación, vamos a probar transformaciones en las variables que obtuvimos peores resultados, ya que sospechamos que puede que no haya linealidad en los datos para explicar ozone, puesto que las soluciones no fueron satisfactorias mediante regresión lineal.

Modelo con transformación: Predecimos *ozone* mediante Regresión Lineal, usando la función *poly()* en la variable *dpg* para ver si se puede explicar ozone mediante dicha trasformación.

```
# Creamos el modelo
m = lm(ozone ~ poly(dpg,2) , data = LAozone.train)
summary(m) # Realizamos un análisis del modelo
```

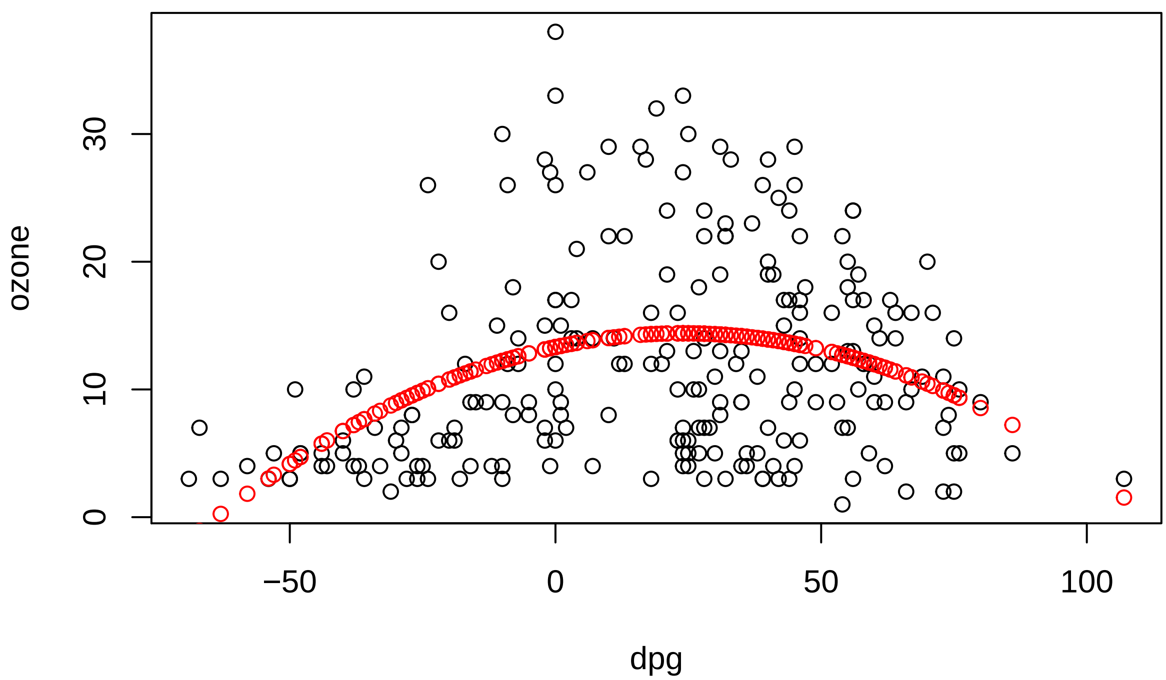
```
##
## Call:
## lm(formula = ozone ~ poly(dpg, 2), data = LAozone.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.718  -5.505  -1.231   4.589  24.681
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     11.9870     0.4901  24.459  < 2e-16 ***
## poly(dpg, 2)1   22.0508     7.4487   2.960   0.0034 **
## poly(dpg, 2)2 -40.5779     7.4487  -5.448 1.32e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 7.449 on 228 degrees of freedom
## Multiple R-squared:  0.1443, Adjusted R-squared:  0.1368
## F-statistic: 19.22 on 2 and 228 DF,  p-value: 1.933e-08
```

```
errores(m) # Obtenemos los errores
```

```
## $Error_Train
## [1] 54.76259
##
## $Error_Test
## [1] 53.87958
```

```
# Vamos a mostrar gráficamente el modelo
plot(ozone ~ dpg, data=LAozone.train)
w = m$coefficients
x = matrix(rep(1, length(dpg)), nrow = length(dpg))
x = cbind (x, poly(dpg,2))
y= apply(x, 1, function(vec) w %*% vec)
points(dpg, y, col=2)
```



Este modelo con una transformación no lineal en los datos mejora el modelo lineal pero queda lejos del modelo lineal con *temp* (Modelo 2) en lo que respecta a errores.

```
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Modelo con transformación: Predecimos *ozone* mediante Regresión Lineal usando la función potencia cúbica en la variable *vis* para ver si se puede explicar ozone mediante dicha trasformación.

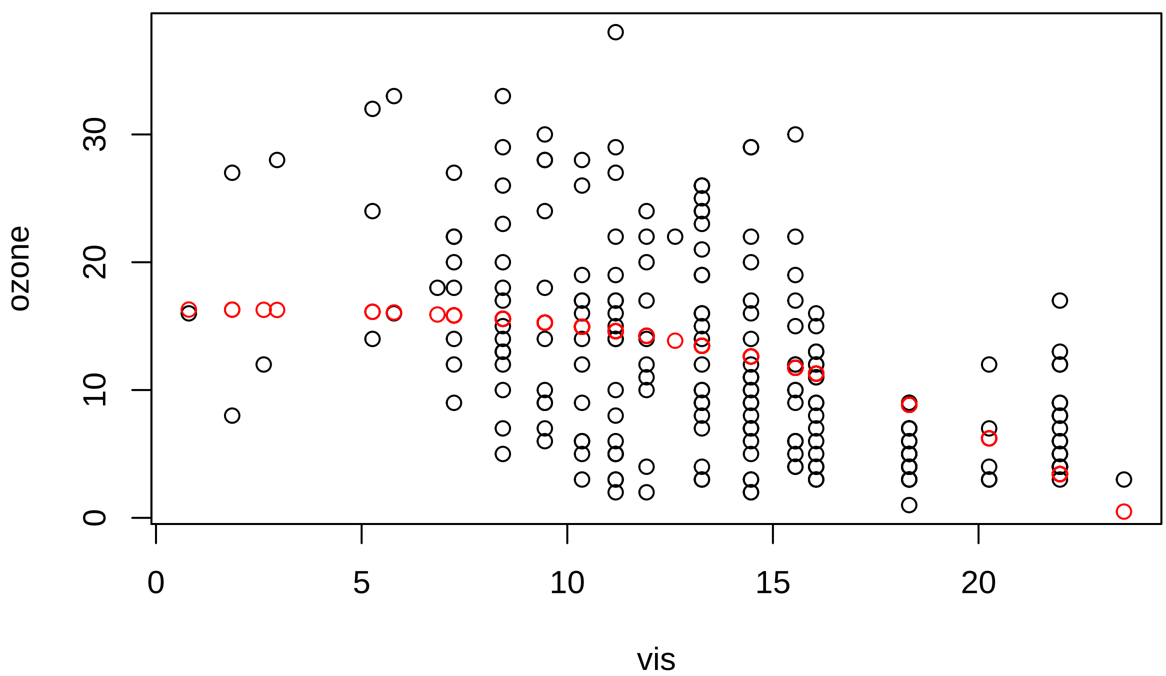
```
# Creamos el modelo
m = lm(ozone ~ I(vis^3), data = LAozone.train)
summary(m) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ I(vis^3), data = LAozone.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.606  -5.279  -1.275   4.267  23.394
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.2994590  0.7138188  22.834  < 2e-16 ***
## I(vis^3)    -0.0012126  0.0001517  -7.994 6.42e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.104 on 229 degrees of freedom
## Multiple R-squared:  0.2182, Adjusted R-squared:  0.2147
## F-statistic:  63.9 on 1 and 229 DF,  p-value: 6.425e-14
```

```r
errores(m) # Obtenemos los errores
```

```
## $Error_Train
## [1] 50.03443
##
## $Error_Test
## [1] 58.12606
```

```r
# Vamos a mostrar gráficamente el modelo
plot(ozone ~ vis, data=LAozone.train)
w = m$coefficients
x = matrix(rep(1, length(vis)), nrow= length(vis))
x = cbind (x, I(vis^3))
y = apply(x, 1, function(vec) w %*% vec)
points(vis, y, col=2)
```



Hemos conseguido mejores resultados que con el modelo lineal pero seguimos sin obtener una mejora trascendental.

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Modelo sin transformación: Predecimos *ozone* mediante Regresión Lineal con la variable *wind*.

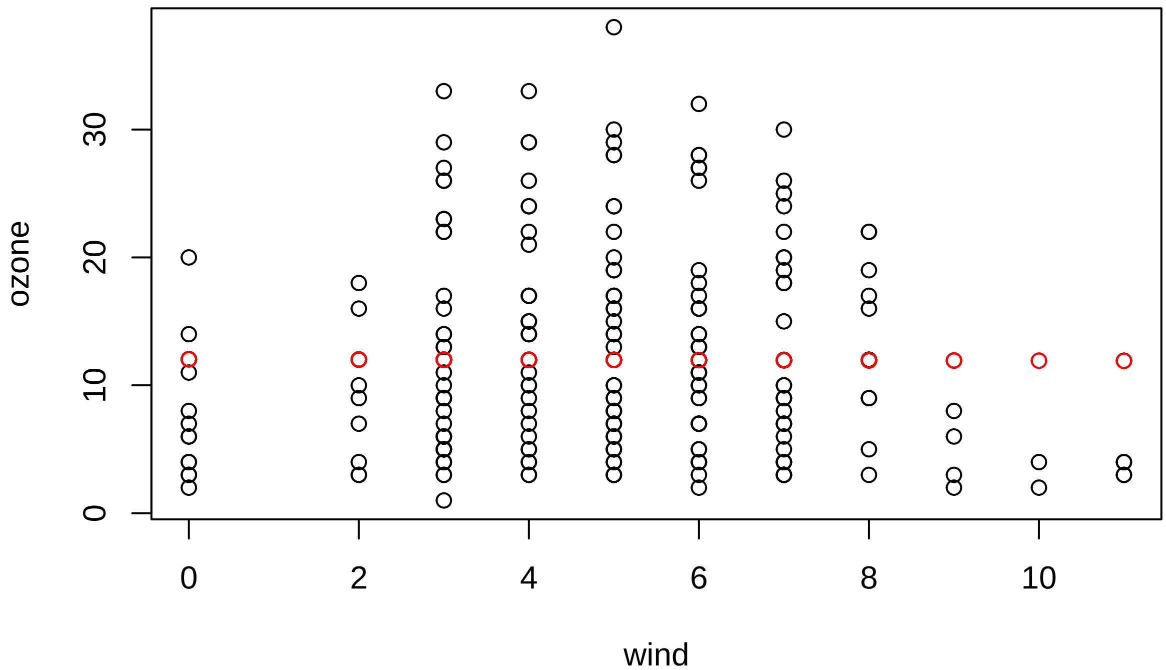
```r
# Creamos el modelo
m = lm(ozone ~ wind, data = LAozone.train)
summary(m) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ wind, data = LAozone.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.008  -6.975  -1.997   5.003  26.014
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.04119    1.29028   9.332   <2e-16 ***
## wind        -0.01108    0.24082  -0.046    0.963
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.035 on 229 degrees of freedom
## Multiple R-squared:  9.251e-06,  Adjusted R-squared:  -0.004358
## F-statistic: 0.002118 on 1 and 229 DF,  p-value: 0.9633
```

```r
errores(m) # Obtenemos los errores
```

```
## $Error_Train
## [1] 63.99491
##
## $Error_Test
## [1] 64.09817
```

```r
# Vamos a mostrar gráficamente el modelo
plot(ozone ~ wind, data=LAozone.train)
w = m$coefficients
x = matrix(rep(1, length(wind)), nrow= length(wind))
x = cbind (x, wind)
y = apply(x, 1, function(vec) w %*% vec)
points(wind, y, col=2)
```



En este caso, hicimos transformaciones, pero ninguna tenía mejoría apreciable respecto al modelo lineal, por lo que presentamos el modelo lineal en el que se ve que la predicción no es buena fijandonos en los errores.

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Combinaciones de variables

Ahora vamos a usar varias variables sin y con transformaciones no lineales para encontrar una predicción de *ozone* más ajustada.

Modelo 5: Regresión Lineal sin transformación con *temp* y *humidity*.

