PRÁCTICA 2

Autores:

Manuel Ed. Escobar Ubreva

Elena Cantón García

Link Github:

https://github.com/elenacanton/PRA2_R.git

Ficheros:

"winequality-red.csv", se trata del fichero origen sobre el que vamos a realizar la practica propuesta.

Enlace para su descarga:

https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009?select=winequality-red.csv

Código R:

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

El conjunto de datos objeto de análisis se ha obtenido a partir de este enlace en Kaggle. El dataset está formado por 12 variables (columnas) y 1599 registros (filas).

Estas son las variables del conjunto de datos:

1 - fixed acidity

2 - volatile acidity

3 - citric acid

4 - residual sugar

5 - chlorides

6 - free sulfur dioxide

7 - total sulfur dioxide

8 - density

9 - pH

10 - sulphates

11 - alcohol Output variable (based on sensory data):

12 - quality (score between 0 and 10)

La principal característica de estos datos es la calidad. Con este estudio se pretende esclarecer cuáles de las características afectan a la puntuación de dicha calidad, con valores de entre 0 y 10.

Además, en la descripción del dataset publicado, ya se nos informa de que no hay valores perdidos.

```
# Carga de los datos
#setwd("C:/")
data <- read.csv('winequality-red.csv', sep=',', header = TRUE)
head(data)
```

```
##   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
## 6         7.4          0.66       0.00          1.8    0.075
##   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
## 6              13                 40  0.9978 3.51     0.56     9.4
##   quality
## 1       5
## 2       5
## 3       5
## 4       6
## 5       5
## 6       5
```

Mostramos el resumen del conjunto de datos.

```
str(data)
```

```
## '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 ...
```

La variable str nos permite ver que hay 12 variables y 1599 observaciones. La variable respuesta es “quality”, la calidad del vino. El conjunto cuenta con 11 variables predictoras de clase numérica, y 1 de tipo integer, quality.

Además, vamos a realizar un análisis exploratorio de los datos. La función summary nos proporciona un resumen estadístico sobre las variables (mínimo, máximo, media, mediana, primer y tercer cuartil). Además, si hay valores faltantes también nos informará.

```
#Revisamos el dataset.
summary(data)
```

```
##  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
```

Enumeramos el número de variables del data set.

```
colnames(data)
```

```
##  [1] "fixed.acidity"       "volatile.acidity"    "citric.acid"
##  [4] "residual.sugar"      "chlorides"           "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"            "pH"
## [10] "sulphates"           "alcohol"             "quality"
```

Visualizamos el tipo de datos de cada variable del conjunto de datos.

```
# Tipo de dato asignado a cada campo
sapply(data, 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 son todas numéricas.

2. Integración y selección de los datos de interés a analizar.

En el caso de los datos utilizados para esta practica solamente hay un dataset. La integración o fusión de los datos consiste en la combinación de datos procedentes de múltiples fuentes, con el fin de crear una estructura de datos coherente y única que contenga mayor cantidad de información. De lo anterior, el aparatado de integración no aplica al conjunto de datos seleccionados.

selección de variables: Ideas a valorar solo analizar las muestras cuya calidad sea superior al 7

En esta fase también es habitual realizar una exploración de los datos (screening, en inglés), con el objetivo de analizar globalmente sus características e identificar fuertes correlaciones entre atributos, de modo que se pueda prescindir de aquella información más redundante.

Idea a valorar: podríamos analizar las variables. análisis de componentes principales (ACP), apartado análisis de componentes principales (ACP),

```
##Análisis de componentes principales (ACP),
data.pca <- prcomp(data[,c(1:11)], center = TRUE, scale = TRUE)
summary(data.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.7604 1.3878 1.2452 1.1015 0.97943 0.81216 0.76406
## Proportion of Variance 0.2817 0.1751 0.1410 0.1103 0.08721 0.05996 0.05307
## Cumulative Proportion  0.2817 0.4568 0.5978 0.7081 0.79528 0.85525 0.90832
##                            PC8     PC9    PC10    PC11
## Standard deviation     0.65035 0.58706 0.42583 0.24405
## Proportion of Variance 0.03845 0.03133 0.01648 0.00541
## Cumulative Proportion  0.94677 0.97810 0.99459 1.00000
```

El resultado son 11 componentes principales (PC1-PC11), cada una de las cuales explica un porcentaje de varianza del dataset original. Así, la primera componente principal explica un

poco menos de las 2/3 de la varianza total y las 5 primeras componentes describen en torno al 80 % de la varianza. Dado que las ocho primeras componentes ya explican en torno al 95 % de la varianza,

3. Limpieza de los datos.

3.1. ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

Tras aplicar la función summary, hemos podido observar que no hay ningún valor perdido en ninguna de las variables del dataset, aunque a veces los datasets, en lugar de contener NA o simplemente no tener ningún valor asignado (campo vacío), tienen 0. Sin embargo en este caso, podemos observar que solo la variable Citric acid tiene algún valor 0.

Igualmente vamos a proceder a realizar algunos test para la gestión de estos valores.

```
# Números de valores desconocidos por campo
sapply(data, function(x) sum(is.na(x)))
```

```
##       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
```

Se observa que no existen valores NA en el dataset, lo cual coincide con la descripción localizada en la url donde se aloja el conjunto de datos estudiado.

A continuación analizamos las variables con valores igual a 0 dentro del conjunto de datos.

```
#vemos qué porcentaje por columnas tienen los valores 0
colSums(data==0)/nrow(data)*100
```

```
##       fixed.acidity    volatile.acidity         citric.acid
##            0.000000            0.000000            8.255159
##      residual.sugar           chlorides free.sulfur.dioxide
##            0.000000            0.000000            0.000000
## total.sulfur.dioxide             density                  pH
##            0.000000            0.000000            0.000000
##           sulphates             alcohol             quality
##            0.000000            0.000000            0.000000
```

```
# Números de valores 0 por campo
colSums(data==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
```

Respuesta: Se observa que existen valores 0, en la variable “citric.acid” hay 132 observaciones igual a 0.

Analizamos la posible existencia de valores no informados.

```
# Número de valores empty por campo
sapply(data, function(x) sum(data==""))
```

```
##      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
```

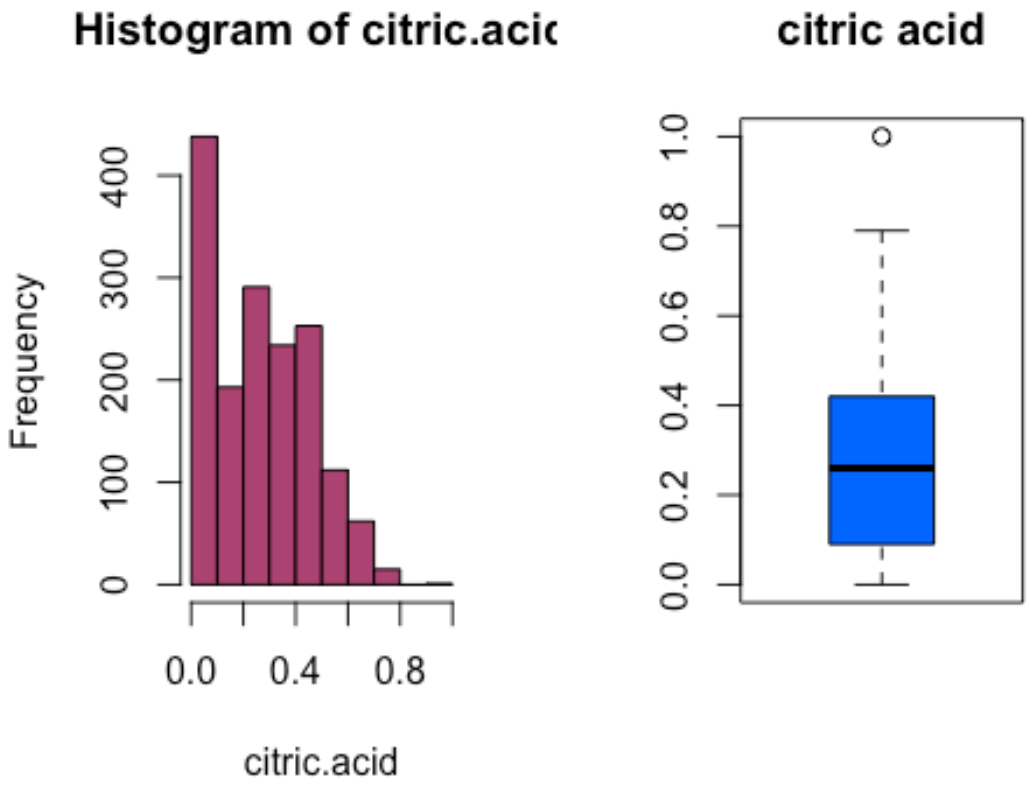
Respuesta: Se observa que no existen campos no informados en el conjunto de datos.

3.2. Identificación y tratamiento de valores extremos.

Respuesta: a continuación vamos a explorar más detalladamente las variables para analizar la existencia de valores extremos.

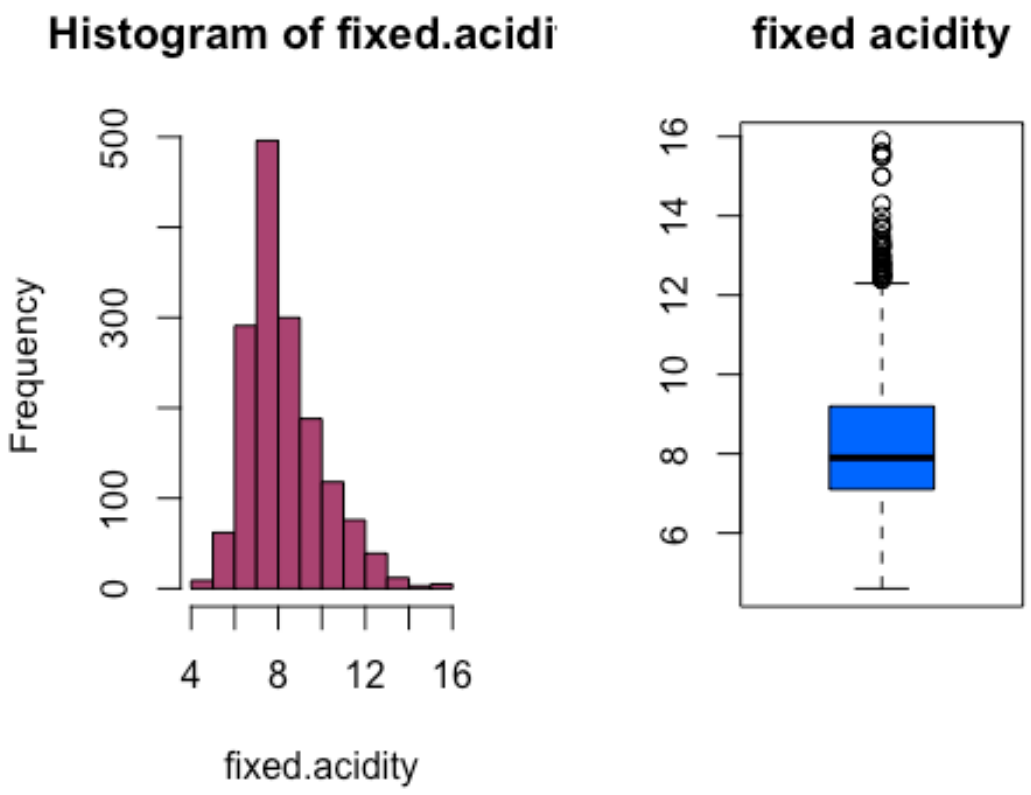
Analizamos la distribución de los valores y realizamos un box-plot y un histograma por cada una de las variables.

```
attach(data)

#Observamos la distribución de los valores de la variable citric.acid
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(citric.acid, col=(c("#AA4371")))
boxplot(citric.acid, main="citric acid", col=(c("#0066FF")))
```



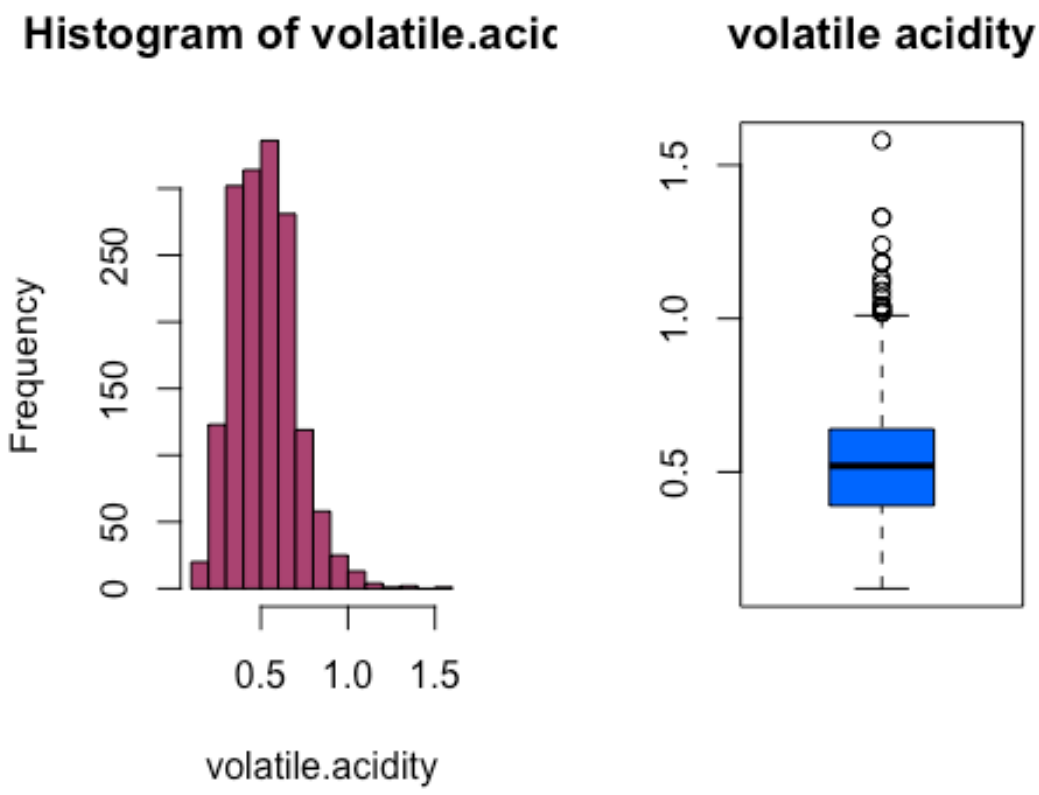
En la Variable “citric.acid” se entiende que las observaciones pueden ser valores reales y que están correctamente informados. De lo anterior no aplicaremos transformaciones para estos datos.

```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(fixed.acidity, col=(c("#AA4371")))
boxplot(fixed.acidity, main="fixed acidity",col=(c("#0066FF")))
```



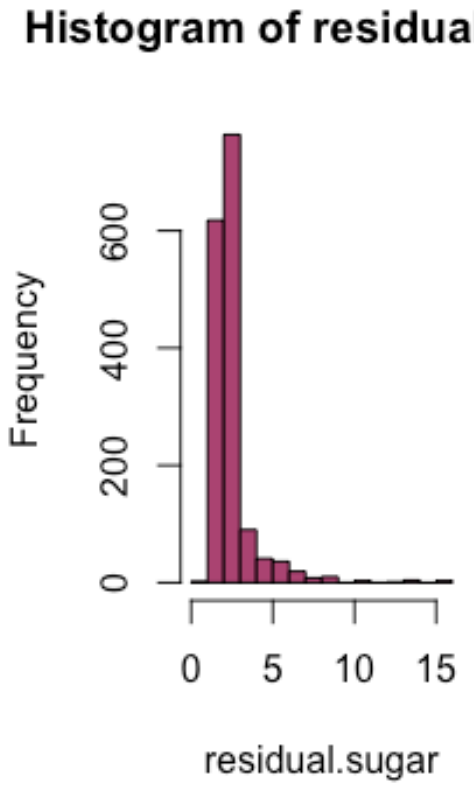
En la Variable “fixed.acidity” se entiende que las observaciones pueden ser valores reales y que están correctamente informados. De lo anterior no aplicaremos transformaciones para estos datos.

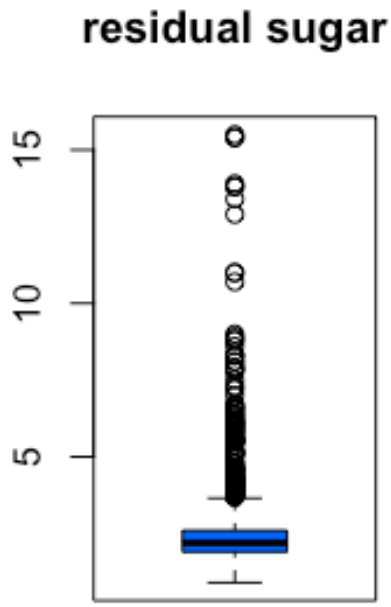
```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(volatile.acidity, col=(c("#AA4371")))
boxplot(volatile.acidity, main="volatile acidity",col=(c("#0066FF")))
```



En la Variable “volatile.acidity” se entiende que las observaciones pueden ser valores reales y que están correctamente informados. De lo anterior no aplicaremos transformaciones para estos datos.

