Pràctica 2

Oriol Medina Marcos, Guiu Riera Riera

20/12/2021

```
# Llibreries a utilitzar
library(ggplot2)
library(dplyr)
library(gridExtra)
library(grid)
library(car)
library(corrplot)
library(Hmisc)
library(caTools)
library(kableExtra)
library(pROC)
```

1. Descripció del dataset. Perquè és important i quina pregunta/problema pretén respondre?

El dataset triat és el proporcionat en aquesta pràctica sota el nom de Red Wine Quality (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009). Aquest dataset conté una sèrie de paràmetres mesurats en diferents tipus de vi i una qualificació final d'aquest en funció de les notes obtingudes en tots els paràmetres.

El dataset es compon 1599 registres en els que s'informa per a cadascun d'aquest els següents paràmetres o columnes:

- fixed acidity
- volatile acidity
- citric acid
- residual sugar
- chlorides
- free sulfur dioxide
- total sulfur dioxide
- density
- pH
- sulphates
- alcohol
- quality (puntuació entre 0 i 10)

Podriem dir que aquest dataset preten estudiar les tendències dels diferents tipus de vins estudiats com podria ser la graduació alcohòlica o trobar un tipus de vi en funció d'uns valors en els paràmetres mesurats.

2. Integració i selecció de les dades d'interès a analitzar.

Per començar amb la integració de les dades, llegirem el fitxer *winequality-red.csv* en el que hi trobem totes les dades. Per fer-ho utilitzem la instrucció *read.csv()* passant com a paràmetres la ruta del fitxer i el separador de columnes que en aquest cas és una coma (,).

```
df<-read.csv("winequality-red.csv", sep=",")
```

Ara tenim en la variable *data* tota la informació carregada i comprovem que cada variable és del tipus que s'espera:

```
sapply(df, function(x) { class(x) })
```

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

Per tal de fer-nos una idea de la informació que contenen les variables utilitzem la funció *summary()* que ens permet veure com estant distribuits cada un dels seus valors

```
summary(df)
```

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

3. Neteja de les dades.

3.1. Les dades contenen zeros o elements buits? Com gestionaries aquests casos?

Comprovem si el joc de dades conté algun registre amb valors buits o nuls

```
# Comprovem si hi ha valors NA o buits
"NA:"
```

```
## [1] "NA:"
```

```
colSums(is.na(df))
```

```
##        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
```

```
"Buits:"
```

```
## [1] "Buits:"
```

```
colSums(df=="")
```

```
##        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
```

```
# Com que amb altres datasets que hem treballat hi havia valors amb "?".
# Comprovem si també n'hi ha
"?:"
```

```
## [1] "?:"
```

```
colSums(df==" ?")
```

```
##        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
```

```r
# Comprovem si hi ha registres amb 0s
"0's:"
```

```
## [1] "0's:"
```

```r
colSums(df==0)
```

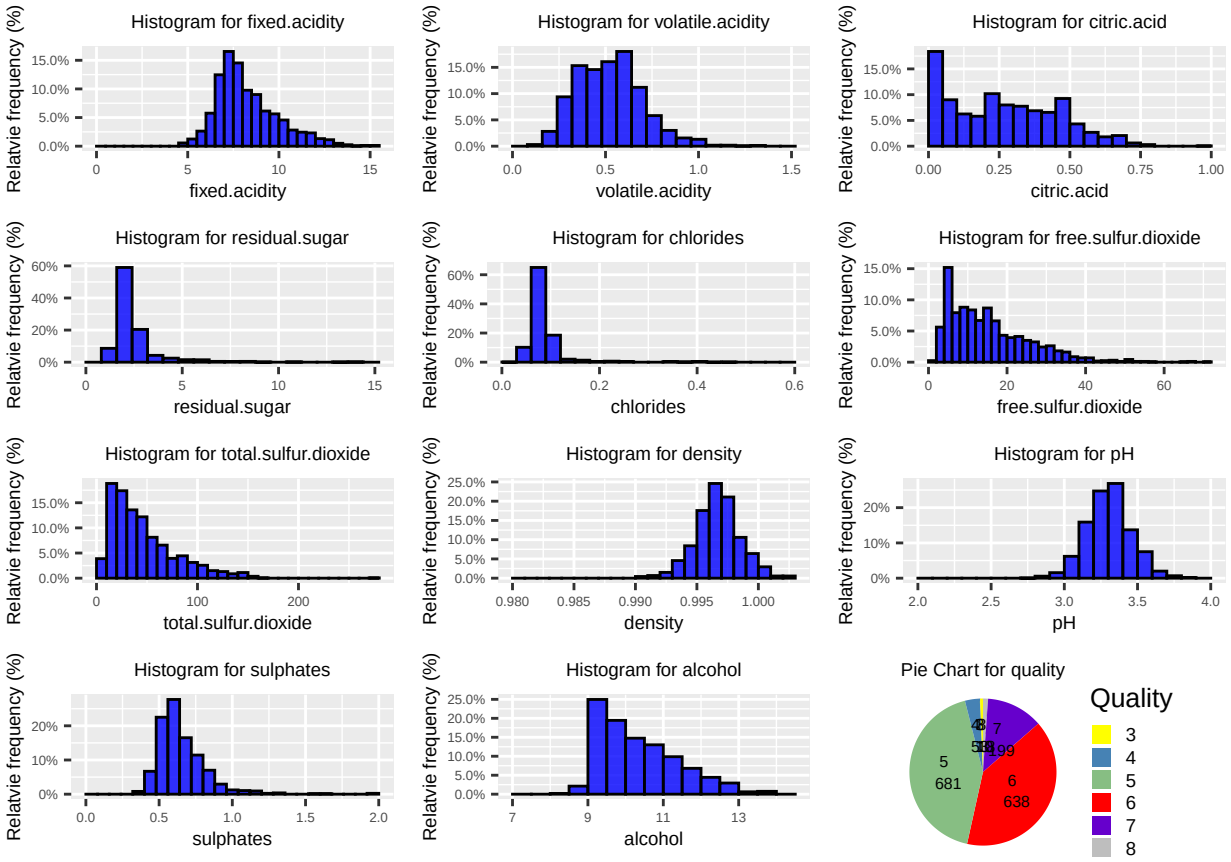
```
##        fixed.acidity     volatile.acidity          citric.acid
##                    0                    0                  132
##       residual.sugar            chlorides  free.sulfur.dioxide
##                    0                    0                    0
## total.sulfur.dioxide         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

Després d'aquest primer anàlisi veiem que l'únic atribut que ofereix registres amb valors 0 és el que fa referència a l'acid cítric, després de consultar vàries fonts (com per exemple: https://wineserver.ucdavis.edu/industry-info/enology/methods-and-techniques/common-chemical-reagents/citric-acid, https://www.randoxfood.com/why-is-testing-for-citric-acid-important-in-winemaking/) desaconsellen utilitzar aquest tipus d'àcid perquè pot provocar el creixament de bactèries no desitjades i en cas de fer-lo servir s'ha de fer en quantitats molt baixes (< 1g/l) per això considerem aquests valors com a vàlids i en aquest apartat no hi ha res més a fer.

3.2. Identificació i tractament de valors extrems.

Per tal d'avaluar els valors extrems ens ajudem de les gràfiques d'histogrames, ja que tots els atributs són de tipus numèric, per tal de veure quina distribució segueixen.

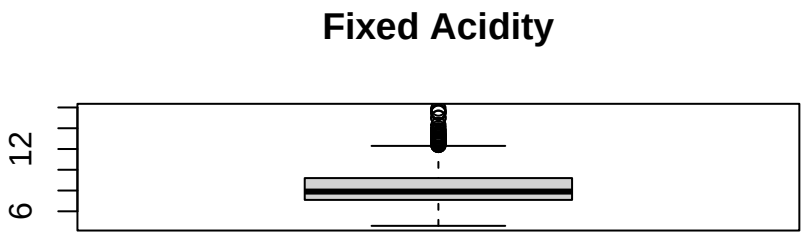
```r
gs <- list(grobTree(ggplotGrob(p1)), grobTree(ggplotGrob(p2)), grobTree(ggplotGrob(p3)),
           grobTree(ggplotGrob(p4)), grobTree(ggplotGrob(p5)), grobTree(ggplotGrob(p6)),
           grobTree(ggplotGrob(p7)), grobTree(ggplotGrob(p8)), grobTree(ggplotGrob(p9)),
           grobTree(ggplotGrob(p10)), grobTree(ggplotGrob(p11)), grobTree(ggplotGrob(p12)))
lay <- rbind(c(1,1,2,2,3,3), c(4,4,5,5,6,6), c(7,7,8,8,9,9), c(10,10,11,11,12,12))
grid.arrange(grobs = gs, layout_matrix = lay)
```



Analitzant els histogrames de les gràfiques anteriors, considerem que els atributs que mereixen ser analitzats en profunditat per detectar els valors extrems són els següents: **fixed.acidity, volatile.acidity, residual.sugar, chlorides, free.sulfur.dioxide, total.sulfur.dioxide i sulphates**.

Donat que el nombre de registres que presenten valors extrems és molt elevat, la nostra decisió és substituir aquests valors per la mitjana que prenen segons la categoria a la qual està classificat cada registre, ja que si obtéssim per eliminar registres considerem que es perdrien moltes mostres. Per tal de validar que la substitució s'ha fet de forma correcte, en cada iteració mostrem el valor abans i després del registre a evaluar.

