Primera Práctica de ADEI

Laboratori 1 - Data Preparation

Alejandro Alarcón

10/19/2021

Contents

```
Loading required package: effects

Loading required package: carData

lattice theme set by effectsTheme()
See ?effectsTheme for details.

Loading required package: FactoMineR

Loading required package: car

Loading required package: factoextra

Loading required package: ggplot2

Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

Loading required package: RColorBrewer

Loading required package: dplyr

Attaching package: 'dplyr'

The following object is masked from 'package:car':

    recode

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

Loading required package: ggmap

Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

Please cite ggmap if you use it! See citation("ggmap") for details.

Loading required package: ggthemes

Loading required package: missMDA

Loading required package: epiDisplay

Loading required package: foreign

Loading required package: survival

Loading required package: MASS

Attaching package: 'MASS'

The following object is masked from 'package:dplyr':

    select

Loading required package: nnet

Attaching package: 'epiDisplay'

The following object is masked from 'package:ggplot2':

    alpha
```

1 Introducción al dataset

Echamos un vistazo al dataset

```
summary( df )
```

```
   model                year          price        transmission
 Length:5000        Min.   :2001   Min.   :   899   Length:5000
 Class :character   1st Qu.:2016   1st Qu.: 13995   Class :character
 Mode  :character   Median :2017   Median : 19498   Mode  :character
                    Mean   :2017   Mean   : 21207
                    3rd Qu.:2019   3rd Qu.: 25980
                    Max.   :2020   Max.   :109495
   mileage         fuelType              tax             mpg
 Min.   :     1   Length:5000        Min.   :  0.0   Min.   :  1.10
 1st Qu.:  5815   Class :character   1st Qu.:125.0   1st Qu.: 45.60
 Median : 17731   Mode  :character   Median :145.0   Median : 53.30
 Mean   : 23590                      Mean   :122.8   Mean   : 53.93
 3rd Qu.: 34130                      3rd Qu.:145.0   3rd Qu.: 61.40
 Max.   :178000                      Max.   :580.0   Max.   :470.80
  engineSize    manufacturer
 Min.   :0.000   Length:5000
 1st Qu.:1.500   Class :character
 Median :2.000   Mode  :character
 Mean   :1.909
 3rd Qu.:2.000
 Max.   :5.500
```

```
names( df )
```

```
 [1] "model"        "year"         "price"        "transmission" "mileage"
 [6] "fuelType"     "tax"          "mpg"          "engineSize"   "manufacturer"
```

1.1 Transformación de variables categóricas a factores

```
#Model
df$model <- factor(paste0(df$manufacturer, "-", df$model))
head(levels(df$model)) #Algunos de los valores para el factor modelo
```

```
[1] "Audi- A1" "Audi- A3" "Audi- A4" "Audi- A5" "Audi- A6" "Audi- A7"
```

```
#Transmission
df$transmission <- factor( df$transmission )
levels( df$transmission )
```

```
[1] "Automatic" "Manual"    "Semi-Auto"
```

```
df$transmission <- factor( df$transmission, levels = c("Manual","Semi-Auto","Automatic"),labels = paste0
head( df )
```

```
      model year price      transmission mileage fuelType tax  mpg engineSize
1  Audi- A1 2017 12500    f.Trans-Manual   15735   Petrol 150 55.4        1.4
6  Audi- A1 2016 13900 f.Trans-Automatic   32260   Petrol  30 58.9        1.4
9  Audi- A3 2015 10200    f.Trans-Manual   46112   Petrol  20 60.1        1.4
23 Audi- A5 2017 22500 f.Trans-Automatic   21649   Diesel 145 58.9        3.0
25 Audi- Q5 2016 20000 f.Trans-Automatic   23789   Diesel 200 47.1        2.0
38 Audi- A6 2016 19400 f.Trans-Automatic   34030   Diesel 125 58.9        2.0
   manufacturer
1          Audi
6          Audi
9          Audi
23         Audi
25         Audi
38         Audi
```

```
#FuelType
df$fuelType <- factor(df$fuelType)
levels(df$fuelType)
```

```
[1] "Diesel" "Hybrid" "Other"  "Petrol"
```

```
df$fuelType <- factor( df$fuelType, levels = c("Diesel","Petrol","Hybrid"), labels = paste0("f.Fuel-",c

#Manufacturer
df$manufacturer <- factor(df$manufacturer)
levels(df$manufacturer)
```

```
[1] "Audi"     "BMW"      "Mercedes" "VW"
```

1.2 Transformación de variables numéricas a factores

```
#Year + Age
summary(df$year)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2001    2016    2017    2017    2019    2020
```

```
df$age <-  2021 - df$year
df$year<-factor(df$year)
summary(df$age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   4.000   3.843   5.000  20.000
```

```
#EngineSize
summary(df$engineSize)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.500   2.000   1.909   2.000   5.500
```

```
df$engineSize <- factor(df$engineSize)
table(df$engineSize)
```

```
   0    1  1.2  1.3  1.4  1.5  1.6  1.8  1.9    2  2.1  2.2  2.3  2.5  2.7  2.9
   9  365  147   63  310  554  345   38    1 2142  412   20    4    7    1   10
   3  3.2  3.5  3.7    4  4.2  4.4  4.7    5  5.5
 512    3    2    1   36    4    7    2    1    4
```

1.3 Exploración de las variables

1.3.1 Factores

```
par(mfrow = c(2, 2))

tab1(df$year)
```

```
df$year :
        Frequency Percent Cum. percent
2001            3     0.1          0.1
2002            1     0.0          0.1
2003            1     0.0          0.1
2004            5     0.1          0.2
2005            3     0.1          0.3
2006            9     0.2          0.4
2007            7     0.1          0.6
2008            5     0.1          0.7
2009            9     0.2          0.9
2010           14     0.3          1.1
2011           16     0.3          1.5
2012           38     0.8          2.2
2013          129     2.6          4.8
2014          217     4.3          9.1
2015          424     8.5         17.6
2016          885    17.7         35.3
2017          895    17.9         53.2
2018          460     9.2         62.4
2019         1563    31.3         93.7
2020          316     6.3        100.0
  Total      5000   100.0        100.0
```

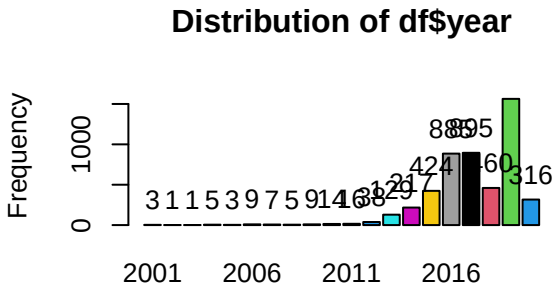
```
tab1(df$engineSize)
```

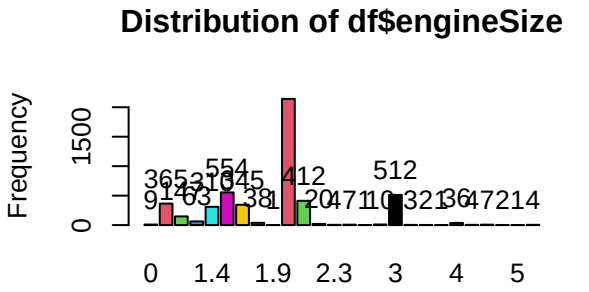
```
df$engineSize :
        Frequency Percent Cum. percent
0               9     0.2          0.2
1             365     7.3          7.5
1.2           147     2.9         10.4
1.3            63     1.3         11.7
1.4           310     6.2         17.9
1.5           554    11.1         29.0
1.6           345     6.9         35.9
1.8            38     0.8         36.6
1.9             1     0.0         36.6
2            2142    42.8         79.5
2.1           412     8.2         87.7
2.2            20     0.4         88.1
2.3             4     0.1         88.2
2.5             7     0.1         88.3
2.7             1     0.0         88.4
2.9            10     0.2         88.6
3             512    10.2         98.8
3.2             3     0.1         98.9
3.5             2     0.0         98.9
3.7             1     0.0         98.9
4              36     0.7         99.6
4.2             4     0.1         99.7
4.4             7     0.1         99.9
4.7             2     0.0         99.9
5               1     0.0         99.9
5.5             4     0.1        100.0
  Total      5000   100.0        100.0
```

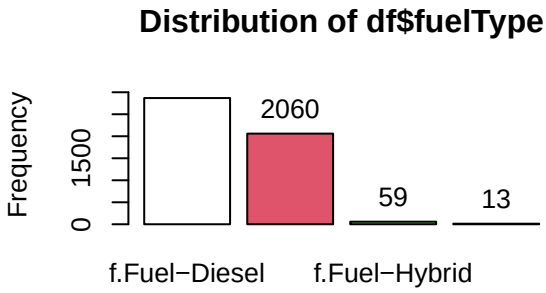
```
tab1(df$fuelType)
```

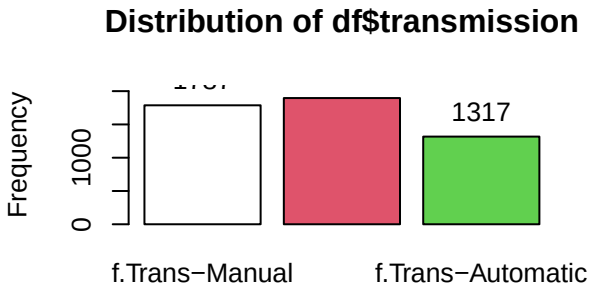
```
df$fuelType :
              Frequency   %(NA+)   %(NA-)
f.Fuel-Diesel      2868     57.4     57.5
f.Fuel-Petrol      2060     41.2     41.3
f.Fuel-Hybrid        59      1.2      1.2
NA's                 13      0.3      0.0
  Total            5000    100.0    100.0
```

```
tab1(df$transmission)
```









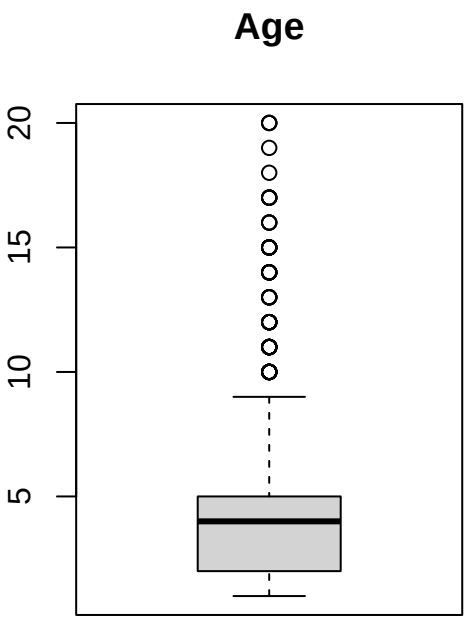

```
df$transmission :
                  Frequency Percent Cum. percent
f.Trans-Manual         1787    35.7         35.7
f.Trans-SemiAuto       1896    37.9         73.7
f.Trans-Automatic      1317    26.3        100.0
  Total                5000   100.0        100.0
```