```r
# Creamos el modelo
m5 = lm(ozone ~ temp + humidity, data=LAozone.train)
summary(m5) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ temp + humidity, data = LAozone.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.8602  -2.9605  -0.5487   3.0230  14.1262
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.546e+01  1.395e+00 -11.080  < 2e-16 ***
## temp         3.879e-01  2.398e-02  16.175  < 2e-16 ***
## humidity     2.726e-03  5.147e-04   5.296 2.78e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 4.763 on 228 degrees of freedom
## Multiple R-squared:   0.65,  Adjusted R-squared:  0.647
## F-statistic: 211.7 on 2 and 228 DF,  p-value: < 2.2e-16
```

```r
errores(m5) # Obtenemos los errores
```

```
## $Error_Train
## [1] 22.3961
##
## $Error_Test
## [1] 24.35143
```

Como se comprueba, el mejor modelo lineal que tenía era con *temp*, al añadirle una combinación con la variable *humidity*, sigue obteniendo buenos resultados.

Modelo 6: Regresión Lineal sin transformación con *temp* e *ibh*.

```r
# Creamos el modelo
m6 = lm(ozone ~ temp + ibh , data=LAozone.train)
summary(m6) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ temp + ibh, data = LAozone.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.3871  -2.9861  -0.2688   3.0151  12.6656
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.7029395  1.9390033  -2.941  0.00361 **
## temp         0.3455207  0.0254574  13.573  < 2e-16 ***
## ibh         -0.0013475  0.0002018  -6.678 1.83e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.617 on 228 degrees of freedom
## Multiple R-squared:  0.6713, Adjusted R-squared:  0.6684
## F-statistic: 232.8 on 2 and 228 DF,  p-value: < 2.2e-16
```

```r
errores(m6) # Obtenemos los errores
```

```
## $Error_Train
## [1] 21.03662
##
## $Error_Test
## [1] 26.30305
```

Modelo 7: Regresión Lineal sin transformación con *ibh* y *humidity*.

```r
# Creamos el modelo
m7 = lm(ozone ~ ibh + humidity , data=LAozone.train)
summary(m7) # Realizamos un análisis del modelo
```

```
##
## Call:
```

```
## lm(formula = ozone ~ ibh + humidity, data = LAozone.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.4787  -3.8669  -0.6324   3.5761  20.8267
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.0099922  1.1441266  11.371  < 2e-16 ***
## ibh         -0.0023918  0.0002159 -11.077  < 2e-16 ***
## humidity     0.0041057  0.0005846   7.023 2.48e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.629 on 228 degrees of freedom
## Multiple R-squared:  0.5114, Adjusted R-squared:  0.5071
## F-statistic: 119.3 on 2 and 228 DF,  p-value: < 2.2e-16
```

```r
errores(m7) # Obtenemos los errores
```

```
## $Error_Train
## [1] 31.26913
##
## $Error_Test
## [1] 45.11003
```

La combinación por sí solas, de *ibh* y *humidity*, no nos aportan mejorías.

Modelo 8: Regresión Lineal sin transformación con *temp*, *ibh* y *humidity*.

```r
# Creamos el modelo
m8 = lm(ozone ~ ibh + temp + humidity , data=LAozone.train)
summary(m8) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ ibh + temp + humidity, data = LAozone.train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.109 -3.130 -0.205  3.104 13.933
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.8068614  1.8553979  -3.669 0.000304 ***
## ibh         -0.0012468  0.0001928  -6.468 6.03e-10 ***
## temp         0.3081184  0.0252981  12.180  < 2e-16 ***
## humidity     0.0024070  0.0004765   5.051 9.02e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.387 on 227 degrees of freedom
## Multiple R-squared:  0.7045, Adjusted R-squared:  0.7006
## F-statistic: 180.4 on 3 and 227 DF,  p-value: < 2.2e-16
```

```r
errores(m8) # Obtenemos los errores
```

```
## $Error_Train
## [1] 18.91107
##
## $Error_Test
## [1] 24.69152
```

Modelo 9: Regresión Lineal con transformación en *dpg* y sin transformación para *temp*.

```r
# Creamos el modelo
m9 = lm(ozone ~ poly(dpg,2) + temp, data=LAozone.train)
summary(m9) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ poly(dpg, 2) + temp, data = LAozone.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4430  -3.2551  -0.2904   2.6612  12.3309
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -13.46546    1.43222  -9.402  < 2e-16 ***
## poly(dpg, 2)1   6.21636    4.83729   1.285      0.2
## poly(dpg, 2)2 -25.69405    4.82818  -5.322 2.46e-07 ***
## temp            0.41321    0.02269  18.212  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.759 on 227 degrees of freedom
## Multiple R-squared:  0.6523, Adjusted R-squared:  0.6477
## F-statistic:   142 on 3 and 227 DF,  p-value: < 2.2e-16
```

```r
errores(m9) # Obtenemos los errores
```

```
## $Error_Train
## [1] 22.25133
##
## $Error_Test
## [1] 23.69307
```

Como se ve en la salida del *summary()* , el peso que está asociado al primer término del polinomio ortogonal aplicado a *dpg* no es bueno, pero si lo quitamos, el E_{test} aumenta, lo cual no es deseable.

Modelo 10: Regresión Lineal con transformaciones en *temp* e *ibh* y sin transformación para *humidity*.

```r
# Creamos el modelo
m10 = lm(ozone ~ I(temp^2) + ibh + humidity + 0, data=LAozone.train)
summary(m10) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ I(temp^2) + ibh + humidity + 0, data = LAozone.train)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.4924 -2.7943 -0.0983  3.0184 11.1803
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## I(temp^2)  0.0028595  0.0001403  20.379  < 2e-16 ***
## ibh       -0.0009943  0.0001160  -8.569 1.59e-15 ***
## humidity   0.0023947  0.0004314   5.551 7.87e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.195 on 228 degrees of freedom
## Multiple R-squared:  0.9164, Adjusted R-squared:  0.9153
## F-statistic: 832.8 on 3 and 228 DF,  p-value: < 2.2e-16
```

```
errores(m10) # Obtenemos los errores
```

```
## $Error_Train
## [1] 17.36707
##
## $Error_Test
## [1] 23.08043
```

En este caso si quitamos el término independiente que añade *lm*, automáticamente tenemos una mejora en el error que produce el ajuste del modelo.

Modelo 11: Regresión Lineal sin trasformaciones usando *ibh*, *dpg* y *temp*.

```
# Creamos el modelo
m11 = lm(ozone ~ ibh + temp + dpg , data=LAozone.train)
summary(m11) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ ibh + temp + dpg, data = LAozone.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.3684  -3.1235  -0.3588   3.0587  13.2338
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.221990   1.941243  -2.690  0.00768 **
## ibh         -0.001397   0.000202  -6.916 4.67e-11 ***
## temp         0.334417   0.025892  12.916  < 2e-16 ***
## dpg          0.017376   0.008677   2.003  0.04642 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.586 on 227 degrees of freedom
## Multiple R-squared:  0.677,  Adjusted R-squared:  0.6727
## F-statistic: 158.6 on 3 and 227 DF,  p-value: < 2.2e-16
```

```
errores(m11) # Obtenemos los errores
```

```
## $Error_Train
## [1] 20.67144
##
## $Error_Test
## [1] 24.91655
```

Modelo 12: Regresión Lineal con transformación con la función potencia cuadrado en *temp* e *ibh* y la función *atan()* en *humidity*.

```
# Creamos el modelo
m12 = lm(ozone ~ I(ibh^2) +  atan(humidity) + I(temp^2), data=LAozone.train)
summary(m12) # Realizamos un análisis del modelo
```

```
##
## Call:
## lm(formula = ozone ~ I(ibh^2) + atan(humidity) + I(temp^2), data = LAozone.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3534 -2.9655 -0.2032  2.4472 11.7147
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -8.546e+02  2.251e+02  -3.796 0.000189 ***
## I(ibh^2)       -1.818e-07  3.370e-08  -5.395 1.72e-07 ***
## atan(humidity)  5.461e+02  1.435e+02   3.807 0.000181 ***
## I(temp^2)       2.880e-03  1.943e-04  14.821  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.28 on 227 degrees of freedom
## Multiple R-squared:  0.7187, Adjusted R-squared:  0.7149
## F-statistic: 193.3 on 3 and 227 DF,  p-value: < 2.2e-16
```

```
errores(m12) # Obtenemos los errores
```

```
## $Error_Train
## [1] 18.00435
##
## $Error_Test
## [1] 22.40201
```

En este modelo, obtenemos una pequeña mejoría, en comparación al modelo 2, donde usábamos *temp* sin ninguna transformación.

Después de haber probado varios modelos, antes de decantarnos por uno de ellos vamos a aplicar regularización sobre los dos mejores (Modelo 10 y Modelo 12) y si obtenemos mejores resultados será necesario aplicar tal regularación.

```
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 5

Discutir la necesidad de regularización y en su caso la función usada

Para utilizar regularización usamos una función que hemos definido nosotros (usada en la práctica 2) para aplicar el modelo de Regresión Rineal con Weight Decay.

```
Regress_LinWD <- function(datos,label,landa){

  # Descomponemos los datos en UDV^T
  descom=svd(datos)

  # Calculamos una pseudo inversa especial para el caso de deacaimiento de pesos
  pseudo_inv_WD=
    descom$v %*% diag(1/(descom$d**2+rep(landa,length(descom$d))))%*%t(diag(descom$d))%*%t(descom$u)

  # Aplicamos la formula que nos devuelve un vector columna
  w = pseudo_inv_WD%*%label

  # Extraemos w en forma de vector fila obteniendo los pesos de la solucion
  t(w)[1,]
}
```

Como hemos comentado antes, vamos a usar regularización en los dos mejores modelos que hemos considerado del Apartado 4.