```
bp.fa <- boxplot(df$fixed.acidity, main = "Fixed Acidity")
```



```
bp.fa$out
```

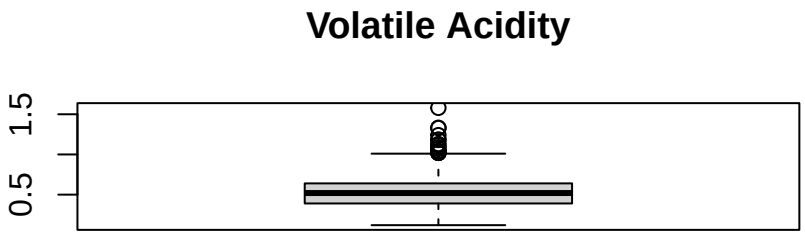
```
##  [1] 12.8 12.8 15.0 15.0 12.5 13.3 13.4 12.4 12.5 13.8 13.5 12.6 12.5 12.8 12.8
## [16] 14.0 13.7 13.7 12.7 12.5 12.8 12.6 15.6 12.5 13.0 12.5 13.3 12.4 12.5 12.9
## [31] 14.3 12.4 15.5 15.5 15.6 13.0 12.7 13.0 12.7 12.4 12.7 13.2 13.2 13.2 15.9
## [46] 13.3 12.9 12.6 12.6
```

```
for(q in unique(df$quality)) {
  idx <- which((df$fixed.acidity > mean(df$fixed.acidity) + 3*sd(df$fixed.acidity))
      & (df$quality == q))
  if(length(idx) > 0) {
    print(q)
    print(df$fixed.acidity[idx])
    m <- mean(df$fixed.acidity[(df$fixed.acidity <= mean(df$fixed.acidity) +
                                      3*sd(df$fixed.acidity)) & (df$quality == q)])

    df$fixed.acidity[idx] <- m
    print(df$fixed.acidity[idx])
  }
}
```

```
## [1] 5
## [1] 15.5 15.5 15.6 15.9
## [1] 8.123191 8.123191 8.123191 8.123191
## [1] 6
## [1] 13.8 14.0 13.7 13.7 14.3
## [1] 8.303318 8.303318 8.303318 8.303318 8.303318
## [1] 7
## [1] 15.0 15.0 15.6
## [1] 8.77551 8.77551 8.77551
```

```
bp.va <- boxplot(df$volatile.acidity, main = "Volatile Acidity")
```




```
bp.va$out
```

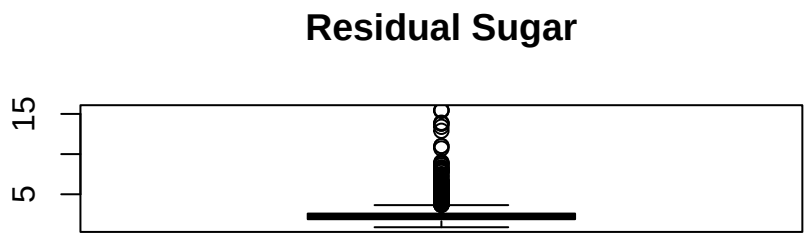
```
##  [1] 1.130 1.020 1.070 1.330 1.330 1.040 1.090 1.040 1.240 1.185 1.020 1.035
## [13] 1.025 1.115 1.020 1.020 1.580 1.180 1.040
```

```
for(q in unique(df$quality)) {
  idx <- which((df$volatile.acidity > mean(df$volatile.acidity) +
                  3*sd(df$volatile.acidity))  & (df$quality == q))
    if(length(idx) > 0) {
    print(q)
    print(df$volatile.acidity[idx])
    m <- mean(df$volatile.acidity[(df$volatile.acidity <= mean(df$volatile.acidity) +
                                       3*sd(df$volatile.acidity)) & (df$quality == q)])

    df$volatile.acidity[idx] <- m
    print(df$volatile.acidity[idx])
  }
}
```

```
## [1] 5
## [1] 1.07 1.33 1.33 1.24 1.18
## [1] 0.5722115 0.5722115 0.5722115 0.5722115 0.5722115
## [1] 4
## [1] 1.130 1.090 1.115
## [1] 0.6689 0.6689 0.6689
## [1] 3
## [1] 1.185 1.580
## [1] 0.76 0.76
```