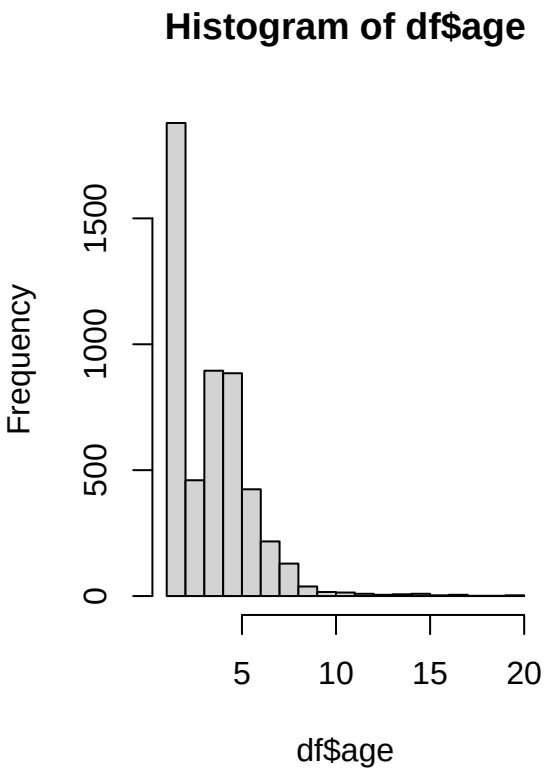
1.3.2 Variables numéricas

```
par(mfrow = c(1,2))
summary(df[c("age", "price", "mileage", "tax", "mpg")])
```

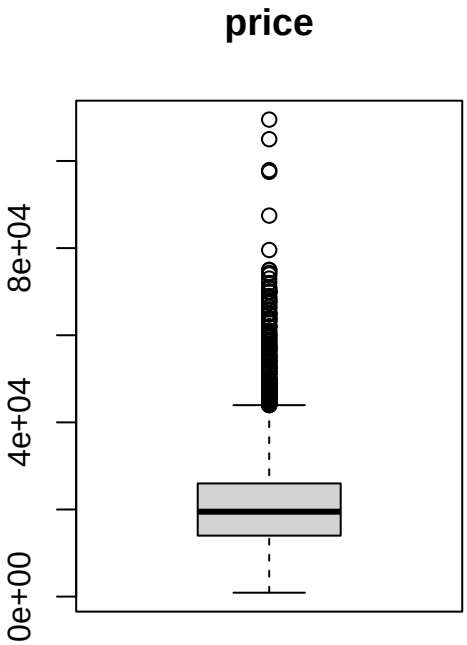
```
      age             price           mileage            tax       
 Min.   : 1.000   Min.   :   899   Min.   :     1   Min.   :  0.0  
 1st Qu.: 2.000   1st Qu.: 13995   1st Qu.:  5815   1st Qu.:125.0  
 Median : 4.000   Median : 19498   Median : 17731   Median :145.0  
 Mean   : 3.843   Mean   : 21207   Mean   : 23590   Mean   :122.8  
 3rd Qu.: 5.000   3rd Qu.: 25980   3rd Qu.: 34130   3rd Qu.:145.0  
 Max.   :20.000   Max.   :109495   Max.   :178000   Max.   :580.0  
      mpg        
 Min.   :  1.10  
 1st Qu.: 45.60  
 Median : 53.30  
 Mean   : 53.93  
 3rd Qu.: 61.40  
 Max.   :470.80  
```

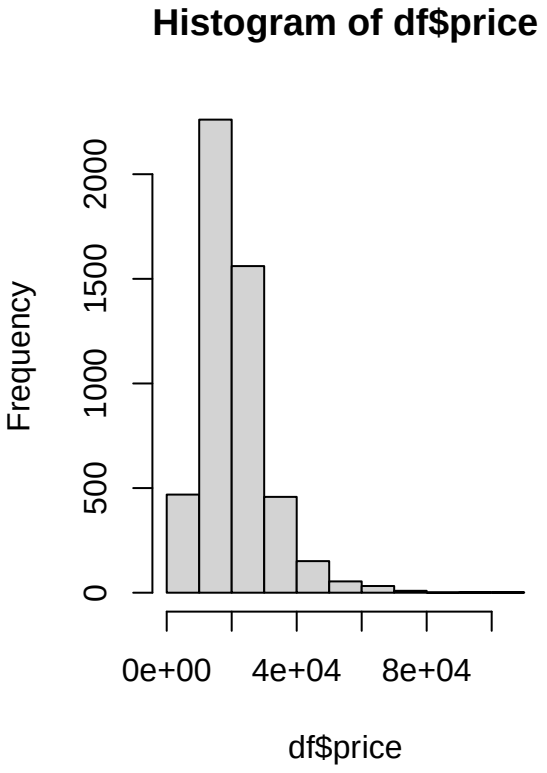
```
boxplot( df$age, main="Age" )
hist( df$age )
```



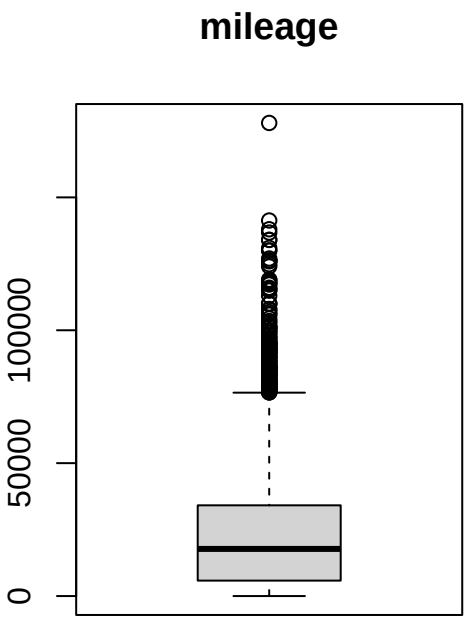


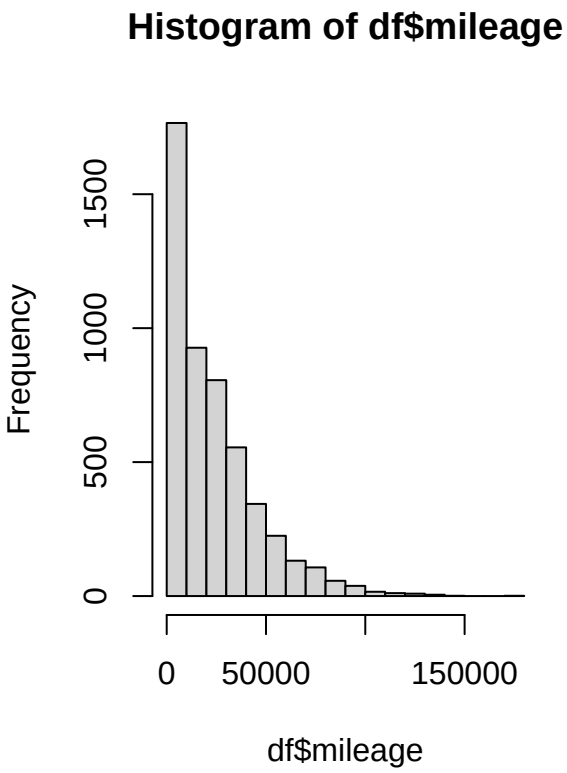

```
boxplot( df$price, main="price" )
hist( df$price )
```



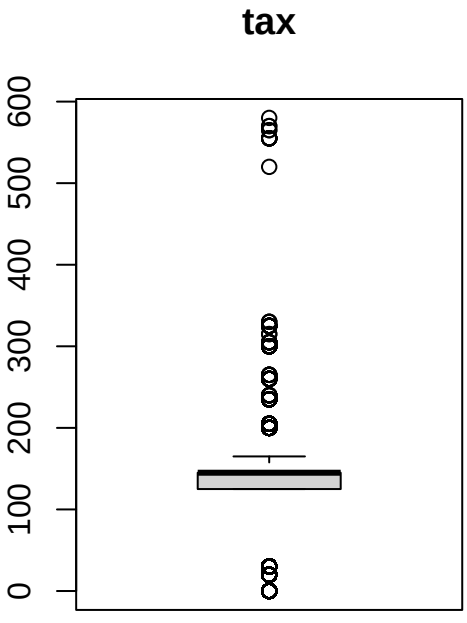


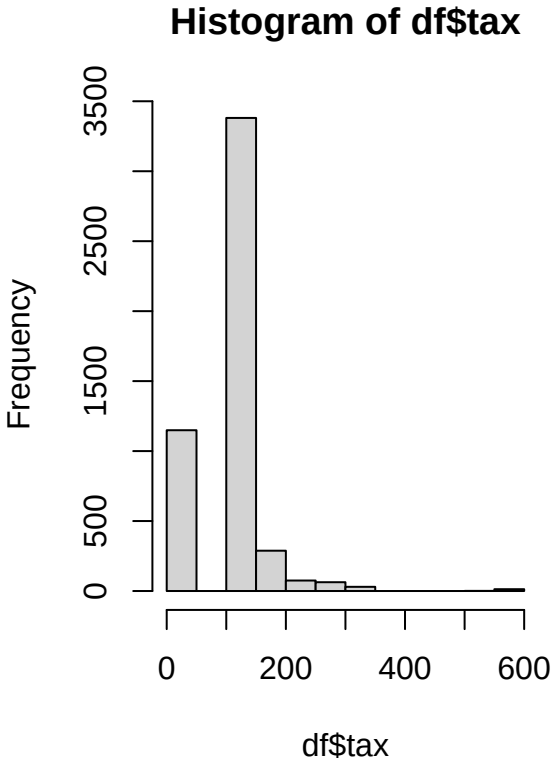

```
boxplot( df$mileage,main="mileage" )
hist( df$mileage )
```



