Tipología y Ciclo de vida de los Datos

PRACTICA 2: Limpieza y validación de los datos

Diana Cózar Salas – 6 ENE 2019

Contents

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

El dataset escogido es el ejemplo dado en el enunciado de esta misma práctica: Red Wine Quality.
Es un dataset libre disponible en Kaggle y se puede encontrar en la siguiente url:
https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

Según la propia descripción del dataset en Kaggle:

Este conjunto de datos está relacionado con las variantes rojas del vino portugués "Vinho Verde". Debido a cuestiones de privacidad y logística, solo están disponibles las variables fisicoquímicas y sensoriales (por ejemplo, no hay datos sobre tipos de uva, marca de vino, precio de venta del vino, etc.).

Es decir, en el presente dataset nos encontraremos con datos referentes a atributos medibles del vino, los cuales nos determinarán la calidad de este. Por tanto, los datos son importantes para poder catalogar los diferentes vinos según sus características y nos ayudarán a vinos futuros poder predecir su calidad en función de las cualidades medibles que presenten.

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

El dataset contiene los atributos descritos a continuación, todos ellos son de tipo numérico. Todos los atributos (exceptuando 'quality') pueden ser usados para el cálculo de la calidad del vino (atributo 'quality') utilizando modelos predictivos.

Por tanto, seleccionaremos todos los atributos de este dataset, ya que son relevantes para analizar las características de un vino y clasificarlo según ellos para obtener su calidad.
Aunque ya vemos por sus descripciones que algunos atributos están relacionados con otros (por ejemplo: la densidad, el nivel de alcohol y el azúcar residual).
Buscaremos dependencias entre atributos en la fase de análisis para posteriormente reducir la dimensionalidad y computar únicamente los atributos significativos.

- **fixed acidity:** muchos ácidos involucrados en el vino son fijos o no volátiles (no se evaporan fácilmente).

- **volatile acidity:** cantidad de ácido acético en el vino, el cual a niveles demasiado elevados puede producir un sabor avinagrado y desagradable.

- **citric acid:** encontrado en pequeñas cantidades, el ácido cítrico puede añadir frescura y sabor al vino.

- **residual sugar:** cantidad de azúcar remanente posterior a los procesos de fermentación, es raro encontrar vinos con menos de 1 gramos por litro, y los vinos con más de 45 gramos por litro son considerados vinos dulces.

- **chlorides:** cantidad de sal en el vino.

- **free sulfur dioxide:** la forma libre del SO2 existe en equilibrio con la molecular SO2 (un gas disuelto) y el ión bisulfito; previene el crecimiento de microbios y la oxidación del vino.

- **total sulfur dioxide:** cantidad de SO2 (en sus dos formas). En bajas concentraciones es prácticamente indetectable en el vino, pero en concentraciones sobre 50 ppm (partes por millón) entonces es evidente tanto en el gusto como en el olor del vino.

- **density:** la densidad del vino es cercana a la densidad del agua en función del porcentaje de alcohol y el contenido de azúcar.

- **pH:** describe lo ácido o básico que es un vino en una escala del 0 (muy ácido) a 14 (muy básico). La mayoría de vinos están en una escala de 3-4 pH.

- **sulphates:** un aditivo del vino que contribuye al gas SO2.

- **alcohol:** porcentaje de alcohol que contiene el vino.

- **quality:** variable de salida que califica el vino (basada en datos sensitivos, con una puntuación entre 0 y 10).

Los datos están en un archivo en formato csv (winequality-red.csv), con lo cual la carga de los datos se realiza en el código R directamente del fichero.

3. Limpieza de los datos

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

Primero de todo cargamos los datos en el entorno R mediante los comandos:

```
setwd('C:/RData')
wine <- read.csv('winequality-red.csv', header = TRUE, encoding = 'UTF-8')
```

Posteriormente, vemos los datos que hemos cargado y miramos las métricas básicas y los tipos a los que pertenecen:

```
View(wine)
summary(wine)
str(wine)
```

La salida de estos comandos es la siguiente: (el View no lo incluiremos, ya que saca por pantalla todas las filas del fichero)
Los tamaños de letra son pequeños para que se adapte al formato de salida de R

```
> summary(wine)
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar     chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900   Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900   1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200   Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539   Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600   3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500   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
```

```
> str(wine)
'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 ...
```

Con estos datos ya podemos verificar que todos los atributos son de tipo numérico (excepto 'quality' que es de tipo int ya que no tiene decimales), y que ninguno tiene valores a cero excepto el atributo 'chlorides'. Gracias a summary también vemos que los datos no contienen valores NA (valores vacíos).

En caso de contener valores vacíos, éstos deberían ser sustituidos por un valor, ya que sino los posteriores análisis no se podrían hacer. El valor por el cual se deberían sustituir los valores nulos sería diferente en

cada atributo, pero como norma general, se podrían sustituir por la mediana para no alterar demasiado las métricas del atributo.
Lo haríamos con el siguiente código, en caso de que fuera necesario:

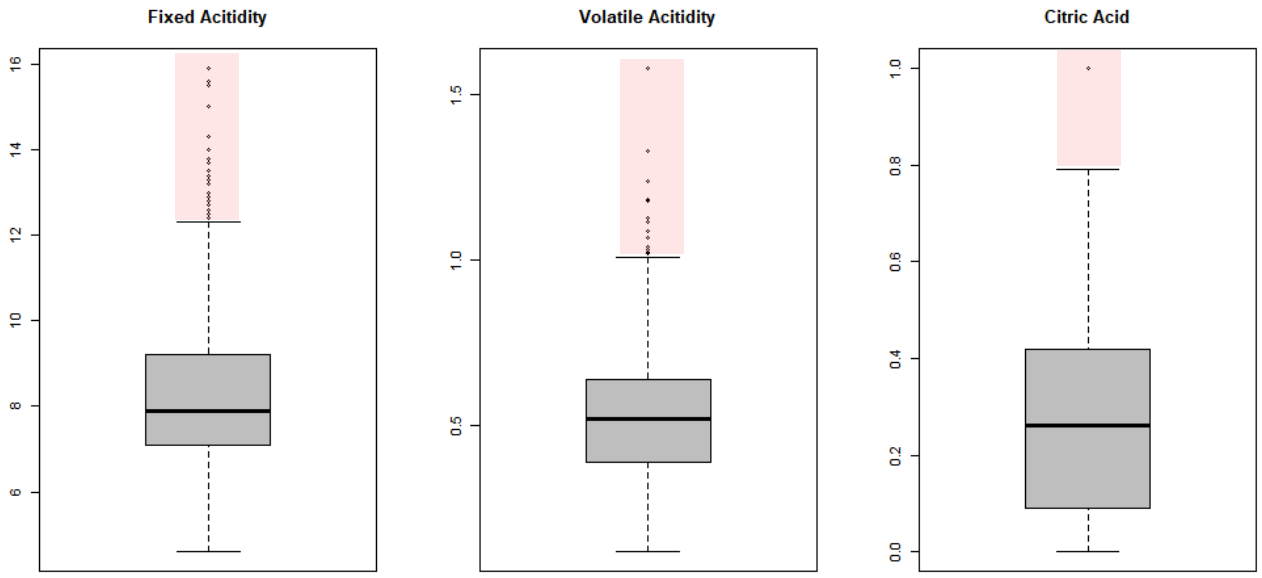
```
wine$<nombre_atributo>[is.na(wine$<nombre_atributo>)]<-median(wine$<nombre_atributo>,na.rm=TRUE)
```

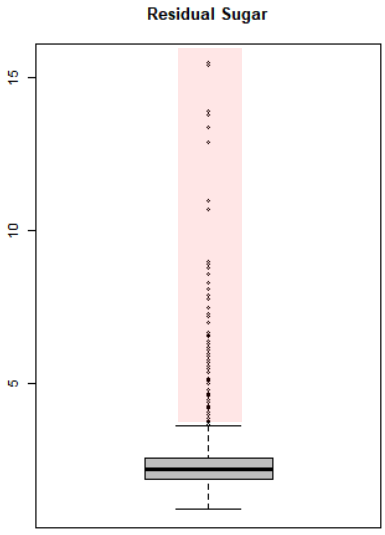
En el código R se incluye el ejemplo de limpieza de valores NA con el atributo 'fixed.acidity'. Esto no genera ningún cambio en el dataset ya que este atributo no contiene NAs.

