Universitat Oberta de Catalunya

uoc.edu

PRAC 2

M2.851 - Tipología y ciclo de vida de los datos

Autor: María del Mar Colino García

14/05/2019

Contenido

1 Objetivo.

Este documento aborda la resolución de la segunda practica dentro de las pruebas de evaluación continua, PRAC2, de la asignatura Tipología y ciclo de vida de los datos, para el segundo semestre del curso 2018-2019.

2 Enunciado

2.1 Presentación

En esta práctica se elabora un caso práctico orientado a aprender a identificar los datos relevantes para un proyecto analítico y usar las herramientas de integración, limpieza, validación y análisis de las mismas. Para hacer esta práctica tendréis que trabajar en grupos de 2 personas.

Tendréis que entregar un solo archivo con el enlace Github (https://github.com) donde se encuentren las soluciones incluyendo los nombres de los componentes del equipo. Podéis utilizar la Wiki de Github para describir vuestro equipo y los diferentes archivos que corresponden a vuestra entrega. Cada miembro del equipo tendrá que contribuir con su usuario Github. Podéis utilizar estos ejemplos como guía:

- Ejemplo: https://github.com/Bengis/nba-gap-cleaning
- Ejemplo complejo (archivo adjunto).

2.2 Competencias

En esta práctica se desarrollan las siguientes competencias del Máster de Data Science:

- Capacidad de analizar un problema en el nivel de abstracción adecuado a cada situación y aplicar las habilidades y conocimientos adquiridos para abordarlo y resolverlo.
- Capacidad para aplicar las técnicas específicas de tratamiento de datos (integración, transformación, limpieza y validación) para su posterior análisis.

2.3 Objetivos

Los objetivos concretos de esta práctica son:

- Aprender a aplicar los conocimientos adquiridos y su capacidad de resolución de problemas en entornos nuevos o poco conocidos dentro de contextos más amplios o multidisciplinares.
- Saber identificar los datos relevantes y los tratamientos necesarios (integración, limpieza y validación) para llevar a cabo un proyecto analítico.
- Aprender a analizar los datos adecuadamente para abordar la información contenida en los datos.
- Identificar la mejor representación de los resultados para aportar conclusiones sobre el problema planteado en el proceso analítico.
- Actuar con los principios éticos y legales relacionados con la manipulación de datos en función del ámbito de aplicación.
- Desarrollar las habilidades de aprendizaje que les permitan continuar estudiando de un modo que tendrá que ser en gran medida autodirigido o autónomo.
- Desarrollar la capacidad de búsqueda, gestión y uso de información y recursos en el ámbito de la ciencia de datos.

2.4 Descripción de la Práctica a realizar

El objetivo de esta actividad será el tratamiento de un dataset, que puede ser el creado en la practica 1 o bien cualquier dataset libre disponible en Kaggle (https://www.kaggle.com).
Algunos ejemplos de dataset con los que podéis trabajar son:

- Red Wine Quality (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009)
- Titanic: Machine Learning from Disaster (https://www.kaggle.com/c/titanic)

El último ejemplo corresponde a una competición activa de Kaggle de manera que, opcionalmente, podéis aprovechar el trabajo realizado durante la práctica para entrar en esta competición.

Siguiendo las principales etapas de un proyecto analítico, las diferentes tareas a realizar (**y justificar**) son las siguientes:

1. Descripción del dataset. ¿Por qué es importante y que pregunta/problema pretende responder?
2. Integración y selección de los datos de interés a analizar.
3. Limpieza de los datos.
   3.1. ¿Los datos contienen ceros o elementos vacíos? ¿Como gestionarías cada uno

4. de estos casos?
   4.1. Identificación y tratamiento de valores extremos.
5. Análisis de los datos.
   5.1. Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).
   5.2. Comprobación de la normalidad y homogeneidad de la varianza.
   5.3. 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.
6. Representación de los resultados a partir de tablas y graficas.
7. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?
8. 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.

2.5 Recursos

Los siguientes recursos son de utilidad para la realizacion de la practica:

- Calvo M., Subirats L., Perez D. (2019). Introduccion a la limpieza y analisis de los datos. Editorial UOC.
- Megan Squire (2015). Clean Data. Packt Publishing Ltd.
- Jiawei Han, Micheine Kamber, Jian Pei (2012). Data mining: concepts and techniques. Morgan Kaufmann.
- Jason W. Osborne (2010). Data Cleaning Basics: Best Practices in Dealing with Extreme Scores. Newborn and Infant Nursing Reviews; 10 (1): pp. 1527-3369.
- Peter Dalgaard (2008). Introductory statistics with R. Springer Science & Business Media.
- Wes McKinney (2012). Python for Data Analysis. O' Reilley Media, Inc.
- Tutorial de Github https://guides.github.com/activities/hello-world

2.6 Criterios de valoración

Todos los apartados son obligatorios. La ponderación de los ejercicios es la siguiente:

- Los apartados 1, 2 y 6 valen 0,5 puntos.
- Los apartados 3, 5 y 7 valen 2 puntos.
- El apartado 4 vale 2,5 puntos.

Se valorará la idoneidad de las respuestas, que deberán ser claras y completas. Las diferentes etapas deberán justificarse y acompañarse del código correspondiente. También se valorara la síntesis y claridad, a través del uso de comentarios, del código resultante, así como la calidad de los datos finales analizados.

2.7 Formato y fecha de entrega

Durante la semana del 27 de mayo el grupo podrá entregar al profesor una entrega parcial opcional. Esta entrega parcial es muy recomendable para recibir asesoramiento sobre la práctica y verificar que la dirección tomada es la correcta. Se entregarán comentarios a los estudiantes que hayan efectuado la entrega parcial pero no contara para la nota de la práctica. En la entrega parcial los estudiantes deberán entregar por correo electrónico (mcalvogonza@uoc.edu) el enlace al repositorio Github con el que hayan avanzado.

En referente a la entrega final, hay que entregar un único fichero que contenga el enlace Github donde haya:

1. Una Wiki con los nombres de los componentes del grupo y una descripcion de los ficheros.
2. Un documento PDF con las respuestas a las preguntas y los nombres de los componentes del grupo. Además, al final del documento, deberá aparecer la siguiente tabla de contribuciones al trabajo, la cual debe firmar cada integrante del grupo con sus iniciales.
   Las iniciales representan la confirmación de que el integrante ha participado en dicho apartado. Todos los integrantes deben participar en cada apartado, por lo que, idealmente, los apartados deberían estar firmados por todos los integrantes.

Contribuciones	Firma
Investigación previa	Integrante 1, Integrante 2, ...
Redacción de las respuestas	Integrante 1, Integrante 2, ...
Desarrollo código	Integrante 1, Integrante 2, ...

3. Una carpeta con el código generado para analizar los datos.
4. El fichero CSV con los datos originales.
5. El fichero CSV con los datos finales analizados.

Este documento de entrega final de la Practica 2 se debe entregar en el espacio de Entrega y Registro de AC del aula antes de las **23:59** del día **11 de junio**. No se aceptarán entregas fuera de plazo.

3 Solución de la Práctica

3.1 Definición del problema a resolver - *problem statement*

A partir del conjunto de datos sobre vinos se plantea realizar un análisis y estudio sobre las características de los vinos con el fin de entender cómo influyen sobre su calidad. Disponer de este tipo de información es de gran utilidad a cualquier productor de vino que desee entender las características que más afectan a la calidad del vino, y como afectan, de cara a establecer estrategias para mejorar su producción.

Adicionalmente, a partir de la información disponible en este conjunto de datos podrán construirse modelos de regresión y/o clasificación que permitan predecir la calidad de nuevas observaciones de vinos en base a sus características. También será viable llevar a cabo contrastes de hipótesis para ayudar a identificar propiedades que pueden resultar interesantes en las muestras y que puedan ser inferidas con respecto a la población.

3.2 Selección, recuperación y persitencia de los datos – *data collection and storage*

El conjunto de datos escogido para llevar a cabo la ejecución de esta práctica se llama **Wine Quality** y podemos encontrarlo en el repositorio UCI: https://archive.ics.uci.edu/ml/datasets/wine+quality, es uno de los propuestos en enunciado de la práctica.

Este conjunto de datos está compuesto por dos *datasets*, disponiéndose de un total de 6497 observaciones con 12 atributos cada una. La información incluida en estas muestras describe una serie de características asociadas a vinos (tinto y blanco) y la calidad que le ha sido asignada.

El conjunto de atributos que forman parte del dataset son:

1. **fixed acidity**: acidez fija, la mayoría de los ácidos relacionados con el vino, o fijos, o no volátiles (no se evaporan fácilmente) (tartaric acid - g / l)
2. **volatile acidity**: acidez fija, la cantidad de ácido acético en el vino, que a niveles demasiado altos puede provocar un sabor desagradable a vinagre (acetic acid - g / l)
3. **citric acid**: ácido cítrico, se encuentra en pequeñas cantidades, el ácido cítrico puede agregar "frescura" y sabor a los vinos (g / l)
4. **residual sugar**: azucar resiual, la cantidad de azúcar restante después de que se detenga la fermentación (g / l)
5. **chlorides**: cloruros, la cantidad de sal en el vino (sodium chloride - g / l
6. **free sulfur dioxide**: dióxido de azufre libre, la forma libre de SO2 existe en equilibrio entre el SO2 molecular (como un gas disuelto) y el ión bisulfito (mg / l)
7. **total sulfur dioxide**: dióxido de azufre total, cantidad de formas libres y unidas de S02 (mg / l)
8. **density**: densidad, la densidad del agua es cercana a la del agua según el porcentaje de alcohol y contenido de azúcar (g / l)
9. **pH**: describe qué tan ácido o básico es un vino en una escala de 0 (muy ácido) a 14 (muy básico)
10. **sulphates**: sultafos, un aditivo de vino que puede contribuir a los niveles de gas de dióxido de azufre (S02) (potassium sulphate - g / l)
11. **alcohol**: el porcentaje de contenido de alcohol del vino (% por volumen)
12. **quality**: calidad asignada al vino, puntuación de 0 a 10

Los autores y propietarios del dataset son (P. Cortez, 2009)

3.3 Pre-procesado de los datos – *data preprocesing*

Con el objetivo de mejorar la calidad y robustez de los análisis que van a ser realizados sobre los datos, comenzaremos por realizar las tareas de limpieza del conjunto de datos de partida.

3.3.1 Integración

Antes de nada, se procederá a instalar y configurar las dependencias que van a ser usadas a lo largo del proceso analítico.

```
# importar dependencias requeridas
if(!require(Rcpp)){
  install.packages('Rcpp', repos='http://cran.us.r-project.org')
  library('Rcpp')
}
if(!require(colorspace)){
  install.packages('colorspace', repos='http://cran.us.r-project.org')
  library('colorspace')
}
if(!require(RColorBrewer)){
  install.packages('RColorBrewer', repos='http://cran.us.r-project.org')
  library('RColorBrewer')
}
if(!require(ggplot2)){
  install.packages('ggplot2', repos='http://cran.us.r-project.org')
  library('ggplot2')
}
if(!require(GGally)){
  install.packages('GGally', repos='http://cran.us.r-project.org')
  library('GGally')
}
if(!require(knitr)){
  install.packages('knitr', repos='http://cran.us.r-project.org')
  library('knitr')
}
if(!require(VIM)){
  install.packages('VIM', repos='http://cran.us.r-project.org')
  library('VIM')
}
if(!require(missForest)){
  install.packages('missForest', repos='http://cran.us.r-project.org')
  library('missForest')
}
if(!require(dplyr)){
  install.packages('dplyr', repos='http://cran.us.r-project.org')
  library(dplyr)
}
if(!require(caret)){
  install.packages('caret', repos='http://cran.us.r-project.org')
  library(caret)
}
if(!require(plyr)){
  install.packages('plyr', repos='http://cran.us.r-project.org')
  library(plyr)
}
if(!require(gridExtra)){
    install.packages('gridExtra', repos='http://cran.us.r-project.org')
    library(gridExtra)
}
if(!require(nortest)){
    install.packages('nortest', repos='http://cran.us.r-project.org')
    library(nortest)
}
if(!require(corrplot)){
  install.packages('corrplot', repos='http://cran.us.r-project.org')
  library('corrplot')
}
if(!require(caTools)){
    install.packages('caTools', repos='http://cran.us.r-project.org')
    library(caTools)
}
if(!require(rpart)){
  install.packages('rpart', repos='http://cran.us.r-project.org')
  library(rpart)
}
if(!require(rpart.plot)){
  install.packages('rpart.plot', repos='http://cran.us.r-project.org')
  library(rpart.plot)
}
```

Antes de proceder a las tareas de integración, será necesario realizar la carga de los datos. Como se ha indicado en el apartado 3.2 SELECCIÓN, RECUPERACIÓN Y PERSITENCIA DE LOS DATOS – DATA COLLECTION AND STORAGE, contamos con dos *datasets* de partida, uno para vinos tintos y otro para vinos blancos, en principio vamos a fusionar los datos de ambos *datasets* añadiendo una nueva propiedad para indicar el tipo del vino al que pertenece cada observación. De esta forma contaremos con un juego de datos lo más completo posible que nos permita realizar análisis considerando todos los ámbitos asociados a una observación.

```
# Cargamos el dataset para vinos tintos
wine_quality_red = read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv", header=TRUE, sep = ';')
# Cargamos el dataset para vinos tintos
wine_quality_white = read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv", header=TRUE, sep = ';')
```

Previo a realizar la integración de los datos, echamos un vistazo a los datos cargados, para asegurarnos que está todo en orden.

En primer lugar, visualizamos el contenido:

```
# Echamos un vistazo al dataset
head(wine_quality_red)
```

```
##   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
```

Y a continuación la estructura:

```
str(wine_quality_red)
```

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

Contamos con 1599 observaciones de vinos tintos.

Y ahora al dataset para vino blanco:

```
# Echamos un vistazo al dataset
head(wine_quality_white)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
## 4           7.2             0.23        0.32            8.5     0.058
## 5           7.2             0.23        0.32            8.5     0.058
## 6           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
## 4                  47                  186  0.9956 3.19      0.40     9.9
## 5                  47                  186  0.9956 3.19      0.40     9.9
## 6                  30                   97  0.9951 3.26      0.44    10.1
##   quality
## 1       6
## 2       6
## 3       6
## 4       6
## 5       6
## 6       6
```

```
str(wine_quality_white)
```

```
## 'data.frame':    4898 obs. of  12 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
```

Contamos con 4898 observaciones de vinos blancos.

Parece que está todo en orden, los datos tienen el formato esperado, por lo que podemos proceder a su integración:

```
# Asignamos etiqueta
wine_quality_red$type = 'red'
wine_quality_white$type = 'white'
# Creamos un único dataset a partir de los dataset para vino tinto y blanco
wine_quality.base = rbind(wine_quality_red, wine_quality_white)
# Establecemos el valor de la etiqueta como factor
wine_quality.base$type = as.factor(wine_quality.base$type)
```

Una vez realizada la integración, verificamos que el resultado es el esperado:

```
# Mostramos la estructura del dataset resultante
```

```
str(wine_quality.base)
```

```
## 'data.frame':    6497 obs. of  13 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 ...
##  $ type                : Factor w/ 2 levels "red","white": 1 1 1 1 1 1 1 1 1 1 ...
```

El resultado es el esperado, obtenemos un nuevo dataset con 6497 observaciones y 13 variables.

Aunque es muy probable que las características de los vinos tintos difieran de las características de los vinos blancos, al menos en parte, el incluir dentro de un único dataset el conjunto completo de observaciones de vino, tinto y blanco, nos permitirá tener un acceso centralizado en cualquier momento a todo el conjunto de datos de partida.

En cualquier caso, con el objetivo de entender las características de los datos por tipo de vino, y poder adquirir una visión general de los datos que nos permita tener un poco más de criterio a la hora de ejecutar los siguientes pasos del proceso, vamos a hacer una mínima exploración (o *screening)* de los datos.

