

# Práctica 2: Limpieza y validación de los datos

Daniel Sánchez Ambite

01/01/2022

## Contents

Descripción del dataset . . . . .	1
Integración y selección de los datos de interés a analizar. . . . .	2
Limpieza de los datos . . . . .	3
Análisis de los datos . . . . .	7
Comprobación de la normalidad . . . . .	22
Modelo para predecir la calidad del vino . . . . .	23
Modelo para predecir el color del vino . . . . .	27
Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema? . . . . .	29
Repositorio y video . . . . .	29

## Descripción del dataset

El dataset elegido era una de las opciones que se planteaban en el enunciado de la práctica, si bien en este caso he decidido coger la versión completa disponible en <https://archive.ics.uci.edu/ml/datasets/Wine+Quality>. Este dataset dispone de 1599 instancias de vino tinto y 4898 de vino blanco y contiene 11 atributos más el output. En su descripción podemos leer que no tiene valores nulos y que algunos de sus atributos están correlacionados por lo que puede ser interesante la selección de atributos.

Las variables del dataset son las siguientes:

**fixed.acidity**: la acidez fija del vino.

**volatile.acidity**: acidez volátil.

**citric.acid**: ácido cítrico.

**residual.sugar**: azúcar residual.

**chlorides**: cloritos.

**free.sulfur.dioxide**: dióxido de azufre libre.

**total.sulfur.dioxide**: dióxido de azufre total.

**density**: densidad.

**pH**: valor del PH (mide el nivel de acidez)

**sulphates**: sulfatos.

**alcohol**: cantidad de alcohol en volumen.

**quality**: calidad del vino (este será el valor que intentaremos estimar con el resto de datos)

**color**: el color del vino.

## ¿Por qué es importante y qué pregunta/problema pretende responder?

El objetivo de nuestro análisis va a consistir en intentar estimar la calidad de los vinos en función de sus propiedades, además al disponer de dos data set diferenciados por dos tipos de vino se van a unir para analizar si es posible además identificar el tipo de vino blanco o rojo en función de sus propiedades.

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

```
library(ggplot2)
library(dplyr)

## 
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union

# Loading data.
white <- read.csv("./datasetwine/winequality-white.csv", header = TRUE, sep = ";")
red <- read.csv("./datasetwine/winequality-red.csv", header = TRUE, sep = ";")
white$color <- "white"
#adding the color before merging
red$color <- "red"
# merging vertically
wine <- rbind(white,red)
summary(wine)
```

```
##   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    density
##   Min. :0.00900    Min. : 1.00      Min. : 6.0       Min. :0.9871
##   1st Qu.:0.03800    1st Qu.:17.00     1st Qu.:77.0      1st Qu.:0.9923
##   Median :0.04700    Median :29.00      Median :118.0     Median :0.9949
##   Mean   :0.05603    Mean   :30.53      Mean   :115.7     Mean   :0.9947
##   3rd Qu.:0.06500    3rd Qu.:41.00      3rd Qu.:156.0     3rd Qu.:0.9970
##   Max.   :0.61100    Max.   :289.00     Max.   :440.0      Max.   :1.0390
##   pH            sulphates      alcohol      quality
##   Min. : 2.720    Min. :0.2200    Min. : 8.00    Min. :3.000
##   1st Qu.:3.110    1st Qu.:0.4300    1st Qu.: 9.50   1st Qu.:5.000
##   Median :3.210    Median :0.5100    Median :10.30   Median :6.000
##   Mean   :3.219    Mean   :0.5313    Mean   :10.49   Mean   :5.818
##   3rd Qu.:3.320    3rd Qu.:0.6000    3rd Qu.:11.30   3rd Qu.:6.000
##   Max.   :4.010    Max.   :2.0000    Max.   :14.90   Max.   :9.000
##   color
##   Length:6497
##   Class :character
```

```
##  Mode :character
##
##
##
```

En la presentación del dataset nos dice que no tiene valores nulos y que todas las variables son numéricas. En este caso nosotros le hemos añadido la variable color antes de unir los dos datasets para poder diferenciar entre vinos tintos y blancos. Comprobaremos no obstante que es cierto que no existen valores nulos.

## Limpieza de los datos

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

```
if (!require('skimr')) install.packages('skimr'); library('skimr')

## Loading required package: skimr
skim(wine)
```

Table 1: Data summary

Name	wine
Number of rows	6497
Number of columns	13
<hr/>	
Column type frequency:	
character	1
numeric	12
<hr/>	
Group variables	None

### Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
color	0	1	3	5	0	2	0

### Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
fixed.acidity	0	1	7.22	1.30	3.80	6.40	7.00	7.70	15.90	
volatile.acidity	0	1	0.34	0.16	0.08	0.23	0.29	0.40	1.58	
citric.acid	0	1	0.32	0.15	0.00	0.25	0.31	0.39	1.66	
residual.sugar	0	1	5.44	4.76	0.60	1.80	3.00	8.10	65.80	
chlorides	0	1	0.06	0.04	0.01	0.04	0.05	0.06	0.61	
free.sulfur.dioxide	0	1	30.53	17.75	1.00	17.00	29.00	41.00	289.00	
total.sulfur.dioxide	0	1	115.74	56.52	6.00	77.00	118.00	156.00	440.00	
density	0	1	0.99	0.00	0.99	0.99	0.99	1.00	1.04	
pH	0	1	3.22	0.16	2.72	3.11	3.21	3.32	4.01	
sulphates	0	1	0.53	0.15	0.22	0.43	0.51	0.60	2.00	
alcohol	0	1	10.49	1.19	8.00	9.50	10.30	11.30	14.90	
quality	0	1	5.82	0.87	3.00	5.00	6.00	6.00	9.00	

Se puede observar que no falta ningún valor en las variables como ya anunciaba el data set con el paquete de `skimr` nos muestra que no existen valores vacíos y que todos las variables son numéricas salvo la de color que hemos introducido nosotros. Los valores nulos también se pueden comprobar de forma nativa en R con:

```
# Numeric values
colSums(is.na(wine))

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

# Looking for empty strings
colSums(wine=="")
```

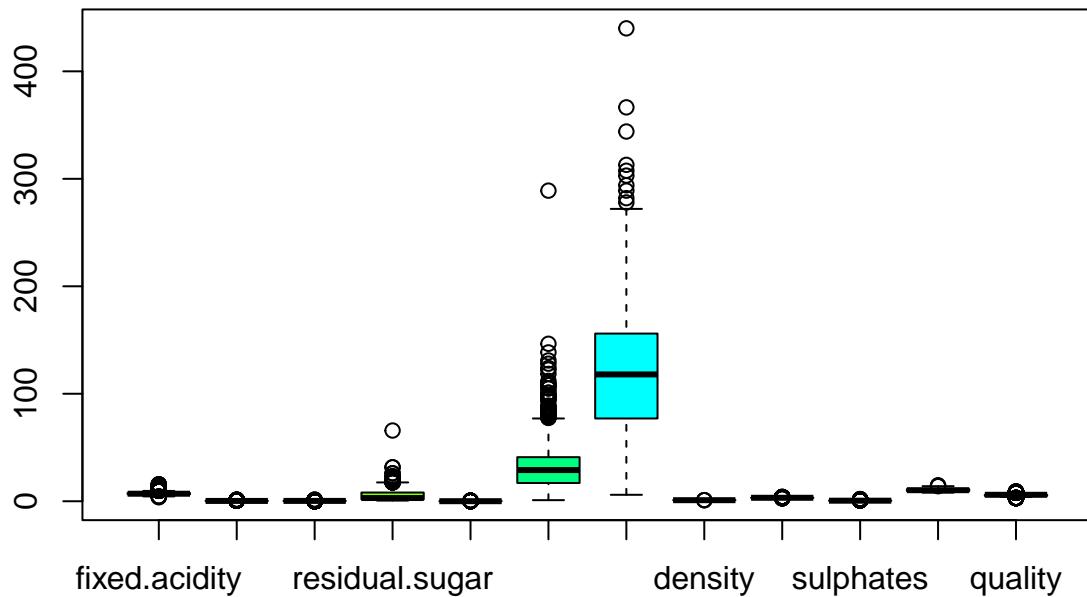
  

```
##      fixed.acidity      volatile.acidity      citric.acid
##                 0                  0                  0
##      residual.sugar      chlorides      free.sulfur.dioxide
##                 0                  0                  0
##      total.sulfur.dioxide      density          pH
##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

### Identificación y tratamiento de valores extremos

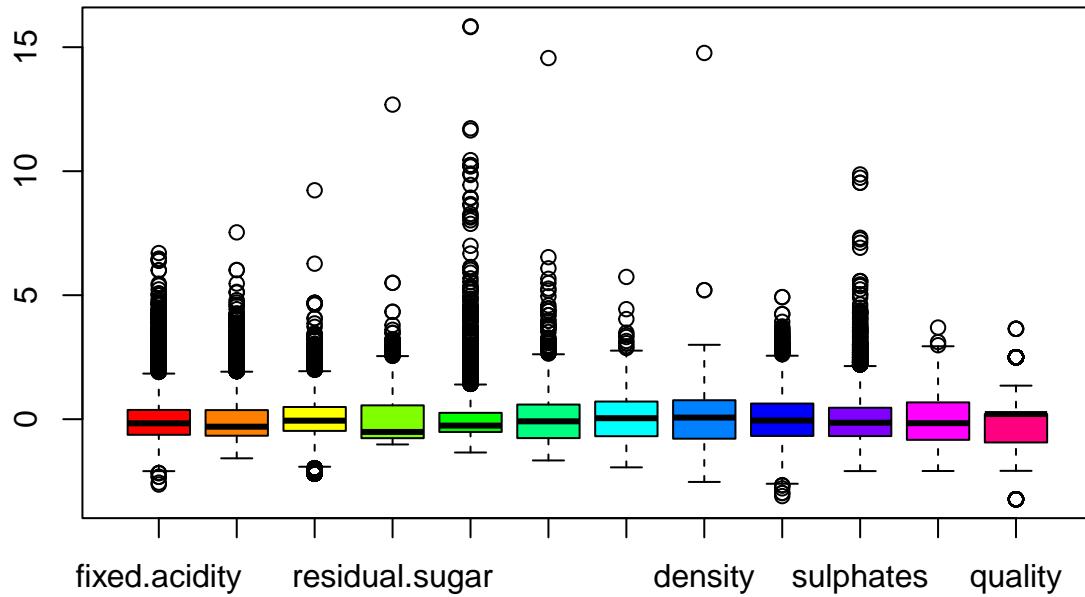
Comenzamos comprobando los outliers que existen en el dataset para ello vamos a usar boxplots que nos lo muestra de una manera visual y además R almacena los outliers en la variable `$out`.