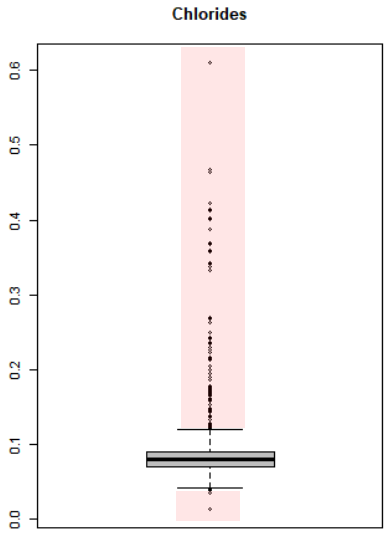
Para el tratamiento de los valores a cero, vemos que el único campo que contiene ceros es el atributo 'citric.acid'. Por la descripción del campo es posible que éste contenga valores a cero, con lo cual no es un error tenerlos en cuenta en el análisis. Un vino puede contener cero valores de ácido cítrico.
A no ser que posteriormente estos valores cero sean considerados posteriormente como valores extremos, no hay necesidad de tratarlos en este caso.

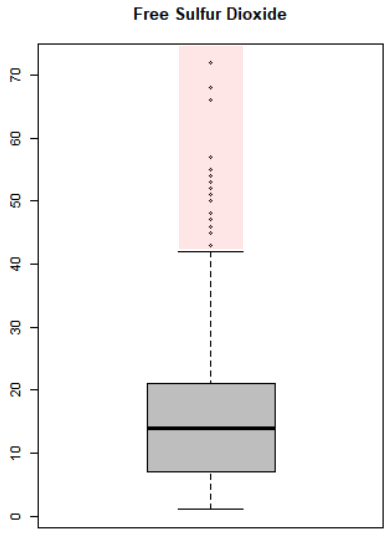
3.2. Identificación y tratamiento de valores extremos

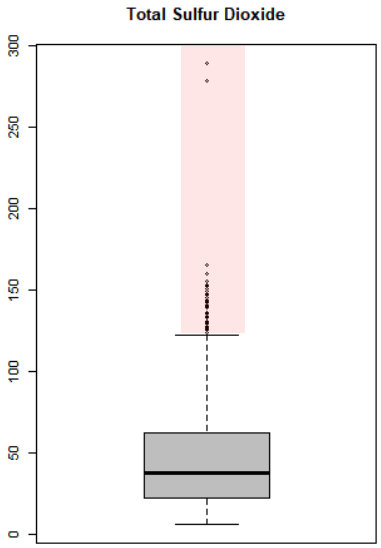
Para la identificación de los valores extremos, procederemos a la representación en boxplot de los atributos (excepto 'quality' que es el valor de salida). Este tipo de gráfico nos permite ver visualmente los valores extremos y las métricas básicas.

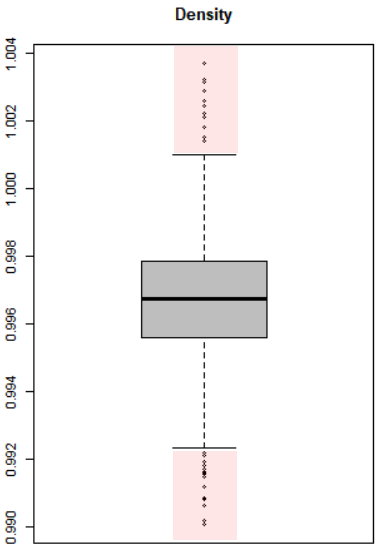


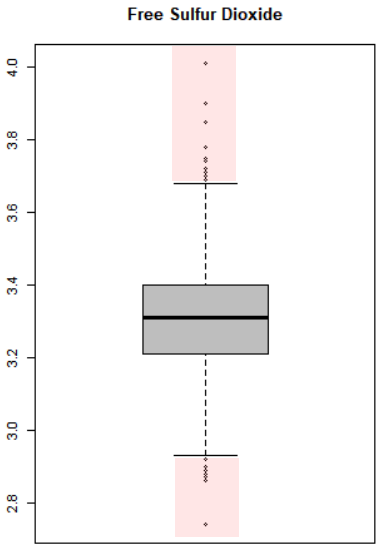

Residual Sugar
15
10
5

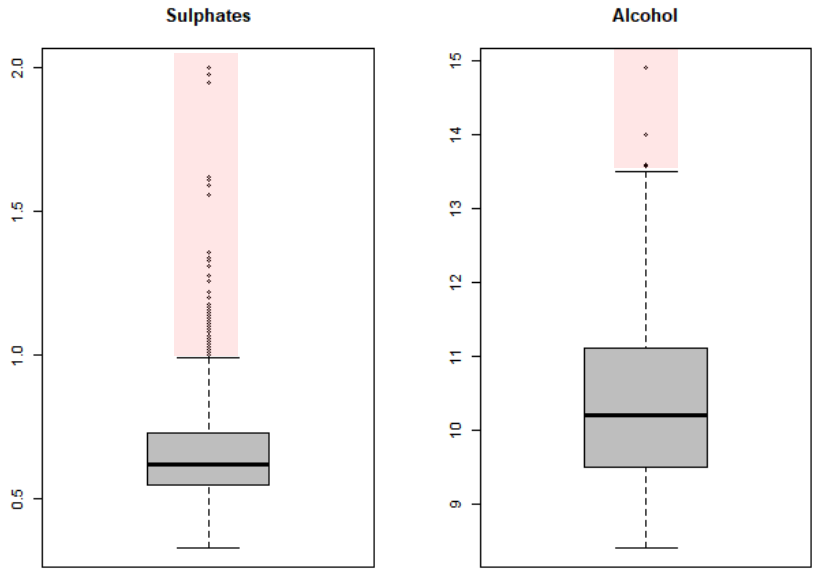

Chlorides
0.6
0.5
0.4
0.3
0.2
0.1
0.0


Free Sulfur Dioxide
70
60
50
40
30
20
10
0


Total Sulfur Dioxide
300
250
200
150
100
50
0


Density
1.004
1.002
1.000
0.998
0.996
0.994
0.992
0.990


Free Sulfur Dioxide
4.0
3.8
3.6
3.4
3.2
3.0
2.8



Como podemos ver, la mayoría de atributos contienen valores outliers, en el boxplot representados como los puntos que se salen del rango intercuartílico representado por las líneas inferior y superior de las gráficas (Q1 -1.5 x IQR) y (Q3 + 1.5 x IQR). Hemos marcado estos valores en rojo en las gráficas.

Paralelamente, antes habíamos detectado valores cero en el atributo 'citric.acid', ahora vemos que precisamente los valores cero de este atributo no son valores outliers.

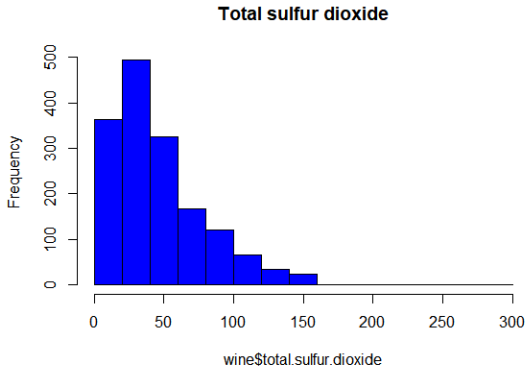
Así pues, vemos que este dataset tiene muchos valores extremos. Vamos a analizarlos más en profundidad, ya que quizás tenga sentido que estén ahí. Creamos una función que nos devuelve, dado un atributo, su desviación estándar y dónde empiezan los valores outliers.

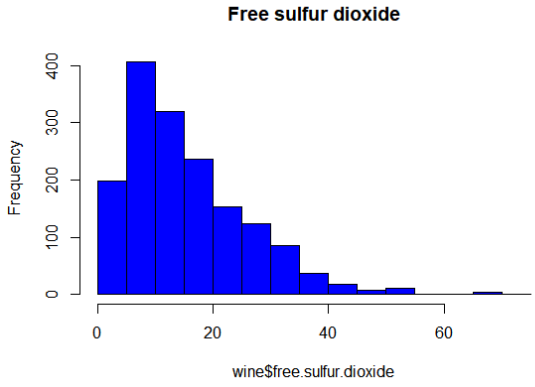
```
> stdesv_y_outliers(wine$fixed.acidity)
[1] "Desviación estándar: 1.7410963181277"
[1] "Outlier más pequeño: 13.9938371134469"
[1] "Número de veces de desvío out/mediana: 1.77137178651227"

> stdesv_y_outliers(wine$volatile.acidity)
[1] "Desviación estándar: 0.179059704153535"
[1] "Outlier más pequeño: 1.14670896453737"
[1] "Número de veces de desvío out/mediana: 2.20520954718726"

> stdesv_y_outliers(wine$citric.acid)
[1] "Desviación estándar: 0.194801137405319"
[1] "Outlier más pequeño: 0.941803980918615"
[1] "Número de veces de desvío out/mediana: 3.62232300353313"

> stdesv_y_outliers(wine$residual.sugar)
[1] "Desviación estándar: 1.40992805950728"
[1] "Outlier más pequeño: 7.13474820827548"
[1] "Número de veces de desvío out/mediana: 3.24306736739794"

> stdesv_y_outliers(wine$chlorides)
[1] "Desviación estándar: 0.0470653020100901"
[1] "Outlier más pequeño: 0.243728557035315"
[1] "Número de veces de desvío out/mediana: 3.08517160804197"

> stdesv_y_outliers(wine$free.sulfur.dioxide)
[1] "Desviación estándar: 10.4601569698097"
```