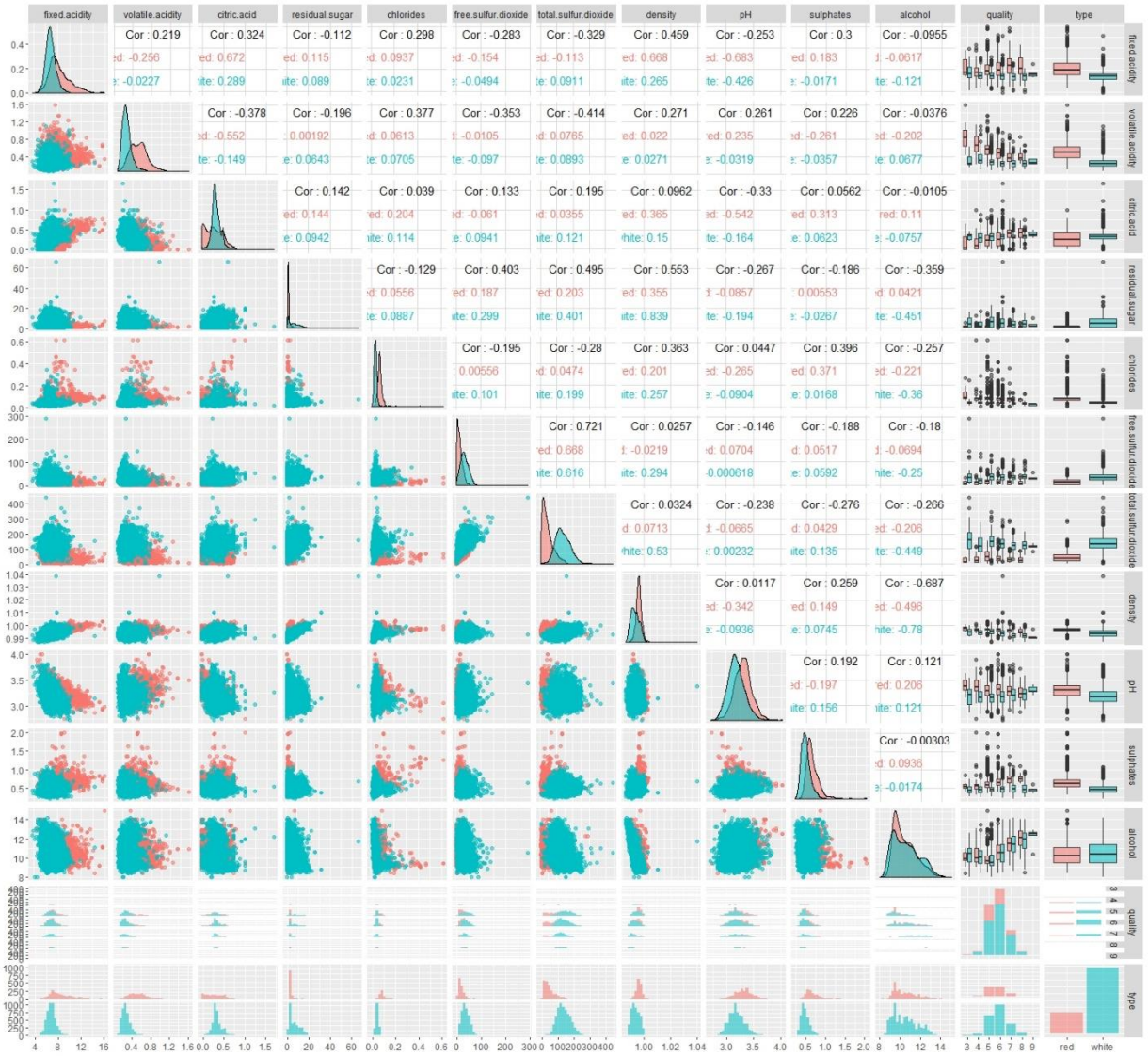
En primer lugar, visualizamos algunos estadísticos descriptivos:

```
summary(wine_quality.base)
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.2500   1st Qu.: 1.800
##  Median : 7.000   Median :0.2900   Median :0.3100   Median : 3.000
##  Mean   : 7.215   Mean   :0.3397   Mean   :0.3186   Mean   : 5.443
##  3rd Qu.: 7.700   3rd Qu.:0.4000   3rd Qu.:0.3900   3rd Qu.: 8.100
##  Max.   :15.900   Max.   :1.5800   Max.   :1.6600   Max.   :65.800
##
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide
##  Min.   :0.00900   Min.   :  1.00      Min.   :  6.0
##  1st Qu.:0.03800   1st Qu.: 17.00      1st Qu.: 77.0
##  Median :0.04700   Median : 29.00      Median :118.0
##  Mean   :0.05603   Mean   : 30.53      Mean   :115.7
##  3rd Qu.:0.06500   3rd Qu.: 41.00      3rd Qu.:156.0
##  Max.   :0.61100   Max.   :289.00      Max.   :440.0
##
##     density             pH          sulphates         alcohol
##  Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00
##  1st Qu.:0.9923   1st Qu.:3.110   1st Qu.:0.4300   1st Qu.: 9.50
##  Median :0.9949   Median :3.210   Median :0.5100   Median :10.30
##  Mean   :0.9947   Mean   :3.219   Mean   :0.5313   Mean   :10.49
##  3rd Qu.:0.9970   3rd Qu.:3.320   3rd Qu.:0.6000   3rd Qu.:11.30
##  Max.   :1.0390   Max.   :4.010   Max.   :2.0000   Max.   :14.90
##
##  quality     type
##  3:  30   red  :1599
##  4: 216   white:4898
##  5:2138
##  6:2836
##  7:1079
##  8: 193
##  9:   5
```

A continuación, vamos a hacer uso de un diagrama que nos permite visualizar gran cantidad de información sobre la distribución de los datos, aprovecharemos para agruparlo por tipo de vino, para confirmar nuestra sospecha sobre que pueden existir diferencias en como se distribuyen las características en función del tipo de vino.

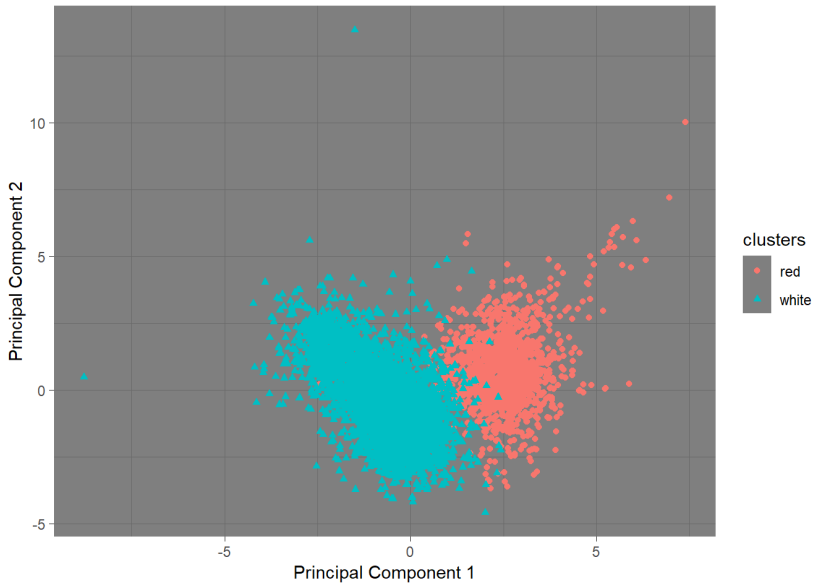
```
library(ggplot2)
library(GGally)
ggpairs(wine_quality.base, aes(colour = type, alpha = 0.4))
```



Como se observa, en un mismo gráfico, y agrupado por tipo de vino, disponemos de gráficos por pares de características, *boxplots*, diagramas de densidad, algo que nos facilita enormemente tener una visión preliminar. Al observar el gráfico, rápidamente podemos confirmar la existencia de una diferencia en el comportamiento de las características en base al tipo de vino, además, de este gráfico también pueden observarse relaciones entre cada característica y la calidad resultante, en función del tipo de vino.

Finalmente, vamos a usar **PCA** (*Principal Component Analisys*) para obtener una representación multidimensional simplificada de los datos, que puede ser usada para visualízalos. El propósito de PCA es, encontrar la mejor representación, con la menor dimensionalidad, que pueda recoger la mayor parte de la variabilidad del conjunto de datos.

```
## Realiza un plot PCA de los datos clusterizados
plotClusterPCA = function(dataset.stand, dataset.clusters, attCluster, shape="clusters", scaled=TRUE)
{
  # si no se suministra el data set estandarizado se estandariza
  if(!scaled) {
    dataset.stand = as.data.frame(scale(dataset.stand))
  }
  ### Construcción componentes principales
  prin_comp <- princomp(dataset.stand)
  nComp = 2
  ### Project cluster on 2 principal componentes
  project = predict(prin comp, newdata=dataset.stand)[,1:nComp]

  ### Crear data frame con los datos usados para realizar plot
  project_data = cbind(as.data.frame(project),
                       cluster = as.factor(dataset.clusters[[attCluster]]))

  ### Construir plot
  ggplot(project_data, aes(x=Comp.1, y=Comp.2)) +
     geom_point(aes(shape=dataset.clusters[[attCluster]], colour = dataset.clusters[[attCluster]])) +
     theme_dark() +
     labs(x = "Principal Component 1", y = "Principal Component 2", shape = shape, colour = shape)
}
## Plot PCA para el conjunto completo de vinos (se excluyen los atributos tipo y la calidad)
wine_quality = wine_quality.base[1:11]
plotClusterPCA(wine_quality, wine_quality.base, "type", scaled = FALSE)
```



Como se observa, visualmente se identifican bastante bien como las observaciones asociadas a vinos tintos y blancos, pueden ser agrupadas en distintos segmentos.

3.3.2 Tratamiento de datos perdidos y valores extremos

3.3.2.1 Datos perdidos

Ahora se llevará a cabo una revisión sobre los valores de los distintos atributos con el fin de identificar valores cero, elementos vacíos y nulos.

- En ocasiones un valor cero puede estar siendo usado para indicar la ausencia de valor, en general cuando asociado a un atributo numérico detectemos un valor fuera del rango posible de valores, este valor puede estar identificando la ausencia de valor para dicho atributo, o también puede tratarse de un error.
- De igual forma, un valor vacío puede tratarse de un valor válido para un atributo de tipo String, cuando dicho atributo sea opcional, o puede estar identificando la ausencia de valor.

- En ocasiones, también será posible encontrarnos otro tipo de valores que identifiquen valores no definidos o nulos, por ejemplo, es habitual encontrar valores como "NA" (*Not Availabe*) identificando a un valor nulo, esto es un valor para el que no se cuenta con información.

En cualquier caso, sea cual sea la casuística, ha de ser identificada y tratada de la forma adecuada.

Comenzamos por estudiar los atributos con valor cero:

```
# Determinar el número de valores cero en el dataset
length(which(wine_quality.base==0))
```

```
## [1] 151
```

Se encuentran 151 valores cero en el dataset, vamos a identificar cuáles son:

```
# Identificar datos con valor cero
colSums(wine_quality.base==0)
```

```
##         fixed.acidity      volatile.acidity           citric.acid
##                     0                     0                   151
##        residual.sugar             chlorides   free.sulfur.dioxide
##                     0                     0                     0
## total.sulfur.dioxide               density                    pH
##                     0                     0                     0
##             sulphates               alcohol               quality
##                     0                     0                     0
##                  type
##                     0
```

Ahora con valor vacío:

```
# Determinar el número de valores vacíos en el dataset
length(which(wine_quality.base==""))
```

```
## [1] 0
```

Y finalizamos con los atributos con valor nulo:

```
# Determinar el número de datos nulos en el dataset
length(which(is.na(wine_quality.base)))
```

```
## [1] 0
```

Por lo tanto, se han localizado 151 valores cero asociados a la características *citric.acid*. Respecto al resto de atributos, no se detecta ningún otro valor cero, nulo o vacío.

Ahora procede determinar cómo se va a actuar frente este hallazgo, y esto depende de como sea interpretado el valor '0' asociado al ácido cítrico. Investigando sobre los valores aceptables para dicho ácido encontramos que el ácido cítrico asume valores pequeños y que son habituales valores desde 0,1 – 1 g/litro.

Por lo que hay que decidir:

- sí 0 se asume como un valor fuera de dicho rango, y que ha sido introducido intencionadamente para identificar un *missing value*,
- se trata de un valor erróneo,
- o es realmente un valor válido para dicha propiedad.

En este caso, dado el carácter formativo del análisis, y dado que no tenemos un experto al que consultar si un valor 0 se puede considerar fuera del rango, vamos a optar por asumir que son valores perdidos y escoger una de las posibles estrategias de resolución.

Pero antes de proceder a realizar modificaciones sobre el dataset, para tener una visión global de los posibles problemas en los datos, se realizará el análisis de los valores extremos.

3.3.2.2 Valores extremos

Un valor extremo (u *outlier*) dentro de una muestra de una población dada, es una observación que se desvía de forma muy significativa del resto, esto es, una observación que cae fuera de la distribución que se considera normal para esa variable, y que hace sospechar que esa observación podría no pertenecer a la población analizada.

La existencia de valores extremos en un *dataset* pude llegar a afectar de forma negativa a los resultados de los análisis sobre los datos, es por ello por lo que conviene identificarlos, y tratarlos, en caso de que se estime oportuno.

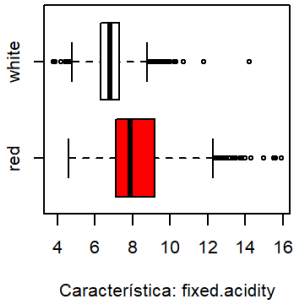
Comenzaremos por su detección, para lo cual podemos utilizar distintas técnicas. Una de las técnicas más simples es la que se basa en reconocer como *outlier* los valores que se alejan más de 3 desviaciones típicas de la media, por ello, aplicando diagramas de *boxplots* podremos identificar muy fácilmente los *outliers*.

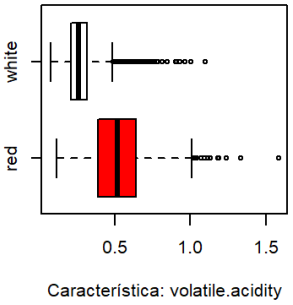
De cara a la detección de los *outliers*, vamos a retomar un momento parte del conocimiento que hemos adquirido hasta el momento.

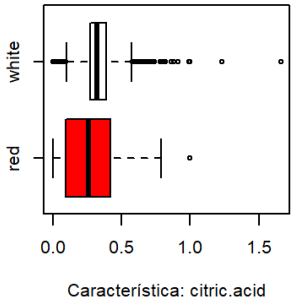
Sabemos que, en base al tipo de vino, tinto o blanco, existen características cuyo comportamiento varía, pudiendo por lo tanto existir variaciones en los valores asociados a una característica para el vino tinto, respecto al vino blanco.

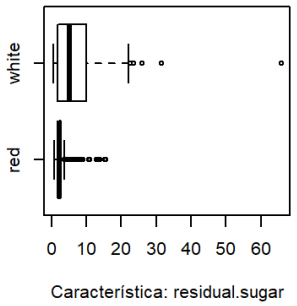
Tomando esto como base, la mejor estrategia para llevar a cabo la detección de *outliers* dadas las características de los datos, es hacer el estudio dividiendo las observaciones del dataset en dos grupos, vino tinto y vino blanco.

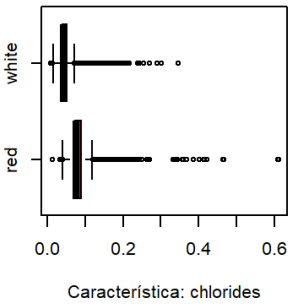
```
# construye los boxplot para el conjunto de columnas suministradas como parametro
plotBoxplotMultiple = function(wine_quality, colnames) {
  par(mfrow=c(2,3))
  for (col in colnames) {
    boxplot(wine_quality[,col]~type,
          data=wine_quality,
          xlab=paste("Característica", sep = ": ",col),
          horizontal=TRUE,
          col=c("red", "white"))
  }
}

# Detección valores extremos haciendo uso de boxplots
# (sectorizando por tipo de vino)
colnames = names(wine_quality.base[1:11]);
plotBoxplotMultiple(wine_quality.base, colnames)
```

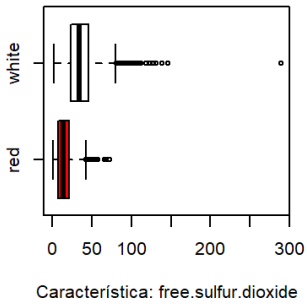


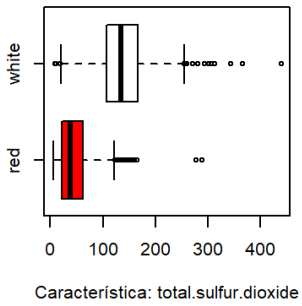


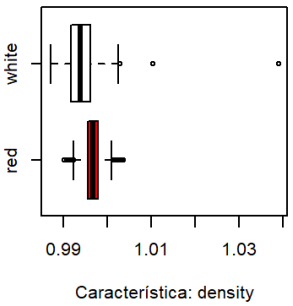


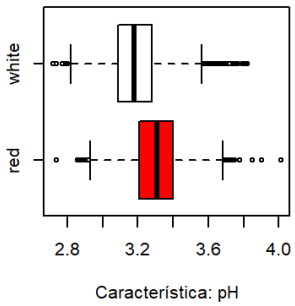


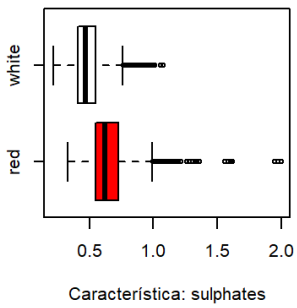


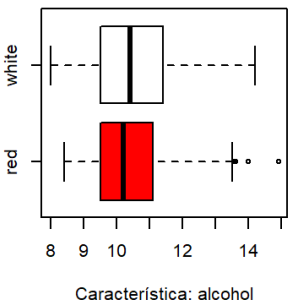












Para tener una visión más detallada de los valores más reseñables involucrados en los diagramas de cajas, obtenemos algunos estadísticos descriptivos haciendo uso de la función *summary*.

