Tipologia y ciclo de vida de los datos - PRA2

Autor: Eduardo Diaz Villanueva e Ignasi Domingo González

Enero 2021

Contents

Descripción del dataset.

```
# Cargamos las diferentes librerias que utilizaremos.
packages = c('stringr', 'dplyr', 'ggplot2', 'corrplot', 'car', 'vcd', 'rpart', 'rpart.plot', 'tidyverse

for(p in packages){
  if(!require(p, character.only = T)){
    install.packages(p)
  }
  library(p, character.only = T)
}
```

```
## Loading required package: stringr
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
## Loading required package: ggplot2
```

```
## Loading required package: corrplot

## corrplot 0.84 loaded

## Loading required package: car

## Loading required package: carData

##
## Attaching package: 'car'

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

## Loading required package: vcd

## Loading required package: grid

## Loading required package: rpart

## Loading required package: rpart.plot

## Loading required package: tidyverse

## -- Attaching packages ---------------------------------------------------------------------------

## v tibble  3.0.0     v purrr   0.3.3
## v tidyr   1.0.2     v forcats 0.5.0
## v readr   1.3.1

## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x car::recode()   masks dplyr::recode()
## x purrr::some()   masks car::some()

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
# Limpiamos la aplicación de datos anteriores y cargo el fichero.
rm(list = ls())
datos <- read.csv("winequality-red.csv", sep=",")
datos_originales <- datos
# Vemos el tamaño del conjunto de datos
dim(datos)
```

```
## [1] 1599   12
```

```
# Visualizamos los primeros elementos del conjunto
head(datos,5)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
```

```
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                  34  0.9978 3.51      0.56     9.4
## 2                  25                  67  0.9968 3.20      0.68     9.8
## 3                  15                  54  0.9970 3.26      0.65     9.8
## 4                  17                  60  0.9980 3.16      0.58     9.8
## 5                  11                  34  0.9978 3.51      0.56     9.4
##   quality
## 1       5
## 2       5
## 3       5
## 4       6
## 5       5
```

El dataset seleccionado contiene 11 variables que describen las propiedades químicas de un vino, como puede ser la acidez, ph nivel de azúcar, etc... estas variables tendrán influencia en la calidad final del vino. Además contiene un atributo que es la calidad del vino, como ha sido clasificado.

Con este ejercicio queremos estudiar que variables son mas representativas y encontrar modelos que puedan predecir la calidad del vino.

Si pensamos por ejemplo en una industria, podríamos reducir el tiempo y coste reduciendo el numero de pruebas de calidad a las variables mas significativas. Incluso mejorar la calidad del producto final, focalizando esfuerzos y recursos a reducir la variabilidad de las variables que mas contribuyan a la calidad final.

Integracion y seleccion de los datos de interes

Realizaremos un primer análisis estadístico para familiarizarnos con las variables y sus tipos de datos.

```
#Resumen del conjunto de datos.
summary(datos)
```

```
##  fixed.acidity   volatile.acidity  citric.acid    residual.sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
##        pH          sulphates         alcohol         quality
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
```

```
# Visualización de los primeros valores de cada atributo.
str(datos)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

```
#Tipo de dato asignado a cada campo
sapply(datos, function(x) class(x))
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##            "numeric"            "numeric"            "numeric"
##       residual.sugar            chlorides  free.sulfur.dioxide
##            "numeric"            "numeric"            "numeric"
## total.sulfur.dioxide              density                   pH
##            "numeric"            "numeric"            "numeric"
##            sulphates              alcohol              quality
##            "numeric"            "numeric"            "integer"
```

Observamos que los tipos de datos asignados a las variables corresponden con las variables que representan. # Limpieza de los datos ## Elementos vacios Analizamos los valores de las variables para detectar falta o ausencia de datos

```
# Analizamos la existencia de datos NA
colSums(is.na(datos))
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##                    0                    0                    0
##       residual.sugar            chlorides  free.sulfur.dioxide
##                    0                    0                    0
## total.sulfur.dioxide              density                   pH
##                    0                    0                    0
##            sulphates              alcohol              quality
##                    0                    0                    0
```

```
# Analizamos la existencia de datos vacios
colSums(datos=="")
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##                    0                    0                    0
##       residual.sugar            chlorides  free.sulfur.dioxide
##                    0                    0                    0
## total.sulfur.dioxide              density                   pH
##                    0                    0                    0
##            sulphates              alcohol              quality
##                    0                    0                    0
```

```
# Analizamos la existencia de datos con valor 0
colSums(datos==0)
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##                    0                    0                  132
##       residual.sugar            chlorides  free.sulfur.dioxide
##                    0                    0                    0
## total.sulfur.dioxide              density                   pH
##                    0                    0                    0
##            sulphates              alcohol              quality
##                    0                    0                    0
```

Observamos la variable "Citric.acid" con una gran cantidad de valores 0. Tras realizar una pequeña investigación, (https://es.wikipedia.org/wiki/%C3%81cidos_en_el_vino#%C3%81cido_c%C3%ADtrico) podemos deducir que en la uva, el ácido cítrico es un componente presente de forma natural pero con muy baja presencia, lo que si no se añade posteriormente puede presentar valores de 0. Consideramos que dicha variable tiene valores correctos y no requiere de ninguna acción.

Elementos duplicados

Dado que no hay presentes registros con elementos vacios, vamos a verificar si hay registros duplicados.

```
# Analizamos la existencia de registros duplicados.
sum(duplicated(datos))
```

```
## [1] 240
```

Algunas filas están duplicadas. A modo de ejemplo, la fila 1 y la fila 5 son la misma.

```
# Ejemplo de registro duplicado
a <- datos %>% filter(row_number() == 1)
b <- datos %>% filter(row_number() == 5)
a == b
```

```
##      fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## [1,]          TRUE             TRUE        TRUE           TRUE      TRUE
##      free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## [1,]                TRUE                 TRUE    TRUE TRUE      TRUE    TRUE
##      quality
## [1,]    TRUE
```

Los registros duplicados no nos dan más información sobre la muestra, por lo que vamos a eliminar las filas duplicadas.

```
# Eliminamos los registros duplicados
datos <- datos[!duplicated(datos), ]
```

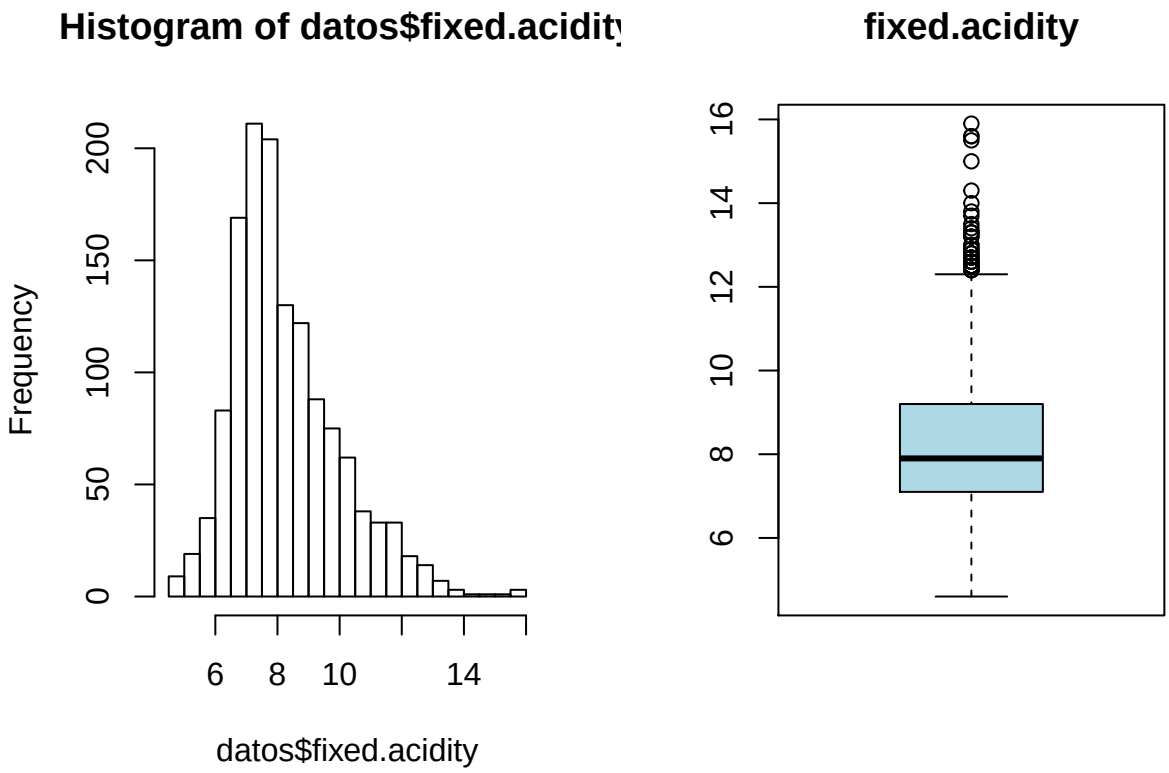
```
# Revisamos el tamaño de nuestro nuevo dataset
dim(datos)
```

```
## [1] 1359   12
```

Valores extremos

Analizaremos individualmente cada una de las variables focalizándonos en la distribución de los datos y sus valores extremos.

```
par(mfrow=c(1,2))
hist(datos$fixed.acidity, breaks  = 30)
boxplot(datos$fixed.acidity,main="fixed.acidity", col="lightblue")
```

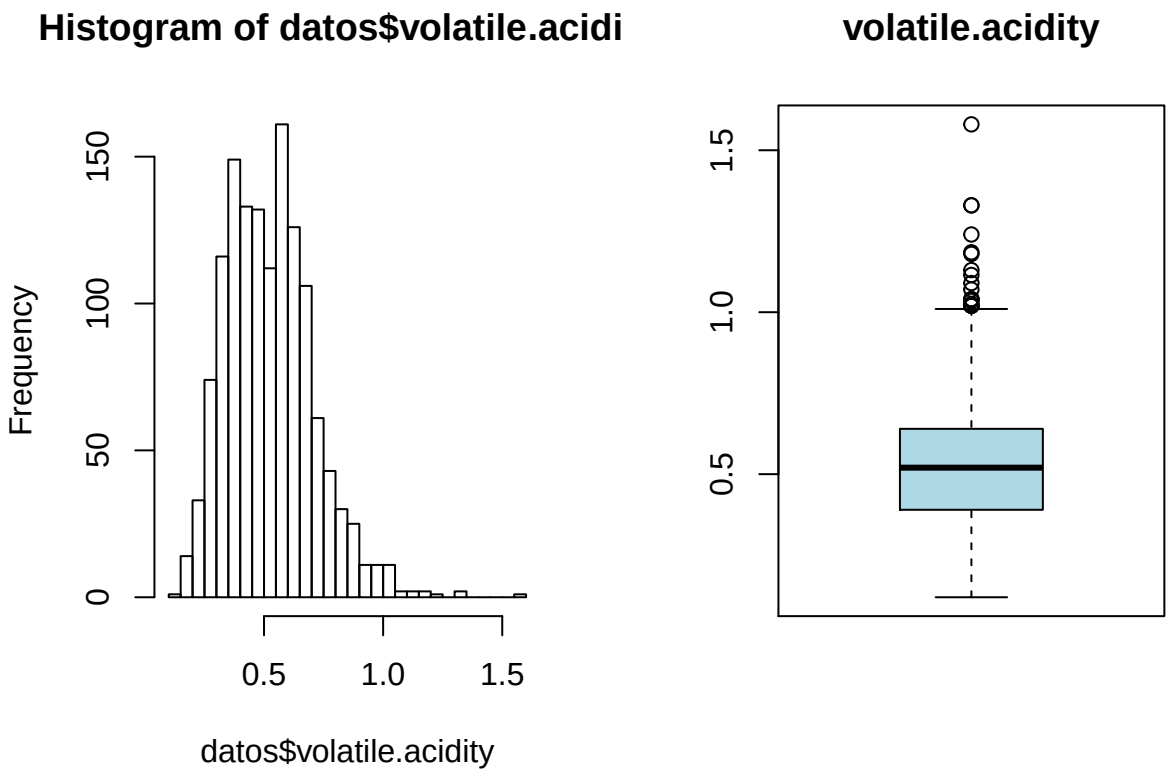



```
boxplot.stats(datos$fixed.acidity)$out
```

```
##  [1] 12.8 15.0 12.5 13.3 13.4 12.4 12.5 13.8 13.5 12.6 12.5 12.8 14.0 13.7 12.7
## [16] 12.5 12.8 12.6 15.6 12.5 13.0 12.5 13.3 12.4 12.5 12.9 14.3 12.4 15.5 15.6
## [31] 13.0 12.7 12.4 12.7 13.2 13.2 15.9 13.3 12.9 12.6 12.6
```

Observamos como el atributo “fixed.acidity” tiene 41 valores extremos, distribuidos entre 12.4 y 16

```
par(mfrow=c(1,2))
hist(datos$volatile.acidity, breaks  = 30)
boxplot(datos$volatile.acidity,main="volatile.acidity", col="lightblue")
```

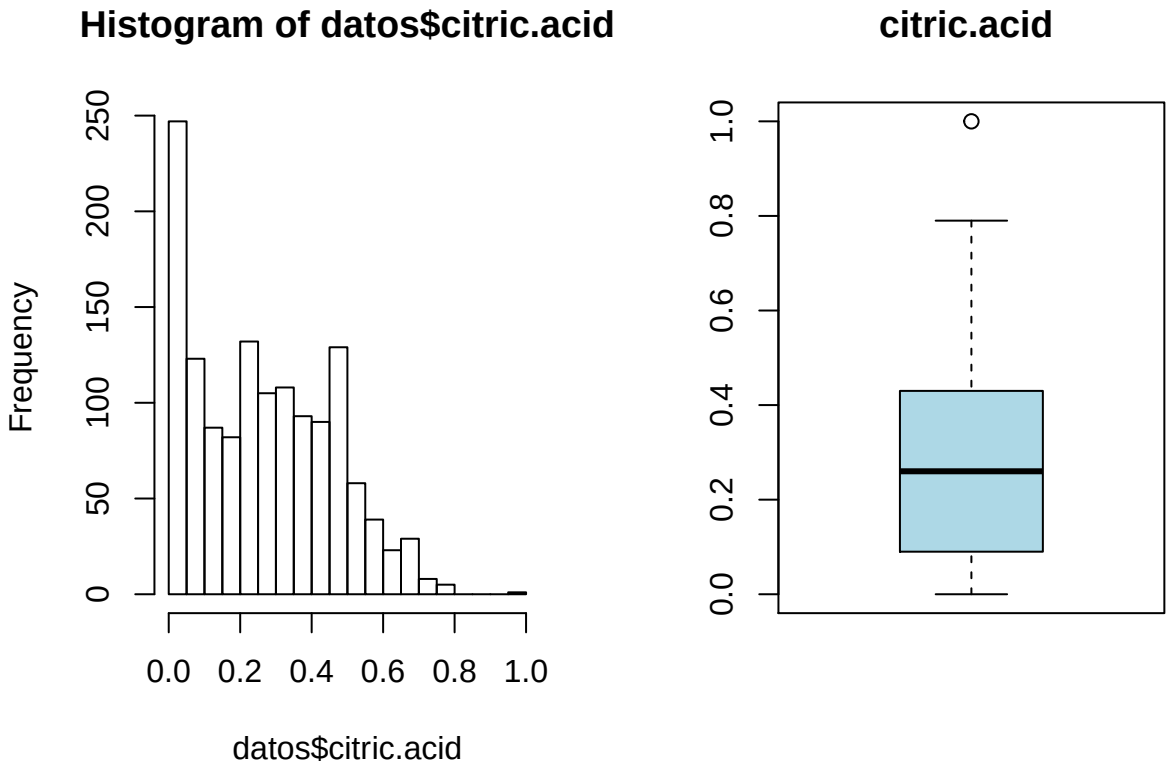



```
boxplot.stats(datos$volatile.acidity)$out
```

```
##  [1] 1.130 1.020 1.070 1.330 1.330 1.040 1.090 1.040 1.240 1.185 1.020 1.035
## [13] 1.025 1.115 1.020 1.020 1.580 1.180 1.040
```

Observamos como el atributo “volatile.acidity” tiene 19 valores extremos, distribuidos entre 1 y 1.6

```
par(mfrow=c(1,2))
hist(datos$citric.acid , breaks  = 30)
boxplot(datos$citric.acid ,main="citric.acid ", col="lightblue")
```

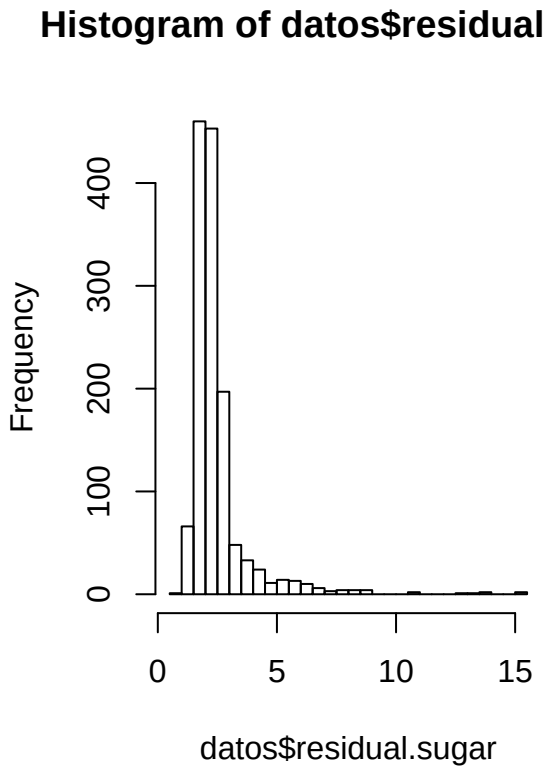


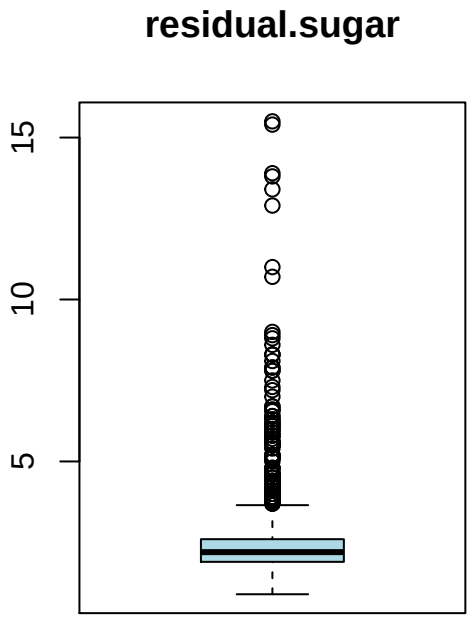

```
boxplot.stats(datos$citric.acid )$out
```

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

Observamos como el atributo “citric.acid” tiene un valor extremo de valor 1.