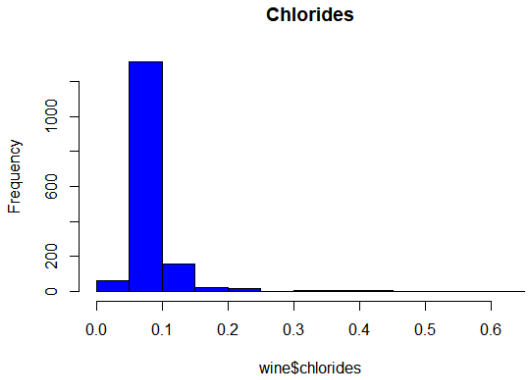
```
[1] "Outlier más pequeño: 50.610549394334"
[1] "Número de veces de desvío out/mediana: 3.61503924245243"

> stdesv_y_outliers(wine$total.sulfur.dioxide)
[1] "Desviación estándar: 32.8953244782991"
[1] "Outlier más pequeño: 153.133635674047"
[1] "Número de veces de desvío out/mediana: 4.02983251773807"

> stdesv_y_outliers(wine$density)
[1] "Desviación estándar: 0.00188733395384256"
[1] "Outlier más pequeño: 1.00335566883845"
[1] "Número de veces de desvío out/mediana: 1.00662720726205"

> stdesv_y_outliers(wine$pH)
[1] "Desviación estándar: 0.154386464903543"
[1] "Outlier más pequeño: 3.8503526271624"
[1] "Número de veces de desvío out/mediana: 1.16324852784363"

> stdesv_y_outliers(wine$sulphates)
[1] "Desviación estándar: 0.16950697959011"
[1] "Outlier más pequeño: 1.21327442856538"
[1] "Número de veces de desvío out/mediana: 1.95689423962159"

> stdesv_y_outliers(wine$alcohol)
[1] "Desviación estándar: 1.06566758184739"
[1] "Outlier más pequeño: 13.9298365364659"
[1] "Número de veces de desvío out/mediana: 1.36567024867313"
```

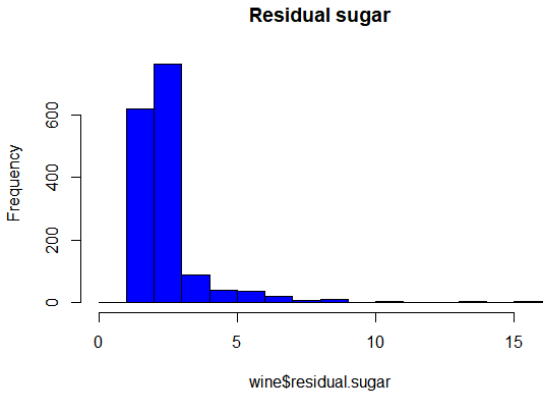
Vemos que los atributos que el desvió del outlier más pequeño respecto a la mediana es mayor son 'total.sulfur.dioxide', 'free.sulfur.dioxide', 'chlorides', 'residual.sugar', 'citric.acid' y 'sulphates'.

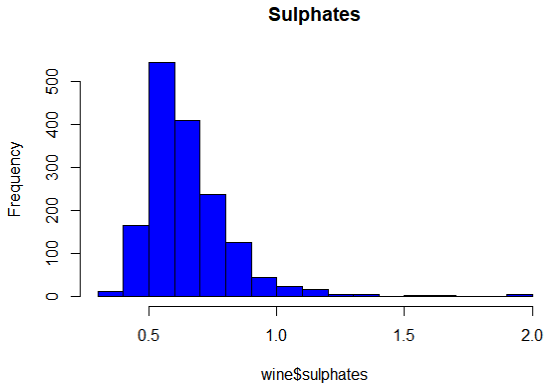
Representamos a través de los histogramas cómo los outliers afectan a la distribución de estas variables:









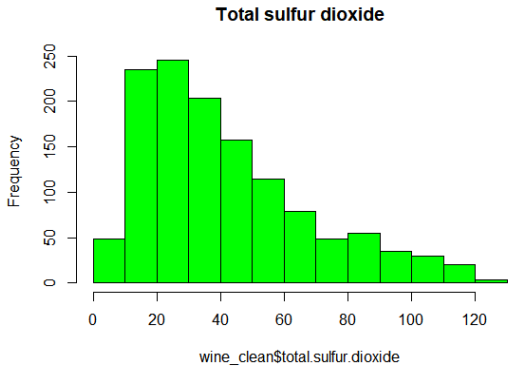


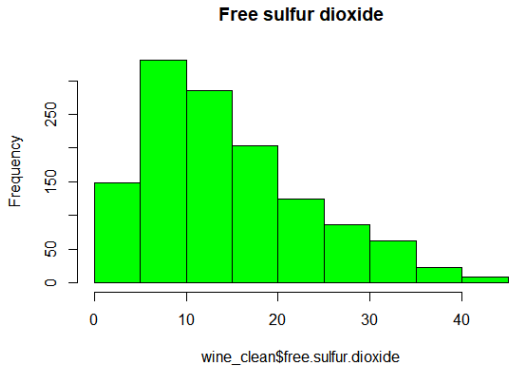
En todas las distribuciones vemos una cola hacia la parte superior de los valores. Así pues, vamos a eliminar los outliers de estos atributos del dataset para tener distribuciones que se aproximen a la normal.

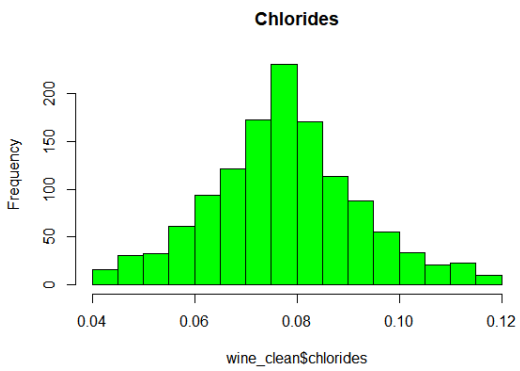
Guardamos los valores outliers de estos cuatro atributos en cuatro vectores y vemos la cantidad de elementos de los que estamos hablando en cada caso:

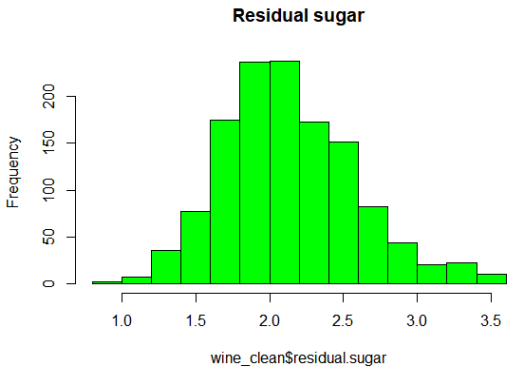
```
> length(outliers_chl)
[1] 112
> length(outliers_fsd)
[1] 30
> length(outliers_tsd)
[1] 55
> length(outliers_rs)
[1] 155
> length(outliers_sul)
[1] 59
```

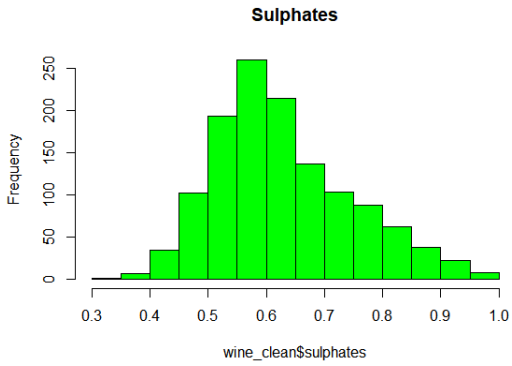
Estamos considerando eliminar un 0.09% de los datos en el caso mayor y un 0.018% en el caso menor. En estos casos, eliminaremos los outliers del dataset.
También sería interesante ver como estos valores outliers afectan a la calidad del vino, así que guardaremos un dataset con los valores originales para realizar el mismo análisis y ver si los resultados varían.
Así obtenemos dos datasets: wine (dataset original) y wine_clean (dataset sin valores outliers en estos cinco atributos), con 1.599 y 1.275 filas respectivamente. Hemos reducido un 0.797 % nuestro dataset original. Y al mostrar de nuevo los histogramas de estos atributos, vemos que su distribución es más similar a la normal:


Total sulfur dioxide
Frequency
0
50
100
150
200
250
0
20
40
60
80
100
120
wine_clean$total.sulfur.dioxide


Free sulfur dioxide
Frequency
0
50
150
250
0
10
20
30
40
wine_clean$free.sulfur.dioxide


Chlorides
Frequency
0
50
100
150
200
0.04
0.06
0.08
0.10
0.12
wine_clean$chlorides


Residual sugar
Frequency
0
50
100
150
200
1.0
1.5
2.0
2.5
3.0
3.5
wine_clean$residual.sugar


Sulphates
Frequency
0
50
100
150
200
250
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
wine_clean$sulphates

4. Análisis de los datos

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

El objetivo de este estudio es determinar la calidad del vino basándonos en datos medibles de sus características (o atributos). Para ello, crearemos un modelo de clasificación predictivo que nos indique la calidad de un nuevo vino en base a los atributos de éste.