Empezamos con el vino tinto:

```
# Obtención de estadísticos descriptivos - para tipos de vino tinto
summary(wine_quality.base[wine_quality.base$type=="red",][1:12])
```

```
##  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
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00
##  Median :0.07900   Median :14.00       Median : 38.00
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00
##     density             pH          sulphates         alcohol
##  Min.   :0.9901   Min.   :2.740   Min.   :0.3300   Min.   : 8.40
##  1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50
##  Median :0.9968   Median :3.310   Median :0.6200   Median :10.20
##  Mean   :0.9967   Mean   :3.311   Mean   :0.6581   Mean   :10.42
##  3rd Qu.:0.9978   3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10
##  Max.   :1.0037   Max.   :4.010   Max.   :2.0000   Max.   :14.90
##     quality
##  Min.   :3.000
##  1st Qu.:5.000
##  Median :6.000
##  Mean   :5.636
##  3rd Qu.:6.000
##  Max.   :8.000
```

Y continuamos con el vino blanco:

```
# Obtención de estadísticos descriptivos - para tipos de vino blanco
summary(wine_quality.base[wine_quality.base$type=="white",][1:12])
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0
##  Median :0.04300   Median : 34.00      Median :134.0
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0
##     density             pH          sulphates         alcohol
##  Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00
##  1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50
##  Median :0.9937   Median :3.180   Median :0.4700   Median :10.40
##  Mean   :0.9940   Mean   :3.188   Mean   :0.4898   Mean   :10.51
##  3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40
##  Max.   :1.0390   Max.   :3.820   Max.   :1.0800   Max.   :14.20
##     quality
##  Min.   :3.000
##  1st Qu.:5.000
##  Median :6.000
##  Mean   :5.878
##  3rd Qu.:6.000
##  Max.   :9.000
```

Además, haciendo uso de toda esta información, podemos obtener una tabla resumen del análisis de los *outliers* aplicables a la distribución de cada característica, en función del tipo de vino:

```
# Definición de función para recuperar los outliers para un tipo de vino y atributo
getOutliers= function (wine_quality, wine_type, col) {
  return(boxplot.stats(wine_quality[wine_quality$type==wine_type,][,col])$out)
}
# Definición de función para extraer un resumen del análisis de outliers
getOutliersTableInfo = function (wine_quality, colnames) {
  # cabecera
  rowHeader = list("type", "att", "left outliers", "max left", "right outliers", "min right")
  # inicialización data.frame con los resultados
  df = data.frame(rowHeader,stringsAsFactors = F)
  # se itera por los atributos
  for (col in colnames) {
    # se itera por los tipos de vino
    for (wine_type in c("red", "white")) {
      # Valores outliers para la distribución de 'fixed.acidity' - vino tinto
      outliers = getOutliers(wine_quality, wine_type, col)
      # Valor Q1 para atributo y tipo de vino
      firstQu= summary(wine_quality[wine_quality$type==wine_type,][,col])[["1st Qu."]]
      # Valor Q3 para atributo y tipo de vino
      thirdQu=summary(wine_quality[wine_quality$type==wine_type,][,col])[["3rd Qu."]]
      # Número de outliers a la izquierda
      leftOutliers = length(outliers[outliers < firstQu])
      # Máximo valor outliers a la izquierda
      if(leftOutliers > 0) { maxLeftOutliers =  max(outliers[outliers < firstQu]) }
      else { maxLeftOutliers = 0 }
      # Número de outliers a la derecha
      rightOutliers = length(outliers[outliers > thirdQu])
      # Mínimo valor outliers a la derecha
      if(rightOutliers > 0) { minRightOutliers = min(outliers[outliers > thirdQu]) }
      else { minRightOutliers = 0 }
      # Crear fila de datos y añadir al data.frame
      row = list(wine_type, col, leftOutliers, maxLeftOutliers, rightOutliers, minRightOutliers);
      df= rbind(df,row)
    }

  }
  return(df)
}
# obetner las columnas sobre las que se desea analizar los outliers
colnames = names(wine_quality.base[1:11])
# invocar el análisis e imprimir el resultado
res = getOutliersTableInfo(wine_quality.base, colnames)
kable(res)
```

type.	att.	left.outliers.	max.left.	right.outliers.	min.right.
red	fixed.acidity	0	0	49	12.4
white	fixed.acidity	14	4.7	105	8.9
red	volatile.acidity	0	0	19	1.02
white	volatile.acidity	0	0	186	0.49
red	citric.acid	0	0	1	1
white	citric.acid	85	0.09	185	0.58
red	residual.sugar	0	0	155	3.7
white	residual.sugar	0	0	7	22.6
red	chlorides	9	0.039	103	0.12
white	chlorides	7	0.014	201	0.072
red	free.sulfur.dioxide	0	0	30	43
white	free.sulfur.dioxide	0	0	50	81

type.	att.	left.outliers.	max.left.	right.outliers.	min.right.
red	total.sulfur.dioxide	0	0	55	124
white	total.sulfur.dioxide	5	19	14	256
red	density	21	0.9922	24	1.0014
white	density	0	0	5	1.00295
red	pH	14	2.92	21	3.69
white	pH	9	2.8	66	3.57
red	sulphates	0	0	59	1
white	sulphates	0	0	124	0.77
red	alcohol	0	0	13	13.5666666666667
white	alcohol	0	0	0	0

En la tabla resultante, para cada tipo de vino y propiedad podemos encontrar el número de *outliers* a la izquierda y a la derecha de la distribución, así como el valor máximo de los *outliers* a la izquierda, y el mínimo de los *outliers* a la derecha, algo que facilita muchísimo comprender las características de los valores extremos existentes.

A partir de toda esta información podremos analizar los datos de *outliers* encontrados, característica por característica del vino, por ejemplo, si nos fijamos en la acidez fija del vino (***fixed.acidity***), podemos observar que:

- para vinos **tintos**, el 50% de la población se encuetra entre los valores 7,10 y 9,20 de acided fija, hallándose un total de 49 *outliers* a la derecha, donde el valor mínimo de este conjunto de *outliers* es 12.4.
- para vinos **blancos**, el 50% de la población se encuentra entre los valores 6,30 y 7,30, hallándose un total de 14 outliers a la izquierda, con un valor máximo de este grupo de outliers de 4,7; así como encontrándose un total de 105 outliers a la derecha, con un valor mínimo de este otro grupo de outliers de 8,9.

Y si además estamos interesados en obtener la lista completa de *outliers*, podremos hacerlo de la siguiente forma:

```
# Outliers para la propiedad 'fixed.acidity' - tipo de vino tinto
print(getOutliers(wine_quality, "red", "fixed.acidity"))
```

```
##  [1] 12.8 12.8 15.0 15.0 12.5 13.3 13.4 12.4 12.5 13.8 13.5 12.6 12.5 12.8
## [15] 12.8 14.0 13.7 13.7 12.7 12.5 12.8 12.6 15.6 12.5 13.0 12.5 13.3 12.4
## [29] 12.5 12.9 14.3 12.4 15.5 15.5 15.6 13.0 12.7 13.0 12.7 12.4 12.7 13.2
## [43] 13.2 13.2 15.9 13.3 12.9 12.6 12.6
```

```
# Outliers para la propiedad 'fixed.acidity' - tipo de vino blanco
print(getOutliers(wine_quality, "white", "fixed.acidity"))
```

```
##   [1]  9.8  9.8 10.2  9.1 10.0  9.2  9.2  9.0  9.1  9.2 10.3  9.4  9.2  9.8
##  [15]  9.6  9.2  9.0  9.3  9.2  9.1  8.9  9.8  8.9  9.2  9.7  9.4 10.3  9.6
##  [29]  9.0  9.7  9.2  9.4  9.6  9.2  9.0  9.2 10.7 10.7  9.0  9.2  9.8  9.2
##  [43] 14.2  8.9  8.9  9.1  9.1  9.8  9.0  9.3  8.9  9.0  9.0  8.9  9.0  9.3
##  [57]  9.2  9.6  9.4  9.4 10.0  8.9  8.9 10.0  9.2  9.2  9.2  9.9  9.5  9.0
##  [71]  9.0  8.9  9.5 11.8  9.4  9.1  9.8  9.9  9.2  8.9  9.2  9.4  9.4  9.4
```

```
##  [85]  4.6  8.9  9.4  9.2  9.2  9.8  9.0  9.0  9.0  8.9  8.9  4.5  9.2  9.6
##  [99]  4.2  9.7  9.7  9.0  4.2  9.4  8.9  8.9  8.9  4.7  4.7  3.8  4.4  4.7
## [113]  9.0  9.0  4.7  4.4  3.9  4.7  4.4
```

Ahora que contamos con toda esta información, procede determinar cómo se va a actuar frente a los valores *outliers* detectados.

De igual forma que en el apartado de 3.3.2.1 DATOS PERDIDOS, puesto que no se cuenta con un conocimiento especializado del rango de valores apropiado para cada uno de las características, y dado que estamos en un ámbito formativo y el mayor reto es suponer los valores como incorrectos o inapropiados, ya que eso implicará tener que aplicar alguna estrategia para su resolución, se van a interpretar los valores *outliers* encontrados como datos incorrectos.

Las estrategias posibles serán del mismo tipo que las que pueden seguirse para la resolución de datos perdidos, en el próximo apartado, se procederá a aplicar las estrategias apropiadas tanto sobre datos perdidos, como sobre valores *outliers*.

3.3.2.3 Correcciones problemáticas encontradas

Tras realizarse el análisis de datos perdidos y valores extremos, se han localizado un conjunto de valores que han decidido considerarse datos perdidos y/o erróneos, y para los que en este apartado aplicaremos una estrategia de resolución.

En general, las principales vías para afrontar la resolución de datos perdidos y/o erróneos, son:
- eliminar los registros donde se localiza el valor,
- imputar valor según una de las estrategias habituales

En nuestro caso, aun disponiéndose de un número suficientemente alto de observaciones, vamos a suponer que no deseamos perder información, y que se prefiere asignar un valor tratando de que ese valor se asemeje el máximo a un posible valor real no considerado como erróneo. En este caso, se puede utilizar una técnica de imputación que haga uso de la mayor cantidad de información para predecir los valores perdidos, ***missForest*** es uno de los métodos más robustos que cumple estos requisitos, y que además goza de amplia popularidad, por lo que aplicaremos este método.

En primer lugar, vamos a asignar valor 'NA' a todos los datos que estamos considerando como perdidos y/o erróneos, para posteriormente pasar a aplicar un método de imputación de valores perdidos.

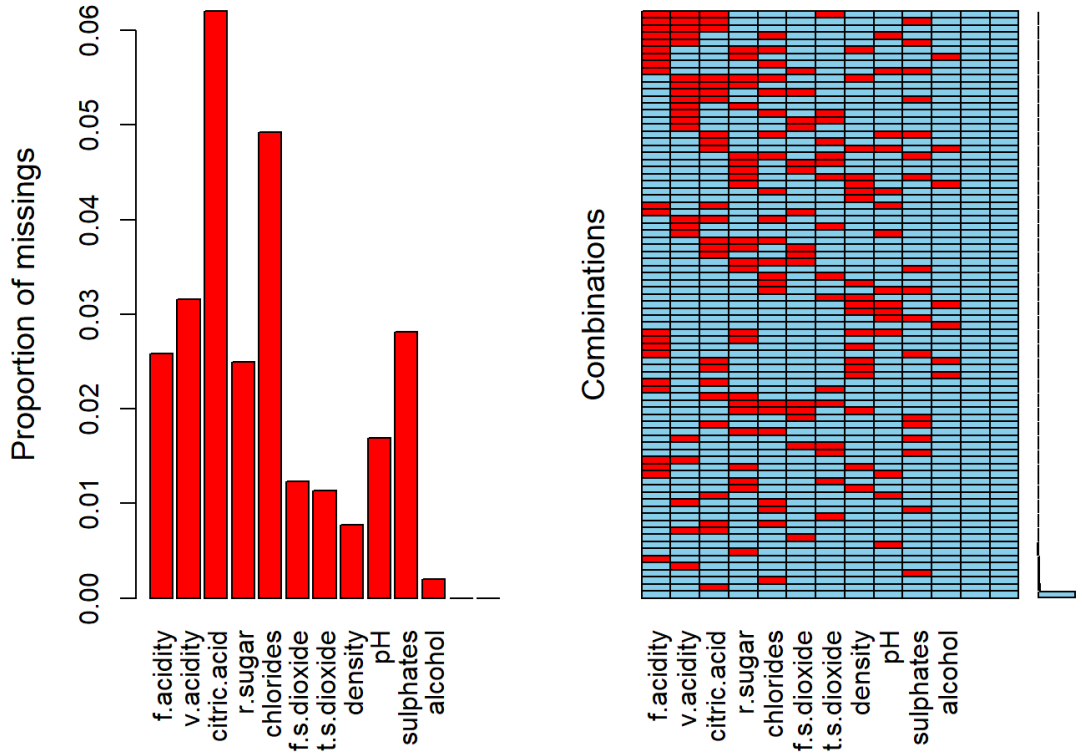
```
# hacemos una copia del data frame base sobre la que vamos a realizar la limpieza
wine_quality.clean = data.frame(wine_quality.base)

# asignamos valor NA
# - a los valores 0 del atributo 'citric.acid'
wine_quality.clean[wine_quality.base$citric.acid==0,]$citric.acid = NA

# - a los valores outliers para 'fixed.acidity' y tipo de vino 'red'
wine_quality.clean[wine_quality.clean$type=="red" &
                        wine quality.clean$fixed.acidity>=12.4,]$fixed.acidity = NA
# - a los valores outliers para 'fixed.acidity' y tipo de vino 'white'
wine_quality.clean[wine_quality.clean$type=="white" &
                          wine_quality.clean$fixed.acidity<=4.7,]$fixed.acidity = NA
wine_quality.clean[wine_quality.clean$type=="white" &
                          !is.na(wine_quality.clean$fixed.acidity) &
wine quality.clean$fixed.acidity>=8.9,]$fixed.acidity = NA

# - a los valores outliers para 'volatile.acidity' y tipo de vino 'red'
wine_quality.clean[wine_quality.clean$type=="red" &
                        wine_quality.clean$volatile.acidity>=1.02,]$volatile.acidity = NA
# - a los valores outliers para 'volatile.acidity' y tipo de vino 'white'
wine_quality.clean[wine_quality.clean$type=="white" &
                          wine_quality.clean$volatile.acidity>=0.49,]$volatile.acidity = NA

# - a los valores outliers para 'citric.acid' y tipo de vino 'red'
```

```
wine_quality.clean[wine_quality.clean$type=="red" &
                         !is.na(wine_quality.clean$citric.acid) &
wine_quality.clean$citric.acid>=1,]$citric.acid = NA
# - a los valores outliers para 'citric.acid' y tipo de vino 'white'
wine_quality.clean[wine_quality.clean$type=="white" &
                           !is.na(wine_quality.clean$citric.acid) &
wine_quality.clean$citric.acid<=0.09,]$citric.acid = NA
wine_quality.clean[wine_quality.clean$type=="white" &
                           !is.na(wine_quality.clean$citric.acid) &
wine_quality.clean$citric.acid>=0.58,]$citric.acid = NA

# - a los valores outliers para 'residual.sugar' y tipo de vino 'red'
wine_quality.clean[wine_quality.clean$type=="red" &
                         wine_quality.clean$residual.sugar>=3.7,]$residual.sugar = NA
# - a los valores outliers para 'residual.sugar' y tipo de vino 'white'
wine_quality.clean[wine_quality.clean$type=="white" &
                           wine_quality.clean$residual.sugar>=22.6,]$residual.sugar = NA

# - a los valores outliers para 'chlorides' y tipo de vino 'red'
wine_quality.clean[wine_quality.clean$type=="red" &
                         wine quality.clean$chlorides<=0.039,]$chlorides = NA
wine_quality.clean[wine_quality.clean$type=="red" &
                         !is.na(wine_quality.clean$chlorides) &
wine_quality.clean$chlorides>=0.12,]$chlorides = NA
# - a los valores outliers para 'chlorides' y tipo de vino 'white'
wine quality.clean[wine quality.clean$type=="white" &
                           wine_quality.clean$chlorides<=0.014,]$chlorides = NA
wine_quality.clean[wine_quality.clean$type=="white" &
                           !is.na(wine_quality.clean$chlorides) &
wine_quality.clean$chlorides>=0.072,]$chlorides = NA

# - a los valores outliers para 'free.sulfur.dioxide' y tipo de vino 'red'
wine_quality.clean[wine_quality.clean$type=="red" &
                         wine_quality.clean$free.sulfur.dioxide>=43,]$free.sulfur.dioxide = NA
# - a los valores outliers para 'free.sulfur.dioxide' y tipo de vino 'white'
wine_quality.clean[wine_quality.clean$type=="white" &
                           wine_quality.clean$free.sulfur.dioxide>=81,]$free.sulfur.dioxide = NA

# - a los valores outliers para 'total.sulfur.dioxide' y tipo de vino 'red'
wine_quality.clean[wine_quality.clean$type=="red" &
                         wine_quality.clean$total.sulfur.dioxide>=124,]$total.sulfur.dioxide = NA
# - a los valores outliers para 'total.sulfur.dioxide' y tipo de vino 'white'
wine_quality.clean[wine_quality.clean$type=="white" &
                           wine_quality.clean$total.sulfur.dioxide<=19,]$total.sulfur.dioxide = NA
wine_quality.clean[wine_quality.clean$type=="white" &
                           !is.na(wine_quality.clean$total.sulfur.dioxide) &
wine_quality.clean$total.sulfur.dioxide>=256,]$total.sulfur.dioxide = NA

# - a los valores outliers para 'density' y tipo de vino 'red'
wine_quality.clean[wine_quality.clean$type=="red" &
                         wine_quality.clean$density<=0.9922,]$density = NA
wine_quality.clean[wine_quality.clean$type=="red" &
                         !is.na(wine_quality.clean$density) &
wine_quality.clean$density>=1.0014,]$density = NA
# - a los valores outliers para 'fixed.acidity' y tipo de vino 'white'
wine_quality.clean[wine_quality.clean$type=="white" &
                           wine_quality.clean$density>=1.00295,]$density = NA

# - a los valores outliers para 'pH' y tipo de vino 'red'
wine_quality.clean[wine_quality.clean$type=="red" &
                         wine_quality.clean$pH<=2.92,]$pH = NA
wine_quality.clean[wine_quality.clean$type=="red" &
                         !is.na(wine_quality.clean$pH) & wine_quality.clean$pH>=3.69,]$pH = NA
# - a los valores outliers para 'pH' y tipo de vino 'white'
wine_quality.clean[wine_quality.clean$type=="white" &
                           wine_quality.clean$pH<=2.8,]$pH = NA
wine_quality.clean[wine_quality.clean$type=="white" &
                           !is.na(wine_quality.clean$pH) & wine_quality.clean$pH>=3.57,]$pH = NA

# - a los valores outliers para 'sulphates' y tipo de vino 'red'
wine_quality.clean[wine_quality.clean$type=="red" &
                         wine_quality.clean$sulphates>=1,]$sulphates = NA
# - a los valores outliers para 'sulphates' y tipo de vino 'white'
wine_quality.clean[wine_quality.clean$type=="white" &
                           wine_quality.clean$sulphates>=0.77,]$sulphates = NA

# - a los valores outliers para 'alcohol' y tipo de vino 'red'
```

```
wine_quality.clean[wine_quality.clean$type=="red" &
                         wine_quality.clean$alcohol>=13.5666666666667,]$alcohol = NA
```

Ahora visualizamos los valores perdidos que acabamos de provocar:

```
# renombramos algunos titulos para que puedan aparecer en el gráfico
orinal names = names(wine quality.clean)
names(wine_quality.clean) = c("f.acidity", "v.acidity", "citric.acid", "r.sugar", "chlorides",
"f.s.dioxide", "t.s.dioxide", "density", "pH", "sulphates", "alcohol")
# pintamos gráfico con informacón de missing values
aggr(wine_quality.clean)
```



Observamos como el *ácido cítrico*, la *chlorides* y el *suphates* son tres de las propiedades para la que se han obtenido mayor cantidad de *missing values*.

Y a continuación se procederá a aplicar el método de imputación escogido:

```
# Aplicamos método 'missForest' para imputar valores perdidos
imp = missForest(wine_quality.clean)
# Revisamos valores imputados
head(imp$ximp)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70     0.12750            1.9     0.076
## 2           7.8             0.88     0.23970            2.6     0.098
## 3           7.8             0.76     0.04000            2.3     0.092
## 4          11.2             0.28     0.56000            1.9     0.075
## 5           7.4             0.70     0.12750            1.9     0.076
## 6           7.4             0.66     0.12955            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 type
## 1       5  red
## 2       5  red
## 3       5  red
## 4       6  red
## 5       5  red
## 6       5  red
```

```
# Asignamos el valor imputado  sobre el dataset.clean
wine_quality.clean$fixed.acidity = imp$ximp$fixed.acidity
wine_quality.clean$volatile.acidity = imp$ximp$volatile.acidity
wine_quality.clean$citric.acid = imp$ximp$citric.acid
wine_quality.clean$residual.sugar = imp$ximp$residual.sugar
wine_quality.clean$chlorides = imp$ximp$chlorides
wine_quality.clean$free.sulfur.dioxide = imp$ximp$free.sulfur.dioxide
wine_quality.clean$total.sulfur.dioxide = imp$ximp$total.sulfur.dioxide
wine_quality.clean$density = imp$ximp$density
wine_quality.clean$pH = imp$ximp$pH
wine_quality.clean$sulphates = imp$ximp$sulphates
wine_quality.clean$alcohol = imp$ximp$alcohol
# Verificamos que se han imputado correctamente todos los valores
length(which(is.na(wine_quality.clean)))
```

```
## [1] 0
```

Como vemos, se han logrado imputar todos los valores adecuadamente.