```
par(mfrow=c(1,2))
hist(datos$residual.sugar, breaks  = 30)
boxplot(datos$residual.sugar,main="residual.sugar", col="lightblue")
```



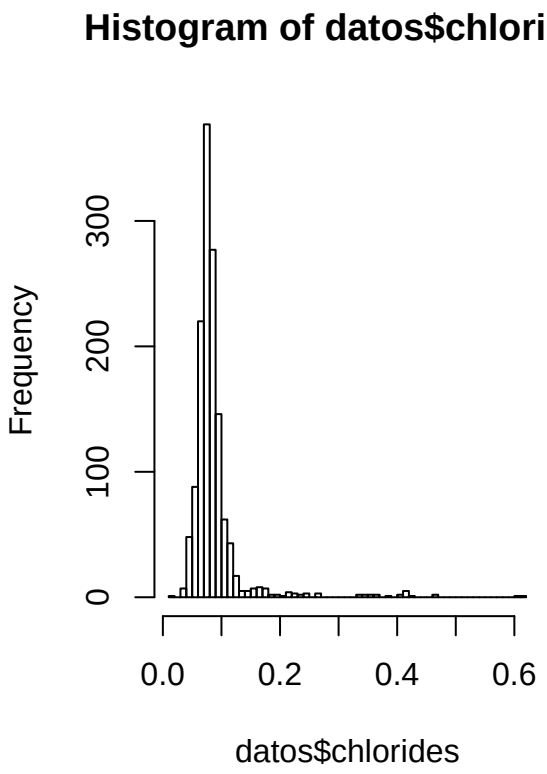


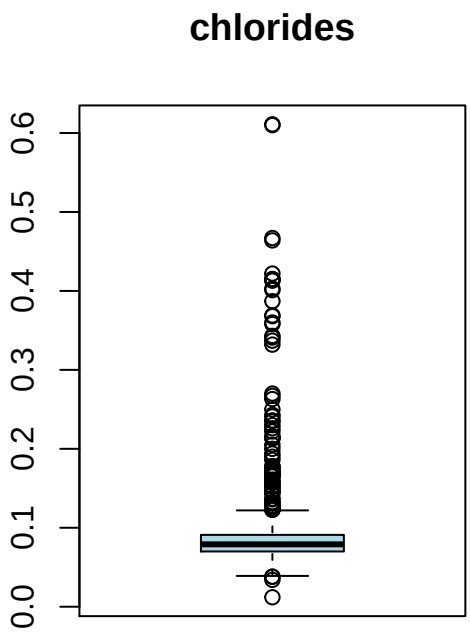

```
boxplot.stats(datos$residual.sugar)$out
```

```
##   [1]  6.10  3.80  3.90  4.40 10.70  5.50  5.90  3.80  5.10  4.65  5.50  5.50
##  [13]  7.30  7.20  3.80  5.60  4.00  4.00  4.00  7.00  6.40  5.60 11.00  4.50
##  [25]  4.80  5.80  3.80  4.40  6.20  4.20  7.90  3.70  4.50  6.70  6.60  3.70
##  [37]  5.20 15.50  4.10  8.30  6.55  4.60  6.10  4.30  5.80  5.15  6.30  4.20
##  [49]  4.60  4.20  4.30  7.90  4.60  5.10  5.60  6.00  8.60  7.50  4.40  4.25
##  [61]  6.00  3.90  4.20  4.00  4.00  6.60  6.00  3.80  9.00  4.60  8.80  5.00
##  [73]  3.80  4.10  5.90  4.10  6.20  8.90  4.00  3.90  8.10  6.40  8.30  8.30
##  [85]  4.70  5.50  4.30  5.50  3.70  6.20  5.60  7.80  4.60  5.80  4.10 12.90
##  [97]  4.30 13.40  4.80  6.30  4.50  4.30  3.90  3.80  5.40  3.80  6.10  3.90
## [109]  5.10  3.90 15.40  4.80  5.20  5.20  3.75 13.80  5.70  4.30  4.10  4.10
## [121]  4.40  3.70  6.70 13.90  5.10  7.80
```

Observamos como el atributo “residual.sugar” tiene 126 valores extremos, distribuidos entre 4 y 16

```
par(mfrow=c(1,2))
hist(datos$chlorides, breaks  = 50)
boxplot(datos$chlorides,main="chlorides", col="lightblue")
```



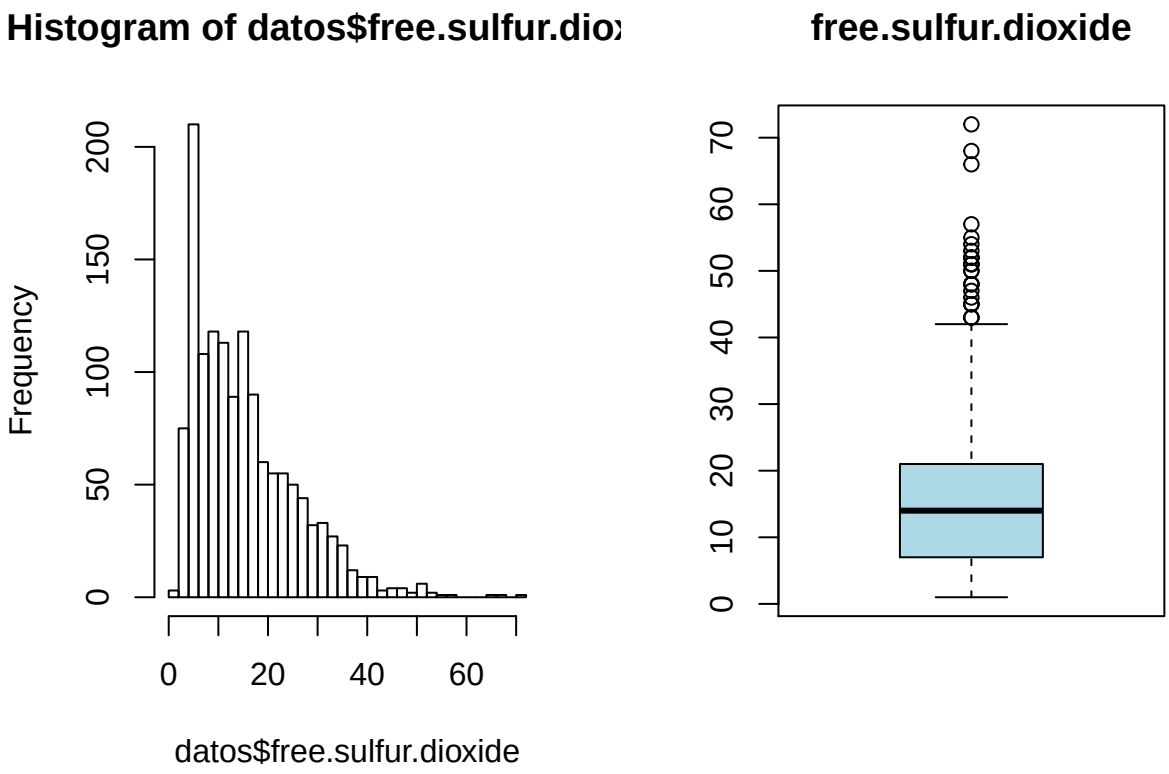



```
boxplot.stats(datos$chlorides)$out
```

```
##  [1] 0.176 0.170 0.368 0.341 0.172 0.332 0.464 0.401 0.467 0.178 0.146 0.236
## [13] 0.610 0.360 0.270 0.337 0.263 0.611 0.358 0.343 0.186 0.213 0.214 0.128
## [25] 0.159 0.124 0.174 0.127 0.413 0.152 0.152 0.125 0.200 0.171 0.226 0.250
## [37] 0.148 0.124 0.143 0.222 0.157 0.422 0.034 0.387 0.415 0.157 0.157 0.243
## [49] 0.241 0.190 0.132 0.126 0.038 0.165 0.145 0.147 0.012 0.194 0.132 0.161
## [61] 0.123 0.414 0.216 0.171 0.178 0.369 0.166 0.166 0.136 0.132 0.123 0.123
## [73] 0.403 0.137 0.414 0.166 0.168 0.415 0.153 0.267 0.123 0.214 0.169 0.205
## [85] 0.235 0.230 0.038
```

Observamos como el atributo “chlorides” tiene 87 valores extremos, distribuidos entre 0 y 0.05 por la parte inferior y entre 0.12 y 0.6 por la parte superior.

```
par(mfrow=c(1,2))
hist(datos$free.sulfur.dioxide, breaks  = 30)
boxplot(datos$free.sulfur.dioxide,main="free.sulfur.dioxide", col="lightblue")
```

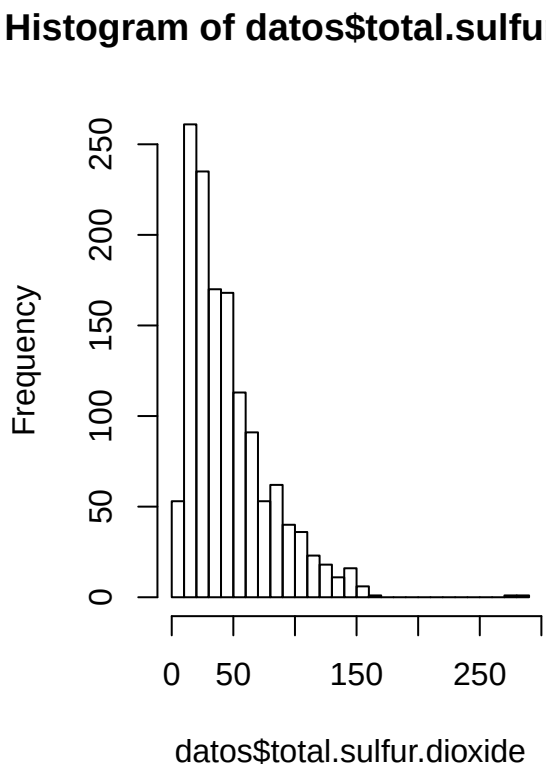


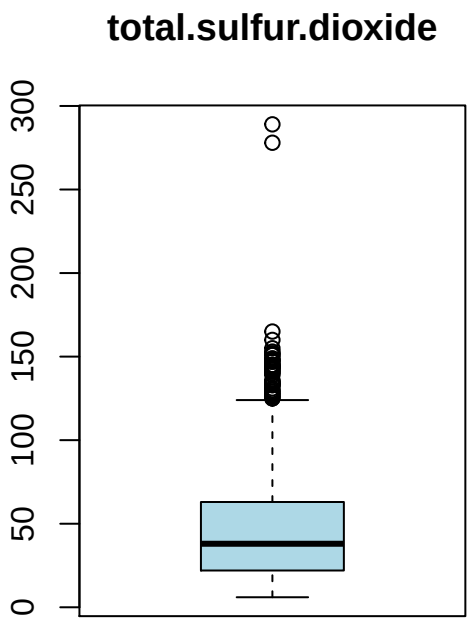

```
boxplot.stats(datos$free.sulfur.dioxide)$out
```

```
##  [1] 52 51 50 68 43 47 54 46 45 53 52 51 45 57 50 45 48 43 48 72 43 51 52 55 48
## [26] 66
```

Observamos como el atributo “free.sulfur.dioxide” tiene 26 valores extremos, distribuidos entre el 43 y el 60

```
par(mfrow=c(1,2))
hist(datos$total.sulfur.dioxide, breaks  = 30)
boxplot(datos$total.sulfur.dioxide,main="total.sulfur.dioxide", col="lightblue")
```



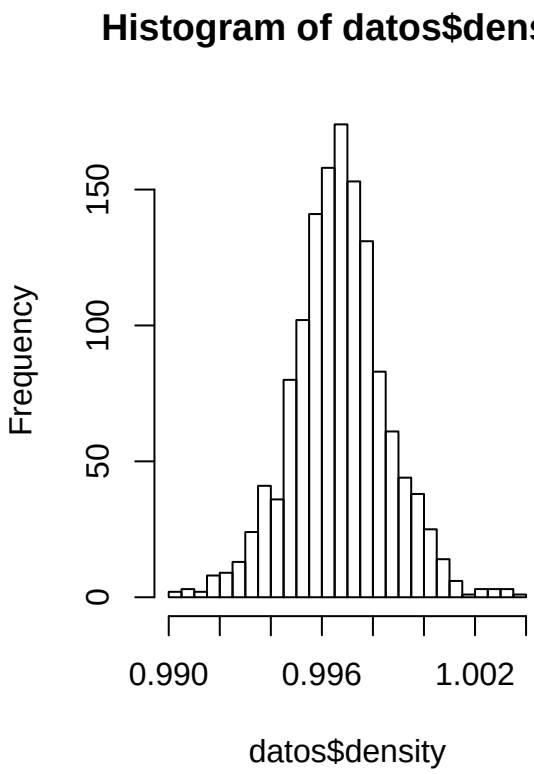


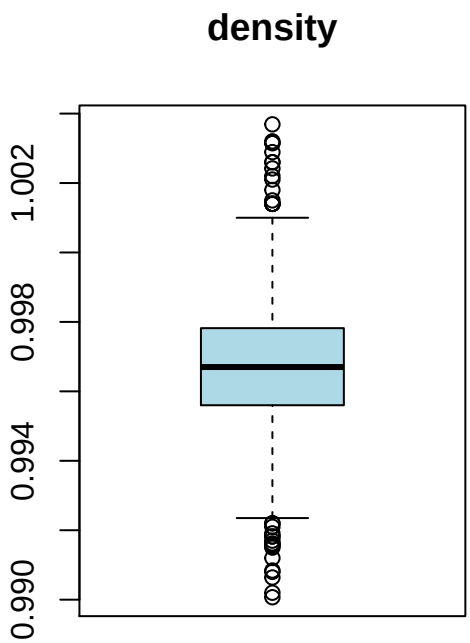

```
boxplot.stats(datos$total.sulfur.dioxide)$out
```

```
##  [1] 145 148 136 125 140 133 153 134 141 129 128 143 144 127 126 145 144 135 165
## [20] 134 129 151 133 142 149 147 145 148 155 151 152 125 127 139 143 144 130 278
## [39] 289 135 160 141 133 147 131
```

Observamos como el atributo "total.sulfur.dioxide" tiene 45 valores extremos, distribuidos entre el 120 y el 300.

```
par(mfrow=c(1,2))
hist(datos$density, breaks  = 30)
boxplot(datos$density,main="density", col="lightblue")
```



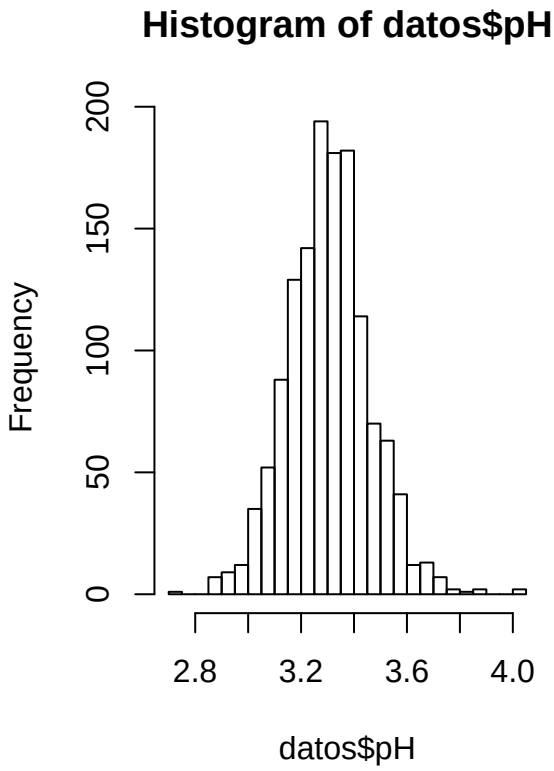


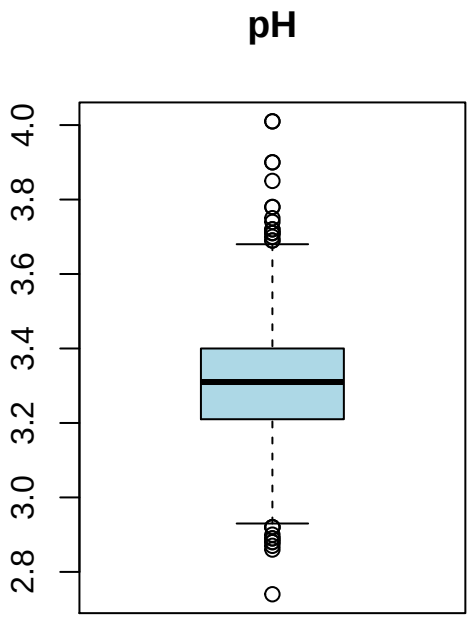

```
boxplot.stats(datos$density)$out
```

```
##  [1] 0.99160 1.00140 1.00150 1.00180 0.99120 1.00220 1.00140 1.00140 1.00140
## [10] 1.00320 1.00260 1.00140 1.00315 1.00315 1.00210 0.99170 0.99220 1.00260
## [19] 0.99210 0.99154 0.99064 1.00289 0.99162 0.99007 0.99020 0.99220 0.99150
## [28] 0.99157 0.99080 0.99084 0.99191 1.00369 1.00242 0.99182 0.99182
```

Observamos como el atributo “density” tiene 35 valores extremos, distribuidos entre 0.990 y 0.992 por la parte inferior y entre 1.001 y 1.004 por la parte superior.

```
par(mfrow=c(1,2))
hist(datos$pH, breaks  = 30)
boxplot(datos$pH,main="pH", col="lightblue")
```



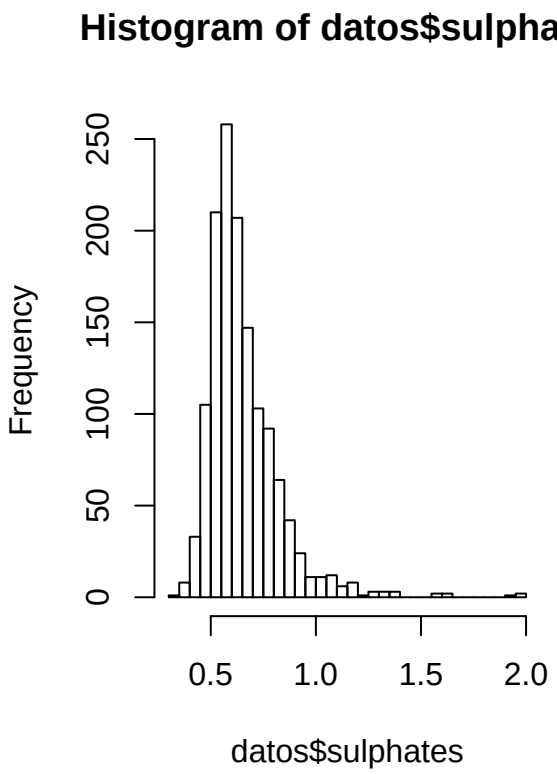


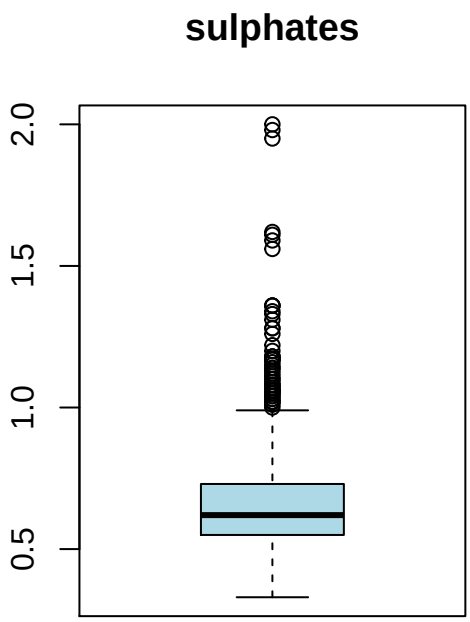

```
boxplot.stats(datos$pH)$out
```

```
##  [1] 3.90 3.75 3.85 2.74 3.69 2.88 2.86 3.74 2.92 2.92 3.72 2.87 2.89 2.92 3.90
## [16] 3.71 3.69 3.71 2.89 3.78 3.70 3.78 4.01 2.90 4.01 3.71 2.88 3.72
```

Observamos como el atributo “pH” tiene 28 valores extremos, distribuidos entre 2.7 y 2.9 por la parte inferior y entre 3.7 y 4 por la parte superior.

```
par(mfrow=c(1,2))
hist(datos$sulphates, breaks  = 30)
boxplot(datos$sulphates,main="sulphates", col="lightblue")
```



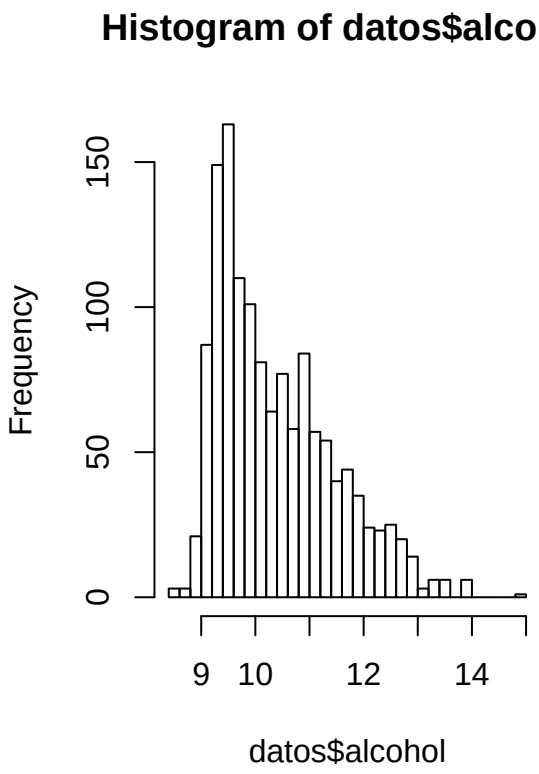


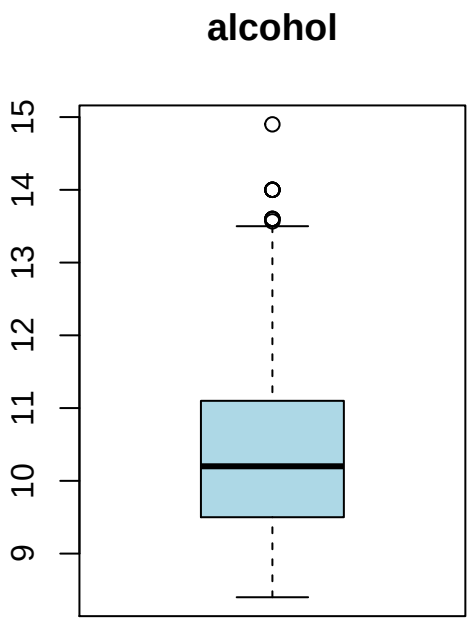

```
boxplot.stats(datos$sulphates)$out
```

```
##  [1] 1.56 1.28 1.08 1.20 1.12 1.28 1.14 1.95 1.22 1.98 1.31 2.00 1.08 1.59 1.02
## [16] 1.03 1.61 1.09 1.26 1.08 1.00 1.36 1.18 1.13 1.04 1.11 1.07 1.06 1.06 1.05
## [31] 1.06 1.04 1.05 1.02 1.14 1.02 1.36 1.36 1.05 1.17 1.62 1.06 1.18 1.07 1.34
## [46] 1.16 1.10 1.15 1.17 1.33 1.18 1.17 1.03 1.10 1.01
```

Observamos como el atributo “sulphates” tiene 55 valores extremos, distribuidos entre 1 y 2.