```
bp.rs <- boxplot(df$residual.sugar, main = "Residual Sugar")
```




```
bp.rs$out
```

```
##   [1]  6.10  6.10  3.80  3.90  4.40 10.70  5.50  5.90  5.90  3.80  5.10  4.65
##  [13]  4.65  5.50  5.50  5.50  5.50  7.30  7.20  3.80  5.60  4.00  4.00  4.00
##  [25]  4.00  7.00  4.00  4.00  6.40  5.60  5.60 11.00 11.00  4.50  4.80  5.80
##  [37]  5.80  3.80  4.40  6.20  4.20  7.90  7.90  3.70  4.50  6.70  6.60  3.70
##  [49]  5.20 15.50  4.10  8.30  6.55  6.55  4.60  6.10  4.30  5.80  5.15  6.30
```

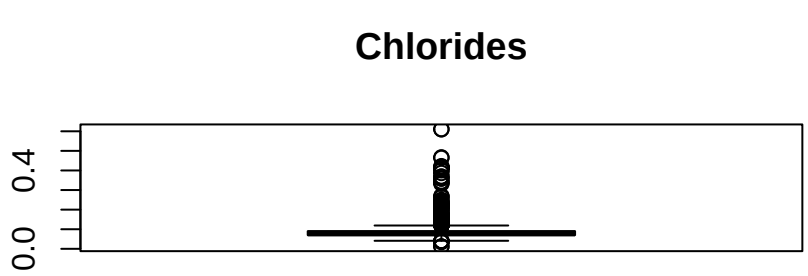
```
##  [61]  4.20  4.20  4.60  4.20  4.60  4.30  4.30  7.90  4.60  5.10  5.60  5.60
##  [73]  6.00  8.60  7.50  4.40  4.25  6.00  3.90  4.20  4.00  4.00  4.00  6.60
##  [85]  6.00  6.00  3.80  9.00  4.60  8.80  8.80  5.00  3.80  4.10  5.90  4.10
##  [97]  6.20  8.90  4.00  3.90  4.00  8.10  8.10  6.40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

```
for(q in unique(df$quality)) {
  idx <- which((df$residual.sugar > mean(df$residual.sugar) + 3*sd(df$residual.sugar))
      & (df$quality == q))
  if(length(idx) > 0) {
    print(q)
    print(df$residual.sugar[idx])
    m <- mean(df$residual.sugar[(df$residual.sugar <= mean(df$residual.sugar) +
                                       3*sd(df$residual.sugar)) & (df$quality == q)])

    df$residual.sugar[idx] <- m
    print(df$residual.sugar[idx])
  }
}
```

```
## [1] 5
##  [1]  7.3  7.2  7.0  7.9  7.9 15.5  7.9  7.5  8.1  8.1  7.8 13.8 13.8  7.8
##  [1] 2.39063 2.39063 2.39063 2.39063 2.39063 2.39063 2.39063 2.39063 2.39063
## [10] 2.39063 2.39063 2.39063 2.39063 2.39063
## [1] 6
##  [1] 10.7 11.0 11.0  8.3  6.3  8.6  9.0  8.8  8.8 13.4 15.4 15.4 13.9
##  [1] 2.30376 2.30376 2.30376 2.30376 2.30376 2.30376 2.30376 2.30376 2.30376
## [10] 2.30376 2.30376 2.30376 2.30376
## [1] 7
##  [1] 5.60 5.60 5.80 5.80 6.70 6.55 6.55 5.80 6.00 6.00 6.00 5.90 6.20 8.90 8.30
## [16] 8.30 5.50 6.20
##  [1] 2.351934 2.351934 2.351934 2.351934 2.351934 2.351934 2.351934 2.351934
##  [9] 2.351934 2.351934 2.351934 2.351934 2.351934 2.351934 2.351934 2.351934
## [17] 2.351934 2.351934
## [1] 4
## [1]  5.6 12.9  6.3
## [1] 2.36 2.36 2.36
## [1] 8
## [1] 6.4 5.2
## [1] 2.175 2.175
## [1] 3
## [1] 5.7
## [1] 2.294444
```

```
bp.ch <- boxplot(df$chlorides, main = "Chlorides")
```




```
bp.ch$out
```

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

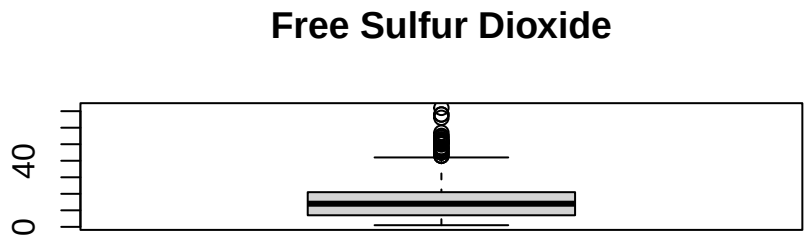
```
for(q in unique(df$quality)) {
  idx <- which((df$chlorides > mean(df$chlorides) + 3*sd(df$chlorides))
      & (df$quality == q))
  if(length(idx) > 0) {
    print(q)
    print(df$chlorides[idx])
    m <- mean(df$chlorides[(df$chlorides <= mean(df$chlorides) +
                                        3*sd(df$chlorides)) & (df$quality == q)])

    df$chlorides[idx] <- m
    print(df$chlorides[idx])
  }
}
```

```
## [1] 5
##  [1] 0.368 0.464 0.401 0.467 0.236 0.360 0.270 0.263 0.611 0.343 0.422 0.387
## [13] 0.414 0.369 0.403 0.415 0.415 0.235
##  [1] 0.08493213 0.08493213 0.08493213 0.08493213 0.08493213 0.08493213
##  [7] 0.08493213 0.08493213 0.08493213 0.08493213 0.08493213 0.08493213
## [13] 0.08493213 0.08493213 0.08493213 0.08493213 0.08493213 0.08493213
## [1] 6
##  [1] 0.341 0.332 0.337 0.213 0.214 0.413 0.226 0.226 0.250 0.222 0.415 0.243
## [13] 0.241 0.190 0.194 0.414 0.214 0.214 0.230
##  [1] 0.07927787 0.07927787 0.07927787 0.07927787 0.07927787 0.07927787
```

```
##  [7] 0.07927787 0.07927787 0.07927787 0.07927787 0.07927787 0.07927787
## [13] 0.07927787 0.07927787 0.07927787 0.07927787 0.07927787 0.07927787
## [19] 0.07927787
## [1] 7
## [1] 0.358 0.216
## [1] 0.07445178 0.07445178
## [1] 4
## [1] 0.172 0.610
## [1] 0.07890196 0.07890196
## [1] 3
## [1] 0.200 0.145 0.267
## [1] 0.08757143 0.08757143 0.08757143
```

```
bp.fsd <- boxplot(df$free.sulfur.dioxide, main = "Free Sulfur Dioxide")
```




```
bp.fsd$out
```

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

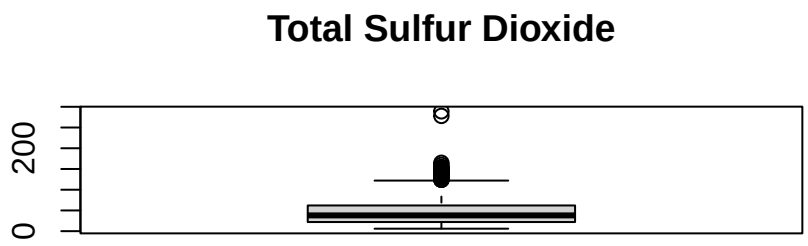
```
for(q in unique(df$quality)) {
  idx <- which((df$free.sulfur.dioxide > mean(df$free.sulfur.dioxide) + 3*sd(df$free.sulfur.dioxide))
      & (df$quality == q))
  if(length(idx) > 0) {
    print(q)
    print(df$free.sulfur.dioxide[idx])
    m <- mean(df$free.sulfur.dioxide[(df$free.sulfur.dioxide <= mean(df$free.sulfur.dioxide) +
                                        3*sd(df$free.sulfur.dioxide)) & (df$quality == q)])

    df$free.sulfur.dioxide[idx] <- m
    print(df$free.sulfur.dioxide[idx])
  }
}
```

```
## [1] 5
##  [1] 52 51 50 68 68 57 48 51 51 52 48 48 66
```

```
##  [1] 16.2515 16.2515 16.2515 16.2515 16.2515 16.2515 16.2515 16.2515 16.2515
## [10] 16.2515 16.2515 16.2515 16.2515
## [1] 6
## [1] 52 51 50 48 72 55 55
## [1] 15.27892 15.27892 15.27892 15.27892 15.27892 15.27892 15.27892
## [1] 7
## [1] 54 53 45 45
## [1] 13.32308 13.32308 13.32308 13.32308
```

```
bp.tsd <- boxplot(df$total.sulfur.dioxide, main = "Total Sulfur Dioxide")
```




```
bp.tsd$out
```

```
##  [1] 145 148 136 125 140 136 133 153 134 141 129 128 129 128 143 144 127 126 145
## [20] 144 135 165 124 124 134 124 129 151 133 142 149 147 145 148 155 151 152 125
## [39] 127 139 143 144 130 278 289 135 160 141 141 133 147 147 131 131 131
```