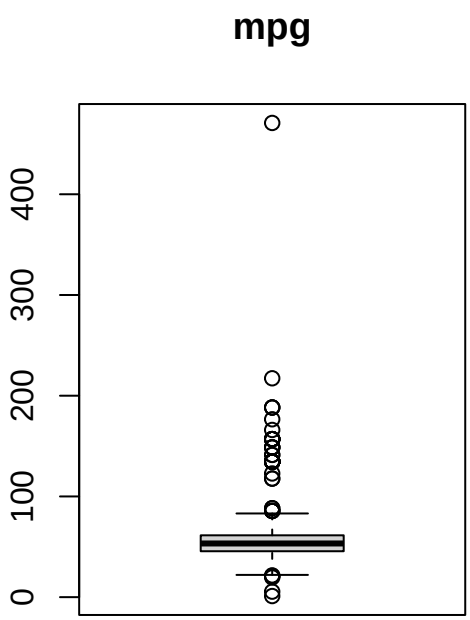


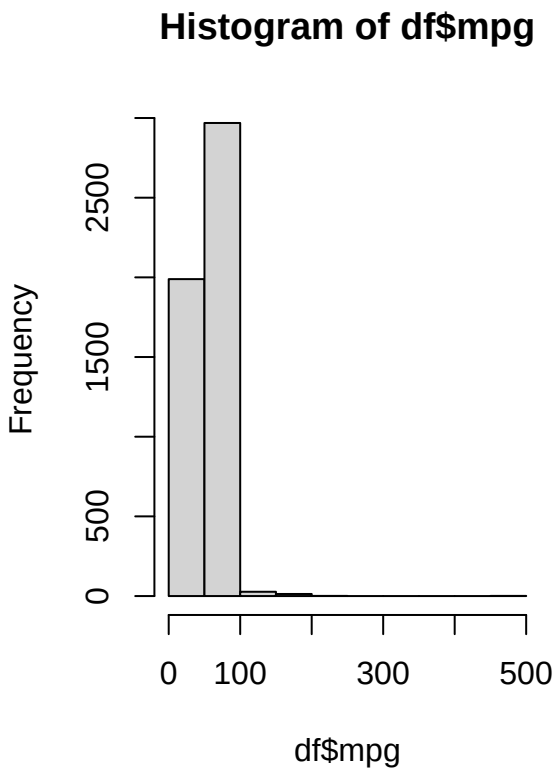

```
boxplot( df$tax, main="tax")
hist(df$tax)
```






```
boxplot( df$mpg, main="mpg")
hist(df$mpg)
```





Funciones útiles

```
calcQ <- function(x) {
  s.x <- summary(x)
  iqr<-s.x[5]-s.x[2]
  list(souti=s.x[2]-3*iqr, mouti=s.x[2]-1.5*iqr, min=s.x[1], q1=s.x[2], q2=s.x[3],
       q3=s.x[5], max=s.x[6], mouts=s.x[5]+1.5*iqr, souts=s.x[5]+3*iqr ) }

countNA <- function(x) {
  mis_x <- NULL
  for (j in 1:ncol(x)) {mis_x[j] <- sum(is.na(x[,j])) }
  mis_x <- as.data.frame(mis_x)
  rownames(mis_x) <- names(x)
  mis_i <- rep(0,nrow(x))
  for (j in 1:ncol(x)) {mis_i <- mis_i + as.numeric(is.na(x[,j])) }
  list(mis_col=mis_x,mis_ind=mis_i) }

countX <- function(x,X) {
  n_x <- NULL
  for (j in 1:ncol(x)) {n_x[j] <- sum(x[,j]==X) }
  n_x <- as.data.frame(n_x)
  rownames(n_x) <- names(x)
  nx_i <- rep(0,nrow(x))
  for (j in 1:ncol(x)) {nx_i <- nx_i + as.numeric(x[,j]==X) }
  list(nx_col=n_x,nx_ind=nx_i) }
```

2 Por cada variable

2.1 Conteo de missings

Iniciamos el recuento de missings creando dos estructuras de datos auxiliares y llamando a la función countNA. Como podemos apreciar, nos aparecene 13 individuos con valores de missing.

```
imis<-rep(0,nrow(df))  # rows - trips
jmis<-rep(0,2*ncol(df))  # columns - variables

mis1<-countNA(df)
imis<-mis1$mis_ind
inds <- which(imis > 0)
inds
```

```
 [1] 2123 2131 3487 4034 4056 4495 4496 4509 4682 4782 4783 4874 4875
```

Con el siguiente comando podemos ver todos los individuos que hemos seleccionado que contenian algún miss. En este caso, todos los miss están agrupados en la misma variable: fuelType. Podríamos pensar que esto podría ser debido a la existencia de coches eléctricos, pero como podemos apreciar, en algunos casos, el tipo de cambio es manual, hecho que no se aplica a los coches eléctricos.

```
df[inds,]
```

```
                model year price      transmission mileage fuelType tax   mpg
21018     BMW- 3 Series 2017 15300 f.Trans-Automatic   39428     <NA>   0 148.7
21110     BMW- 3 Series 2017 16000 f.Trans-Automatic   47495     <NA> 135 134.5
34344 Mercedes- C Class 2020 40999 f.Trans-Automatic     400     <NA> 135 217.3
39938          VW- Golf 2019 16889    f.Trans-Manual   12954     <NA> 150  45.6
40147          VW- Golf 2016 13795 f.Trans-Automatic   24463     <NA>  30  53.3
44517          VW- Polo 2019 12889    f.Trans-Manual   13016     <NA> 145  48.7
44524          VW- Polo 2019 13649    f.Trans-Manual    5000     <NA> 145  48.7
44653          VW- Polo 2019 14995 f.Trans-Automatic   10763     <NA> 145  45.6
46378        VW- Tiguan 2019 24999 f.Trans-Automatic    8491     <NA> 145  36.2
47541            VW- Up 2015  6799    f.Trans-Manual   28291     <NA>  20  62.8
47543            VW- Up 2020 10899    f.Trans-Manual    5000     <NA> 145  54.3
48422       VW- Touareg 2014 20995 f.Trans-Automatic   30523     <NA> 300  39.2
48423       VW- Touareg 2015 19995 f.Trans-Automatic   59115     <NA> 235  42.8
      engineSize manufacturer age
21018          2          BMW   4
21110          2          BMW   4
34344          2     Mercedes   1
39938          1           VW   2
40147        1.4           VW   5
44517          1           VW   2
44524          1           VW   2
44653          1           VW   2
46378        1.5           VW   2
47541          1           VW   6
47543          1           VW   1
48422          3           VW   7
48423          3           VW   6
```

Por último, como podemos ver en el recuento de misses por variable, se puede apreciar como todos lo misses se acumulan en la variable que hemos comentado anteriormente: fuelType.

```
mis1$mis_col # Number of missings for the current set of variables
```

```
            mis_x
model           0
year            0
```

```
price            0
transmission     0
mileage          0
fuelType        13
tax              0
mpg              0
engineSize       0
manufacturer     0
age              0
```

En conclusión, solo aparecen un total de 13 missing values en todo el dataframe. Si miramos por variables, estos 13 missings aparecen en la columna de fuelType. Por otro lado, y si miramos por individuos, podemos ver como para los 13 individuos que tienen missings, este está en la columna de fuelType.

2.2 Conteo de outliers

Iniciamos el recuento de outliers creando dos estructuras de datos auxiliares y una función que retornara los individuos con extreme outliers y mild outliers por separado.

```
iouts<-rep(0,nrow(df))  # rows - trips
jouts<-rep(0,2*ncol(df))  # columns - variables

# Funcion que recibe como parametro una columna y devuelve los ids de los individuos outlier
outliers_column <- function(x){
  outs <- NULL
  out_bounds <- calcQ(x)
  ex <- which((x<out_bounds$souti)|(x>out_bounds$souts))
  mild <- which(((x>out_bounds$souti)&(x<out_bounds$mouti))|((x<out_bounds$souts)&(x>out_bounds$mouts)))

  boxplot(x)
  abline(h=out_bounds$mouti,col="orange")
  abline(h=out_bounds$mouts,col="orange")
  abline(h=out_bounds$souti,col="red")
  abline(h=out_bounds$souts,col="red")

  outs <- rep(0,length(x))
  outs[mild] <- 1
  outs[ex]  <- 2
  if(length(ex)==0){
      outs <- factor(outs, labels=c("Non-Outlier","Mild-Outlier"))
  }else if(length(mild)==0){
      outs <- factor(outs, labels=c("Non-Outlier","Extreme-Outlier"))
  }else{
    outs <- factor(outs, labels=c("Non-Outlier","Mild-Outlier","Extreme-Outlier"))
  }
  list(extreme=ex, mild=mild, outs=outs)
}
```

Aplicaremos la función definida previamente a nuestras columnas numéricas.

```
# Estos son los outliers tanto mild como extreme de las variables numéricas
par(mfrow = c(1, 5))
outs_price <- outliers_column(df$price)
length(outs_price$mild) + length(outs_price$extreme)
```

```
[1] 193
```

```
outs_mileage <- outliers_column(df$mileage)
length(outs_mileage$mild) + length(outs_mileage$extreme)
```

```
[1] 171
```

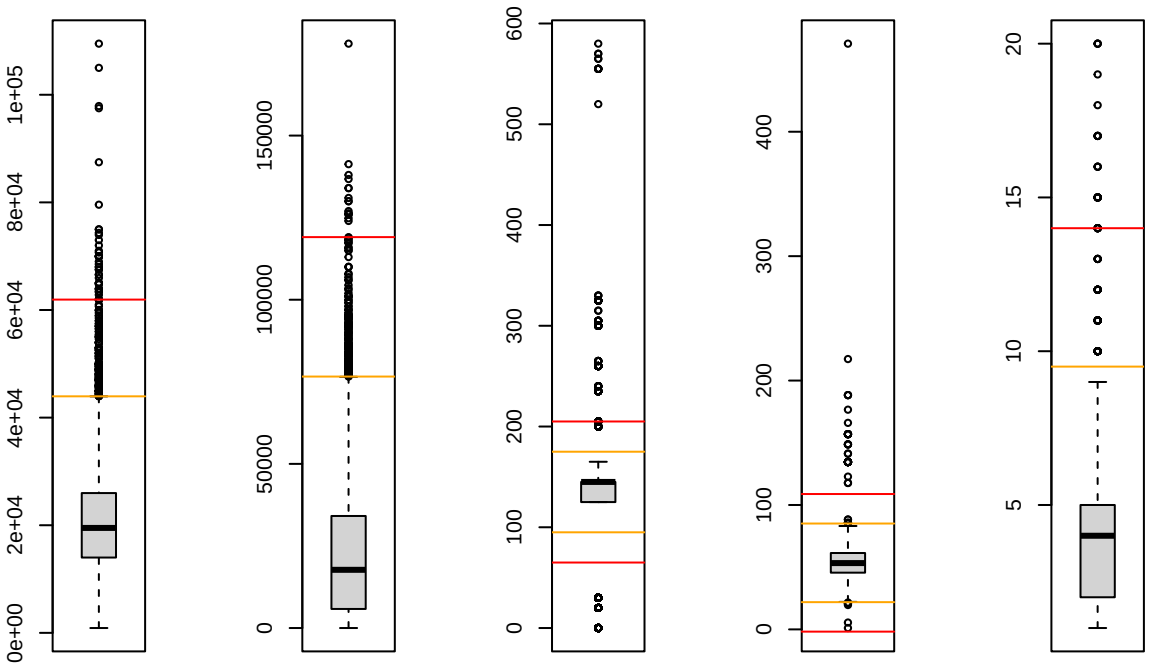
```
outs_tax <- outliers_column(df$tax)
length(outs_tax$mild) + length(outs_tax$extreme)
```

```
[1] 1425
```

```
outs_mpg <- outliers_column(df$mpg)
length(outs_mpg$mild)
```

```
[1] 12
```

```
outs_age <- outliers_column(df$age)
```




```
length(outs_age$mild) + length(outs_age$extreme)
```

```
[1] 66
```