Modelo 1: Regresión Lineal con WD usando las variables trasformadas siguiente (anteriormente modelo 12):

- *ibh* con la función potencial al cuadrado.
- *humidity* con la función arcotangente.
- *temp* con la función potencial al cuadrado.

```
datos = cbind(I(ibh^2) , atan(humidity) , I(temp^2),1)
datos.test = cbind(I(LAozone.test$ibh^2) , atan(LAozone.test$humidity) ,
                   I(LAozone.test$temp^2),1)

# Establecemos por defecto un e_test alto ya que nunca llegaremos a ese valor
Etest = 100

# Calculamos el error para varias lambdas
for(landa in seq(3.7*10**8,4.2*10**8, length.out=10)){
  w = Regress_LinWD(datos,ozone,landa)
  EtestAc = mean((LAozone.test$ozone - datos.test%*%w)^2)

  # Nos quedamos con los mejores resultados
  if(EtestAc < Etest) {
    Etest = EtestAc
    wmej = w
    landamej = landa
  }

  cat(" Landa: ",landa," Etest: ", EtestAc,"\n")
}
```

```
##  Landa:  3.7e+08  Etest:  21.82045
##  Landa:  375555556  Etest:  21.81849
##  Landa:  381111111  Etest:  21.81721
##  Landa:  386666667  Etest:  21.81659
##  Landa:  392222222  Etest:  21.81664
```

```
##  Landa:  397777778  Etest:  21.81735
##  Landa:  403333333  Etest:  21.8187
##  Landa:  408888889  Etest:  21.8207
##  Landa:  414444444  Etest:  21.82334
##  Landa:  4.2e+08  Etest:  21.82662
```

```
cat("El mejor landa es ",landamej, " con un Etest de ",Etest,
    " que da la solución: w=[",wmej,"]\n")
```

```
## El mejor landa es  386666667  con un Etest de  21.81659  que da la solución: w=[ -1.097776e-07 9.845
```

Tras hacer regularización un un intervalo amplio de landas (desde 0.005 hasta 10^9), hemos identificado el valor óptimo de landa para que el error en el conjunto de test sea mínimo.

```
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Modelo 2: Regresión Lineal con WD usando las variables trasformadas y no trasformadas siguientes (anteriormente modelo 10):

- *ibh* sin trasformación.
- *humidity* sin trasformación.
- *temp* con la función potencial al cuadrado.
- Sin término independiente.

```
datos = cbind(I(temp^2) , ibh, humidity)
datos.test = cbind(I(LAozone.test$temp^2) , LAozone.test$ibh , LAozone.test$humidity)

# Establecemos por defecto un e_test alto ya que nunca llegaremos a ese valor
Etest = 100

# Calculamos el error para varias lambdas
for(landa in seq(2.6*10**8,2.8*10**8,length.out=10)){
  w = Regress_LinWD(datos,ozone,landa)
  EtestAc = mean((LAozone.test$ozone - datos.test%*%w)^2)

  # Nos quedamos con los mejores resultados
  if(EtestAc < Etest) {
    Etest = EtestAc
    wmej = w
    landamej = landa
  }

  cat(" Landa: ",landa," Etest: ", EtestAc,"\n")
}
```

```
##  Landa:  2.6e+08  Etest:  21.34591
##  Landa:  262222222  Etest:  21.34512
##  Landa:  264444444  Etest:  21.34447
##  Landa:  266666667  Etest:  21.34396
##  Landa:  268888889  Etest:  21.34358
##  Landa:  271111111  Etest:  21.34335
##  Landa:  273333333  Etest:  21.34325
##  Landa:  275555556  Etest:  21.34329
##  Landa:  277777778  Etest:  21.34346
```

```
##  Landa:  2.8e+08  Etest:  21.34376
```

```
cat("El mejor landa es ",landamej," con un Etest de ", Etest,
    " que da la solución: w=[",wmej,"]\n")
```

```
## El mejor landa es  273333333  con un Etest de  21.34325  que da la solución: w=[ 0.002842689 -0.00053
```

Tras hacer regularización un un intervalo amplio de landas(desde 0.005 hasta 10^9), hemos identificado el valor óptimo de landa para que el error en el conjunto de test sea mínimo.

```
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

El **modelo 2** da mejores resultados que el modelo uno, auque sin regularización esto no era así por lo que se justifica su uso.

Apartado 6

Definir los modelos a usar y estimar sus parámetros e hiperparámetros

Vimos en el apartado anterior que usar regularización es buena idea ya que podemos reducir el error de los modelos. El λ más óptimo para el modelo dos se encuentra alrededor de $2.73 \cdot 10^8$.

Apartado 7

Selección y ajuste modelo final

El modelo que elegimos es el obtenido mediante Regresión Lineal con WD con los siguientes parámetros y variables:

- *ibh* sin trasformación.
- *humidity* sin trasformación.
- *temp* con la función potencial al cuadrado.
- Sin término independiente.
- $\lambda = 273333333$

```
datos = cbind(I(temp^2) , ibh , humidity)
datos.test = cbind(I(LAozone.test$temp^2) , LAozone.test$ibh , LAozone.test$humidity)

# Calculamos los pesos
w = Regress_LinWD(datos, ozone, 273333333)

# Calculamos los errores
Ein = mean((LAozone.train$ozone - datos%*%w)^2)
Etest = mean((LAozone.test$ozone - datos.test%*%w)^2)

cat("Etest: ", Etest,"\nEin: ",Ein,"\n")
```

```
## Etest:  21.34325
## Ein:  19.40672
```

Nos hemos decantando por el anterior modelo puesto que es el que menor error en el conjunto de test nos ha salido.

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 8

Estimacion del error E_{out} del modelo lo más ajustada posible.

Para estimar el error E_{out}, nos hemos basado en E_{test}, porque con la de E_{in}, obtendríamos una cota peor.

Vamos a obtener la cota E_{out} basada en E_{test}, para ello hacemos uso del libro *Learning from Data*, en concreto en la *página 40*, donde nos viene la ecuación que necesitamos, que está basada en la *desigualdad de Hoeffding* :

$$P[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2N\epsilon^2}$$

Resolviendo esta fórmula, llegamos a:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} ln\left(\frac{2M}{\delta}\right)}$$

Debemos cambiar E_{in} por E_{test}, ya que esta ecuación es para cuando E_{in} tiene un conjunto finito de hipótesis de tamaño M y en este caso, nuestro conjunto de hipótesis es infinito. Por eso, debemos tomar $M = 1$.

$$E_{out}(g) \leq E_{test}(g) + \sqrt{\frac{1}{2N} ln\left(\frac{2M}{\delta}\right)}$$

$$E_{out}(g) \leq E_{test}(g) + \sqrt{\frac{1}{2N} ln\left(\frac{2}{0.05}\right)}$$

```r
datos.test = cbind(I(LAozone.test$temp^2),LAozone.test$ibh,LAozone.test$humidity,1)

# Obtenemos el tamaño de los datos
N <- nrow(datos.test)

# Calculamos el segundo término de la fórmula
x <- sqrt( (1/(2*N)) * log( 2 / 0.05) )

# Obtenemos el valor de Eout
cota_Etest_Eout <- Etest + x
cat ("Cota de E_out basada en E_test: ", cota_Etest_Eout)
```

```
## Cota de E_out basada en E_test:  21.47974
```

Esta es la cota a nivel de confianza 95%, por lo tanto es una toleracia de 0.05. Se observa que E_{out} es ligeramente mayor que E_{test} aunque sigue siendo un buen valor para dar por bueno el modelo seleccionado.

Apartado 9

Discutir y justificar la calidad del modelo encontrado y las razones por las que considera que dicho modelo es un buen ajuste que representa adecuadamente los datos muestrales

Después de haber realizado los anteriores apartados hemos llegado a la conclusión de que el modelo es bueno por las siguientes razones:

- El modelo tiene una cota para E_{out} de 21.48.
- No es un modelo complejo y usa pocas variables:
  - *ibh* sin trasformación.
  - *humidity* sin trasformación.
  - *temp* con la función potencial al cuadrado.
  - Sin término independiente.
- Es un modelo al que aplicamos regularización hasta encontrar el valor óptimo para el E_{test} que determina la cota de generalización que se da para E_{out} que en última instancia es el valor que queremos hacer más pequeño.
- Los pesos solución del modelo lineal son: $w = (0.002842689, -0.0005390458, 0.00114263)$

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Problema de Clasificación

Apartado 1

Comprender el problema a resolver

Para el problema de clasificación, hemos escogido la base de datos South african Heart Disease, la cual tiene las siguientes columnas:

1. **row.names**: Identificador.
2. **sbp**: Presión arterial sistólica.
3. **tobacco**: Tabaco acumulado (kg).
4. **ldl**: Lipoproteína de baja densidad (Colesterol malo).
5. **adiposity**: Adiposidad.
6. **famhist**: Historia familiar de enfermedad cardíaca (presente o ausente).
7. **typea**: Comportamiento Tipo-A.
8. **obesity**: Obesidad.
9. **alcohol**: Consumo de alcohol actual.
10. **age**: Edad de inicio.
11. **chd**: Variable respuesta, enfermedad coronaria.

Ninguna variable podrá tomar valores *NA*. Además, nuestra variable de respuesta será **chd** como nos proporciona el enlace de la base de datos. Por consiguiente, clasificaremos en función de **chd**.

Nuestro objetivo será buscar un modelo para encontrar que factores son los que más afectan a las personas que pueden padecer una enfermedad coronaria.

Antes de nada, vamos a leer nuestro data.frame:

```r
# Guardamos el data frame en una vaarible
SAheart <- read.csv("datos/SAheart.data")
```

```r
# Visualizamos los datos (la tabla)
head(SAheart)
```

```
##   row.names sbp tobacco  ldl adiposity famhist typea obesity alcohol age
## 1         1 160   12.00 5.73     23.11 Present    49   25.30   97.20  52
## 2         2 144    0.01 4.41     28.61  Absent    55   28.87    2.06  63
## 3         3 118    0.08 3.48     32.28 Present    52   29.14    3.81  46
## 4         4 170    7.50 6.41     38.03 Present    51   31.99   24.26  58
## 5         5 134   13.60 3.50     27.78 Present    60   25.99   57.34  49
## 6         6 132    6.20 6.47     36.21 Present    62   30.77   14.14  45
##   chd
## 1   1
## 2   1
## 3   0
## 4   1
## 5   1
## 6   0
```

```r
# Mostramos la dimensión que tiene (462 filas y 11 columnas)
dim(SAheart)
```

```
## [1] 462  11
```

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 2

Los conjuntos de training, validación y test usados en su caso.

Como se nos proporciona solo un archivo, somos nosotros quiénes tenemos que hacer la partición de los datos. Para ello, vamos a usar el mismo procedimiento usado por la profesora en la documentación proporcionada. Por tanto, partimos el data.frame en un 70% para el training y en un 30% para el test.

```r
# Establecemos una semilla por defecto para hacer el mismo tipo de partición
set.seed(1)

# Nos quedamos con los indices para el training
train = sample (nrow(SAheart), round(nrow(SAheart)*0.7))

# Reservamos por separado training y tes
SAheart.train = SAheart[train,]
SAheart.test = SAheart[-train,]
```

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 3

Preprocesado de los datos: Falta de datos, categorización, normalización, reducción de dimensionalidad, etc.

Como pudimos ver en la cabecera del data.frame, la primera columna es un identificador por lo cual no nos sirve para ajustar los modelos, por lo que la podemos despreciar.

```r
# Borramos la primera columna tanto para training como para test
SAheart.train <- SAheart.train[,-1]
SAheart.test <- SAheart.test[,-1]
```

La quinta columna es una variable binaria dada como factores, por lo cual la cambiamos a ceros y unos para poder trabajar con ella.