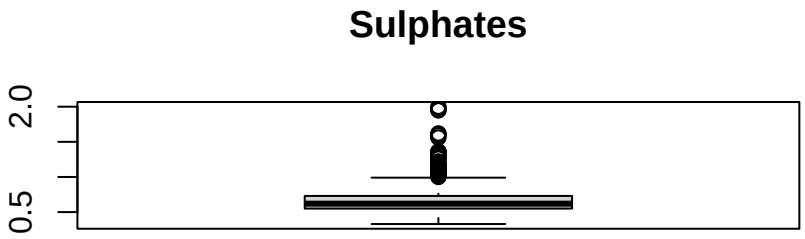
```
for(q in unique(df$quality)) {
  idx <- which((df$total.sulfur.dioxide > mean(df$total.sulfur.dioxide) + 3*sd(df$total.sulfur.dioxide))
      & (df$quality == q))
  if(length(idx) > 0) {
    print(q)
    print(df$total.sulfur.dioxide[idx])
    m <- mean(df$total.sulfur.dioxide[(df$total.sulfur.dioxide <= mean(df$total.sulfur.dioxide) +
                                       3*sd(df$total.sulfur.dioxide)) & (df$quality == q)])

    df$total.sulfur.dioxide[idx] <- m
    print(df$total.sulfur.dioxide[idx])
  }
}
```

```
## [1] 5
## [1] 148 153 151 147 155 151 152 147 147
## [1] 55.26042 55.26042 55.26042 55.26042 55.26042 55.26042 55.26042 55.26042
## [9] 55.26042
## [1] 6
```

```
## [1] 165 149 148 160
## [1] 40.14669 40.14669 40.14669 40.14669
## [1] 7
## [1] 278 289
## [1] 32.49746 32.49746
```

```
bp.su <- boxplot(df$sulphates, main = "Sulphates")
```




```
bp.su$out
```

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

```
for(q in unique(df$quality)) {
  idx <- which((df$sulphates > mean(df$sulphates) + 3*sd(df$sulphates))
      & (df$quality == q))
  if(length(idx) > 0) {
    print(q)
    print(df$sulphates[idx])
    m <- mean(df$sulphates[(df$sulphates <= mean(df$sulphates) +
                                      3*sd(df$sulphates)) & (df$quality == q)])

    df$sulphates[idx] <- m
    print(df$sulphates[idx])
  }
}
```

```
## [1] 5
##  [1] 1.56 1.28 1.20 1.28 1.22 1.98 1.31 1.59 1.26 1.17 1.62 1.18 1.34 1.17 1.17
## [16] 1.17 1.17
##  [1] 0.6027259 0.6027259 0.6027259 0.6027259 0.6027259 0.6027259 0.6027259
##  [8] 0.6027259 0.6027259 0.6027259 0.6027259 0.6027259 0.6027259 0.6027259 0.6027259
## [15] 0.6027259 0.6027259 0.6027259
## [1] 6
```

```
##  [1] 1.95 1.95 1.61 1.18 1.11 1.36 1.36 1.16 1.33 1.18
##  [1] 0.6634873 0.6634873 0.6634873 0.6634873 0.6634873 0.6634873 0.6634873
##  [8] 0.6634873 0.6634873 0.6634873
## [1] 7
## [1] 1.08 1.36 1.13 1.13
## [1] 0.732359 0.732359 0.732359 0.732359
## [1] 4
## [1] 1.12 2.00 1.08
## [1] 0.5482 0.5482 0.5482
## [1] 8
## [1] 1.1
## [1] 0.7482353
```

4. Anàlisi de les dades.

4.1. Selecció dels grups de dades que es volen analitzar/comparar (planificació dels anàlisis a aplicar).

Per tal de poder realitzar els anàlisis de manera correcte considerem com a millor opció la de crear un atribut nou de tipus categòric que ens permeti agrupar les qualitats dels vins en bones (si **quality** $>= 6$) i normals (si **quality** < 6). Ja que més endavant volem fer un contrast d'hipòtesis sobre un dels atributs i una regressió logística per tal de crear un model de classificació en dues categories, per això hem cregut que el més adient era crear aquest atribut nou.