Generamos la suma de todos los outliers.

```
df$outs <- rep(0,nrow(df))
df$outs[which(outs_mileage$outs!="Non-Outlier")] <- df$outs[which(outs_mileage$outs!="Non-Outlier")] + 1
df$outs[which(outs_price$outs!="Non-Outlier")] <- df$outs[which(outs_price$outs!="Non-Outlier")] + 1
df$outs[which(outs_tax$outs!="Non-Outlier")] <- df$outs[which(outs_tax$outs!="Non-Outlier")] + 1
df$outs[which(outs_mpg$outs!="Non-Outlier")] <- df$outs[which(outs_mpg$outs!="Non-Outlier")] + 1
df$outs[which(outs_age$outs!="Non-Outlier")] <- df$outs[which(outs_age$outs!="Non-Outlier")] + 1

summary(df$outs)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.3818  1.0000  4.0000
```

```
# Conteo de extreme outliers
sum_extreme_outliers <- length(outs_price$extreme) + length(outs_mileage$extreme) + length(outs_tax$extr
sum_extreme_outliers
```

```
[1] 1419
```

```
# Conteo de mild outliers
sum_mild_outliers <- length(outs_price$mild) + length(outs_mileage$mild) + length(outs_tax$mild) + lengt
sum_mild_outliers
```

```
[1] 490
```

Como podemos apreciar, una vez realizado el sumatorio de los diferentes tipos de outliers para todo el dataframe, vemos que el número de extreme outliers es mayor al de mild outliers.

Ponemos los extreme outliers como missings.

```
df[outs_mileage$extreme,"mileage"]<-NA
df[outs_tax$extreme,"tax"]<-NA
df[outs_mpg$extreme,"mpg"]<-NA
df[outs_age$extreme,"age"]<-NA
```

2.3 Conteo de errores

Procedemos al conteo de los errores.

Para este apartado, en la mayoría de los casos, simplemente comprobamos que los valores insertados no sean negativos o cero, aunque tal vez se podrían tener en cuenta otros casos si se examina el dataframe de manera técnica (i.e. consumos o distancias exageradamente elevados).

```
# Price
  err_price <- which(df$price<0)
  #df <- df[ -err_price, ]    #En este caso está vacío, así que no es necesario
  length(err_price)
```

```
[1] 0
```

```
# Age
  err_age <- which(df$age<0)
  df[err_age,"age"] <- NA
  length(err_age)
```

```
[1] 0
```

```
# Mileage
  err_mileage <- which(df$mileage<0)
  df[err_mileage,"mileage"] <- NA
  length(err_mileage)
```

```
[1] 0
```

```
# mpg
  err_mpg <- which(df$mpg<0)
  df[err_mpg,"mpg"] <- NA
  length(err_mpg)
```

```
[1] 0
```

```
# engineSize
  err_engineSize <- which(df$engineSize == "0")
  df[err_engineSize,"engineSize"] <- NA
  length(err_engineSize)
```

```
[1] 9
```

```
# tax
  err_tax <- which(df$tax<0)
  df[err_tax,"tax"] <- NA
  length(err_tax)
```

```
[1] 0
```

3 Imputación

A continuación, vamos a proceder a la imputación de los missings, errors y extreme outliers que hemos encontrado previamente.

En primer lugar, separaremos nuestras variables en numéricas y categóricas. También declaramos price y manufacturer como variables respuesta.

```
library(missMDA)
names(df)
```

```
 [1] "model"        "year"         "price"        "transmission" "mileage"
 [6] "fuelType"     "tax"          "mpg"          "engineSize"   "manufacturer"
[11] "age"          "outs"
```

```
vars_num <- names(df)[c(5,7,8,11)]
vars_cat <- names(df)[c(1,2,4,6,9)]
vars_res <- names(df)[c(3,10)]
```

3.1 Imputación de variables numéricas

Podemos echar un pequeño vistazo a nuestras variables numéricas.

```
summary(df[,vars_num])
```

```
    mileage            tax             mpg              age        
 Min.   :     1   Min.   :125    Min.   : 1.10   Min.   : 1.000  
 1st Qu.:  5800   1st Qu.:145    1st Qu.:45.60   1st Qu.: 2.000  
 Median : 17632   Median :145    Median :53.30   Median : 4.000  
 Mean   : 23238   Mean   :147    Mean   :53.07   Mean   : 3.786  
 3rd Qu.: 34000   3rd Qu.:145    3rd Qu.:61.40   3rd Qu.: 5.000  
 Max.   :119000   Max.   :205    Max.   :88.30   Max.   :14.000  
 NA's   :16       NA's   :1297   NA's   :42      NA's   :22      
```

Aplicamos la función imputePCA con 2 componentes primarias. (Teniendo en cuenta el PCA que realizamos en laboratorios posteriores, dos componentes son suficientes para aglutinar más del 80% de la variabilidad).

```
res.impca <- imputePCA(df[,vars_num], ncp = 2)
summary(res.impca$completeObs)
```

```
    mileage            tax              mpg              age        
 Min.   :     1   Min.   :125.0   Min.   : 1.1   Min.   : 1.000  
 1st Qu.:  5807   1st Qu.:145.0   1st Qu.:45.6   1st Qu.: 2.000  
 Median : 17706   Median :145.0   Median :53.3   Median : 4.000  
 Mean   : 23318   Mean   :146.9   Mean   :53.1   Mean   : 3.798  
 3rd Qu.: 34095   3rd Qu.:147.5   3rd Qu.:61.4   3rd Qu.: 5.000  
 Max.   :119000   Max.   :205.0   Max.   :88.3   Max.   :14.000  
```

Vamos a ver para cada variable, el valor que acaban recibiendo los individuos que hemos inputado:

```
head(res.impca$completeObs[ outs_age$extreme, "age" ])
```

```
    9880    10553    17953    19848    20506    20544 
5.904202 8.309153 8.394285 2.174381 4.318005 6.577762 
```

```
head(res.impca$completeObs[ outs_mileage$extreme, "mileage" ])
```

```
    18967     18980     19848     19860     19884     20904 
57572.874 59283.687  5550.296 37522.427 34222.870 47606.733 
```

```
head(res.impca$completeObs[ outs_tax$extreme, "tax" ])
```

```
       6        9       41       54       70       72 
146.2654 148.8502 147.3746 146.4441 139.2268 144.8021 
```

```
head(res.impca$completeObs[ outs_mpg$extreme, "mpg" ])
```

```
    5148     6308     8876    11341    11561    14452 
51.71888 51.10256 60.79850 55.06192 54.69687 59.37574 
```

Sustituimos en el dataframe original:

```
df[, vars_num] <- res.impca$completeObs
```

Como podemos ver, en los valores que previamente teniamos NA, ahora aparecen los valores que se han obtenido de la imputación.

```
df[outs_age$extreme,"age"]
```

```
 [1] 5.904202 8.309153 8.394285 2.174381 4.318005 6.577762 6.740687 7.283228
 [9] 8.193843 4.372187 7.933087 6.730148 6.096680 8.645744 9.221560 4.991256
[17] 6.311744 8.904037 6.853158 9.699162 1.857385 2.500580
```

```
df[outs_mpg$extreme,"mpg"]
```

```
 [1] 51.71888 51.10256 60.79850 55.06192 54.69687 59.37574 52.87664 52.07685
 [9] 54.03751 60.34226 54.44925 54.29924 53.68512 54.49085 50.50866 52.70174
[17] 57.74811 56.76985 55.44259 55.47986 56.78424 55.18505 55.73419 55.05140
[25] 61.91662 54.44355 56.01543 55.67018 55.08581 56.05532 53.34436 55.86386
[33] 54.38946 58.29225 61.23032 51.08887 57.28788 57.63969 59.87276 58.42167
[41] 56.60836 57.04605
```

3.2 Imputación de factores

Vamos a empezar echando un vistazo al summary de las variables categóricas.

Podemos apreciar que en el caso de fuelType, nos aparecen 13 valores NA (los missings que hemos encontrado).

En el caso de engineSize, podemos ver que aparecen 9 NAs (los errores que hemos encontrado).

```
summary(df[,vars_cat])
```

```
               model           year                 transmission
 VW- Golf          : 471   2019   :1563   f.Trans-Manual   :1787
 Mercedes- C Class: 376   2017   : 895   f.Trans-SemiAuto :1896
 VW- Polo          : 337   2016   : 885   f.Trans-Automatic:1317
 BMW- 3 Series     : 274   2018   : 460
 Mercedes- A Class: 260   2015   : 424
 Mercedes- E Class: 201   2020   : 316
 (Other)           :3081   (Other): 457
          fuelType      engineSize
 f.Fuel-Diesel:2868   2      :2142
 f.Fuel-Petrol:2060   1.5    : 554
 f.Fuel-Hybrid:  59   3      : 512
 NA's          :  13   2.1    : 412
                      1      : 365
                      (Other):1006
                      NA's   :   9
```

Procedemos con la imputación:

```
res.immca <- imputeMCA(df[,vars_cat],ncp=10)
summary(res.immca$completeObs)
```

```
               model           year                 transmission
 VW- Golf          : 471   2019   :1563   f.Trans-Manual   :1787
 Mercedes- C Class: 376   2017   : 895   f.Trans-SemiAuto :1896
 VW- Polo          : 337   2016   : 885   f.Trans-Automatic:1317
 BMW- 3 Series     : 274   2018   : 460
 Mercedes- A Class: 260   2015   : 424
 Mercedes- E Class: 201   2020   : 316
 (Other)           :3081   (Other): 457
          fuelType      engineSize
 f.Fuel-Diesel:2875   2      :2149
 f.Fuel-Petrol:2066   1.5    : 554
 f.Fuel-Hybrid:  59   3      : 513
                      2.1    : 412
                      1      : 366
                      1.6    : 345
                      (Other): 661
```

Podemos ver como en el summary que acabamos de realizar, ya no aparecen valores NA para ninguna de las variables.