```
#removing the color column
wine_n <- wine
wine_n$color <- NULL
g_caja <- boxplot(wine_n, col = rainbow(ncol(wine_n)))
```



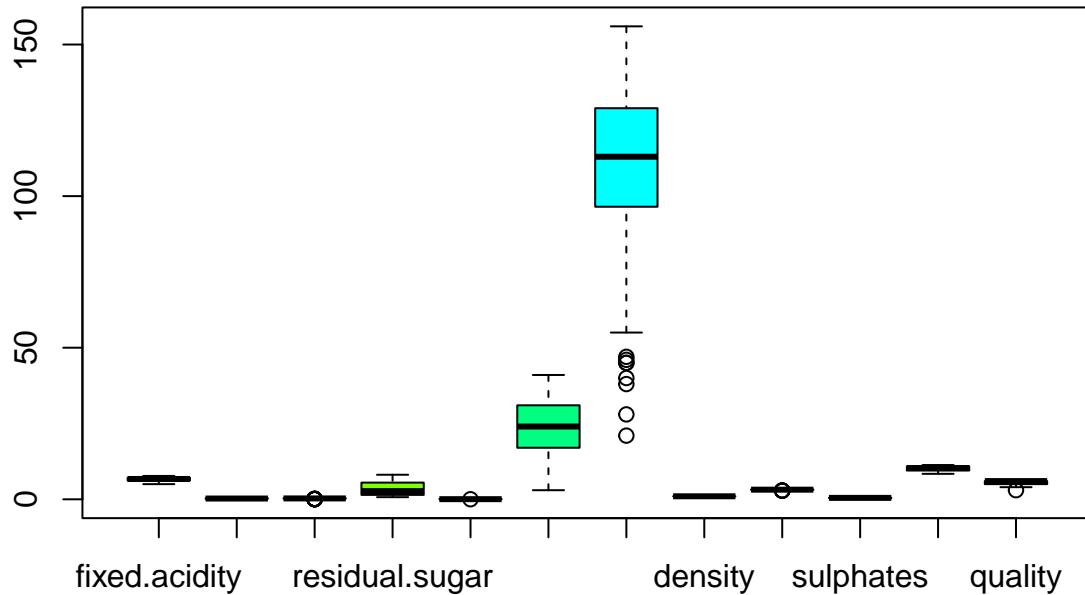
En este gráfico se puede observar como las variables free.sulfur.dioxide y total.sulfur.dioxide no se encuentran en la misma escala por lo que desvirtuan el gráfico, podemos escalar o normalizar los datos para reducir el problema:

```
# use the function scale to normalize the data
scaled_wine <- scale(wine_n)
g_caja_e <- boxplot(scaled_wine, col = rainbow(ncol(wine_n)))
```



Podemos ver como esta normalización distorsiona bastante los valores así que no la vamos a usar y vamos a quitar los outliers del dataset sin normalizar:

```
#remove the outliers and change them for NA
for(i in 1:ncol(wine_n)){
  wine_n[, i][wine_n[, i] > quantile(wine_n[, i], 3/4)] = NA
}
# Remove NA values
clean_wine <- wine_n[complete.cases(wine_n),]
# Check the new box
clean_box <- boxplot(clean_wine, col= rainbow(ncol(wine_n)))
```



```
clean_box$out
```

```
## [1] 0.030 0.040 0.100 0.040 0.100 0.100 0.100 0.100 0.100 0.100 0.100 0.090
## [11] 0.100 0.017 46.000 45.000 45.000 28.000 21.000 40.000 47.000 38.000
## [21] 2.790 2.880 2.880 2.880 2.870 3.000
```

```
nrow(clean_wine)
```

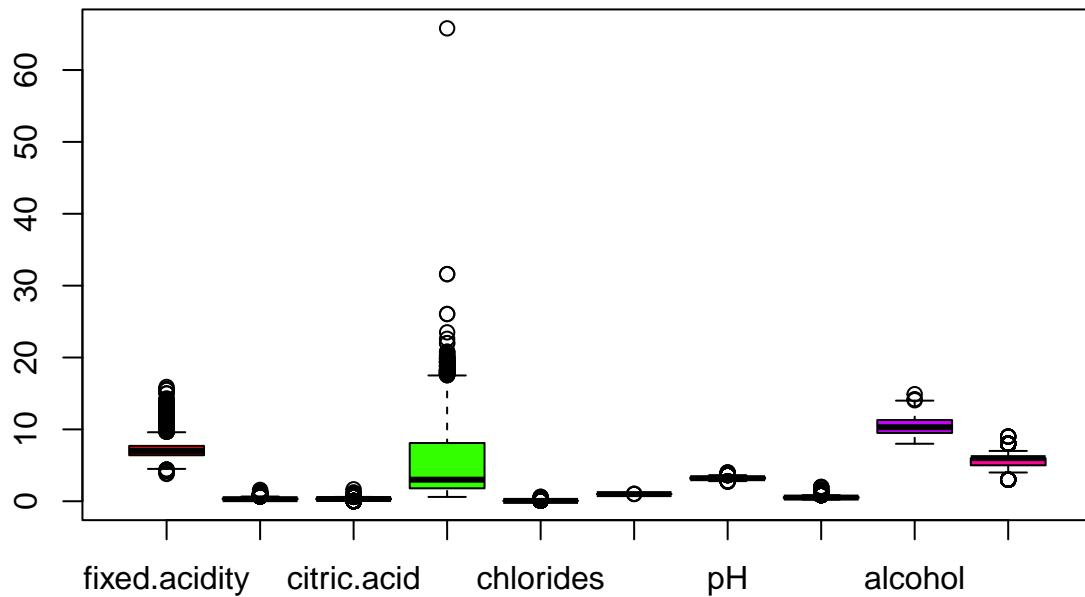
```
## [1] 439
```

Con esta aproximación si eliminamos todos los outliers de todas las columnas acabamos con un dataset muy reducido por lo que vamos a intentar limpiar solo los outliers de las columnas que distorsionen realmente el resultado si es necesario.

## Análisis de los datos

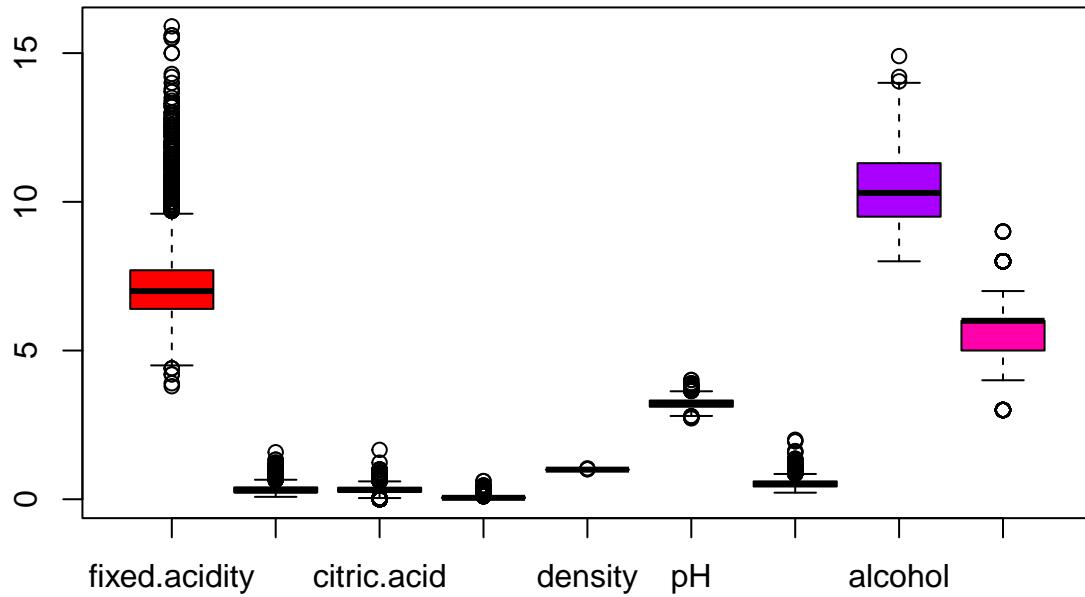
Vamos a continuar con el dataset con outliers pero vamos a quitar las dos columnas fuera del rango para ver como se comportan el resto

```
library("dplyr")
#restorting wine_n
wine_n <- wine
wine_n$color <- NULL
sub_wine1 <- select(wine_n, -c(free.sulfur.dioxide, total.sulfur.dioxide))
boxplot(sub_wine1, col= rainbow(ncol(sub_wine1)))
```



Observamos la caja verde que tiene una varianza superior al resto esta caja se corresponde con el azúcar residual seguido del alcohol que es la caja morada. Para poder observar con más detalle como se comportan el resto vamos a eliminar el azúcar residual:

```
sub_wine2 <- select(sub_wine1, -c(residual.sugar))
boxplot(sub_wine2, col= rainbow(ncol(sub_wine2)))
```

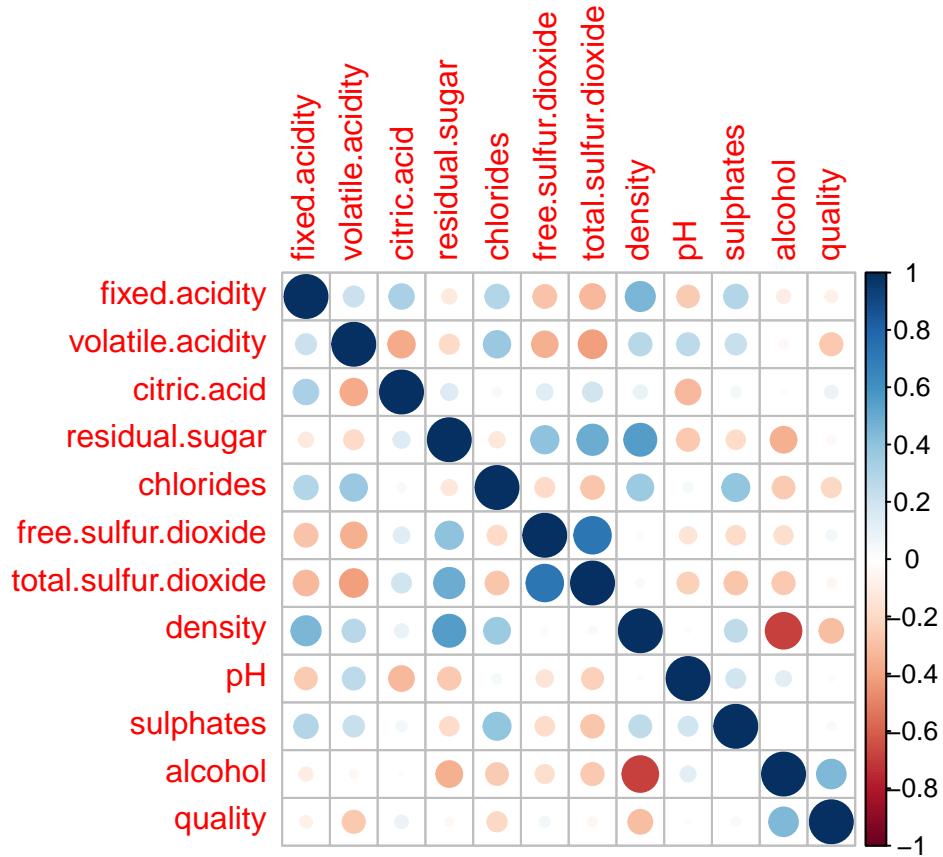


Y aquí podemos observar como la varianza de la ácidez y el alcohol son las más significativas después de la del azúcar y además vemos como la caja rosa que es la calidad tiene el borde superior coincidente con la mediana.

Se va a analizar la correlación entre las distintas variables del dataset:

```
#knitr::opts_chunk$set(fig.width=40, fig.height = 40, fi)
if (!require('corrplot')) install.packages('corrplot'); library('corrplot')

## Loading required package: corrplot
## corrplot 0.92 loaded
var_cor <- cor(wine_n)
#show visual correlation
corrplot(var_cor)
```



```
var_cor
```

```
##          fixed.acidity volatile.acidity citric.acid residual.sugar
## fixed.acidity      1.00000000   0.21900826  0.32443573 -0.11198128
## volatile.acidity   0.21900826   1.00000000 -0.37798132 -0.19601117
## citric.acid        0.32443573 -0.37798132   1.00000000  0.14245123
## residual.sugar     -0.11198128 -0.19601117  0.14245123  1.00000000
## chlorides          0.29819477  0.37712428  0.03899801 -0.12894050
## free.sulfur.dioxide -0.28273543 -0.35255731  0.13312581  0.40287064
## total.sulfur.dioxide -0.32905390 -0.41447619  0.19524198  0.49548159
## density            0.45890998  0.27129565  0.09615393  0.55251695
## pH                 -0.25270047  0.26145440 -0.32980819 -0.26731984
## sulphates          0.29956774  0.22598368  0.05619730 -0.18592741
## alcohol             -0.09545152 -0.03764039 -0.01049349 -0.35941477
## quality            -0.07674321 -0.26569948  0.08553172 -0.03698048
##          chlorides free.sulfur.dioxide total.sulfur.dioxide
## fixed.acidity       0.29819477  -0.28273543  -0.32905390
## volatile.acidity    0.37712428  -0.35255731  -0.41447619
## citric.acid         0.03899801   0.13312581  0.19524198
## residual.sugar      -0.12894050   0.40287064  0.49548159
## chlorides           1.00000000  -0.19504479  -0.27963045
## free.sulfur.dioxide -0.19504479   1.00000000  0.72093408
## total.sulfur.dioxide -0.27963045   0.72093408  1.00000000
## density             0.36261466   0.02571684  0.03239451
## pH                  0.04470798  -0.14585390  -0.23841310
## sulphates           0.39559331  -0.18845725  -0.27572682
## alcohol             -0.25691558  -0.17983843  -0.26573964
```

```

## quality      -0.20066550      0.05546306     -0.04138545
## density      density      pH      sulphates      alcohol
## fixed.acidity 0.45890998 -0.25270047  0.299567744 -0.095451523
## volatile.acidity 0.27129565  0.26145440  0.225983680 -0.037640386
## citric.acid   0.09615393 -0.32980819  0.056197300 -0.010493492
## residual.sugar 0.55251695 -0.26731984 -0.185927405 -0.359414771
## chlorides      0.36261466  0.04470798  0.395593307 -0.256915580
## free.sulfur.dioxide 0.02571684 -0.14585390 -0.188457249 -0.179838435
## total.sulfur.dioxide 0.03239451 -0.23841310 -0.275726820 -0.265739639
## density      1.00000000  0.01168608  0.259478495 -0.686745422
## pH           0.01168608  1.00000000  0.192123407  0.121248467
## sulphates    0.25947850  0.19212341  1.000000000 -0.003029195
## alcohol       -0.68674542  0.12124847 -0.003029195  1.000000000
## quality      -0.30585791  0.01950570  0.038485446  0.444318520
## quality
## fixed.acidity -0.07674321
## volatile.acidity -0.26569948
## citric.acid   0.08553172
## residual.sugar -0.03698048
## chlorides      -0.20066550
## free.sulfur.dioxide 0.05546306
## total.sulfur.dioxide -0.04138545
## density      -0.30585791
## pH           0.01950570
## sulphates    0.03848545
## alcohol       0.44431852
## quality      1.00000000

```

La correlación más alta sucede entre free.sulfur.dioxide y total.sulfur.dioxide con un valor de 0.72. También destaca la alta correlación entre las variables density y alcohol con un valor de -0.6867 es negativa por lo que al aumentar una variable disminuye la otra. Las variables density con residual sugar tienen una correlación de 0.552 y en este rango o un poco menor también están los pares total.sulfur.dioxide y volatile.acidity, density y fixed.acidity, total.sulfur.dioxide y residual sugar.

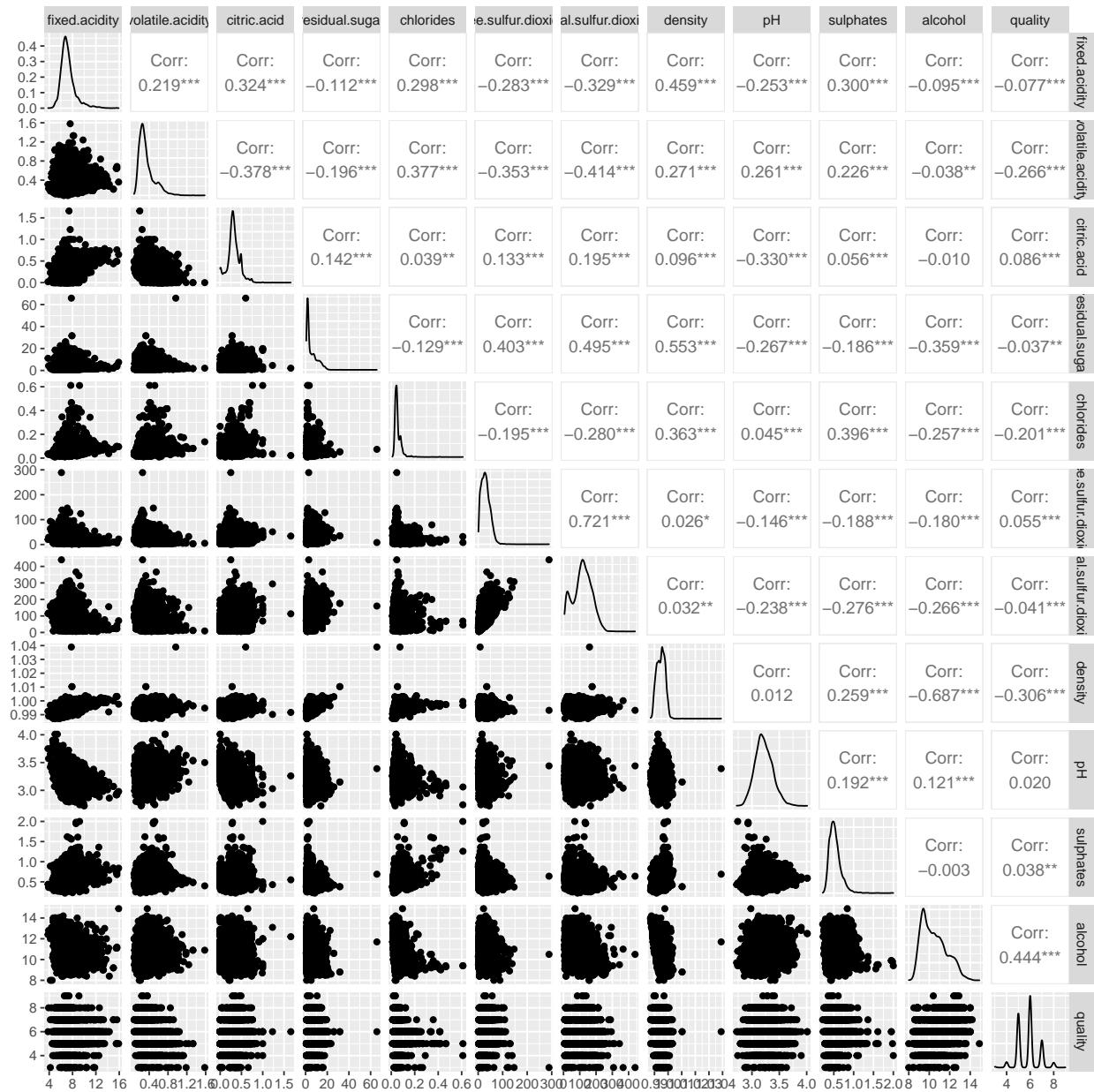
Para ver estas relaciones se muestra otra gráfica en la que podemos observar visualmente los datos 2 a 2

```

if (!require('GGally')) install.packages('GGally'); library('GGally')