En esta línea, estos algoritmos utilizan los atributos como dimensiones de entrada, en nuestro caso tenemos 11 atributos (quality no cuenta ya que es la variable de salida). Para empezar, dividiremos el dataset entonces en dos datasets, uno con los atributos de entrada (X) y otro que será el vector de salida (y) para el dataset manipulado wine_clean.

Como hemos comentado anteriormente, sería interesante ver cómo los valores outliers afectan a nuestros resultados; así pues, trabajaremos paralelamente con el dataset original para comparar resultados. Así que también vamos a generar los conjuntos de entrada (XX) y salida (yy) para el dataset wine original.

```
> X <- wine_clean[,1:11]
> y <- wine_clean[,12]

> XX <- wine[,1:11]
> yy <- wine[,12]
```

Una vez hecho esto, pasaremos a comprobar la varianza y aplicaremos correlación entre atributos para detectar los principales componentes. Después generaremos un modelo clasificatorio predictivo con los dos conjuntos (el resultante después de tratar los outliers y la reducción de dimensionalidad) y el original (XX).

Validaremos el modelo separando los datos originales es un conjunto de test y otro de entrenamiento, obteniendo así su fiabilidad. Posteriormente graficaremos los resultados.

4.2. Comprobación de la normalidad y homogeneidad de la varianza

Para comprobar la normalidad de la muestra, aplicamos el test de Shapiro al dataframe tratado (wine_clean) y obtenemos los siguientes resultados:

```
> shap_test_wine_clean <- lapply(wine_clean, shapiro.test)
> shap_test_wine_clean[[1]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.94651, p-value < 2.2e-16

> shap_test_wine_clean[[2]]
```

```
        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.9792, p-value = 1.366e-12

> shap_test_wine_clean[[3]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.94879, p-value < 2.2e-16

> shap_test_wine_clean[[4]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.97645, p-value = 1.434e-13

> shap_test_wine_clean[[5]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.99238, p-value = 3.733e-06

> shap_test_wine_clean[[6]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.93032, p-value < 2.2e-16

> shap_test_wine_clean[[7]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.91326, p-value < 2.2e-16

> shap_test_wine_clean[[8]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.99476, p-value = 0.0001977

> shap_test_wine_clean[[9]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.99188, p-value = 1.748e-06

> shap_test_wine_clean[[10]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.96911, p-value = 7.36e-16

> shap_test_wine_clean[[12]]

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.8508, p-value < 2.2e-16
```

Como el p-valor es menor que 0.05 siempre, esto nos dice que no se cumple la Ho, así que no podemos afirmar que los datos se distribuyen siguiendo una normal. Aplicamos el mismo test para el dataframe original y obtenemos los mismos resultados.

Para comprobar la homogeneidad de la varianza, realizaremos el test de Bartlett con los atributos del dataframe tratado (wine_clean).

```
> bartlett.test(wine_clean$fixed.acidity~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$fixed.acidity by wine_clean$quality
Bartlett's K-squared = 26.372, df = 5, p-value = 7.558e-05

> bartlett.test(wine_clean$volatile.acidity~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$volatile.acidity by wine_clean$quality
Bartlett's K-squared = 13.237, df = 5, p-value = 0.02126

> bartlett.test(wine_clean$citric.acid~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$citric.acid by wine_clean$quality
Bartlett's K-squared = 9.85, df = 5, p-value = 0.0796

> bartlett.test(wine_clean$residual.sugar~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$residual.sugar by wine_clean$quality
Bartlett's K-squared = 3.7969, df = 5, p-value = 0.579

> bartlett.test(wine_clean$chlorides~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$chlorides by wine_clean$quality
Bartlett's K-squared = 8.9449, df = 5, p-value = 0.1113

> bartlett.test(wine_clean$free.sulfur.dioxide~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$free.sulfur.dioxide by wine_clean$quality
Bartlett's K-squared = 5.9027, df = 5, p-value = 0.3158

> bartlett.test(wine_clean$total.sulfur.dioxide~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$total.sulfur.dioxide by wine_clean$quality
Bartlett's K-squared = 58.294, df = 5, p-value = 2.736e-11

> bartlett.test(wine_clean$density~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$density by wine_clean$quality
Bartlett's K-squared = 33.276, df = 5, p-value = 3.317e-06

> bartlett.test(wine_clean$pH~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances
```

```
data:  wine_clean$pH by wine_clean$quality
Bartlett's K-squared = 6.228, df = 5, p-value = 0.2847

> bartlett.test(wine_clean$sulphates~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$sulphates by wine_clean$quality
Bartlett's K-squared = 13.777, df = 5, p-value = 0.01709

> bartlett.test(wine_clean$alcohol~wine_clean$quality, wine_clean)

         Bartlett test of homogeneity of variances

data:  wine_clean$alcohol by wine_clean$quality
Bartlett's K-squared = 77.752, df = 5, p-value = 2.477e-15
```

En este caso, para los atributos donde el p-valor es mayor que 0.05 no podemos rechazar la hipótesis nula que afirma que la varianza es la misma para los grupos, con lo cual podemos considerar homogeneidad en la varianza. Esto es así para los atributos citric.acid, residual.sugar, chlorides, free.sulfur.dioxide, pH.

Aplicando el mismo test pata el dataset original sin tratar, tenemos que los atributos que cumplen homogeneidad de la varianza son citric.acid, residual.sugar, sulfur.dioxide, pH.

Vemos que los atributos que habíamos tratado en eliminación de outliers (residual.sugar, chlorides, free.sulfur.dioxide) presentan un p-valor mayor en el conjunto de datos tratado que en dataset original. De hecho, chlorides originalmente no presenta homogeneidad en la varianza y en el dataset tratado sí. Por tanto, la eliminación de valores extremos nos aporta homogeneidad en la varianza de las variables.

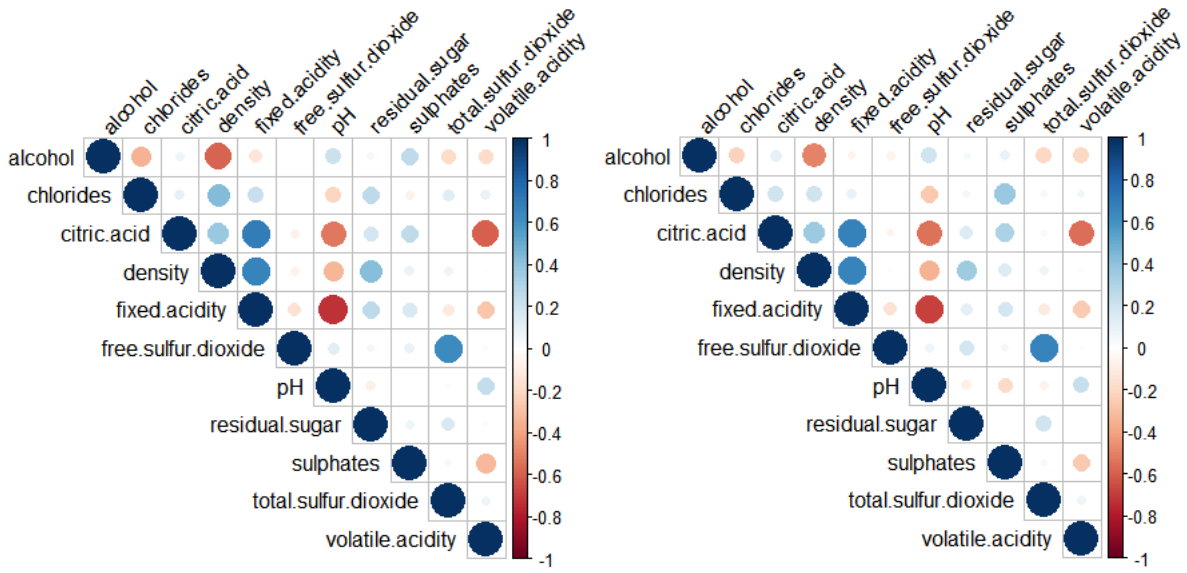
```
> bartlett.test(wine$citric.acid~wine$quality, wine)

         Bartlett test of homogeneity of variances

data:  wine$citric.acid by wine$quality
Bartlett's K-squared = 6.9368, df = 5, p-value = 0.2254

> bartlett.test(wine$residual.sugar~wine$quality, wine)

         Bartlett test of homogeneity of variances

data:  wine$residual.sugar by wine$quality
Bartlett's K-squared = 9.7537, df = 5, p-value = 0.08252

> bartlett.test(wine$chlorides~wine$quality, wine)

         Bartlett test of homogeneity of variances

data:  wine$chlorides by wine$quality
Bartlett's K-squared = 190.99, df = 5, p-value < 2.2e-16

> bartlett.test(wine$free.sulfur.dioxide~wine$quality, wine)

         Bartlett test of homogeneity of variances

data:  wine$free.sulfur.dioxide by wine$quality
Bartlett's K-squared = 8.5558, df = 5, p-value = 0.1282
```

Ahora bien, teniendo en cuenta esta distribución de los datos, antes de generar el modelo predictivo escalaremos los datasets (y al aplicar el PCA utilizaremos los parámetros “scale” and “center” a TRUE por este mismo motivo).