```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(residual.sugar, col=(c("#AA4371")))
boxplot(residual.sugar, main="residual sugar", col=(c("#0066FF")))
```



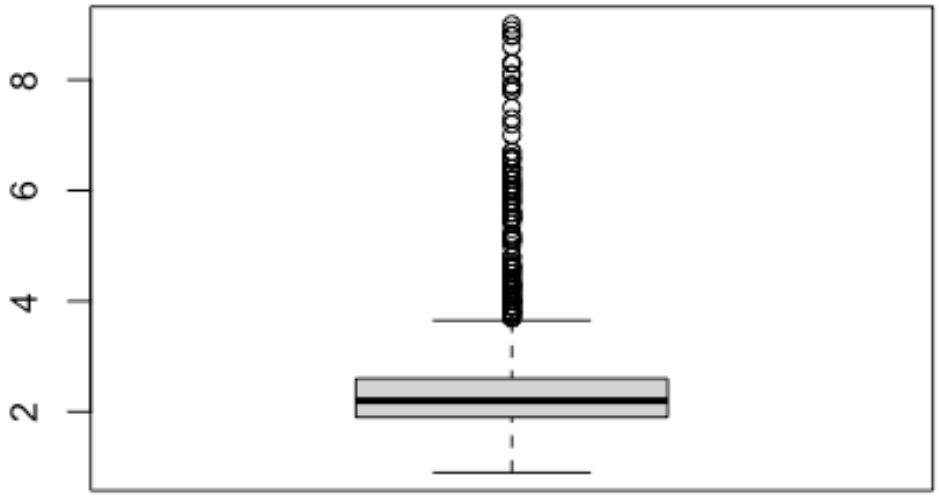



```
#existen más de 150 valores extremos en esta variable. Vamos a visualizarlos:
boxplot.stats(residual.sugar)$out
```

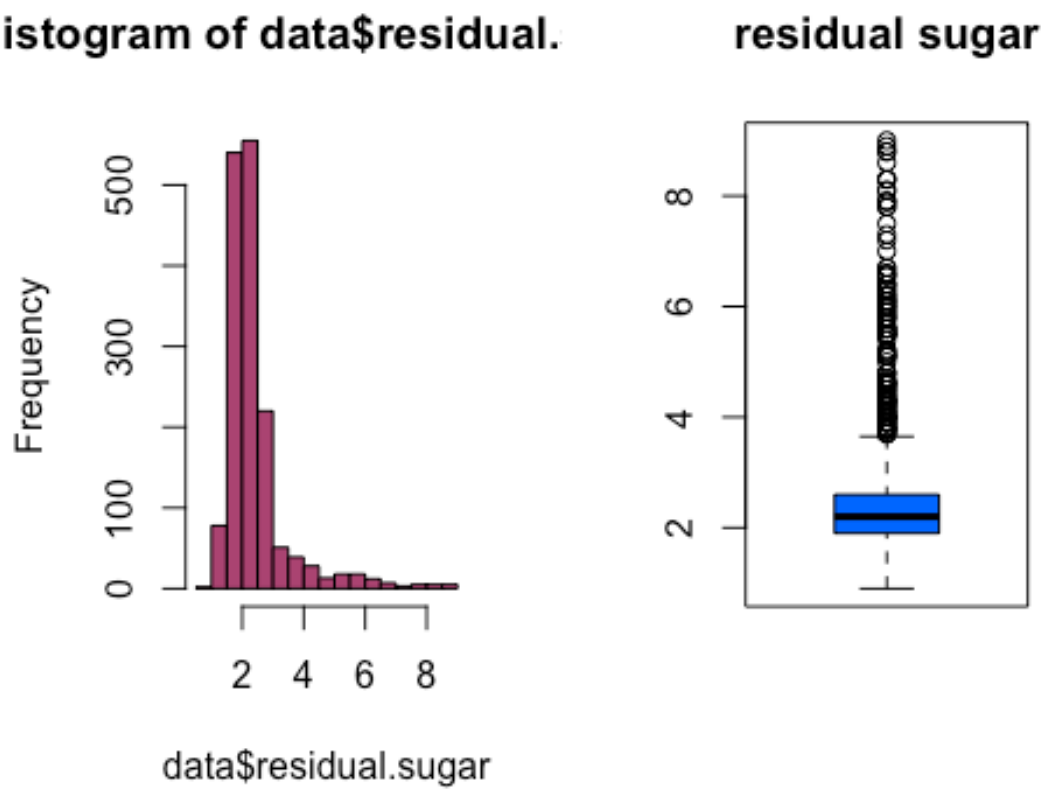
```
##   [1]  6.10  6.10  3.80  3.90  4.40 10.70  5.50  5.90  5.90  3.80  5.10  4.65
##  [13]  4.65  5.50  5.50  5.50  5.50  7.30  7.20  3.80  5.60  4.00  4.00  4.00
##  [25]  4.00  7.00  4.00  4.00  6.40  5.60  5.60 11.00 11.00  4.50  4.80  5.80
##  [37]  5.80  3.80  4.40  6.20  4.20  7.90  7.90  3.70  4.50  6.70  6.60  3.70
##  [49]  5.20 15.50  4.10  8.30  6.55  6.55  4.60  6.10  4.30  5.80  5.15  6.30
##  [61]  4.20  4.20  4.60  4.20  4.60  4.30  4.30  7.90  4.60  5.10  5.60  5.60
##  [73]  6.00  8.60  7.50  4.40  4.25  6.00  3.90  4.20  4.00  4.00  4.00  6.60
##  [85]  6.00  6.00  3.80  9.00  4.60  8.80  8.80  5.00  3.80  4.10  5.90  4.10
##  [97]  6.20  8.90  4.00  3.90  4.00  8.10  8.10  6.40  6.40  8.30  8.30  4.70
## [109]  5.50  5.50  4.30  5.50  3.70  6.20  5.60  7.80  4.60  5.80  4.10 12.90
## [121]  4.30 13.40  4.80  6.30  4.50  4.50  4.30  4.30  3.90  3.80  5.40  3.80
## [133]  6.10  3.90  5.10  5.10  3.90 15.40 15.40  4.80  5.20  5.20  3.75 13.80
## [145] 13.80  5.70  4.30  4.10  4.10  4.40  3.70  6.70 13.90  5.10  7.80
```

Se decide que los valores por encima de 10 van a ser considerados outliers.

```
#asignamos el valor NA a los casos que van a ser considerados outliers
data$residual.sugar[data$residual.sugar>10 ] <- NA
boxplot(data$residual.sugar)
```

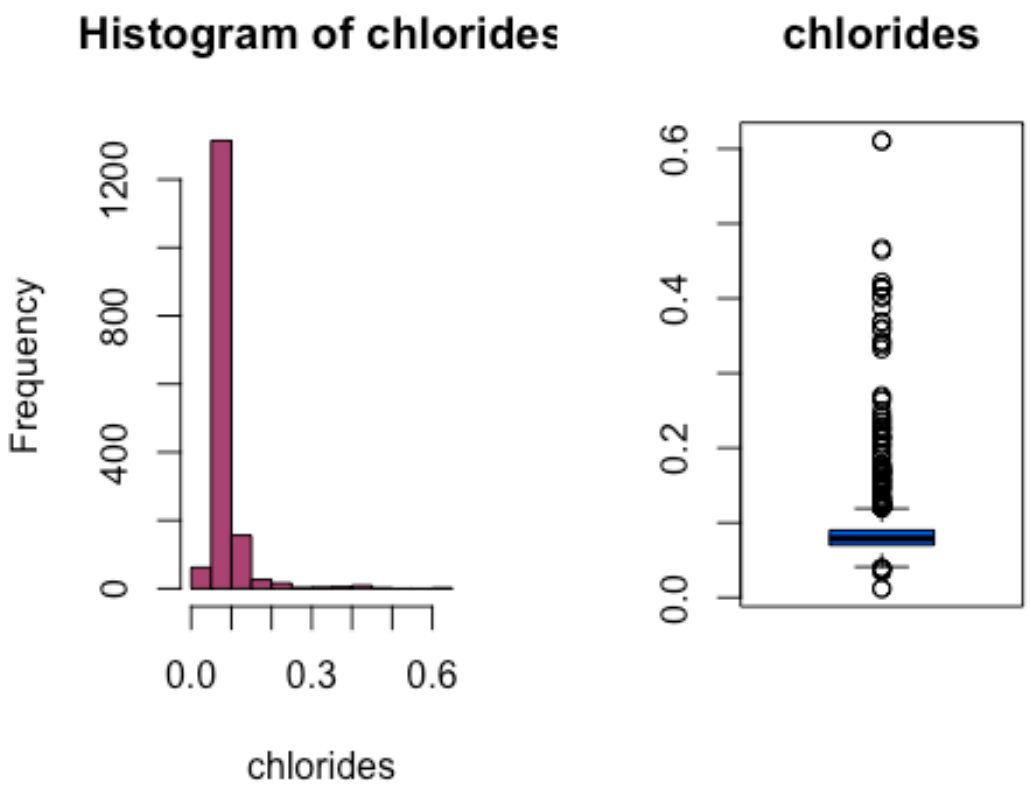



```
idx <- which(is.na(data$residual.sugar))
length(idx) #número de valores perdidos

## [1] 11

#Imputamos el valor de la mediana a los valores NA.
for (i in 1:length(idx)){
index <- idx[i]
data[index,]$residual.sugar <- median( data$residual.sugar, na.rm=TRUE ) #imputación
}
#mostramos los valores imputados a los NA
data$residual.sugar[idx] #mostramos el resultado

##  [1] 2.2 2.2 2.2 2.2 2.2 2.2 2.2 2.2 2.2 2.2 2.2

par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(data$residual.sugar, col=(c("#AA4371")))
boxplot(data$residual.sugar, main="residual sugar", col=(c("#0066FF")))
```



Volvemos a mostrar la distribución de los datos de la variable residual.sugar , para comprobar que no existen valores por encima de 10.

```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(chlorides, col=(c("#AA4371")))
boxplot(chlorides, main="chlorides", col=(c("#0066FF")))
```



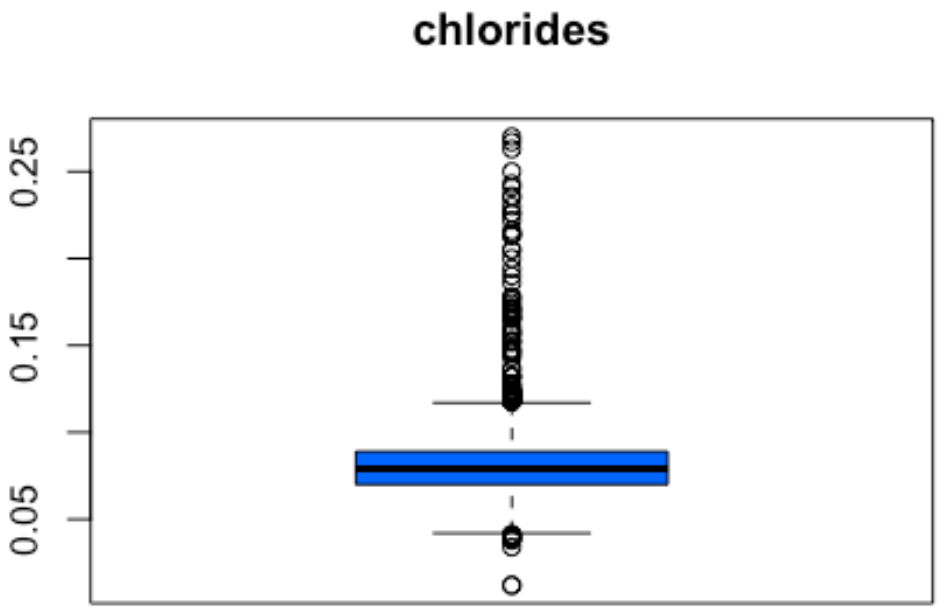

```
#existen más de 100 valores extremos en esta variable
boxplot.stats(chlorides)$out
```

```
##   [1] 0.176 0.170 0.368 0.341 0.172 0.332 0.464 0.401 0.467 0.122 0.178 0.146
##  [13] 0.236 0.610 0.360 0.270 0.039 0.337 0.263 0.611 0.358 0.343 0.186 0.213
##  [25] 0.214 0.121 0.122 0.122 0.128 0.120 0.159 0.124 0.122 0.122 0.174 0.121
##  [37] 0.127 0.413 0.152 0.152 0.125 0.122 0.200 0.171 0.226 0.226 0.250 0.148
##  [49] 0.122 0.124 0.124 0.143 0.222 0.039 0.157 0.422 0.034 0.387 0.415 0.157
##  [61] 0.157 0.243 0.241 0.190 0.132 0.126 0.038 0.165 0.145 0.147 0.012 0.012
##  [73] 0.039 0.194 0.132 0.161 0.120 0.120 0.123 0.123 0.414 0.216 0.171 0.178
##  [85] 0.369 0.166 0.166 0.136 0.132 0.132 0.123 0.123 0.123 0.403 0.137 0.414
##  [97] 0.166 0.168 0.415 0.153 0.415 0.267 0.123 0.214 0.214 0.169 0.205 0.205
## [109] 0.039 0.235 0.230 0.038
```

De manera análoga a la variable anterior, se decide considerar outliers a los valores superiores a 0,3. A estas observaciones se les asignará el valor de la mediana.

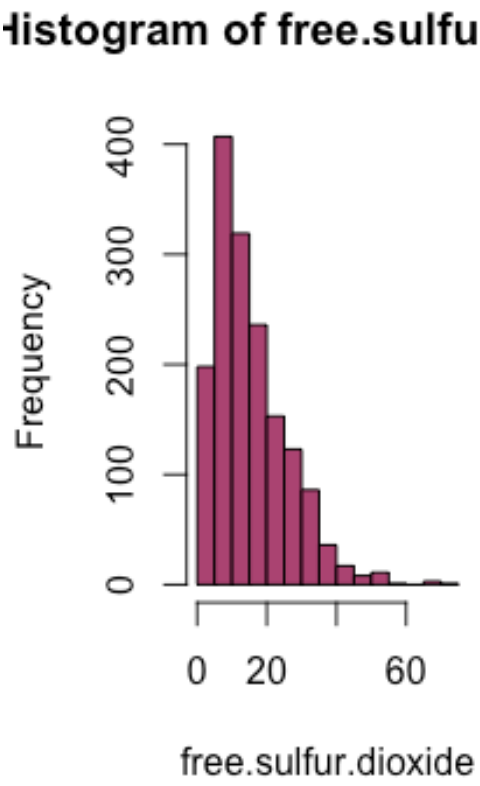
```
#asignamos el valor NA a los casos que van a ser considerados outliers
data$chlorides[data$chlorides>0.3 ] <- NA

idx <- which(is.na(data$chlorides))
length(idx) #número de valores perdidos
```

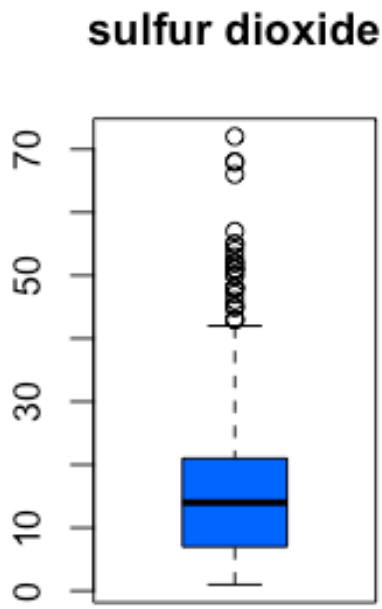
```
## [1] 22

#Imputamos el valor de la mediana a los valores NA.
for (i in 1:length(idx)){
index <- idx[i]
data[index,]$chlorides <- median( data$chlorides, na.rm=TRUE ) #imputación
}

boxplot(data$chlorides, main="chlorides", col=(c("#0066FF")))
```



Volvemos a mostrar la distribución de los datos de la variable chlorides , para comprobar que no existen valores por encima de 10.

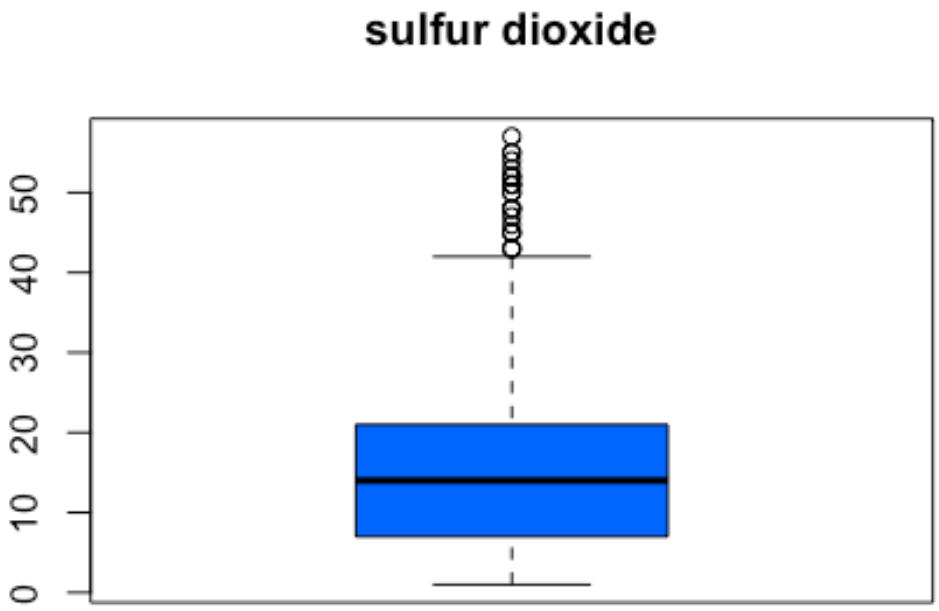
```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(free.sulfur.dioxide, col=(c("#AA4371")))
boxplot(free.sulfur.dioxide, main="sulfur dioxide", col=(c("#0066FF")))
```





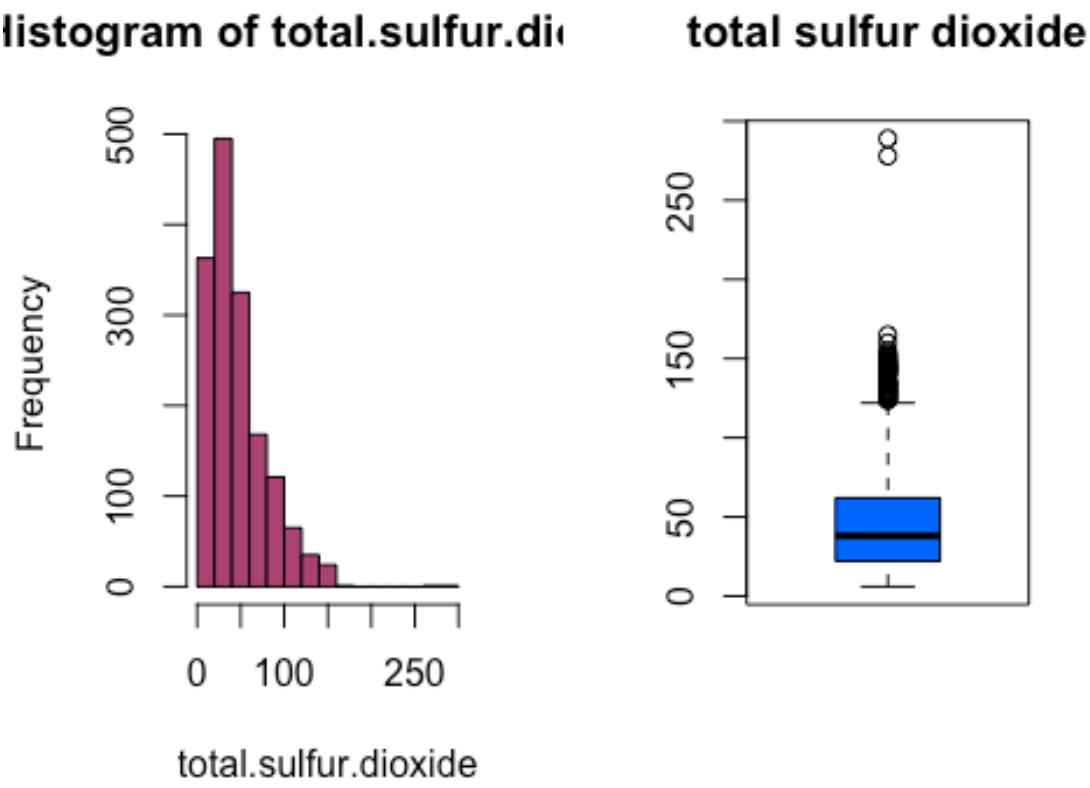
Vamos a considerar outliers los valores por encima de 60. A estas observaciones se les asignará el valor de la mediana.