- Cero significa la ausencia de la característica.
- Uno significa la presencia de de la característica

```r
# Cambiamos a numérico la columna del training
SAheart.train[,5] <- as.numeric(SAheart.train[,5]) # Obtenemos 2 y 1
SAheart.train[,5][SAheart.train[,5] == 1] = 0 # Cambiamos el 1 por el 0
SAheart.train[,5][SAheart.train[,5] == 2] = 1 # Cambiamos el 2 por el 2

# Cambiamos a numérico la columna del training
SAheart.test[,5] <- as.numeric(SAheart.test[,5]) # Obtenemos 2 y 1
SAheart.test[,5][SAheart.test[,5] == 1] = 0 # Cambiamos el 1 por el 0
SAheart.test[,5][SAheart.test[,5] == 2] = 1 # Cambiamos el 2 por el 2
```

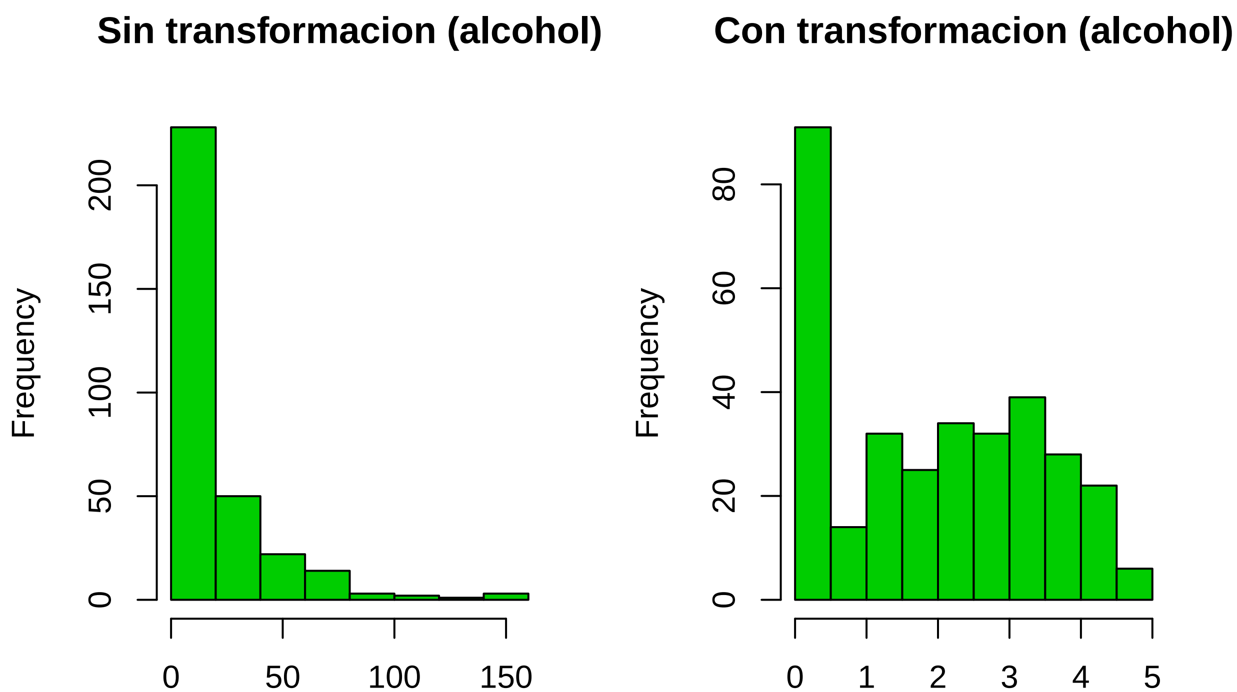
Ahora procedemos a transformar las variables que tengan una distribución de los datos asimétrica. Para ello seguimos el mismo procedimiento que en el problema de regresión, salvo que puesto que las varibales *alcohol* y *tobacco* tienen ceros en su columna, hacemos una traslación sumando uno a todos los valores para poder realizar la trasformación que mitiga la asimetría.

```r
# Obtenemos el valor de asimetría de los datos training
v_asimetria = apply(SAheart.train, 2, skewness)

# Ordenamos de mayor a menor los valores obtenido
sort(abs(v_asimetria), decreasing = T)
```

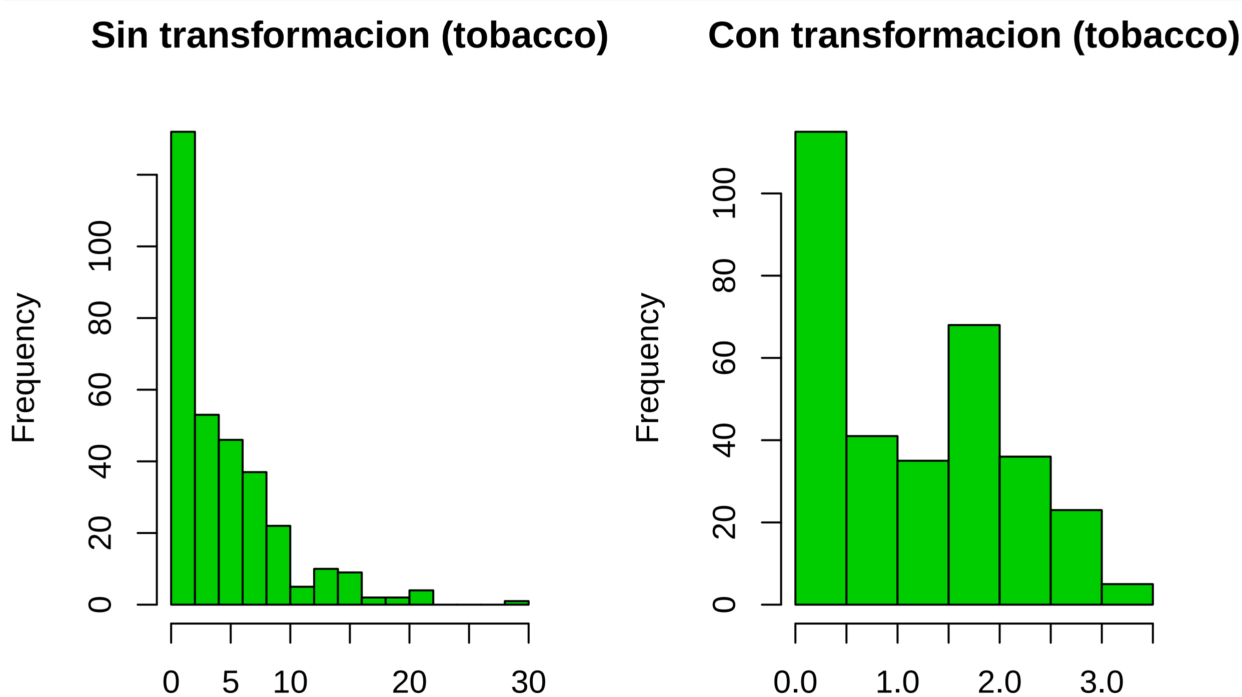
```
##   alcohol   tobacco       ldl       sbp   obesity       chd     typea
## 2.4885752 1.7699384 1.4282215 1.2353155 0.8241934 0.5985251 0.4159875
##       age   famhist adiposity
## 0.4095289 0.3055339 0.2857652
```

Luego, transformamos los datos a partir de un umbral de 0.8. Elegimos este umbral, ya que los valores más cercanos a 0 serán los más simétricos. Por lo tanto, realizaremos transformaciones para *alcohol*, *tobacco*, *ldl*, *sbp* y *obesity*.

```r
# Transformación para la variable alcohol
# ------------------------------------------------------------------------------
par(mfrow=c(1:2)) # Dividimos la región en dos partes
alcohol_trans = BoxCoxTrans(SAheart.train$alcohol+1) # Obtenemos la transformación
# Comparamos los histogramas con transformación y sin transformación
hist(SAheart.train$alcohol+1, main = "Sin transformacion (alcohol)", xlab = "", col = 3)
hist(predict(alcohol_trans,SAheart.train$alcohol+1),
     main = "Con transformacion (alcohol)", xlab = "", col = 3)
```

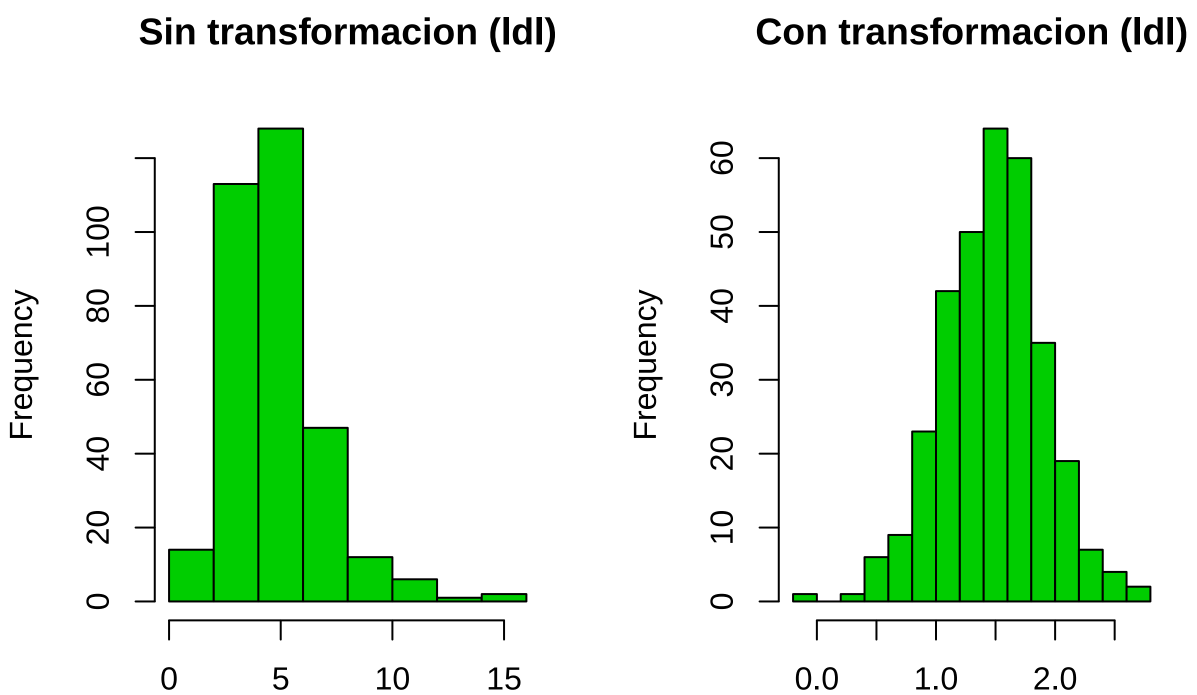



```r
# Transformación para la variable tobacco
# --------------------------------------------------------------------------------
par(mfrow=c(1:2)) # Dividimos la región en dos partes
tobacco_trans = BoxCoxTrans(SAheart.train$tobacco+1) # Obtenemos la transformación
# Comparamos los histogramas con transformación y sin transformación
hist(SAheart.train$tobacco+1, main = "Sin transformacion (tobacco)", xlab = "", col = 3)
hist(predict(tobacco_trans,SAheart.train$tobacco+1),
     main = "Con transformacion (tobacco)", xlab = "", col = 3)
```