Sustituimos la salida de la imputación en nuestro dataframe:

```
df[,vars_cat]<-res.immca$completeObs
```

4 Discretización de variables numéricas en factores

Una vez hemos realizado la imputación, tanto de las variables numéricas como de los factores, vamos a proceder a la discretización de las variables numéricas.

En los casos de age o tax, esta discretización se ha hecho de manera pseudo-aleatoria, ya que la discretización por cuantiles no funcionaba bien debido a la distribución de estas variables.

```
vars_num
```

```
[1] "mileage" "tax"     "mpg"     "age"
```

Como hemos podido ver con anterioridad, las distribuciones de estas variables son bastante diferentes entre sí.

En el caso de price o mpg, por ejemplo, podemos ver que la distribución se aproxima más a una normal.

Por otro lado, en la variable mileage, esta se podría aproximar mejor con una distribuxión de chi-cuadrado o exponencial.

Finalmente, para la variable tax, podemos ver que existe una gran acumulación en valores cercanos a 145.

A continuación lo estudiaremos con más profundidad.

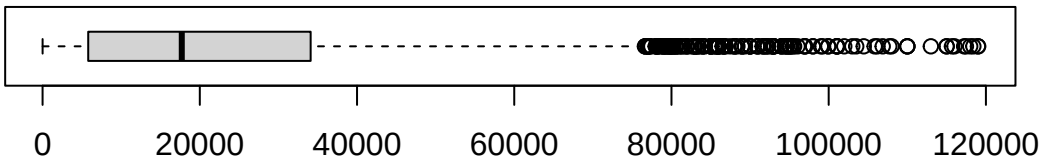
4.1 Mileage

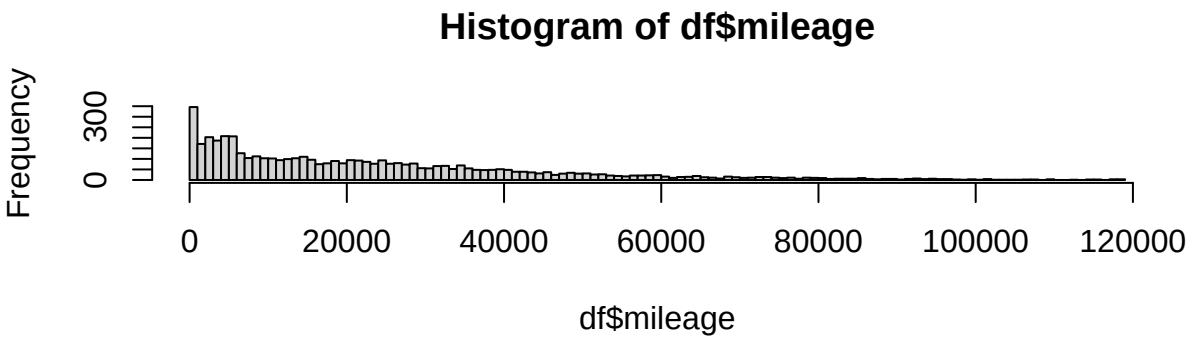
En primer lugar vamos a echar un vistazo a esta variable.

```
summary(df$mileage)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      1    5807   17706   23318   34095  119000 
```

```
par(mfrow = c(2,1))
boxplot(df$mileage,horizontal = TRUE)
hist(df$mileage, breaks=100)
```





Procedemos a la discretización según cuantiles, ya que en este caso si que era bastante representativa.

```
quants <- quantile(df$mileage,seq(0,1,0.25),na.rm=TRUE)

df$aux <- factor(cut(df$mileage, breaks=quants[1:5], include.lowest = TRUE))
df$f.miles<-factor(cut(df$mileage/1000,breaks=quants[1:5]/1000,include.lowest = TRUE ))
levels(df$f.miles)<-paste("f.miles-",levels(df$f.miles),sep="")
table(df$f.miles)
```

```
f.miles-[0.001,5.81]  f.miles-(5.81,17.7]  f.miles-(17.7,34.1] 
                1250                 1251                 1249 
  f.miles-(34.1,119] 
                1250 
```

```
#Esto cuadra ya que estamos tomando como referencia los quantiles y podemos ver como en todos hay el mis
```

Como podemos apreciar, aparecen 1250 elementos en cada cuantil, que es lo que esperariamos en condiciones ideales.

4.2 Tax

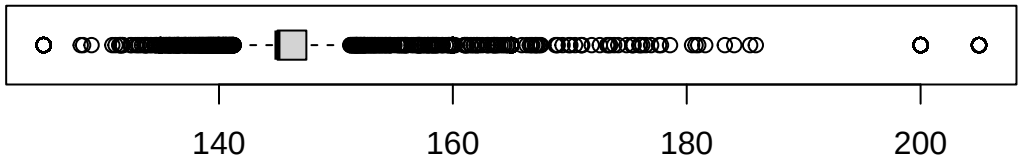
Para la variables tax la cosa se complica un poco.

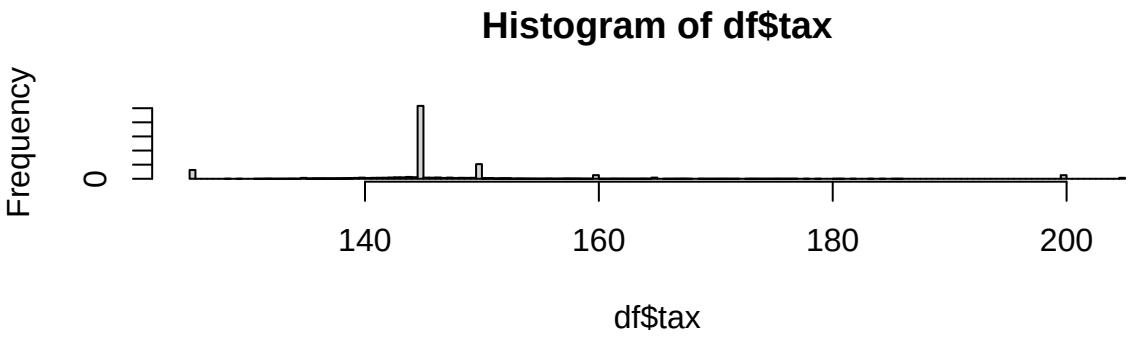
Si echamos un vistazo a algunos gráficos, podemos ver como hay una gran acumulación en valores entre 140 y 155. Esto hace que, al generar los cuantiles, el q1 y el q2 tengan el mismo valor, de modo que vamos a proceder con una discretización alternativa.

```
summary(df$tax)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  125.0   145.0   145.0   146.9   147.5   205.0
```

```
par(mfrow = c(2, 1))
boxplot(df$tax, horizontal = TRUE)
hist(df$tax, breaks=200)
```






```
df$aux <- factor(cut(df$tax, breaks=c(0,144,145,155,205), include.lowest = TRUE))
table(df$aux)
```

```
  [0,144] (144,145] (145,155] (155,205]
      918      2640       974       468
```

En la tabla podemos ver el gran pico en el valor 145, que acumula 2647 elementos, más de la mitad del tamaño de la muestra.

Por último, generamos una variable factor con etiquetas para los diferentes intervalos que hemos definido.

```
df$f.tax<-factor(cut(df$tax,breaks=c(0,144,145,155,205),include.lowest = TRUE ))
levels(df$f.tax)<-paste("f.tax-",levels(df$f.tax),sep="")
table(df$f.tax)
```

```
  f.tax-[0,144] f.tax-(144,145] f.tax-(145,155] f.tax-(155,205]
            918             2640             974             468
```

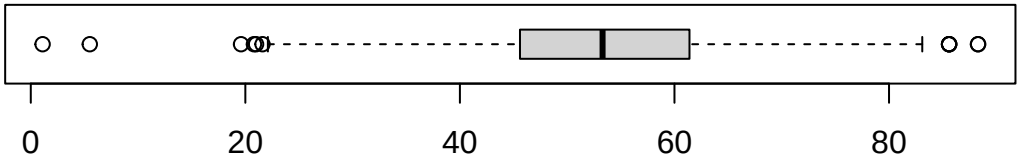
4.3 Mpg

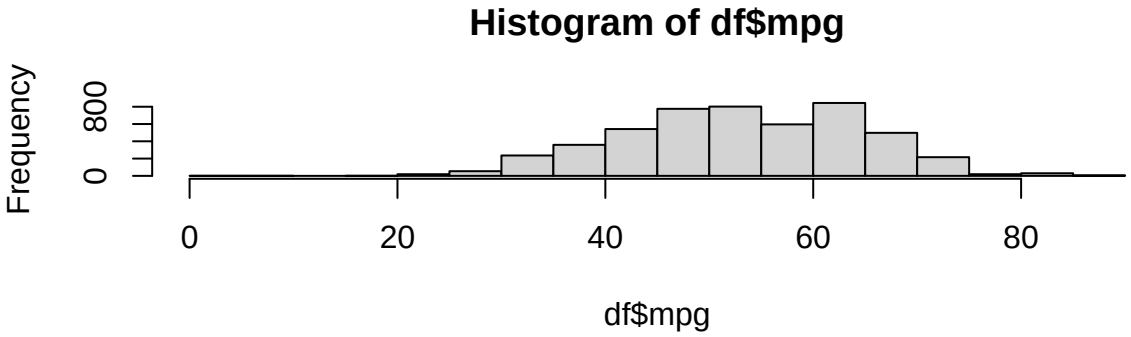
En el caso de la variable mpg, vamos a proceder con una discretización por cuantiles, ya que su distribución lo permite y el resultado es bastante explicativo junto con las etiquetas que vamos añadir al factor.

```
summary(df$mpg)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.1    45.6    53.3    53.1    61.4    88.3
```

```
par(mfrow = c(2, 1))
boxplot(df$mpg, horizontal = TRUE)
hist(df$mpg)
```






```
quants <- quantile(df$mpg,seq(0,1,0.25),na.rm=TRUE)

df$aux <- factor(cut(df$mpg, breaks=quants[1:5], include.lowest = TRUE))
df$f.mpg<-factor(cut(df$mpg,breaks=quants[1:5],include.lowest = TRUE ))
levels(df$f.mpg)<-paste("f.mpg-",c("muy bajo","bajo","medio","alto"),sep="")
table(df$f.mpg)
```

```
f.mpg-muy bajo     f.mpg-bajo    f.mpg-medio     f.mpg-alto
          1320           1327           1194           1159
```