***Nota**: Se ha de tener en cuenta que aplicar cambios sobre los valores del dataset puede implicar perdida de información. Aunque en el ámbito de esta práctica, por motivos formativos hemos decidido tratar una serie de datos como valores erróneos o perdidos, en un contexto real, en caso de no disponerse de la posibilidad de consultar a un experto en la materia, la mejor opción sería realizar distintas pruebas llevando a cabo el análisis tanto con los datos originales, como con datos corregidos. De esta forma se facilitará la comprensión de los datos y de en qué medida se ve afectado un resultado analítico por los cambios en los mismos.*

3.3.3 Selección y reducción de los datos

Una vez han sido realizadas las tareas de tratamiento de datos perdidos y valores extremos, se va a realizar una selección y reducción de los datos.

Con el fin de poder realizar estudios comparativos entre vinos de tipo tinto y blanco, sobre un número equivalente de observaciones, se van a aplicar un mecanismo de reducción de la cantidad, para obtener el mismo número de muestras para vino blanco, que las que disponemos para vino tinto.

En concreto se aplicará una técnica de ***sampling*** (de tipo muestreo aleatorio simple sin sustitución), que es una vía simple para obtener el resultado deseado:

```
# seleccionamos vinos tintos
wine_quality_red.clean = wine_quality.clean[wine_quality.clean$type=="red",]

# seleccionamos una muestra de vinos blancos
# - establecemos semilla (esto permitira que el resultado sea reproducible)
set.seed(10)
# - generamos la muestra aplicando un mmuestreo aleatorio simple sin sustitución
wine_quality_white.clean = sample_n(wine_quality.clean[wine_quality.clean$type=="white",], 1599,
replace = FALSE)

# creamos un unico dataset a partir de los dos previos
wine_quality.clean.total = wine_quality.clean
wine_quality.clean = data.frame(wine_quality_red.clean)
wine_quality.clean = rbind(wine_quality.clean, wine_quality_white.clean)
```

Respecto a los atributos a considerar, en principio no se van a realizar tareas para llevar a cabo una reducción de la dimensionalidad, ya que se desea poder trabajar con el conjunto completo de características para facilitar el análisis de en qué medida afecta cada característica a la calidad de los vinos resultantes.

***Nota**: se ha preferido realizar esta fase tras el paso de tratamiento de datos perdidos y valores extremos, ya que en esa fase se aplican mecanismo para imputación de valores que darán mejor resultando contando con la mayor cantidad de información.*

3.3.4 Conversiones

Finalmente, previo a comenzar con el análisis de los datos, se estudia si conviene realizar algún tipo de transformación sobre los datos que pueda ayudar a que los posteriores análisis sean más eficientes, y/o puedan facilitar la comprensión de los resultados.

Con el objetivo de mejorar la comprensión del resultado de alguno de los análisis se va a generar un nuevo atributo, a partir de la discretización del atributo de calidad, estableciéndose los siguientes rangos:

- low: [1-4]
- medium: [5-6]
- high: [7-10]

```
# Estandarización
# - Quality level 'low' : [1-4]
wine_quality.clean$quality_level[wine_quality.clean$quality >= 1 & wine_quality.clean$quality <= 4] =
"low"
# - Quality level 'medium' : [5-6]
wine_quality.clean$quality_level[wine_quality.clean$quality >= 5 & wine_quality.clean$quality <= 6] =
"medium"
# - Quality level 'high' : [7-10]
wine_quality.clean$quality_level[wine_quality.clean$quality >= 7 & wine_quality.clean$quality <= 10]
= "high"

wine_quality.clean$quality_level = as.factor(wine_quality.clean$quality_level)
```

Adicionalmente, puesto que en análisis posteriores se utilizarán algunas técnicas basadas en las distancias, y estas pueden verse muy afectadas por la falta de normalización de los datos, se generará un juego de datos normalizado para asegurarnos la calidad de los resultados de dichos análisis:

```
# Normalización de los atributos numéricos
wine_quality.clean.scaled = as.data.frame(scale(wine_quality.clean[1:11]))
wine_quality.clean.scaled$quality = wine_quality.clean$quality
wine_quality.clean.scaled$quality_level = wine_quality.clean$quality_level
wine_quality.clean.scaled$type = wine_quality.clean$type
```

3.3.5 Exportación de los datos preprocesados

Una vez hemos realizado todas las tareas para la limpieza de los datos, procedemos a volcarlos, para tener un juego de datos limpio del que partir para posteriores análisis.

Se llevará a cabo el volcado tanto del dataset estandarizado como no estandarizado, para poder hacer uso de los datos en el formato que convenga según la tarea a ser realizada.

```
# Escritura en .csv de los datos sin estandarizar
write.csv(wine_quality.clean, "datasets/wine_quality.csv",row.names=FALSE)
# Escritura en .csv de los datos estandarizarizados
write.csv(wine_quality.clean.scaled, "datasets/wine_quality_scaled.csv",row.names=FALSE)
```

3.4 Análisis de los datos y construcción de modelos – *data mining*

3.4.1 Selección grupos de datos a analizar y planificación de los análisis

Con el objetivo de poder llevar a cabo los estudios necesarios para poder satisfacer los objetivos analíticos que nos hemos planteado al inicio de este trabajo, en este apartado se dejarán cargados y preparados los *datasets* que puedan ser necesarios.

En nuestro caso partiremos de los juegos de datos que han sido generados en la etapa previa, y para poder realizar estudios comparativos sobre los grupos de datos asociados a cada tipo de vino, se realizará una subdivisión de los datos por esta característica.

```
# carga de juego de datos vinos - normalizada
wine_quality.clean.scaled = read.csv("datasets/wine_quality_scaled.csv")
wine_quality_red.clean.scaled = wine_quality.clean.scaled[wine_quality.clean.scaled$type == "red",]
wine_quality_white.clean.scaled = wine_quality.clean.scaled[wine_quality.clean.scaled$type ==
"white",]

# carga de juego de datos vinos - sin normalizada
wine_quality.clean = read.csv("datasets/wine_quality.csv")
wine_quality_red.clean = wine_quality.clean.scaled[wine_quality.clean.scaled$type == "red",]
wine_quality_white.clean = wine_quality.clean.scaled[wine_quality.clean.scaled$type == "white",]
```

Una vez han sido cargados los *datasets* que van a ser usados, enumeramos el conjunto de análisis que van a ser realizados:

- verificación de la normalidad y homogeneidad de la varianza, paso previo a poder aplicar algunos tipos de test estadísticos,
- comparación entre grupos, trataremos de determinar si el tipo de vino puede vincularse de alguna forma a la calidad,
- estudios de correlación, trataremos de comprender en mayor nivel de profundidad como las distintas características influencian la calidad, y se influencian entre ellas,
- construcción de modelos de clasificación, perseguiremos de entender en mayor profundidad la relación de las características del vino con la calidad, así como obtener un mecanismo que nos permita predecir el nivel de calidad para nuevas observaciones de vino,
- exploraciones visuales, trataremos de consolidar, resumir y representar el conocimiento adquirido.

3.4.2 Comprobación normalidad y homogeneidad de la varianza

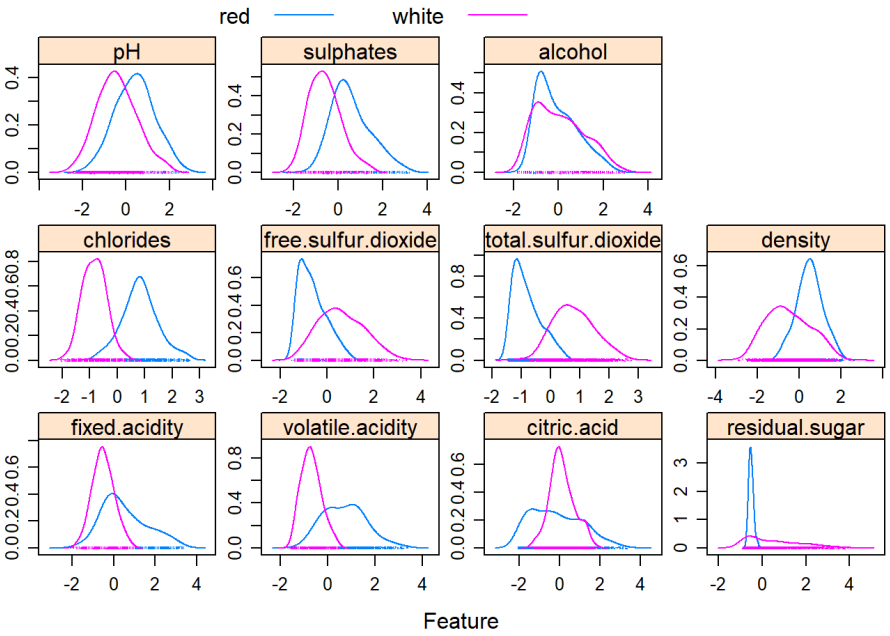
A la hora de llevar a cabo el análisis de los datos, se ha de considerar que algunas pruebas estadísticas, como puede ser por ejemplo el contraste de hipótesis, requieren contar con el conocimiento de si la distribución de los datos sobre los que se trabaja es normal, también es habitual que otro tipo de test usados por ejemplo cuando se desea realizar pruebas de comparación entre grupos, requieran partir del conocimiento de si existe homogeneidad de la varianza entre los distintos grupos.

Es por ello de gran importancia llevar a cabo estas verificaciones previamente a la aplicación de pruebas estadísticas que requieran partir de este conocimiento.

Comenzamos por la **verificación de la normalidad**, la cual llevaremos a cabo haciendo uso en primer lugar de unos diagramas de densidad, que nos darán una impresión inicial de la distribución que siguen las distintas variables, y confirmando la primera impresión a través de la aplicación de unos test de normalidad (se aplicarán dos de los más robustos, el de ***Shapiro-Wilk*** y el ***Anderson-Darling***).

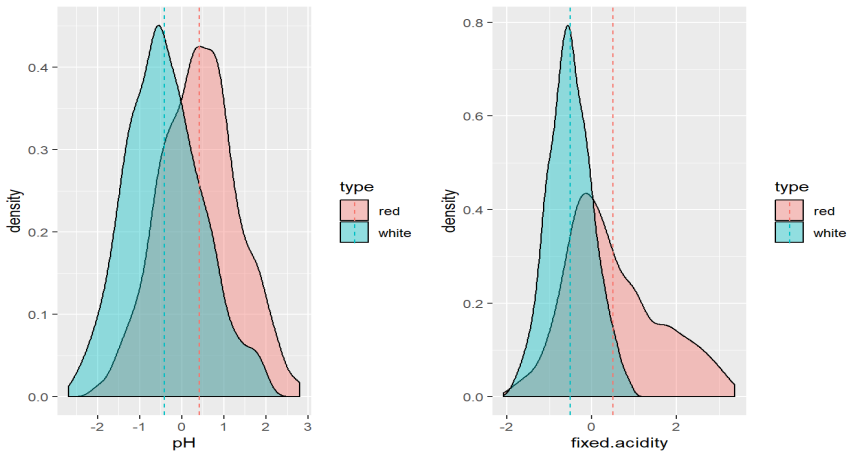
```
# construir un plot con los diagramas de densidad por características y tipo de vino
wine_quality = wine_quality.clean.scaled
featurePlot(x = wine_quality[, 1:11],
```

```
        y = wine_quality$type, plot = "density",
        scales = list(x = list(relation="free"), y = list(relation="free")),
        adjust = 1.5, pch = ".",
        layout = c(4, 3), auto.key = list(columns = 3))
```



Al visualizar los diagramas de densidad agrupados por clase de vino, vemos que la distribución de algunas variables tienen un aspecto similar a una distribución normal, observemos un poco más de cerca la distribución para las variables *pH* y *fixed.acidity*:

```
# diagrama de densidad para el 'pH' por tipo de vino, indicando valor central
mu = ddply(wine_quality, "type", summarise, grp.mean=mean(pH))
plot1 = ggplot(wine_quality, aes(x=pH, fill=type)) +
                              geom_density(alpha=0.4) +
                              geom_vline(data=mu, aes(xintercept=grp.mean,
color=type),linetype="dashed")
# diagrama de densidad para el 'fixed.acidity' por tipo de vino, indicando valor central
mu = ddply(wine_quality, "type", summarise, grp.mean=mean(fixed.acidity))
plot2 = ggplot(wine_quality, aes(x=fixed.acidity, fill=type)) +
                              geom_density(alpha=0.4) +
                              geom_vline(data=mu, aes(xintercept=grp.mean,
color=type),linetype="dashed")

grid.arrange(plot1, plot2, ncol=2)
```



Vemos como la media de la distribución se desplaza del centro de esta, por lo que no parece ser tan normal.

Para salir de dudas ejecutemos los test de normalidad para verificar si contamos o no con alguna característica que siga una distribución normal. Como se ha comentado se aplicarán el test de ***Shapiro-Wilk*** y el ***Anderson-Darling***, ambos se basan en la realización de un contraste de hipótesis del siguiente tipo:

$H_0 = la\ población\ está\ distribuida\ normalmente$

$H_1 = la\ población\ no\ está\ distribuida\ normalmente$

$p\_valor < \alpha \rightarrow Rechazamos\ la\ hipótesis\ nula, los\ datos\ no\ siguen\ una\ distribución\ normal$

$p\_valor > \alpha \rightarrow No\ se\ puede\ \ rechazar\ la\ hipótesis\ nula, se\ asume\ distribución\ normal$

$para\ \alpha = 0{,}05$

```
# definición función para ejecutar test de normalidad sobre los atributos
# del dataset de vinos, en función del tipo de vino, y retornar los resultados
getNormlityTestInfo = function(wine_quality, colnames, alpha) {
  # cabecera
  rowHeader = list("tipo","att", "ad.p_val", "ad.test", "shapiro.p_val", "shapiro.test")
  # inicialización datma.frame con los resultados
  df = data.frame(rowHeader,stringsAsFactors = F)

  # se itera por las características del vino
  for (col in colnames) {
    # se itera por los tipos de vino
    for (wine type in c("red", "white")) {
      ad.p_val = ad.test(wine_quality[wine_quality$type==wine_type,][,col])$p.value
      shapiro.p_val = shapiro.test(wine_quality[wine_quality$type==wine_type,][,col])$p.value
      ad.result = 'yes'
      if(ad.p_val < alpha) {
        ad.result = 'no'
      }
      shapiro.result = 'yes'
      if(shapiro.p_val < alpha) {
        shapiro.result = 'no'
      }
      row = list(wine type, col, ad.p val, ad.result, shapiro.p val, shapiro.result);
      df= rbind(df,row)
    }
  }
  return(df)
}

# se ejecuta la función para pasar los test de normalidad
colnames = names(wine_quality.clean.scaled[1:12]);
alpha = 0.05
infoNormalityTest = getNormlityTestInfo(wine_quality.clean.scaled, colnames, alpha)
kable(infoNormalityTest)
```

Tipo	Att	ad.p_val.	ad.test.	shapiro.p_val.	shapiro.test.
red	fixed.acidity	3.7e-24	no	3.96358680364882e-19	no
white	fixed.acidity	5.83617895752229e-05	no	0.000167373318738328	no
red	volatile.acidity	1.24580006566986e-09	no	2.66007290902294e-09	no
white	volatile.acidity	1.03331998425636e-13	no	5.05530187782354e-12	no
red	citric.acid	3.7e-24	no	9.39356206274636e-21	no
white	citric.acid	3.7e-24	no	8.28479716148848e-13	no
red	residual.sugar	8.61871508918918e-22	no	1.32261430737927e-16	no
white	residual.sugar	3.7e-24	no	3.85697907588753e-31	no
red	chlorides	1.11830665113359e-08	no	1.97385576979077e-07	no

Tipo	Att	ad.p_val.	ad.test.	shapiro.p_val.	shapiro.test.
white	chlorides	0.000689576864456304	no	0.00256495761673471	no
red	free.sulfur.dioxide	3.7e-24	no	4.57390007477333e-26	no
white	free.sulfur.dioxide	1.27299498643234e-12	no	9.49310884517731e-11	no
red	total.sulfur.dioxide	3.7e-24	no	7.02565529305316e-28	no
white	total.sulfur.dioxide	2.28941107150678e-09	no	1.28732465952878e-07	no
red	density	0.0180505676694264	no	0.000274463387213671	no
white	density	1.95598383703793e-19	no	3.70192363456708e-13	no
red	pH	0.0215310866230915	no	0.00336264773493402	no
white	pH	1.94146323988665e-06	no	2.33827687596885e-07	no
red	sulphates	3.7e-24	no	2.48539055709117e-17	no
white	sulphates	5.02847579925355e-16	no	4.67158010298543e-13	no
red	alcohol	3.7e-24	no	6.76027474654803e-26	no
white	alcohol	3.7e-24	no	2.95638295392442e-21	no
red	quality	3.7e-24	no	9.51508538364454e-36	no
white	quality	3.7e-24	no	1.02968548473584e-32	no

Parece ser que ninguna de las características del vino sigue una distribución normal.

Comprobemos ahora la **homogeneidad de las varianzas**, para lo cual aplicaremos el test no paramétrico de ***Fligner-Killeen*** ya que partimos de un conjunto de variables que no siguen una distribución normal. En este caso también se trata de un contraste de hipótesis:

$H_0: \sigma_r = \sigma_w \quad (igualdad\ de\ las\ varianzas)$
$H_1: \sigma_r <> \sigma_w \quad (desigualdad\ de\ las\ varianzas)$
$p\_valor < \alpha \rightarrow Rechazamos\ la\ hipótesis\ nula, no\ existe\ homogeneidad\ de\ las\ varianzas$
$p\_valor > \alpha \rightarrow No\ se\ puede\ \ rechazar\ la\ hipótesis\ nula, se\ asume\ homogeneidad\ de\ las\ varianzas$
$para\ \ \alpha = 0{,}05$

En nuestro caso vamos a estudiar la homogeneidad referente a los **grupos de vinos de tipo tinto y blanco**, para la **variable *quality*** de vino.

```
# Verificación de la homogeneidad con el test de Fligner-Killeen
fligner.test(quality~type, data = wine_quality.clean.scaled)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  quality by type
## Fligner-Killeen:med chi-squared = 1.8516, df = 1, p-value = 0.1736
```

Obteneos un *p-valor* superior al nivel de significación, por lo que no se puede rechazar la hipótesis nula, pudiendo asumir en consecuencia homogeneidad de las varianzas de los grupos para la variable 'quality'.