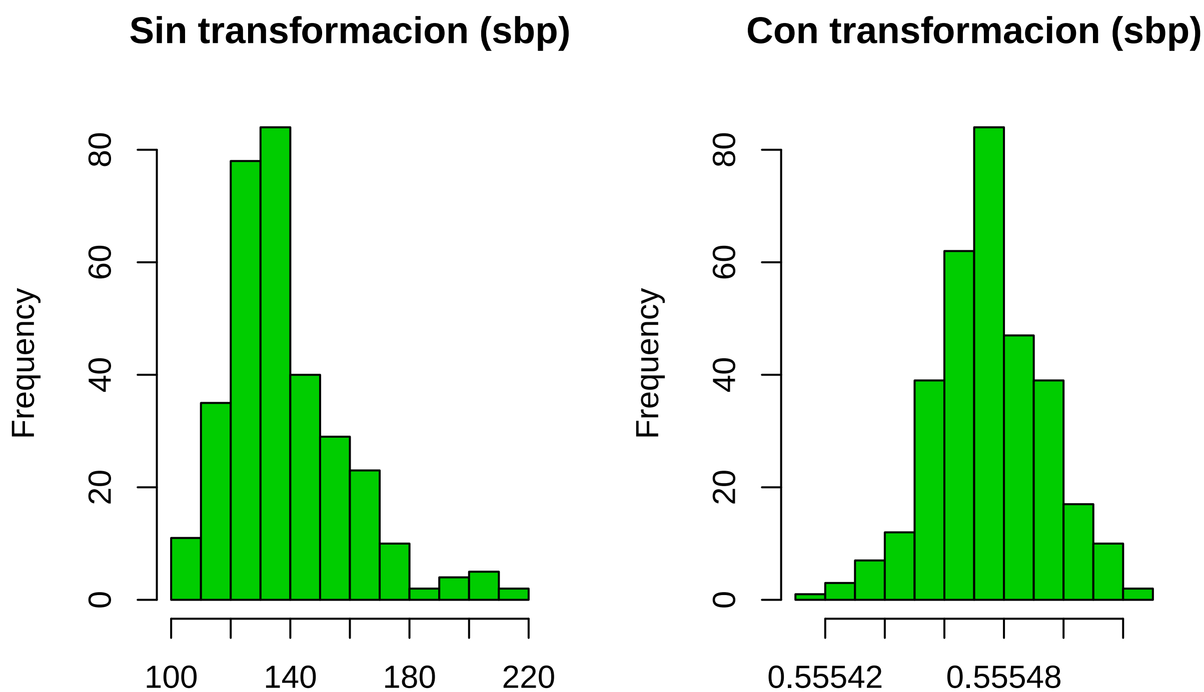



```r
# Transformación para la variable ldl
# --------------------------------------------------------------------------------
```

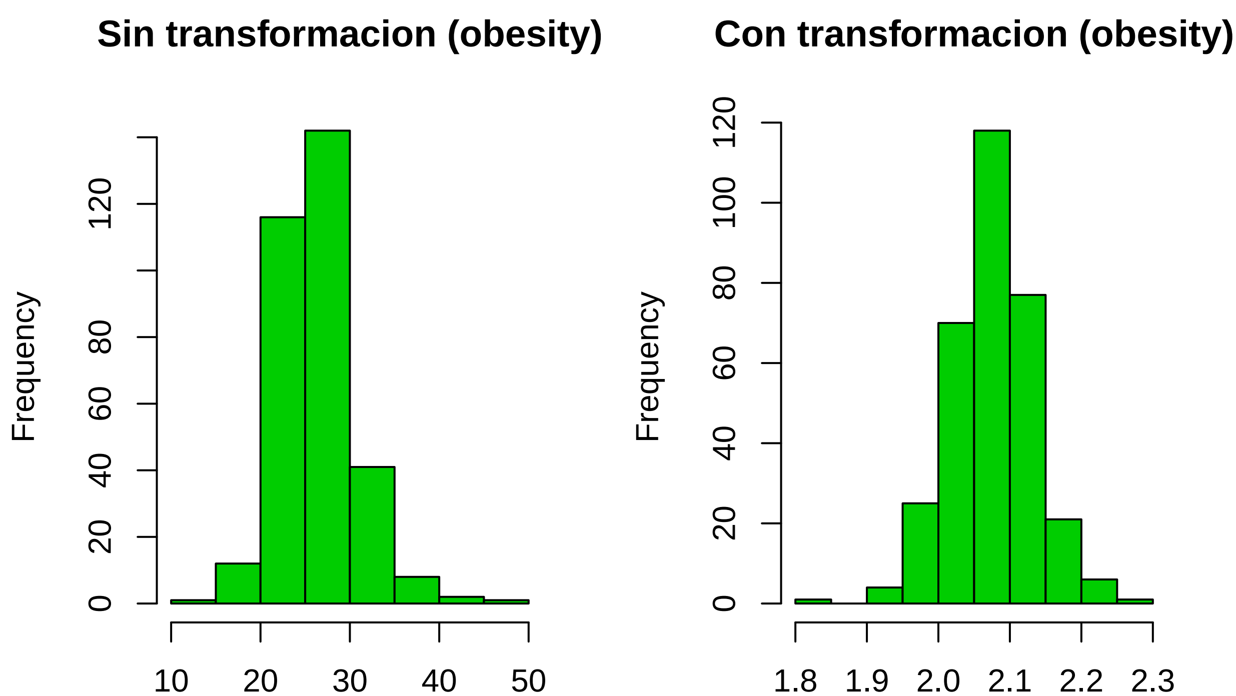
```
par(mfrow=c(1:2)) # Dividimos la región en dos partes
ldl_trans = BoxCoxTrans(SAheart.train$ldl) # Obtenemos la transformación
# Comparamos los histogramas con transformación y sin transformación
hist(SAheart.train$ldl, main = "Sin transformacion (ldl)", xlab = "", col = 3)
hist(predict(ldl_trans,SAheart.train$ldl),
     main = "Con transformacion (ldl)", xlab = "", col = 3)
```




```
# Transformación para la variable sbp
# --------------------------------------------------------------------------------
par(mfrow=c(1:2)) # Dividimos la región en dos partes
sbp_trans = BoxCoxTrans(SAheart.train$sbp) # Obtenemos la transformación
# Comparamos los histogramas con transformación y sin transformación
hist(SAheart.train$sbp, main = "Sin transformacion (sbp)", xlab = "", col = 3)
hist(predict(sbp_trans,SAheart.train$sbp), main = "Con transformacion (sbp)",
     xlab = "", col = 3)
```




```r
# Transformación para la variable obesity ----------------------------------------------------------------
par(mfrow=c(1:2)) # Dividimos la región en dos partes
obesity_trans = BoxCoxTrans(SAheart.train$obesity) # Obtenemos la transformación
# Comparamos los histogramas con transformación y sin transformación
hist(SAheart.train$obesity, main = "Sin transformacion (obesity)", xlab = "", col = 3)
hist(predict(obesity_trans,SAheart.train$obesity),
     main = "Con transformacion (obesity)", xlab = "", col = 3)
```




```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Como se puede comprobar los valores ya son más simétricos. Entonces, ya solo nos queda guardar esas transformaciones para el train.

```
# Transformación alcohol (train)
SAheart.train$alcohol = predict(alcohol_trans,SAheart.train$alcohol+1)

# Transformación tobacco (train)
SAheart.train$tobacco = predict(tobacco_trans,SAheart.train$tobacco+1)

# Transformación ldl (train)
SAheart.train$ldl = predict(ldl_trans,SAheart.train$ldl)

# Transformación sbp (train)
SAheart.train$sbp = predict(sbp_trans,SAheart.train$sbp)

# Transformación obesity (train)
SAheart.train$obesity = predict(obesity_trans,SAheart.train$obesity)
```

También guardamos esas mismas transformaciones para el test.

```
# Transformación alcohols (test)
SAheart.test$alcohol = predict(alcohol_trans,SAheart.test$alcohol+1)

# Transformación tobacco (test)
SAheart.test$tobacco = predict(tobacco_trans,SAheart.test$tobacco+1)

# Transformación ldl (test)
SAheart.test$ldl = predict(ldl_trans,SAheart.test$ldl)

# Transformación sbp (test)
SAheart.test$sbp = predict(sbp_trans,SAheart.test$sbp)

# Transformación obesity (test)
SAheart.test$obesity = predict(obesity_trans,SAheart.test$obesity)
```

Apartado 4

Selección de clases de funciones a usar

```
attach(SAheart.train) # para prescendir y simplificar el prefijo SAheart
```

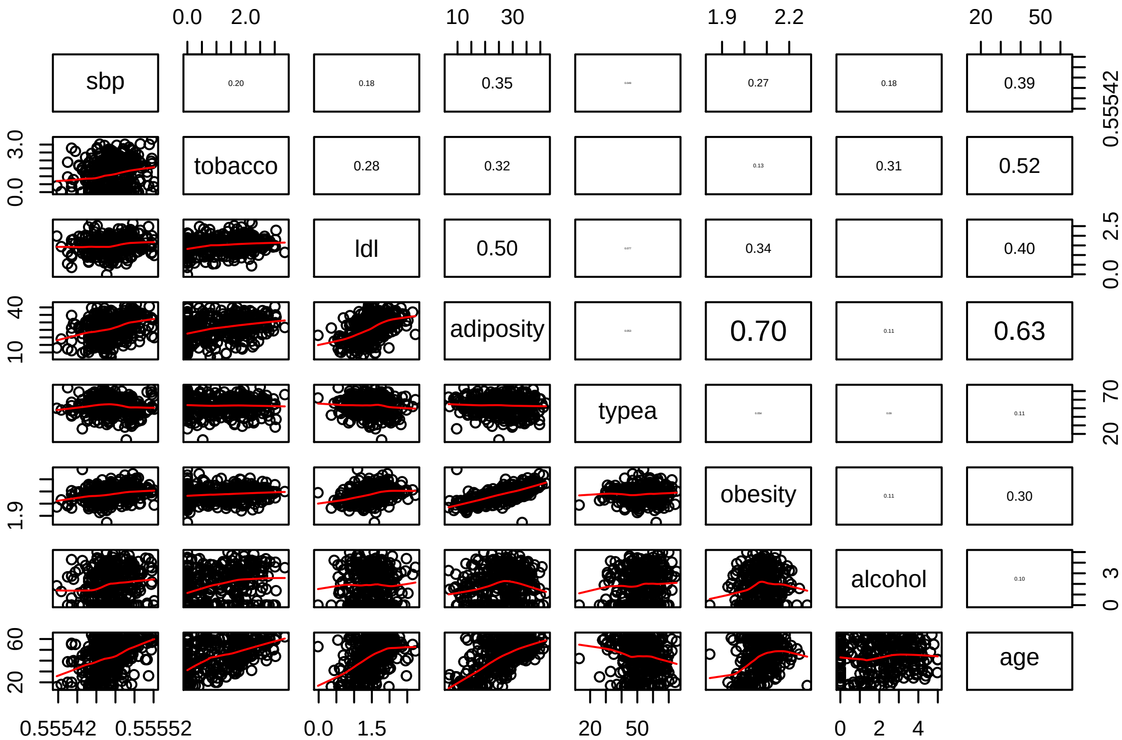
Antes de nada, implementamos una función que tiene el mismo funcionamiento que *pairs* pero en el triángulo superior mostramos los coeficientes de correlación lineal que nos dicen como de relacionadas linealmente están las variables. Estos coeficientes se imprimen con un tamaño proporcional a su valor.