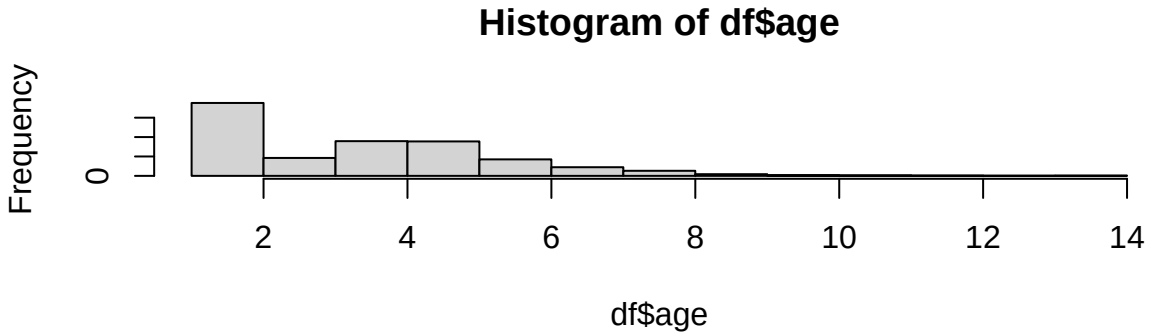
4.3.1 Age

Por último, la variable age, similarmente a la variable mileage, tiene una distribución que se acumula en valores bajos. De modo que vamos a aplicar una dicretización según los primeros cuantiles generando tres intervalos.

```
summary(df$age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   4.000   3.798   5.000  14.000
```

```
par(mfrow = c(2, 1))
boxplot(df$age,horizontal = TRUE)
hist(df$age)
```




```
quants <- quantile(df$age,seq(0,1,0.25),na.rm=TRUE)

df$aux <- factor(cut(df$age, breaks=c(quants[1:3],max(df$age)), include.lowest = TRUE))
summary(df$aux)
```

```
 [1,2]  (2,4] (4,14]
  1880   1357   1763
```

```
df$f.age<-factor(cut(df$age,breaks=c(quants[1:3],max(df$age)),include.lowest = TRUE ))
levels(df$f.age)<-paste("f.age-",c(levels(df$f.age)[1:2],"(+4)"),sep="")
table(df$f.age)
```

```
f.age-[1,2] f.age-(2,4]  f.age-(+4)
       1880        1357        1763
```

4.4 Generación del target categórico Audi

Vamos a proceder a generar nuestro target categórico.

Como indica la documentación de la práctica, este va a consistir en una variable que nos indique si el fabricante de un vehículo es Audi.

```
df$Audi<-ifelse(df$manufacturer == "Audi",1,0)
df$Audi<-factor(df$Audi,labels=paste("Audi",c("No","Yes")))
summary(df$Audi)
```

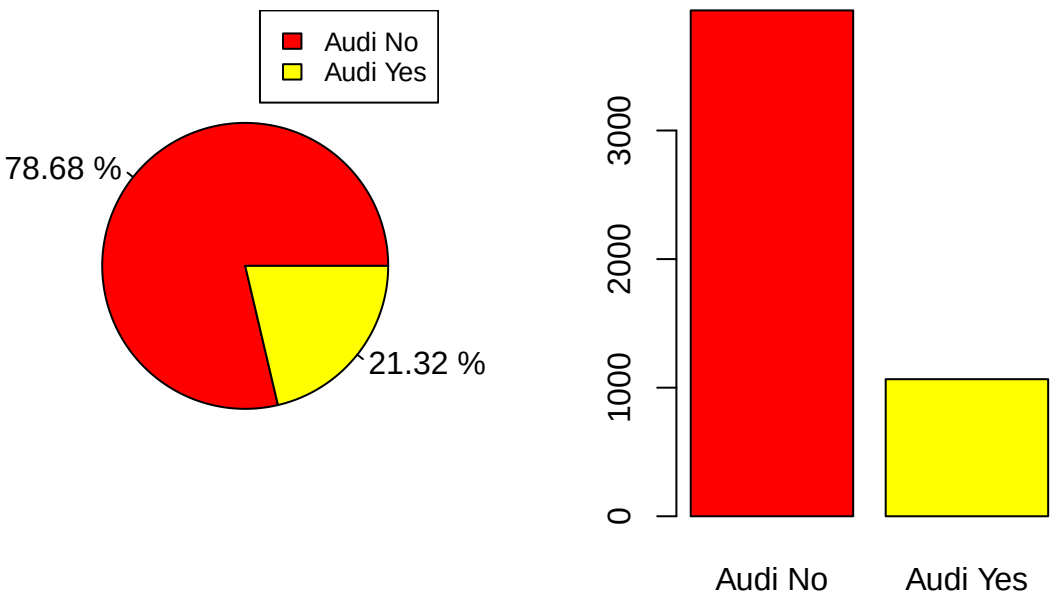
```
 Audi No Audi Yes
    3934     1066
```

A continuación algunos plots que nos ayudan a explicar esta varible.

```
par(mfrow = c(1, 2))
# Pie
piepercent<-round(100*(table(df$Audi)/nrow(df)),dig=2); piepercent
```

```
 Audi No Audi Yes
   78.68    21.32
```

```
pie(table(df$Audi),col=heat.colors(2),labels=paste(piepercent,"%"))
legend("topright", levels(df$Audi), cex = 0.8, fill = heat.colors(2))
# Bar Chart
barplot(table(df$Audi),col=c("red","yellow"))
```



5 Identificación los outliers multivariantes

En este apartado vamos a proceder con la identificación de los outliers multivariante. En primer lugar, vamos a cargar la librería y echaremos un primer vistazo a los posibles outliers.

```
library(mvoutlier)
```

```
Loading required package: sgeostat
```

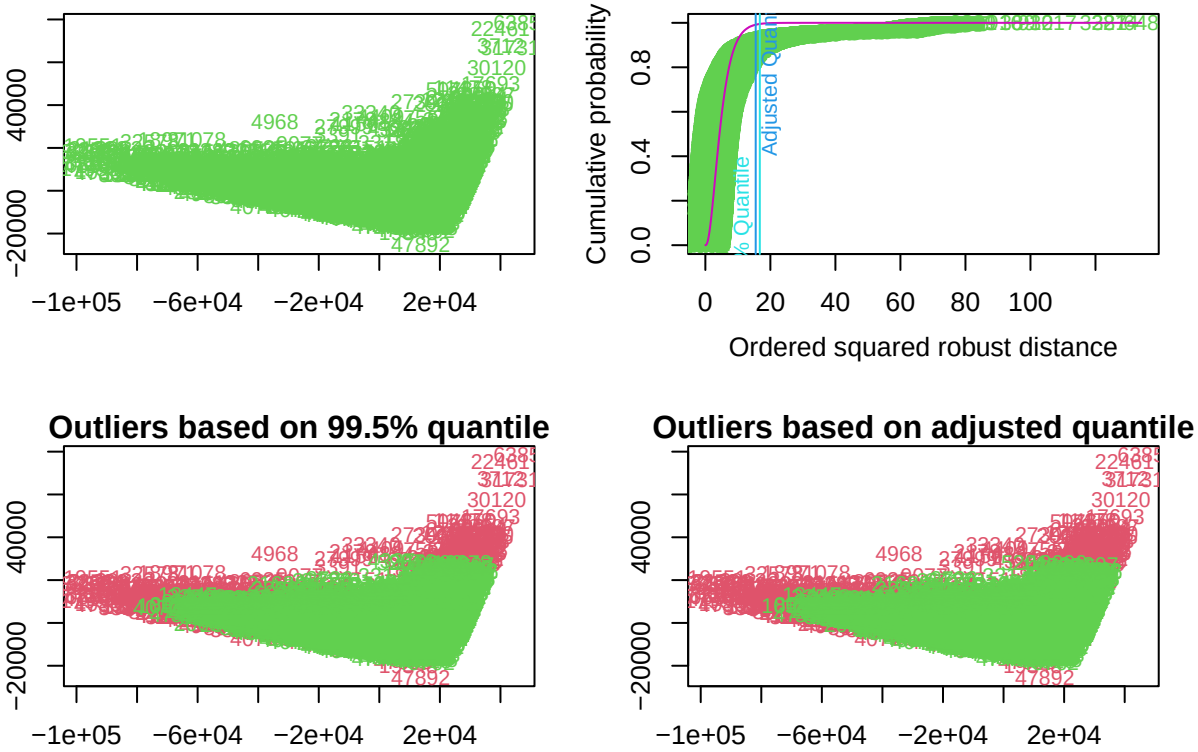
```
ll<-which(is.na(df$price)) #vacío
summary(df[,c(vars_res[1],vars_num)])
```

```
     price           mileage            tax             mpg        
 Min.   :   899   Min.   :     1   Min.   :125.0   Min.   : 1.1  
 1st Qu.: 13995   1st Qu.:  5807   1st Qu.:145.0   1st Qu.:45.6  
 Median : 19498   Median : 17706   Median :145.0   Median :53.3  
 Mean   : 21207   Mean   : 23318   Mean   :146.9   Mean   :53.1  
 3rd Qu.: 25980   3rd Qu.: 34095   3rd Qu.:147.5   3rd Qu.:61.4  
 Max.   :109495   Max.   :119000   Max.   :205.0   Max.   :88.3  
      age        
 Min.   : 1.000  
 1st Qu.: 2.000  
 Median : 4.000  
 Mean   : 3.798  
 3rd Qu.: 5.000  
 Max.   :14.000  
```

La ejecución de la función aq.plot (Adjusted Quantile) genera 4 gráficos. * En el de arriba a la izquierda podemos ver los datos originales. * En el de arriba a la derecha podemos ver la aproximación de estos datos a una distribucion de chi-cuadrado. * En el de abajo a la izquierda podemos ver los outliers determinados por el cuantil especificado de la chi-cuadrado (99.5%). * En el de abajo a la derecha podemos ver los outliers determinados por el Adjusted Quantile (99.5%).

```
mout<-aq.plot(df[,c(vars_res[1],vars_num)],delta=qchisq(0.995,5),quan=0.995)
```

```
Projection to the first and second robust principal components.
Proportion of total variation (explained variance): 0.9993919
```



A continuación, vamos a usar la función Moutlier para mostrar más información sobre los posibles outliers segun la distancia de Mahalanobis. Cabe destacar que el output de esta función nos devuelve tanto la distancia de Mahalanobis clásica como la robusta.