## Loading required package: GGally
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
ggpairs(wine_n)

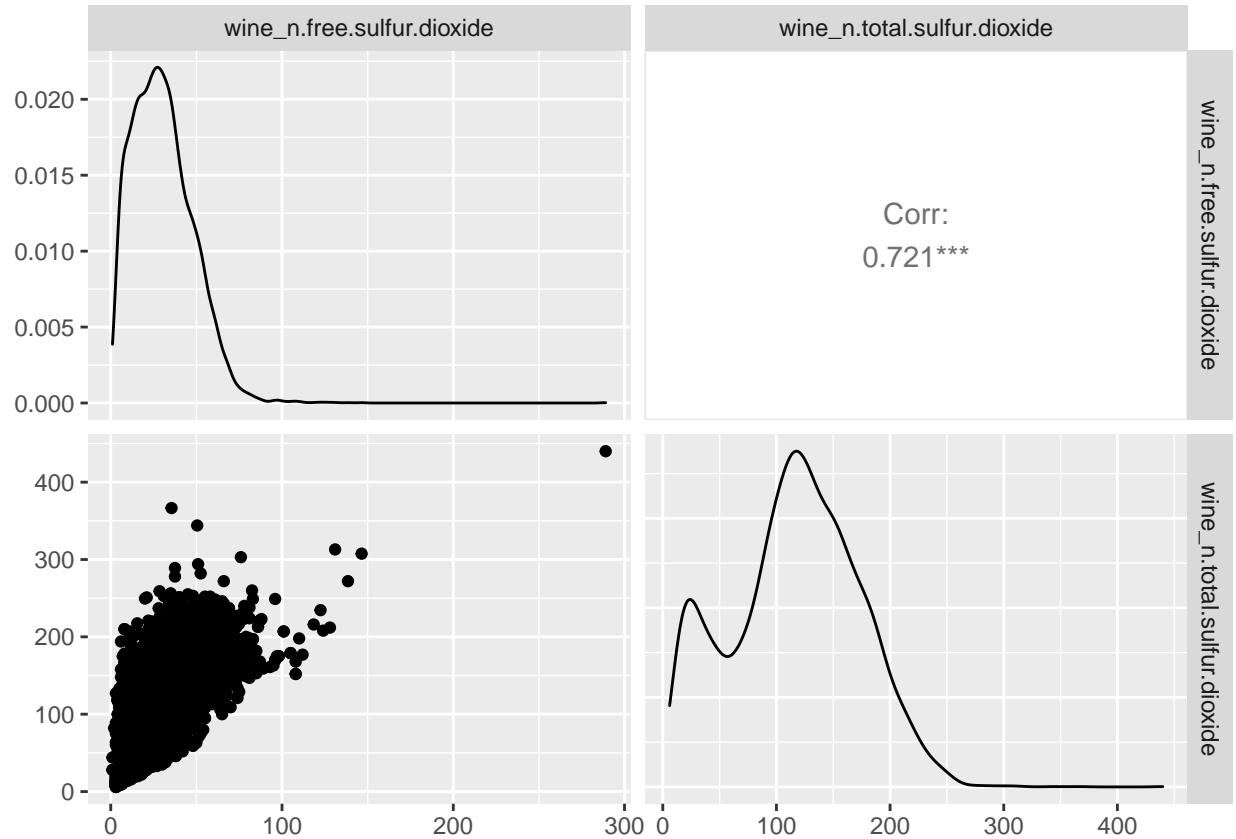
```



Como ya hemos podido observar que variables están relacionadas vamos a crear el mismo plot pero solo para ellas, y poder analizarlas en más detalle,

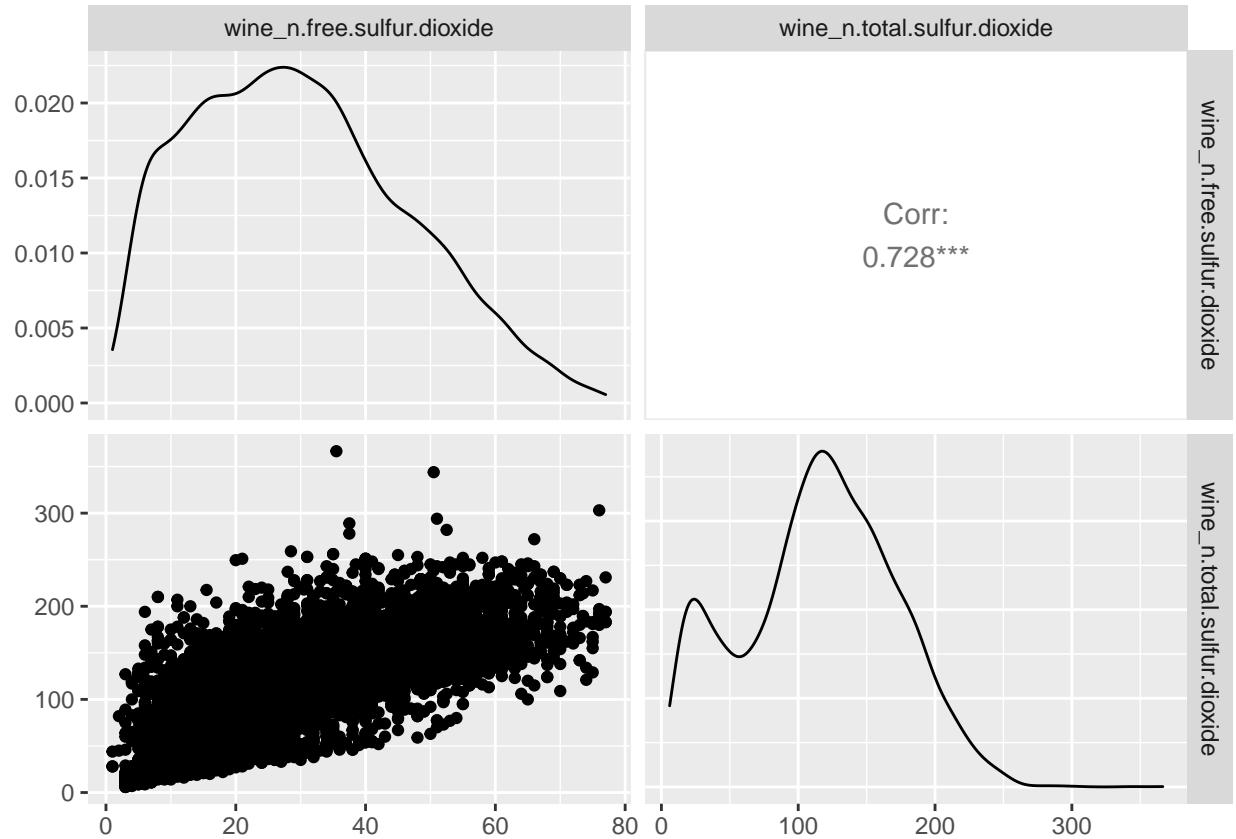
### 1. free y total sulfur dioxide:

```
free_total_dioxide <- data.frame(wine_n$free.sulfur.dioxide, wine_n$total.sulfur.dioxide)
ggpairs(free_total_dioxide)
```



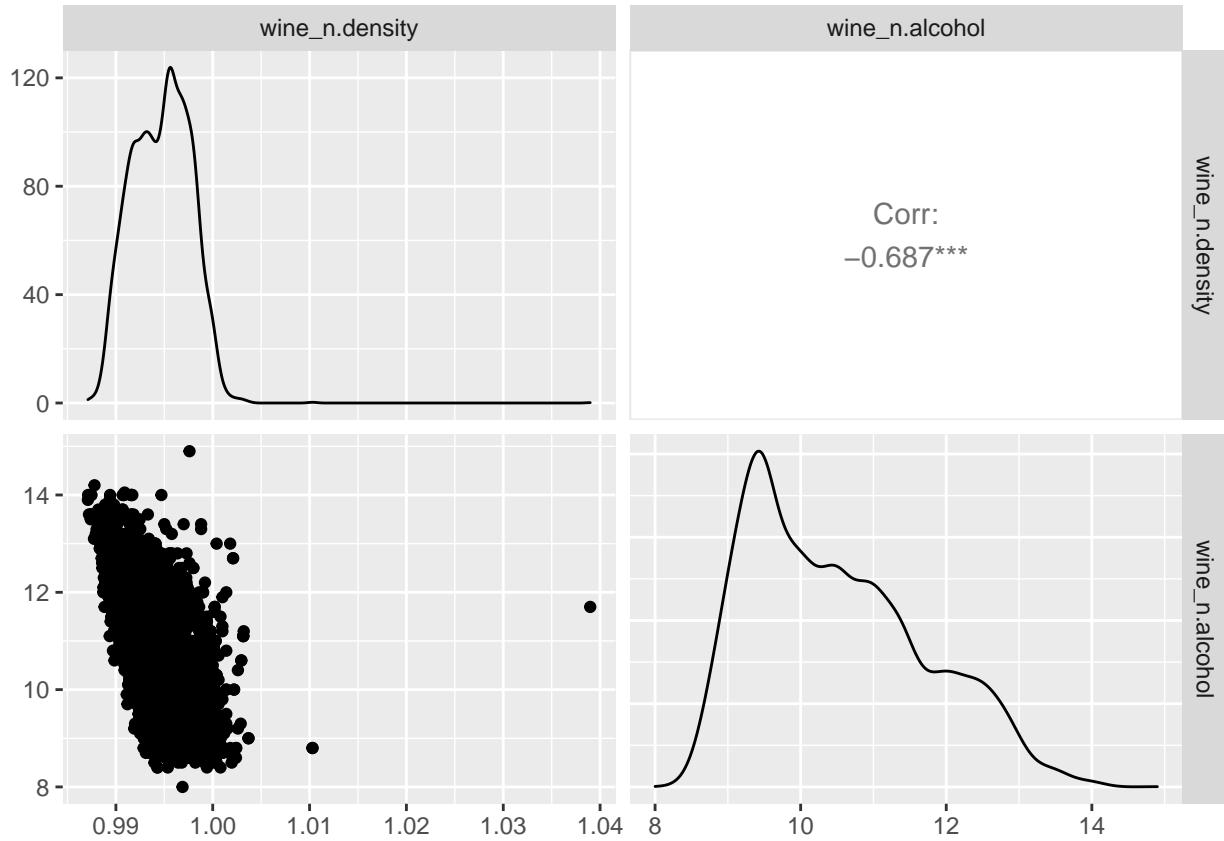
Vemos como hay un outlier que distorsiona la gráfica por lo que lo vamos a eliminar

```
wine_n <- wine_n[!(wine_n$free.sulfur.dioxide %in% boxplot.stats(wine_n$free.sulfur.dioxide)$out),]
#reprint the value
free_total_dioxide <- data.frame(wine_n$free.sulfur.dioxide, wine_n$total.sulfur.dioxide)
ggpairs(free_total_dioxide)
```



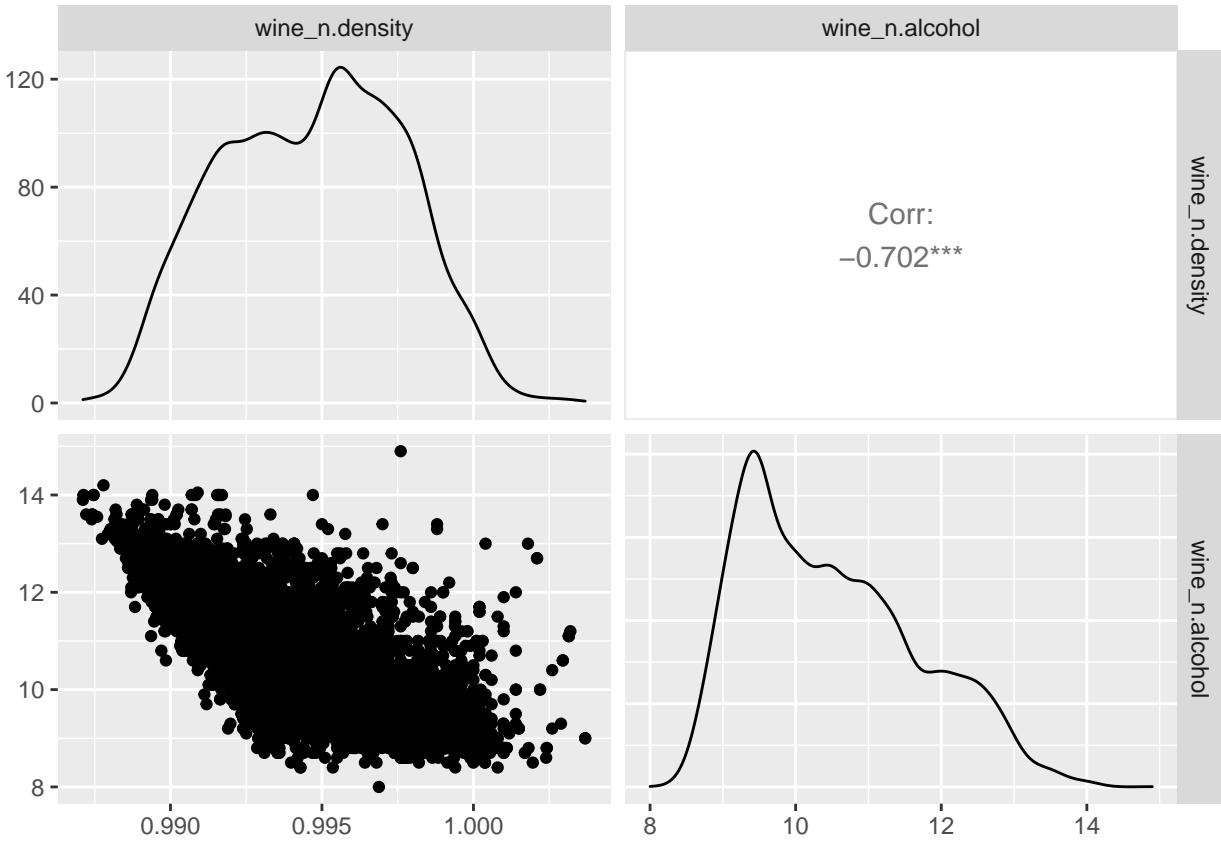
2. density y alcohol:

```
density_alcohol <- data.frame(wine_n$density, wine_n$alcohol)
ggpairs(density_alcohol)
```



Igual que sucedía antes tenemos algún valor que distorsiona la gráfica por lo que vamos a eliminarlo

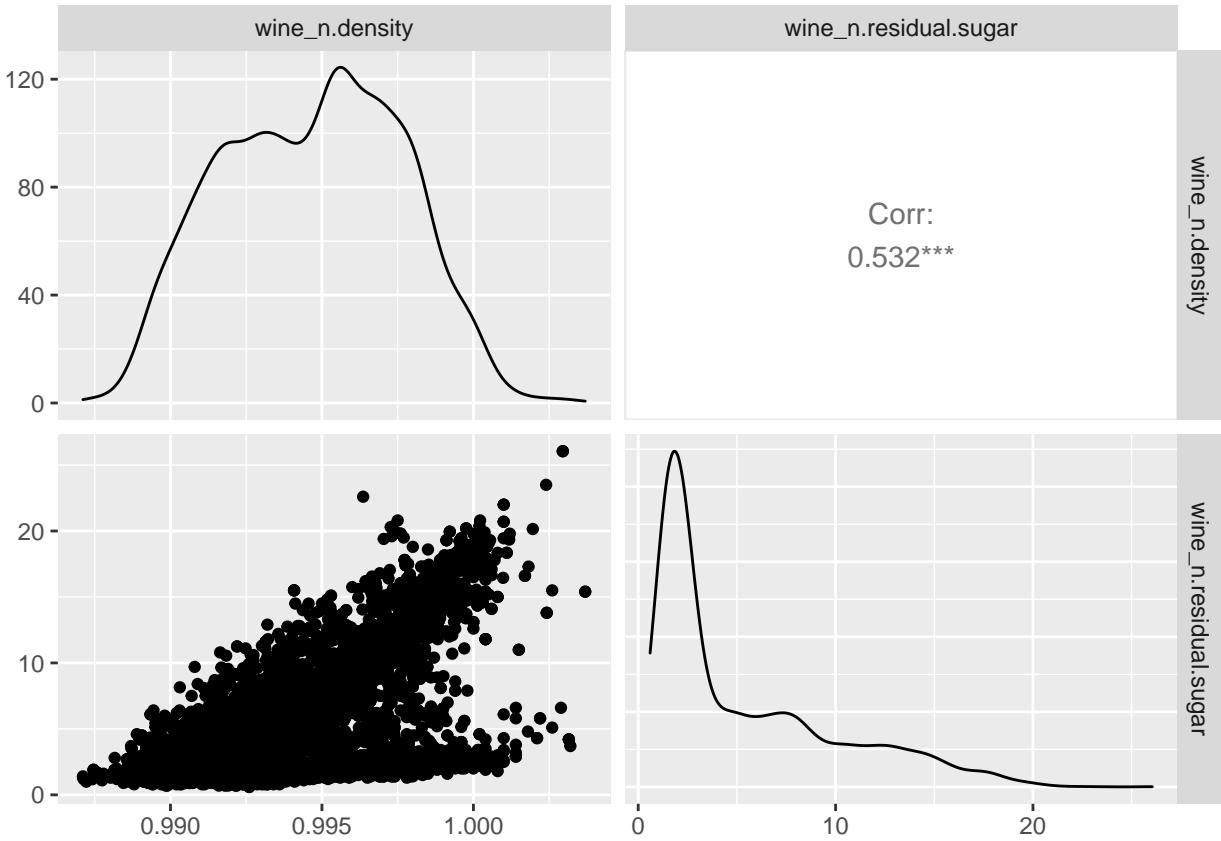
```
wine_n <- wine_n[!(wine_n$density %in% boxplot.stats(wine_n$density)$out),]
#reprint the value
density_alcohol <- data.frame(wine_n$density, wine_n$alcohol)
ggpairs(density_alcohol)
```



Vemos como se puede ver la correlación y en este caso además la pendiente de la recta de regresión es negativa ya que al aumentar uno de los valores disminuye el otro y viceversa.

3. Density y residual sugar:

```
density_sugar <- data.frame(wine_n$density, wine_n$residual.sugar)
ggpairs(density_sugar)
```



En este caso como ya eliminamos los valores outliers de la variable density nos sale directamente sin tener que limpiar por lo que podemos ver la correlación positiva que existe. Podríamos continuar realizando las gráficas 2 a dos entre los distintos atributos del dataset e iríamos observando como las correlaciones son cada vez más pequeñas y el gráfico comenzará a dejar de formar una línea a lo largo de la regresión.

### Análisis de componentes principales

Vamos a realizar el PCA que es un método de reducción de la dimensionalidad que transforma conjuntos de datos con numerosas variables en conjuntos más pequeños con atributos que contienen la mayoría de la información relevante. Esto es muy útil para datasets con un gran número de atributos, en nuestro caso que solo tenemos 11 no es algo muy necesario pero lo vamos a realizar a modo de demostración.

La idea subyacente es que vamos a comprobar como los distintos atributos afectan a la variabilidad del dataset y como las dimensiones son en realidad variables formadas por combinaciones de los atributos iniciales.

Vamos a utilizar prcomp para el análisis como lo que pretendemos encontrar es la calidad del vino lo vamos a quitar del dataset y es importante que le pasemos el flag de scale TRUE ya que sin este tendríamos unos atributos descompensados y el PC1 tendría un peso en la varianza muy alto

```
wine_pca_init <- wine_n
wine_pca_init$quality <- NULL
wine_pca <- prcomp(x=wine_pca_init, scale= TRUE)
summary(wine_pca)
```

```
## Importance of components:
##                               PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation    1.755  1.5832  1.2475  0.97637  0.83922  0.77493  0.72277
## Proportion of Variance 0.280  0.2279  0.1415  0.08666  0.06403  0.05459  0.04749
```

```

## Cumulative Proportion  0.280 0.5079 0.6494 0.73603 0.80006 0.85465 0.90214
##                               PC8      PC9     PC10    PC11
## Standard deviation     0.69682 0.57680 0.47752 0.17370
## Proportion of Variance 0.04414 0.03025 0.02073 0.00274
## Cumulative Proportion  0.94628 0.97653 0.99726 1.00000

```

Podemos ver un resumen de como afecta cada componente a la variabilidad de los datos y la proporción de cada atributo a la varianza de los datos y como estos están ordenados de mayor a menor. Vamos a ver gráficamente como afecta cada componente

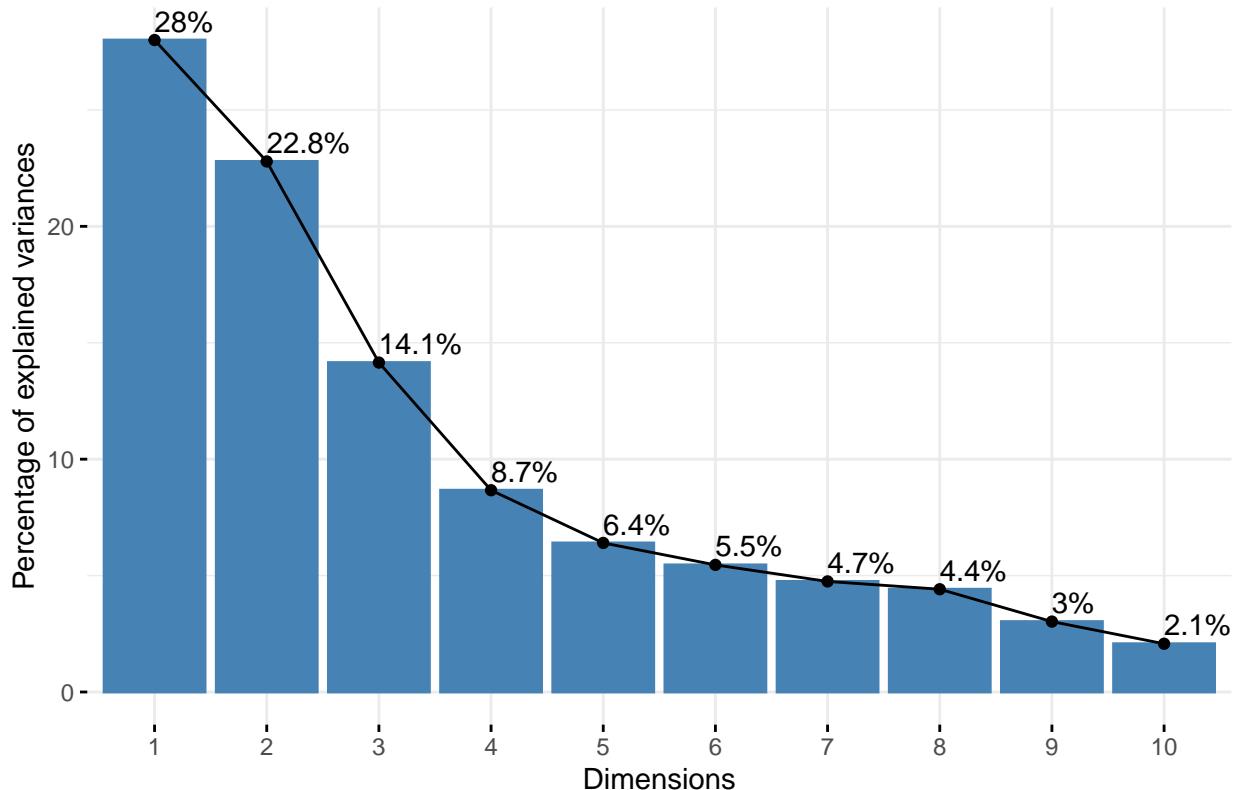
```

if (!require('factoextra')) install.packages('factoextra'); library('factoextra')

## Loading required package: factoextra
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
fviz_eig(wine_pca, addlabels= TRUE, center= TRUE)

```

Scree plot



Para conseguir identificar cuento influye cada variable y cada componente en las distintas dimensiones tenemos que ver el contenido de rotation:

```
wine_pca$rotation
```

```

##                                PC1        PC2        PC3        PC4        PC5
## fixed.acidity   0.23907329 -0.33531154  0.43479539 -0.17943698  0.1917455
## volatile.acidity 0.37860824 -0.11125845 -0.30675377 -0.22392417 -0.1504352
## citric.acid    -0.15288560 -0.18356929  0.59110059  0.27486381  0.1312269
## residual.sugar -0.34836593 -0.32467868 -0.16702591 -0.15106062  0.3227102
## chlorides       0.29009088 -0.31453394 -0.01466735  0.23454642 -0.6469011

```