```
#asignamos el valor NA a los casos que van a ser considerados outliers
data$free.sulfur.dioxide[data$free.sulfur.dioxide>60 ] <- NA
idx <- which(is.na(data$free.sulfur.dioxide))
length(idx) #número de valores perdidos

## [1] 4

#Imputamos el valor de la mediana a los valores NA.
for (i in 1:length(idx)){
index <- idx[i]
data[index,]$free.sulfur.dioxide <- median( data$free.sulfur.dioxide, na.rm=TRUE ) #imputación
}

boxplot(data$free.sulfur.dioxide, main="sulfur dioxide", col=(c("#0066FF")))
```



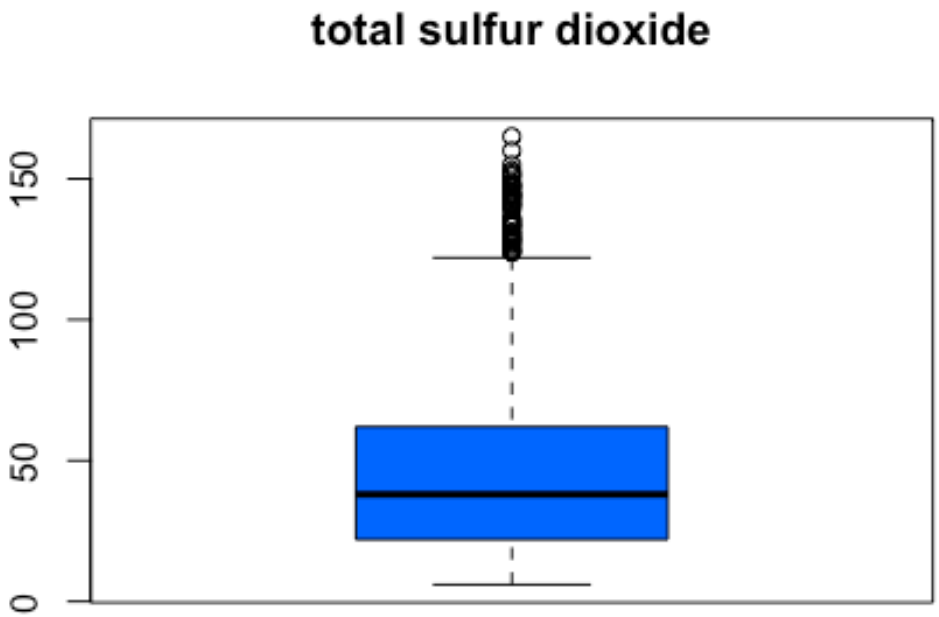
Revisamos que ya no existan valores por encima de 60 en esta variable.

```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(total.sulfur.dioxide, col=(c("#AA4371")))
boxplot(total.sulfur.dioxide, main="total sulfur dioxide", col=(c("#0066FF")))
```



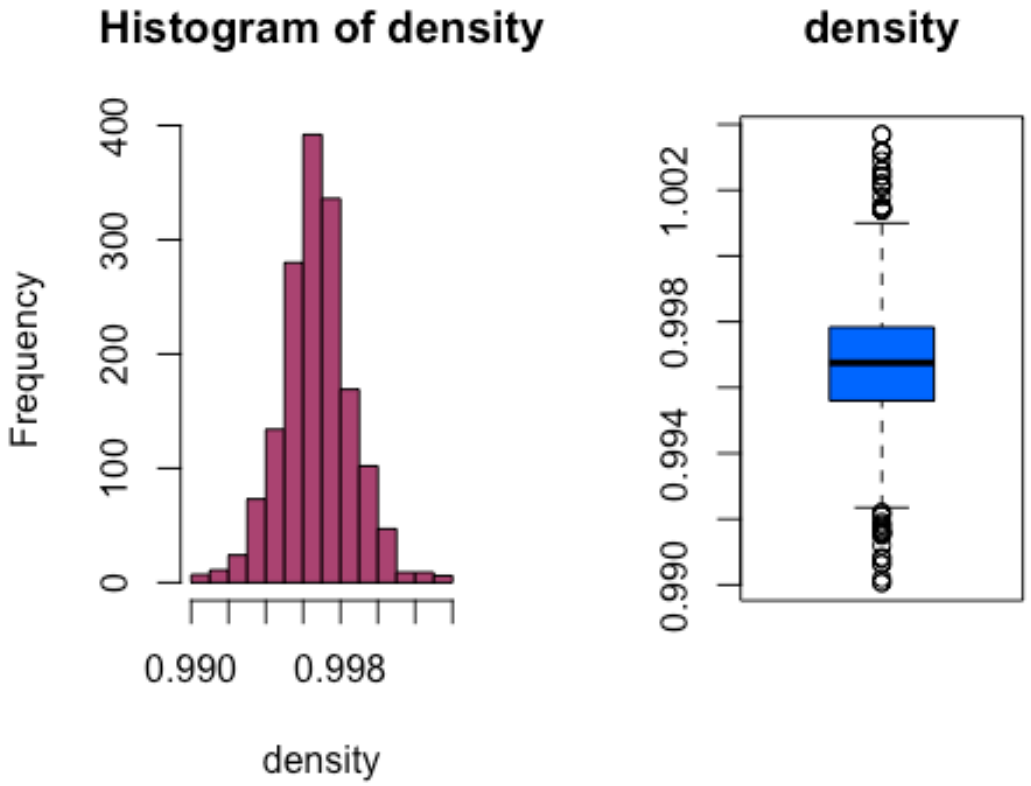
Vamos a considerar outliers los valores por encima de 170. A estas observaciones se les asignará el valor de la mediana.

```
#asignamos el valor NA a los casos que van a ser considerados outliers
data$total.sulfur.dioxide[data$total.sulfur.dioxide>170 ] <- NA
idx <- which(is.na(data$total.sulfur.dioxide))
length(idx) #número de valores perdidos

## [1] 2

#Imputamos el valor de la mediana a los valores NA.
for (i in 1:length(idx)){
index <- idx[i]
data[index,]$total.sulfur.dioxide <- median( data$total.sulfur.dioxide, na.rm=TRUE ) #imputación
}

boxplot(data$total.sulfur.dioxide, main="total sulfur dioxide", col=(c("#0066FF")))
```



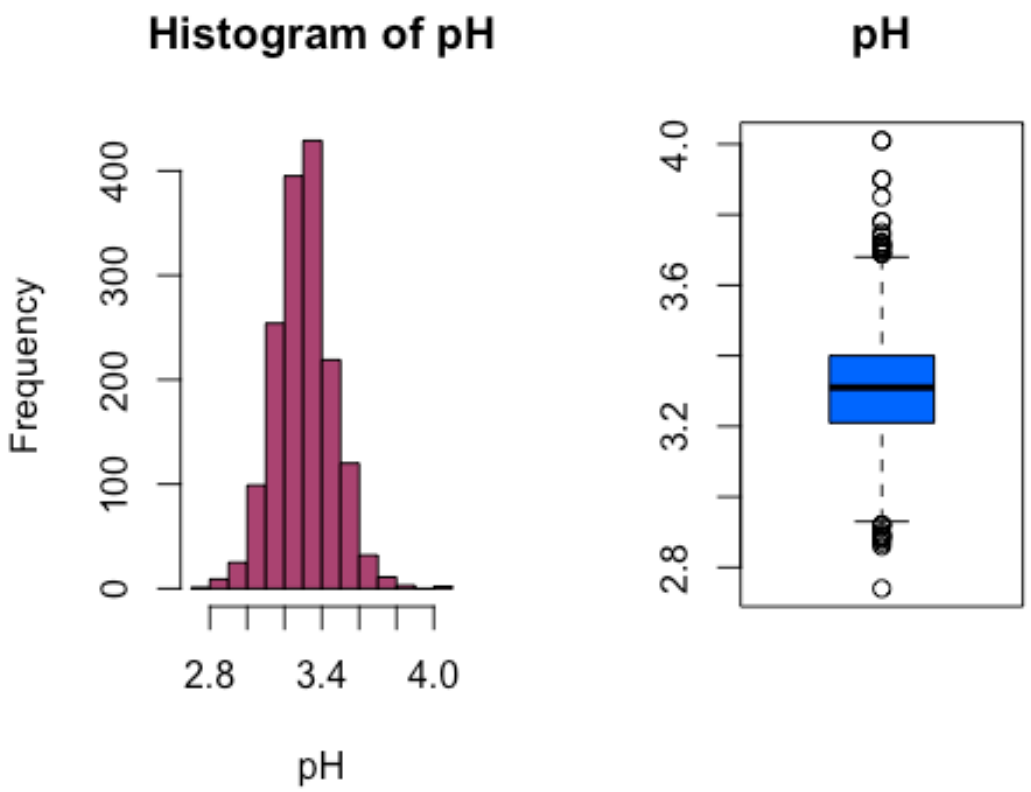
Comprobamos que ya no existen valores por encima de 170.

```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(density, col=(c("#AA4371")))
boxplot(density, main="density", col=(c("#0066FF")))
```



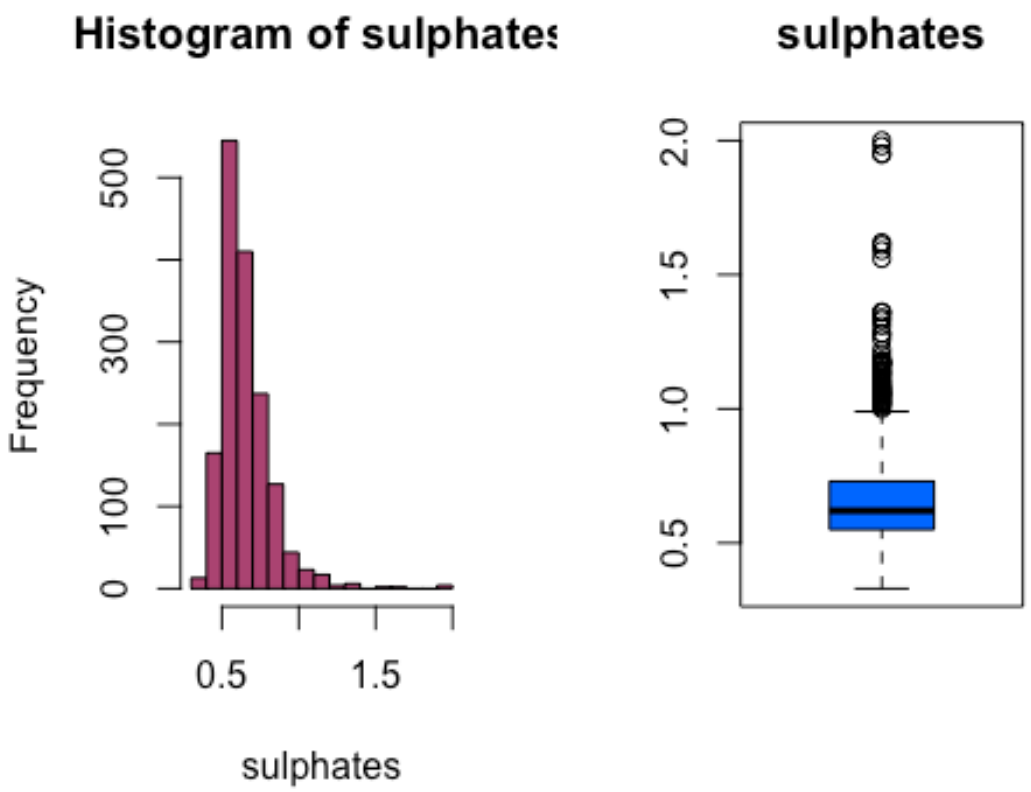
En la Variable “density” se entiende que las observaciones pueden ser valores reales y que están correctamente informadas. De lo anterior no aplicaremos transformaciones para estos datos.

```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(pH, col=(c("#AA4371")))
boxplot(pH, main="pH", col=(c("#0066FF")))
```



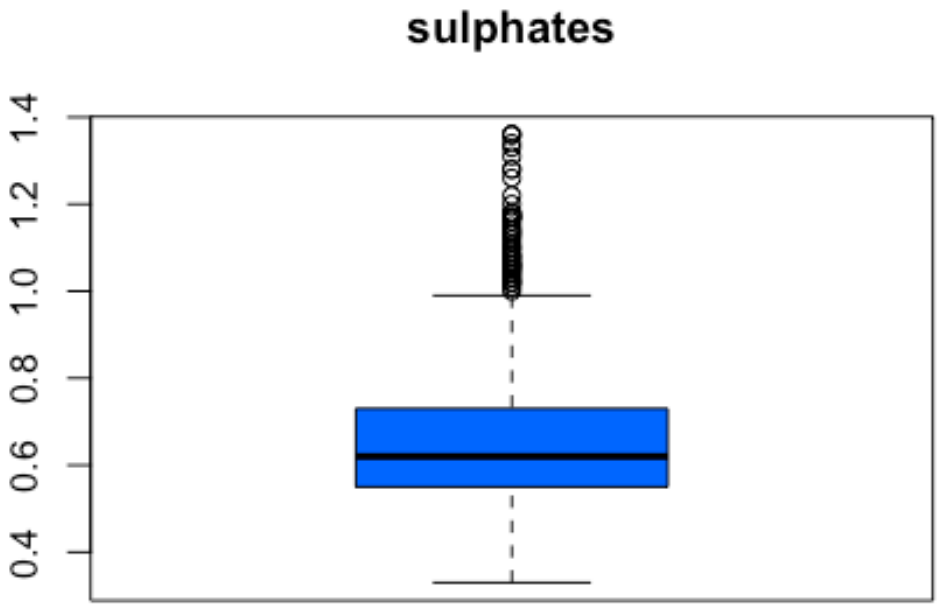
En la Variable “ph” se entiende que las observaciones pueden ser valores reales y que están correctamente informados. De lo anterior no aplicaremos transformaciones para estos datos.

```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(sulphates, col=(c("#AA4371")))
boxplot(sulphates, main="sulphates", col=(c("#0066FF")))
```



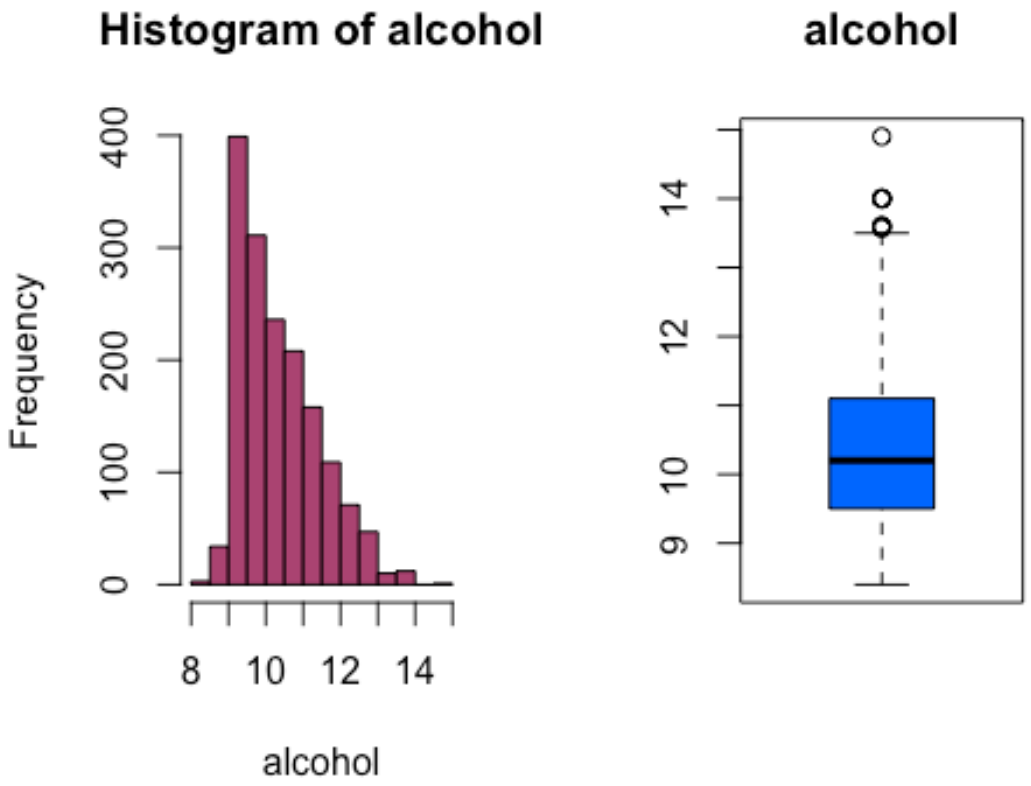
Vamos a considerar outliers los valores por encima de 1,5. A estas observaciones se les asignará el valor de la mediana.