```
par(mfrow=c(1,2))
hist(datos$alcohol, breaks  = 30)
boxplot(datos$alcohol,main="alcohol", col="lightblue")
```



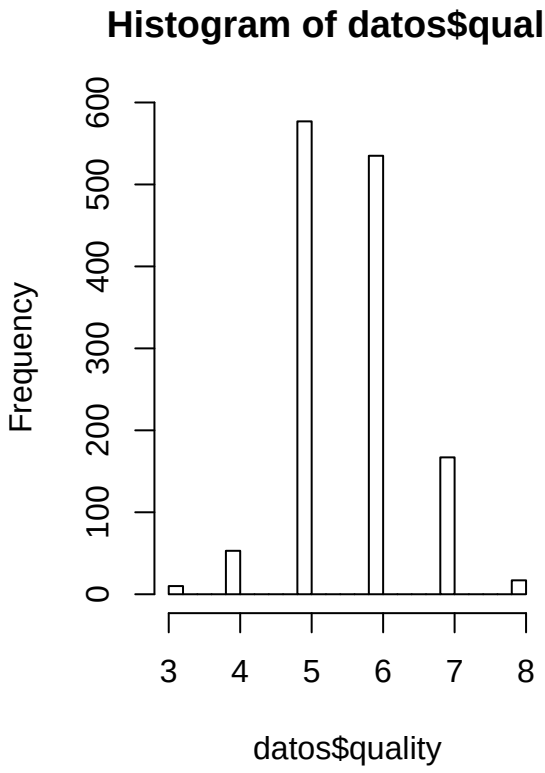


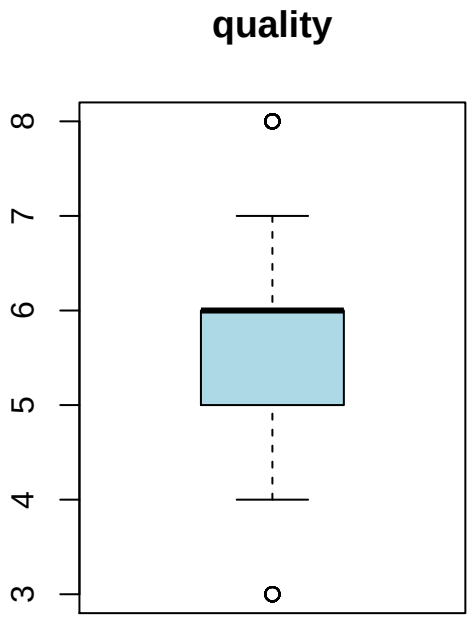

```
boxplot.stats(datos$alcohol)$out
```

```
##  [1] 14.00000 14.00000 14.00000 14.90000 14.00000 13.60000 13.60000 13.60000
##  [9] 14.00000 14.00000 13.56667 13.60000
```

Observamos como el atributo “alcohol” tiene 12 valores extremos, distribuidos entre el 13.5 y el 14.

```
par(mfrow=c(1,2))
hist(datos$quality, breaks  = 30)
boxplot(datos$quality,main="quality", col="lightblue")
```





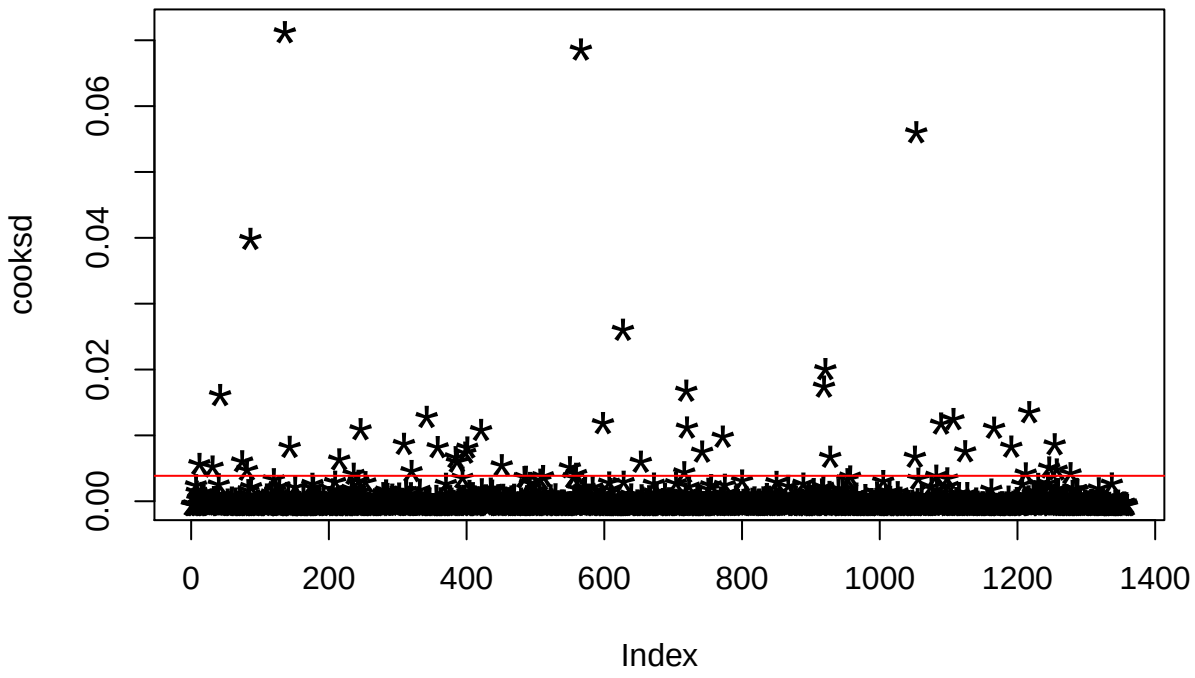

```
boxplot.stats(datos$quality)$out
```

```
##  [1] 8 8 8 8 8 3 8 8 3 8 3 8 3 3 8 8 8 8 8 3 3 8 8 3 3 3 8
```

Observamos como el atributo "quality" tiene 27 valores extremos, distribuidos entre el 3 por la parte inferior, y el 8 en la parte superior. Este valor es una valoración del vino, por lo que estos valores no se pueden considerar extremos.

En conclusión, con el análisis realizado para cada variable, el número de valores extremos es muy dispar, siendo bajo en algunas variables y relativamente alto en otras. Como es posible que algunos valores extremos de una variable coincidan en registro con los de otra, para identificar correctamente los valores extremos vamos a utilizar la distancia de Cook, estimando el grado de influencia de cada uno de los valores al realizar un análisis de regresión por mínimos cuadrados. Para realizarlo, tendremos en cuenta todos los atributos a excepción de "quality".

```
# Cálculo y visualización de resultados de aplicar la distancia de Cook a nuestros datos.
outliers = c()
for ( i in 1:11 ) {
  stats = boxplot.stats(datos[[i]])$stats
  bottom_outlier_rows = which(datos[[i]] < stats[1])
  top_outlier_rows = which(datos[[i]] > stats[5])
  outliers = c(outliers , top_outlier_rows[ !top_outlier_rows %in% outliers ] )
  outliers = c(outliers , bottom_outlier_rows[ !bottom_outlier_rows %in% outliers ] )
}
mod = lm(quality ~ ., data = datos)
cooksd = cooks.distance(mod)
plot(cooksd, pch = "*", cex = 2, main = "Observaciones relevantes en función de la distancia de Cook")
abline(h = 4*mean(cooksd, na.rm = T), col = "red")
```




```
# Obtenemos el listado de los valores extremos que afectan sensiblemente a los datos.
head(datos[cooksd > 4 * mean(cooksd, na.rm=T), ])
```

```
##    fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 14           7.8            0.610        0.29            1.6     0.114
## 34           6.9            0.605        0.12           10.7     0.073
## 46           4.6            0.520        0.15            2.1     0.054
## 80           8.3            0.625        0.20            1.5     0.080
## 87           8.6            0.490        0.28            1.9     0.110
## 93           8.6            0.490        0.29            2.0     0.110
##    free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 14                   9                   29  0.9974 3.26      1.56     9.1
## 34                  40                   83  0.9993 3.45      0.52     9.4
## 46                   8                   65  0.9934 3.90      0.56    13.1
## 80                  27                  119  0.9972 3.16      1.12     9.1
## 87                  20                  136  0.9972 2.93      1.95     9.9
## 93                  19                  133  0.9972 2.93      1.98     9.8
##    quality
## 14       5
## 34       6
## 46       4
## 80       4
## 87       6
## 93       5
```

Si visualizamos las primeras entradas, todos ellos tienen valores atípicos en una o más variables. El registro 14 tiene los "chlorides" y los "sulphates" muy altos. El registro 34 tiene el "residual.sugar" muy alto. El

registro 46 tiene el “pH” muy alto. El registro 80 tiene los “sulphates” altos. El registro 87 y 93 tienen los “chlorides”, los “sulphates” y el “total.sulfur.dioxide” altos.

Vamos a eliminar los valores extremos detectados.

```
# Identificamos los registros a eliminar
coutliers = as.numeric(rownames(datos[cooksd > 4 * mean(cooksd, na.rm=T), ]))
outliers = c(outliers , coutliers[ !coutliers %in% outliers ] )

# Eliminamos los elementos detectados como extremos.
datos_clean = datos[-outliers, ]

# Visualizamos el tamaño y los valores básicos de nuestro nuevo conjunto de datos.
dim(datos_clean)
```

```
## [1] 980  12
```

```
summary(datos_clean)
```

```
##  fixed.acidity    volatile.acidity  citric.acid    residual.sugar
##  Min.   : 5.100   Min.   :0.1200   Min.   :0.000   Min.   :1.200
##  1st Qu.: 7.100   1st Qu.:0.3900   1st Qu.:0.080   1st Qu.:1.900
##  Median : 7.800   Median :0.5200   Median :0.240   Median :2.100
##  Mean   : 8.155   Mean   :0.5216   Mean   :0.249   Mean   :2.196
##  3rd Qu.: 9.000   3rd Qu.:0.6300   3rd Qu.:0.400   3rd Qu.:2.500
##  Max.   :12.300   Max.   :1.0100   Max.   :0.730   Max.   :3.650
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.03900   Min.   : 1.00       Min.   :  6.00       Min.   :0.9923
##  1st Qu.:0.06900   1st Qu.: 8.00       1st Qu.: 22.00       1st Qu.:0.9955
##  Median :0.07800   Median :13.00       Median : 36.00       Median :0.9965
##  Mean   :0.07841   Mean   :15.01       Mean   : 42.38       Mean   :0.9965
##  3rd Qu.:0.08700   3rd Qu.:20.00       3rd Qu.: 56.00       3rd Qu.:0.9976
##  Max.   :0.12200   Max.   :42.00       Max.   :124.00       Max.   :1.0010
##        pH          sulphates         alcohol         quality
##  Min.   :2.940   Min.   :0.3700   Min.   : 8.70   Min.   :3.00
##  1st Qu.:3.230   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.00
##  Median :3.320   Median :0.6100   Median :10.10   Median :6.00
##  Mean   :3.322   Mean   :0.6313   Mean   :10.39   Mean   :5.64
##  3rd Qu.:3.400   3rd Qu.:0.7000   3rd Qu.:11.00   3rd Qu.:6.00
##  Max.   :3.680   Max.   :0.9800   Max.   :13.40   Max.   :8.00
```

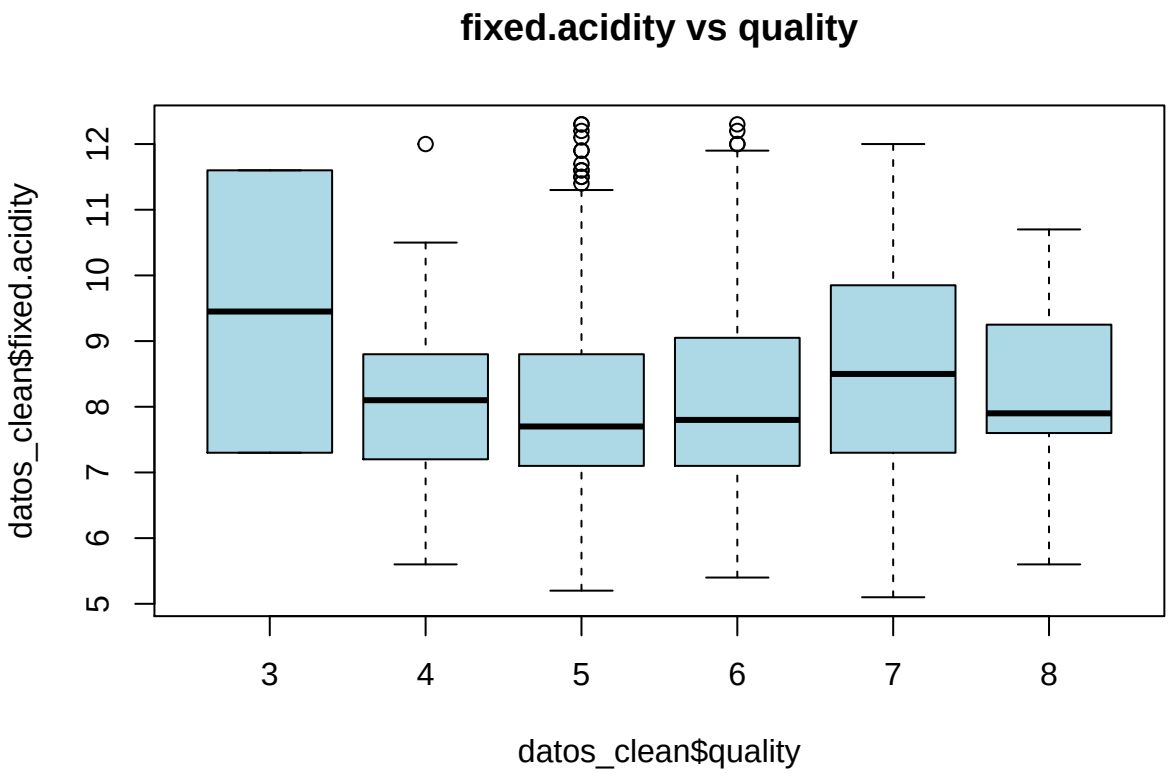
Análisis de los datos

Antes de comenzar con el análisis guardaremos una copia de los datos después del proceso de limpieza

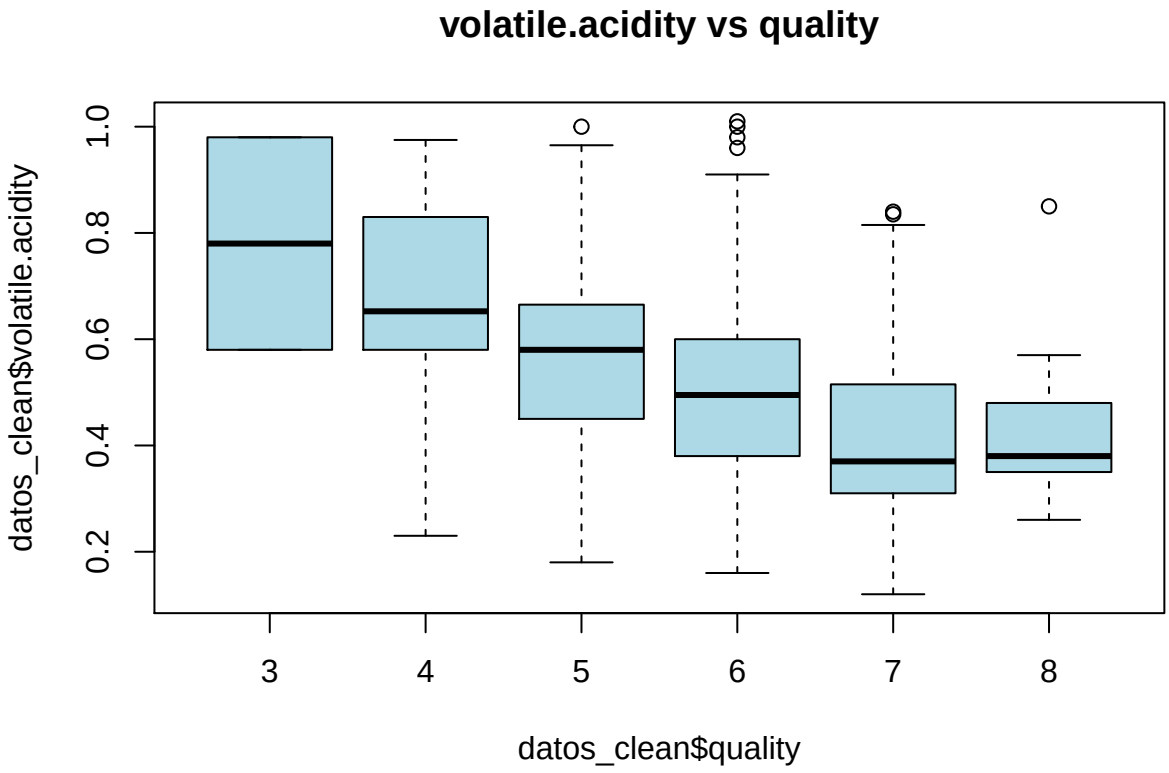
```
# Exportación de los datos limpios en .csv
write.csv(datos_clean, "winequality-red-clean.csv")
```

Analizaremos las variables frente a la calidad para decidir cuales utilizar en el resto del análisis

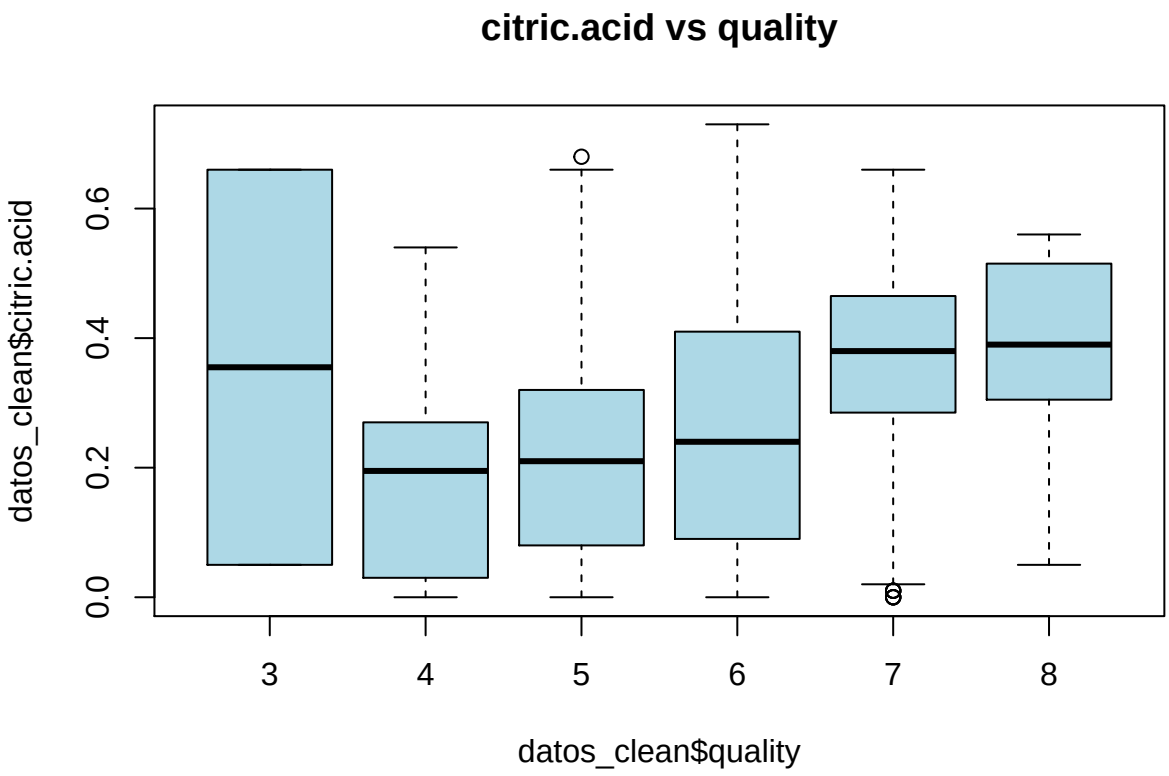
```
boxplot(formula = datos_clean$fixed.acidity ~ datos_clean$quality, main="fixed.acidity vs quality", col=
```