```

## free.sulfur.dioxide -0.43625253 -0.07950033 -0.12938505 0.33177112 -0.2069470
## total.sulfur.dioxide -0.48245737 -0.09017793 -0.10561485 0.20200640 -0.1620839
## density 0.05204371 -0.58322515 -0.17563532 -0.06349442 0.3073175
## pH 0.21710964 0.15797649 -0.45228462 0.43751714 0.4471505
## sulphates 0.29574948 -0.19344117 0.07335881 0.63700436 0.1150731
## alcohol 0.10109368 0.46921445 0.26581338 0.11033077 0.1550218
## PC6 PC7 PC8 PC9
## fixed.acidity -0.1679632 0.24227350 -0.404108729 0.36492796
## volatile.acidity -0.4775374 0.43150087 0.002840176 -0.48267870
## citric.acid 0.2270146 0.40056917 0.255699689 -0.41633364
## residual.sugar -0.2706747 -0.15334859 0.572920602 0.07893007
## chlorides 0.1447857 0.09291241 0.433520718 0.28816538
## free.sulfur.dioxide -0.3080561 0.29235780 -0.248945254 0.38901986
## total.sulfur.dioxide -0.1466554 0.13576943 -0.199694304 -0.30403700
## density 0.0272517 0.04167649 0.009949258 0.10613871
## pH 0.3037002 0.40010235 0.025875139 0.12622357
## sulphates -0.3389422 -0.51844424 -0.134239899 -0.19952059
## alcohol -0.5231810 0.16732221 0.366724819 0.24589565
## PC10 PC11
## fixed.acidity -0.269847903 -0.3408766427
## volatile.acidity 0.153014737 -0.0822885418
## citric.acid 0.223694426 0.0033523820
## residual.sugar -0.019093147 -0.4321492279
## chlorides -0.201666147 -0.0443071998
## free.sulfur.dioxide 0.489940485 -0.0005272779
## total.sulfur.dioxide -0.710588267 0.0606416241
## density -0.008057254 0.7168734766
## pH -0.142979270 -0.2064459849
## sulphates 0.054954234 -0.0787392721
## alcohol -0.210214883 0.3491408096

```

Por ejemplo podemos ver como las variables que más afectan a la dimensión 1 son total sulfur dioxide de forma negativa con -0.4824 y free sulfur dioxide también de forma negativa seguida de volatile acidity. Podríamos ir analizando el comportamiento de cada atributo para las distintas dimensiones.

A continuación vamos a obtener los valore propios o eigenvalues

```
get_eigenvalue(wine_pca)
```

```

## eigenvalue variance.percent cumulative.variance.percent
## Dim.1 3.08010870 28.0009882 28.00099
## Dim.2 2.50658648 22.7871499 50.78814
## Dim.3 1.55637077 14.1488252 64.93696
## Dim.4 0.95329631 8.6663301 73.60329
## Dim.5 0.70428345 6.4025769 80.00587
## Dim.6 0.60051338 5.4592126 85.46508
## Dim.7 0.52239605 4.7490550 90.21414
## Dim.8 0.48555437 4.4141307 94.62827
## Dim.9 0.33269547 3.0245043 97.65277
## Dim.10 0.22802484 2.0729531 99.72573
## Dim.11 0.03017016 0.2742742 100.00000

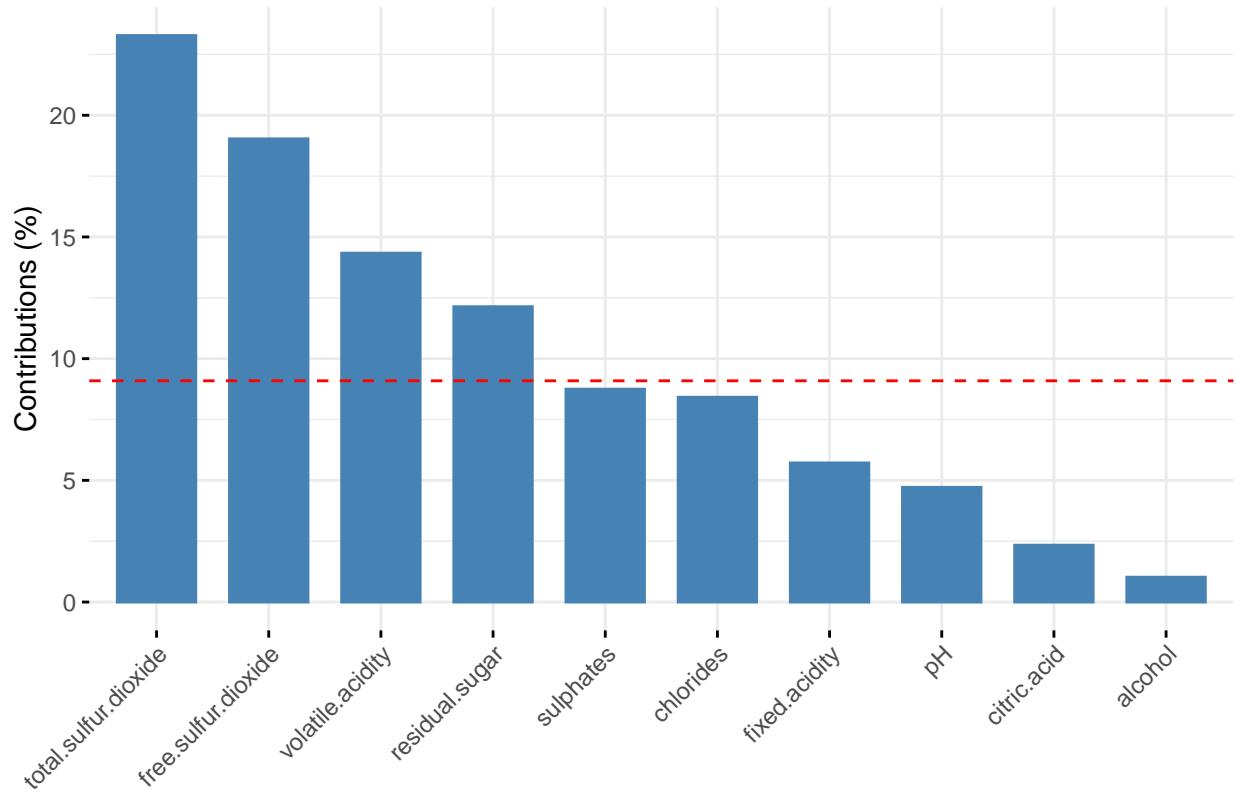
```

Los valores propios indican las varianzas de los componentes principales. Para seleccionar los componentes podemos usar la regla de Kaiser-Futman en la que se seleccionan los valores por encima de 1 que en este caso sería los 3 primeros.

También podemos mostrar gráficamente como afectan o contribuyen las distintas variables a las dimensiones:

```
#Variable contribution to dimension 1  
fviz_contrib(wine_pca, choice = "var", axes = 1, top = 10)
```

Contribution of variables to Dim-1

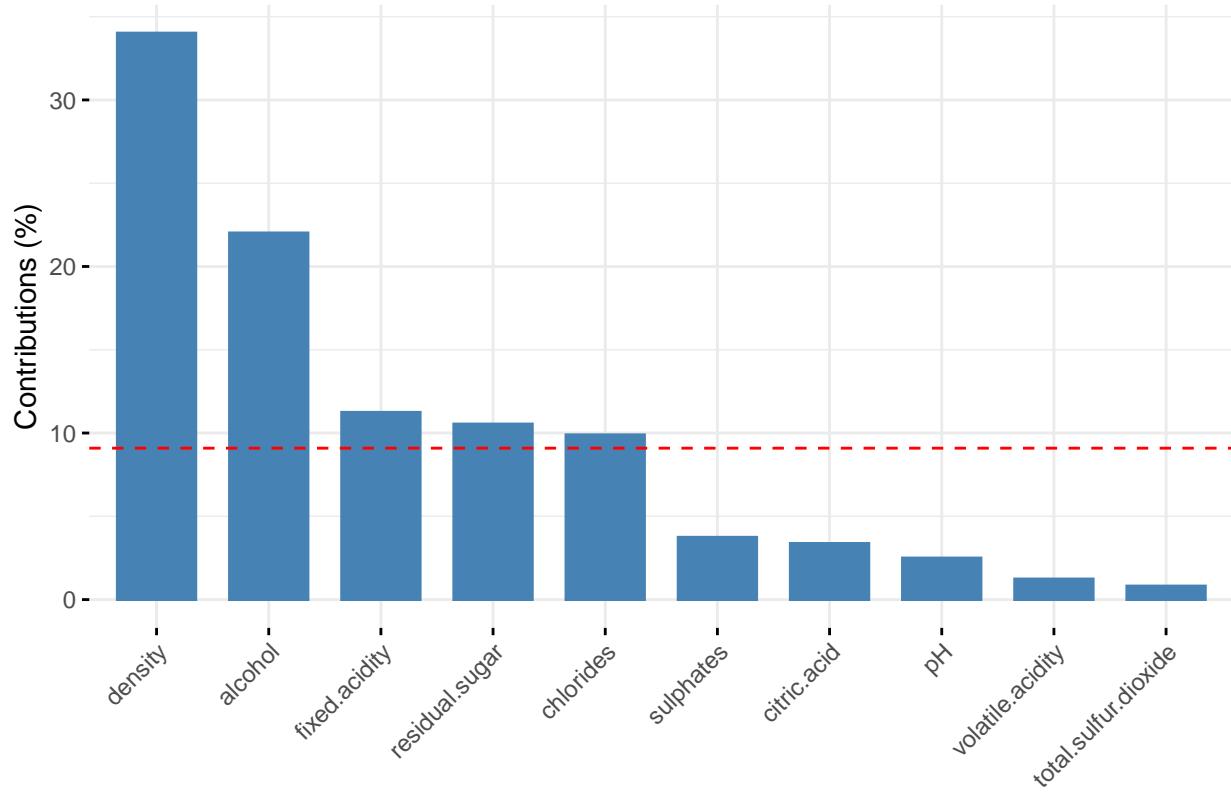


Como ya vimos en rotation simplemente aquí lo vemos visualmente que resulta más sencillo.

Dimensión 2:

```
#Variable contribution to dimension 2  
fviz_contrib(wine_pca, choice = "var", axes = 2, top = 10)
```

## Contribution of variables to Dim–2

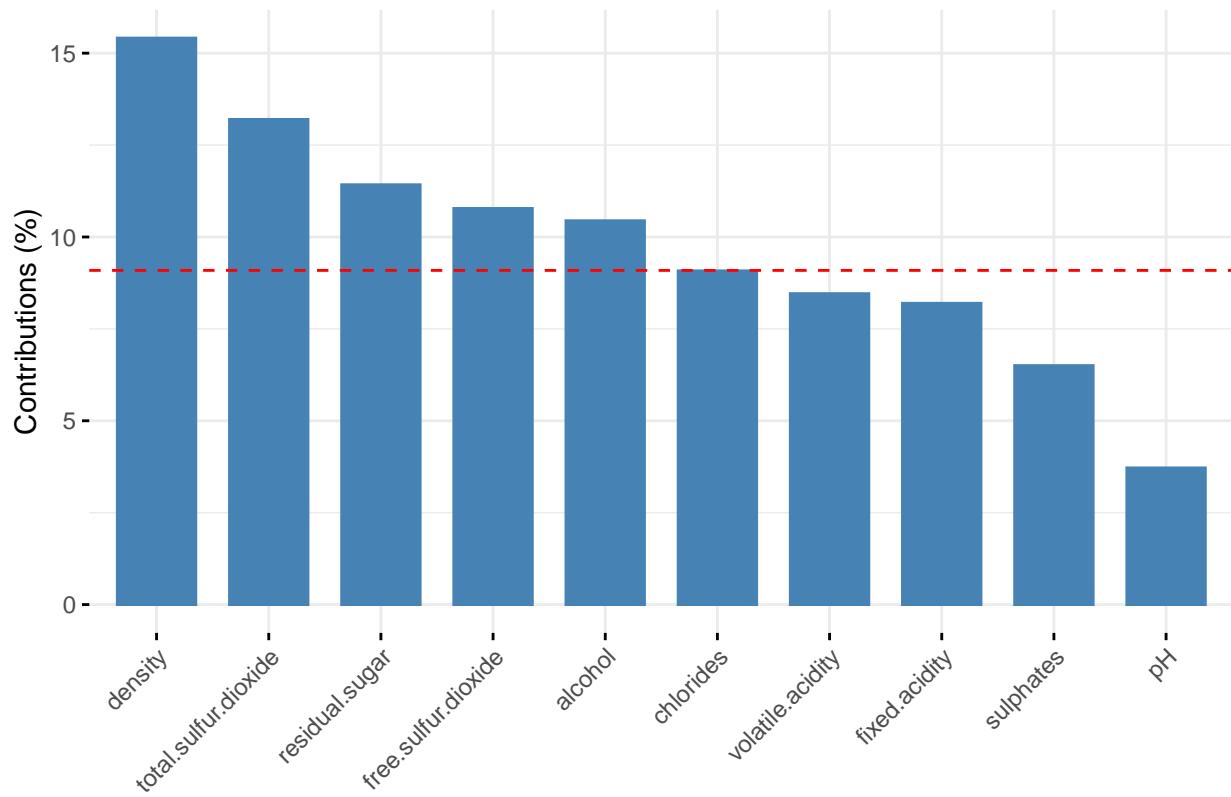


En este caso vemos como a la dimensión dos las variables que más afectan son density y alcohol.

Como las dimensiones 1 y 2 son las que más peso tienen sobre el dataset vamos a ver como contribuyen las variables a la mezcla de las 2

```
#Variable contribution to dimension 1  
fviz_contrib(wine_pca, choice = "var", axes = 1:2, top = 10)
```

## Contribution of variables to Dim-1–2



Aquí vemos como la que más influye es la densidad y como le siguen el total de dioxido de sulfato y el azúcar residual.

### Conclusiones del análisis de componentes principales

Se ha observado como las 7 primeras dimensiones contribuyen al 90% de la variabilidad del dataset siendo los componentes 1 y 2 muy importantes ya que entre los dos suman un 50% y a su vez hemos visto como los atributos que más afectan a las dos primeras componentes son la densidad el dioxido de sulfato total y el azúcar residual.

### Comprobación de la normalidad

Se va a proceder a comprobar la normalidad de las distintas variables que corresponden a las propiedades físicas del vino mediante la prueba de normalidad de Anderson-Darling. Para ver si corresponde o no a una distribución normal se compara con un p-valor de 0.05 si es superior consideraremos que la variable sigue una distribución normal.

```
if(!require(nortest)){
  install.packages('nortest', repos='http://cran.us.r-project.org')
  library(nortest)
}

## Loading required package: nortest
alpha = 0.05
columns = colnames(wine_n)
for (i in 1:ncol(wine_n)) {
  if (is.integer(wine_n[,i]) | is.numeric(wine_n[,i])) {
```

```

p_val = ad.test(wine_n[,i])$p.value
if (p_val > alpha) {
  cat("Sigue una distribución normal -> ")
  cat(columns[i])
  cat("\n")
}
else {
  cat("No sigue una distribución normal -> ")
  cat(columns[i])
  cat("\n")
}
}

## No sigue una distribución normal -> fixed.acidity
## No sigue una distribución normal -> volatile.acidity
## No sigue una distribución normal -> citric.acid
## No sigue una distribución normal -> residual.sugar
## No sigue una distribución normal -> chlorides
## No sigue una distribución normal -> free.sulfur.dioxide
## No sigue una distribución normal -> total.sulfur.dioxide
## No sigue una distribución normal -> density
## No sigue una distribución normal -> pH
## No sigue una distribución normal -> sulphates
## No sigue una distribución normal -> alcohol
## No sigue una distribución normal -> quality

```

## Modelo para predecir la calidad del vino

Para realizar el modelo tenemos que preparar en primer lugar el dataset para ello vamos a aleatorizar el orden y separarlo en una proporción de 70-30 para tener tanto datos para entrenar el modelo como para comprobar la calidad del mismo.

Primero aleatorizamos la muestra fijando un seed

```

set.seed(123)
wine_random_q <- wine[sample(nrow(wine)),]
#Quitamos la columna color ya que aquí no nos hace falta
wine_random_q$color <-NULL

wine_train <- sample_frac(wine_random_q, .7)
wine_test <- setdiff(wine_random_q, wine_train)

```

Creamos el modelo de regresión:

```

lm_model <- lm(quality ~ fixed.acidity + volatile.acidity + citric.acid + chlorides + free.sulfur.dioxide + total.sulfur.dioxide + density + pH + sulphates + alcohol, data = wine_train)

## Call:
## lm(formula = quality ~ fixed.acidity + volatile.acidity + citric.acid +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol, data = wine_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -3.5075  -0.8765  -0.0714   0.8765   3.5075 
## 
```

```

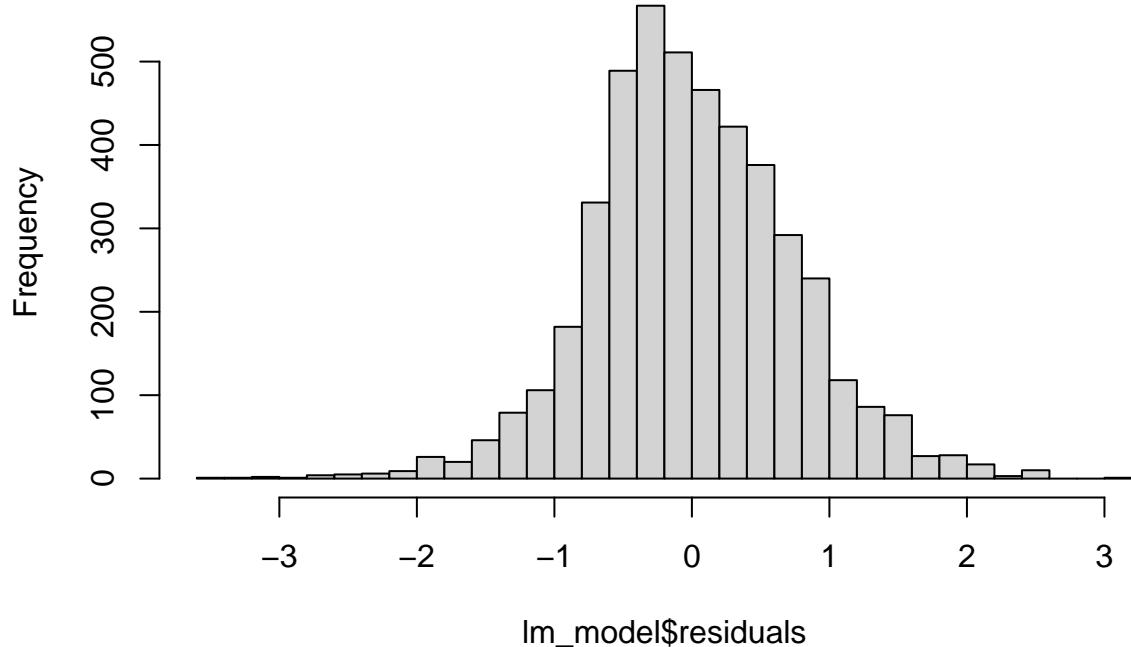
## -3.5959 -0.4652 -0.0381  0.4660  3.1234
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -3.132e+01  6.412e+00 -4.885 1.07e-06 ***
## fixed.acidity          -1.431e-02  1.287e-02 -1.112 0.266189
## volatile.acidity       -1.513e+00  8.848e-02 -17.095 < 2e-16 ***
## citric.acid           -1.130e-01  9.607e-02 -1.177 0.239350
## chlorides              -1.488e+00  4.179e-01 -3.560 0.000374 ***
## free.sulfur.dioxide    8.138e-03  9.120e-04  8.923 < 2e-16 ***
## total.sulfur.dioxide -2.119e-03  3.202e-04 -6.619 4.03e-11 ***
## density                3.403e+01  6.463e+00  5.265 1.46e-07 ***
## pH                     1.746e-03  8.209e-02  0.021 0.983029
## sulphates              5.930e-01  8.479e-02  6.994 3.06e-12 ***
## alcohol                3.527e-01  1.450e-02 24.328 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7351 on 4537 degrees of freedom
## Multiple R-squared:  0.2867, Adjusted R-squared:  0.2851
## F-statistic: 182.3 on 10 and 4537 DF,  p-value: < 2.2e-16

```

Podemos comprobar la normalidad del modelo con el test the shapiro

```
hist(lm_model$residuals, breaks = 30)
```

**Histogram of lm\_model\$residuals**



```
shapiro_test <- shapiro.test(lm_model$residuals)
shapiro_test
```

```
##
##  Shapiro-Wilk normality test
##
## data: lm_model$residuals
## W = 0.99195, p-value = 2.223e-15
```

El test de Shapiro nos indica que la distribución no es normal ya que el p-value es menor a 0.05

Vamos a comprobar como funciona el modelo

```
results = round(predict(lm_model, newdata = wine_test))
```

Podemos utilizar la matriz de confusión para ver que tal ha quedado la clasificación:

```
if(!require(caret)){
  install.packages('caret', repos='http://cran.us.r-project.org')
  library(caret)
}

## Loading required package: caret
## Loading required package: lattice
confusionMatrix(factor(results,levels=3:8), factor(wine_test$quality,levels=3:8))
```

```
## Confusion Matrix and Statistics

##          Reference
## Prediction 3 4 5 6 7 8
##           3 0 0 0 0 0 0
##           4 0 1 0 0 0 0
##           5 4 32 225 109 6 1
##           6 5 24 236 453 148 29
##           7 1 0 4 35 62 12
##           8 0 0 0 0 0 0

## Overall Statistics
##
##          Accuracy : 0.5342
## 95% CI : (0.5076, 0.5608)
## No Information Rate : 0.4304
## P-Value [Acc > NIR] : 5.15e-15
##
##          Kappa : 0.2467
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##          Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity      0.00000 0.017544 0.4839 0.7588 0.28704 0.00000
## Specificity      1.00000 1.000000 0.8351 0.4405 0.95559 1.00000
## Pos Pred Value    NaN 1.000000 0.5968 0.5061 0.54386     NaN
## Neg Pred Value 0.99279 0.959596 0.7624 0.7073 0.87903 0.96972
```

```

## Prevalence          0.00721 0.041096  0.3353  0.4304  0.15573  0.03028
## Detection Rate    0.00000 0.000721  0.1622  0.3266  0.04470  0.00000
## Detection Prevalence 0.00000 0.000721  0.2718  0.6453  0.08219  0.00000
## Balanced Accuracy  0.50000 0.508772  0.6595  0.5997  0.62132  0.50000

```

Si observamos aunque la predicción exacta es bastante mala con tan solo un 54.67% de acierto si que podemos ver en la matriz de confusión como la mayoría se encuentran cercanos a la diagonal por lo que en la mayor parte de los casos hay un error de  $\pm 1$  en la calidad esto es principalmente porque la calidad se expresaba en enteros y esto hace que igual sea más interesante crear un clasificador y pasar los valores numéricos a bueno malo o normal.