```
#asignamos el valor NA a los casos que van a ser considerados outliers
data$sulphates[data$sulphates>1.5 ] <- NA

idx <- which(is.na(data$sulphates))
length(idx) #número de valores perdidos

## [1] 8

#Imputamos el valor de la mediana a los valores NA.
for (i in 1:length(idx)){
index <- idx[i]
data[index,]$sulphates <- median( data$sulphates, na.rm=TRUE ) #imputación
}

boxplot(data$sulphates, main="sulphates", col=(c("#0066FF")))
```



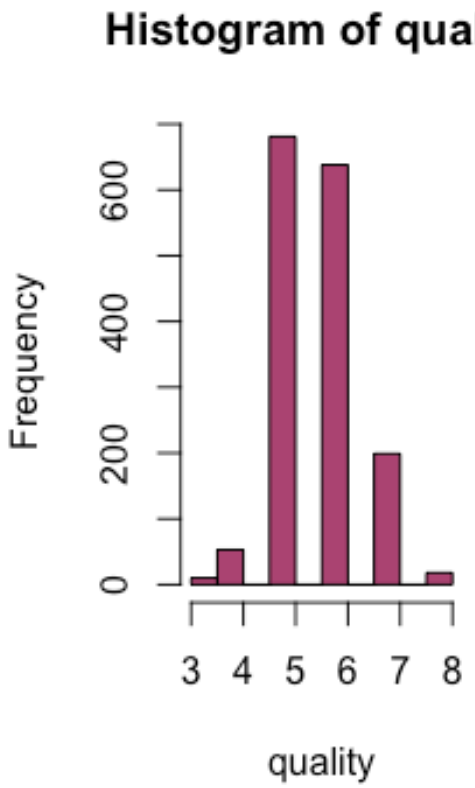
Comprobamos que ya no existen valores por encima de 1,5

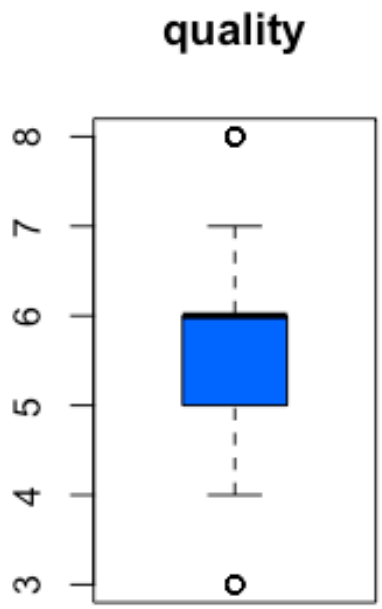
```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(alcohol, col=(c("#AA4371")))
boxplot(alcohol, main="alcohol", col=(c("#0066FF")))
```



En la Variable “alcohol” se entiende que las observaciones pueden ser valores reales y que están correctamente informadas. De lo anterior no aplicaremos transformaciones para estos datos.

```
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
hist(quality, col=(c("#AA4371")))
boxplot(quality, main="quality", col=(c("#0066FF")))
```





###podemos considerarla categoricas

Conclusiones de los criterios adoptados:

residual.sugar: se decide que los valores por encima de 10 van a ser considerados outliers. chlorides: se decide considerar outliers a los valores superiores a 0,3 free.sulfur.dioxide: se decide considerar outliers los valores por encima de 60. total.sulfur.dioxide: se considerar outliers los valores por encima de 170. sulphates: se decide considerar outliers los valores por encima de 1,5.

A todos esos casos se ha decidido imputar el valor de la mediana, excluidos los casos considerados outliers.
En el resto de variables. no se han realizado modificaciones de los datos.

4. Análisis de los datos.

4.1. Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

Análisis de los casos según el nivel de alcohol.

```
vinos.calidadAlta <- data[data$quality >= 7,]
#Se obtienen 217 observaciones
NROW(vinos.calidadAlta)
```

```
## [1] 217
```

```
#En términos porcentuales supone un 13,57%
NROW(vinos.calidadAlta)/NROW(data)
```

```
## [1] 0.1357098
```

Existen 217 muestras cuyos valores de la variable quality es igual o superior a 7, lo que supone un 13,57% sobre el total del dataset.

4.2. Comprobación de la normalidad y homogeneidad de la varianza.

Indicar que por el teorema del límite central, podemos asumir normalidad, puesto que tenemos una muestra de tamaño grande n=1600 observaciones y se desea realizar un test sobre la media.

Analizamos la normalidad la variables con el ad.test de la libreria nortest.

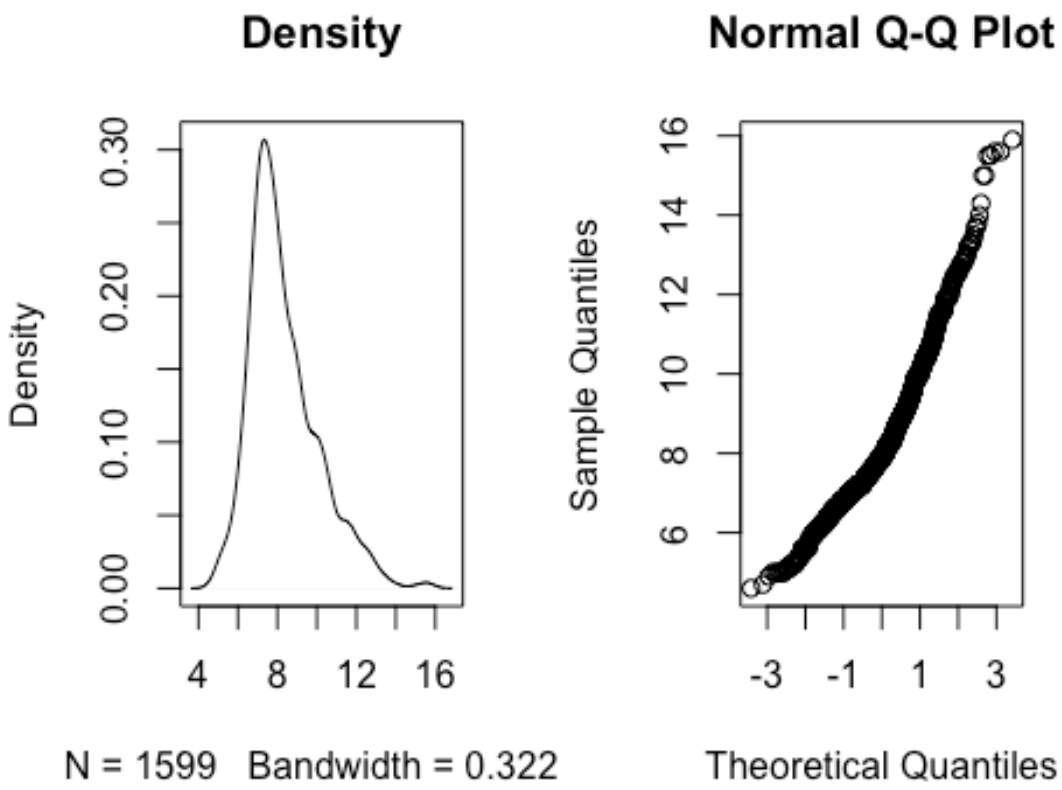
```
#install.packages("nortest")
library(nortest)
alpha = 0.05
col.names = colnames(data)
for (i in 1:ncol(data)) {
if (i == 1) cat("Variables que no siguen una distribución normal:\n")
if (is.integer(data[,i]) | is.numeric(data[,i])) {
p_val = ad.test(data[,i])$p.value
if (p_val < alpha) {
cat(col.names[i])
# Format output
if (i < ncol(data) - 1) cat("\n")
if (i %% 3 == 0) cat("")
}
}
}
```

```
## Variables que no siguen una distribución normal:
## fixed.acidity
## volatile.acidity
## citric.acid
## residual.sugar
## chlorides
## free.sulfur.dioxide
## total.sulfur.dioxide
## density
## pH
```

```
## sulphates
## alcoholquality
```

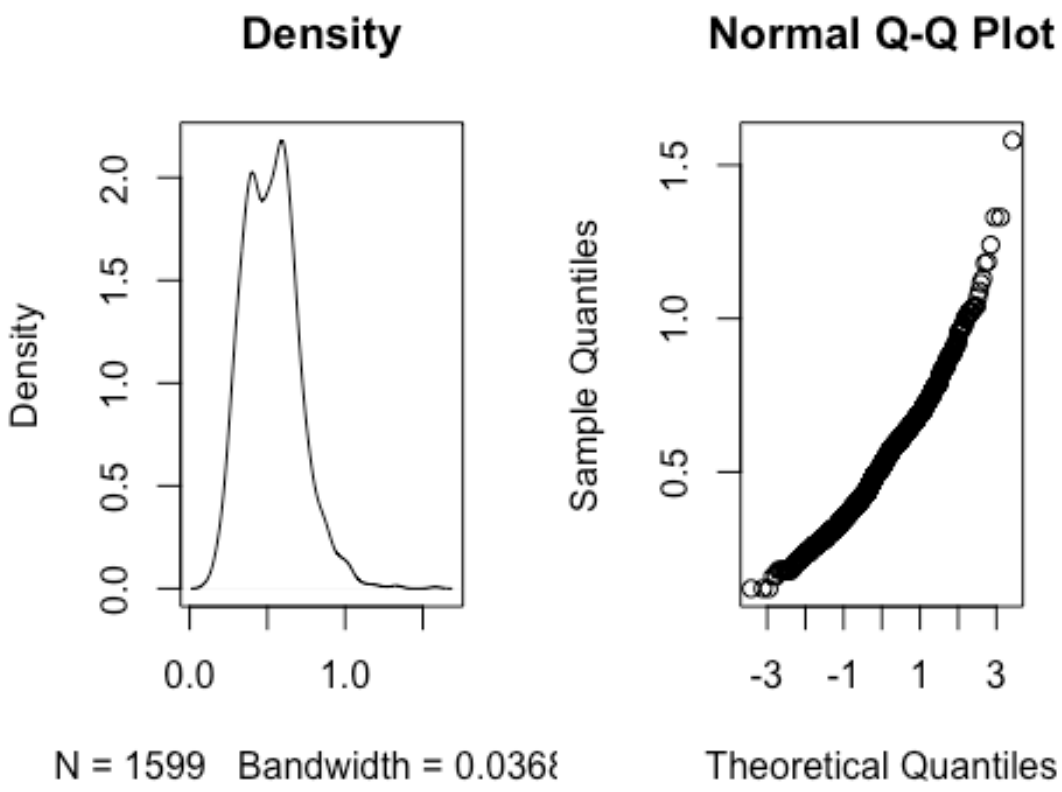
Analizamos la normalidad con el ‘density plot’ y el ‘qqplot’

```
par(mfrow=c(1,2))
plot(density(fixed.acidity),main="Density")
qqnorm(fixed.acidity)
```



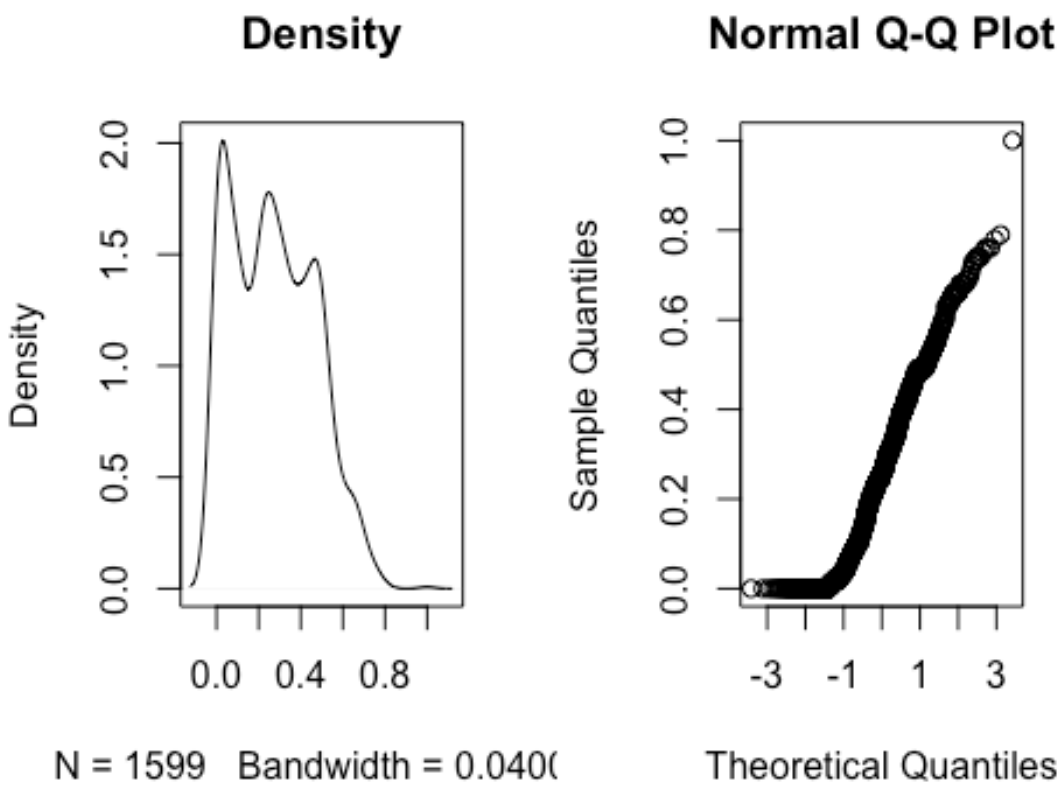
fixed.acidity: parece seguir una distribución normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(volatile.acidity),main="Density")
qqnorm(volatile.acidity)
```



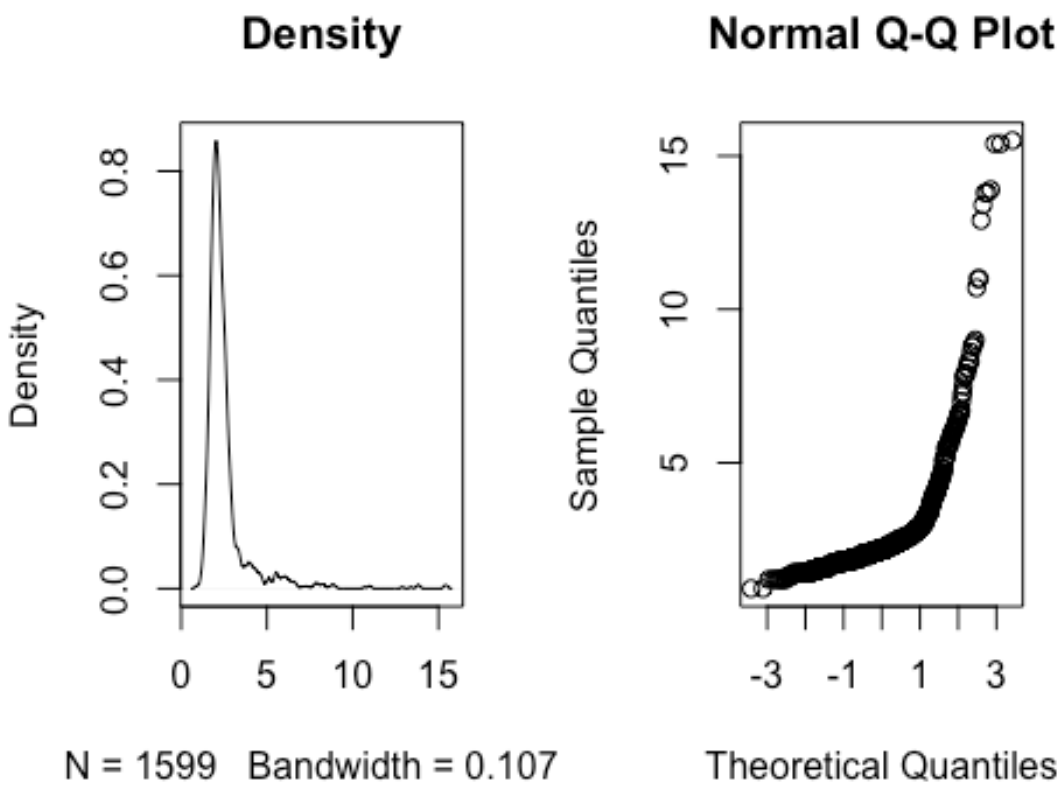
volatile.acidity: parece seguir una distribución normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(citric.acid),main="Density")
qqnorm(citric.acid)
```



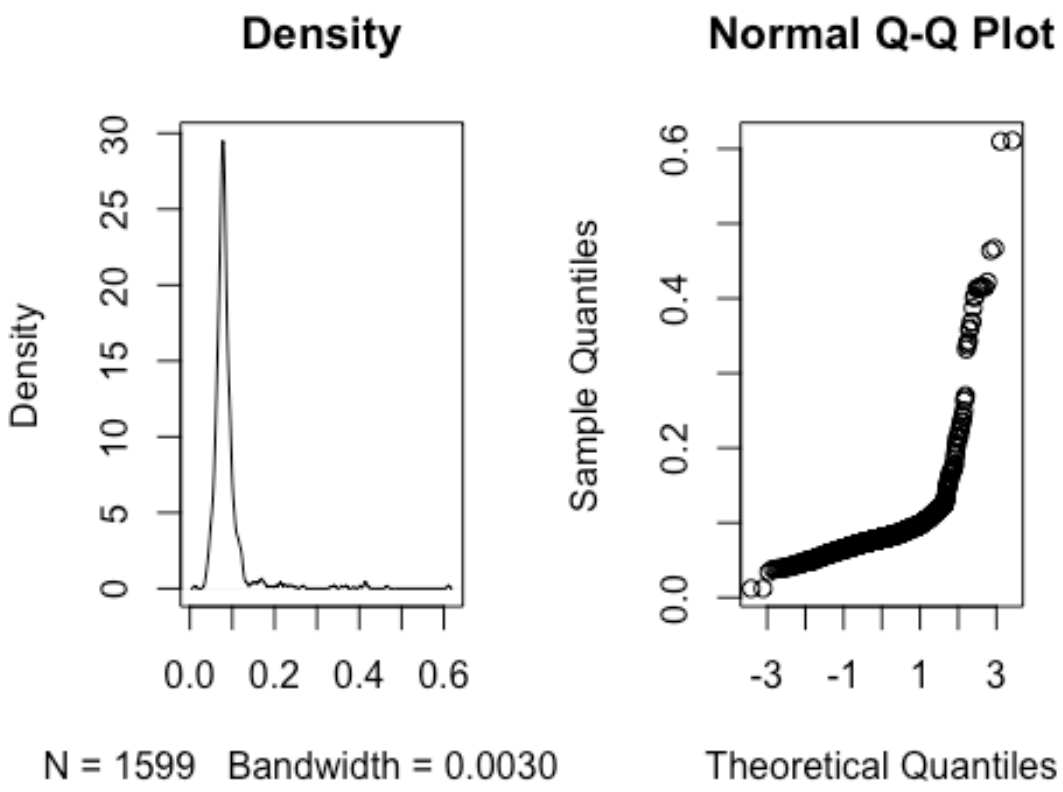
citric.acid: su extremo izquierdo se desvía de la normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(residual.sugar),main="Density")
qqnorm(residual.sugar)
```