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.

Llegados a este punto consideramos si el número de filas u atributos van a ser demasiado pesados para los algoritmos de análisis. Tenemos 11 atributos y 1.275 filas. El número de filas claramente es de un orden asumible. El número de atributos también lo es, pero existe otro problema: puede haber redundancia entre las características del vino.

Para ello veremos la matriz de correlación entre los atributos para ver si hay atributos que son redundantes entre sí y podemos reducir la dimensionalidad del análisis aplicando procesos específicos de reducción de la dimensionalidad (como podría ser el método PCA – Principal Component Analysis).



Las matrices de correlación del conjunto de atributos X e XX respectivamente. Las correlaciones positivas se muestran en azul y las negativas en color rojo. La intensidad del color y el tamaño del círculo son proporcionales a los coeficientes de correlación. En el lado derecho del correlograma, el color de la leyenda muestra los coeficientes de correlación y los colores correspondientes.

Analizando los resultados, en el conjunto de atributos X las variables fixed.acidity está muy relacionada con citric.acid, density y pH. También alcohol está relacionado con density. Y free.sulfur.dioxid con total.sulfur.dioxid.

Para el conjunto de atributos XX vemos que las correlaciones significativas se mantienen.

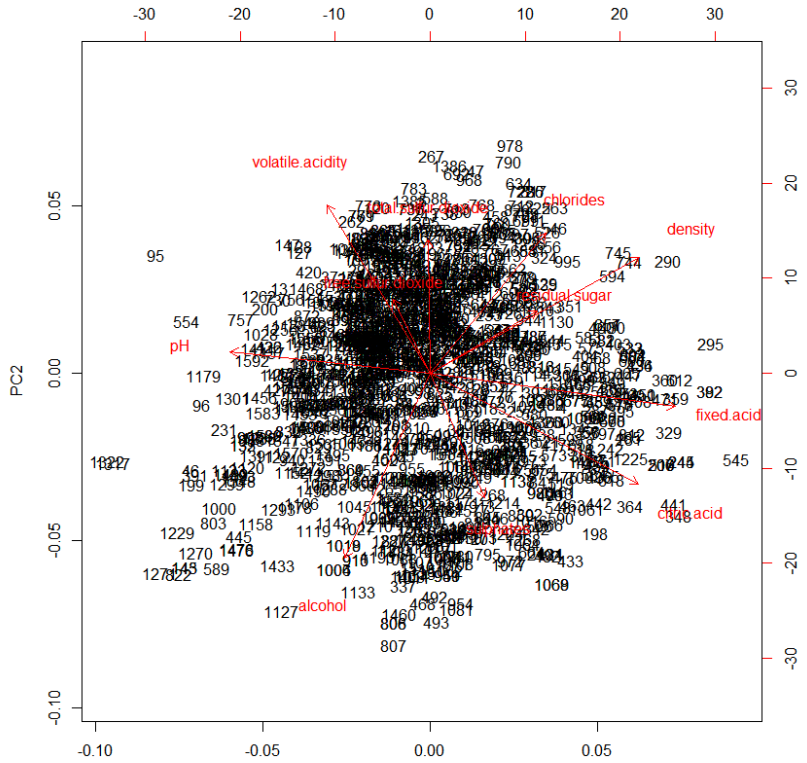
Con esto vemos que hay redundancia en los atributos para la clasificación, así que vamos a aplicar un algoritmo de PCA para reducir la dimensionalidad del conjunto de atributos X (el XX lo dejaremos tal y

como está para así poder hacer la comparativa final entre el modelo de clasificación sin outliers y con reducción de dimensionalidad, y el modelo con el dataset original).

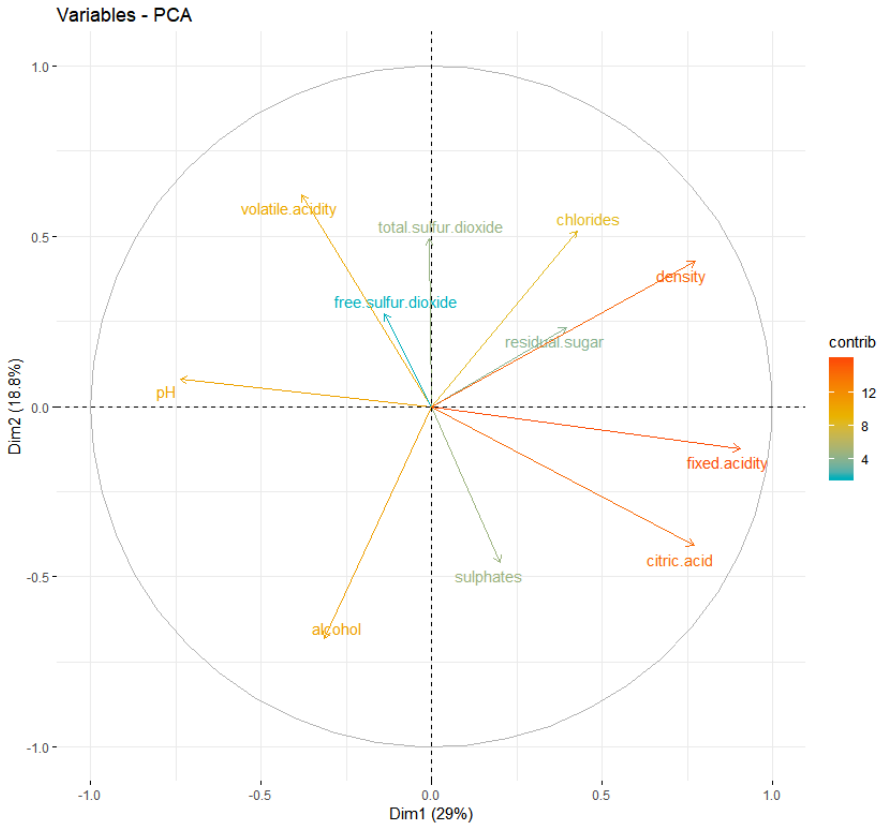
Como detectamos varios atributos que están relacionados con el resto, vamos a aplicar un PCA para obtener las componentes principales.

```
> pca <- prcomp(X,center = TRUE,scale. = TRUE)

> print(pca)
Standard deviations (1, .., p=11):
 [1] 1.7872486 1.4376824 1.2843047 1.0409147 0.9048104 0.8218113 0.7601597 0.6015773 0.5902765 0.4129541
[11] 0.2304746

Rotation (n x k) = (11 x 11):
                              PC1          PC2          PC3          PC4          PC5          PC6          PC7
fixed.acidity         0.506112488 -0.08594057  0.06151118 -0.03432196 -0.099979452  0.25120892 -0.219410299
volatile.acidity     -0.213257660  0.43234893  0.23720136  0.21217497 -0.098854442  0.36953351 -0.538112620
citric.acid           0.430417616 -0.28405721 -0.14879550 -0.11112372 -0.116203020 -0.07280482  0.178714932
residual.sugar        0.221458627  0.16007108 -0.18132507  0.68404676 -0.387781538  0.10605292  0.263566552
chlorides             0.238645083  0.35852364  0.04678428  0.17184895  0.073741340 -0.83016329 -0.270550674
free.sulfur.dioxide  -0.077145267  0.18923931 -0.64888653 -0.18257053 -0.003343186  0.07371892 -0.136885169
total.sulfur.dioxide -0.003586607  0.34217468 -0.57335330 -0.19607736 -0.132532154  0.03615833 -0.005467581
density               0.432290060  0.29594267  0.05906207  0.16586568  0.281206794  0.25241686  0.201148750
pH                   -0.412732315  0.05571143 -0.09134277  0.35266145  0.299777504 -0.05909019  0.445112992
sulphates             0.112859586 -0.31911598 -0.30038367  0.32554573  0.672085657  0.09064013 -0.371482002
alcohol              -0.176613687 -0.47478659 -0.17576779  0.33817659 -0.415568969 -0.11160983 -0.303870351
                             PC8          PC9         PC10         PC11
fixed.acidity        -0.2804660  0.10837480  0.41492015 -0.58965673
volatile.acidity     -0.2669336 -0.18862536 -0.35126644 -0.01411565
citric.acid          -0.4088995 -0.29887205 -0.62591065  0.02309684
residual.sugar        0.3687204  0.10227057 -0.15894095 -0.14754405
chlorides            -0.1059706  0.03163885  0.01769002 -0.04937857
free.sulfur.dioxide  -0.1920270  0.64066397 -0.17921395  0.01502803
total.sulfur.dioxide  0.0818546 -0.63982553  0.28355668 -0.01843679
density              -0.2512568  0.06730422  0.25012539  0.61730754
pH                   -0.5284060 -0.07782305  0.09857838 -0.32766425
sulphates             0.2418863 -0.14326747 -0.09693020 -0.05360675
alcohol              -0.3021185 -0.03084957  0.30482118  0.36809352

> summary(pca)
Importance of components:
                          PC1    PC2    PC3    PC4     PC5    PC6     PC7    PC8     PC9    PC10    PC11
Standard deviation     1.7872 1.4377 1.2843 1.0409 0.90481 0.8218 0.76016 0.6016 0.59028 0.4130 0.23047
Proportion of Variance 0.2904 0.1879 0.1499 0.0985 0.07443 0.0614 0.05253 0.0329 0.03168 0.0155 0.00483
Cumulative Proportion  0.2904 0.4783 0.6282 0.7267 0.80116 0.8626 0.91509 0.9480 0.97967 0.9952 1.00000
```