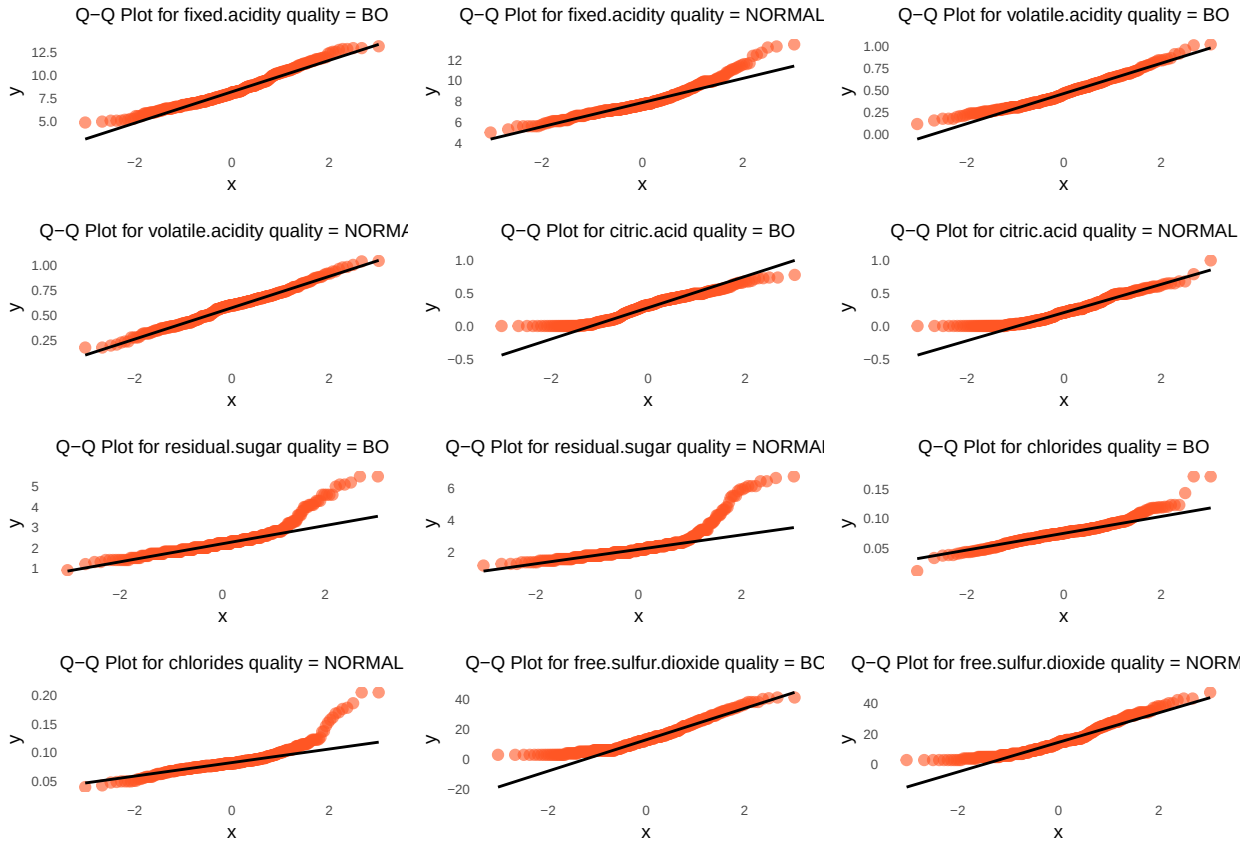
```
df$new.quality <- rep("NORMAL", length(df$quality))
idx <- which(df$quality >= 6)
df$new.quality[idx] <- rep("BO", length(idx))
df$new.quality <- factor(df$new.quality, levels = c("BO", "NORMAL"),
                         labels = c("BO", "NORMAL"))
summary(df)
```

```
##  fixed.acidity    volatile.acidity  citric.acid    residual.sugar
##  Min.   : 4.600   Min.   :0.1200   Min.   :0.000   Min.   :0.900
##  1st Qu.: 7.100   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.:1.900
##  Median : 7.900   Median :0.5200   Median :0.260   Median :2.200
##  Mean   : 8.271   Mean   :0.5242   Mean   :0.271   Mean   :2.347
##  3rd Qu.: 9.150   3rd Qu.:0.6350   3rd Qu.:0.420   3rd Qu.:2.500
##  Max.   :13.500   Max.   :1.0400   Max.   :1.000   Max.   :6.700
##    chlorides     free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.012   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
##  1st Qu.:0.070   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.079   Median :13.32       Median : 38.00       Median :0.9968
##  Mean   :0.081   Mean   :15.30       Mean   : 45.33       Mean   :0.9967
##  3rd Qu.:0.088   3rd Qu.:21.00       3rd Qu.: 61.00       3rd Qu.:0.9978
##  Max.   :0.205   Max.   :47.00       Max.   :145.00       Max.   :1.0037
##        pH          sulphates         alcohol         quality      new.quality
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000   BO    :855
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000   NORMAL:744
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6427   Mean   :10.42   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7200   3rd Qu.:11.10   3rd Qu.:6.000
##  Max.   :4.010   Max.   :1.1500   Max.   :14.90   Max.   :8.000
```

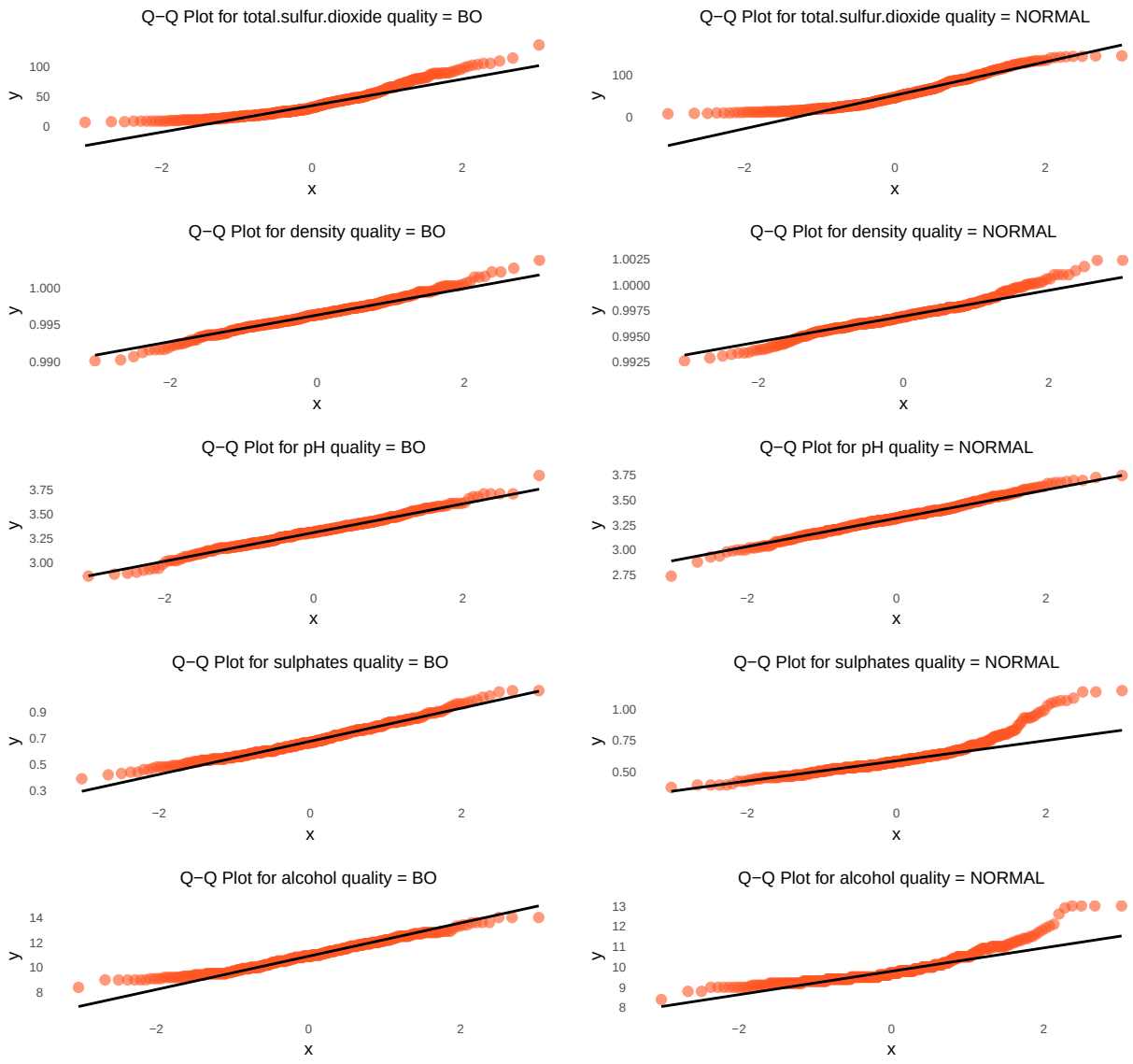
4.2. Comprovació de la normalitat i homogeneïtat de la variància.