```
boxplot(formula = datos_clean$volatile.acidity ~ datos_clean$quality, main="volatile.acidity vs quality"
```

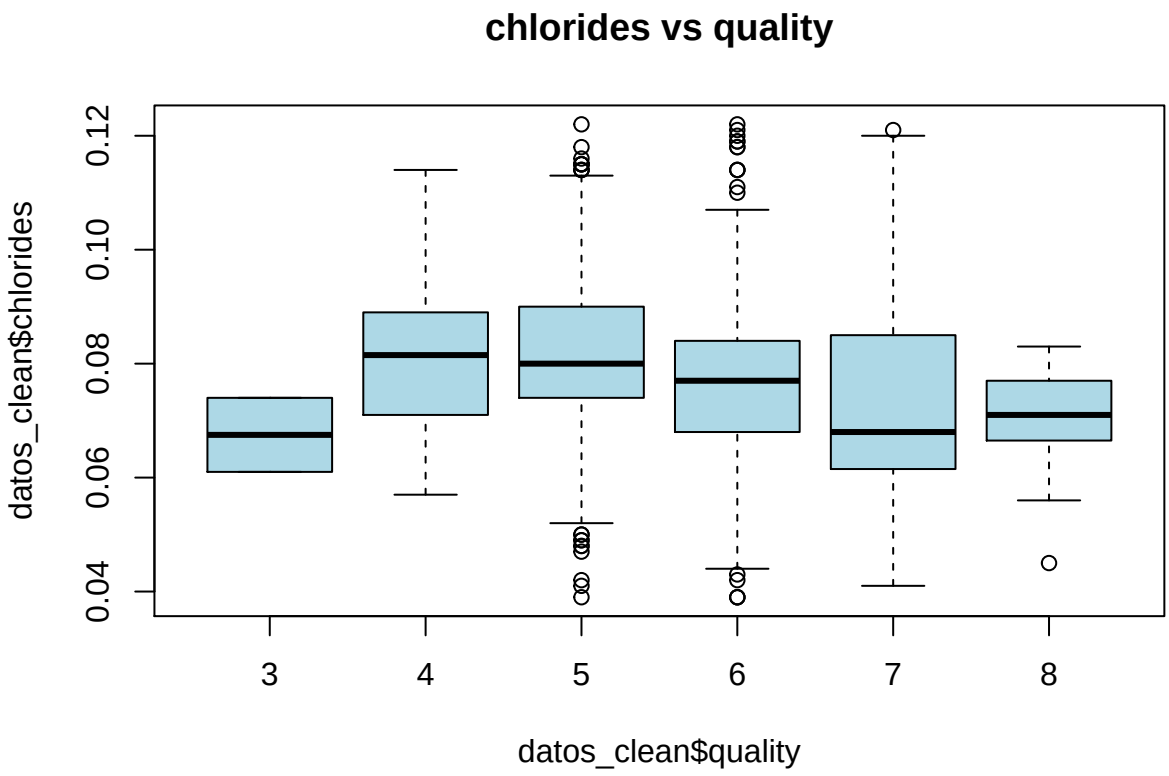



```
boxplot(formula = datos_clean$citric.acid ~ datos_clean$quality, main="citric.acid vs quality", col="lig
```

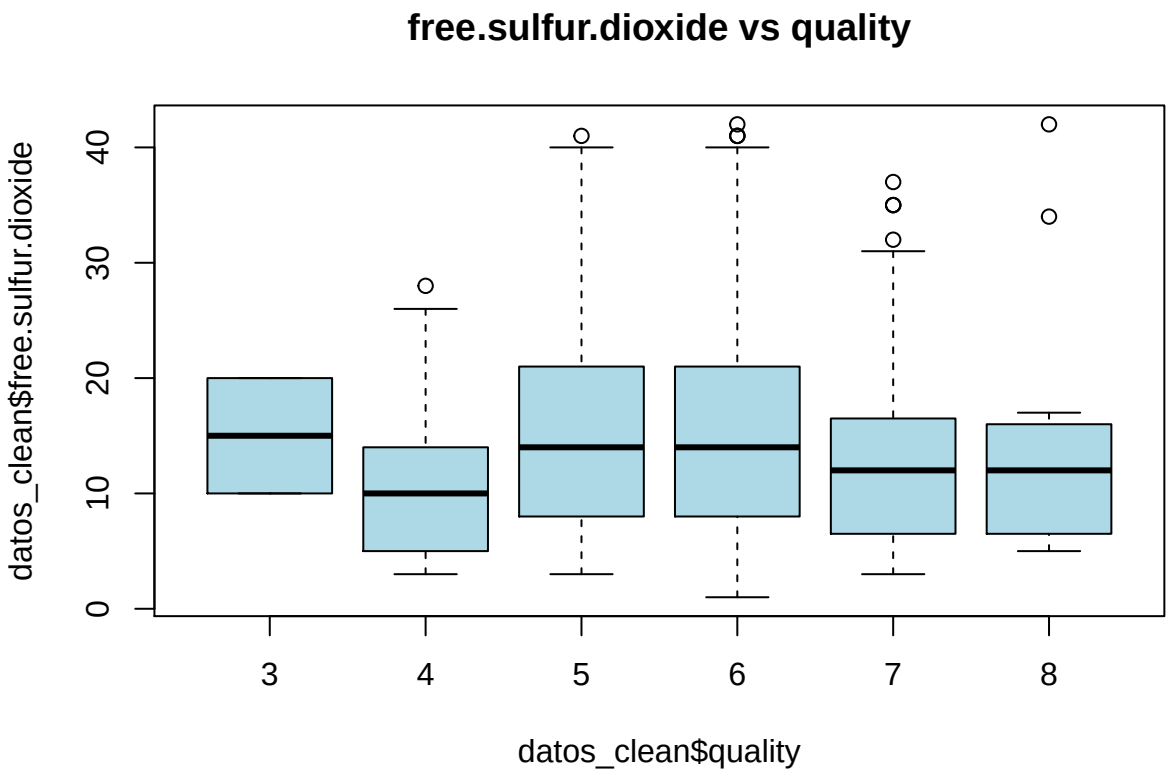



```
boxplot(formula = datos_clean$residual.sugar ~ datos_clean$quality, main="residual.sugar vs quality", c
```

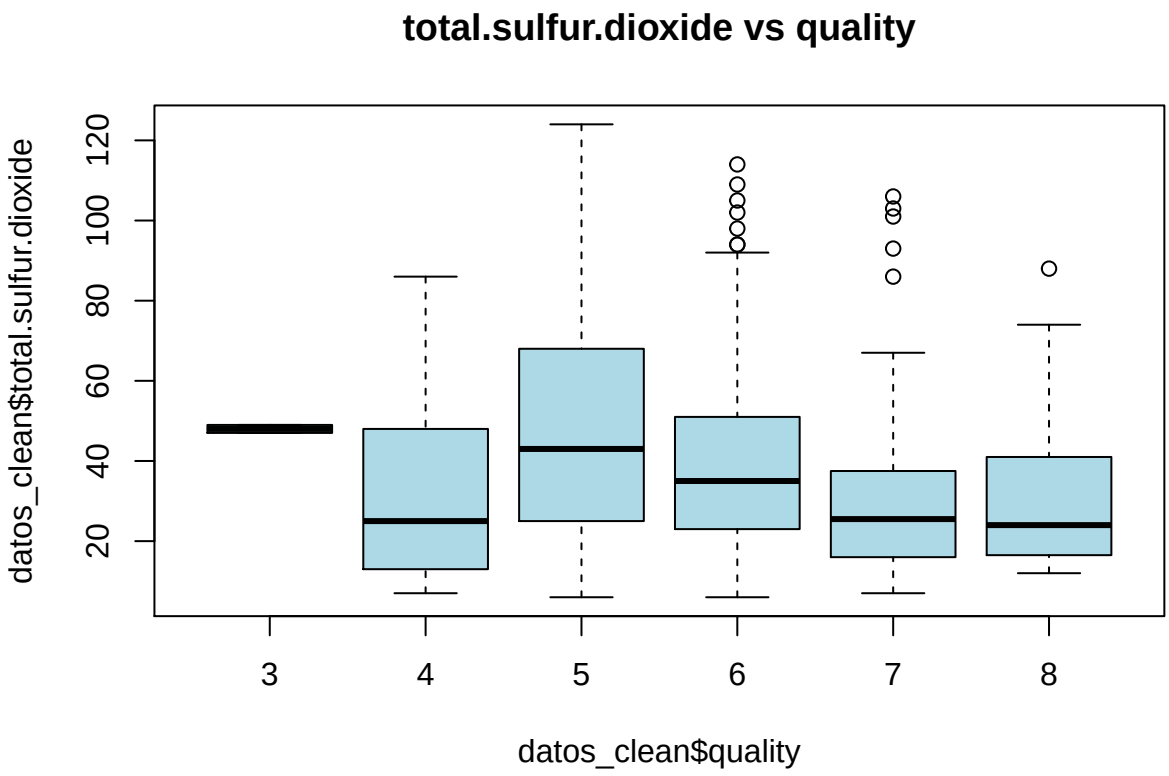

```
boxplot(formula = datos_clean$chlorides ~ datos_clean$quality, main="chlorides vs quality", col="lightbl
```



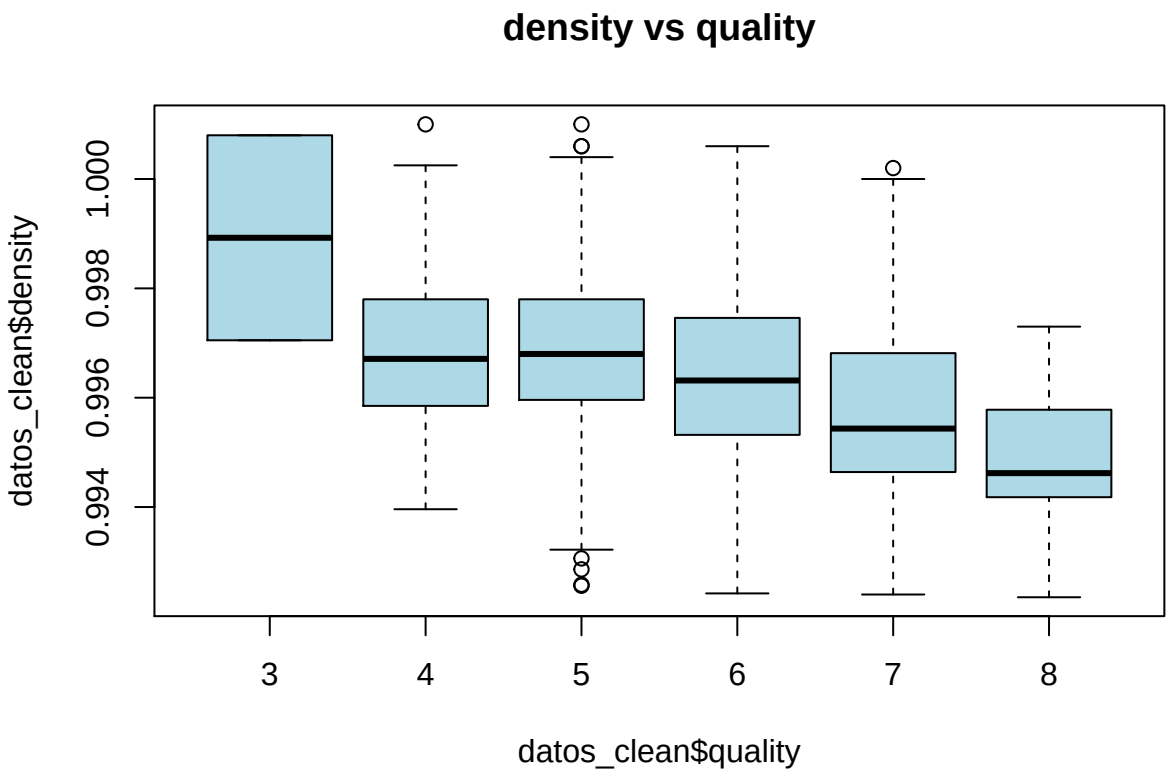

```
boxplot(formula = datos_clean$free.sulfur.dioxide ~ datos_clean$quality, main="free.sulfur.dioxide vs qu
```



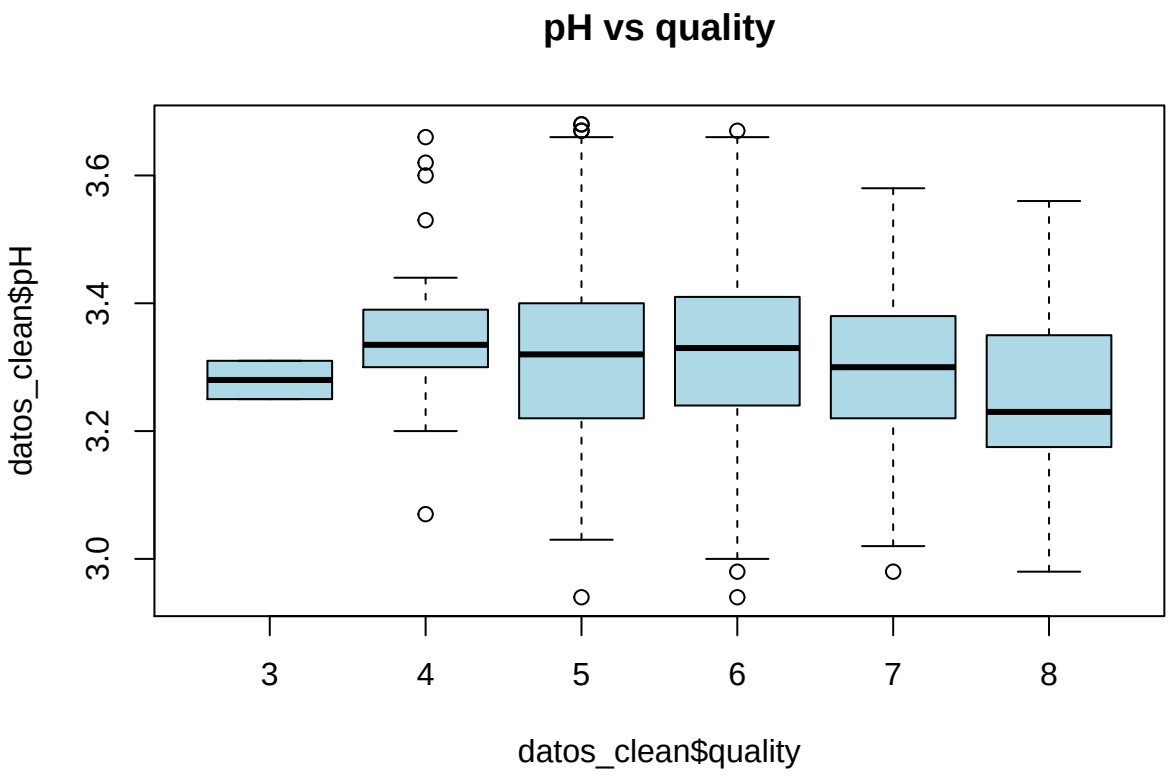

```
boxplot(formula = datos_clean$total.sulfur.dioxide ~ datos_clean$quality, main="total.sulfur.dioxide vs
```



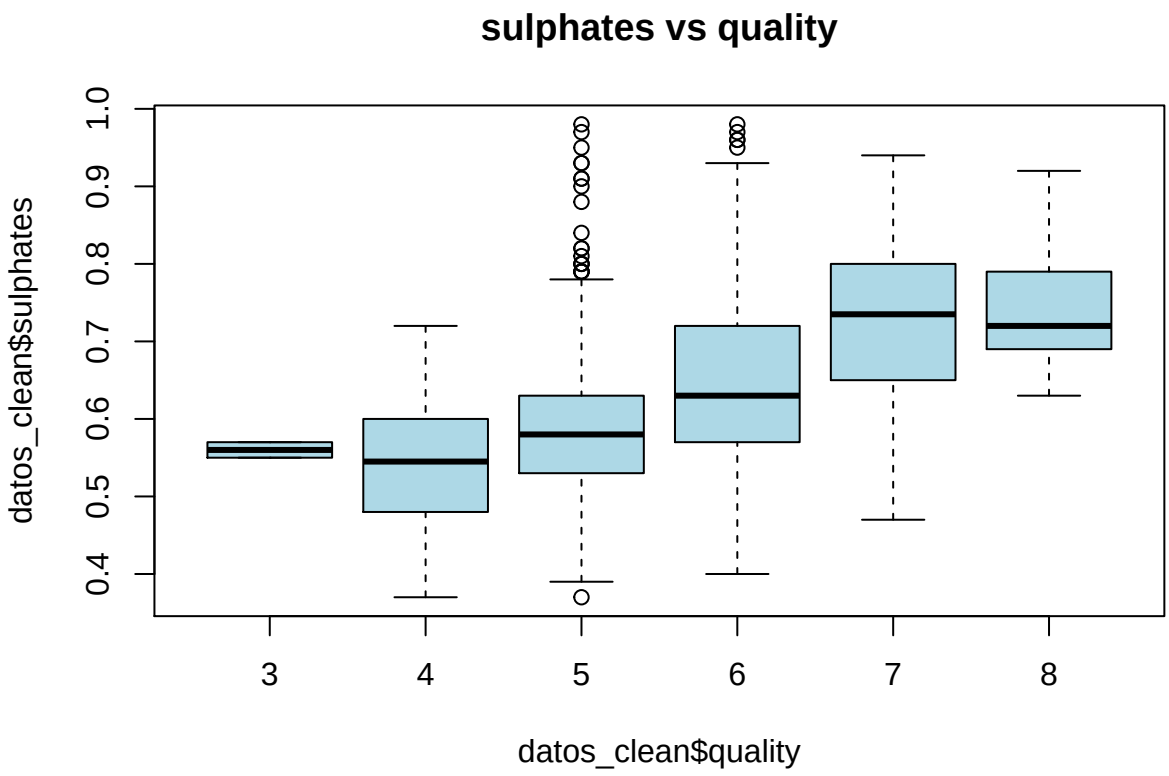