```

if(!require(lares)){
  install.packages('laras', repos='http://cran.us.r-project.org')
  library(lares)
}

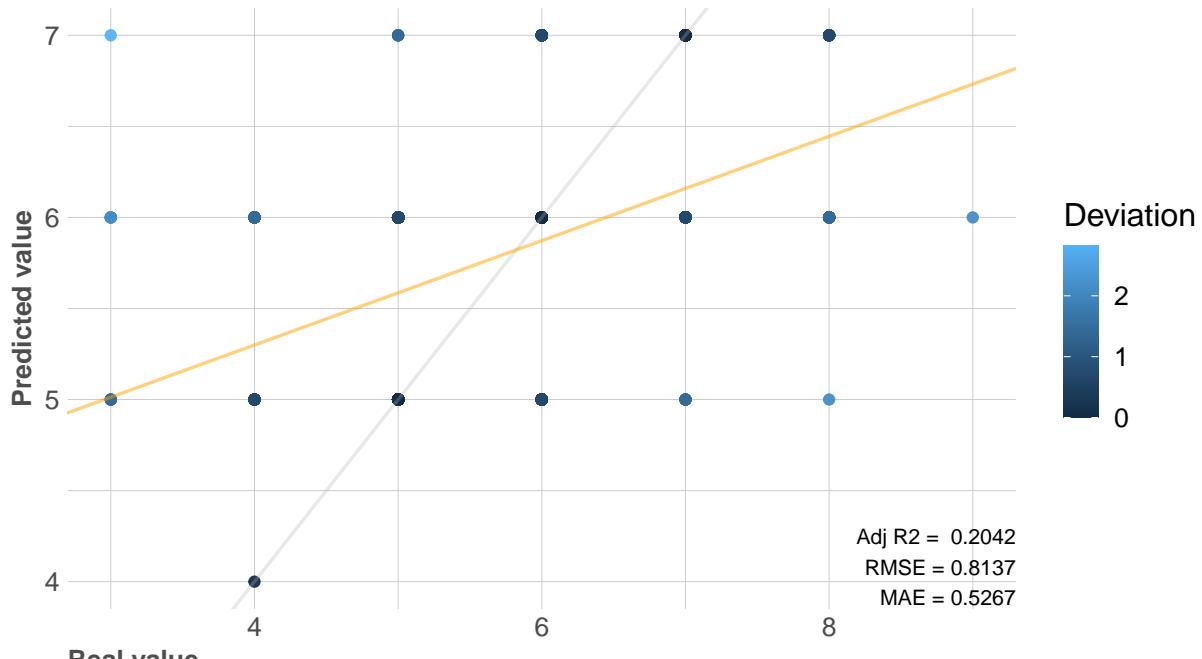
## Loading required package: lares
laras::mplot_lineal(tag = wine_test$quality,
                     score = results,
                     subtitle = "Wine quality regression model",
                     model_name = "wine_quality_1")

## Warning in .font_global(font, quiet = FALSE): Font 'Arial Narrow' is not
## installed, has other name, or can't be found

```

## Regression Model Results

*Wine quality regression model*



Voy a probar con un modelo clasificador para ver si mejora ya que al haber solo 4 valores discretos en la calidad puede ser que sea más efectivo con la clasificación que la regresión.

```

set.seed(123)
wine_random_qc <- wine[sample(nrow(wine)),]
#Quitamos la columna color ya que no nos hace falta
wine_random_qc$color <- NULL
#separamos las y que será el color del resto del dataset x
y <- wine_random_qc[,12]
X <- wine_random_qc[,-12]

# Separamos 1/3
split_prop <- 3
index = sample(1:nrow(wine), size=floor(((split_prop-1)/split_prop)*nrow(wine)))
wine_trainX <- X[index,]
wine_testX <- X[-index,]
wine_trainy <- y[index]
wine_testy <- y[-index]

```

Creamos el modelo:

```

wine_trainy = as.factor(wine_trainy)
model_q <- C50::C5.0(wine_trainX, wine_trainy, rules=TRUE)
#summary(model_q)

```

Observamos que a simple vista mejora el modelo de regresión al reducir los errores al 21% vamos a comprobar como predice

```

predicted_model_q <- predict(model_q, wine_testX, type="class")
print(sprintf("El modelo clasifica la calidad del vino con una precisión del: %.4f %%", 100*sum(predicted_
## [1] "El modelo clasifica la calidad del vino con una precisión del: 56.0480 %"

```

Vemos que sigue siendo batante malo y apenas hemos mejorado y solo predice la calidad en un 56% de los casos.

## Modelo para predecir el color del vino

Como al comenzar unimos los dataset de vino blanco y vino tinto vamos a utilizar un modelo de árbol de decisión para ver si es posible predecir el color de vino por sus propiedades. Para ello vamos a coger el dataset y aleatorizarlo y eliminar la columna quality:

```

set.seed(123)
wine_random_c <- wine[sample(nrow(wine)),]
#Quitamos la columna quality ya que no nos hace falta
wine_random_c$quality <- NULL
#separamos las y que será el color del resto del dataset x
y <- wine_random_c[,12]
X <- wine_random_c[,-12]

# Separamos 1/3
split_prop <- 3
index = sample(1:nrow(wine), size=floor(((split_prop-1)/split_prop)*nrow(wine)))
wine_trainX <- X[index,]
wine_testX <- X[-index,]
wine_trainy <- y[index]
wine_testy <- y[-index]

```

Creamos el modelo:

```
wine_trainy = as.factor(wine_trainy)
model <- C50::C5.0(wine_trainX, wine_trainy, rules=TRUE)
#summary(model)
```

El modelo parece bastante bueno ya que solo hay 23 errores de 4331 lo que supone un 0.5% de error. Vamos a ver como se comporta prediciendo el conjunto de test:

```
predicted_model <- predict( model, wine_testX, type="class" )
print(sprintf("El modelo clasifica los vinos con una precisión del: %.4f %%", 100*sum(predicted_model ==
```

```
## [1] "El modelo clasifica los vinos con una precisión del: 98.8920 %"
```

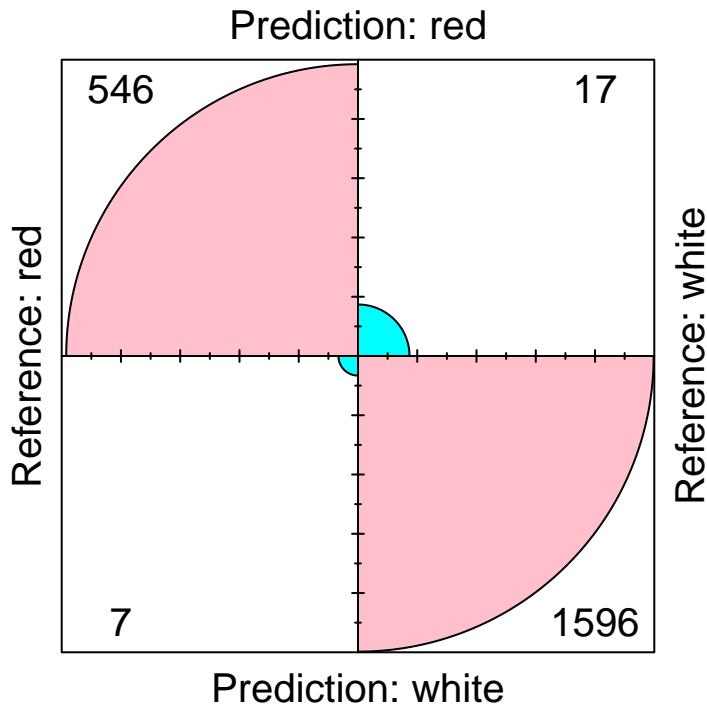
Vemos que nuestro clasificador es muy bueno y que por lo tanto clasifica de manera correcta el vino en blanco o tinto en función de sus cualidades con una precisión del 98.89%

Vamos a comprobar la matriz de confusión en este caso:

```
reference <- as.factor(wine_testy)
cm <- confusionMatrix(data=predicted_model, reference = reference, positive="white")
cm
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction red white
##       red     546    17
##       white     7 1596
##
##                  Accuracy : 0.9889
##                          95% CI : (0.9836, 0.9929)
##      No Information Rate : 0.7447
##      P-Value [Acc > NIR] : < 2e-16
##
##                  Kappa : 0.971
##
##      Mcnemar's Test P-Value : 0.06619
##
##                  Sensitivity : 0.9895
##                  Specificity : 0.9873
##      Pos Pred Value : 0.9956
##      Neg Pred Value : 0.9698
##                  Prevalence : 0.7447
##      Detection Rate : 0.7368
##      Detection Prevalence : 0.7401
##      Balanced Accuracy : 0.9884
##
##      'Positive' Class : white
##
fourfoldplot(cm$table, color = c("cyan", "pink"),
             conf.level = 0, margin = 1, main = "Confusion Matrix Color Wine")
```

## Confusion Matrix Color Wine



Vemos que solo tenemos mal clasificados 7 blancos y 17 rojos por lo que el clasificador es bastante bueno.

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

Tras realizar el análisis de los datos hemos creado dos modelos uno de regresión para intentar predecir la calidad del vino en función de sus atributos y en este caso el resultado no ha sido muy bueno tanto si utilizamos un modelo de regresión como si usamos un árbol para clasificar. Esto es porque la calidad de los vinos tiene un rango muy pequeño y no se ha podido predecir la calidad de forma satisfactoria. En el caso del modelo para predecir el tipo de vino (blanco, tinto) si que ha sido bastante satisfactorio y podemos predecir el tipo de vino en función de sus atributos con más de un 98% de precisión por lo que a la vista de los resultados obtenidos si que podemos predecir el tipo de vino por sus atributos pero no su calidad.

### Repositorio y video

El link del repositorio es el siguiente <https://github.com/Ambrotd/tcvd-wine/>

El video ha sido subido a Google Drive <https://drive.google.com/file/d/1D0zbKzF9seoLecW2LUNvfjD8tLyLgsc/view?usp=sharing>