Per tal de poder comprovar la normalitat dels diferents atributs amb les dues categories noves creades hem decidit fer-ho per inspecció visual amb els gràfics Q-Q. Aquests ens permeten veure les mostres en un pla i una recta, si les mostres en aquest pla segueixen la línia recta es pot considerar que la mostra segueix una distribució normal.

```
gs <- list(grobTree(ggplotGrob(q1)), grobTree(ggplotGrob(q2)), grobTree(ggplotGrob(q3)),
           grobTree(ggplotGrob(q4)), grobTree(ggplotGrob(q5)), grobTree(ggplotGrob(q6)),
           grobTree(ggplotGrob(q7)), grobTree(ggplotGrob(q8)), grobTree(ggplotGrob(q9)),
           grobTree(ggplotGrob(q10)), grobTree(ggplotGrob(q11)), grobTree(ggplotGrob(q12)))
lay <- rbind(c(1,2,3), c(4,5,6), c(7,8,9), c(10,11,12))
grid.arrange(grobs = gs, layout_matrix = lay)
```




```
gs <- list(grobTree(ggplotGrob(q13)), grobTree(ggplotGrob(q14)), grobTree(ggplotGrob(q15)),
           grobTree(ggplotGrob(q16)), grobTree(ggplotGrob(q17)), grobTree(ggplotGrob(q18)),
           grobTree(ggplotGrob(q19)), grobTree(ggplotGrob(q20)), grobTree(ggplotGrob(q21)),
           grobTree(ggplotGrob(q22)))
lay <- rbind(c(1,2), c(3,4),c(5,6), c(7,8), c(9,10))
grid.arrange(grobs = gs, layout_matrix = lay)
```



Després de graficar tots els atributs en el gràfic Q-Q corresponen podem considerar que tots segueixen de forma aproximada la recta que marca la normalitat, per tant podem afirmar que tots els atributs passen de manera satisfactòria la comprovació de normalitat.

A continuació passarem a comprovar la homogeneïtat entre variàncies, per fer-ho disposem del mètode paramètric (Test de Levene) o del no paramètric (Fligner-Killeen), en el nostre cas, com que anteriorment ja hem comprovat que totes les variables es podrien considerar que segueixen una distribució normal utilitzarem el Test de Levene per comprovar la homoscedasticitat de tots els atributs numèrics.

```
with(df, leveneTest(fixed.acidity, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value   Pr(>F)
## group    5  6.2078 1.04e-05 ***
##       1593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
with(df, leveneTest(volatile.acidity, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value  Pr(>F)
## group    5  2.4283 0.03335 *
##       1593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
with(df, leveneTest(citric.acid, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value  Pr(>F)
## group    5  2.4189 0.03397 *
##       1593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
with(df, leveneTest(residual.sugar, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value Pr(>F)
## group    5  2.0597 0.0678 .
##       1593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
with(df, leveneTest(chlorides, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value Pr(>F)
## group    5  1.0163 0.4064
##       1593
```

```
with(df, leveneTest(free.sulfur.dioxide, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value Pr(>F)
## group    5  1.5049 0.1851
##       1593
```

```
with(df, leveneTest(total.sulfur.dioxide, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value    Pr(>F)
## group    5  27.562 < 2.2e-16 ***
##       1593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
with(df, leveneTest(density, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value    Pr(>F)
## group    5  9.7725 3.274e-09 ***
##       1593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
with(df, leveneTest(pH, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value Pr(>F)
## group    5  0.2978 0.9143
##       1593
```

```
with(df, leveneTest(sulphates, quality))
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value  Pr(>F)
## group    5  2.5651 0.02547 *
##       1593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
with(df, leveneTest(alcohol, quality))
```

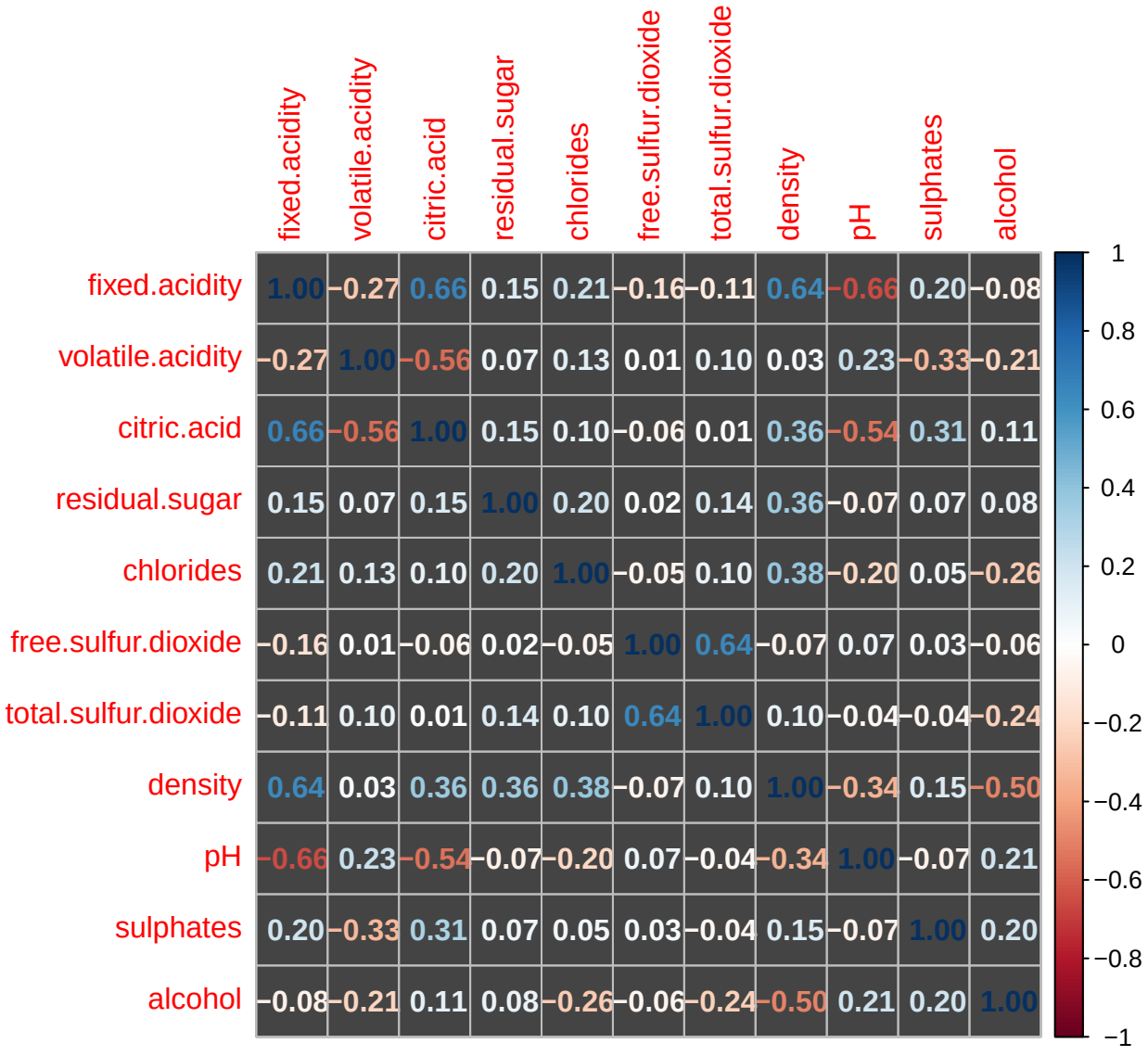
```
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value    Pr(>F)
## group    5  24.226 < 2.2e-16 ***
##       1593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tal com ens mostra la funció d'R només hi ha cinc atributs que passen el test d'homogeneïtat de la variància ja que els seu p-valor > 0.05, aquests atributs són els següents: **residual.sugar, chlorides, free.sulfur.dioxide, pH i sulphates** els quals ens permetrant utilitzar testos de tipus paramètric, per la resta hauríem d'utilitzar testos de tipus no paramètric.

4.3. Aplicació de proves estadístiques per comparar els grups de dades. En funció de les dades i de l'objectiu de l'estudi, aplicar proves de contrast d'hipòtesis, correlacions, regressions, etc. Aplicar almenys tres mètodes d'anàlisi diferents.

Com que l'objectiu final del nostre estudi és crear un model de regressió logística per poder classificar els vins en bons o normals amb el nou atribut creat anteriorment, creiem necessari primer realitzar un test de correlació per tal de descartar els atributs que mostrin una forta correlació, ja que si es dóna el cas, el nostre model donarà més importància a aquests atributs i possiblement no classificarà de manera correcte els vins.