Graficando los resultados del PCA, obtenemos la siguiente representación:



Y aquí podemos ver la contribución de cada uno de los atributos a los dos componentes principales (PCA1 y PCA2) que explican el 47.83% de la varianza del dataset:



Se han generado 11 componentes PCA en los cuales cada atributo del dataset original tiene una contribución. El PC1 explica un 29% de la varianza del dataset, el PC2 un 18.7%, el PC3 un 15%, el PC4 casi un 10% y el PC5 un 7.4%, etc. Elegiremos 6 componentes principales, hasta el PC6, ya que éstos explican un 86% de la varianza. El nuevo dataset que contiene los principales componentes es X_pca.

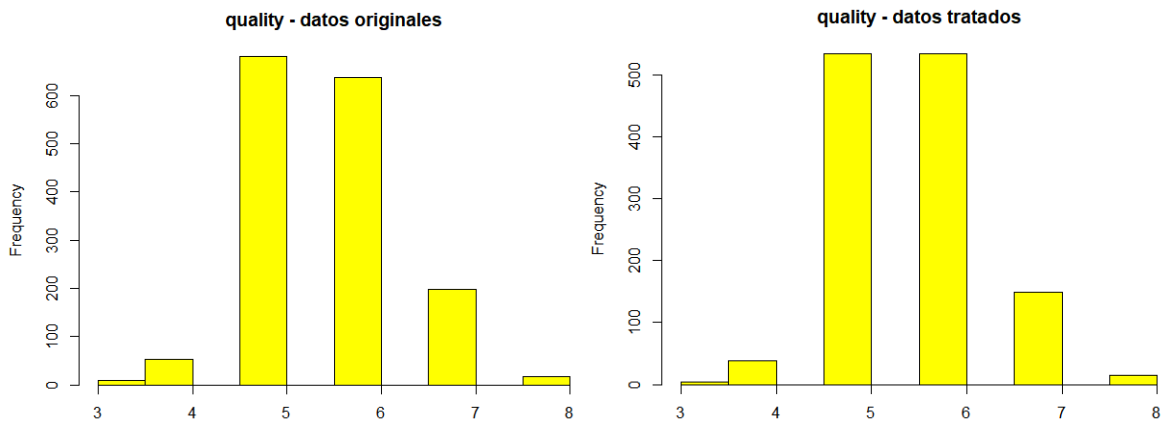
```
> X_pca <- pca$x[,1:6]
> X_pca
              PC1           PC2           PC3           PC4           PC5           PC6
1   -1.288364665  1.4552490525  1.2748135802  0.0721963693  0.9307149240  0.5068838957
2   -0.159679351  2.6196440725 -0.6508281288  0.7205159512 -0.1083156059  0.0487306119
3   -0.221618614  1.7572789207  0.3231028504  0.4016312737  0.2129820655 -0.0696325797
4    2.540642367 -0.5210343463 -0.5206666951 -1.7932992021 -0.2501639025  0.2445794450
5   -1.288364665  1.4552490525  1.2748135802  0.0721963693  0.9307149240  0.5068838957
6   -1.323396255  1.4169033056  0.9786579016 -0.2274116217  1.0036335457  0.4803526727
7   -1.042854390  1.1840512090  0.6999279051 -1.8030928944 -0.1924785650  0.5486978376
8   -2.456396568  0.1232713925  1.5365287609 -1.8022547252  0.0815316044  0.3225440363
9   -0.796819419  0.5205124362  1.5729676606 -0.1984266688  0.5049353054  0.3733625993
11  -0.732153843  1.7898449778  0.3153608678 -1.1606135163  0.2343550101 -1.2055014219
13  -2.873382827  1.3076151491  0.1986619918 -0.7019970807  0.4392531228 -1.2892608414
17   1.189684497  0.0386337523 -3.4646510302 -1.4640944968  0.6525946955 -1.0990186796
21   0.717370527  0.0420929906 -1.4417486251 -1.9935317528  0.2018755078 -0.4891492359
22   0.157007807  0.9259935705 -1.4975196303  0.0675347608  0.8722014931 -0.2496095768
23   0.982916442 -0.2291224116  0.2382617810 -0.4003974132  2.4164056636 -1.6426124572
24   0.549975116  1.2291763479  0.4756462389 -0.7420518550 -0.5584688043 -0.1841249787
25  -0.451221181  0.6930628541 -0.5183810992  0.2615131895  0.5865457876 -0.6534441524
26  -1.130145927 -0.1267119436  1.1838818785 -1.6547149067  0.9206232157 -0.9906230064
```

Por tanto, procederemos a generar un modelo de clasificación utilizando el dataset X_pca como entrada y el vector y como output. Y otro modelo utilizando el dataset original XX como entrada y el vector yy como salida.
Para poder entrenar y después testear el modelo, separaremos los datos en dos conjuntos: uno de train (entrenamiento) y otro de test. Tendremos, así pues:

X_pca, y separados en X_pca_train, X_pca_test, y_train, y_test
XX, yy separados en XX_train, XX_test, yy_train, yy_test

Donde el conjunto train tiene 2/3 de los datos del dataset y el de test tiene el 1/3 restante.

Una vez tenemos estos conjuntos, ya podemos generar un modelo predictivo que clasifique los vinos en función de los atributos a su valor de calidad. Para ello primero vamos a ver el vector de calidad qué aspecto tiene.



Vemos que la calidad del vino en los datos que tenemos actualmente está entre 3 y 8, aunque por definición la calidad puede estar entre 0 y 10, en este conjunto de datos no tenemos valores bajos (de 0 a 2) ni altos (9 y 10).

Por tanto no nos interesa un algoritmo de clasificación al cual tengamos que decirle a priori cuantos conjuntos debe generar (como sería el K-means), sino uno que determine el número de conjuntos a generar basándose en los datos disponibles (como el K-nn). Así que generaremos un modelo utilizando el k-nearest neighbors.