3.4.3 Aplicación de pruebas estadísticas

3.4.3.1 Comparación entre grupos

Como se ha comentado inicialmente, estamos interesados en comprender los factores que afectan a la calidad del vino, y dado que partimos de dos grandes grupos de observaciones, las correspondientes a los vinos tintos y a los blancos, nos preguntamos ***¿existen diferencias significativas en la calidad del vino en función de si es tinto o blanco?***

Para responder a esta pregunta vamos a comparar la calidad de los vinos entre los grupos de vinos tinto y blanco, para ver si hay diferencias significativas. Para ello vamos a plantear un contraste de hipótesis sobre las muestras para determinar si un tipo de vino obtiene valores de calidad superiores al otro tipo de vino.

Se ha de considerar, que dado que en las verificaciones previas se ha concluido que la distribución de la variable *quality* no es normal, se deberían aplicar pruebas estadísticas no paramétricas para llevar a cabo esta comparación, pero puesto que el tamaño de la muestra n > 30, y según el **TCL** (teorema central del límite), en estos casos la media se puede aproximar a una distribución normal, y sería viable aplicar un contrate con la técnica ***t de Student*** del siguiente tipo:

$$H_0: \mu_r - \mu_w = 0 \quad (igualdad\ de\ las\ medias)$$
$$H_0: \mu_r - \mu_w < 0 \quad (media\ para\ tintos\ inferior\ a\ media\ para\ blancos)$$
$$p\_valor < \alpha \rightarrow Rechazamos\ la\ hipótesis\ nula, diferencias\ en\ la\ calidad\ en\ función\ del\ tipo\ de\ vino$$
$$p\_valor > \alpha \rightarrow No\ se\ puede\ \ rechazar\ la\ hipótesis\ nula, no\ existen\ difrencias\ en\ la\ calidad$$
$$para\ \ \alpha = 0{,}05$$

```
# Realizar t de Student sobre la variable qualitypara los dos tipos de vino
t.test(wine_quality_red.clean.scaled$quality, wine_quality_white.clean.scaled$quality, alternative =
"less")
```

```
##
##  Welch Two Sample t-test
##
## data:  wine_quality_red.clean.scaled$quality and wine_quality_white.clean.scaled$quality
## t = -8.1955, df = 3176.2, p-value < 2.2e-16
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##        -Inf -0.1949365
## sample estimates:
## mean of x mean of y
##  5.636023  5.879925
```

Si preferimos plantear el contraste de hipótesis apoyándonos en un test no paramétrico, dado que partimos de datos donde no se cumple la normalidad, podríamos hacerlo de la siguiente forma con un test de **Mann-Whitney**:

$$H_0: distribución_r = distribución_w$$
$$H_0: distribución_r <> distribución_w$$
$$p\_valor < \alpha \rightarrow Rechazamos\ la\ hipótesis\ nula$$
$$p\_valor > \alpha \rightarrow No\ se\ puede\ \ rechazar\ la\ hipótesis\ nula$$
$$para\ \ \alpha = 0{,}05$$

```
wilcox.test(quality~type, data = wine_quality.clean.scaled )
```

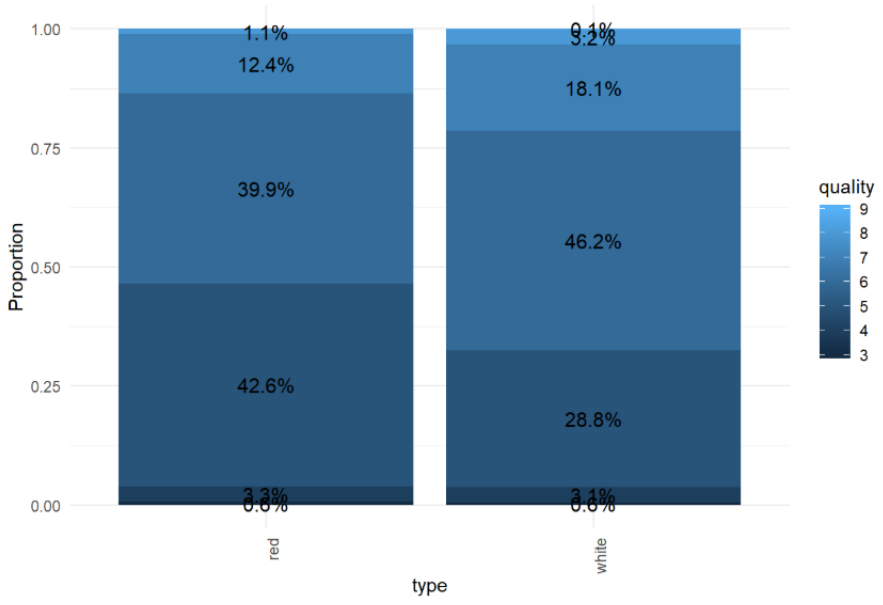
```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  quality by type
```

```
## W = 1072500, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

En ambos casos, observamos que se obtiene un *p-value* que impide aceptar la hipótesis nula, por lo que confirmamos la diferencia en la calidad existente en base al tipo de vino.

Esta misma conclusión podemos obtenerla de forma visual a través del siguiente plot bivariable, en la que se muestra la calidad agrupada por tipo de vino:

```
## construye un diagrama de barras de frecuencias para 2 variables
barplotBiVariable = function(dataset, att1, att2, labsize=4) {
  plotdata = dataset %>%
      dplyr::group_by(dataset[,att1], dataset[,att2]) %>%
      dplyr::summarize(n = n()) %>%
      dplyr::mutate(pct = n/sum(n),
             lbl = scales::percent(pct))

  names(plotdata) = c("VAR1", "VAR2", "n", "pct", "lbl")

  ## plot
  plot1 = ggplot(data = plotdata,
                 aes(x = VAR1,
                 y = pct,
                 fill = VAR2,
                 cumulative = TRUE)) +
                 geom_col() +
          labs(y = "Proportion", x=att1, fill=att2, title="title")  +
          geom_text(aes(label = lbl), size=labsize,
                    position = position_stack(vjust = 0.5)) +
          theme_minimal()+
          theme(axis.text.x = element_text(angle = 90, hjust = 1))
  return(plot1)
}
# mostramos una gráfica bi variable por tipo de vino y calidad
plot1 = barplotBiVariable(wine_quality.clean.scaled, "type", "quality")
plot1
```



3.4.3.2 Correlación

Hasta el momento hemos descubierto que los vinos blancos consiguen mejor puntuación de calidad que los vinos tintos, y ahora estamos interesados en saber **¿en qué medida las distintas características del vino influyen sobre su calidad?**

Para lograr responder a esta pregunta, podemos llevar a cabo un **análisis de la correlación**, que nos ayudara a medir el nivel de influencia que ejerce cada característica del vino sobre la calidad.

Dado que nuestros datos no siguen una distribución normal, vamos a aplicar una prueba no paramétrica, el coeficiente de ***correlación de Spearman***:

```
# Definición de función para extraer un resumen del análisis de outliers
getCorrelationTableInfo = function (wine_quality, colnames) {
  # cabecera
  rowHeader = list("type", "att", "estimate", "p-value")
  # inicialización data.frame con los resultados
  df = data.frame(rowHeader,stringsAsFactors = F)
  # se itera por los atributos
  for (col in colnames) {
    # se itera por los tipos de vino
    for (wine_type in c("red", "white")) {
      # aplicar el test de correlación
      spearman_test = cor.test(wine_quality[wine_quality$type==wine_type,][,col],
                               wine_quality[wine_quality$type==wine_type,]$quality,
                               method="spearman", exact=FALSE)
      correlation_coef = spearman_test$estimate
      p_value = spearman_test$p.value

      # Crear fila de datos y añadir al data.frame
      row = list(wine_type, col, correlation_coef, p_value);
      df= rbind(df,row)
    }
  }
  return(df)
}
# obtener las columnas sobre las que se desea analizar los outliers
colnames = names(wine_quality.base[1:11])
# invocar el análisis e imprimir el resultado
res = getCorrelationTableInfo(wine_quality.clean.scaled, colnames)
kable(res)
```

type.	att.	estimate.	p.value.
red	fixed.acidity	0.115131599958497	3.91999167418541e-06
white	fixed.acidity	-0.0394530823745914	0.114793155252787
red	volatile.acidity	-0.374114411841474	2.74196551886805e-54
white	volatile.acidity	-0.185674864571651	7.20749633002524e-14
red	citric.acid	0.216224496606583	2.26715224370346e-18
white	citric.acid	0.0709050732869421	0.0045586703197026
red	residual.sugar	0.0430790455656413	0.0850558320788889
white	residual.sugar	-0.0793231782468937	0.00150116650673849
red	chlorides	-0.181183662200922	2.87955740856384e-13
white	chlorides	-0.276181943866384	2.19325346356455e-29
red	free.sulfur.dioxide	-0.0546965376278153	0.0287348898994526
white	free.sulfur.dioxide	0.0237652315112967	0.342263129311513
red	total.sulfur.dioxide	-0.194256300843008	4.63234136867916e-15
white	total.sulfur.dioxide	-0.171607334137447	4.91231497968582e-12
red	density	-0.17965280705073	4.58021240594639e-13
white	density	-0.324220811958486	1.86907246265241e-40
red	pH	-0.047191013217482	0.0592106537897018

type.	att.	estimate.	p.value.
white	pH	0.106294238046111	2.05171045814644e-05
red	sulphates	0.416775343629145	3.2570361897382e-68
white	sulphates	-0.00861743468026149	0.730600705070177
red	alcohol	0.479276174488506	1.3005046084849e-92
white	alcohol	0.422945113944542	2.11498301086771e-70

Como sabemos, el coeficiente de correlación asume valores entre [-1, 1], donde los extremos indican una correlación perfecta, y el 0 indica que no existe ninguna correlación, de igual forma el signo indica la dirección de la correlación.

Del resultado podemos observar que existen algunas características con una correlación mayor respecto a otras. Para obtener las correlaciones más relevantes hemos de considerar también el valor de *p-value*, ya que esto nos indica el nivel de significación asociado a la correlación, siendo más significativa a menores valores de *p-value* (dado que la hipótesis nula es la no existencia de correlación).

En base a los resultados y en función del tipo de vino, listamos algunas de las características que influencian más la calidad:

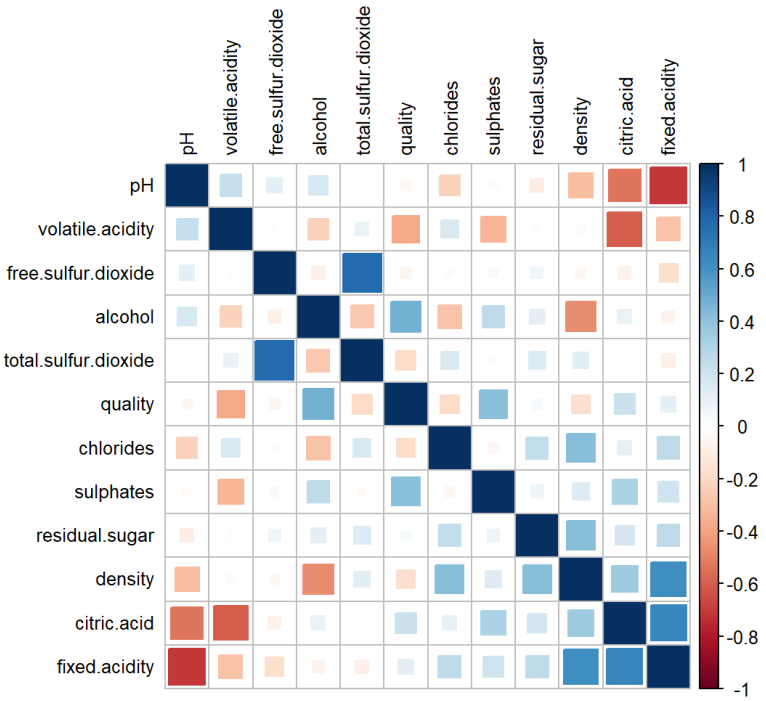
- ***red***:
  - correlación directa: *alcohol*, *sulphates*, *citric.acid*, *fixed.acidity*
  - correlación inversa: *volatile.acidity*, *chlorides*, *total.sulfur.dióxide*, *density*
- ***white***:
  - correlación directa: *alcohol, pH, citric.acid*
  - correlación inversa: *density, chlorides, volatile.acidity, total.sulfur.dióxide*

Será interesante estudiar también la interrelación de unas características respecto a otras, vamos a ayudarnos a través de un plot con información de correlación entre pares de características, por tipo de vino:

```
# función para realizar plot de la correlación, según el método suministado
# como parámetro (por defecto 'spearman')
plotCorrelation = function(wine_quality, method="spearman") {
  nc=ncol(wine_quality )
  df <- wine_quality [,1:12]
  correlations <- cor(df,method="spearman")
  corrplot(correlations, number.cex = .9, method = "square",
                hclust.method = "ward", order = "FPC",
                type = "full", tl.cex=0.8,tl.col = "black")
}

# plot de correlación para vino tinto
wine_quality = wine_quality_red.clean.scaled
plotCorrelation(wine_quality)
# plot de correlación para vino blanco
wine_quality = wine_quality_white.clean.scaled
plotCorrelation(wine_quality)
```

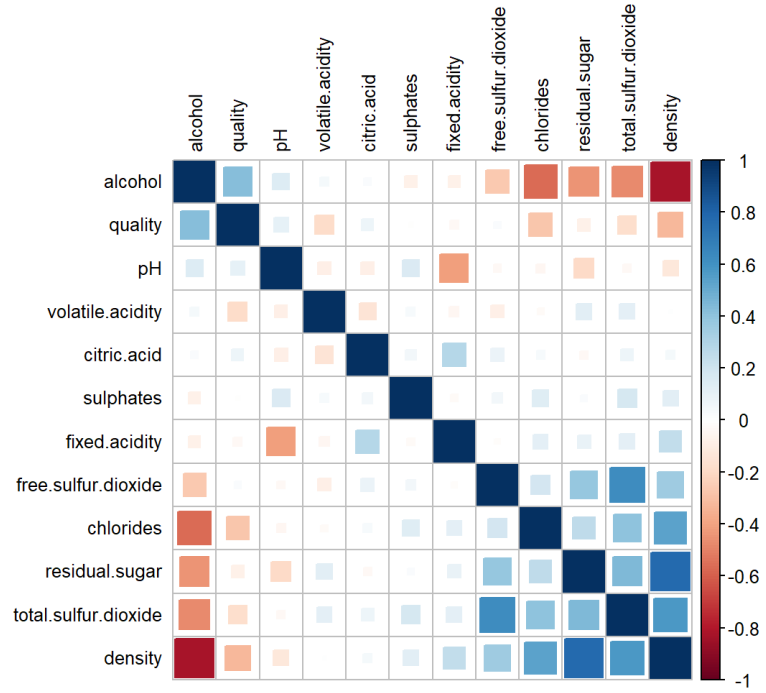
Correlaciones

Vino Tinto



Correlaciones

Vino blanco



A continuación, se incluyen algunas de las correlaciones más reseñables:

Vino tinto	
Correlación directa	**Correlación inversa**

• correlación relativamente alta:
  ○ *total.sulfur.dioxide* con *free.sulfur.dioxide*
  ○ fixed.acidity con *density* y *citric.acid*
• correlación relativa:
  ○ ***quality*** con ***alcohol***
  ○ *sulphates* con *quality*
  ○ density con *chlorides*, *residual.sugar* y *citric.acid*
	• correlación muy alta:
	  ○ *pH* con *fixed.acidity*
	• correlación relativamente alta:
	  ○ *citric.acid* con *volatile.acidity*
	• correlación relativa:
	  ○ ***quality*** con ***volatile.acidity***
	  ○ *citric.acid* con *pH*
	  ○ *density* con *alcohol*

Vino blanco	
Correlación directa	**Correlación inversa**

• correlación relativamente alta:
  ○ *density* con *residual.sugar*
• correlación relativa:
  ○ *total.sulfur.dioxide* con *free.sulfur.dioxide*
  ○ *density* con *total.sulfur.dioxide y chlorides*
  ○ ***quality*** con ***alcohol***
	• correlación muy alta:
	  ○ *density* con *alcohol*
	• correlación relativa:
	  ○ *chlorides* con *alcohol*
	  ○ *total.sulfur.dioxide* con *alcohol*
	  ○ *fixed.acidity* con pH
	  ○ ***quality*** con ***density***

3.4.3.3 Clasificación

La tarea de **Clasificación** es una tarea de *data mining* de tipo **predictivo**, es decir, hace uso de las variables de los datos disponibles con el objetivo de lograr la predicción de otras variables y/o valores desconocidos de interés, en este caso la variable objetivo se tratará de un valor que se encuentra dentro de un conjunto de clases predefinidos.

En este tipo de tareas, se aplican generalmente métodos **supervisados**, en los que se requiere intervención humana, y cuya entrada es un conjunto de observaciones etiquetada.

En nuestro caso, resulta de gran interés aplicar un método de clasificación para generar un modelo que podrá cumplir un doble papel:

- ayudarnos a realizar una mejor descripción del dominio de los datos, es decir, puede ayudarnos a entender la relación existente entre las distintas características de los vinos y su calidad,
- servirnos como método predictivo de la calidad para nuevas observaciones de datos.

Para llevar a cabo la construcción del modelo, se partirá de los conjuntos de datos asociados a vino tinto y vino blanco, creados en el apartado 3.4.1 SELECCIÓN GRUPOS DE DATOS A ANALIZAR, y se generará un modelo para cada tipo de vino, siendo la variable objetivo el *quality_level*.

3.4.3.3.1 Modelo clasificación nivel de calidad de vino tinto

Existen diversos métodos para construir modelos de clasificación, en nuestro caso vamos a hacer uso de un método de construcción basado en árboles de decisión, conocido como **CART** (*Classification and Reggression Trees*) que es un método muy extendido y potente. Se usará el algoritmo **RPART**, ya que cuenta con una gran capacidad para generar árboles fácilmente interpretables.