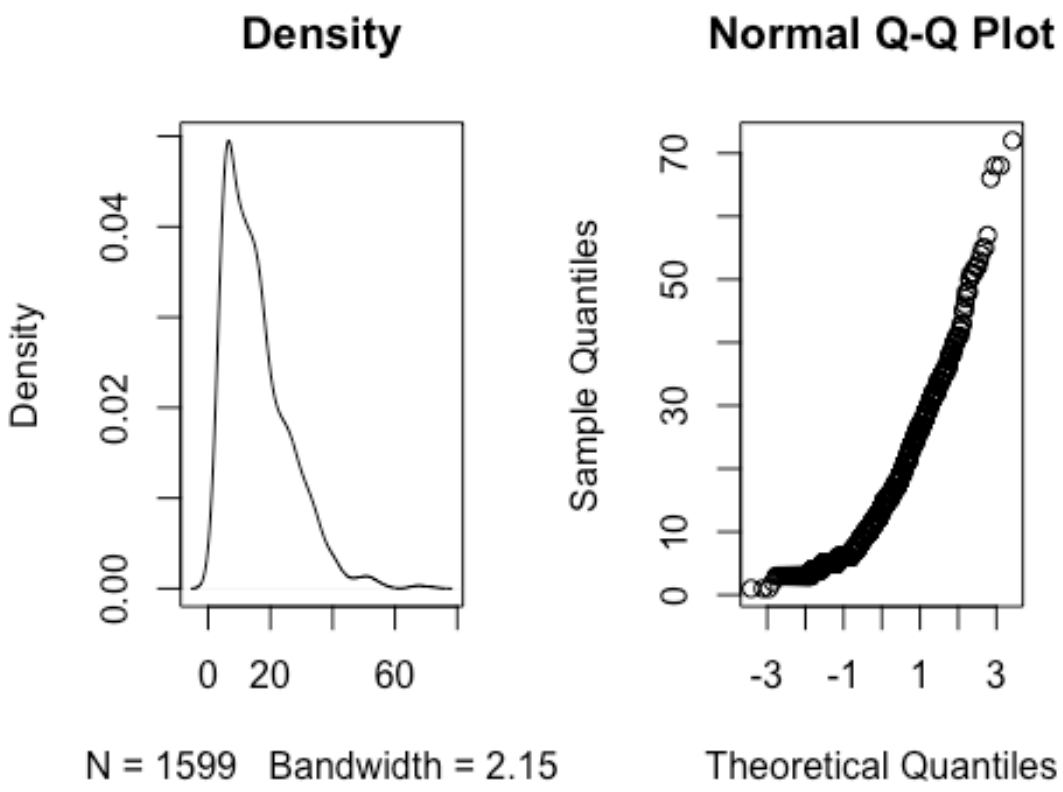
residual.sugar: los valores centrales se desvían de la normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(chlorides),main="Density")
qqnorm(chlorides)
```



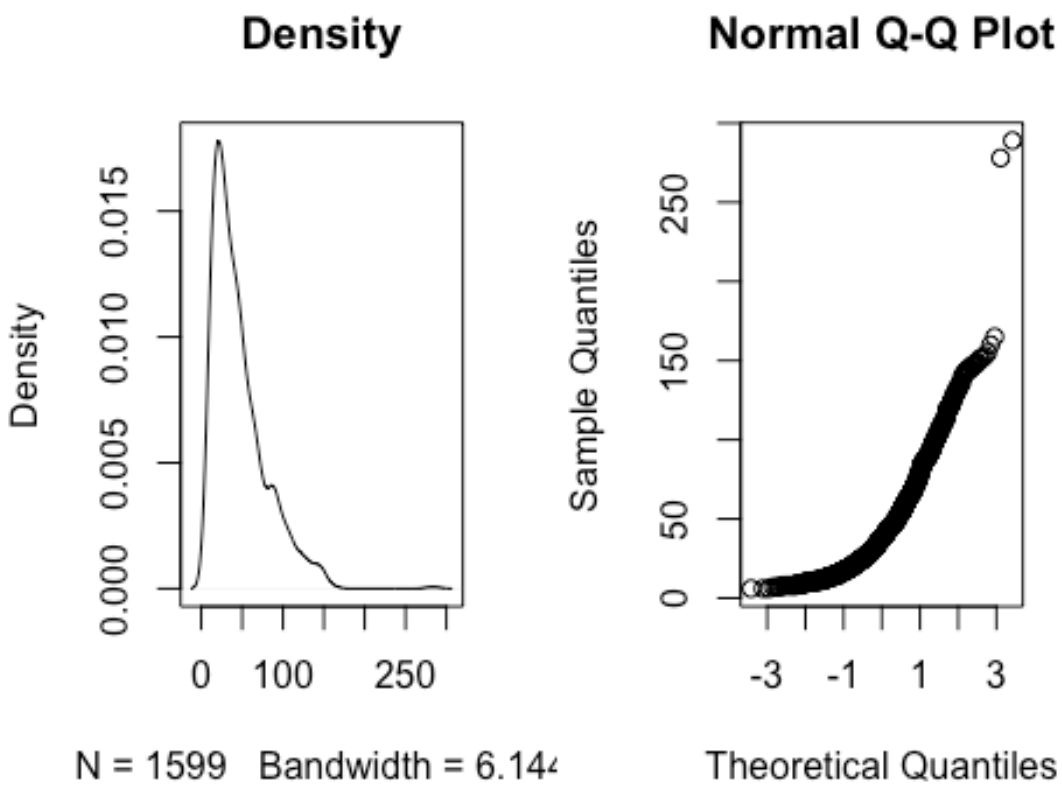
chlorides: los valores centrales se desvían de la normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(free.sulfur.dioxide),main="Density")
qqnorm(free.sulfur.dioxide)
```



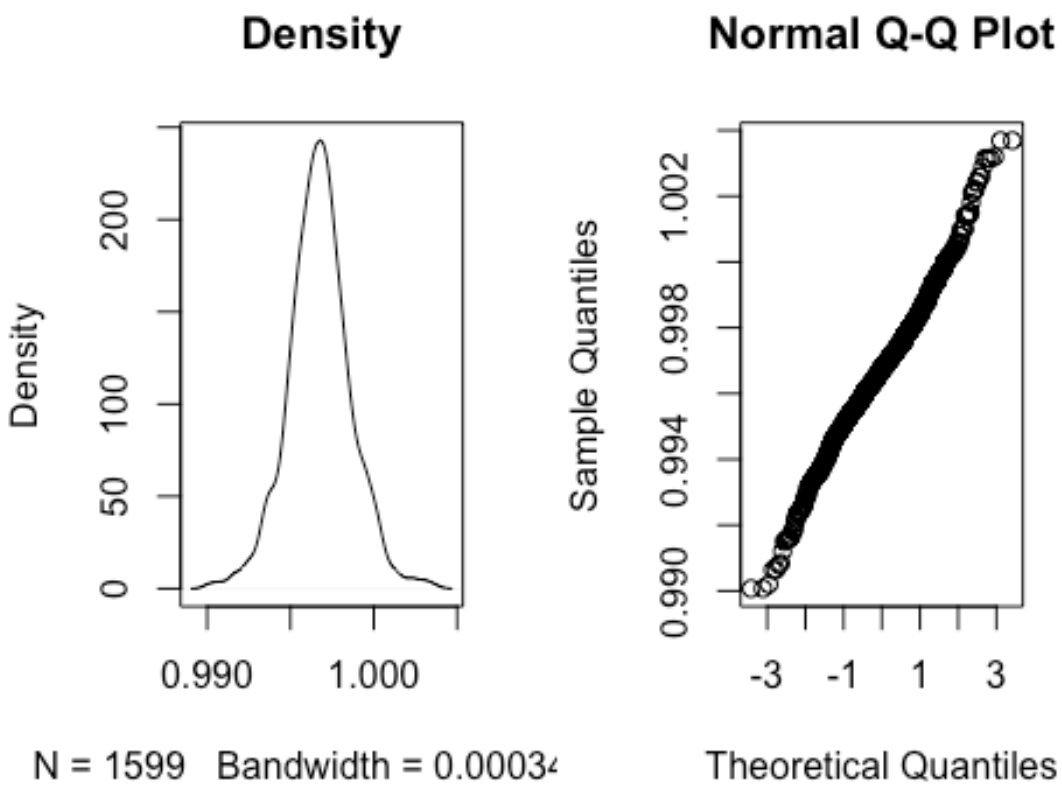
free.sulfur.dioxide: parece seguir una distribución normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(total.sulfur.dioxide),main="Density")
qqnorm(total.sulfur.dioxide)
```



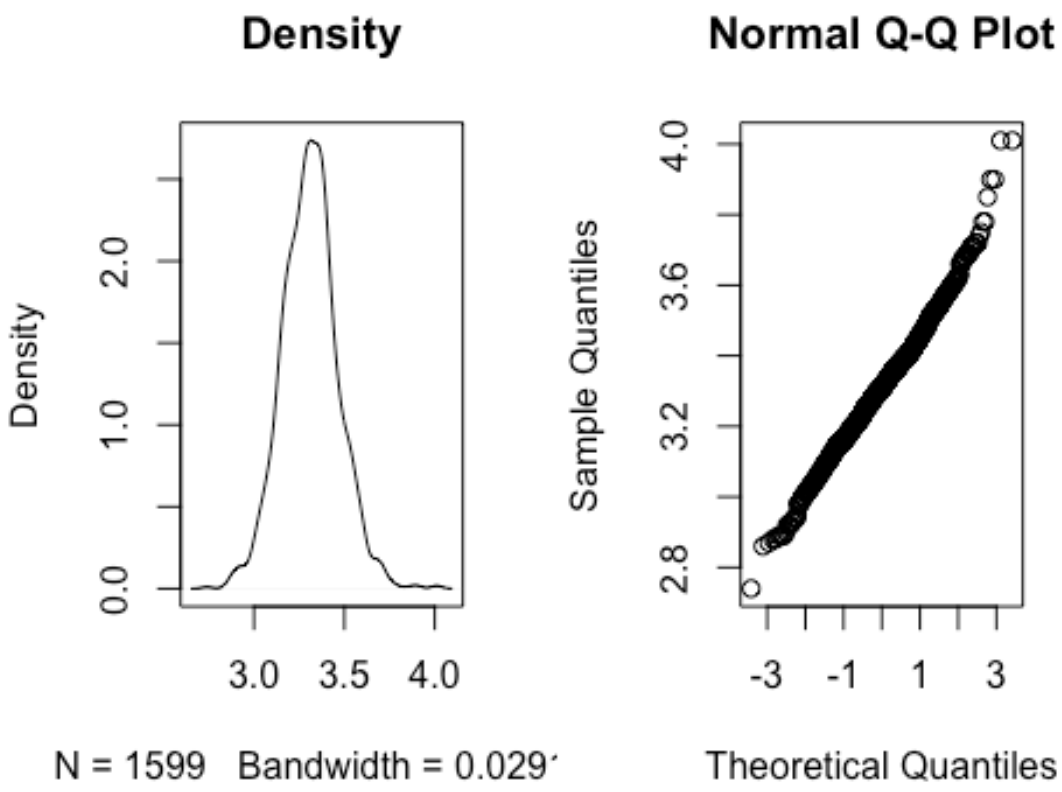
total.sulfur.dioxide: parece seguir una distribución normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(density),main="Density")
qqnorm(density)
```



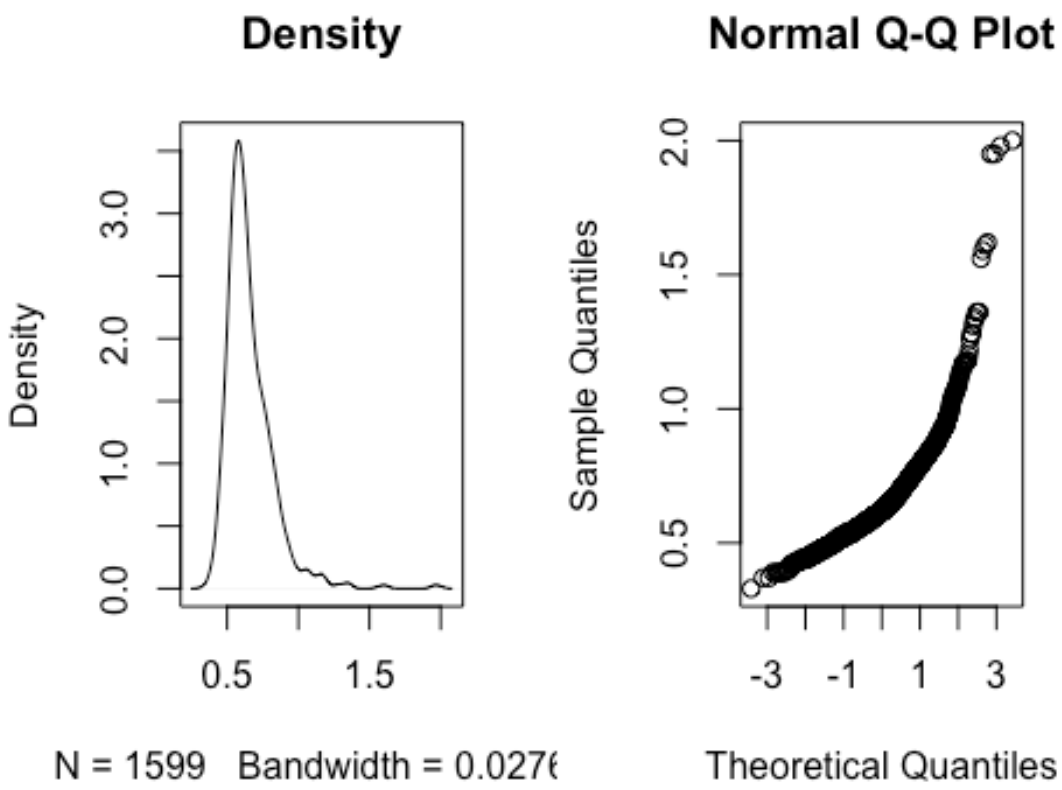
Density: parece seguir una distribución normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(pH),main="Density")
qqnorm(pH)
```



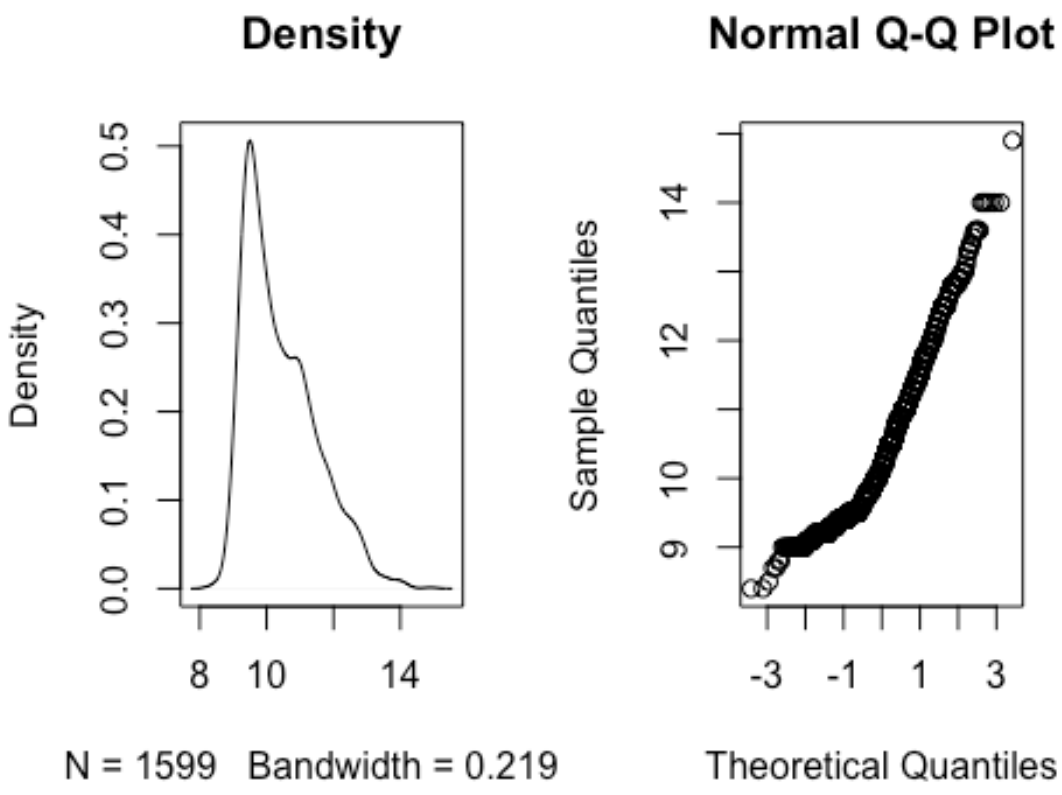
density(pH): parece seguir una distribución normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(sulphates),main="Density")
qqnorm(sulphates)
```



sulphates: parece seguir una distribución normal observando el grafico : q-q

```
par(mfrow=c(1,2))
plot(density(alcohol),main="Density")
qqnorm(alcohol)
```



Seguidamente, pasamos a estudiar la homogeneidad de varianzas mediante la aplicación de un test de Fligner-Killeen. En este caso, estudiaremos esta homogeneidad. En el siguiente test, la hipótesis nula consiste en que ambas varianzas son iguales.

```
fligner.test(quality ~ residual.sugar, data = data)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  quality by residual.sugar
## Fligner-Killeen:med chi-squared = 75.977, df = 82, p-value = 0.6664
```

Puesto que obtenemos un p-valor superior a 0,05, aceptamos la hipótesis de que las varianzas de ambas muestras son homogéneas.

4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

```
summary(alcohol)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.40    9.50   10.20   10.42   11.10   14.90
```

Contraste de hipótesis:

Se aplica la prueba estadística de contraste de hipótesis sobre dos muestras para determinar si el nivel de calidad el vino es diferente dependiendo de la categoría calculada en función del alcohol. Se considera nivel bajo de alcohol las muestras con un valor por debajo de la media (10.42) y nivel alto cuando la muestra tiene un valor superior o igual a 10.42.

Para ello, tendremos dos categorías: - Vinos con nivel bajo de alcohol - Vinos con nivel alto de alcohol

```
data_AlcoholBajo <- data[data$alcohol < 10.42,]$quality
data_AlcoholAlto <- data[data$alcohol >= 10.42,]$quality
```

Nos preguntamos si la calidad del los vinos puede ser diferente en función de la categoría creada en función del grado de alcohol de la muestras. (anteriormente descrita)

En este escenario la hipótesis nula y la alternativa.