Ahora bien, para aplicar el algoritmo k-nn con éxito, es importante tener datos normalizados, ya que este procedimiento clasifica los valores dentro de una agrupación basándose en la proximidad de sus puntos vecinos. Si los datos están normalizados, se evita la creación de grupos ficticios muy alejados de sus vecinos compuestos por valores lejanos, que mediante la normalización se "acercan" más a la mayoría.
Escalamos los datos de nuestros datasets de train y test:

```
> X_pca_train <- scale(X_pca_train)
> X_pca_test <- scale(X_pca_test)
> XX_train <- scale(XX_train)
> XX_test <- scale(XX_test)
```

Una vez tenemos los datasets listos, generamos los modelos de k-nn:

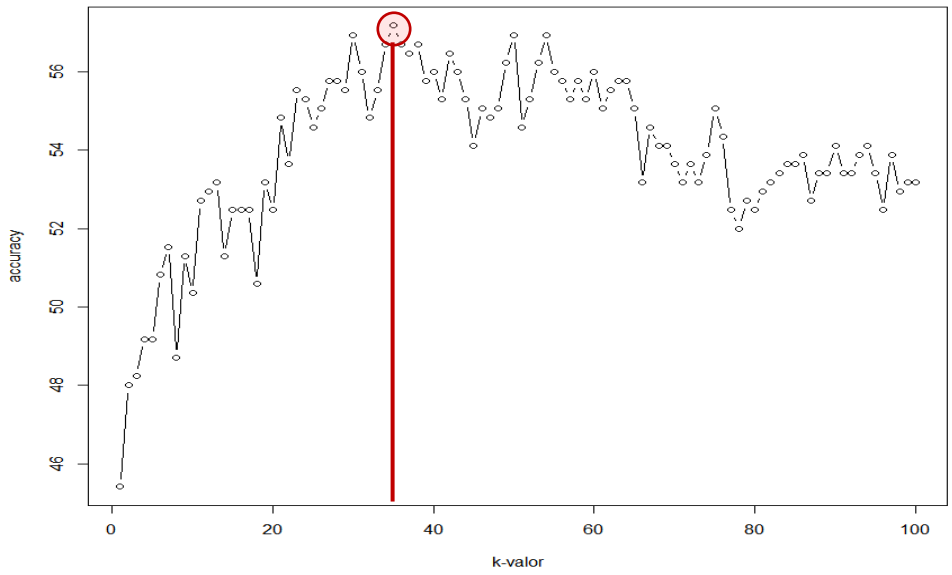
- Generaremos y entrenaremos un modelo para el conjunto de datos tratado (X_pca_train, y_train) y los testearemos (X_pca_test, y_test).
- Generaremos y entrenaremos otro modelo para el conjunto de datos original (XX_train, yy_train) y lo testearemos (XX_test, y_test).

Empezamos por el dataset tratado. Antes que generar el modelo k-nn debemos saber el valor óptimo de k (distancia a considerar entre vecinos más cercamos para clasificarlos dentro del mismo grupo). Así pues, vamos a considerar valores posibles entre 0 y 100.
Generamos los modelos iterativamente para todos estos valores de k y mostramos una gráfica que nos indique la precisión del modelo para cada caso:

```
> i = 1
> k_optimo = 1
> for (i in 1:100){
```

```
+   knn_mod <-  knn(train = X_pca_train, test = X_pca_test, cl = y_train, k = i)
+   k_optimo[i] <- 100 * sum(y_test == knn_mod)/length(y_test)
+   k=i
+ }
> plot(k_optimo, type="b", xlab='k-valor',ylab='accuracy')
```

Esto nos da como resultado la siguiente gráfica, donde vemos que el valor óptimo es k = 36.



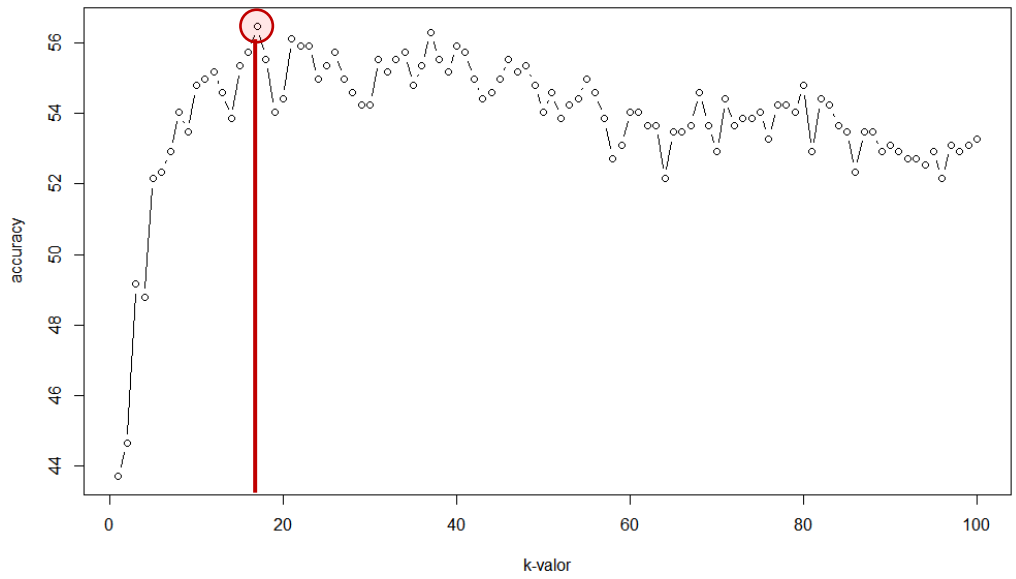
Generamos el modelo con este valor de k y los conjuntos de datos X_pca_train, X_pca_test, y_train, y_test. Comprobamos su precisión y obtenemos un 57.4%.

```
> knn <-  knn(train = X_pca_train, test = X_pca_test, cl = y_train, k = 36)
> acc <- 100 * sum(y_test == knn)/length(y_test)
> acc
[1] 57.41176
```

A continuación, hacemos el mismo ejercicio para el dataset original. Realizamos los mismos pasos: bucle para obtener el valor óptimo de k y generación del modelo y testeo de este.

```
> i = 1
> k_optimo = 1
> for (i in 1:100){
+   knn_mod <-  knn(train = XX_train, test = XX_test, cl = yy_train, k = i)
+   k_optimo[i] <- 100 * sum(yy_test == knn_mod)/length(yy_test)
+   k=i
+ }
> plot(k_optimo, type="b", xlab='k-valor',ylab='accuracy')
```

Esto nos da como resultado la siguiente gráfica, donde vemos que el valor óptimo es k = 18.



Generamos el modelo con este valor de k y los conjuntos de datos XX_train, XX_test, yy_train, yy_test. Comprobamos su precisión y obtenemos un 56.29%.

```
> knn_orig <-  knn(train = XX_train, test = XX_test, cl = yy_train, k = 18)
> acc_orig <- 100 * sum(yy_test == knn_orig)/length(yy_test)
> acc_orig
[1] 56.28518
```

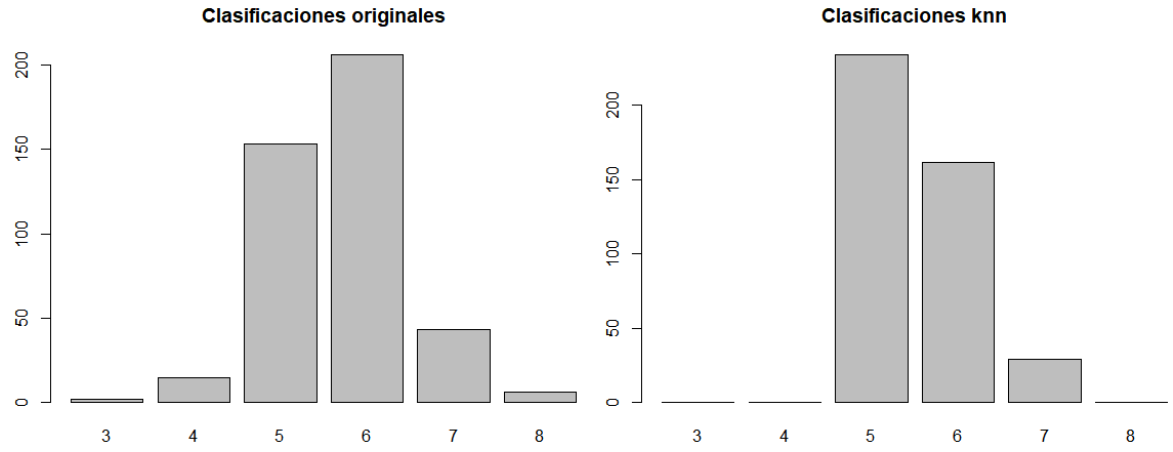
5. Representación de los resultados a partir de tablas y gráficas

Para representar los resultados de los modelos generados, primero veremos la fiabilidad de los mismos y hacia dónde se desvía el error.