```
boxplot(formula = datos_clean$density ~ datos_clean$quality, main="density vs quality", col="lightblue"
```



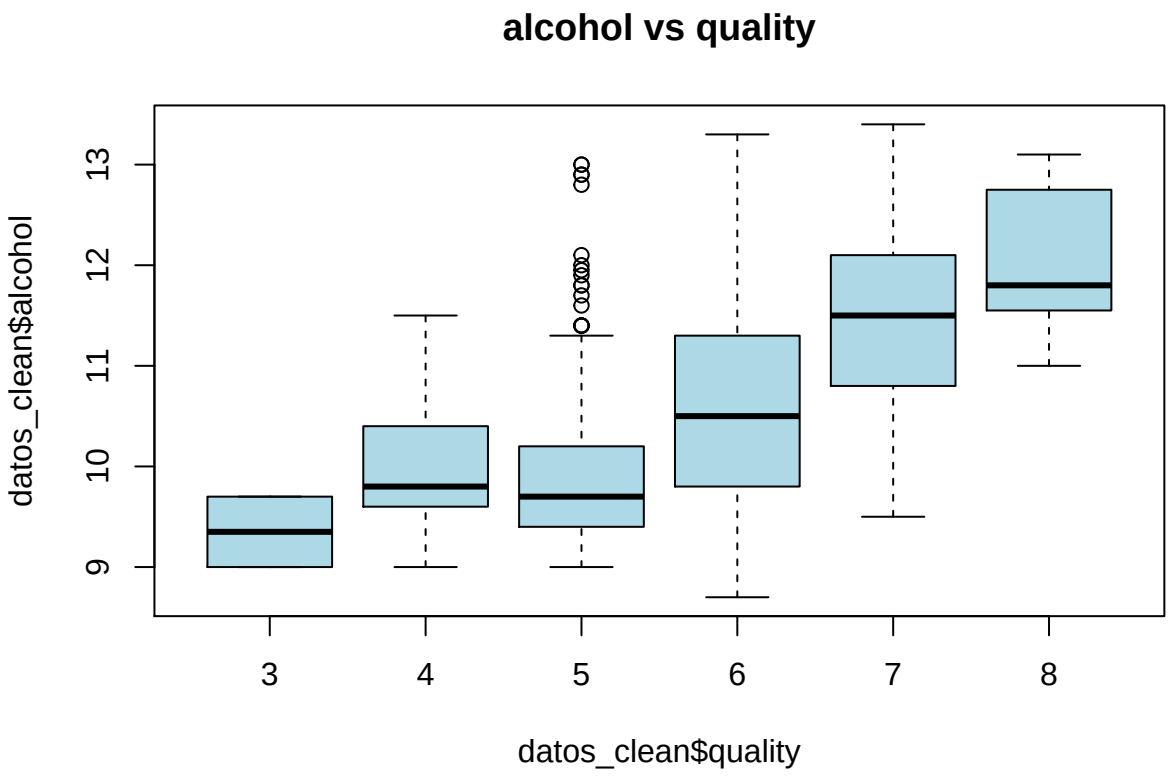

```
boxplot(formula = datos_clean$pH ~ datos_clean$quality, main="pH vs quality", col="lightblue")
```




```
boxplot(formula = datos_clean$sulphates ~ datos_clean$quality, main="sulphates vs quality", col="lightbl
```




```
boxplot(formula = datos_clean$alcohol ~ datos_clean$quality, main="alcohol vs quality", col="lightblue"
```



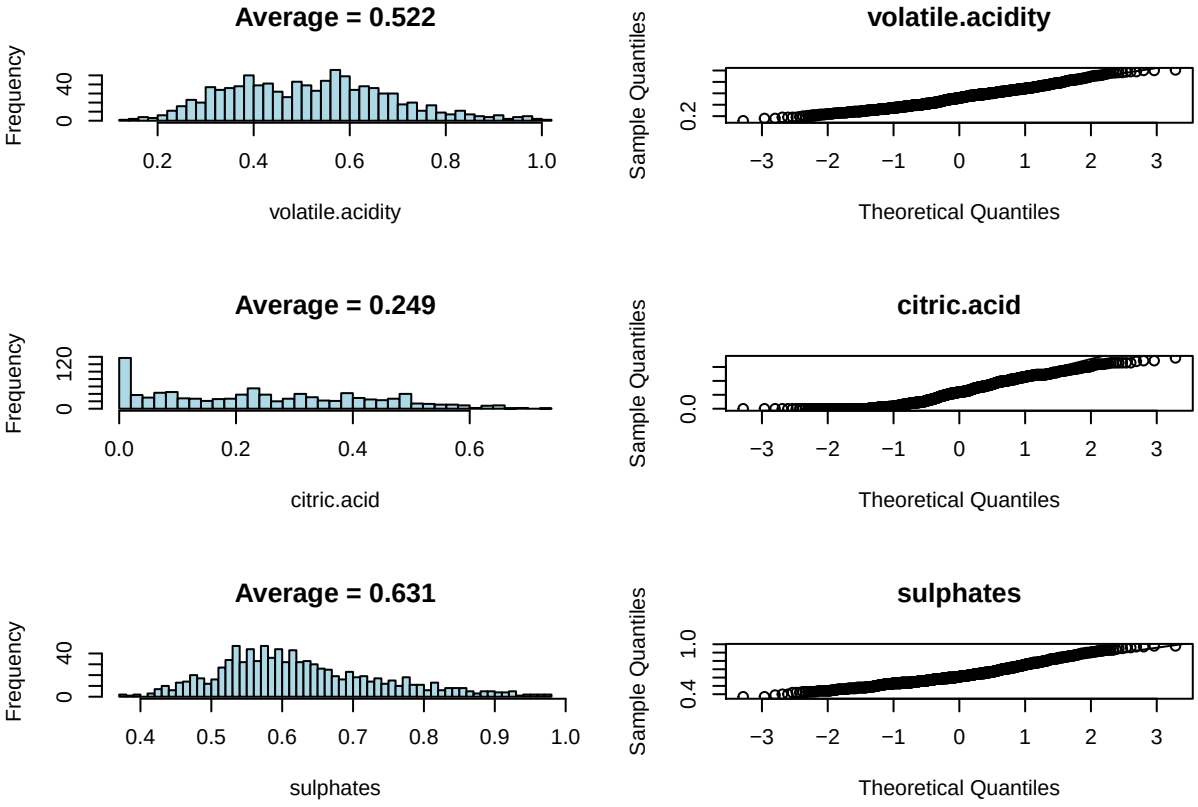
Seleccion grupo de datos De la observación del grupo de datos nos interesa seleccionar los que pudieran tener una mayor relación con el resultado de calidad. Por ello vamos a seleccionar las que se intuye una cierta relación lineal para poder aplicar modelos de predicción. Las variables "citrix acid" , "alcohol" y "sulphates", conforme aumentan, aumenta el valor de la calidad. Por el contrario para que aumente el valor de la calidad es necesario que disminuyan "volatile acidity", "density". Crearemos un subconjunto de datos con estas cinco variables

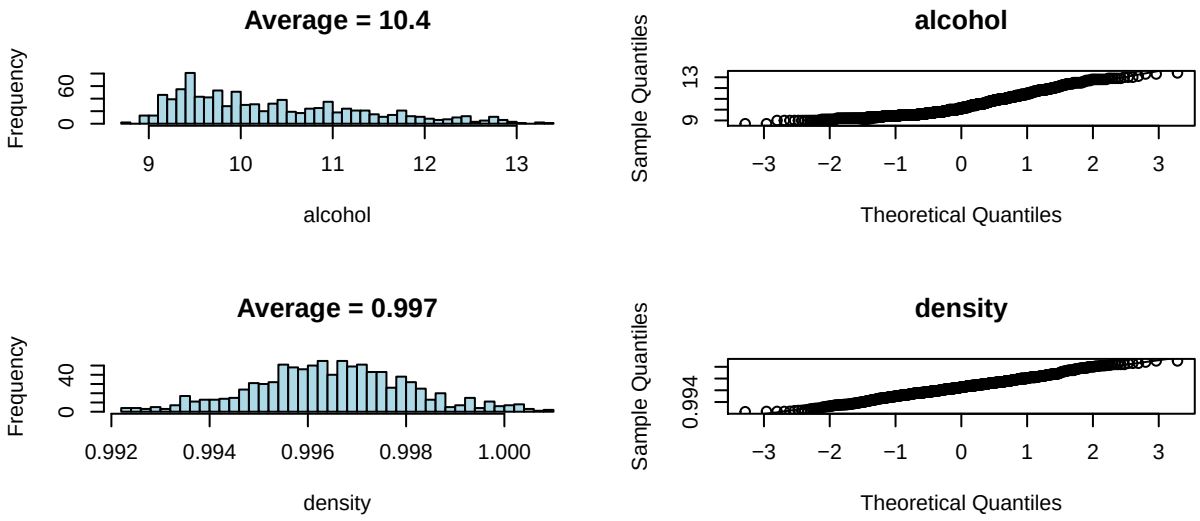
```
subdatos <- select(datos_clean, volatile.acidity, citric.acid, sulphates, alcohol, density, quality)
```

Comprobación de la normalidad y homogeneidad de la varianza.

Existen diferentes maneras de comprobar la normalidad de los datos, la del test de Shapiro es la más habitual, pero también es posible realizar dicha comprobación de forma visual mediante los histogramas y las gráficas quantilie-quantile. Este método nos permite identificar más fácilmente las distribuciones que se alejan de la normalidad.

```
# Gráficas QQ de comprobación de normalidad e histogramas
par(mfrow=c(3,2))
for (i in 1:(ncol(subdatos)-1)) {
  hist(subdatos[[i]], xlab = names(subdatos)[i], col = 'lightblue', main = paste("Average =", signif(mea
  qqnorm(subdatos[,i], main = colnames(subdatos[i]))
  qqline(subdatos[,i])

}
```


Average = 0.522
Frequency
0
40
0.2
0.4
0.6
0.8
1.0
volatile.acidity
volatile.acidity
Sample Quantiles
0.2
−3
−2
−1
0
1
2
3
Theoretical Quantiles
Average = 0.249
Frequency
0
120
0.0
0.2
0.4
0.6
citric.acid
citric.acid
Sample Quantiles
0.0
−3
−2
−1
0
1
2
3
Theoretical Quantiles
Average = 0.631
Frequency
0
40
0.4
0.5
0.6
0.7
0.8
0.9
1.0
sulphates
sulphates
Sample Quantiles
0.4
1.0
−3
−2
−1
0
1
2
3
Theoretical Quantiles



Los resultados gráficos nos indican que las variables “volatile.acidity”, “sulphates” y “density” podrían tener una distribución normal. Las variables “citric.acid” y “alcohol” no presentan visualmente una distribución normal.la mayoría de los atributos se acercan mucho a una distribución normal.

Vamos a verificar la normalidad de los datos con un test de normalidad para cada valor. El test asume como hipótesis nula la distribución normal de los datos.

```
# Test de normalidad para las diferentes variables de nuestro subconjunto de datos
shapiro.test(subdatos$volatile.acidity)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  subdatos$volatile.acidity
## W = 0.98705, p-value = 1.291e-07
```

```
shapiro.test(subdatos$citric.acid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  subdatos$citric.acid
## W = 0.95083, p-value < 2.2e-16
```

```
shapiro.test(subdatos$sulphates)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  subdatos$sulphates
## W = 0.96736, p-value = 5.089e-14
```

```
shapiro.test(subdatos$alcohol)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  subdatos$alcohol
## W = 0.9291, p-value < 2.2e-16
```

```
shapiro.test(subdatos$density)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  subdatos$density
## W = 0.99616, p-value = 0.01611
```

Observamos que p-value está por debajo del valor de significancia, para todas las variables a excepción de "density". La única varible que se puede considerar que tiene una distribución normal de los datos es "density".

Para la verificación de la homocedasticidad, vamos a utilizar el test de Fligner-Killen, que se puede aplicar sobre datos que no cumplen con la condición de normalidad. El test asume como hipótesis nula la igualdad de varianzas en los diferentes grupos de datos.

```
# Visualización de las distribuciones de datos de densidad en función de la calidad del vino.
#library(car)
fligner.test(volatile.acidity ~ quality, data = subdatos)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  volatile.acidity by quality
## Fligner-Killeen:med chi-squared = 10.526, df = 5, p-value = 0.06162
```

```
fligner.test(citric.acid ~ quality, data = subdatos)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  citric.acid by quality
## Fligner-Killeen:med chi-squared = 15.158, df = 5, p-value = 0.009707
```

```
fligner.test(sulphates ~ quality, data = subdatos)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  sulphates by quality
## Fligner-Killeen:med chi-squared = 16.821, df = 5, p-value = 0.004853
```

```
fligner.test(alcohol ~ quality, data = subdatos)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  alcohol by quality
## Fligner-Killeen:med chi-squared = 72.362, df = 5, p-value = 3.301e-14
```

```
fligner.test(density ~ quality, data = subdatos)
```

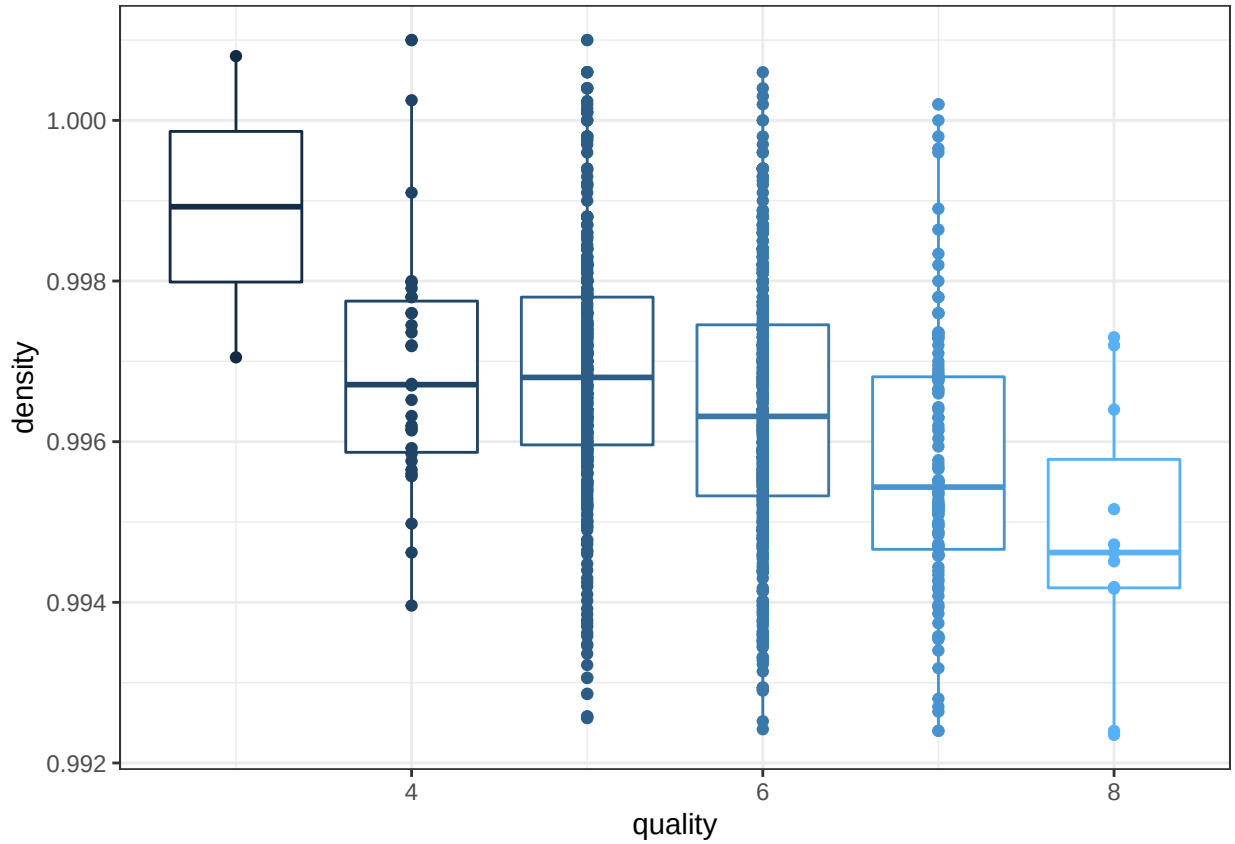
```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  density by quality
## Fligner-Killeen:med chi-squared = 9.5547, df = 5, p-value = 0.08888
```

En este caso, observamos que las variables "volitile.acidity", y "density" tiene un p-value por encima del nivel de significancia, lo que nos indica que si presentan homocedasticidad, en el resto de variables el valor está por debajo del nivel de significancia, por lo que dichas variables presentan varianzas estadísticamente diferentes para los diferentes grupos de "quality".

En el caso de que se cumplan ambas premisas, podemos realizar una comparación entre los dos grupos de datos mediante una prueba t de Student. La test asume como hipótesis nula que las medias de los grupos de datos son las mismas.

Como tenemos que comparar datos de dos conjuntos, vamos a crear un subconjunto de datos con dos categorías de vinos, y los compararemos entre ellos.

```
# Visualización de las distribuciones de datos de densidad en función de la calidad del vino.
ggplot(data = subdatos, aes(x = quality, y = density, colour = quality, group=quality)) +
  geom_boxplot() +
  geom_point() +
  theme_bw() +
  theme(legend.position = "none")
```



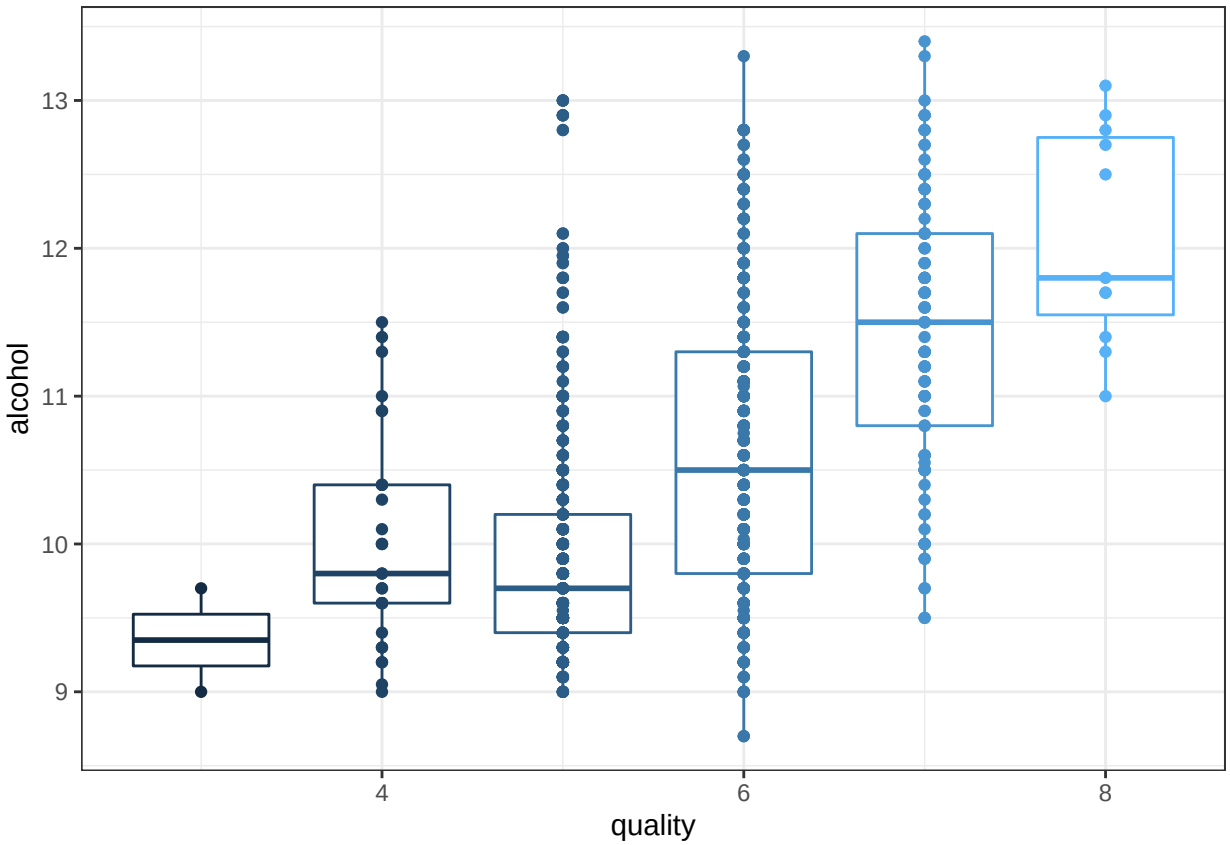
```r
# Selección de los conjuntos a comparar según la calidad del vino.
calidad <- filter(.data = subdatos, quality %in% c("4", "7"))

# Test que nos permite comprobar si se observan diferencias estadísticamente significativas de la densi
t.test(density ~ quality, data = calidad)
```

```
##
##  Welch Two Sample t-test
##
## data:  density by quality
## t = 3.6269, df = 49.503, p-value = 0.0006773
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.0005191277 0.0018084437
## sample estimates:
## mean in group 4 mean in group 7
##       0.9968970       0.9957332
```

En este caso tenemos un valor de p-value menor al nivel de significancia, lo que significa que se observan diferencias estadísticamente significativas entre los dos grupos de datos de la calidad del vino escogidos para la variable densidad.

Para el resto de variables, podemos aplicar las pruebas de Wilcox, ya que no requieren de la premisa de normalidad y homocedasticidad. Vamos a ver la como lo aplicamos para la variable "alcohol".