H0 : µAlcBajo = µAlcAlto

H1 : µAlcBajo =! µAlcAlto

Es un test de dos muestras sobre la media con varianzas desconocidas. Por el teorema del límite central, podemos asumir normalidad. Comprobamos igualdad de varianzas:

```
var.test( data_AlcoholBajo, data_AlcoholAlto )

##
##  F test to compare two variances
##
## data:  data_AlcoholBajo and data_AlcoholAlto
## F = 0.59219, num df = 915, denom df = 682, p-value = 1.624e-13
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.5142841 0.6809068
## sample estimates:
## ratio of variances
##          0.5921906
```

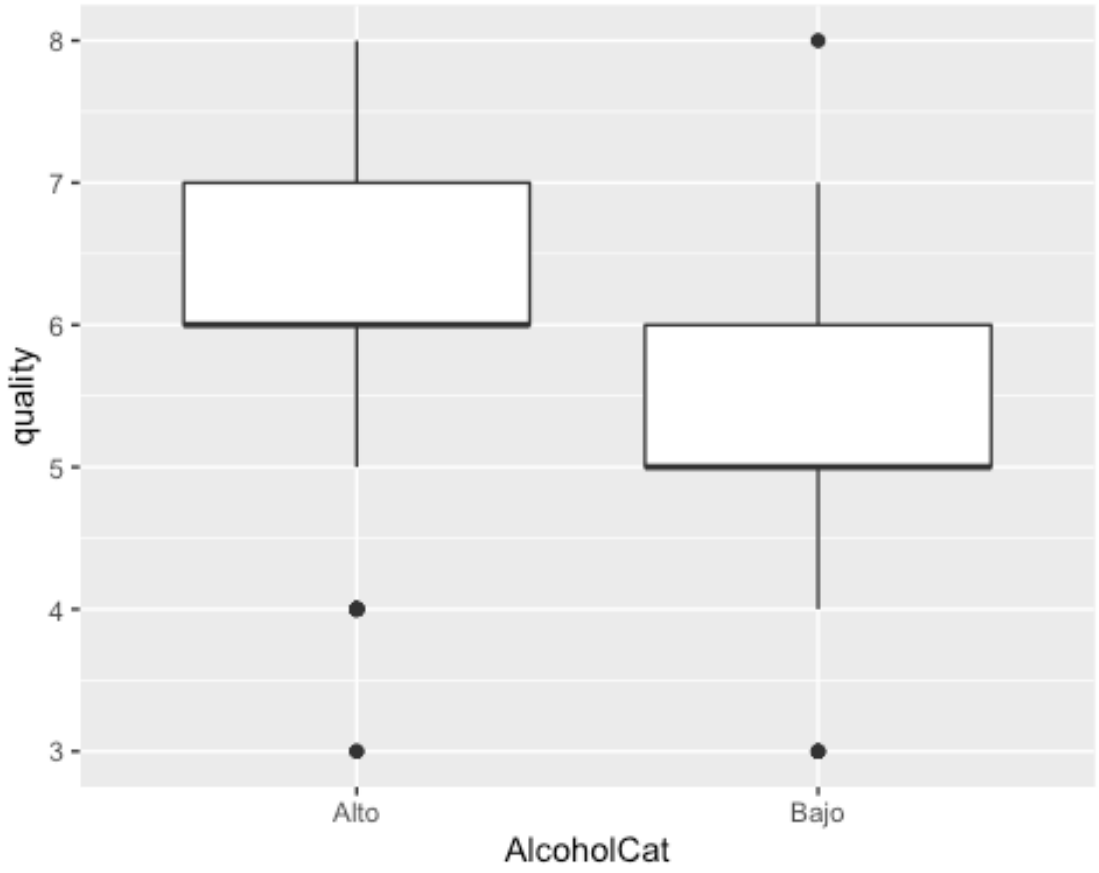
El resultado del test es un valor p<0.05. Por tanto, asumimos diferencias de varianzas. En consecuencia, el test se corresponde con un test de dos muestras independientes sobre la media con varianzas desconocidas diferentes. El test es bilateral.

```
t.test( data_AlcoholBajo, data_AlcoholAlto,
var.equal=FALSE, alternative="two.sided")

##
##  Welch Two Sample t-test
##
## data:  data_AlcoholBajo and data_AlcoholAlto
```

```
## t = -17.688, df = 1237.4, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.7569536 -0.6057987
## sample estimates:
## mean of x mean of y
##  5.344978  6.026354
```

Conclusión: Nos encontramos con un p-value por debajo de 0.05 por lo que podemos indicar que podemos aceptar la hipótesis alternativa. Podemos indicar que la calidad de los vinos es diferente entre los considerados como nivel bajo de alcohol y los vinos con un nivel alto de alcohol con un nivel de confianza del 95%.

```
#install.packages("ggplot2")

library("ggplot2")
data$AlcoholCat <- ifelse(data$alcohol < 10.42 , "Bajo", "Alto")
 #diagramas de caja:  distribucion de la variable 'Weight' según la variable 'portero'
  ggplot(data =data, aes(x=AlcoholCat,y=quality)) + geom_boxplot()
```



Representación gráfica del nivel de calidad frente a las categorías creadas en función de las variables de alcohol consideradas.

Análisis de la correlación.

```
corr_matrix <- matrix(nc = 2, nr = 0)
colnames(corr_matrix) <- c("estimate", "p-value")

# Calcular el coeficiente de correlación para cada variable cuantitativa
# con respecto al campo "quality"
for (i in 1:(ncol(data) - 1)) {
if (is.integer(data[,i]) | is.numeric(data[,i])) {
spearman_test = cor.test(data[,i],data$quality,method = "spearman")
corr_coef = spearman_test$estimate
p_val = spearman_test$p.value
# Add row to matrix
pair = matrix(ncol = 2, nrow = 1)
pair[1][1] = corr_coef
pair[2][1] = p_val
corr_matrix <- rbind(corr_matrix, pair)
rownames(corr_matrix)[nrow(corr_matrix)] <- colnames(data)[i]
}
}

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties
```

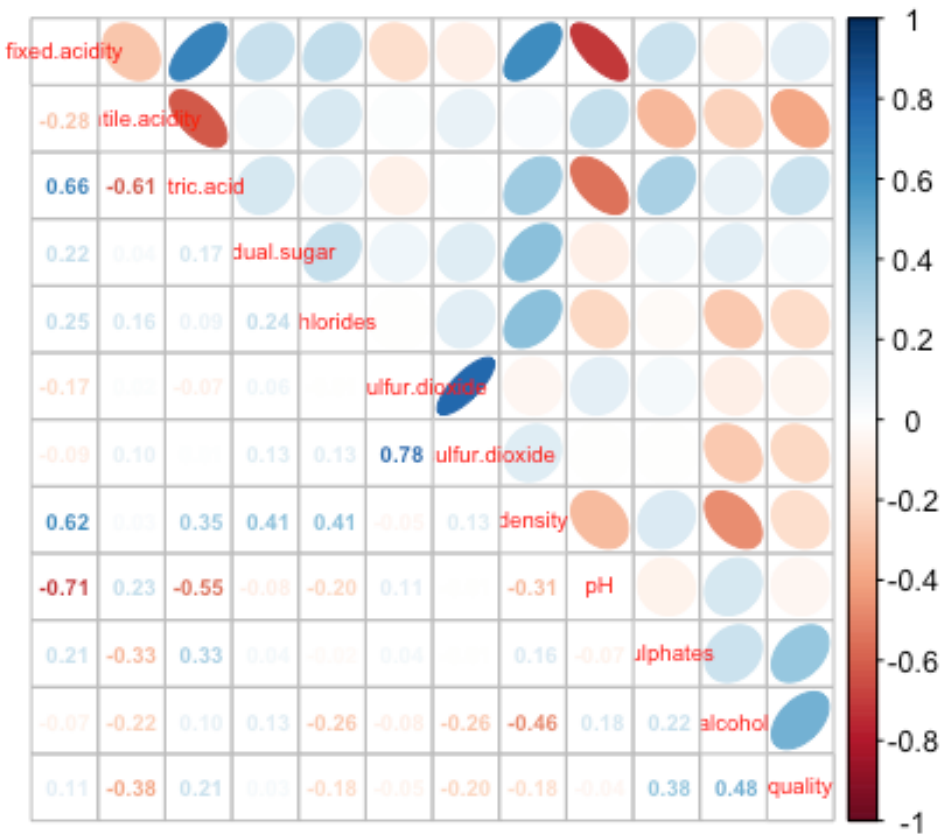
```
## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

## Warning in cor.test.default(data[, i], data$quality, method = "spearman"):
## Cannot compute exact p-value with ties

corr_matrix

##                       estimate      p-value
## fixed.acidity        0.11408367 4.801220e-06
## volatile.acidity    -0.38064651 2.734944e-56
## citric.acid          0.21348091 6.158952e-18
## residual.sugar       0.03270283 1.912021e-01
## chlorides           -0.18332346 1.494955e-13
## free.sulfur.dioxide  -0.05485346 2.827937e-02
## total.sulfur.dioxide -0.20048430 5.826244e-16
## density             -0.17707407 9.918139e-13
## pH                  -0.04367193 8.084594e-02
## sulphates            0.38395957 2.537306e-57
## alcohol              0.47853169 2.726838e-92
## quality              1.00000000 0.000000e+00
```

Así, identificamos cuáles son las variables más correlacionadas con la calidad del vino en función de su proximidad con los valores -1 y +1. Teniendo esto en cuenta, queda patente cómo la variable más relevante en la calidad del vino es el alcohol . Nota. Para cada coeficiente de correlación se muestra también su p-valor asociado, puesto que éste puede dar información acerca del peso estadístico de la correlación obtenida.

Correlación con la variable Quality:

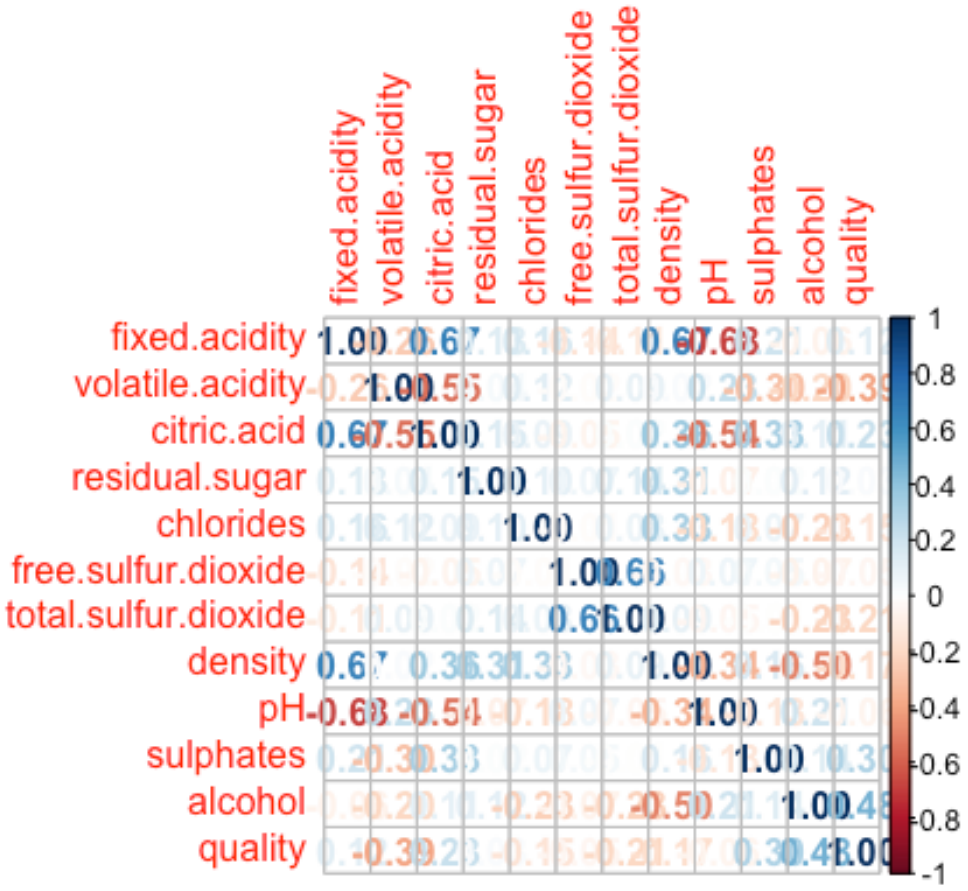
```
corr.res<-cor(data[,-13], method="spearman")
corr.res

##                       fixed.acidity volatile.acidity  citric.acid residual.sugar
## fixed.acidity          1.00000000     -0.27828222  0.661708417     0.22061984
## volatile.acidity      -0.27828222      1.00000000 -0.610259467     0.03838806
## citric.acid            0.66170842     -0.61025947  1.000000000     0.17281554
## residual.sugar         0.22061984      0.03838806  0.172815538     1.00000000
## chlorides              0.24948756      0.16292441  0.085052679     0.23681569
## free.sulfur.dioxide   -0.17026090      0.01908994 -0.074407514     0.06093500
## total.sulfur.dioxide  -0.08851961      0.09732612  0.005880251     0.13128825
## density                0.62307076      0.02501412  0.352285261     0.41263833
## pH                    -0.70667359      0.23357152 -0.548026276    -0.08406679
## sulphates              0.21263218     -0.32845414  0.329490517     0.04180563
## alcohol               -0.06657566     -0.22493168  0.096455544     0.12950596
## quality                0.11408367     -0.38064651  0.213480914     0.03270283
```

```
##                       chlorides free.sulfur.dioxide total.sulfur.dioxide
## fixed.acidity        0.249487559       -0.170260896        -0.088519608
## volatile.acidity     0.162924412        0.019089943         0.097326120
## citric.acid          0.085052679       -0.074407514         0.005880251
## residual.sugar       0.236815693        0.060935002         0.131288245
## chlorides            1.000000000       -0.005331721         0.128941252
## free.sulfur.dioxide -0.005331721        1.000000000         0.783791037
## total.sulfur.dioxide 0.128941252        0.783791037         1.000000000
## density              0.414069187       -0.045902592         0.133054752
## pH                  -0.204374649        0.114175693        -0.006302514
## sulphates            -0.022873613        0.044009451        -0.008090304
## alcohol             -0.261929083       -0.080067784        -0.261431180
## quality             -0.183323461       -0.054853464        -0.200484301
##                        density         pH    sulphates     alcohol
## fixed.acidity        0.62307076 -0.706673595  0.212632177 -0.06657566
## volatile.acidity     0.02501412  0.233571519 -0.328454141 -0.22493168
## citric.acid          0.35228526 -0.548026276  0.329490517  0.09645554
## residual.sugar       0.41263833 -0.084066795  0.041805632  0.12950596
## chlorides            0.41406919 -0.204374649 -0.022873613 -0.26192908
## free.sulfur.dioxide -0.04590259  0.114175693  0.044009451 -0.08006778
## total.sulfur.dioxide 0.13305475 -0.006302514 -0.008090304 -0.26143118
## density              1.00000000 -0.312055078  0.159942063 -0.46244458
## pH                  -0.31205508  1.000000000 -0.067479602  0.17993243
## sulphates            0.15994206 -0.067479602  1.000000000  0.21641153
## alcohol             -0.46244458  0.179932427  0.216411535  1.00000000
## quality             -0.17707407 -0.043671935  0.383959571  0.47853169
##                       quality
## fixed.acidity        0.11408367
## volatile.acidity    -0.38064651
## citric.acid          0.21348091
## residual.sugar       0.03270283
## chlorides           -0.18332346
## free.sulfur.dioxide -0.05485346
## total.sulfur.dioxide -0.20048430
## density             -0.17707407
## pH                  -0.04367193
## sulphates            0.38395957
## alcohol              0.47853169
## quality              1.00000000

#install.packages("corrplot")
#library(corrplot)
corrplot::corrplot.mixed(corr.res, upper="ellipse",number.cex=.6,tl.cex=.6)
```



Se analiza adicionalmente la correlación de forma gráfica.

```
library(corrplot)

## corrplot 0.88 loaded

par(mfrow = c(1,1))
cor.data <- cor(data[,-13])
corrplot(cor.data, method = 'number')
```



Tanto en la gráfica de elipses como en la numérica, se pueden apreciar relaciones débiles entre residual sugar, free sulfur dioxide, ph, fixed acidity y density con respecto a la variable que tratamos de explicar, quality.

Modelo de regresión lineal:

Incialmente realizamos un modelo lineal con las 4 variables mas correlacionadas con la variable "quality"

```
modelo1 <- lm(quality ~ alcohol +volatile.acidity + sulphates + citric.acid )
summary(modelo1)

##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
##     citric.acid)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.71408 -0.38590 -0.06402  0.46657  2.20393
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)      2.64592    0.20106  13.160  < 2e-16 ***
## alcohol          0.30908    0.01581  19.553  < 2e-16 ***
## volatile.acidity -1.26506    0.11266 -11.229  < 2e-16 ***
## sulphates        0.69552    0.10311   6.746 2.12e-11 ***
## citric.acid     -0.07913    0.10381  -0.762    0.446
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6588 on 1594 degrees of freedom
## Multiple R-squared:  0.3361, Adjusted R-squared:  0.3345
## F-statistic: 201.8 on 4 and 1594 DF,  p-value: < 2.2e-16
```

```
# Intervalos de confianza
confint(modelo1)
```

```
##                   2.5 %     97.5 %
## (Intercept)      2.2515575  3.0402782
## alcohol          0.2780728  0.3400835
## volatile.acidity -1.4860436 -1.0440733
## sulphates        0.4932780  0.8977542
## citric.acid     -0.2827462  0.1244961
```

```
# En caso de querer los intervalos basados en el error estándar.
confint.default(modelo1)
```

```
##                   2.5 %     97.5 %
## (Intercept)      2.2518569  3.0399788
## alcohol          0.2780963  0.3400600
## volatile.acidity -1.4858758 -1.0442411
## sulphates        0.4934316  0.8976006
## citric.acid     -0.2825916  0.1243415
```

Posteriormente realizamos un modelo lineal con las 5 variables más correlacionadas con la variable “quality” (simplemente añadimos total sulfur dioxide al modelo anterior)

```
modelo2 <- lm(quality ~ alcohol +volatile.acidity + sulphates + citric.acid + total.sulfur.dioxide)
summary(modelo2)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
##     citric.acid + total.sulfur.dioxide)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.72463 -0.38380 -0.06689  0.44606  2.14550
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)          2.8431068  0.2050732  13.864  < 2e-16 ***
## alcohol              0.2953419  0.0160375  18.416  < 2e-16 ***
## volatile.acidity    -1.2223102  0.1124774 -10.867  < 2e-16 ***
## sulphates            0.7207881  0.1027039   7.018 3.32e-12 ***
## citric.acid         -0.0427246  0.1035810  -0.412     0.68
## total.sulfur.dioxide -0.0022182  0.0005126  -4.327 1.60e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6552 on 1593 degrees of freedom
## Multiple R-squared:  0.3439, Adjusted R-squared:  0.3418
## F-statistic:   167 on 5 and 1593 DF,  p-value: < 2.2e-16
```

```
# Intervalos de confianza
confint(modelo2)
```

```
##                       2.5 %      97.5 %
## (Intercept)          2.440865047  3.245348455
## alcohol              0.263885167  0.326798655
## volatile.acidity    -1.442929393 -1.001690996
## sulphates            0.519339183  0.922237030
## citric.acid         -0.245894093  0.160444848
## total.sulfur.dioxide -0.003223673 -0.001212781
```

```
# En caso de querer los intervalos basados en el error estándar.
confint.default(modelo2)
```

```
##                       2.5 %      97.5 %
## (Intercept)          2.441170667  3.245042835
## alcohol              0.263909067  0.326774754
## volatile.acidity    -1.442761768 -1.001858621
## sulphates            0.519492243  0.922083970
## citric.acid         -0.245739727  0.160290481
## total.sulfur.dioxide -0.003222909 -0.001213545
```