Empezamos, como en el punto anterior, por el modelo generado a partir del dataset tratado. Comparamos la predicción del k-nn con el conjunto de test (y_test) factorizándolo y obtenemos los siguientes resultados:

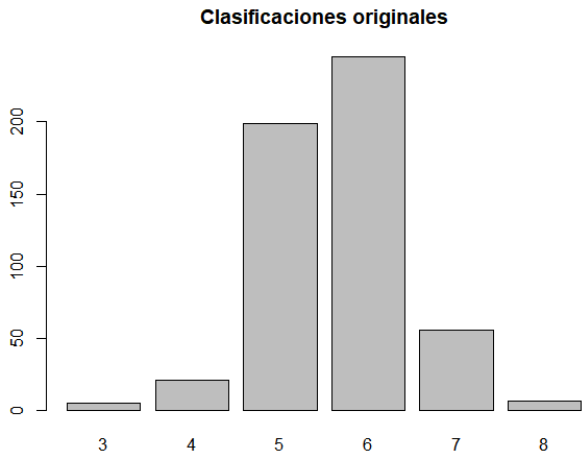
```
> summary(knn)
  3   4   5   6   7   8
  0   0 234 162  29   0

> summary(y_test_f)
  3   4   5   6   7   8
  2  15 153 206  43   6
```

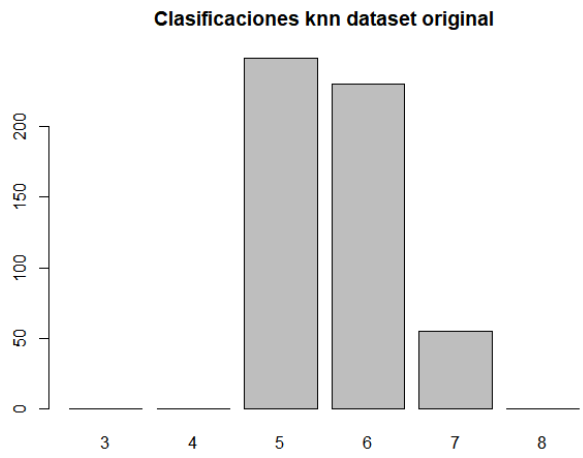
Vemos que el k-nn ha generado una clasificación con los mismos valores que el conjuntod de test, pero las predicciones se equivocan en los valores céntricos. En la predicción del k-nn hay muchos más vinos clasificados como calidad 5 que el original y en cambio el 6 tiene menos:



Ahora comparamos con los resultados de la clasificación obtenida con el conjunto de datos original:

```
> summary(knn_orig)
  3   4   5   6   7   8
  0   0 248 230  55   0
> summary(yy_test_f)
  3   4   5   6   7   8
  5  21 199 245  56   7
```

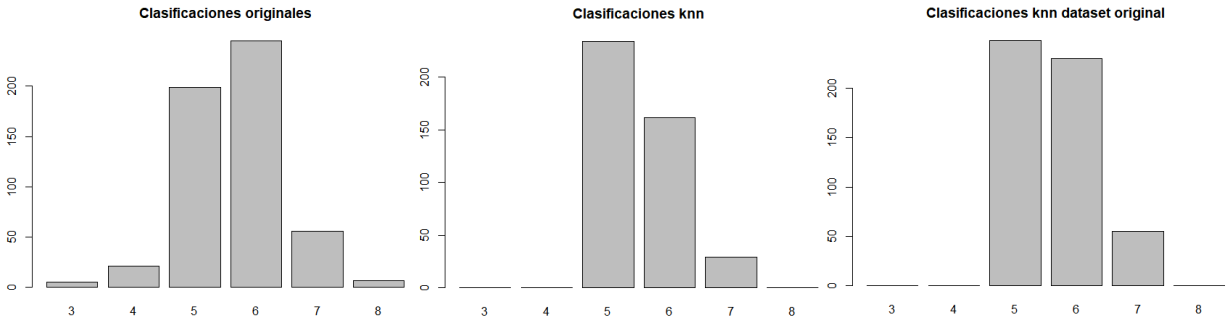




Con el conjunto de datos roginal tenemos un resultado similar aunque ahora la calidad tipo 6 es más correcta. Aun así, siguen habiendo valores tipo 4 que se engloban como un tipo 5.

Comparando los resultados tenemos que en el conjunto de datos tratado la precisión de la clasificación es un 1.13% mayor, pero en cambio la clasificación de las calidades intermedias (valores 5 y 6) es muy imprecisa.

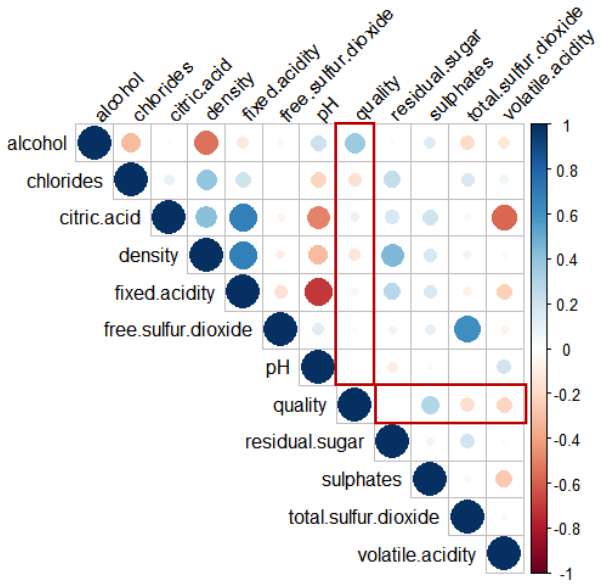
Con el dataset original, la precisión del modelo es menor pero las calidades intermedias respetan más la distribución.



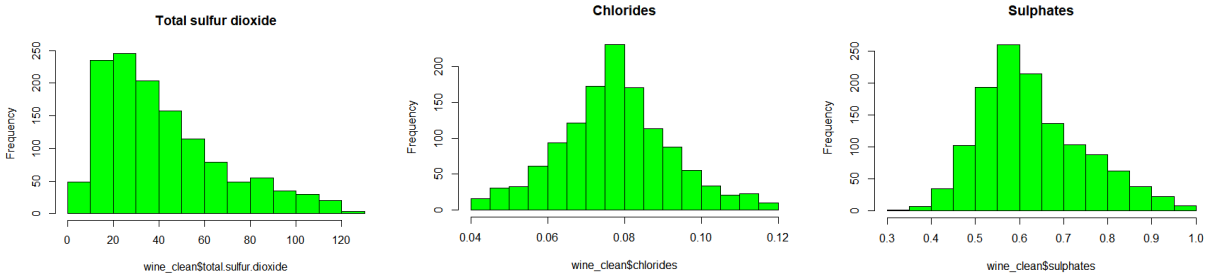
Al ver estos resultados, se nos plantea la duda: ¿qué atributos del vino influyen más en las calidades 5 y 6? Parece que éstas son las más conflictivas de clasificar con precisión, así que generaremos una matriz de correlación sólo con estos valores para el dataset original:

```
> q<-c(5,6)
> wine_q <- wine_clean[which(wine_clean$quality %in% q),]
> cor_matrix_q <- cor(wine_q)
> corrplot(cor_matrix_q, type = "upper", order = "alphabet", tl.col = "black", tl.srt = 45)
```

Obetenemos la siguiete matriz de correlación, la cual nos indica que los atributos más relacionados con la calidad (cuando ésta es 5 y 6) son alcohol, sulphates, chlorides, total.sulfur.dioxide, volatile.acidity.



Si recordamos los boxplots y los histogramas de los valores outliers generados en puntos anteriores, vemos que efectivamente los atributos que influyen en la calidad son algunos de aquellos que tratamos para eliminar los outliers.



Por esto podemos ver que eliminar los outliers del dataset nos ayuda a conseguir más precisión en el modelo de clasificación, pero en cambio produce más imprecisión en las clasificaciones de valores medios (5 y 6).

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

Hemos obtenido un modelo clasificatorio basándonos en el dataset tratado con la reducción de dimensionalidad y eliminación de valores outliers, con el cual hemos conseguido más precisión a la hora de clasificar los vinos que el dataset original.

Aun así, en valores intermedios de calidad hemos visto que la eliminación de valores outliers en atributos que tienen una correlación fuerte con el valor de salida (quality) produce una incerteza mayor a la hora de clasificar valores intermedios de calidad (5 y 6).

Por tanto, si, los resultados del análisis y el tratamiento de los datos permiten resolver el problema, aunque nos hemos dado cuenta de que también tienen efectos colaterales en el sesgo de los resultados. Con lo cual, hay outliers que deben ser eliminados de la muestra, pero hay que tener cuidado puesto que la eliminación masiva de valores extremos puede producir efectos adversos en modelos de predicción.

Los resultados del análisis son, por un lado, el dataset con valores originales: los atributos únicamente escalados, el valor de salida quality y la predicción del modelo k-nn generado ("winequality-red-out.csv").

Y, por otro lado, el dataset tratado: eliminación de algunos valores outliers y reducción de la dimensionalidad aplicando el método PCA. Tiene como columnas los componentes principales PC1 a PC6 utilizados para generar el modelo de datos, el output (quality) y la predicción de salida del modelo k-nn ("winequality-red-clean-out.csv").

Hay que tener en cuenta que ambos ficheros de salida tendrán únicamente el conjunto de datos de test, ya que la predicción se realiza sobre estos datos (no sobre el conjunto de entrenamiento). También como fichero de salida se proporciona el dataset entero (conjuntos de entrenamiento y test) tratado con la eliminación de outliers y la reducción de dimensionalidad. Tendrá por tanto todos los datos y como columnas los componentes principales PC1 a PC6 y el valor de salida quality. El fichero es "winequality-red-clean.csv"

7. Código

El código R utilizado para esta práctica se encuentra en el repositorio con el siguiente nombre:
RWQ_code.R