```
## constucción modelo CART

# se seleccionan los datos usados para la construcción del modelo
# (incluye solo características)
wine_quality = data.frame(wine_quality_red.clean[c(1:11,13)])
#wine_quality$quality = as.factor(wine_quality$quality )
# Se establece una semilla para que los resultados sean reproducibles
set.seed(10)
## partimos 75% observaciones para construcción del modelo, 25% para test
split = sample.split(wine_quality$quality_level, SplitRatio = 0.75)
training_set = subset(wine_quality, split == TRUE)
test_set = subset(wine_quality, split == FALSE)

## contruimos los parámetros que necesarios para la construcción del modelo
numatributes = length(training_set)
## variables explicativas
training_setX = training_set[c(1:11)]
test_setX = test_set[c(1:11)]
## variable objetivo
training_setY = training_set[,numatributes] #CLUSTER_RFM_LAB attribute
```

```
test_setY = test_set[,numatributes]

## Construimos el modelo
modelCARTIt1 = rpart(formula = quality_level ~ .,
                  data=training_set, method="class")

## Visualizar resultados del entrenamiento
printcp(modelCARTIt1)
```

```
##
## Classification tree:
## rpart(formula = quality_level ~ ., data = training_set, method = "class")
##
## Variables actually used in tree construction:
## [1] alcohol              citric.acid          density
## [4] pH                   residual.sugar       sulphates
## [7] total.sulfur.dioxide volatile.acidity
##
## Root node error: 210/1199 = 0.17515
##
## n= 1199
##
##         CP nsplit rel error  xerror     xstd
## 1 0.066667      0   1.00000 1.00000 0.062673
## 2 0.022222      2   0.86667 0.96667 0.061837
## 3 0.021429      5   0.80000 0.89524 0.059955
## 4 0.014286      7   0.75714 0.87143 0.059298
## 5 0.012698      9   0.72857 0.90000 0.060085
## 6 0.011905     12   0.69048 0.90000 0.060085
## 7 0.010000     14   0.66667 0.90000 0.060085
```

En el resultado, se puede observar el detalle de la construcción del modelo, algunos datos relevantes que podemos encontrar son las variables usadas para la construcción del modelo, y el porcentaje de error en función del nivel de partición del árbol, que puedes ser obtenido del *root node error*, y a partir de la tabla que figura a continuación de dicho valor.

Una vez construido el modelo, podemos realizar la predicción de las observaciones de test, y visualizar el resultado, y la matriz de contingencias asociada.

```
# cálculamos predicciones forzando a que el retorno sea una predicción de la clase más probable
predicted_modelCARTIt1 = predict( modelCARTIt1, test_setX, type="class" )

# mostramos una tabla en la que pueden observarse la ubicación de las predicciones.
modelCARTIt1_pred_table = table(predicted_modelCARTIt1)
modelCARTIt1_pred_table
```

```
## predicted_modelCARTIt1
##   high    low medium
##     40      0    360
```

```
## matriz de confusión
mat_conf<-table(test_setY,Predicted=predicted_modelCARTIt1)
mat_conf
```

```
##          Predicted
## test_setY high low medium
##    high     18   0     36
##    low       0   0     16
##    medium   22   0    308
```

Y a partir de los datos obtenidos podemos estimar la precisión del modelo:

```
## realizamos una estimación de la precisión del árbol
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_modelCARTIt1 == test_setY) /
length(predicted_modelCARTIt1)))
```

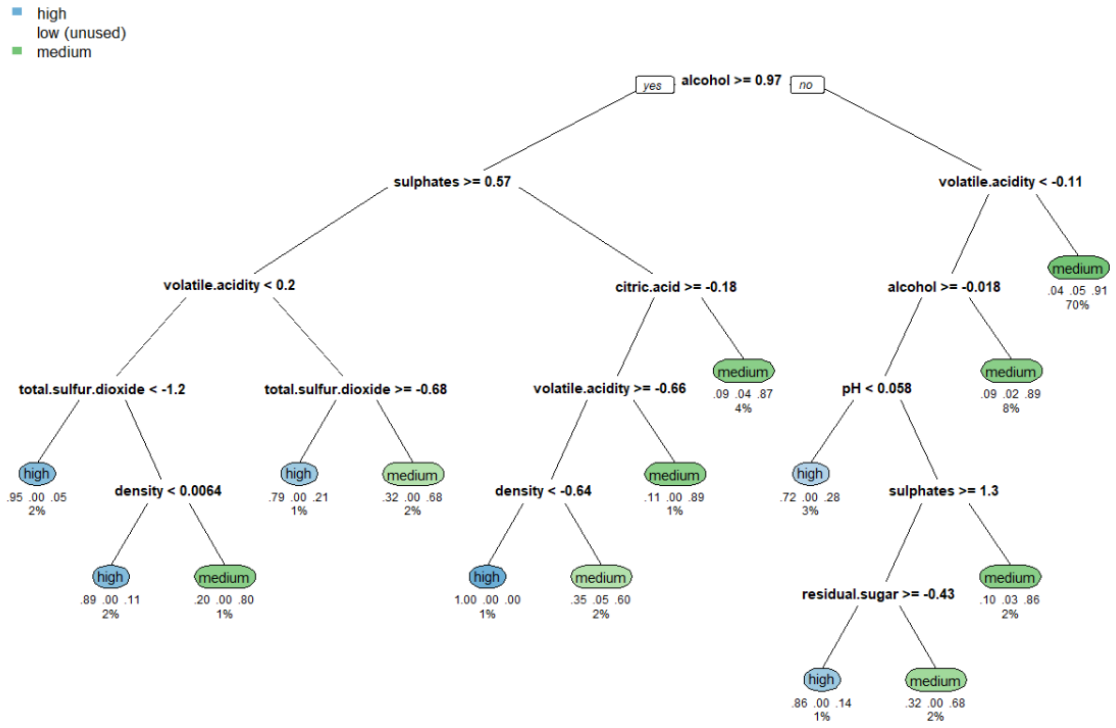
```
## [1] "La precisión del árbol es: 81.5000 %"
```

También podemos visualizar las reglas del árbol obtenido de la siguiente forma:

```
## imprimimos las reglas
rpart.rules(modelCARTIt1)
```

```
##  quality_level  high  low medi
##           high [ .72  .00  .28] when alcohol is -0.018 to 0.972 & volatile.acidity <  -0.11
& pH <  0.058
##           high [ .79  .00  .21] when alcohol >= 0.972 & volatile.acidity >=  0.20 & sulphates >= 0.57 &
total.sulfur.dioxide >= -0.68
##           high [ .86  .00  .14] when alcohol is -0.018 to 0.972 & volatile.acidity <  -0.11 & sulphates >= 1.32 &
pH >= 0.058 & residual.sugar >= -0.43
##           high [ .89  .00  .11] when alcohol >= 0.972 & volatile.acidity <   0.20 & sulphates >= 0.57 &
total.sulfur.dioxide >= -1.21 & density <   0.0064
##           high [ .95  .00  .05] when alcohol >= 0.972 & volatile.acidity <   0.20 & sulphates >= 0.57 &
total.sulfur.dioxide <  -1.21
##           high [1.00  .00  .00] when alcohol >= 0.972 & volatile.acidity >= -0.66 & sulphates <  0.57 &
citric.acid >= -0.18 & density <  -0.6411
##         medium [ .35  .05  .60] when alcohol >= 0.972 & volatile.acidity >= -0.66 & sulphates <  0.57 &
citric.acid >= -0.18 & density >= -0.6411
##         medium [ .32  .00  .68] when alcohol >= 0.972 & volatile.acidity >=  0.20 & sulphates >= 0.57 &
total.sulfur.dioxide <  -0.68
##         medium [ .32  .00  .68] when alcohol is -0.018 to 0.972 & volatile.acidity <  -0.11 & sulphates >= 1.32 &
pH >= 0.058 & residual.sugar <  -0.43
##         medium [ .20  .00  .80] when alcohol >= 0.972 & volatile.acidity <   0.20 & sulphates >= 0.57 &
total.sulfur.dioxide >= -1.21 & density >=  0.0064
##         medium [ .10  .03  .86] when alcohol is -0.018 to 0.972 & volatile.acidity <  -0.11 & sulphates <  1.32 &
pH >= 0.058
##         medium [ .09  .04  .87] when alcohol >= 0.972 & sulphates <  0.57 & citric.acid <  -0.18
##         medium [ .11  .00  .89] when alcohol >= 0.972 & volatile.acidity <  -0.66 & sulphates <  0.57 &
citric.acid >= -0.18
##         medium [ .09  .02  .89] when alcohol <  -0.018 & volatile.acidity <  -0.11
##         medium [ .04  .05  .91] when alcohol <   0.972 & volatile.acidity >= -0.11
```

Así como hacer un plot del árbol de clasificación:



En conclusión, obtenemos un modelo predictivo que nos ofrece una calidad estimada del 81.5%, no muy mala, pero si mejorable.

A nivel predictivo cometerá errores, pero a nivel de comprensión del dominio de los datos podrá ofrecer una muy buena para mejorar en el nivel de conocimiento de las características del vino que influyen sobre la calidad de este, ya que se obtiene una visión muy fácilmente entendible.

3.4.3.3.2 Modelo clasificación nivel de calidad de vino blanco

```
# se seleccionan los datos usados para la construcción del modelo
# (incluye solo características)
wine_quality = data.frame(wine_quality_white.clean[c(1:11,13)])
#wine_quality$quality = as.factor(wine_quality$quality )
# Se establece una semilla para que los resultados sean reproducibles
set.seed(10)
## partimos 75% observaciones para construcción del modelo, 25% para test
split = sample.split(wine quality$quality level, SplitRatio = 0.75)
training_set = subset(wine_quality, split == TRUE)
test_set = subset(wine_quality, split == FALSE)

## contruimos los parámetros que necesarios para la construcción del modelo
numatributes = length(training_set)
## variables explicativas
training_setX = training_set[c(1:11)]
test_setX = test_set[c(1:11)]
## variable objetivo
training_setY = training_set[,numatributes] #CLUSTER_RFM_LAB attribute
test_setY = test_set[,numatributes]

## Construimos el modelo
modelCARTIt2 = rpart(formula = quality_level ~ .,
                  data=training_set, method="class")

## Visualizar resultados del entrenamiento
printcp(modelCARTIt2)
```

```
##
## Classification tree:
## rpart(formula = quality_level ~ ., data = training_set, method = "class")
##
## Variables actually used in tree construction:
## [1] alcohol          chlorides        citric.acid      density
## [5] pH               residual.sugar   sulphates        volatile.acidity
##
## Root node error: 300/1198 = 0.25042
##
## n= 1198
##
##         CP nsplit rel error  xerror     xstd
## 1 0.035000      0   1.00000 1.00000 0.049986
## 2 0.018333      2   0.93000 0.94333 0.049007
## 3 0.012222      9   0.79000 0.96333 0.049361
## 4 0.010000     12   0.75333 0.98667 0.049762
```

Una vez construido el modelo, podemos realizar la predicción de las observaciones de test y visualizar la matriz de contingencias asociada.

```
# cálculamos predicciones forzando a que el retorno sea una predicción de la clase más probable
predicted_modelCARTIt2 = predict( modelCARTIt2, test_setX, type="class" )

# mostramos una tabla en la que pueden observarse la ubicación de las predicciones.
table(predicted_modelCARTIt2)
```

```
## predicted_modelCARTIt2
##   high    low medium
##     60      0    341
```

```
## matriz de confusión
mat_conf<-table(test_setY,Predicted=predicted_modelCARTIt2)
mat_conf
```

```
##           Predicted
## test_setY high low medium
##    high     32   0     54
##    low       3   0     12
##    medium   25   0    275
```

Y a partir de los datos obtenidos podemos estimar la precisión del modelo:

```
## realizamos una estimación de la precisión del árbol
print(sprintf("La precisión del árbol es: %.4f %%",100*sum(predicted_modelCARTIt2 == test_setY) /
length(predicted_modelCARTIt2)))
```

```
## [1] "La precisión del árbol es: 76.5586 %"
```

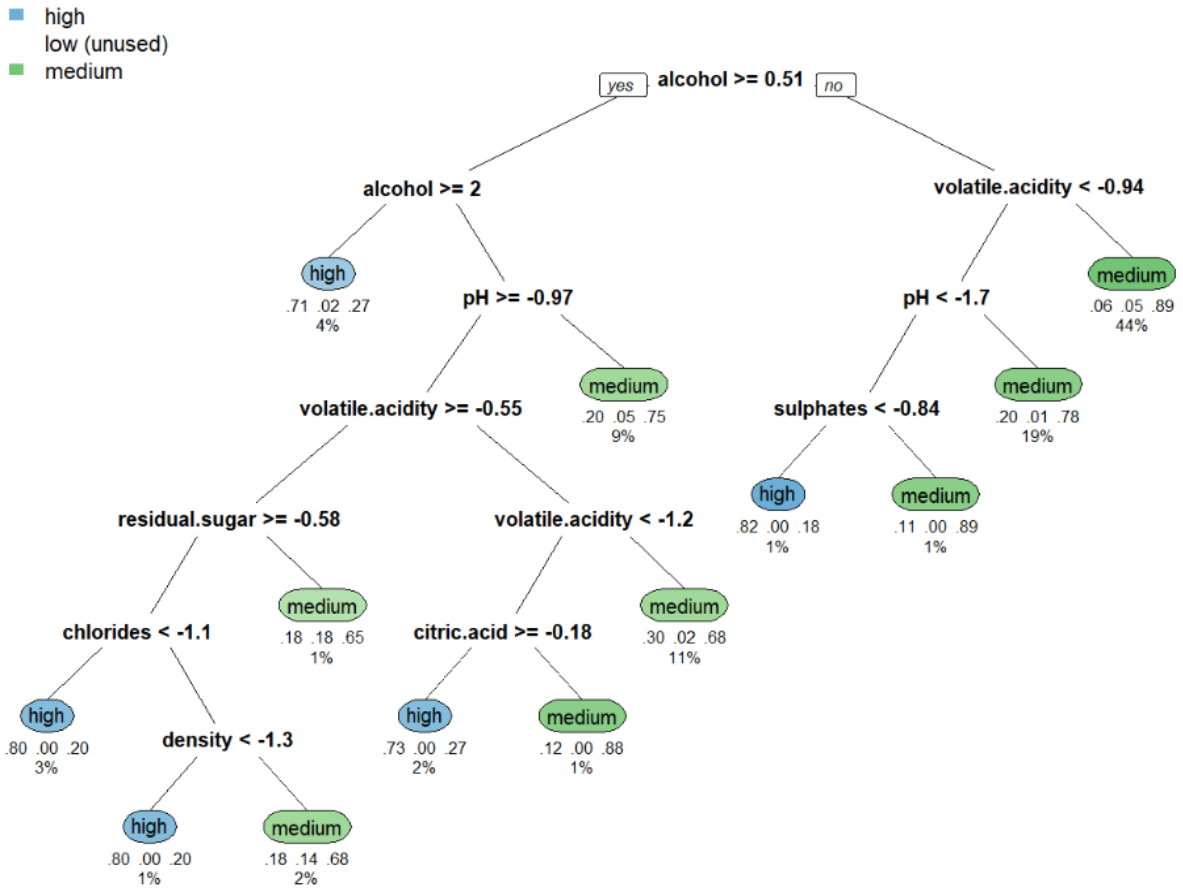
También podemos visualizar las reglas del árbol obtenido de la siguiente forma:

```
## imprimimos las reglas
rpart.rules(modelCARTIt2)
```

```
##  quality_level  hig low med
##           high [.71 .02 .27] when alcohol >= 2.03
##           high [.73 .00 .27] when alcohol is 0.51 to 2.03 & pH >= -0.97 & volatile.acidity <  -1.20
& citric.acid >= -0.18
##           high [.80 .00 .20] when alcohol is 0.51 to 2.03 & pH >= -0.97 & volatile.acidity >= -0.55 &
residual.sugar >= -0.58 & chlorides <  -1.1
##           high [.80 .00 .20] when alcohol is 0.51 to 2.03 & pH >= -0.97 & volatile.acidity >= -0.55 &
residual.sugar >= -0.58 & chlorides >= -1.1 & density <  -1.3
##           high [.82 .00 .18] when alcohol <  0.51 & pH <  -1.75 & volatile.acidity <  -0.94 & sulphates <  -0.84
##         medium [.18 .18 .65] when alcohol is 0.51 to 2.03 & pH >= -0.97 & volatile.acidity >= -0.55 &
residual.sugar <  -0.58
##         medium [.18 .14 .68] when alcohol is 0.51 to 2.03 & pH >= -0.97 & volatile.acidity >= -0.55 &
residual.sugar >= -0.58 & chlorides >= -1.1 & density >= -1.3
##         medium [.30 .02 .68] when alcohol is 0.51 to 2.03 & pH >= -0.97 & volatile.acidity is -1.20 to -0.55
##         medium [.20 .05 .75] when alcohol is 0.51 to 2.03 & pH <  -0.97
##         medium [.20 .01 .78] when alcohol <  0.51 & pH >= -1.75 & volatile.acidity <  -0.94
##         medium [.12 .00 .88] when alcohol is 0.51 to 2.03 & pH >= -0.97 & volatile.acidity <  -1.20
& citric.acid <  -0.18
##         medium [.11 .00 .89] when alcohol <  0.51 & pH <  -1.75 & volatile.acidity <  -0.94
& sulphates >= -0.84
##         medium [.06 .05 .89] when alcohol <  0.51 & volatile.acidity >= -0.94
```

Así como hacer un plot del árbol de clasificación resultante:

```
## realizamos plot del árbol resultante
rpart.plot(modelCARTIt2, type = 0, clip.right.lab = FALSE, branch = .3, under = TRUE, box.palette =
"auto", tweak = 1.0,  fallen.leaves=FALSE )
```



En conclusión, obtenemos un modelo predictivo que nos ofrece una calidad estimada del 76.55%, inferior que la lograda con el modelo de clasificación para vinos tintos. En cualquier caso, al igual que con los vinos tintos, este modelo cumplirá suficientemente con los objetivos esperados para este proyecto analítico.

3.5 Representación y visualización de los datos – *data visualization*

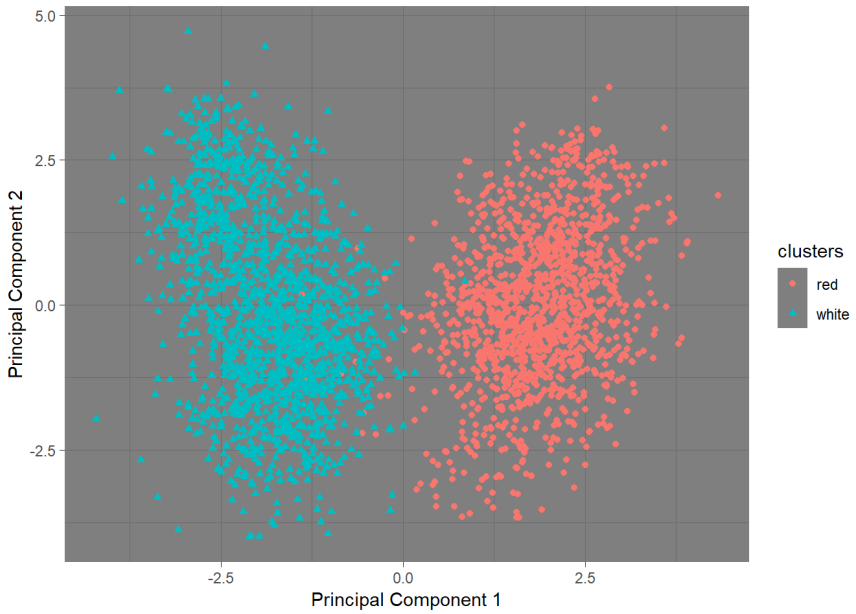
Previo a analizar el conjunto de resultados obtenidos, se hará uso de estrategias de visualización de los datos con el fin de sintetizar la mayor cantidad de información que describa los datos, de tal forma que pueda servir de apoyo a la obtención de conclusiones.

Como sabemos, nuestro objetivo es comprender en que medida las distintas características del vino influyen sobre la calidad de este, por ello, a continuación, se llevará a cabo una exploración visual enfocándonos en esta área de descubrimiento.

Dado que a lo largo del presente trabajo hemos estado enfocados en esta misma área de análisis, ya se han presentado algunos diagramas que son interesantes, a continuación, se incluirán algunos de los diagramas ya vistos, y alguno nuevo.

Partimos de un conjunto de datos que contiene observaciones para vinos tintos y blancos, y en primer lugar queremos entender si existen diferencias en las características de estos, para ello se hizo una represión multidimensional aplicando la técnica **PCA**:

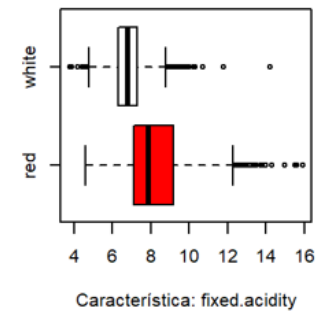
```
# Inicializo el dataset para realizar el plot PCA y lo ejecuto
wine_quality = wine_quality.clean.scaled[1:11]
plotClusterPCA(wine_quality, wine_quality.clean.scaled, "type")
```

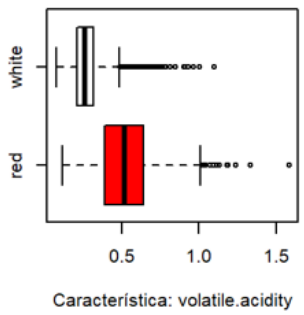


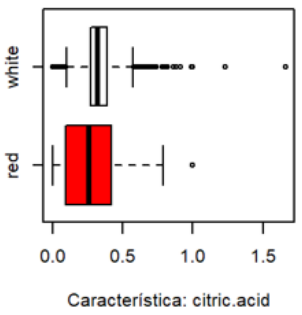
Confirmando como el conjunto de observaciones pueden ser sectorizadas en base al tipo de vino.

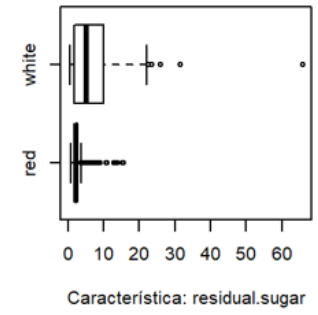
Haciendo uso de *boxplots* echamos un vistazo a como se distribuyen las características de cada tipo de vino. Este tipo de gráfica nos facilita una visualización muy condensada y sencilla de parte de los estadísticos descriptivos más relevantes, algo que nos facilita enormemente comprender las diferencias en las características entre los grupos de vino analizados.

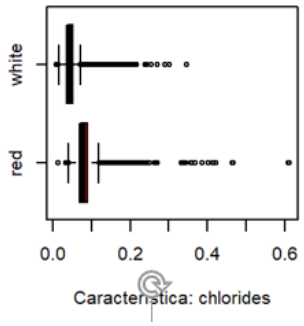
```
wine_quality = wine_quality.base

# Visualización caracerísticas por tipo de vino
colnames = names(wine_quality[1:11]);
plotBoxplotMultiple(wine_quality, colnames)
```

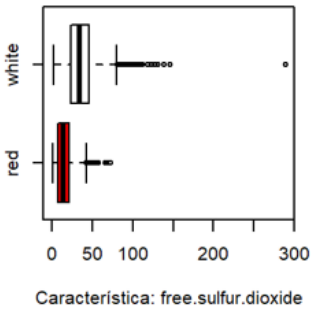


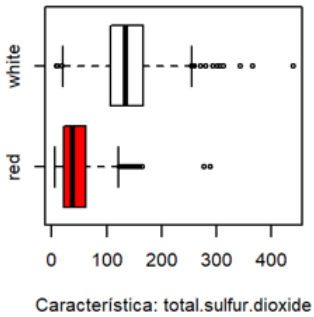


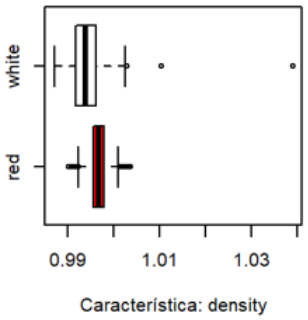


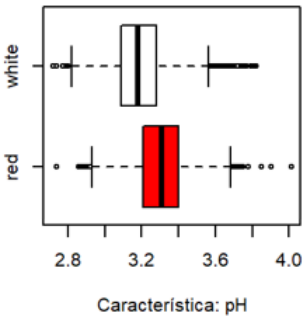


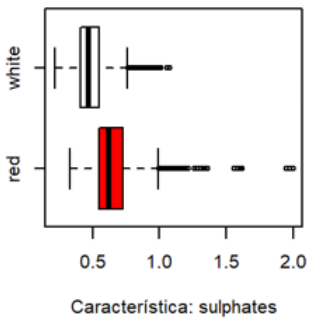


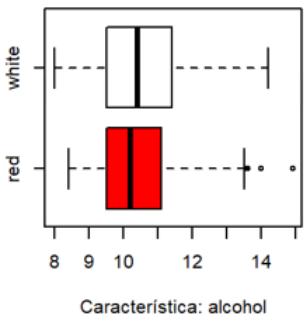






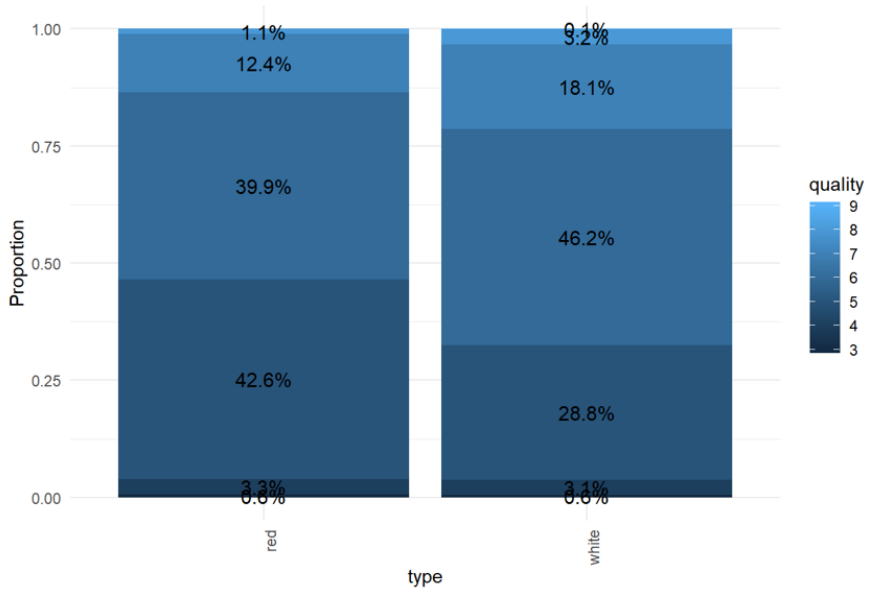






Estamos interesados en saber si la calidad se ve influenciada por el tipo de vino, y para ello se hizo uso de un plot bi-variable, evaluando la proporción de observaciones de cada calidad existentes para cada tipo de vino:

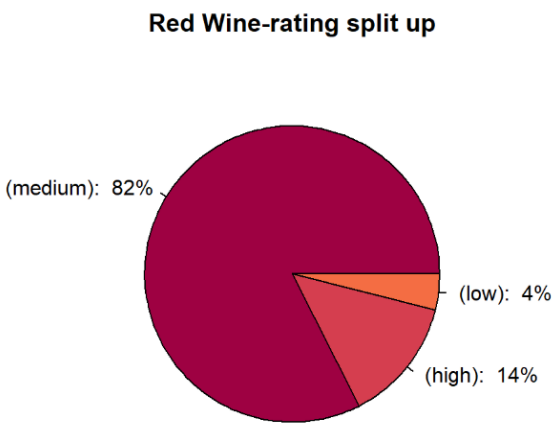
```
# mostramos una gráfica bi variable por tipo de vino y calidad
barplotBiVariable(wine_quality.clean.scaled, "type", "quality")
```

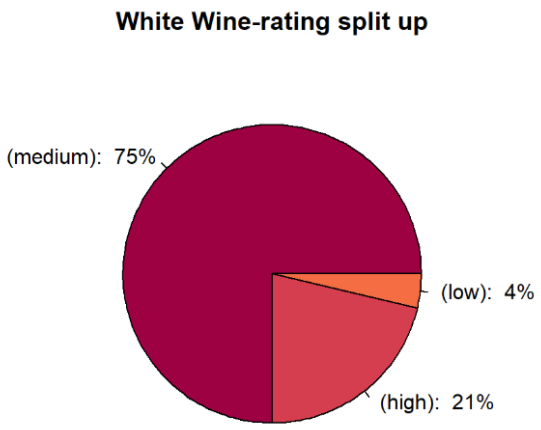


De lo que se concluyó, que para los vinos blancos se observaban una proporción mayor de vinos con calidades más altas.

A continuación, podemos ver el detalle de la proporción para cada nivel de calidad.

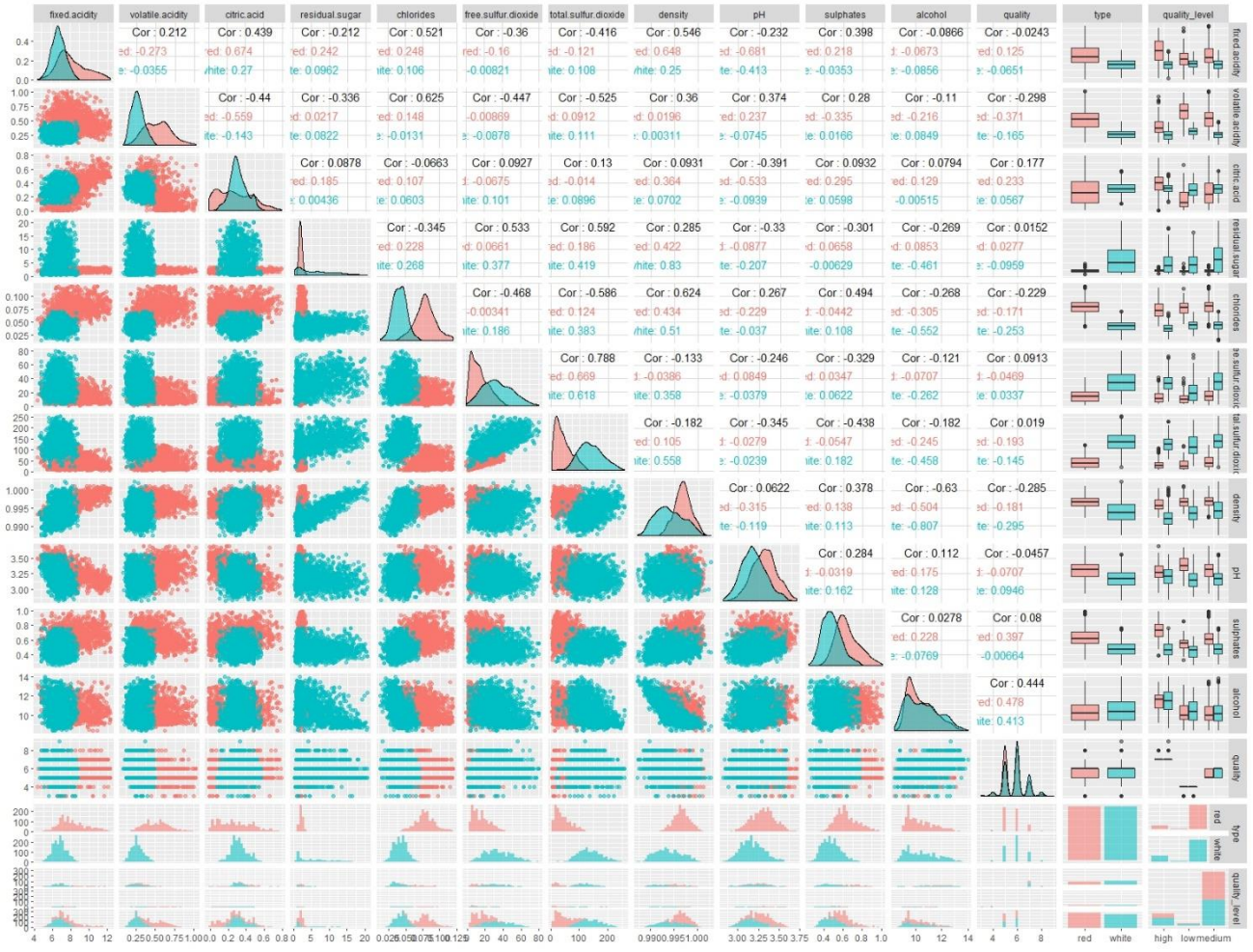
```
wine_quality = wine_quality.clean.scaled

## Generaremos un gráfico de sectores con la función pie.
pie_values <- sort(table(wine_quality[wine_quality$type=="red",]$quality_level), decreasing = TRUE)
pct = round(pie_values/sum(pie_values)*100)
lbls = c('', '', '')
lbls = paste(lbls,names(pie_values),sep=" (")
lbls = paste(lbls,"): ",sep="")
lbls = paste(lbls, pct) ## añadir porcentajes a las label
lbls = paste(lbls,"%",sep="") ## añadir caracer % a las label
pie(pie_values, labels = lbls, main="Red Wine-rating split up", col=brewer.pal(10, "Spectral"))
## Generaremos un gráfico de sectores con la función pie.
pie_values <- sort(table(wine_quality[wine_quality$type=="white",]$quality_level), decreasing = TRUE)
pct = round(pie_values/sum(pie_values)*100)
lbls = c('', '', '')
lbls = paste(lbls,names(pie_values),sep=" (")
lbls = paste(lbls,"): ",sep="")
lbls = paste(lbls, pct) ## añadir porcentajes a las label
lbls = paste(lbls,"%",sep="") ## añadir caracer % a las label
pie(pie_values, labels = lbls, main="White Wine-rating split up", col=brewer.pal(10, "Spectral"))
```





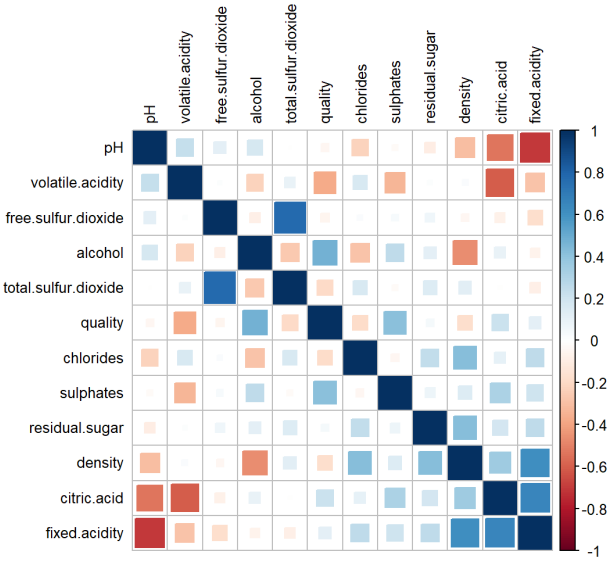
Queremos entender que características del vino influencian la calidad del mismo, para lo que nos apoyamos en gráficos que facilitan la visualización de las correlaciones entre variables.