Finalmente realizamos un modelo lineal con las 7 variables mas correlacionadas con la variable “quality” (simplemente añadimos total density al modelo anterior)

```
modelo3 <- lm(quality ~ alcohol +volatile.acidity + sulphates + citric.acid + chlorides +total.sulfur.dioxide
+density)
summary(modelo3)
```

```
##
## Call:
## lm(formula = quality ~ alcohol + volatile.acidity + sulphates +
##     citric.acid + chlorides + total.sulfur.dioxide + density)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -2.67819 -0.38067 -0.06311  0.44428  2.05461
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.9529094 11.9896065  -0.079    0.937
## alcohol              0.2802878  0.0202744  13.825  < 2e-16 ***
## volatile.acidity    -1.1144889  0.1197464  -9.307  < 2e-16 ***
## sulphates            0.9025477  0.1122700   8.039 1.75e-15 ***
## citric.acid          0.0444070  0.1236454   0.359    0.720
## chlorides           -1.7474785  0.4056573  -4.308 1.75e-05 ***
## total.sulfur.dioxide -0.0023195  0.0005138  -4.514 6.82e-06 ***
## density              3.9230934 11.9441494   0.328    0.743
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6517 on 1591 degrees of freedom
## Multiple R-squared:  0.3517, Adjusted R-squared:  0.3488
## F-statistic: 123.3 on 7 and 1591 DF,  p-value: < 2.2e-16
```

Al revisar R-squared, se observa cómo al ir incluyendo más variables se mejora, aunque no mucho, la precisión del modelo.

```
# Intervalos de confianza
confint(modelo3)
```

```
##                          2.5 %      97.5 %
## (Intercept)         -24.469996853 22.564178123
## alcohol               0.240520525  0.320055120
## volatile.acidity     -1.349366182 -0.879611611
## sulphates             0.682334966  1.122760383
## citric.acid          -0.198118011  0.286932040
## chlorides            -2.543157645 -0.951799434
## total.sulfur.dioxide  -0.003327297 -0.001311684
## density             -19.504831974 27.351018799
```

```
# En caso de querer los intervalos basados en el error estándar.
confint.default(modelo3)
```

```
##                          2.5 %      97.5 %
## (Intercept)         -24.452106293 22.546287563
## alcohol               0.240550778  0.320024867
## volatile.acidity     -1.349187499 -0.879790293
## sulphates             0.682502492  1.122592856
## citric.acid          -0.197933511  0.286747540
## chlorides            -2.542552334 -0.952404745
## total.sulfur.dioxide  -0.003326531 -0.001312451
## density             -19.487009244 27.333196069
```

Regresion logistica:

Creamos una variable dicotómica para aplicar la regresión logística. CalidadAlta será igual a 1 cuando los valores de la variable quality sean superiores o iguales a 6

```
data$CalidadAlta <- ifelse( data$quality >=6 , 1, 0)
#revisamos los valores
table(data$quality, data$CalidadAlta)

##
##      0   1
##   3  10   0
##   4  53   0
##   5 681   0
##   6   0 638
##   7   0 199
##   8   0  18
```

Revisamos los valores introducidos en la nueva variable y su relación con el contenido en la variable “quality”

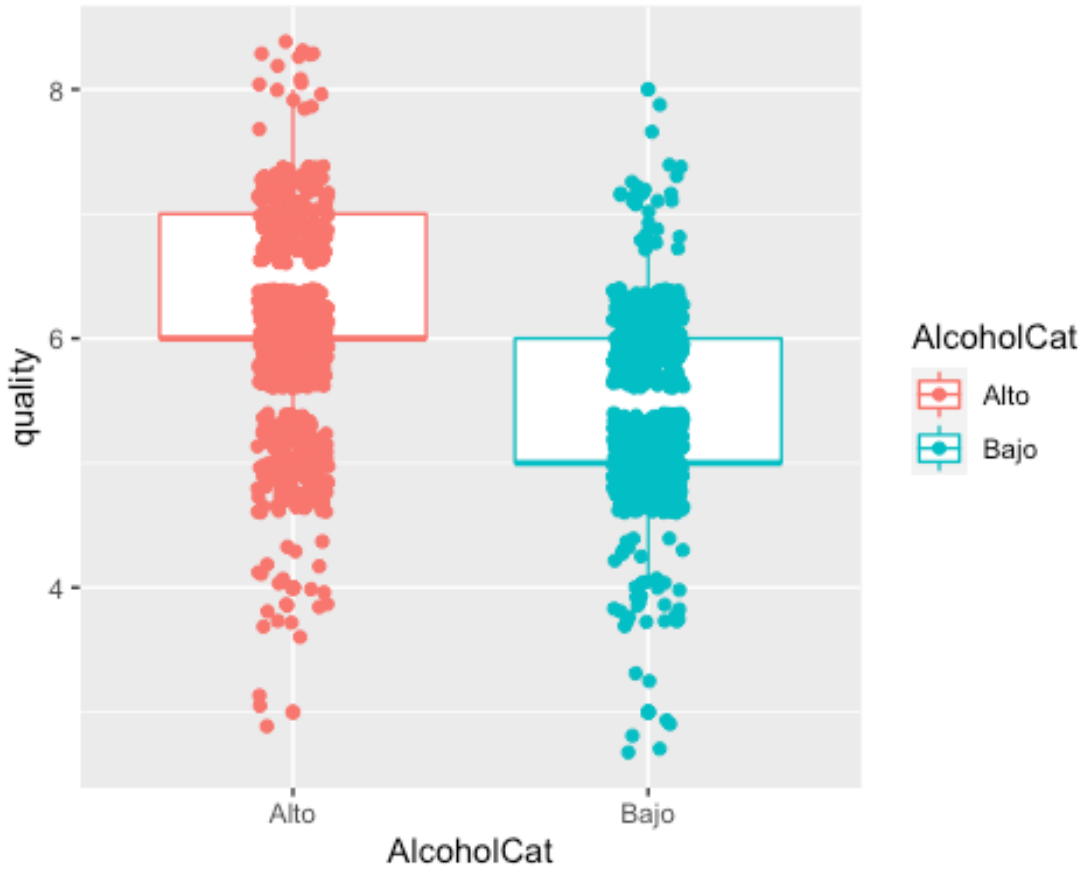
```
model.logist1=glm(formula=CalidadAlta ~ alcohol +volatile.acidity +sulphates+citric.acid +chlorides+total
.sulfur.dioxide,   data = data,family=binomial(link=logit))
summary(model.logist1)

##
## Call:
## glm(formula = CalidadAlta ~ alcohol + volatile.acidity + sulphates +
##     citric.acid + chlorides + total.sulfur.dioxide, family = binomial(link = logit),
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0992  -0.8477   0.3166   0.8562   2.3680
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -8.48525    0.86548  -9.804  < 2e-16 ***
## alcohol               0.90564    0.07112  12.734  < 2e-16 ***
## volatile.acidity     -3.50967    0.45134  -7.776 7.48e-15 ***
## sulphates             2.91628    0.44170   6.602 4.05e-11 ***
## citric.acid          -0.89010    0.39249  -2.268   0.0233 *
## chlorides             0.50032    2.45127   0.204   0.8383
## total.sulfur.dioxide -0.01226    0.00205  -5.979 2.25e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2209.0  on 1598  degrees of freedom
## Residual deviance: 1668.9  on 1592  degrees of freedom
```

```
## AIC: 1682.9
##
## Number of Fisher Scoring iterations: 4
```

5. Representación de los resultados a partir de tablas y gráficas.

Relativo al contraste de hipotesis, podemos observar:

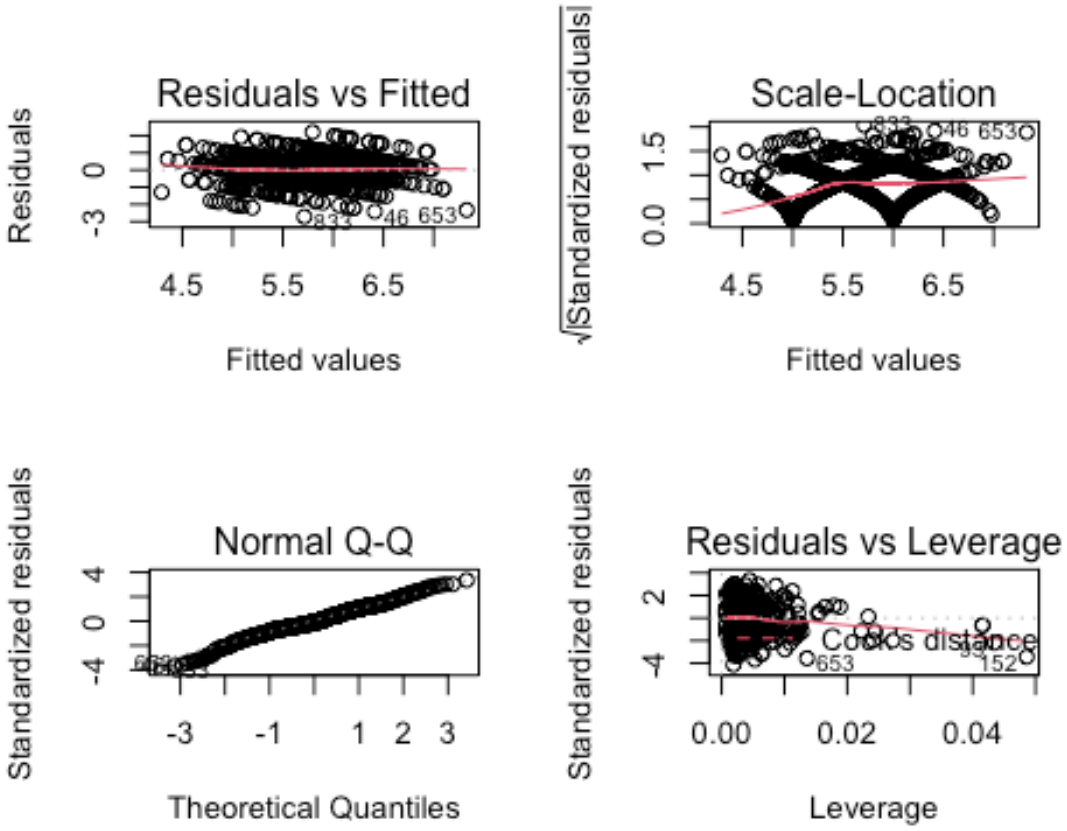
```
library("ggplot2")
data$AlcoholCat <- ifelse(data$alcohol < 10.42 , "Bajo", "Alto")
 ggplot(data =data, aes(x=AlcoholCat,y=quality, color = AlcoholCat )) + geom_boxplot() +
 geom_jitter(width = 0.1)
```



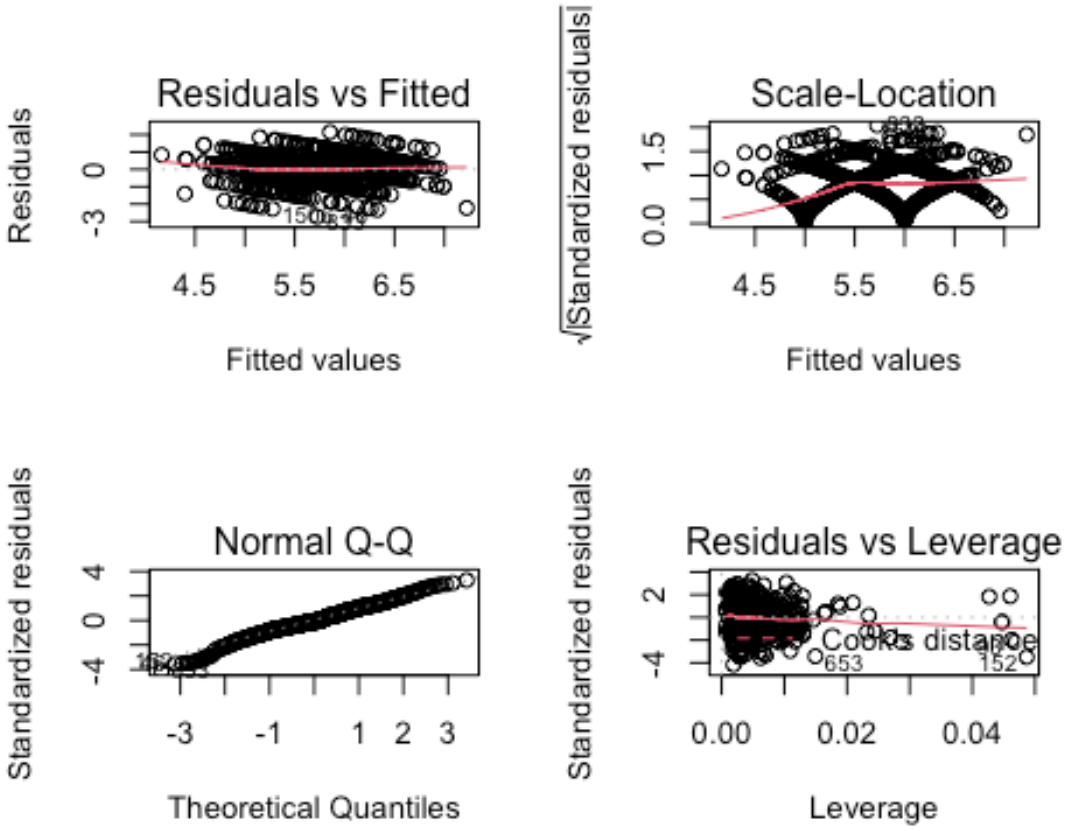
En esta representación de la calidad relacionada con el alcohol de las muestras. Podemos observar que para las muestras consideradas nivel de alcohol Alto, se obtiene un valor de quality superior a la quality de las muestras consideras nivel bajo de alcohol.

A continuación vamos a Visualizar los datos asociados a cada modelo, en especial el grafico Normal Q-Q y el gráfico de residuos frente valores ajustados,

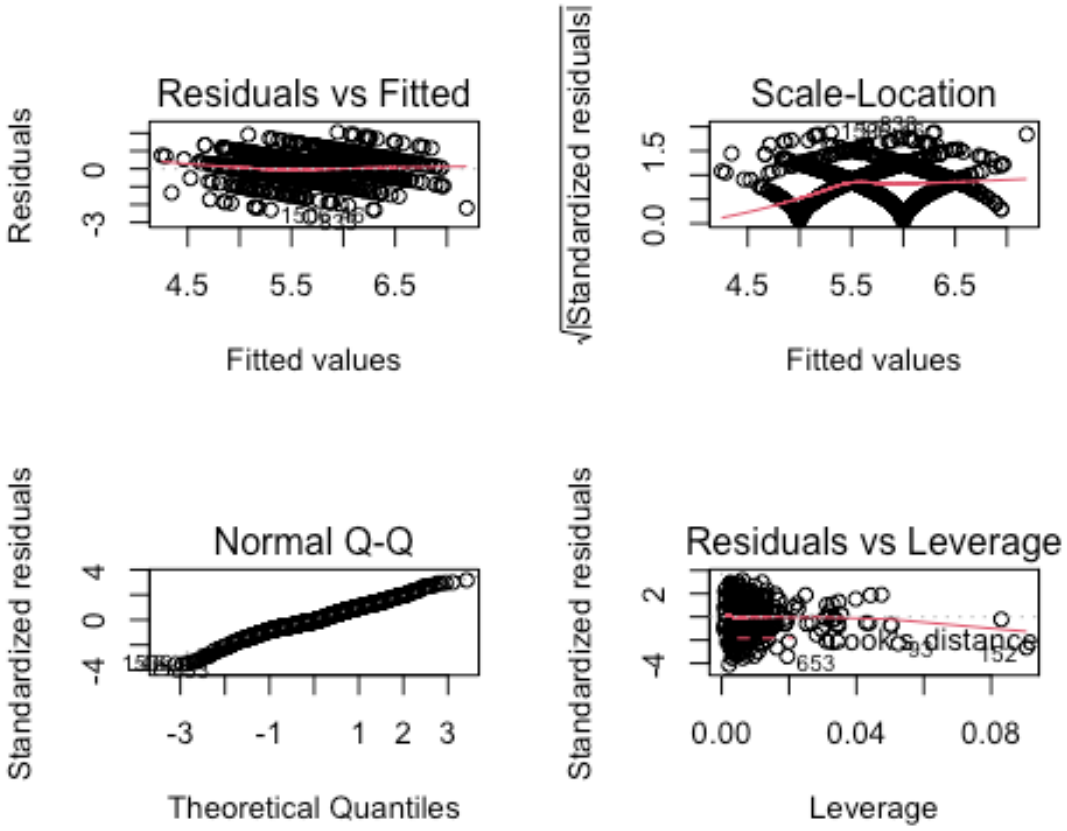
```
layout(matrix(c(1,2,3,4),2,2))
plot(modelo1)
```




```
layout(matrix(c(1,2,3,4),2,2))
plot(modelo2)
```




```
layout(matrix(c(1,2,3,4),2,2))
plot(modelo3)
```



 # Con las siguientes representaciones, nos hacemos una idea de cómo se comportan las variables alcohol ,volatile.acidity, sulphates, citric.acid, total.sulfur.dioxide y density, en relación con la calidad del vino

```
scatter.smooth(data$quality, data$alcohol)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1
```