Esta función se debe aplicar a conjuntos de datos qus siguen una distribución normal, de modo que, según el estudio que hemos realizado previamente, solo lo aplicaremos a las variables de price, mileage, mpg y distance.

Disclaimer: Para las variables que menciono anteriormente, el programa funciona bien. Si añadimos tax, falla. Pero si nos fijamos en los histogramas/barplots de las variables que mencionamos, las únicas en las que podríamos suponer una normalidad en los datos serían price y mpg, ya que age o mileage tienden a una acumulación en valores bajos siguiendo distrubuciones que podrían ser parecidas a chi-cuadrado, log normal o exponencial.

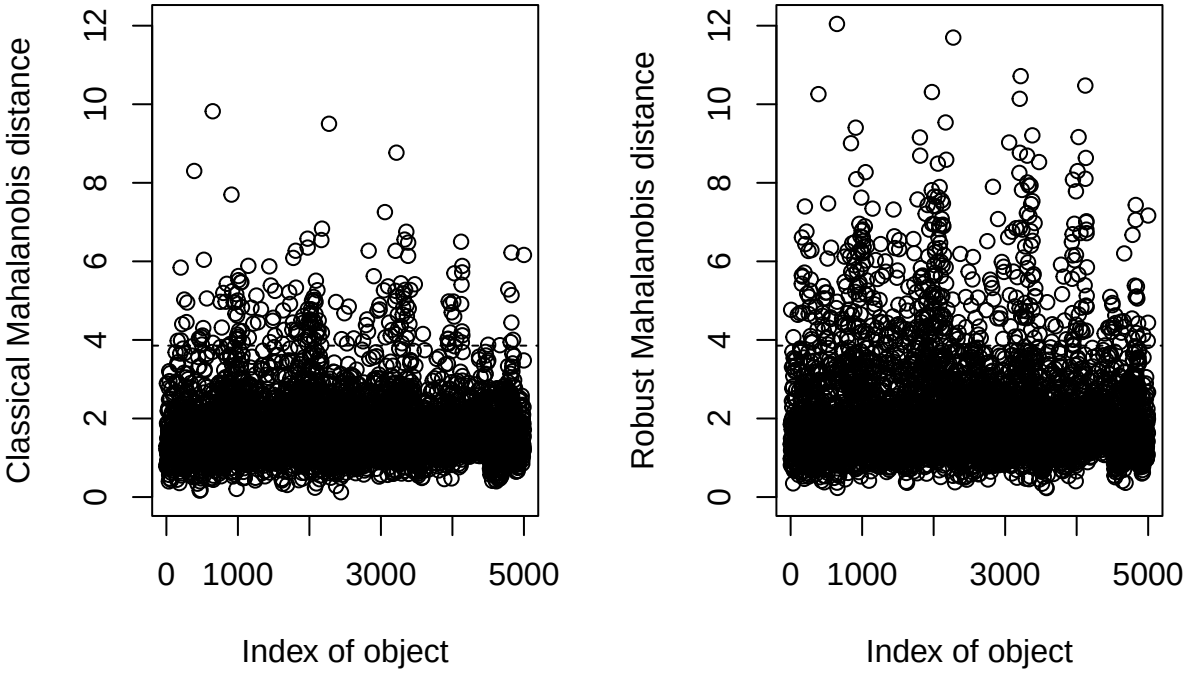
Seguimos con el análisis de Outliers Multivariantes.

En este caso, aplicaremos la función Moutlier a las variables que hemos mencionado anteriormente, mostrando las distancias de Mahalanobis clásica y robusta.

```
library(chemometrics)
```

```
Loading required package: rpart
```

```
mout<-Moutlier(df[,c("price","mileage","mpg","age")],quantile = 0.995, plot = TRUE)
```




```
ll<-which(mout$rd>5)
```

Vamos a echar un vistazo a las propiedades de las individuos considerados como Multivariant Outliers.

```
summary(df[ll,c("price","mileage","mpg","age")])
```

```
     price           mileage            mpg            age        
 Min.   :   899   Min.   :     1   Min.   :19.6   Min.   : 1.000  
 1st Qu.:  8000   1st Qu.: 30921   1st Qu.:33.6   1st Qu.: 4.000  
 Median : 12495   Median : 79651   Median :47.9   Median : 6.000  
 Mean   : 24749   Mean   : 65397   Mean   :48.7   Mean   : 6.009  
 3rd Qu.: 45895   3rd Qu.: 91707   3rd Qu.:62.8   3rd Qu.: 7.283  
 Max.   :109495   Max.   :119000   Max.   :83.1   Max.   :14.000  
```

Finalmente, crearemos una variable auxiliar que nos marcará, para cada individuo, si es Multivariant Outlier o no.

```
df$mout <- 0
df$mout[ ll ]<-1
df$mout <- factor( df$mout, labels=c( "NoMOut","YesMOut"))
table(df$mout)
```

```
 NoMOut YesMOut 
   4759     241 
```

6 Profiling

Por último, vamos a realizar el profiling de nuestro dataframe según nuestras variables target: Audi como variable categórica y price como variable numérica.

6.1 Target numérico (Price)

```
library(FactoMineR)
summary(df$price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    899   13995   19498   21207   25980  109495
```

```
vars_num <- c("mileage","tax","mpg","age")
vars_cat <- c("model","year","transmission","fuelType","engineSize","f.miles","f.tax","f.mpg","f.age","A
```

Vamos a proceder al profiling del target numérico a partir de la función condes del paquete FactoMineR

```
res.condes<-condes(df[,c("price",vars_num,vars_cat,"manufacturer")],1)
```

```
res.condes$quanti  # Global association to numeric variables
```

```
        correlation p.value
mileage  -0.5291971       0
mpg      -0.5799013       0
age      -0.5814226       0
```

Como podemos apreciar en la anterior salida, las variables numéricas que más correlación tienen con nuestra variable target (price) son mileage, mpg y age. En este caso, cabe destacar que se correlacionan de manera negativa. Esto indica que son inversamente proporcionales, es decir, que a más mileage/mpg/age, menor price (y viceversa).

```
res.condes$quali # Global association to factors
```

```
                     R2       p.value
model        0.498113659  0.000000e+00
year         0.368876026  0.000000e+00
engineSize   0.417317006  0.000000e+00
f.miles      0.302896034  0.000000e+00
f.mpg        0.286791156  0.000000e+00
f.age        0.331018734  0.000000e+00
transmission 0.222053178 3.454238e-273
f.tax        0.155741942 4.820306e-183
manufacturer 0.085248015  3.575883e-96
fuelType     0.005311573  1.663919e-06
Audi         0.004578046  1.678942e-06
```

En el caso de las variables categóricas (o factores), podemos apreciar que existe una clara relación entre los factores model, year, engineSize. También con algunos de los factores que hemos creado derivados de las variables numéricas. Vamos a analizarlo más a fondo:

Ahora nos fijaremos en las primeras lineas de la salida de condes$category:

```
head(res.condes$category)  # Partial association to significative levels in factors
```

```
                             Estimate       p.value
f.age=f.age-[1,2]             7516.899  0.000000e+00
f.mpg=f.mpg-muy bajo          9207.286  0.000000e+00
f.miles=f.miles-[0.001,5.81]  7983.258 6.380064e-232
year=2019                    16354.381 6.510724e-224
f.tax=f.tax-(144,145]         5858.741 9.173133e-174
engineSize=3                  8313.245 2.099854e-173
```

En esta salida podemos apreciar como para coches nuevos, y con un consumo o kilometraje bajos, el precio estimado es más alto. Como casos notables también podemos mencionar algunos casos de coches premium, para los que el valor estimado es más alto:

```
  model=Mercedes- G Class          45588.6851  5.645865e-07
  model=BMW- X7                    42925.9709  9.727955e-37
```

Y si nos fijamos en las últimas:

```
tail(res.condes$category)
```

```
                             Estimate       p.value
f.tax=f.tax-[0,144]         -4062.936  4.491610e-87
manufacturer=VW             -4880.202  2.247380e-94
f.mpg=f.mpg-alto            -5547.111 4.457754e-106
f.miles=f.miles-(34.1,119]  -7304.049 3.927777e-191
transmission=f.Trans-Manual -6839.754 4.814587e-273
f.age=f.age-(+4)            -6623.392 2.639305e-287
```

Podemos apreciar como factores como el cambio de marchas manual, un kilometraje alto o un cosumo alto tienden a abaratar el vehículo.

Algunos casos especiales son modelos de VW, que ven su precio realmente reducido:

```
  model=VW- CC                    -17098.5649  7.630716e-03
  model=VW- Beetle                -20721.5371  1.122196e-04
```

6.2 Target factor (AUDI)

Vamos a proceder a ejecutar la función catdes con para identificar las asociaciones hacia el target categórico que hemos generado. Para esto, y ya que simplemente considero que no es indicativo, no usaré las variable modelo ni manufacturer, ya que no aportan ningún tipo de infomación al análisis.

```
res.catdes<-catdes(df[,c("Audi",vars_num,vars_cat[2:9])],1)
```

Procedemos a ver las variables que parecen estar más correlacionadas con nuestra variable target:

```
res.catdes$quanti.var  # Global association to numeric variables
```

```
               Eta2      P-value
mpg     0.014005511 4.632358e-17
tax     0.003617358 2.084464e-05
age     0.001202533 1.419867e-02
mileage 0.001099604 1.903503e-02
```

Si lo analizamos un poco más las relaciones con variables numéricas, podemos ver que los vehículos Audi tienen consumos más bajos y kilometrages, antigüedad e impuestos más altos que la muestra que estudiamos. Si analizamos los vehículos que no son Audi, veremos lo contrario.

```
res.catdes$quanti # Partial association of numeric variables to levels of outcome factor
```

```
$`Audi No`
           v.test Mean in category Overall mean sd in category   Overall sd
mpg      8.367410        53.785122    53.097238      11.081567    11.166173
mileage -2.344551     22943.522121 23318.245205   21593.705113 21708.552196
age     -2.451828         3.762023     3.798202       1.984473     2.004245
tax     -4.252431       146.478947   146.865790      11.688011    12.355981
             p.value
mpg     5.889854e-17
mileage 1.904999e-02
```

```
age     1.421326e-02
tax     2.114622e-05

$`Audi Yes`
           v.test Mean in category Overall mean sd in category   Overall sd
tax      4.252431        148.29341   146.865790      14.468957    12.355981
age      2.451828          3.93172     3.798202       2.070118     2.004245
mileage  2.344551      24701.13508 23318.245205   22072.237729 21708.552196
mpg     -8.367410         50.55865    53.097238      11.110319    11.166173
             p.value
tax     2.114622e-05
age     1.421326e-02
mileage 1.904999e-02
mpg     5.889854e-17
```