```
panel.cor <- function(x, y, digits=2, prefix="", cex.cor) {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits=digits)[1]
  txt <- paste(prefix, txt, sep="")
  if(missing(cex.cor)) cex <- 0.8/strwidth(txt)
```

```r
  text(0.5, 0.5, txt, cex = cex * r)
}
```

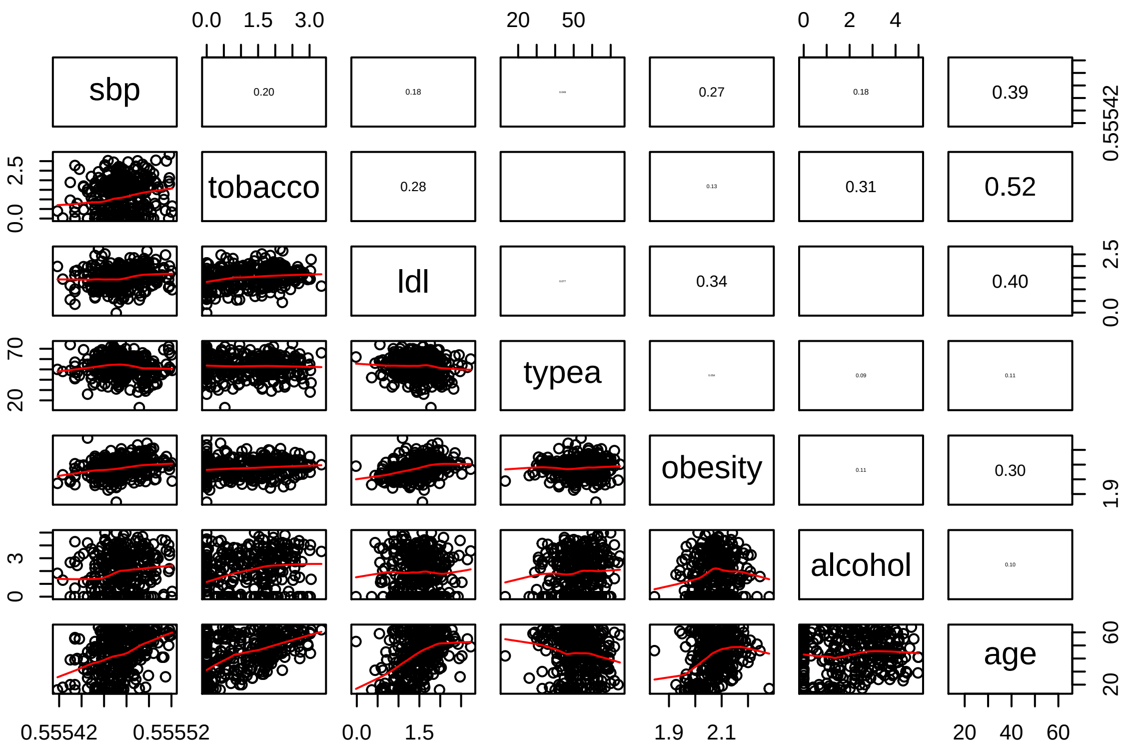
Hacemos uso de la función, y mostramos gráficamente todas las variables con todas, salvo la variable respuesta (*chd*) y la binaria (*famhist*).

```r
pairs(SAheart.train[,c(-5,-10)], lower.panel=panel.smooth,upper.panel=panel.cor)
```



Puesto que *adiposity* está muy relacionada linealmente con dos variables, optamos por eliminarla de los atributos que usaremos para los modelos.

```r
# Quitamos adiposity tanto para train como para test
SAheart.train <- SAheart.train[,-4]
SAheart.test <- SAheart.test[,-4]

# Y volvemos a mostrar el gráfico
pairs(SAheart.train[,c(-4,-9)], lower.panel=panel.smooth,upper.panel=panel.cor)
```



Observamos que las variables restantes no tienen una relacion lineal fuerte para decidir eliminarlas. Por lo que trabajaremos con ellas.

```
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Una vez arreglados nuestros datos, ya podemos pasar a la selección del modelo. Para empezar a seleccionar el mejor modelo, nos basaremos dos condiciones:

1. En el error E_{test}, debido a que éste se aproxima al error fuera de la muestra, E_{out}, mejor que E_{in}. Por ello creamos una función que nos calcule los errores E_{in} y E_{test}.

2. Nos preocuparemos en obtener unos buenos resultados de la matriz de confusión. Nuestro objetivo será disminuir la variable Y, ya que esa variable significa *las personas que predecimos que están sanas, pero en realidad están enfermas*. Sin embargo, tendremos que controlar también la variable Z, ya que son *las personas que predecimos que están enfermas, pero en realidad están sanas*.

	0 (Positivo)	1 (Negativo)
0 (Positivo)	X	Y
1 (Negativo)	Z	X

Por lo tanto, para elegir el mejor modelo, intentaremos mantener equilibrado ambas condiciones.

Ahora, creamos una función que dado un modelo calcula el error dentro de la muestra (E_{in}) y el error en el conjunto de test (E_{test})

```
# Función que calcula los errores E_in y E_test para regresión logística
errorres_regresion_logistica <- function(m){

  probTr = predict(m, type="response")
```

```r
  probTst = predict(m, data.frame(SAheart.test), type="response")

  predTst = rep(0, length(probTst)) # predicciones por defecto 0
  predTst[probTst >= 0.5] = 1 # >= 0.5 clase 1

  predTr = rep(0, length(probTr)) # predicciones por defecto 0
  predTr[probTr >= 0.5] = 1 # >= 0.5 clase 1

  print(table(predTst, Real=SAheart.test$chd)) # Para el calculo del Etest

  # Calculamos los errores
  Ein = mean(predTr != SAheart.train$chd)
  Etest = mean(predTst != SAheart.test$chd)

  list(Etest=Etest, Ein=Ein)
}
```

Modelo 1: Regresión Logística con todas las variables para predecir *chd*.

```r
# Creamos el modelo
ml1 = glm(chd ~ sbp + tobacco + ldl + typea + obesity + alcohol + age, family = binomial(logit),
          data = SAheart.train)
summary(ml1) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ sbp + tobacco + ldl + typea + obesity + alcohol +
##     age, family = binomial(logit), data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9932  -0.8428  -0.4262   0.9708   2.2184
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -180.79795 4289.13268  -0.042 0.966377
## sbp          330.92859 7723.30569   0.043 0.965823
## tobacco        0.45694    0.16701   2.736 0.006219 **
## ldl            1.28626    0.38217   3.366 0.000764 ***
## typea          0.04487    0.01464   3.064 0.002186 **
## obesity       -5.22582    2.55984  -2.041 0.041205 *
## alcohol       -0.03683    0.09395  -0.392 0.695020
## age            0.05271    0.01303   4.046 5.21e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 339.62  on 315  degrees of freedom
## AIC: 355.62
##
## Number of Fisher Scoring iterations: 5
```

```r
errorres_regresion_logistica(ml1) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 78 25
##       1 16 20
```

```
## $Etest
## [1] 0.294964
##
## $Ein
## [1] 0.2941176
```

Como se puede ver en la salida del *summary()*, los coeficientes más importantes son *ldl* y *age*, ya que tienen las 3 estrellas. Sin embargo, no podemos despreciar a *typea* y *tobacco* que tienen dos estrellas. La primera comparación que haremos será de *chd* con las cuatro mejores, sin niguna transformación.

Modelo 2: Predecimos *chd* con la variable *ldl* mediante Regresión Logística.

```r
# Creamos el modelo
ml2 = glm(chd ~ ldl, family = binomial(logit), data = SAheart.train)
summary(ml2) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ ldl, family = binomial(logit), data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7806  -0.9472  -0.7003   1.2024   2.1993
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.0291     0.5153  -5.879 4.14e-09 ***
## ldl           1.6064     0.3233   4.969 6.72e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 392.01  on 321  degrees of freedom
## AIC: 396.01
##
## Number of Fisher Scoring iterations: 4
```

```r
errorres_regresion_logistica(ml2) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 81 35
##       1 13 10
```

```
## $Etest
## [1] 0.3453237
##
```

```
## $Ein
## [1] 0.3219814
```

Modelo 3: Predecimos *chd* con la variable *age* mediante Regresión Logística.

```r
# Creamos el modelo
ml3 = glm(chd ~ age , family = binomial(logit), data = SAheart.train)
summary(ml3) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ age, family = binomial(logit), data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4385  -0.9313  -0.5328   1.0877   2.1464
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.46812    0.49641  -6.986 2.82e-12 ***
## age          0.06350    0.01015   6.258 3.91e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 371.84  on 321  degrees of freedom
## AIC: 375.84
##
## Number of Fisher Scoring iterations: 4
```

```r
errorres_regresion_logistica(ml3) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 78 27
##       1 16 18
```

```
## $Etest
## [1] 0.3093525
##
## $Ein
## [1] 0.3250774
```

Modelo 4: Predecimos *chd* con la variable *tobacco* mediante Regresión Logística.

```r
# Creamos el modelo
ml4 = glm(chd ~ tobacco , family = binomial(logit), data = SAheart.train)
summary(ml4) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ tobacco, family = binomial(logit), data = SAheart.train)
##
## Deviance Residuals:
```

```
##      Min       1Q   Median       3Q      Max  
## -1.5607  -0.9208  -0.6174   1.1117   1.8716  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  -1.5607     0.2170  -7.191 6.44e-13 ***
## tobacco       0.8025     0.1387   5.786 7.21e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 383.31  on 321  degrees of freedom
## AIC: 387.31
## 
## Number of Fisher Scoring iterations: 4
```

```r
errorres_regresion_logistica(ml4) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 79 30
##       1 15 15

## $Etest
## [1] 0.323741
## 
## $Ein
## [1] 0.3219814
```

Modelo 5: Predecimos *chd* con la variable *typea* mediante Regresión Logística.

```r
# Creamos el modelo
ml5 = glm(chd ~ typea, family = binomial(logit), data = SAheart.train)
summary(ml5) # Realizamos un análisis del modelo
```

```
## 
## Call:
## glm(formula = chd ~ typea, family = binomial(logit), data = SAheart.train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.0905  -0.9526  -0.8839   1.3747   1.6683  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -1.67689    0.67046  -2.501   0.0124 *
## typea        0.02040    0.01236   1.651   0.0987 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 420.61  on 322  degrees of freedom
```

```
## Residual deviance: 417.83  on 321  degrees of freedom
## AIC: 421.83
##
## Number of Fisher Scoring iterations: 4
```

```
errorres_regresion_logistica(ml5) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 94 45
```

```
## $Etest
## [1] 0.323741
##
## $Ein
## [1] 0.3560372
```

No hay ningún atributo que explique por sí sola la variable *chd* con amplia diferencia atendiendo a los errores que produce la regresión lineal ya la matriz de confusión. Ya que los modelos 2, 3 y 4, obtienen resultados parecidos.

Ahora vamos a usar varias combinaciones de las 4 mejores variables para encontrar una predicción de *chd* más ajustada.