```r
# Visualización de las distribuciones de datos de densidad en función de la calidad del vino.

ggplot(data = subdatos, aes(x = quality, y = alcohol, colour = quality, group=quality)) +
  geom_boxplot() +
  geom_point() +
  theme_bw() +
  theme(legend.position = "none")
```




```
# Selección de los conjuntos a comparar según la calidad del vino.
calidad <- filter(.data = subdatos, quality %in% c("4", "7"))

# Test que nos permite comprobar si se observan diferencias estadísticamente significativas de la densi
wilcox.test(alcohol ~ quality, data = calidad)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  alcohol by quality
## W = 381.5, p-value = 8.559e-11
## alternative hypothesis: true location shift is not equal to 0
```

En este caso, podemos observar que p-value está por debajo del nivel de significancia, por lo que sí se observan diferencias estadísticamente significativas en la calidad del vino en terminos del alcohol presente.

Nuestro objetivo es la predicción de la calidad del vino con los diferentes parámetros de sus componentes, por lo que que vamos a proseguir con el análisis de correlación de los datos.

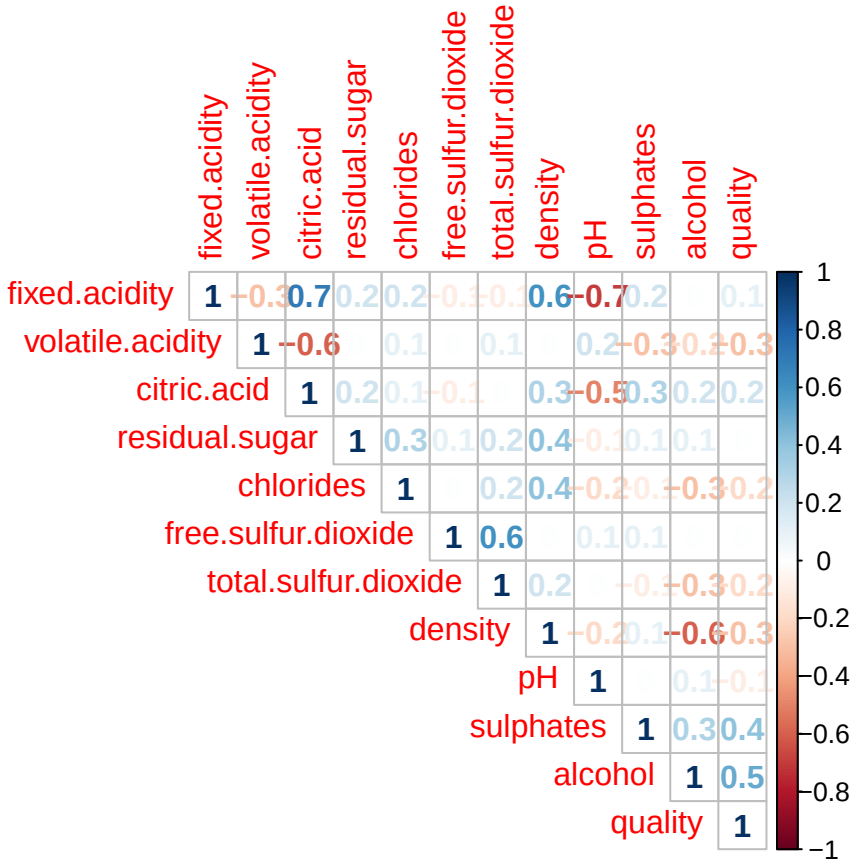
Aplicación de pruebas estadísticas para comparar los grupos de datos

Aplicaremos diversas pruebas estadísticas para analizar la relación de los datos y poder crear el modelo de predicción de la calidad. Comenzaremos por analizar los valores de correlación de las variables con la variable "quality" ### Correlacion

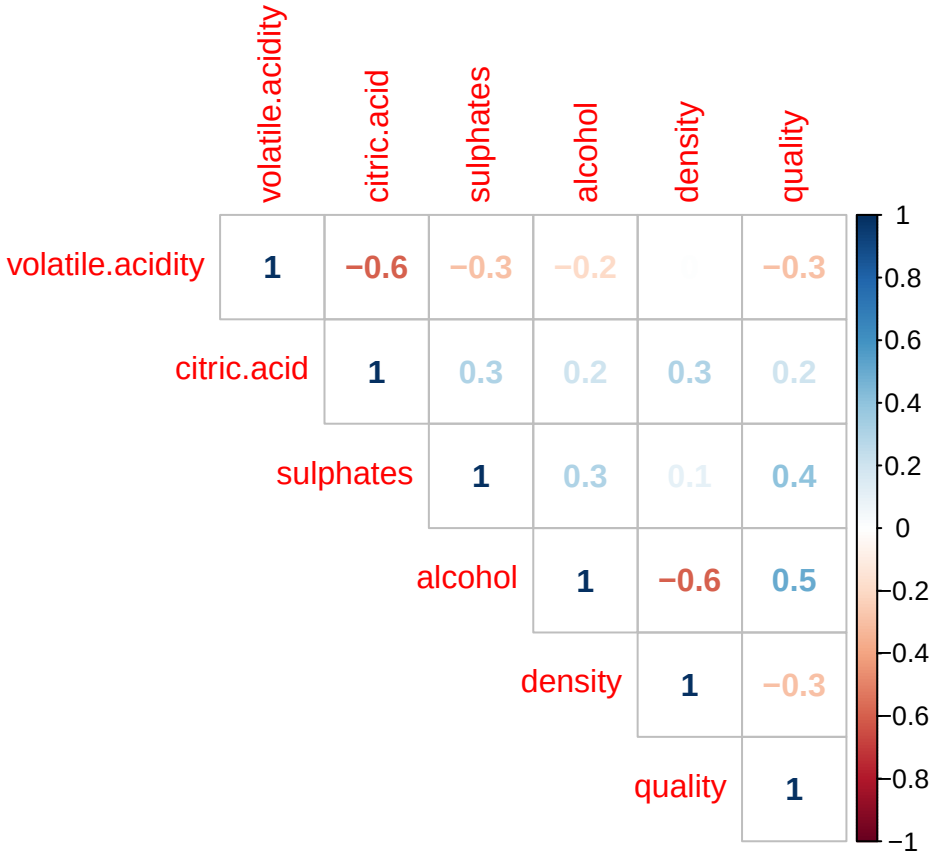
Vamos a analizar la correlación entre las variables.

```
# Visualizaremos la matriz de correlación de variables de todo el conjunto de datos.
correlacion_dc<-round(cor(datos_clean), 1)
```

```
corrplot(correlacion_dc, method="number", type="upper")
```




```
# Visualizaremos la matriz de correlación de las variables seleccionadas en nuestro subconjunto.
correlacion_sc<-round(cor(subdatos), 1)
corrplot(correlacion_sc, method="number", type="upper")
```



Guardaremos los datos de correlación en una matriz ordenada para decir que variables utilizar en siguientes estudios. Dado que hemos observado que no siempre se cumple el criterio de homocedasticidad, vamos a utilizar la correlación de Spearman, ya que este método no conlleva ninguna suposición sobre la distribución de datos.

```
# Creamos la matriz para almacenar los datos
matrixcor <- matrix(nc = 2, nr = 0)
colnames(matrixcor) <- c("Variable","correlacion")
# Recorremos el dataset ejecutando el test
for (i in 1:(ncol(subdatos)-1)) {

  coef <- cor(x=subdatos$quality, y = subdatos[,i], method="spearman")
  # Añadimos los datos a la matriz
  pair = matrix(ncol = 2, nrow = 1)
  pair[1][1] = colnames(subdatos[i])
  pair[2][1] = coef
  matrixcor <- rbind(matrixcor, pair)
}
# Ordenamos por el valor de correlacion
matrixcor[order(matrixcor[,"correlacion"]), ]
```

```
##      Variable           correlacion
## [1,] "density"          "-0.24034979639258"
## [2,] "volatile.acidity" "-0.349885983236109"
## [3,] "citric.acid"      "0.204129810206267"
## [4,] "sulphates"        "0.424537224897477"
## [5,] "alcohol"          "0.508513759148933"
```

Observamos como algunos valores tiene una correlación positiva importante con la calidad del vino. Igualmente podemos observar como existen valores con una relevante correlación negativa, que también deben ser considerados.

Regresión lineal

Con este grupo de datos y las relaciones observadas tanto en las gráficas de caja como los datos de correlación estimaremos por mínimos cuadrados ordinarios un modelo lineal que explique la variable "quality". Vamos a tener en cuenta todas las variables que tengan un valor de correlación superior a 0.2, tanto positivas como negativas.

```
# Creamos el modelo
modelo <- (lm(formula = quality ~
                alcohol +
                sulphates +
                citric.acid +
                volatile.acidity +
                density,
              data = subdatos))
summary(modelo)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + sulphates + citric.acid + volatile.acidity +
##     density, data = subdatos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.00672 -0.37652 -0.07092  0.45075  2.03816
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       26.39348   17.51451   1.507    0.132
## alcohol            0.29156    0.02722  10.712  < 2e-16 ***
## sulphates          1.80703    0.19097   9.462  < 2e-16 ***
## citric.acid       -0.09384    0.15898  -0.590    0.555
## volatile.acidity  -0.89314    0.16090  -5.551 3.66e-08 ***
## density          -24.51964   17.46415  -1.404    0.161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6117 on 974 degrees of freedom
## Multiple R-squared:   0.38,  Adjusted R-squared:  0.3769
## F-statistic: 119.4 on 5 and 974 DF,  p-value: < 2.2e-16
```

Podemos observar que las variables "citric.acid" y "density" son variables poco significativas para el modelo. Con un valor de R cuadraddo ajustado del 37.69%. Si lo comparamos con la tabla de correlación, solo se han considerado las variables con un valor de correlación superior a 0.3

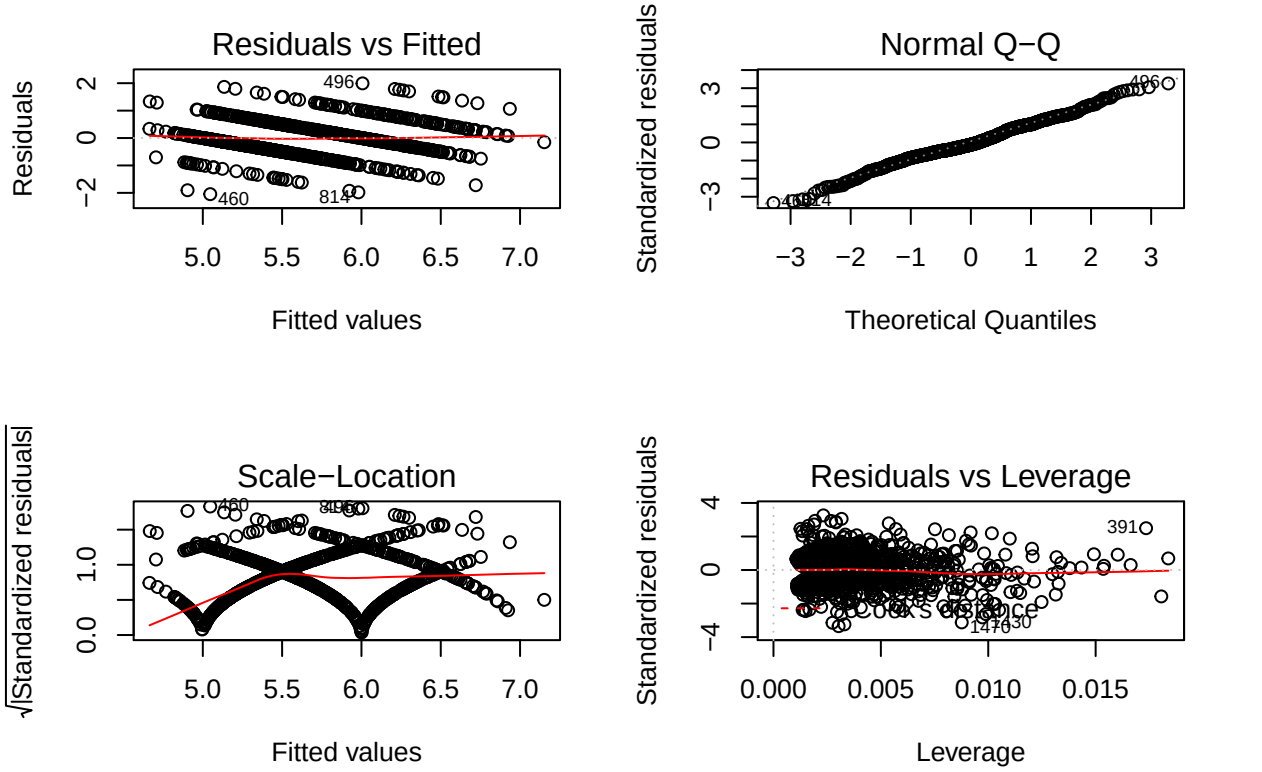
```
# Creamos un modelo reducido
modelo_reducido <- (lm(formula = quality ~ alcohol + sulphates + volatile.acidity, data = subdatos))
summary(modelo_reducido)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + sulphates + volatile.acidity,
```

```
##     data = subdatos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.04685 -0.37443 -0.07396  0.45867  1.99324
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.70257    0.24707   6.891 9.93e-12 ***
## alcohol            0.31658    0.02068  15.309  < 2e-16 ***
## sulphates          1.70920    0.18341   9.319  < 2e-16 ***
## volatile.acidity -0.82619    0.12447  -6.638 5.28e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6123 on 976 degrees of freedom
## Multiple R-squared:  0.3775, Adjusted R-squared:  0.3755
## F-statistic: 197.3 on 3 and 976 DF,  p-value: < 2.2e-16
```

Como podemos observar, todas las variables utilizadas se consideran significativas, y la exclusión de las tres variables no significativas respecto al modelo anterior no ha supuesto una merma relevante en la calidad del modelo, con estas tres variables podemos explicar el 37.56% de la clasificación de quality. Finalmente, analizamos estadísticamente el modelo.

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



La gráfica de residuos frente a Fitted muestra si los residuos tienen patrones no lineales. Los residuos

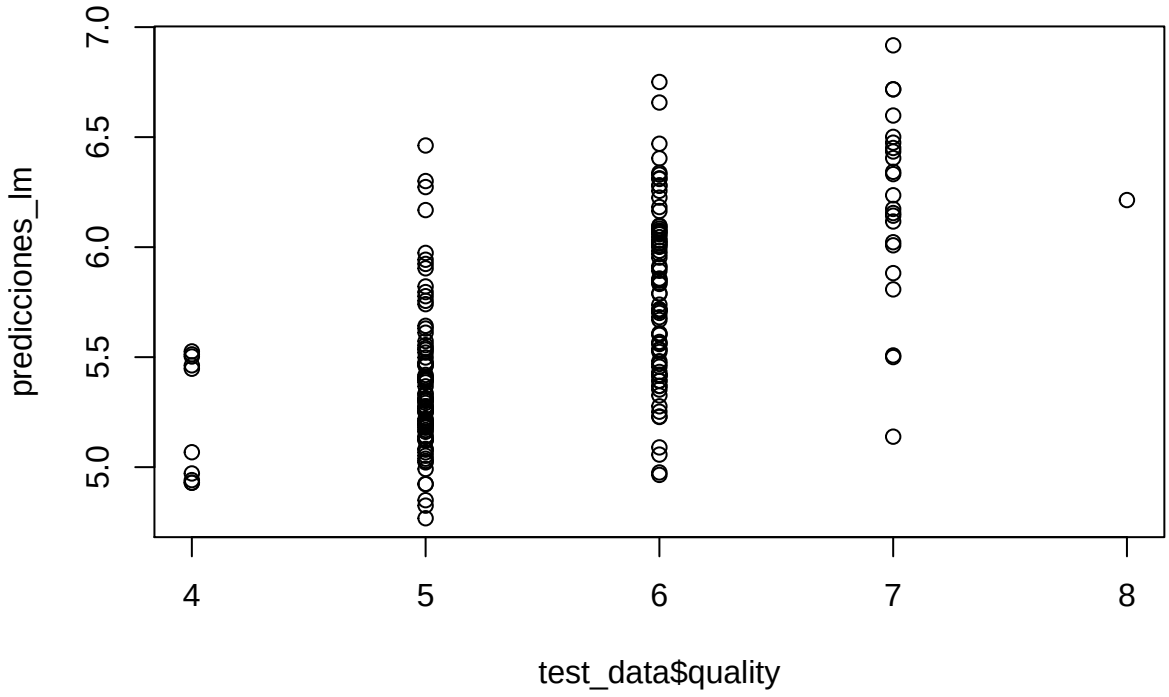
alrededor de una línea horizontal sin patrones distintos indican que tenemos relaciones lineales. La gráfica QQ plot normal muestra los residuos que se ajustan a la línea normal distribuidos. La grafica Scale-Location muestra si los residuos se distribuyen por igual a lo largo de los rangos de predictores de forma que podemos verificar el supuesto de varianza igual (homocedasticidad). Podemos observar una linea horizontal con puntos de dispersión iguales. El gráfico de residuos vs apalancamiento tiene un aspecto típico cuando hay algún caso influyente. Apenas puede ver las líneas de distancia de Cook (una línea punteada roja) porque casi todos los casos están dentro de la distancia de Cook.

Vamos a generar el modelo de regresión lineal con los dos conjuntos de datos, uno de entrenamiento y otro de test, con el objetivo de ver como de bueno es el modelo.

```
# Generamos los dos conjuntos de datos
set.seed(1701)
trainIndex <- createDataPartition(subdatos$quality, p = 0.8, list = FALSE)
train_data <- subdatos[trainIndex, ]
test_data <- subdatos[-trainIndex, ]



# Creamos un modelo reducido
modelo_reducido_lm <- (lm(formula = quality ~ alcohol + sulphates + volatile.acidity, data = train_data)
summary(modelo_reducido_lm)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + sulphates + volatile.acidity,
##     data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.05817 -0.38543 -0.08355  0.45826  1.98769
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.7715     0.2760   6.419 2.37e-10 ***
## alcohol             0.3111     0.0227  13.706  < 2e-16 ***
## sulphates           1.7148     0.2089   8.209 9.15e-16 ***
## volatile.acidity   -0.8465     0.1430  -5.921 4.80e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6109 on 782 degrees of freedom
## Multiple R-squared:  0.3787, Adjusted R-squared:  0.3763
## F-statistic: 158.9 on 3 and 782 DF,  p-value: < 2.2e-16
```

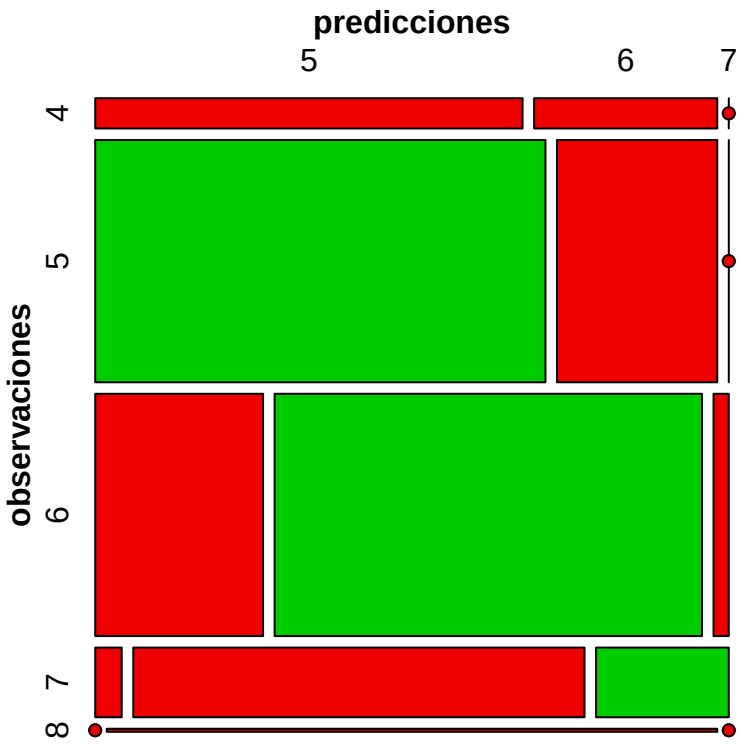
```
# Predicción sobre los datos de test
predicciones_lm <- predict(modelo_reducido_lm, test_data)
plot(test_data$quality, predicciones_lm)
```




```
# Creamos la matriz de confusión de los valores
matriz_confusion_lm <- table(test_data$quality , round(predicciones_lm), dnn = c("observaciones", "predi
matriz_confusion_lm
```

```
##              predicciones
## observaciones  5  6  7
##             4  7  3  0
##             5 59 21  0
##             6 22 56  2
##             7  1 17  5
##             8  0  1  0
```

```
# Matriz de confusión del modelo de regresión lineal con los tres parámetros significativos.
mosaic(matriz_confusion_lm, shade = T, colorize = T, gp = gpar(fill = matrix(c("red2",   "green3",   "r
                                                                              "red2",   "red2",     "gr
                                                                              "red2",   "red2",     "re
```



Podemos observar que los valores obtenidos en la predicción están dentro del rango de valores esperado, por lo que el modelo está realizando correctamente los cálculos. En este caso observamos a partir de la matriz de confusión como el modelo tiene aproximadamente un 62% de acierto en la predicción.

La interpretación del modelo sería una equación lineal, que se puede recrear a partir de los parámetros obtenidos por el modelo.

prediccion = 1.771545 + (0.311126 * alcohol) + (1.714843 * sulphates) - (0.846597 * volatile.acidity)

Regresión logistica

Con el objetivo de tener un modelo que nos permitiera tener un control de calidad excluyente y que nos permitiera decidir si el producto final está dentro de los vinos considerados de alta calidad, con valoración de 6, 7 u 8. Vamos a crear un modelo de regresión logística y compararemos resultados.