Si estudiamos las varibles categóricas, podemos ver que los factores generados con las discretizaciones que hemos realizado anteriormente se relacionan de manera estrecha con nuestro target, además del engineSize, el fuelType o la transmission.

```
res.catdes$test.chi2 # Global association to factors
```

```
                  p.value df
engineSize   3.613671e-90 24
f.mpg        1.584246e-16  3
fuelType     1.129965e-07  2
transmission 1.413295e-04  2
f.tax        6.712343e-04  3
f.miles      1.622641e-03  3
f.age        3.813175e-02  2
```

Por último, vamos a entrar más en detalle en la relación de las variables categóricas con nuestro target.

```
res.catdes$category  # Partial association to significative levels in factors
```

```
$`Audi No`
                                Cla/Mod     Mod/Cla Global      p.value
engineSize=2.1                100.00000 10.47280122   8.24 9.327968e-46
engineSize=1.2                100.00000  3.73665480   2.94 2.714851e-16
f.mpg=f.mpg-alto               84.38309 24.86019319  23.18 2.960521e-08
engineSize=1.3                100.00000  1.60142349   1.26 2.471998e-07
engineSize=1.5                 85.74007 12.07422471  11.08 7.887020e-06
fuelType=f.Fuel-Diesel         80.83478 59.07473310  57.50 1.626774e-05
f.mpg=f.mpg-medio              82.99832 25.19064565  23.88 2.161084e-05
f.miles=f.miles-(5.81,17.7]    82.25420 26.15658363  25.02 3.048507e-04
transmission=f.Trans-SemiAuto  81.32911 39.19674631  37.92 3.253466e-04
fuelType=f.Fuel-Hybrid         94.91525  1.42348754   1.18 6.320005e-04
engineSize=2.2                100.00000  0.50838841   0.40 8.180915e-03
year=2019                      80.48624 31.97763091  31.26 3.475673e-02
engineSize=1                   82.78689  7.70208439   7.32 4.298931e-02
f.mpg=f.mpg-bajo               76.71439 25.87697001  26.54 4.257922e-02
f.miles=f.miles-(34.1,119]     76.48000 24.30096594  25.00 2.943481e-02
f.age=f.age-(+4)               76.68746 34.36705643  35.26 1.150630e-02
engineSize=2.9                 40.00000  0.10167768   0.20 1.006985e-02
engineSize=2.5                 28.57143  0.05083884   0.14 6.759033e-03
engineSize=4.2                  0.00000  0.00000000   0.08 2.056946e-03
engineSize=2                   76.22150 41.63701068  42.98 2.392416e-04
engineSize=4                   50.00000  0.45754957   0.72 1.564847e-04
f.tax=f.tax-(155,205]          71.58120  8.51550585   9.36 1.343933e-04
transmission=f.Trans-Manual    75.65753 34.36705643  35.74 1.111890e-04
engineSize=1.8                 47.36842  0.45754957   0.76 2.457585e-05
fuelType=f.Fuel-Petrol         75.21781 39.50177936  41.32 6.003236e-07
f.mpg=f.mpg-muy bajo           71.74242 24.07219115  26.40 2.195198e-12
```

```
engineSize=1.4                 45.16129  3.55871886   6.20 6.336915e-41
                                  v.test
engineSize=2.1                 14.198736
engineSize=1.2                  8.185363
f.mpg=f.mpg-alto                5.543756
engineSize=1.3                  5.159811
engineSize=1.5                  4.468228
fuelType=f.Fuel-Diesel          4.310783
f.mpg=f.mpg-medio               4.247563
f.miles=f.miles-(5.81,17.7]     3.611143
transmission=f.Trans-SemiAuto   3.594235
fuelType=f.Fuel-Hybrid          3.417496
engineSize=2.2                  2.644511
year=2019                       2.111181
engineSize=1                    2.023814
f.mpg=f.mpg-bajo               -2.027814
f.miles=f.miles-(34.1,119]     -2.177614
f.age=f.age-(+4)               -2.526934
engineSize=2.9                 -2.573421
engineSize=2.5                 -2.708489
engineSize=4.2                 -3.081884
engineSize=2                   -3.673510
engineSize=4                   -3.780546
f.tax=f.tax-(155,205]          -3.818265
transmission=f.Trans-Manual    -3.864781
engineSize=1.8                 -4.218660
fuelType=f.Fuel-Petrol         -4.991113
f.mpg=f.mpg-muy bajo           -7.021487
engineSize=1.4                -13.396516

$`Audi Yes`
                                  Cla/Mod    Mod/Cla Global      p.value
engineSize=1.4                 54.838710 15.9474672   6.20 6.336915e-41
f.mpg=f.mpg-muy bajo           28.257576 34.9906191  26.40 2.195198e-12
fuelType=f.Fuel-Petrol         24.782188 48.0300188  41.32 6.003236e-07
engineSize=1.8                 52.631579  1.8761726   0.76 2.457585e-05
transmission=f.Trans-Manual    24.342473 40.8067542  35.74 1.111890e-04
f.tax=f.tax-(155,205]          28.418803 12.4765478   9.36 1.343933e-04
engineSize=4                   50.000000  1.6885553   0.72 1.564847e-04
engineSize=2                   23.778502 47.9362101  42.98 2.392416e-04
engineSize=4.2                100.000000  0.3752345   0.08 2.056946e-03
engineSize=2.5                 71.428571  0.4690432   0.14 6.759033e-03
engineSize=2.9                 60.000000  0.5628518   0.20 1.006985e-02
f.age=f.age-(+4)               23.312535 38.5553471  35.26 1.150630e-02
f.miles=f.miles-(34.1,119]     23.520000 27.5797373  25.00 2.943481e-02
f.mpg=f.mpg-bajo               23.285607 28.9868668  26.54 4.257922e-02
engineSize=1                   17.213115  5.9099437   7.32 4.298931e-02
year=2019                      19.513756 28.6116323  31.26 3.475673e-02
engineSize=2.2                  0.000000  0.0000000   0.40 8.180915e-03
fuelType=f.Fuel-Hybrid          5.084746  0.2814259   1.18 6.320005e-04
transmission=f.Trans-SemiAuto  18.670886 33.2082552  37.92 3.253466e-04
f.miles=f.miles-(5.81,17.7]    17.745803 20.8255159  25.02 3.048507e-04
f.mpg=f.mpg-medio              17.001675 19.0431520  23.88 2.161084e-05
fuelType=f.Fuel-Diesel         19.165217 51.6885553  57.50 1.626774e-05
engineSize=1.5                 14.259928  7.4108818  11.08 7.887020e-06
engineSize=1.3                  0.000000  0.0000000   1.26 2.471998e-07
f.mpg=f.mpg-alto               15.616911 16.9793621  23.18 2.960521e-08
engineSize=1.2                  0.000000  0.0000000   2.94 2.714851e-16
engineSize=2.1                  0.000000  0.0000000   8.24 9.327968e-46
                                  v.test
engineSize=1.4                 13.396516
f.mpg=f.mpg-muy bajo            7.021487
fuelType=f.Fuel-Petrol          4.991113
engineSize=1.8                  4.218660
```

```
transmission=f.Trans-Manual     3.864781
f.tax=f.tax-(155,205]           3.818265
engineSize=4                    3.780546
engineSize=2                    3.673510
engineSize=4.2                  3.081884
engineSize=2.5                  2.708489
engineSize=2.9                  2.573421
f.age=f.age-(+4)                2.526934
f.miles=f.miles-(34.1,119]      2.177614
f.mpg=f.mpg-bajo                2.027814
engineSize=1                   -2.023814
year=2019                      -2.111181
engineSize=2.2                 -2.644511
fuelType=f.Fuel-Hybrid         -3.417496
transmission=f.Trans-SemiAuto  -3.594235
f.miles=f.miles-(5.81,17.7]    -3.611143
f.mpg=f.mpg-medio              -4.247563
fuelType=f.Fuel-Diesel         -4.310783
engineSize=1.5                 -4.468228
engineSize=1.3                 -5.159811
f.mpg=f.mpg-alto               -5.543756
engineSize=1.2                 -8.185363
engineSize=2.1                -14.198736
```

Vamos a interpretar un poco la salida anterior. Si cogemos este ejemplo,

```
'Audi No'
                               Cla/Mod     Mod/Cla Global      p.value     v.test
engineSize=2.1                100.00000 10.47280122   8.24 9.327968e-46  14.198736
```

podemos entender como el 100% de los vehiculos con engineSize=2.1, no están fabricados por Audi. También podemos extraer que el 10.47% de los vehículos no fabricados por Audi tienen engineSize=2.1.

Otro ejemplo:

```
'Audi Yes'
                                 Cla/Mod    Mod/Cla Global      p.value     v.test
transmission=f.Trans-Manual    24.342473 40.8067542  35.74 1.111890e-04   3.864781
```

En este caso podemos ver como el 24.34% de los vehículos con transmisión manual son Audi. Además, el 40.80% de los vehículos fabricados por Audi, tienen transmisión manual.

Añadimos la discretización de la variable price que necesitaremos para la segunda entrega en 7 levels:

```
summary(df$price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    899   13995   19498   21207   25980  109495
```

```
quants <- quantile(df$price,seq(0,1,0.125),na.rm=TRUE)

df$aux <- factor(cut(df$price, breaks=c(quants[1:8],max(df$price)), include.lowest = TRUE))
summary(df$aux)
```

```
      [899,1.1e+04]   (1.1e+04,1.4e+04]   (1.4e+04,1.7e+04]  (1.7e+04,1.95e+04]
                630                 641                 604                 625
 (1.95e+04,2.2e+04]   (2.2e+04,2.6e+04]  (2.6e+04,3.15e+04] (3.15e+04,1.09e+05]
                632                 619                 624                 625
```

```
df$f.price<-factor(cut(df$price,breaks=c(quants[1:8],max(df$price)),include.lowest = TRUE ))
levels(df$f.price)<-paste("f.price-",c(levels(df$f.price)),sep="")
table(df$f.price)
```

```
     f.price-[899,1.1e+04]   f.price-(1.1e+04,1.4e+04]
                       630                         641
 f.price-(1.4e+04,1.7e+04]  f.price-(1.7e+04,1.95e+04]
                       604                         625
f.price-(1.95e+04,2.2e+04]   f.price-(2.2e+04,2.6e+04]
                       632                         619
f.price-(2.6e+04,3.15e+04] f.price-(3.15e+04,1.09e+05]
                       624                         625
```