Modelo 6: Predecimos *chd* con las variables *ldl*, *age* y *tobacco* mediante Regresión Logística.

```
# Creamos el modelo
ml6 = glm(chd ~ ldl + age + tobacco, family = binomial(logit), data = SAheart.train)
summary(ml6) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ ldl + age + tobacco, family = binomial(logit),
##     data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7128  -0.8823  -0.4577   1.0183   2.3777
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.60656    0.68257  -6.749 1.49e-11 ***
## ldl          0.98315    0.34752   2.829 0.004669 **
## age          0.04302    0.01166   3.691 0.000224 ***
## tobacco      0.45453    0.15557   2.922 0.003482 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 352.96  on 319  degrees of freedom
## AIC: 360.96
##
## Number of Fisher Scoring iterations: 4
```

```r
errorres_regresion_logistica(ml6) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 81 24
##       1 13 21

## $Etest
## [1] 0.2661871
##
## $Ein
## [1] 0.2910217
```

Modelo 7: Predecimos *chd* con las variables *ldl*, *age* y *typea* mediante Regresión Logística.

```r
# Creamos el modelo
ml7 = glm(chd ~ ldl + age + typea, family = binomial(logit), data = SAheart.train)
summary(ml7) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ ldl + age + typea, family = binomial(logit),
##     data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7417  -0.9012  -0.4612   1.0363   2.3120
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.32202    1.16811  -6.268 3.65e-10 ***
## ldl          1.13534    0.34700   3.272  0.00107 **
## age          0.06124    0.01132   5.408 6.38e-08 ***
## typea        0.04152    0.01420   2.924  0.00346 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 352.63  on 319  degrees of freedom
## AIC: 360.63
##
## Number of Fisher Scoring iterations: 4
```

```r
errorres_regresion_logistica(ml7) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 79 27
##       1 15 18

## $Etest
## [1] 0.3021583
##
## $Ein
```

```
## [1] 0.2972136
```

Modelo 8: Predecimos *chd* con las variables *ldl*, *age*, *typea* y *tobacco* mediante Regresión Logística.

```r
# Creamos el modelo
ml8 = glm(chd ~ ldl + age + typea + tobacco, family = binomial(logit), data = SAheart.train)
summary(ml8) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ ldl + age + typea + tobacco, family = binomial(logit),
##     data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8605  -0.8683  -0.4374   0.9894   2.1774
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.22526    1.18876  -6.078 1.22e-09 ***
## ldl          1.05824    0.35433   2.987  0.00282 **
## age          0.04965    0.01227   4.047 5.18e-05 ***
## typea        0.04137    0.01445   2.863  0.00420 **
## tobacco      0.44724    0.15678   2.853  0.00433 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 344.30  on 318  degrees of freedom
## AIC: 354.3
##
## Number of Fisher Scoring iterations: 5
```

```r
errorres_regresion_logistica(ml8) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 81 24
##       1 13 21
```

```
## $Etest
## [1] 0.2661871
##
## $Ein
## [1] 0.2848297
```

De entre las combinaciones posibles, comprobamos que con los modelos 6 y 8, obtenemos una mejoría y mejor resultados en las dos condiciones necesarias para elegir el modelo final. Pero, vamos a intentar mejorar aún más.

A continuación, vamos a probar transformaciones en las variables ya que sospechamos que puede que no haya linealidad en los datos para explicar *chd*, puesto que las soluciones no fueron satisfactorias.

Modelo 9: Predecimos *chd* mediante Regresión Logística, usando la potencia de elevar *age* a cinco.

```r
# Creamos el modelo
ml9 = glm(chd ~ I(age^5) , family = binomial(logit), data = SAheart.train)
summary(ml9) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ I(age^5), family = binomial(logit), data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5855  -0.8234  -0.7045   1.1507   1.7468
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.287e+00  1.818e-01  -7.081 1.43e-12 ***
## I(age^5)     2.057e-09  3.823e-10   5.381 7.41e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 389.48  on 321  degrees of freedom
## AIC: 393.48
##
## Number of Fisher Scoring iterations: 4
```

```r
errorres_regresion_logistica(ml9) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 84 28
##       1 10 17
```

```
## $Etest
## [1] 0.2733813
##
## $Ein
## [1] 0.3312693
```

Además, usaremos combinaciones variables con y sin transformaciones no lineales para encocntrar una predicción de chd más ajustada.

Modelo 10: Predecimos *chd* mediante Regresión Logística, usando la potencia de elevar *age* a cinco y sin transformación para *ldl*.

```r
# Creamos el modelo
ml10 = glm(chd ~ I(age^5) + ldl , family = binomial(logit), data = SAheart.train)
summary(ml10) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ I(age^5) + ldl, family = binomial(logit),
##     data = SAheart.train)
##
## Deviance Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.7233  -0.8478  -0.6078   1.1049   2.2951
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.186e+00  5.310e-01  -6.000 1.98e-09 ***
## I(age^5)     1.734e-09  3.962e-10   4.378 1.20e-05 ***
## ldl          1.317e+00  3.322e-01   3.965 7.32e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 372.22  on 320  degrees of freedom
## AIC: 378.22
##
## Number of Fisher Scoring iterations: 4
```

```
errorres_regresion_logistica(ml10) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 85 28
##       1  9 17
```

```
## $Etest
## [1] 0.2661871
##
## $Ein
## [1] 0.3188854
```

Modelo 11: Predecimos *chd* mediante Regresión Logística usando las variables sin transformar *age* y *tobacco* y con la transformación de *ldl*.

```
# Creamos el modelo
ml11 = glm(chd ~ atan(ldl) + age + tobacco, family = binomial(logit), data = SAheart.train)
summary(ml11) # Realizamos un análisis del modelo
```

```
##
## Call:
## glm(formula = chd ~ atan(ldl) + age + tobacco, family = binomial(logit),
##     data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6766  -0.8986  -0.4728   1.0225   2.4790
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.64541    1.08374  -5.209  1.9e-07 ***
## atan(ldl)    2.58586    1.08231   2.389 0.016885 *
## age          0.04384    0.01166   3.762 0.000169 ***
## tobacco      0.45354    0.15518   2.923 0.003471 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 355.00  on 319  degrees of freedom
## AIC: 363
##
## Number of Fisher Scoring iterations: 4
```

```
errorres_regresion_logistica(ml11) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 81 23
##       1 13 22
```

```
## $Etest
## [1] 0.2589928
##
## $Ein
## [1] 0.2879257
```

Modelo 12: Predecimos *chd* mediante Regresión Logística usando las variables sin transformar *typea* y *tobacco* y con la transformación de *age*.

```
ml12 = glm(chd ~  I(age^5) + tobacco + typea,
          family = binomial(logit), data = SAheart.train)
summary(ml12)
```

```
##
## Call:
## glm(formula = chd ~ I(age^5) + tobacco + typea, family = binomial(logit),
##     data = SAheart.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0265  -0.8734  -0.5534   1.0478   2.1065
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.924e+00  8.432e-01  -4.653 3.27e-06 ***
## I(age^5)     1.737e-09  4.297e-10   4.042 5.29e-05 ***
## tobacco      6.362e-01  1.481e-01   4.297 1.73e-05 ***
## typea        3.704e-02  1.418e-02   2.613  0.00897 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.61  on 322  degrees of freedom
## Residual deviance: 362.75  on 319  degrees of freedom
## AIC: 370.75
##
## Number of Fisher Scoring iterations: 4
```

```r
errorres_regresion_logistica(ml12) # Obtenemos los errores
```

```
##        Real
## predTst  0  1
##       0 84 26
##       1 10 19
```

```
## $Etest
## [1] 0.2589928
##
## $Ein
## [1] 0.3003096
```

En este modelo, obtenemos una pequeña mejoría, en comparación a los demás modelos, donde usábamos.

Después de probar varios modelos, antes de decantarnos por uno de ellos vamos a aplicar regularización sobre los dos mejores (Modelo 6 y Modelo 12) y si obtenemos mejores resultados será necesario tal regularización.

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 5

Discutir la necesidad de regularización y en su caso la función usada.

Antes de comenzar a hacer el ejercicio, debemos tener en cuenta que a nuestra matriz de confusión le hemos dado una penalización a los valores donde *la persona esté enferma y le decimos que está sana*. La matriz de costes que hemos usado ha sido:

	0 (Positivo)	1 (Negativo)
0 (Positivo)	0	1.5
1 (Negativo)	1	0

Para utilizar regularización usamos una función que hemos definido nosotros (usada en la práctica 2) donde aplicaremos el modelo de Regresión Lineal con Weight Decay, cuyos pesos serán pasados al PLA_Pocket (implementado en la práctica 2).

```r
# Función PLA_Pocket
PLA_pocket=function(datos, label, max_iter, vini, c1){

  # Establecemos una semilla por defecto
  set.seed(79)

  if(max_iter == 0)
    return(list(w=vini))

  # Este es el numero de datos que tenemos, uno por cada fila
  numDatos = nrow(datos)

  # Esta variable indicara si hemos conseguido ajustar los coeficientes
  # de tal manera que clasifiquen bien todos los puntos
  malAjustado = F
```

```r
# En esta variable devolveremos el numero de iteraciones usado
# que como vemos si no se modifica es el maximo
numIter = max_iter

# En esta variable guardamos la mejor solucion hasta el momento
mejorVini = vini
# En esta el error menor alcanzado hasta el momento
mejorErr = mean(sign(datos%*%vini)!=label)+(c1-1)*mean(sign(datos%*%vini)==-1 & label==1)

# Bucle principal tendra como maximo max_iter iteraciones
for(n in 1:max_iter) {

  # Calculamos indices aleatorios que dan el orden en
  # que explorar los puntos de entrada
  indices=sample(1:numDatos,numDatos)

  # Para cada incide del vector
  for( i in indices) {

    # Comprobamos si el signo que asigna el w actual
    # coincide con con la etiqueta
    if(sign(datos[i,]%*%vini) != label[i]) {

      # si no es asi probamos con un w nuevo
      vini = vini+datos[i,]*label[i]
      malAjustado = T

      # Calculamos el error de la solucion nueva
      errActual = mean(sign(datos%*%vini)!=label)+(c1-1)*mean(sign(datos%*%vini)==-1 & label==1)
      # Si el w nuevo es mejor lo cambiamos y volvemos a empezar
      # la exploracion
      if(errActual<mejorErr) {
        mejorVini = vini
        mejorErr = errActual
        break
      }
    }
  }

  # Si no esta mal ajustado quiere decir que se
  # clasificaron todos los puntos bien y hemos terminado
  if(!malAjustado){
    # Guardamos el numero de iteraciones que fueron necesarias
    numIter=n
    # Salimos del bucle principal
    break

  } else {
    # Si esta mal ajustado ponemos la variable a false para
    # comprobar en la siguiente iteracion
    malAjustado = F
  }
}
```

```r
  # Devolvemos la salida en esta lista
  list(w=mejorVini,num_iteraciones=numIter,error=mejorErr)
}
```

Nos volvemos a crear una función que calcula los errores, ya que ahora estamos en clasificación. Haremos un cambio de etiquetas, es decir, pasaremos nuestras etiquetas a -1 y 1, para que las funciones de antes puedan ser usadas, y luego pasaremos el -1 a 0, ya que nuestra variable respuesta consta de 0 y 1.

```r
# Función que calcula el error
RegPLA <- function(datos,datostest,landa,maxiter,c=1.5){

  # Hacemos un cambio de numeración para las etiquetas (train y test)
  realEtiqTr = SAheart.train$chd
  realEtiqTr[realEtiqTr == 0] = -1
  realEtiqTst = SAheart.test[,9]
  realEtiqTst[realEtiqTst == 0] = -1

  # Calculamos RL con WD
  w = Regress_LinWD(datos,realEtiqTr,landa)
  # Le pasamos los pesos
  w = PLA_pocket(datos,realEtiqTr,maxiter,w,c)$w #rep(0,ncol(datos))

  # Volvemos a la numeración correcta de las etiquetas (train y test)
  etiqTr=sign(datos%*%w)
  etiqTr[etiqTr == -1] = 0
  etiqTst=sign(datostest%*%w)
  etiqTst[etiqTst == -1] = 0

  # Pintamos nuestra matriz de confusión
  print(table(etiqTst,Real=SAheart.test$chd))

  # Calculamos los errores
  Ein=mean(etiqTr != SAheart.train$chd)
  Etest=mean(etiqTst != SAheart.test$chd)

  list(Ein=Ein, Etest=Etest, w=w)
}
```

Una vez implementadas, realizamos la misma idea que para el problema anterior, es decir, cogemos los dos mejores modelos (Modelo 6 y Modelo 12) que hemos obtenido con Regresión Logística e intentamos mejorarlos, introduciendo un lambda (regularización).