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

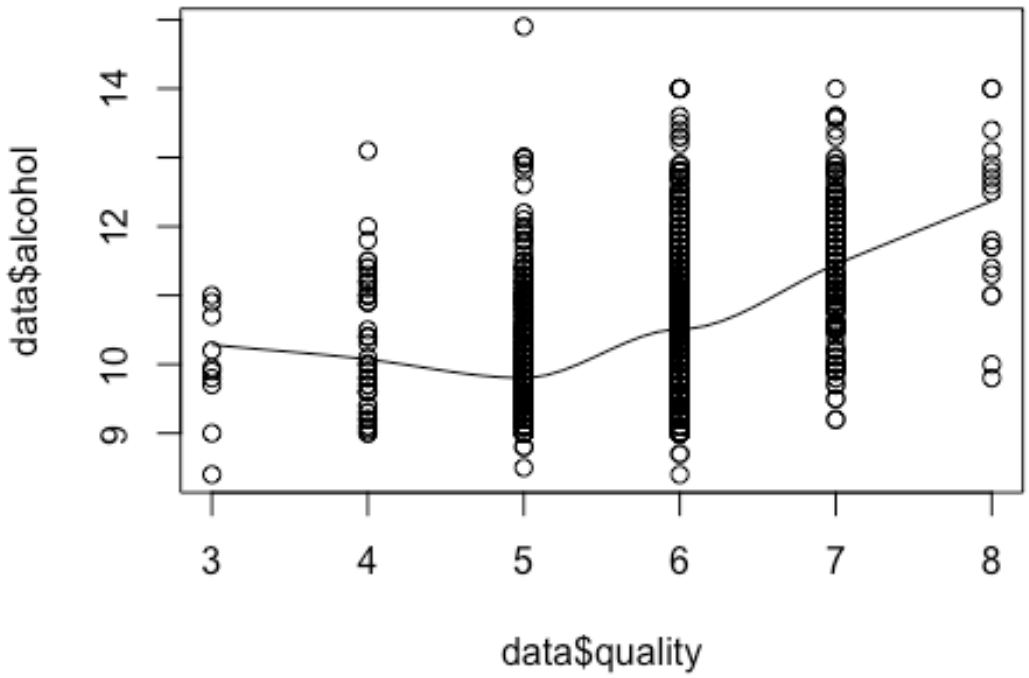
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1
```

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

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```




```
scatter.smooth(data$quality, data$volatile.acidity)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1
```

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

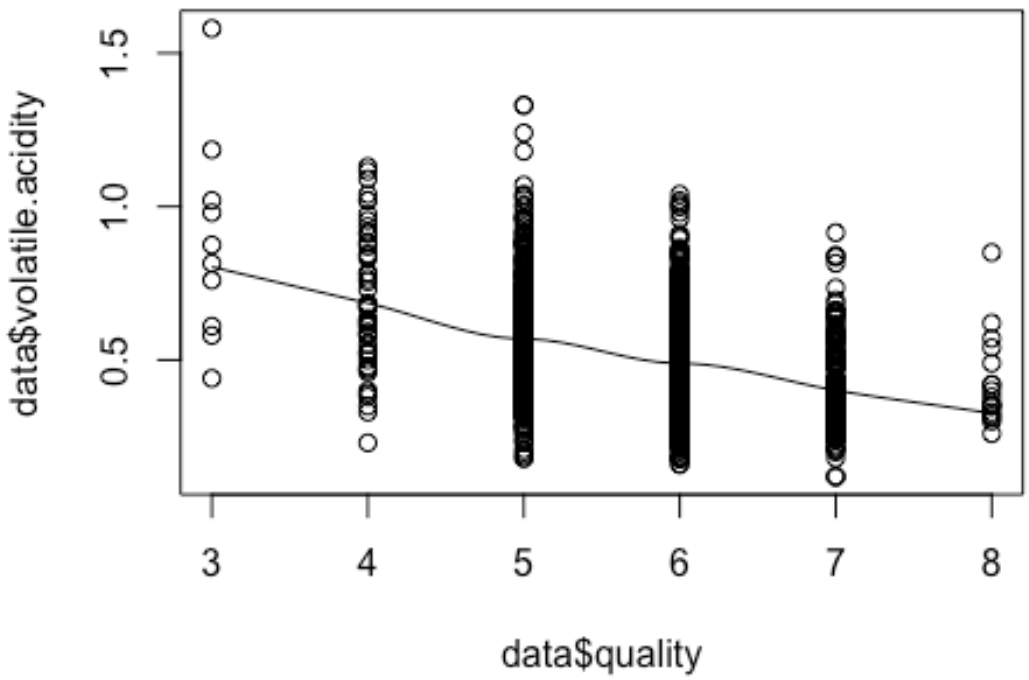
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5
```

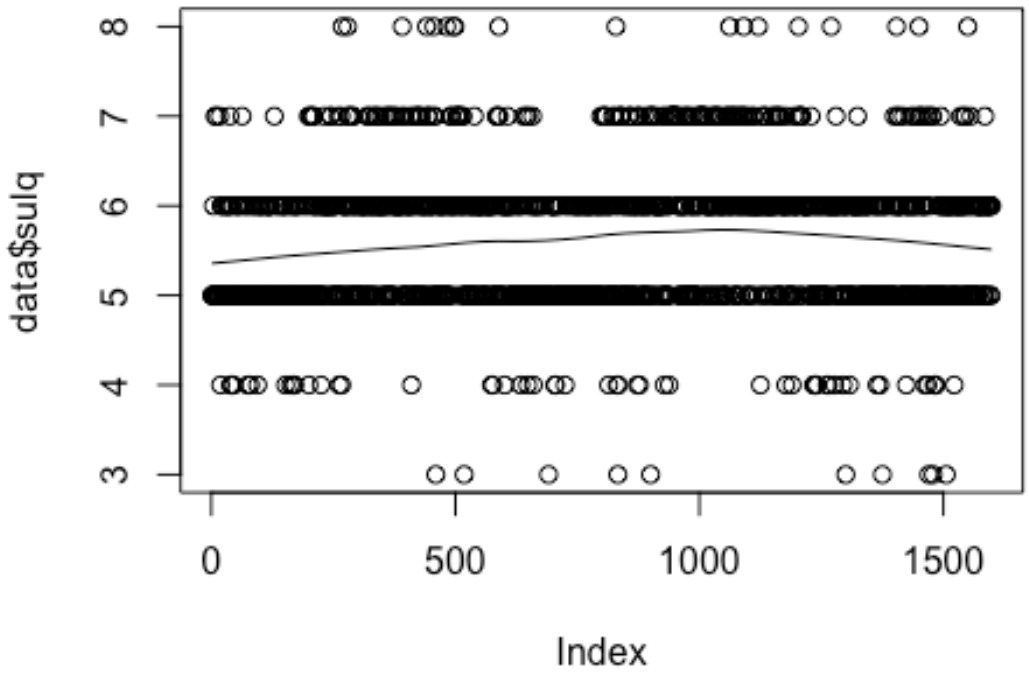
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1
```

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

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```




```
scatter.smooth(data$quality, data$sulq)
```




```
scatter.smooth(data$quality, data$citric.acid)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1
```

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

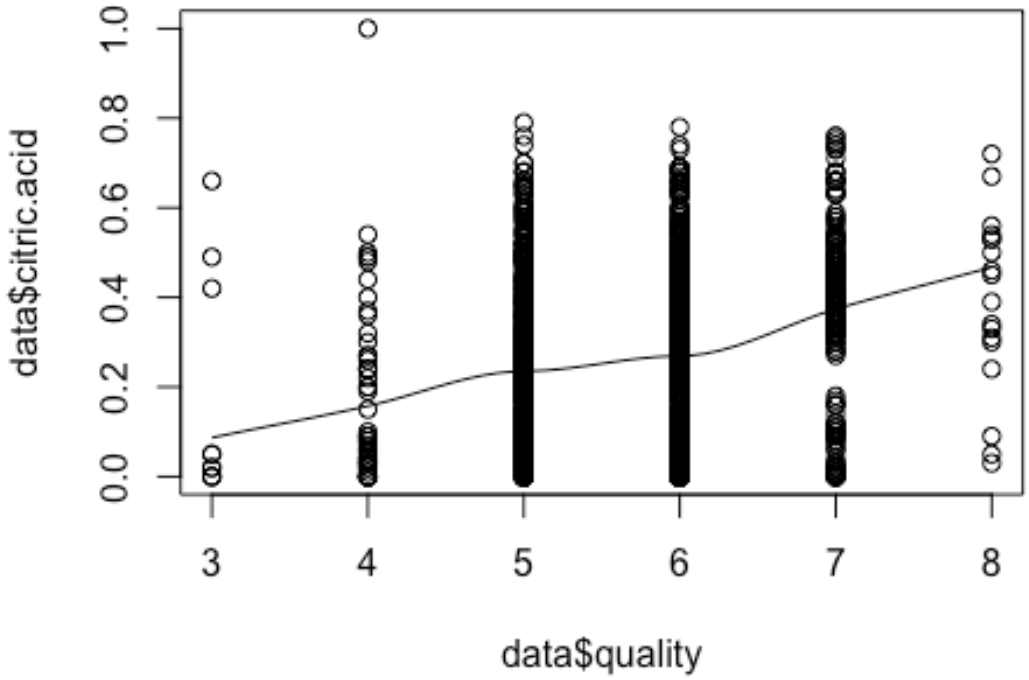
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1
```

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

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```




```
scatter.smooth(data$quality, data$total.sulfur.dioxide)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1
```

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

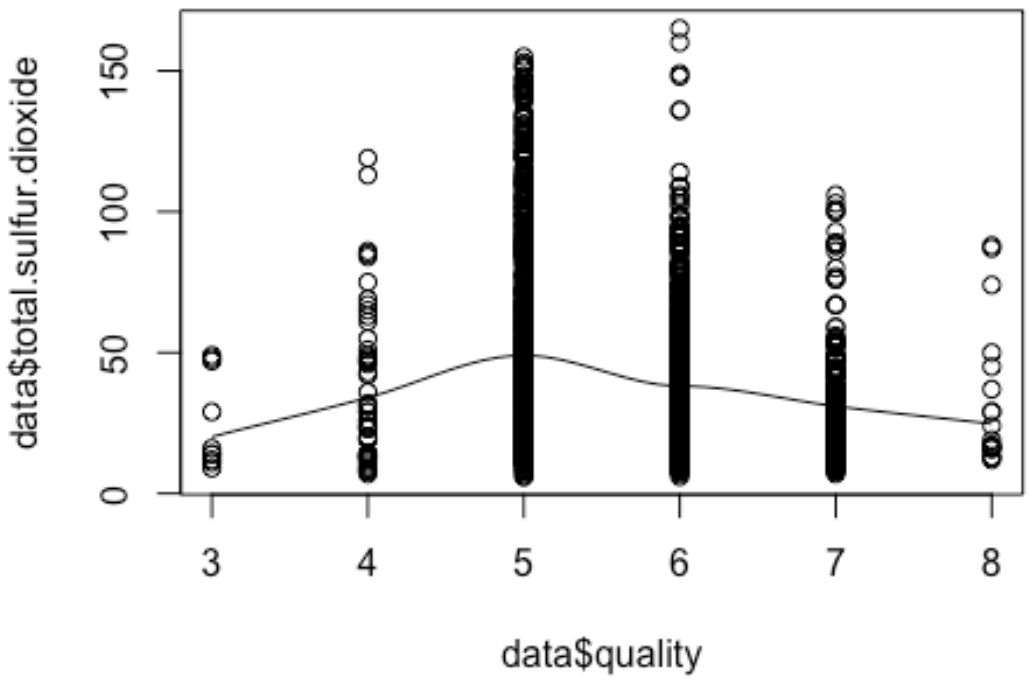
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1
```

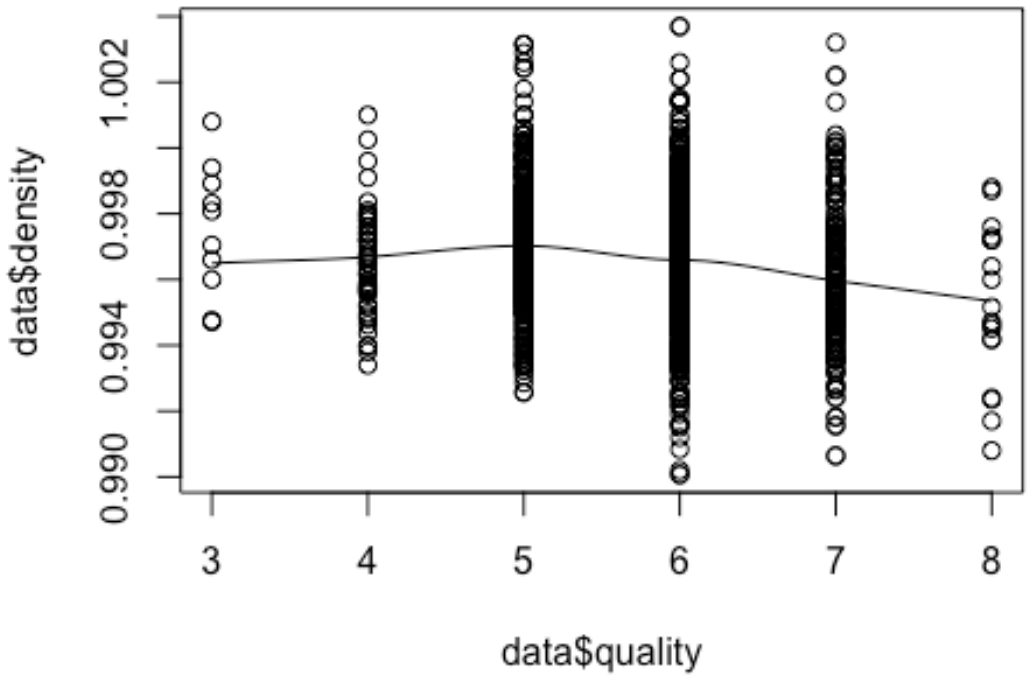
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```




```
scatter.smooth(data$quality, data$density)

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 5

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 1

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 1
```



Realizamos una tabla de comparación entre los modelos lineales implementados.

```
# Tabla con los coeficientes de determinación de cada modelo
tabla.coeficientes <- matrix(c(
1, summary(modelo1)$r.squared,
2, summary(modelo2)$r.squared,
3, summary(modelo3)$r.squared), ncol = 2, byrow = TRUE)
colnames(tabla.coeficientes) <- c("Modelo", "R^2")
tabla.coeficientes
```

```
##      Modelo      R^2
## [1,]      1 0.3361393
## [2,]      2 0.3438525
## [3,]      3 0.3516932
```

Se observa que el modelo lineal 3 es ligeramente mejor que sus predecesores.

Respecto al modelo logístico podemos realizar la visualización de los odds ratio de las variables regresoras mediante un intervalo de confianza del 95% e interpreta los intervalos obtenidos. ¿Qué regresor tiene más impacto en la probabilidad de calidad alta del vino?

```
exp(cbind(coef(model.logist1),confint(model.logist1)))
```

```
## Waiting for profiling to be done...

##                                  2.5 %      97.5 %
## (Intercept)          2.064915e-04 3.704236e-05 1.103863e-03
## alcohol              2.473523e+00 2.157031e+00 2.851013e+00
## volatile.acidity     2.990667e-02 1.217130e-02 7.148419e-02
## sulphates            1.847251e+01 7.840845e+00 4.438928e+01
## citric.acid          4.106158e-01 1.894415e-01 8.832982e-01
## chlorides            1.649255e+00 1.269212e-02 1.944006e+02
## total.sulfur.dioxide 9.878187e-01 9.838110e-01 9.917551e-01
```

Podemos indicar que el mayor impacto en incrementar la calidad del vino es alcohol.

```
#install.packages("caret")
#install.packages("e1071")

#Import required library
library(caret)

## Loading required package: lattice
```

Estimación de la precisión del modelo logístico Vamos a proporcionar la tabla de confusión correspondiente al modelo.

Se adopta como criterio que las predicciones obtenidas con el modelo logístico por encima de 0,5 pertenecerían al grupo de calidad alta. Anteriormente hemos definido: CalidadAlta será igual a 1 cuando los valores de la variable quality sean superiores o iguales a 6

```
confusionMatrix(table(predict(model.logist1, type="response") >= 0.5,data$CalidadAlta == "1"))

## Confusion Matrix and Statistics
##
##
##        FALSE TRUE
##   FALSE   539  213
##   TRUE    205  642
##
##             Accuracy : 0.7386
##               95% CI : (0.7163, 0.76)
##    No Information Rate : 0.5347
##    P-Value [Acc > NIR] : <2e-16
##
##                  Kappa : 0.475
##
##  Mcnemar's Test P-Value : 0.7321
##
##            Sensitivity : 0.7245
##            Specificity : 0.7509
##         Pos Pred Value : 0.7168
```

```
##          Neg Pred Value : 0.7580
##              Prevalence : 0.4653
##          Detection Rate : 0.3371
##    Detection Prevalence : 0.4703
##       Balanced Accuracy : 0.7377
##
##       'Positive' Class : FALSE
##
```

Se obtiene de esta manera que el modelo logístico tiene una precisión superior al 70%.

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

En el preprocesamiento de los datos valores extremos (outliers), se han utilizado los siguientes criterios adoptados según las variables del conjunto de datos:

- residual.sugar: Se decide que los valores por encima de 10 van a ser considerados outliers.

- chlorides: se decide considerar outliers a los valores superiores a 0,3

- free.sulfur.dioxide: se decide considerar outliers los valores por encima de 60.

- total.sulfur.dioxide: se considerar outliers los valores por encima de 170.

- sulphates: se decide considerar outliers los valores por encima de 1,5.

A todos esos casos se ha decidido imputar el valor de la mediana, excluidos los casos considerados outliers.
En el resto de variables no se han realizado modificaciones de los datos.

De análisis de los datos, podemos indicar que:

Del contraste de hipótesis realizado, la calidad de los vinos es diferente entre los considerados como nivel bajo de alcohol y nivel alto, con un nivel de confianza del 95%. Por lo observado, parece que el nivel de calidad puede ser superior para los vinos con un nivel alto de alcohol.

Del análisis de la correlaciones podemos indicar que las variables alcohol, volatile.acidity, sulphates, citric.acid y chlorides son las más correlacionadas con el nivel de calidad de los vinos del dataset. Esto significa que son las variables del dataset que ejercen una mayor influencia sobre el la calidad del vino.

Adicionalmente hemos realizados diferentes modelos de regresión, lineales y logística, que serían útiles para realizar predicciones sobre la calidad dadas unas características concretas, de las otras variables.

De lo anterior podemos indicar que del análisis de los datos podemos aclarar cómo algunas de las variables del conjunto de datos están relacionadas con el problema planteado, conocer la calidad del vino.

7. Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.

#exportacion de datos tras el tratamiento:

```
# write.csv(data,"C:\\Users\\Usuario\\winquality_modificado.csv", row.names = FALSE) # Para especificar un directorio distinto al actual
# Con la siguiente función se exportan los datos a un fichero csv llamado winequality_modificado en el working directory en el que estemos trabajando.
write.csv(data,'winequality_modificado.csv', row.names = FALSE)
```

CONTRIBUCIONES SEGÚN APARTADOS:

CONTRIBUCIONES	FIRMA
Investigación previa	ME / EC
Redacción de las respuestas	ME / EC
Desarrollo código	ME / EC