```
# Creamos una variable calidad de tipo factor
subdatos$quality_factor <- as.factor(subdatos$quality)

# Creamos un variable calidad binomial donde especificamos que queremos vinos con valoraciones de 6 o ma
subdatos$category[subdatos$quality < 6] <- 0
subdatos$category[subdatos$quality >= 6] <- 1
subdatos$category <- as.factor(subdatos$category)
```

Creamos el modelo de regresión logística

```
# Generamos el modelo con las variables seleccionadas
modelo_log <- glm(category ~ alcohol + sulphates + volatile.acidity, data = subdatos, family = "binomial
summary(modelo_log)
```

```
##
```

```
## Call:
## glm(formula = category ~ alcohol + sulphates + volatile.acidity,
##     family = "binomial", data = subdatos)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6924  -0.8389   0.2923   0.8382   2.3321
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -12.47164    1.11249 -11.211  < 2e-16 ***
## alcohol            1.02284    0.09456  10.817  < 2e-16 ***
## sulphates          5.29518    0.77638   6.820 9.08e-12 ***
## volatile.acidity  -2.29750    0.49250  -4.665 3.09e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1353.0  on 979  degrees of freedom
## Residual deviance: 1025.2  on 976  degrees of freedom
## AIC: 1033.2
##
## Number of Fisher Scoring iterations: 4
```

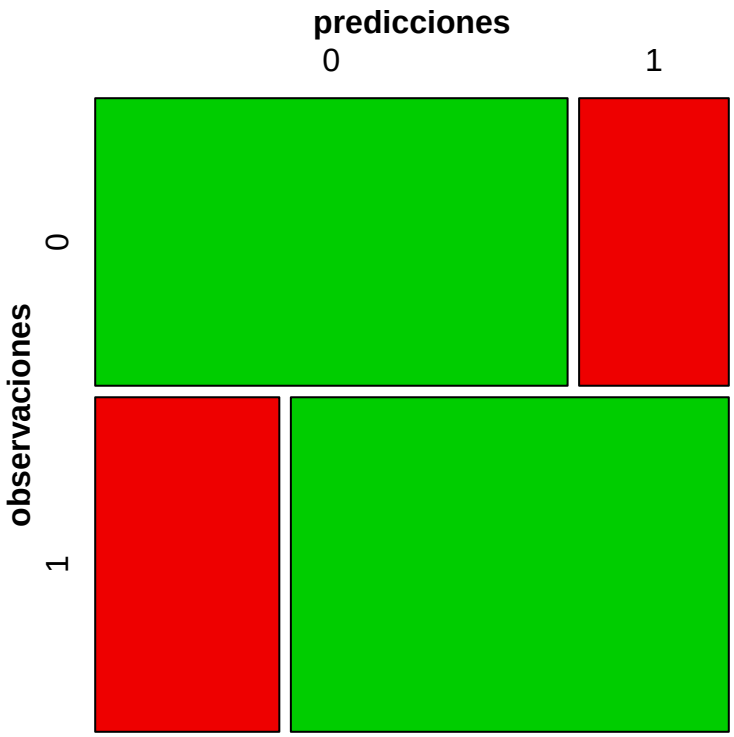
Creamos la tabla de confusión correspondiente al modelo.

```
predicciones <- ifelse(test = modelo_log$fitted.values > 0.5, yes = 1, no = 0)
matriz_confusion <- table(modelo_log$model$category , predicciones, dnn = c("observaciones", "prediccio
matriz_confusion
```

```
##              predicciones
## observaciones   0   1
##             0 344 109
##             1 156 371
```

Visualizamos los resultados.

```
mosaic(matriz_confusion, shade = T, colorize = T, gp = gpar(fill = matrix(c("green3", "red2", "red2", "
```



El modelo de regresión logística nos ha dado un resultado de predicción de un 73% de acierto. Hemos de tener en cuenta que hemos simplificado la predicción a dos valores, considerando los vinos con puntuación de 5 o menos como malos y de 6, 7 u 8 como buenos. En este caso, hemos entrenado el modelo con todos los datos y lo hemos evaluado sobre los mimos datos.

Vamos a crear un modelo predictivo sobre la variable quality, dividiendo los datos en un conjunto de entrenamiento y otro de test. En este caso, vamos a intentar precedir la calidad del vino en todas sus categorias en función de los parámetros.

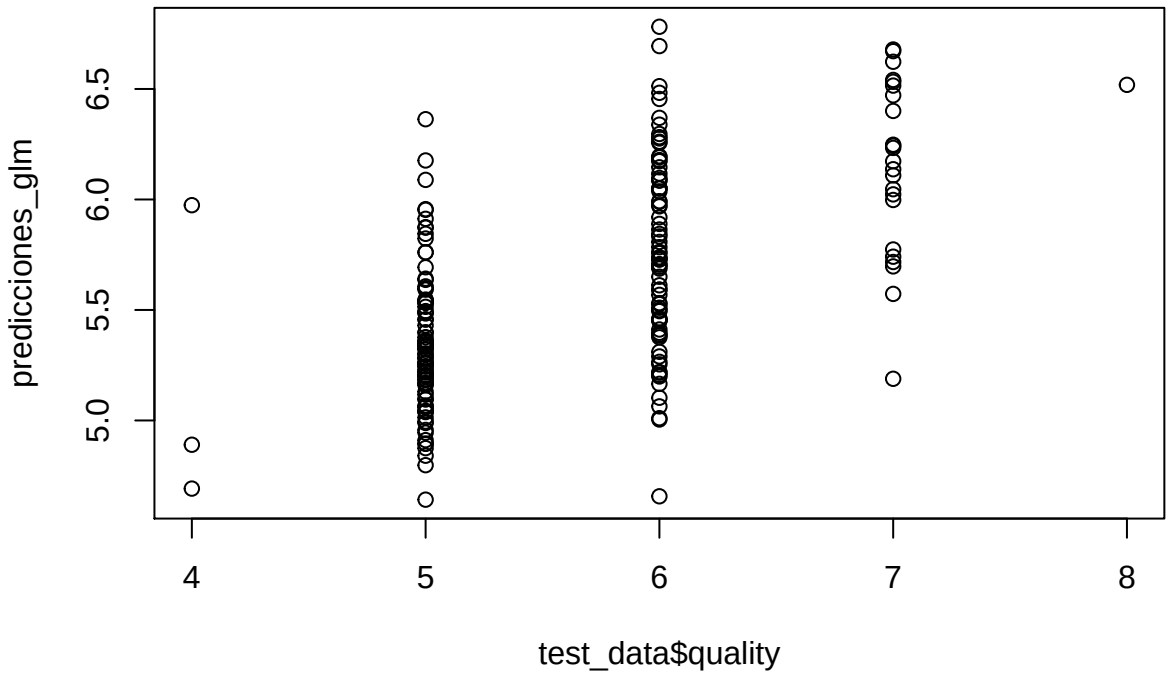
```
# Generamos los dos conjuntos de datos
set.seed(1234)
trainIndex <- createDataPartition(subdatos$quality, p = 0.8, list = FALSE)
train_data <- subdatos[trainIndex, ]
test_data <- subdatos[-trainIndex, ]

# Generamos nuestro modelo de regresión logística con las variables escogidas
modelo_reducido_glm <- glm(quality ~ alcohol + sulphates + volatile.acidity, data = train_data, family =
summary(modelo_reducido_glm)
```

```
##
## Call:
## glm(formula = quality ~ alcohol + sulphates + volatile.acidity,
##     family = "gaussian", data = train_data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.02630  -0.37656  -0.07482   0.46472   1.99944
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.55313    0.27832   5.580 3.31e-08 ***
## alcohol            0.32456    0.02331  13.924  < 2e-16 ***
## sulphates          1.77793    0.21136   8.412  < 2e-16 ***
## volatile.acidity -0.79527    0.13908  -5.718 1.53e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.3825123)
##
##     Null deviance: 485.66  on 785  degrees of freedom
## Residual deviance: 299.12  on 782  degrees of freedom
## AIC: 1481.2
##
## Number of Fisher Scoring iterations: 2
```

Probamos el modelo con los datos de test. Las predicciones son de valores continuos, y en nuestro caso tenemos valores categóricos, por lo que realizaremos un ajuste del resultado por redondeo para obtener la matriz de confusión.

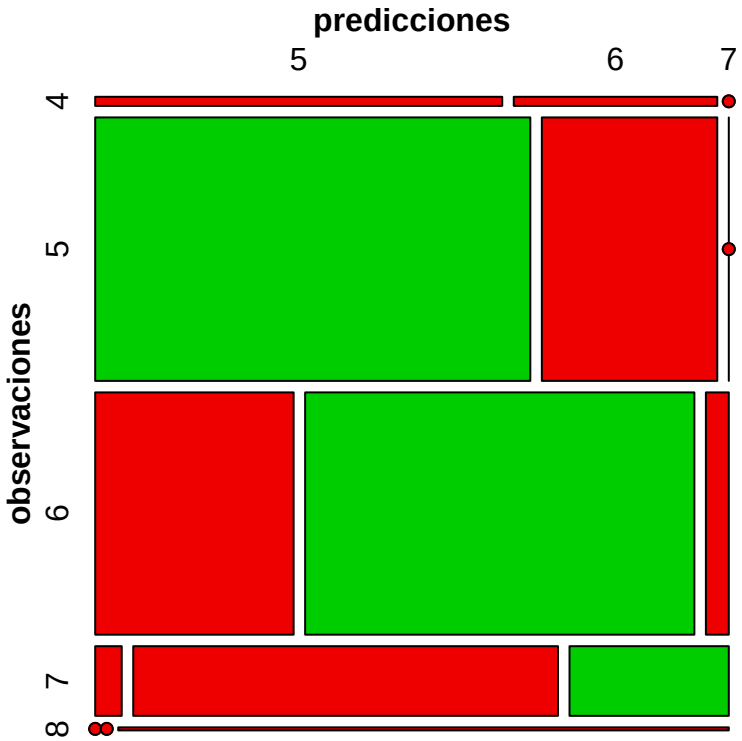
```
# Predicción sobre los datos de test
predicciones_glm <- predict(modelo_reducido_glm, test_data)
plot(test_data$quality, predicciones_glm)
```




```
# Creamos la matriz de confusión de los valores
matriz_confusion_glm <- table(test_data$quality , round(predicciones_glm), dnn = c("observaciones", "pr
matriz_confusion_glm
```

```
##              predicciones
## observaciones  5  6  7
##             4  2  1  0
##             5 62 25  0
##             6 26 51  3
##             7  1 16  6
##             8  0  0  1
```

```
# Visualizamos la matriz de confusión
mosaic(matriz_confusion_glm, shade = T, colorize = T, gp = gpar(fill = matrix(c("red2",    "green3",    "
                                                                                "red2",    "red2",      "gr
                                                                                "red2",    "red2",      "re
```



Podemos observar que los valores obtenidos en la predicción están dentro del rango de valores esperado, por lo que el modelo está realizando correctamente los cálculos. En este caso observamos a partir de la matriz de confusión como el modelo tiene un 61,34% de acierto en la predicción.

Modelos de clasificacion

También nos seria útil tener algún modelo de clasificación que pudiéramos determinar la calidad del vino en función de sus características. Podría agrupar los productos o producción en varios productos de venta, etc... Crearemos un modelo supervisado.

Arbol de decision Necesitaremos prepara los datos para crear un grupo de datos de entrenamiento y de prueba.

```
# Copiamos los datos y eliminamos algunas columnas
datos_arbol <- subdatos
datos_arbol$quality <- NULL
datos_arbol$quality_factor <- NULL
#datos_arbol$category <- NULL
# Creamos los conjuntos de datos de entrenamiento y de test.
set.seed(666)
datos_training <- sample_frac(datos_arbol, .7)
datos_test <- setdiff(datos_arbol, datos_training)
```

Entrenamos el modelo basándonos en la variable categoria

```
arbol <- rpart(formula = category~ ., data = datos_training)
```

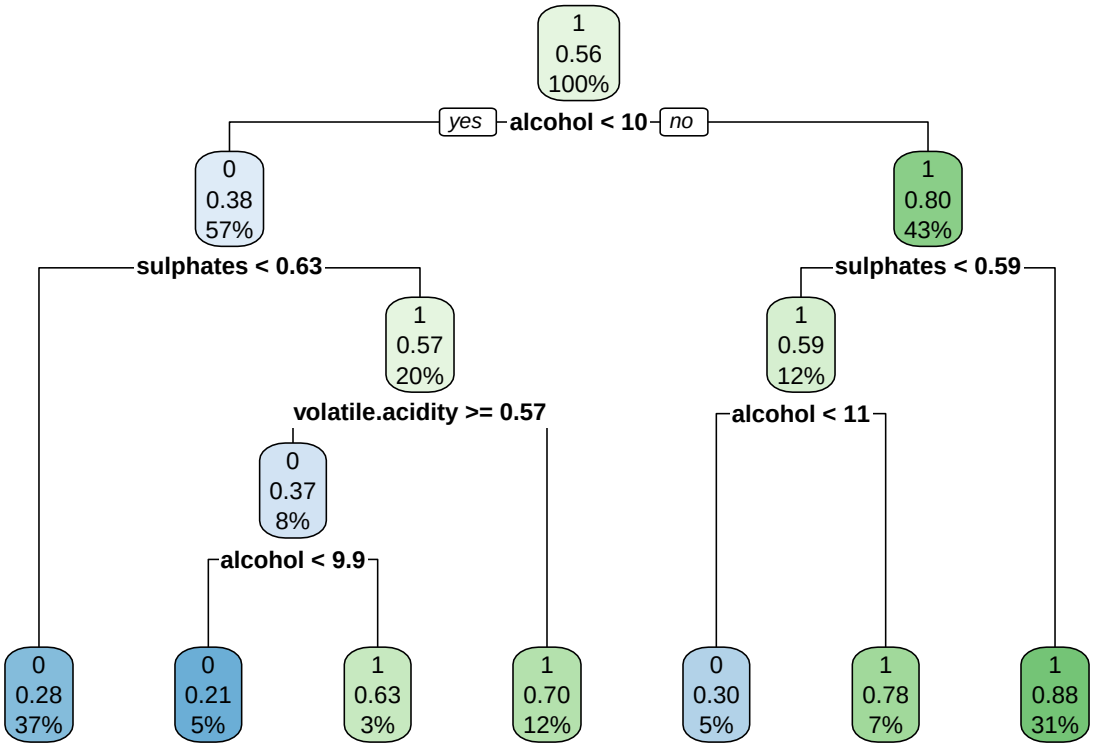
Evaluando el modelo

```
arbol
```

```
## n= 686
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 686 301 1 (0.4387755 0.5612245)
##    2) alcohol< 10.45 388 147 0 (0.6211340 0.3788660)
##      4) sulphates< 0.625 253  70 0 (0.7233202 0.2766798) *
##      5) sulphates>=0.625 135  58 1 (0.4296296 0.5703704)
##       10) volatile.acidity>=0.565 52  19 0 (0.6346154 0.3653846)
##         20) alcohol< 9.85 33   7 0 (0.7878788 0.2121212) *
##         21) alcohol>=9.85 19   7 1 (0.3684211 0.6315789) *
##       11) volatile.acidity< 0.565 83  25 1 (0.3012048 0.6987952) *
##    3) alcohol>=10.45 298  60 1 (0.2013423 0.7986577)
##      6) sulphates< 0.585 82  34 1 (0.4146341 0.5853659)
##       12) alcohol< 11.1 33  10 0 (0.6969697 0.3030303) *
##       13) alcohol>=11.1 49  11 1 (0.2244898 0.7755102) *
##      7) sulphates>=0.585 216  26 1 (0.1203704 0.8796296) *
```

Mostramos el arbol de desicion con la funcion plot

```
rpart.plot(arbol)
```



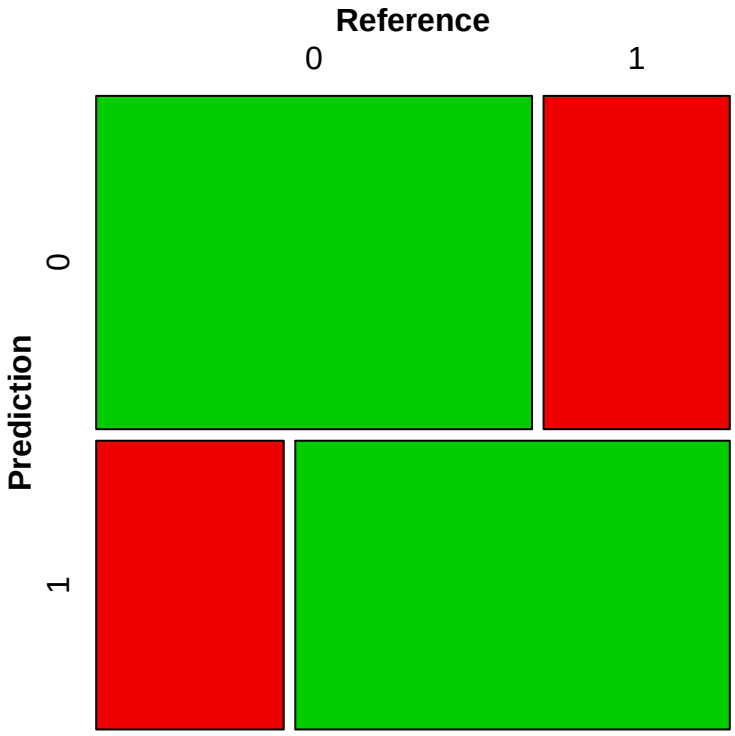
Creamos la matrix de confusion para evaluar el arbol.