Modelo 1: Regresión Lineal usando WD, cuyos pesos son introducidos al PLA_Pocket (anteriormente modelo 6), con los siguientes atributos:

- *ldl* sin transformación.
- *age* sin transformación.
- *tobacco* sin transformación.

```r
datos = cbind(SAheart.train$ldl , SAheart.train$age, SAheart.train$tobacco,1)
datos.test = cbind(SAheart.test$ldl , SAheart.test$age, SAheart.test$tobacco,1)

# Establecemos por defecto un e_test alto ya que nunca llegaremos a ese valor
```

```r
Etest = 100

# Calculamos el error para varias lambdas
for(landa in seq(0,0.001,length.out=10)){
  sol = RegPLA(datos,datos.test,landa,50)
  EtestAc = sol$Etest

  # Nos quedamos con los mejores resultados
  if(EtestAc < Etest){
    Etest = EtestAc
    wmej = w
    landamej = landa
  }

  cat(" Landa: ",landa," Etest: ", EtestAc,"\n")
}
```

```
##         Real
## etiqTst  0  1
##        0 81 24
##        1 13 21
##  Landa:  0  Etest:  0.2661871
##         Real
## etiqTst  0  1
##        0 81 24
##        1 13 21
##  Landa:  0.0001111111  Etest:  0.2661871
##         Real
## etiqTst  0  1
##        0 81 24
##        1 13 21
##  Landa:  0.0002222222  Etest:  0.2661871
##         Real
## etiqTst  0  1
##        0 81 24
##        1 13 21
##  Landa:  0.0003333333  Etest:  0.2661871
##         Real
## etiqTst  0  1
##        0 81 24
##        1 13 21
##  Landa:  0.0004444444  Etest:  0.2661871
##         Real
## etiqTst  0  1
##        0 81 24
##        1 13 21
##  Landa:  0.0005555556  Etest:  0.2661871
##         Real
## etiqTst  0  1
##        0 81 24
##        1 13 21
##  Landa:  0.0006666667  Etest:  0.2661871
##         Real
## etiqTst  0  1
```

```
##       0 81 24
##       1 13 21
##  Landa:  0.0007777778  Etest:  0.2661871
##        Real
## etiqTst  0  1
##       0 81 24
##       1 13 21
##  Landa:  0.0008888889  Etest:  0.2661871
##        Real
## etiqTst  0  1
##       0 81 24
##       1 13 21
##  Landa:  0.001  Etest:  0.2661871
```

```
cat("El mejor landa es ",landamej," con un Etest de ",Etest,
    " que da la solución: w=[",wmej,"]\n")
```

```
## El mejor landa es  0  con un Etest de  0.2661871  que da la solución: w=[ 0.002842689 -0.0005390458 
```

Tras hacer regularización un un intervalo amplio de landas escogimos dentro del intervalo $[0, 0.01]$, que el valor óptimo de landa sea 0 para que el error en el conjunto de test fuera mínimo. Por lo que no haría falta usar regularización.

```
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Modelo 2: Regresión Lineal usando WD, cuyos pesos son introducidos al PLA_Pocket (anteriormente modelo 12), con los siguientes atributos:

- *age* con la función potencia a la quinta
- *tobacco* sin transformación.
- *typea* sin transformación.

```
datos = cbind(I(SAheart.train$age^5), SAheart.train$tobacco,SAheart.train$typea,1)
datos.test = cbind(I(SAheart.test$age^5),SAheart.test$tobacco,SAheart.test$typea,1)

# Establecemos por defecto un e_test alto ya que nunca llegaremos a ese valor
Etest = 100

# Calculamos el error para varias lambdas
for(landa in seq(0,0.001,length.out=10)){
  sol = RegPLA(datos,datos.test,landa,100)
  EtestAc = sol$Etest

  # Nos quedamos con los mejores resultados
  if(EtestAc<Etest){
    Etest = EtestAc
    wmej = w
    landamej = landa
  }
  cat(" Landa: ",landa," Etest: ", EtestAc,"\n")
}
```

```
##        Real
## etiqTst  0  1
```

```
##       0 84 26
##       1 10 19
##  Landa:  0  Etest:  0.2589928
##        Real
## etiqTst  0  1
##       0 84 26
##       1 10 19
##  Landa:  0.0001111111  Etest:  0.2589928
##        Real
## etiqTst  0  1
##       0 84 26
##       1 10 19
##  Landa:  0.0002222222  Etest:  0.2589928
##        Real
## etiqTst  0  1
##       0 84 26
##       1 10 19
##  Landa:  0.0003333333  Etest:  0.2589928
##        Real
## etiqTst  0  1
##       0 84 26
##       1 10 19
##  Landa:  0.0004444444  Etest:  0.2589928
##        Real
## etiqTst  0  1
##       0 84 26
##       1 10 19
##  Landa:  0.0005555556  Etest:  0.2589928
##        Real
## etiqTst  0  1
##       0 84 26
##       1 10 19
##  Landa:  0.0006666667  Etest:  0.2589928
##        Real
## etiqTst  0  1
##       0 84 26
##       1 10 19
##  Landa:  0.0007777778  Etest:  0.2589928
##        Real
## etiqTst  0  1
##       0 84 26
##       1 10 19
##  Landa:  0.0008888889  Etest:  0.2589928
##        Real
## etiqTst  0  1
##       0 84 26
##       1 10 19
##  Landa:  0.001  Etest:  0.2589928
```

```
cat("El mejor landa es ", landamej, " con un Etest de ", Etest,
    " que da la solución: w=[",wmej,"]\n")
```

```
## El mejor landa es  0  con un Etest de  0.2589928  que da la solución: w=[ 0.002842689 -0.0005390458
```

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Tras hacer regularización un un intervalo amplio de landas escogimos dentro del intervalo $[0, 0.001]$, que el valor óptimo de landa sea 0 para que el error en el conjunto de test fuera mínimo. Por lo que no haría falta usar regularización.

Aunque el modelo 2, da un menor error en el conjunto del test, obtenemos una peor matriz de confusión y nuestro objetivo es mantener un equilibradro entre ambos, por eso seleccionamos el modelo 1:

1. Donde obtenemos un E_{test} bajo.
2. Donde la matriz de confusión, predice menos falsos positivos, es decir, gente que está mala y le decimos que está bien.

Apartado 6

Definir los modelos a usar y estimar sus parámetros e hyperparámetros.

Vimos en el apartado anterior que usar regularización no es una buena idea ya que no pudimos reducir el error de los modelos que obtuvimos con Regresión Logística. Ya que el λ más óptimo para el modelo uno es 0. Por lo tanto, no podemos justificar el uso de la regularización.

Apartado 7

Selección y ajuste modelo final.

El modelo que elegimos es el obtenido mediante Regresión Logística con los siguientes parámetros y variables:

- *ldl* sin transformación.
- *age* sin transformación.
- *tobacco* sin transformación.

```r
ml6 = glm(chd ~ ldl + age + tobacco, family = binomial(logit), data = SAheart.train)

# Obtenemos los errores y la matriz de confusión
errorres_regresion_logistica(ml6)
```

```
##        Real
## predTst  0  1
##       0 81 24
##       1 13 21
```

```
## $Etest
## [1] 0.2661871
##
## $Ein
## [1] 0.2910217
```

Nos hemos decantando por el anterior modelo, puesto que es uno de los que menor error en el conjunto de test nos ha salido y donde obtenemos el menor número de casos de falsos positivos en la matriz de confusión.

```r
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 8

Estimacion del error Eout del modelo lo más ajustada posible.

Para estimar el error E_{out}, nos hemos basado en E_{test}, porque con la de E_{in}, obtendríamos una cota peor.

Vamos a obtener la cota E_{out} basada en E_{test}, para ello hacemos uso del libro *Learning from Data*, en concreto en la *página 40*, donde nos viene la ecuación que necesitamos, que está basada en la *desigualdad de Hoeffding* :

$$P[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2N\epsilon^2}$$

Resolviendo esta fórmula, llegamos a:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} ln\left(\frac{2M}{\delta}\right)}$$

Debemos cambiar E_{in} por E_{test}, ya que esta ecuación es para cuando E_{in} tiene un conjunto finito de hipótesis de tamaño M y en este caso, nuestro conjunto de hipótesis es infinito. Por eso, debemos tomar $M = 1$.

$$E_{out}(g) \leq E_{test}(g) + \sqrt{\frac{1}{2N} ln\left(\frac{2M}{\delta}\right)}$$

$$E_{out}(g) \leq E_{test}(g) + \sqrt{\frac{1}{2N} ln\left(\frac{2}{0.05}\right)}$$

```
datos.test = cbind(SAheart.test$ldl , SAheart.test$age, SAheart.test$tobacco,1)

# Obtenemos el Etest del anterior apartado
Etest = 0.2661871

# Obtenemos el tamaño de los datos
N <- nrow(datos.test)

# Calculamos el segundo término de la fórmula
x <- sqrt( (1/(2*N)) * log( 2 / 0.05) )

# Obtenemos el valor de Eout
cota_Etest_Eout <- Etest + x
cat ("Cota de E_out basada en E_test: ", cota_Etest_Eout)
```

```
## Cota de E_out basada en E_test:  0.3813798
```

Esta es la cota a nivel de confianza 95%, por lo tanto es una toleracia de 0.05. Se observa que E_{out} es mayor que E_{test} aunque sigue siendo un buen valor para dar por bueno el modelo seleccionado.

```
# Después de crear una gráfica o iniciar un apartado paramos la ejecución 3 segundos
Sys.sleep(3)
```

Apartado 9

Discutir y justificar la calidad del modelo encontrado y las razones por las que considera que dicho modelo es un buen ajuste que representa adecuadamente los datos muestrales.

Después de haber realizado los anteriores apartados hemos llegado a la conclusión de que el modelo es bueno por las siguientes razones:

- El modelo tiene una cota para E_{out} de 0.3813798, siendo así un error considerable fuera de la muestra.
- Su matriz de confusión es la mejor obtenida, ya que obtenemos en la diagonal inversa mejores resultados y cuyos valores son los que dan más penalización a nuestro modelo.
- No es un modelo complejo (no usa ninguna transformación) y usa pocas variables:
  - *ldl* sin transformación.
  - *age* sin transformación.
  - *tobacco* sin transformación.
- Es un modelo al que no hace falta aplicarle regularización para encontrar el valor óptimo para E_{test} que determina la cota de generalización que se da para E_{out}
- Los pesos solución del modelo lineal son: $w = (0.002859783, -0.0009940073, 0.00239286, 0.0002009144)$