```
df_corr <- df[,c(-12,-13)]
array_corr <- rcorr(as.matrix(df_corr))
corrplot(array_corr$r, method = 'number', bg = '#444444')
```



	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol
fixed.acidity	1.00	−0.27	0.66	0.15	0.21	−0.16	−0.11	0.64	−0.66	0.20	−0.08
volatile.acidity	−0.27	1.00	−0.56	0.07	0.13	0.01	0.10	0.03	0.23	−0.33	−0.21
citric.acid	0.66	−0.56	1.00	0.15	0.10	−0.06	0.01	0.36	−0.54	0.31	0.11
residual.sugar	0.15	0.07	0.15	1.00	0.20	0.02	0.14	0.36	−0.07	0.07	0.08
chlorides	0.21	0.13	0.10	0.20	1.00	−0.05	0.10	0.38	−0.20	0.05	−0.26
free.sulfur.dioxide	−0.16	0.01	−0.06	0.02	−0.05	1.00	0.64	−0.07	0.07	0.03	−0.06
total.sulfur.dioxide	−0.11	0.10	0.01	0.14	0.10	0.64	1.00	0.10	−0.04	−0.04	−0.24
density	0.64	0.03	0.36	0.36	0.38	−0.07	0.10	1.00	−0.34	0.15	−0.50
pH	−0.66	0.23	−0.54	−0.07	−0.20	0.07	−0.04	−0.34	1.00	−0.07	0.21
sulphates	0.20	−0.33	0.31	0.07	0.05	0.03	−0.04	0.15	−0.07	1.00	0.20
alcohol	−0.08	−0.21	0.11	0.08	−0.26	−0.06	−0.24	−0.50	0.21	0.20	1.00

De la taula anterior podem veure que no hi ha cap atribut que estigui fortament correlacionat amb algun altre, sí que s'observa alguna correlació per sobre del 60%, però aquests valors no són prou significatius per haver de descartar algun atribut, ja que el model no quedarà esbiaixat per culpa d'aquesta dependència.

La següent prova estadística que volem realitzar és un contrast d'hipòtesis sobre un dels atributs separat per la categoria de vi bo o vi normal. Per tal de fer això, hem de formular la hipòtesi nul · la i l'alternativa. En aquest cas ens centrarem en el nivell d'acidesa dels vins (pH), per tant, la nostra hipòtesi nul · la és que els vins bons tenen un valor de pH igual als vins normals. Al ser un test de tipus paramètric el que fem és comparar les mitjanes poblacionals de cada grup, per tant, el que estem afirmant amb la hipòtesi nul · la és que les mitjana de pH han de ser iguals:

$$H_0 : \mu_1 = \mu_2 \rightarrow H_0 : \mu_1 - \mu_2 = 0$$

Com a hipòtesi alternativa volem verificar que els vins bons tenen un valor de pH inferior als vins normals,

és a dir, que els vins bons són menys àcids. El que estem afirmant amb la hipòtesi alternativa és que la mitjana de pH dels vins bons és inferior a la mitjana de pH dels vins normals:

$$H_0 : \mu_1 < \mu_2 \rightarrow H_0 : \mu_1 - \mu_2 < 0$$

```
t.test(df$pH[df$new.quality == "BO"], df$pH[df$new.quality == "NORMAL"],
       alternative = "less", var.equal = FALSE, paired = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  df$pH[df$new.quality == "BO"] and df$pH[df$new.quality == "NORMAL"]
## t = -0.13045, df = 1567.3, p-value = 0.4481
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##        -Inf 0.01173186
## sample estimates:
## mean of x mean of y
##  3.310643  3.311653
```

Un cop realitzat el test T de Student veiem que la hipòtesi nul · la no es compleix, per tant queda descartat que els vins bons i normals tenen el mateix nivell de pH. Aquest test també ens mostra el valor de les mitjanes poblacionals de cada grup i podem verificar com la nostra hipòtesi alternativa és correcta, els vins normals tenen un nivell de pH més gran que els bons.

Finalment, la última prova estadística que volem aplicar sobre el joc de dades és la de crear un model de regressió logística per tal de ser capaços de dir si un vi serà bo o normal en funció dels diferents atributs del joc de dades. Per començar el que hem de fer és separar el joc de dades entre entrenament i test, normalment es sol agafar el 80% de les mostres per l'entrenament i el 20% restant per la validació del model i veure que és capaç de generalitzar correctament.

```
df_2 <- df[,-12]
sample = sample.split(df_2$new.quality, SplitRatio = .8)
train = subset(df_2, sample == TRUE)
test  = subset(df_2, sample == FALSE)
```

Un cop ja tenim el joc de dades separat en entrenament i test el que fem és entrenar el model amb les dades d'entrenament.

```
Model <- glm(formula = new.quality ~ .,
                family = binomial(link = "logit"), data = train)
summary(Model)
```

```
##
## Call:
## glm(formula = new.quality ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4322  -0.8259  -0.2892   0.8534   2.2190
##
## Coefficients:
```

```
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           34.741224  72.865114   0.477  0.63351
## fixed.acidity         -0.076759   0.082855  -0.926  0.35423
## volatile.acidity       3.094215   0.547826   5.648 1.62e-08 ***
## citric.acid            1.182264   0.599566   1.972  0.04862 *
## residual.sugar         0.130970   0.108840   1.203  0.22885
## chlorides             10.569345   3.862831   2.736  0.00622 **
## free.sulfur.dioxide   -0.017227   0.010313  -1.670  0.09483 .
## total.sulfur.dioxide   0.015147   0.003479   4.353 1.34e-05 ***
## density              -28.867179  73.800293  -0.391  0.69568
## pH                     0.899081   0.666172   1.350  0.17714
## sulphates             -4.017837   0.601522  -6.679 2.40e-11 ***
## alcohol               -0.911642   0.108150  -8.429  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1766.9  on 1278  degrees of freedom
## Residual deviance: 1307.0  on 1267  degrees of freedom
## AIC: 1331
##
## Number of Fisher Scoring iterations: 4
```

Un cop fet l'entrenament, la sortida de la funció d'R ens mostra amb *** els atributs més rellevants o que tenen més importància a l'hora de classificar les dades. Un cop ja tenim el model entrenat, el següent pas és validar-lo per veure si és capaç de generalitzar de manera correcte i classificar bé dades que no ha vist durant l'entrenament. Primer farem la predicció i després mostrarem una taula els resultats obtinguts de la predicció.