```
# visualización múltiple
ggpairs(wine_quality.clean, aes(colour = type, alpha = 0.4))
# plot de correlación para vino tinto
wine_quality = wine_quality_red.clean.scaled
plotCorrelation(wine_quality)
# plot de correlación para vino blanco
wine_quality = wine_quality_white.clean.scaled
plotCorrelation(wine_quality)
```



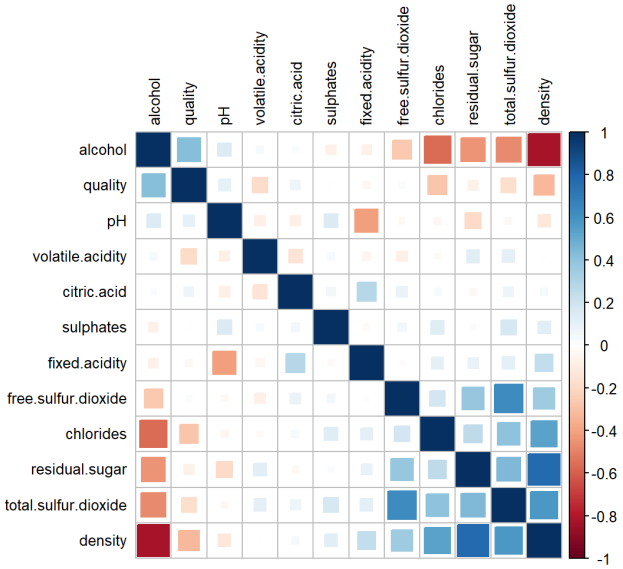
Correlaciones

Vino Tinto



Correlaciones

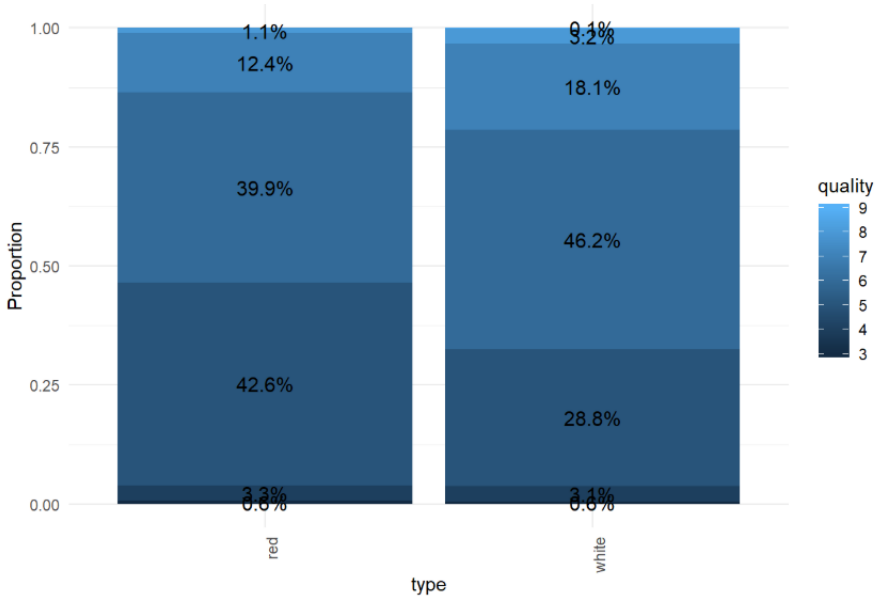
Vino blanco



Como habíamos observado previamente, existen algunas características que están más significativamente correladas con la calidad, si retomamos las tablas resumen de correlaciones para vino tinto que incluimos en el apartado **0 #** *# construye un diagrama de barras de frecuencias para 2 variables*

```
barplotBiVariable = function(dataset, att1, att2, labsize=4) {
  plotdata = dataset %>%
      dplyr::group_by(dataset[,att1], dataset[,att2]) %>%
      dplyr::summarize(n = n()) %>%
      dplyr::mutate(pct = n/sum(n),
             lbl = scales::percent(pct))

  names(plotdata) = c("VAR1", "VAR2", "n", "pct", "lbl")

  ## plot
  plot1 = ggplot(data = plotdata,
                 aes(x = VAR1,
                 y = pct,
                 fill = VAR2,
                 cumulative = TRUE)) +
                 geom_col() +
          labs(y = "Proportion", x=att1, fill=att2, title="title")  +
          geom_text(aes(label = lbl), size=labsize,
                    position = position_stack(vjust = 0.5)) +
          theme_minimal()+
```

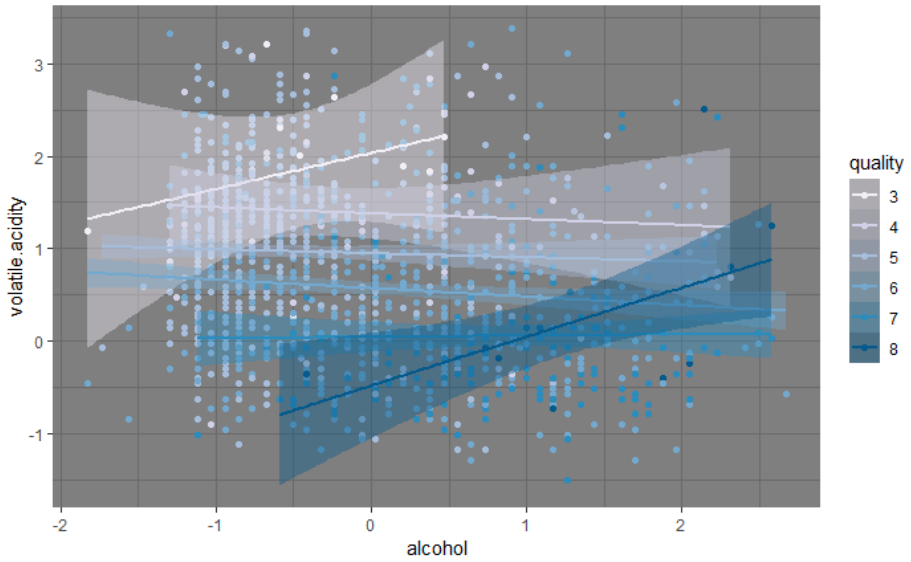
```
          theme(axis.text.x = element_text(angle = 90, hjust = 1))
  return(plot1)
}
# mostramos una gráfica bi variable por tipo de vino y calidad
plot1 = barplotBiVariable(wine_quality.clean.scaled, "type", "quality")
plot1
```



Correlación, podremos analizar tanto como se ve influenciada la calidad por cada característica del vino, como la forma en que cada variable se ve influenciada por el resto.

Por ejemplo, en el siguiente gráfico estudiamos un poquito más la relación entre las variables que mayor correlación tenían con la calidad para el **vino tinto**, *volatile.acidity, alcohol*:

```
# función para pintar un plot con la regresión lineal entre dos variables, agrupado por quality
plotQualityRegression = function(wine_quality, x_att, y_att, title="") {
  ggplot(wine_quality, aes(x = wine_quality[,x_att], y = wine_quality[,y_att], color = quality, fill
= quality)) +
          geom_point() +
          geom_smooth(method = 'lm') +
          scale_fill_brewer(palette = "PuBu") +
          scale_color_brewer(palette = "PuBu") +
          xlab(x_att) + ylab(y_att) +
          labs(title = title) +
          theme_dark()
}
# análisis relación alcohol, volatile.acidity y quality - vino tinto
wine_quality = data.frame(wine_quality_red.clean.scaled)
wine_quality$quality = as.factor(wine_quality$quality)
plotQualityRegression(wine_quality,  "alcohol", "volatile.acidity", "Red Whine")
```

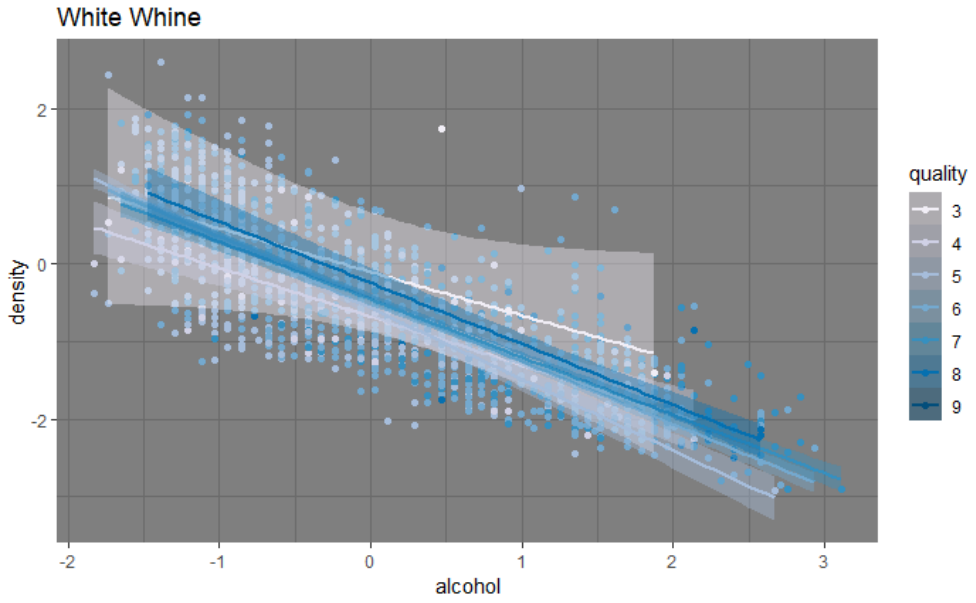


Observando la gráfica vemos como los vinos de baja calidad se localizan en la la esquina superior izquierda, donde el *alcohol* es bajo y *volatile.acidity* es alta; y sin embargo los vinos de alta calidad se localizan por la esquina inferior derecha; aunque si observamos la correlación entre el alcohol y *volatile.acidity*, observamos que es muy baja.

Pero esto no implica que un valor de *alcohol* alto y un valor de *volatile.acidity* bajo de lugar a una alta calidad, ya que la calidad se ve influenciada por más características, estas en concreto definen una correlación más reseñable.

Si ahora analizamos la relación entre las variables *las variables que mayor correlación tenían con la calidad para el* **vino blanco**, *density, según toda la información previa sabemos que* obtendremos una correlación inversa muy marcada:

```
# análisis relación alcohol, volatile.acidity y quality - vino blanco
wine_quality = data.frame(wine_quality_white.clean.scaled)
wine_quality$quality = as.factor(wine_quality$quality)
plotQualityRegression(wine_quality,  "alcohol", "density", "White Whine")
```

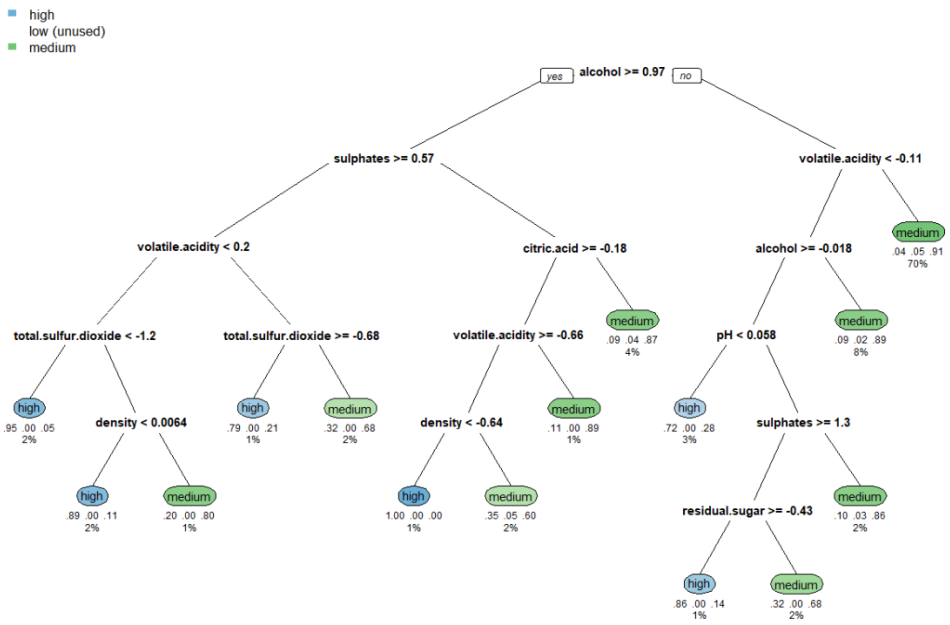


Si ahora, hacemos uso de la información del modelo de clasificación construido, vemos que para **vino tinto** contamos con la siguiente información:

- características usadas:
  alcohol, *citric.acid*, *density*, *pH*, *residual.sugar*, *sulphates*, *total.sulfur.dioxide*, *volatile.acidity*.
- características excluidas:
  free.sulfur.dioxide (lógico, ya que mantiene una correlación bastante alta con *total.sulfur.dióxido*, que si está incluido), *chlorides*, *fixed.acidity* (lógico, ya que mantiene una correlación bastante alta con *pH*, que si está incluido).

En el siguiente diagrama se muestra el árbol de clasificación obtenido:

```
## realizamos plot del árbol de clasificación - vino tinto
rpart.plot(modelCARTIt1, type = 0, clip.right.lab = FALSE, branch = .3, under = TRUE, box.palette =
"auto", tweak = 1.0,  fallen.leaves=FALSE )
```

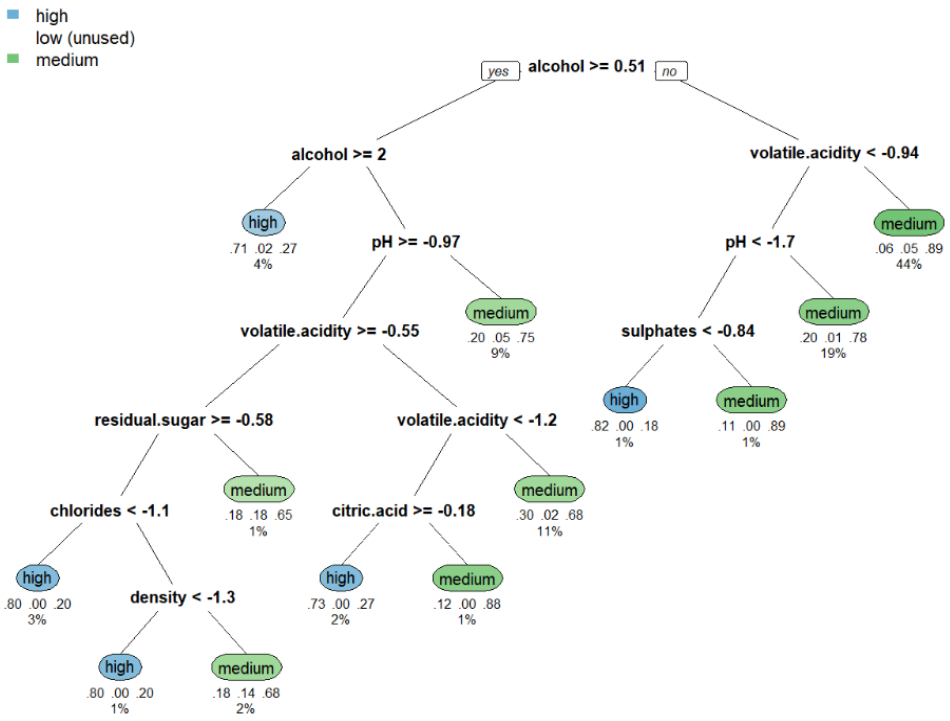


Y si revisamos el árbol de clasificación para vino blanco contamos con:

- características usadas:
  alcohol, *chlorides*, *citric.acid, density, pH, residual.sugar, sulphates, volatile.acidity*.
- características excluidas:
  fixed.acidity (mantiene una correlación significativa con el *pH, que si está incuido*), *free.sulfur.dioxide* (mantiene una correlación significativa con el *residual.sugar, que si está incuido*), *total.sulfur.dioxide* (mantiene una alta correlación con *density* que si está incluido)

En el siguiente diagrama se muestra el árbol de clasificación obtenido:

```
## realizamos plot del árbol de clasificación - vino blanco
rpart.plot(modelCARTIt2, type = 0, clip.right.lab = FALSE, branch = .3, under = TRUE, box.palette =
"auto", tweak = 1.0,  fallen.leaves=FALSE )
```



En ambos casos, el detalle de las reglas de clasificación obtenidas, que podemos encontrar en el apartado 3.4.3.3.1 MODELO CLASIFICACIÓN NIVEL DE CALIDAD DE VINO TINTO y 3.4.3.3.2 MODELO CLASIFICACIÓN NIVEL DE CALIDAD DE VINO BLANCO, nos dará una idea muy clara como influencian las distintas características a la calidad obtenida.

3.6 Resolución del problema – *problem resolution*

Finalmente, llegados a este punto, toca hacer una evaluación del conjunto de actividades realizadas con el fin de interpretar los resultados y llegar a conclusiones.

A lo largo de los apartados previos se han llevado a cabo las fases principales de un proyecto analítico.

- En la etapa de **definición del problema** hemos acotado el objetivo del proyecto analítico, estableciéndose como objetivo principal el lograr adquirir una idea de cómo las distintas características asociadas a un vino llegan a influenciar la calidad de este.

- En la etapa de **selección, recuperación y persistencia de los datos**, se han obtenido los distintos conjuntos de datos desde su fuente de datos original, y se han llevado a cabo tareas de integración y limpieza.

  En la integración y selección de los datos se han considerado tanto vinos tintos, como blancos, asegurándonos de contar con el mismo número de observaciones para ambos tipos de vino para contar con una muestra equitativa; además se han realizado tareas de limpieza de valores identificados como *missing values* y *outliers*, aplicando el método *missForest* para la imputación de valores.

  Adicionalmente se han realizado algunas transformaciones, normalizando y estandarizando alguno de los valores, con el fin de preparar los datos para los distintos análisis a ser realizados.

- En la etapa de **análisis de los datos** en primer lugar se han seleccionado los datos sobre los que se van a realizar los análisis, para posteriormente llevarse a cabo verificaciones de la normalidad y homogeneidad en dichos datos, y comparaciones entre grupos, con el fin de determinar si existen diferencias significativas en la calidad del vino en función de si es tinto o blanco, concluyéndose que, si existen diferencias significativas, contando el vino blanco con mayor calidad que el vino tinto.

  A continuación, se ha analizado la correlación de las distintas características respecto a la calidad del vino, así como la correlación de las características entre sí, obteniéndose el conjunto de características que mayor correlación tienen con la calidad, y entre ellas.

  Con el fin de adentrarse más en el conocimiento de como las distintas características influencian la calidad del vino, así como de lograr obtener un modelo de predicción de la calidad para nuevas observaciones, se ha construido un modelo de clasificación que logra con una calidad decente ambos objetivos.

- Para completar la etapa de análisis, en la etapa de **representación y visualización de los datos**, se han revisado y completado los análisis visuales, obteniéndose un conjunto de gráficos que aprovechan la capacidad visual humana para la detección de patrones y tendencias.

  En esta etapa han podido ser organizado, resumido y confirmado el conocimiento adquirido a lo largo de las etapas previas.

3.7 Código

Disponible en https://github.com/mmcolino/WineQualityAnalysis

4 Anexos

4.1 Bibliografía

H. Jarman, K. (2013). *The Art of Data Analysis - How to Answer Almost Any Question Using Basic Statistics.* New Jersey: Wiley.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining. Conchepts and Techniques.* Waltham, USA: Morgan Kaufmann Publishers.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning (with Applications in R).* New York: Springer.

Osborne, J. W. (2010). Data Cleaning Basics: Best Practices in Dealing with Extreme Scores. Newborn and Infant Nursing Reviews. 10 (1): pp. 1527-3369.

P. Cortez, A. C. (2009). Modeling wine preferences by data mining from physicochemical properties. . *Elsevier*, 547-553.

Squire, M. (2015). *Clean Data.* Packt Publishing Ltd.