```
prediccion <- predict(arbol, newdata = datos_test, type = "class")
confusionMatrix(prediccion, datos_test[["category"]])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 110  47
##          1  41  95
##
##                Accuracy : 0.6997
##                  95% CI : (0.6436, 0.7516)
##     No Information Rate : 0.5154
##     P-Value [Acc > NIR] : 1.076e-10
##
##                   Kappa : 0.398
##
##  Mcnemar's Test P-Value : 0.594
##
##             Sensitivity : 0.7285
##             Specificity : 0.6690
##          Pos Pred Value : 0.7006
##          Neg Pred Value : 0.6985
##              Prevalence : 0.5154
##          Detection Rate : 0.3754
```

```
##    Detection Prevalence : 0.5358
##       Balanced Accuracy : 0.6987
##
##        'Positive' Class : 0
##
```

```
confusionMatrix_table <- confusionMatrix(prediccion, datos_test[["category"]])$table

mosaic(confusionMatrix_table, shade = T, colorize = T, gp = gpar(fill = matrix(c("green3", "red2", "red2
```



Vemos que la precisión del modelo es 69.73%. Con un valor del factor Kappa de 0.3946

Representación de los resultados

A partir del conjunto de datos inicial, y tras una revisión y eliminación de valores perdidos, elementos duplicados y valores extremos, se ha obtenido un conjunto de datos limpios con el cual empezar a analizar.

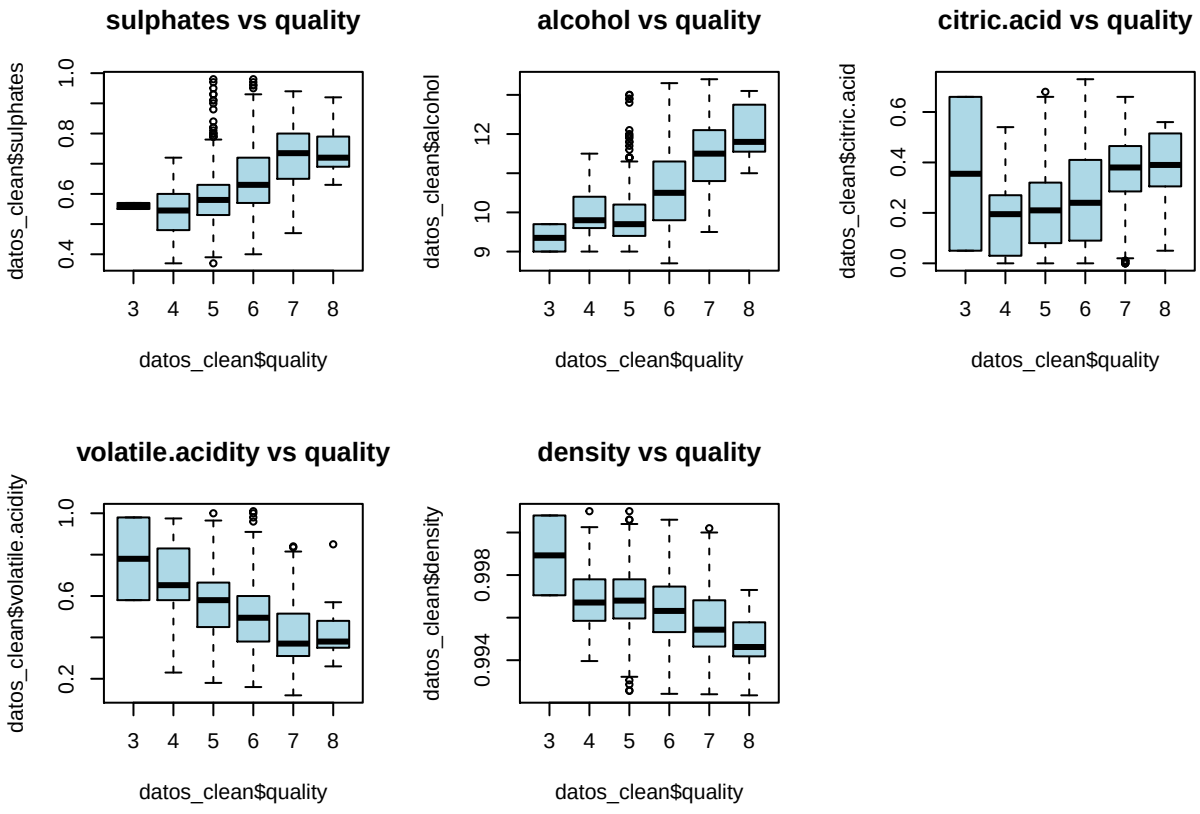
```
summary(datos_clean)
```

```
##  fixed.acidity    volatile.acidity  citric.acid    residual.sugar
##  Min.   : 5.100   Min.   :0.1200   Min.   :0.000   Min.   :1.200
##  1st Qu.: 7.100   1st Qu.:0.3900   1st Qu.:0.080   1st Qu.:1.900
##  Median : 7.800   Median :0.5200   Median :0.240   Median :2.100
##  Mean   : 8.155   Mean   :0.5216   Mean   :0.249   Mean   :2.196
##  3rd Qu.: 9.000   3rd Qu.:0.6300   3rd Qu.:0.400   3rd Qu.:2.500
##  Max.   :12.300   Max.   :1.0100   Max.   :0.730   Max.   :3.650
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.03900   Min.   : 1.00       Min.   :  6.00       Min.   :0.9923
```

```
##  1st Qu.:0.06900   1st Qu.: 8.00       1st Qu.: 22.00       1st Qu.:0.9955
##  Median :0.07800   Median :13.00       Median : 36.00       Median :0.9965
##  Mean   :0.07841   Mean   :15.01       Mean   : 42.38       Mean   :0.9965
##  3rd Qu.:0.08700   3rd Qu.:20.00       3rd Qu.: 56.00       3rd Qu.:0.9976
##  Max.   :0.12200   Max.   :42.00       Max.   :124.00       Max.   :1.0010
##        pH          sulphates         alcohol         quality
##  Min.   :2.940   Min.   :0.3700   Min.   : 8.70   Min.   :3.00
##  1st Qu.:3.230   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.00
##  Median :3.320   Median :0.6100   Median :10.10   Median :6.00
##  Mean   :3.322   Mean   :0.6313   Mean   :10.39   Mean   :5.64
##  3rd Qu.:3.400   3rd Qu.:0.7000   3rd Qu.:11.00   3rd Qu.:6.00
##  Max.   :3.680   Max.   :0.9800   Max.   :13.40   Max.   :8.00
```

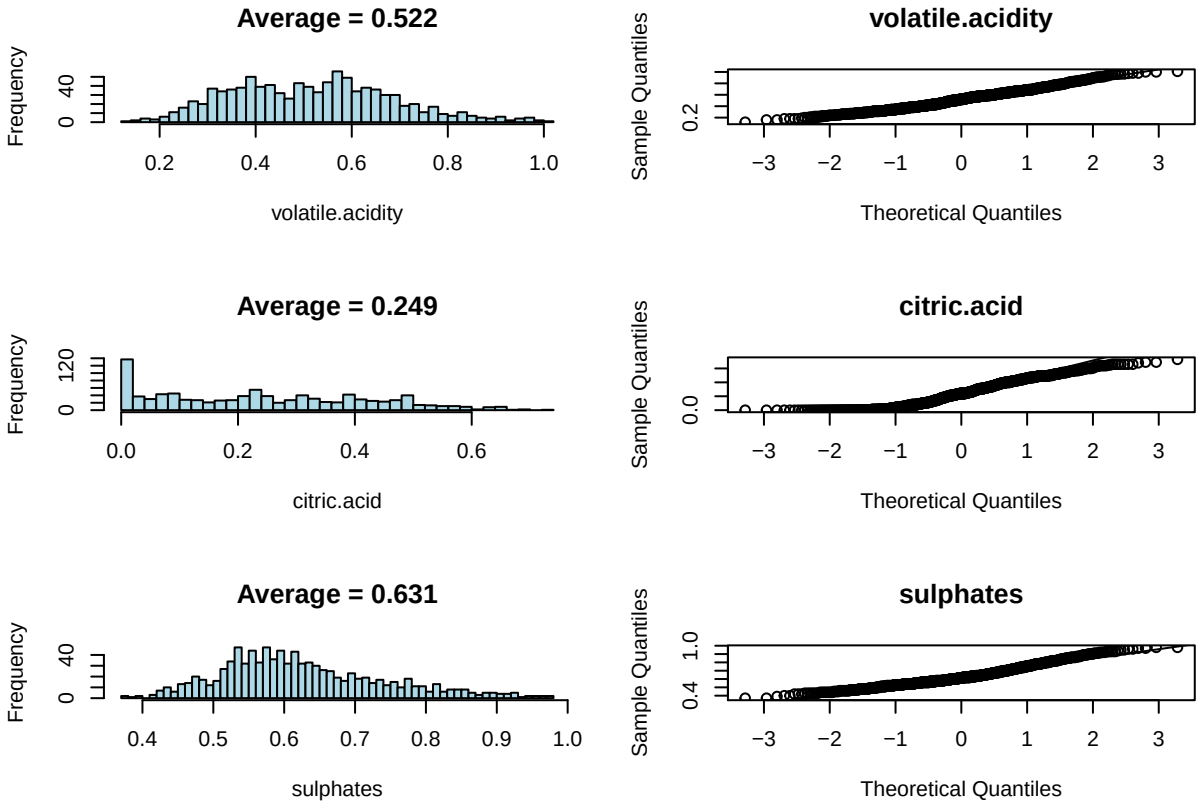
Hemos analizado la distribución de los datos en función de la variable "quality" y hemos identificado visualmente los elementos que afectan significativamente a la calidad del vino, tanto positivamente como negativamente.

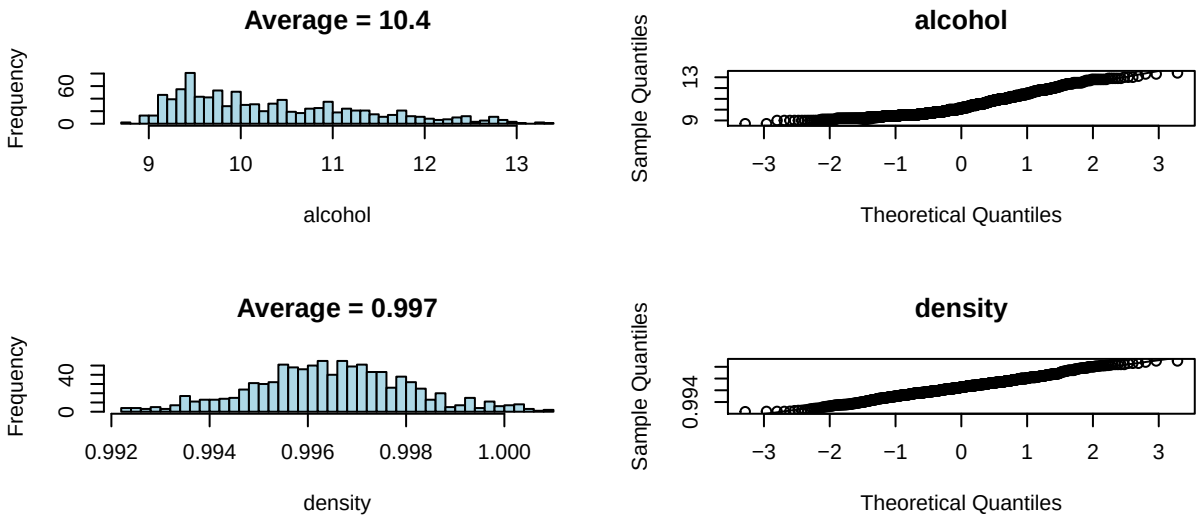
```
par(mfrow=c(2,3))

boxplot(formula = datos_clean$sulphates ~ datos_clean$quality, main="sulphates vs quality", col="lightbl
boxplot(formula = datos_clean$alcohol ~ datos_clean$quality, main="alcohol vs quality", col="lightblue"
boxplot(formula = datos_clean$citric.acid ~ datos_clean$quality, main="citric.acid vs quality", col="lig
boxplot(formula = datos_clean$volatile.acidity ~ datos_clean$quality, main="volatile.acidity vs quality
boxplot(formula = datos_clean$density ~ datos_clean$quality, main="density vs quality", col="lightblue"
```



Hemos analizado la normalidad y homocedasticidad de los datos de este subconjunto.

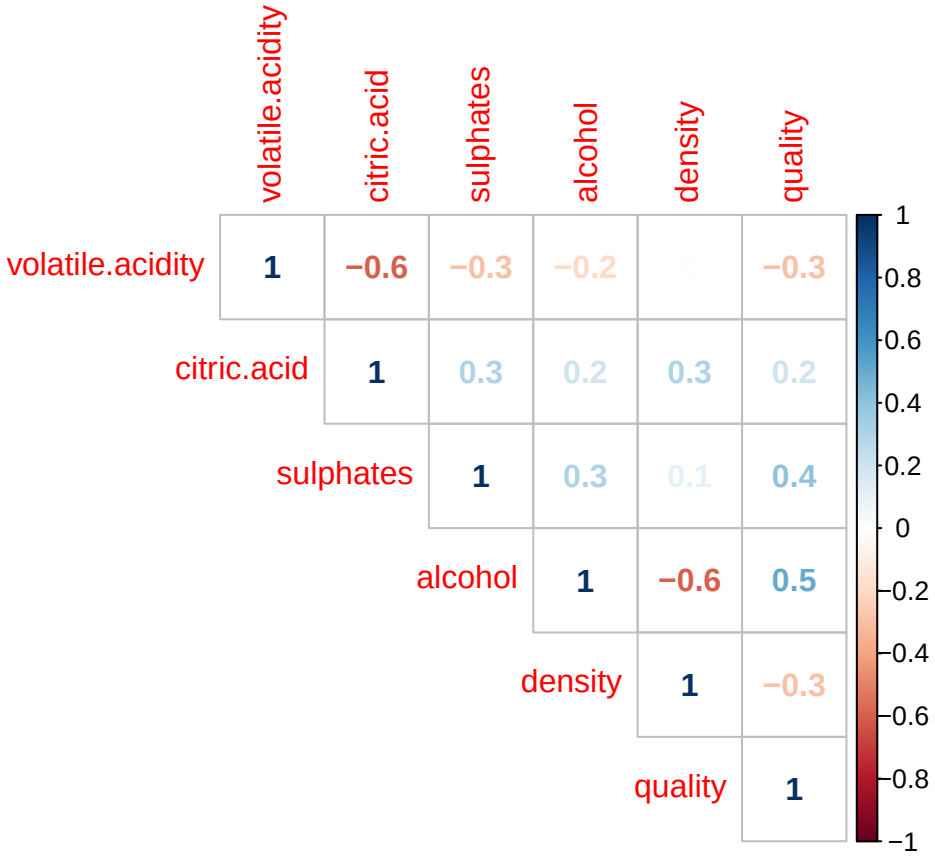
```r
# Gráficas QQ de comprobación de normalidad e histogramas
par(mfrow=c(3,2))
for (i in 1:5) {
  hist(subdatos[[i]], xlab = names(subdatos)[i], col = 'lightblue', main = paste("Average =", signif(mea
  qqnorm(subdatos[,i], main = colnames(subdatos[i]))
  qqline(subdatos[,i])

}
```





Hemos analizado la correlación entre variables.

```
corrplot(correlacion_sc, method="number", type="upper")
```

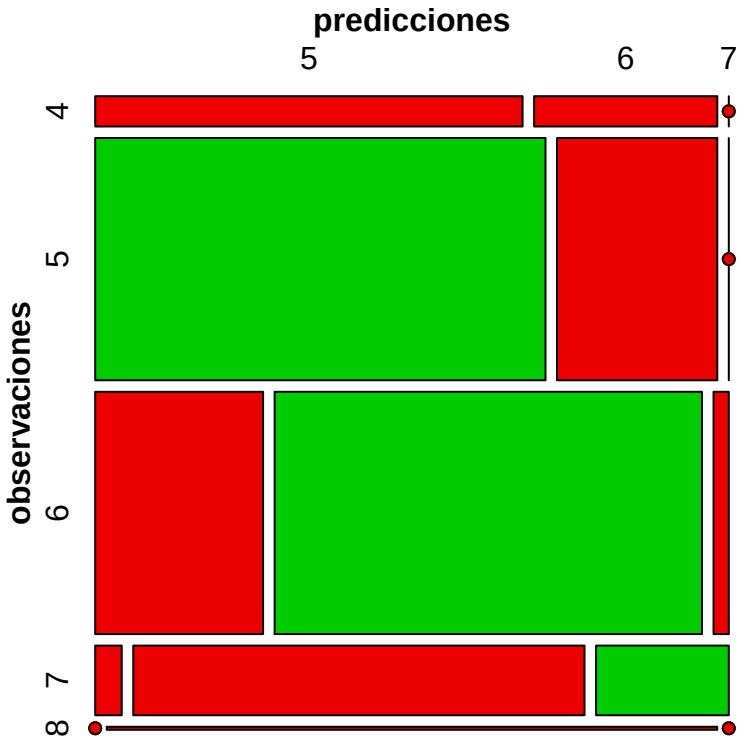


Con toda la información analizada, se ha decidido el analizar la creación de modelos de predicción y clasificación con estas variables.

Tras una primera versión de un modelo de regresión lineal, se ha detectado que el conjunto de variables siginificativas se puede reducir a tres, “alcohol”, “sulphates” y “Volatile.acidity”, sin que afecte al resultado del modelo.

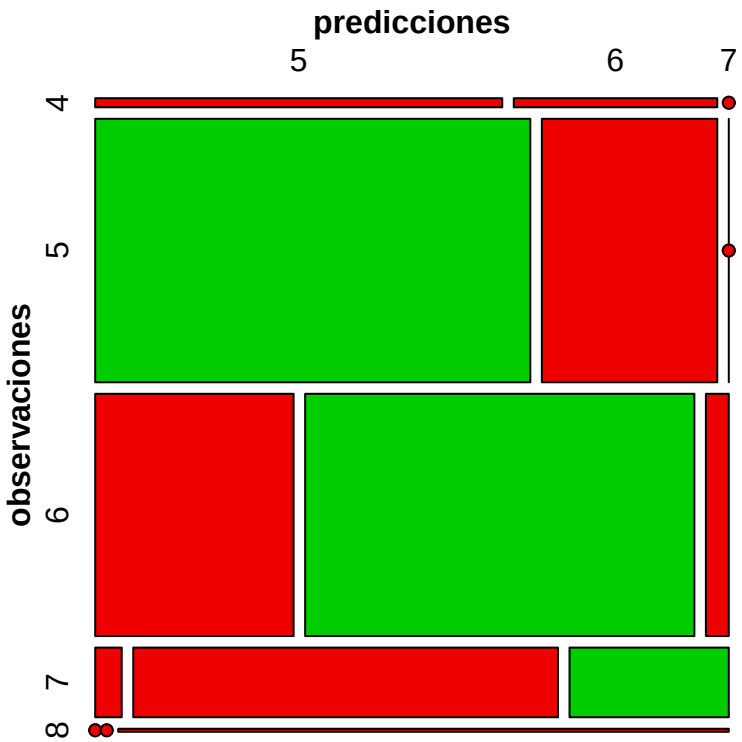
Con estas variables hemos realizado varios modelos.

```
# Matriz de confusión del modelo de regresión lineal con los tres parámetros significativos.
mosaic(matriz_confusion_lm, shade = T, colorize = T, gp = gpar(fill = matrix(c("red2",   "green3",   "r
                                                                               "red2",   "red2",     "gr
                                                                               "red2",   "red2",     "re
```



En este caso observamos a partir de la matriz de confusión como el modelo tiene aproximadamente un 62% de acierto en la predicción.

```
# Matriz de confusión del modelo de regresión logística con los tres parámetros significativos.
mosaic(matriz_confusion_glm, shade = T, colorize = T, gp = gpar(fill = matrix(c("red2",   "green3",   "
                                                                                "red2",   "red2",     "gr
                                                                                "red2",   "red2",     "re
```

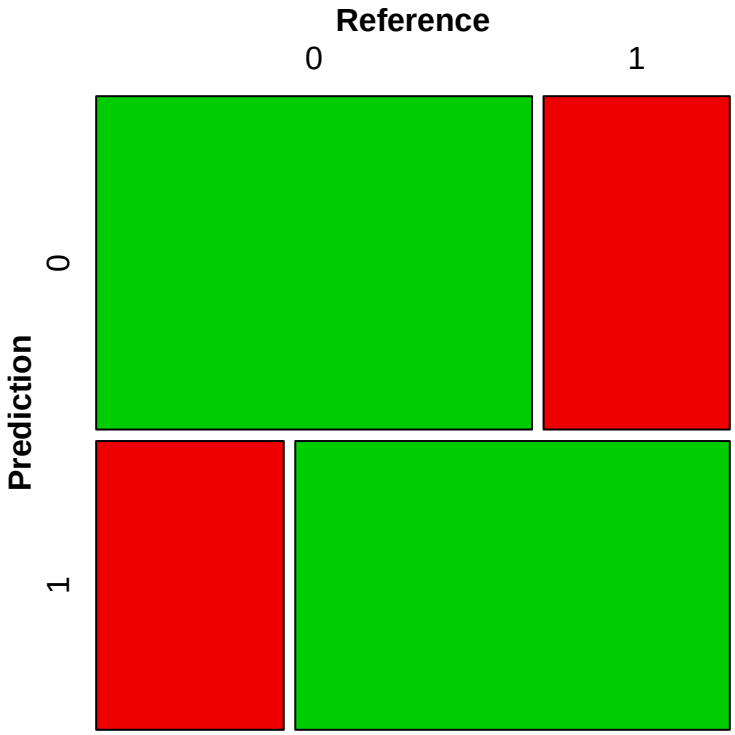


En este caso observamos a partir de la matriz de confusión como el modelo tiene un 61,34% de acierto en la predicción.

Ambos modelos de regresión nos das resultados muy similares.

Vamos a probar con un modelo de clasificación por árbol de decisión.

```
# Matriz de confusión del modelo de clasificación por árbol de decisión con los tres parámetros signifi
mosaic(confusionMatrix_table, shade = T, colorize = T, gp = gpar(fill = matrix(c("green3", "red2", "red
```



Vemos que en este caso hemos mejorado la precisión del modelo al aumentarla hasta el 69.73%.

Resolución del problema

Como conclusión, podemos decir que hay tres variables más representativas que el resto en la clasificación de calidad del vino, estas son “alcohol”, “sulphates” y “Volatile.acidity”.

A partir de dichas variables, y mediante un modelo de clasificación por árbol de decisión, podemos obtener con una predicción de la calidad del vino, con un nivel de acierto del 70%.