```
pred <- predict(Model, test, type = "response")
t_conf <- table(test$new.quality, pred >= 0.5)
VP <- t_conf[1,1]
VN <- t_conf[2,2]
FP <- t_conf[1,2]
FN <- t_conf[2,1]
```

Table 1: Matriu de confusió

	BO	NORMAL
BO	131	40
NORMAL	34	115

```
t_conf%>%
  kbl(caption = "Matriu de confusió", align = 'c', col.names = c("BO", "NORMAL")) %>%
  kable_classic(c("striped", "hover", "condensed", "responsive"), full_width = F) %>%
  row_spec(0, bold = TRUE) %>%
  column_spec(1, bold = TRUE)
```

Segons la taula de confusió el nostre model classifica correctament 246 mostres, per contra classifica incorrectament 74 mostres, de les quals 40 són falsos positius i 34 són falsos negatius. Amb això tenim que el percentatge d'encert del model és del 76.875%, tot i que és un valor força bo, un bon model de classificació hauria de tenir un percentatge més elevat.

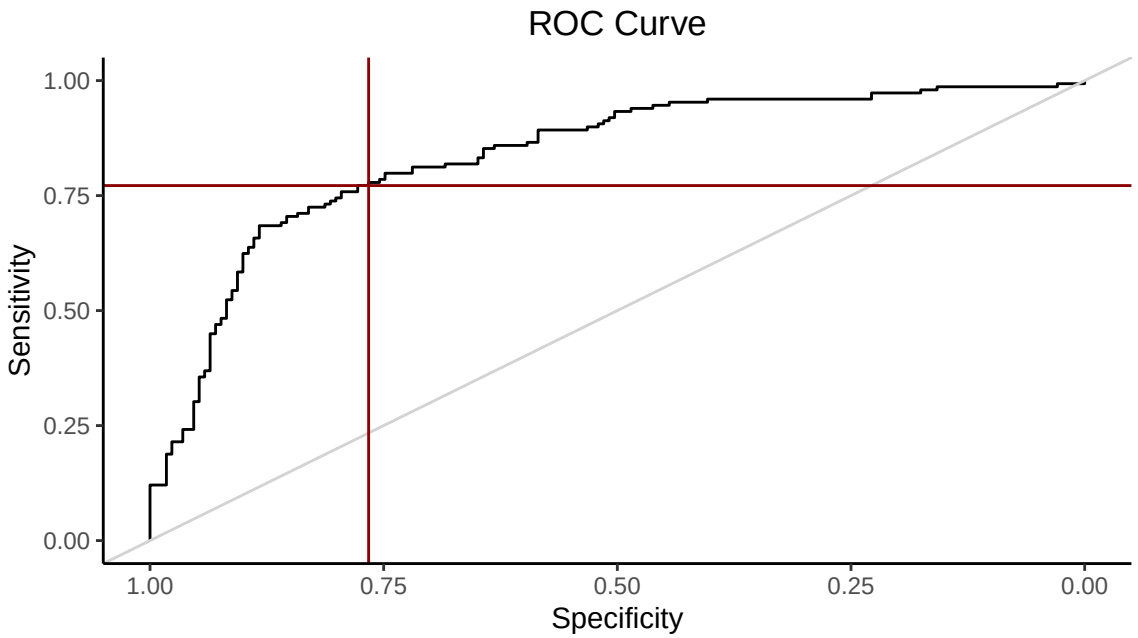
Si analitzem la sensibilitat (proporció de classificats correctament amb resposta positiva) és del 77.181%, el qual ens indica que no és capaç de classificar correctament tots els casos positius i que genera algun fals negatiu.

Si analitzem l'especificitat (proporció de classificats correctament amb resposta negativa) és del 76.608%, aquest valor, igual que abans, ens indica que no és capaç de classificar correctament tots els casos negatius i que genera algun fals positiu.

Si comparem sensibilitat contra especificitat podem concloure que el model classifica millor els casos positius que els negatius, segurament perquè té alguna mostra més d'aquests i tingui una mica de viaix cap aquesta direcció.

A continuació fem el gràfic de la curva ROC del model generat i on les línies horitzontal i vertical corresponen al valor de l'especificitat i la sensibilitat per un valor llindar del 50%.

```
r <- roc(test$new.quality, pred, data=test)
ggroc(r) +
  labs(title="ROC Curve",
       x="Specificity", y = "Sensitivity") +
  geom_abline(intercept = 1, slope = 1, color="lightgrey") +
  geom_hline(yintercept = t_conf[2,2]/(t_conf[2,2] + t_conf[2,1]), color="darkred") +
  geom_vline(xintercept = t_conf[1,1]/(t_conf[1,1] + t_conf[1,2]), color="darkred") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))
```



Finalment calculem el valor de l'àrea sota la corva ROC:

```
auc(r)
```

```
## Area under the curve: 0.8379
```

Tot i que un valor d'àrea sota la corba més gran de 0.8 es pot considerar un bon model, com que l'exactitud està per sota del 80% la nostra conclusió és que és un bon model però no excel · lent que ens permeti discriminar correctament entre un bon vi i un de normal.

Amb aquest valor podem concloure que el model discrimina de forma adequada tot i que ja hem comprovat que no és molt bo en detectar els casos positius (jugador internacional).

5. Representació dels resultats a partir de taules i gràfiques.

A mesura que hem anat resolent els apartats anteriors ja hem fet les gràfiques i taules corresponents, de manera que el fil de lectura no queda truncat en cas de concentrar-ho tot aquí.

6. Resolució del problema. A partir dels resultats obtinguts, quines són les conclusions? Els resultats permeten respondre al problema?

El problema principal que ens havíem plantejat és si a partir dels atributs mesurats d'un vi podríem arribar a trobar un vi BO o NORMAL, per tal de poder construir un model que ens permeti resoldre aquest problema hem creat un atribut nou derivat del de qualitat, de manera que hem agrupat els vins en BO i NORMAL en funció de si la qualitat era igual o superior a 6 o no, respectivament. En vista dels resultats obtinguts en el model de regressió logística podem dir que el model construït no és del tot bo, ja que una exactitud del voltant del 75% considerem que no és prou bona per un model de classificació. Així podem concloure que aquest estudi no ens ha servit per respondre a la pregunta principal que ens havíem plantejat, segurament utilitzant algun altre tipus d'algorisme d'aprenentatge supervisat podríem aconseguir resultats millors i que realment ens ajudessin a respondre la principal pregunta d'aquest problema.

7. Codi: Cal adjuntar el codi, preferiblement en R, amb el que s'ha realitzat la neteja, anàlisi i representació de les dades. Si ho preferiu, també podeu treballar en Python.

Es pot trobar al repositori de github de cada un de nosaltres

```
write.csv(df, "final_dataset.csv", row.names = FALSE)
```

TAULA DE CONTRIBUCIONS		
INVESTIGACIÓ PRÈVIA	OMM	GRR
REDACCIÓ DE LES RESPOSTES	OMM	GRR
DESENVOLUPAMENT DEL CODI	OMM	GRR