UNIVERSITÉ NORBERT ZONGO
U F R/S T
MATHÉMATIQUES
Master 1 Semestre 2
Année académique 2024-2025

BURKINA FASO

La Patrie ou la Mort
nous Vaincrons

Projet de Data Mining

Projet 1

Thème:
magentahttps://archive.ics.uci.edu/ml/datasets/Wine+Quality: régression logistique des vins blancs

Rédigé par :
BARGO Alfred

Enseignant : Dr. SOME Sobom

Juin 2025

Contents

1 Introduction générale **3**

2 Survol de la littérature **3**
2.1 Historiques et concepts de base . . . 3
2.2 Autres modèles . . . 3
2.3 Méthodes d'optimisation et techniques d'entraînement . . . 4
2.4 Régulation et lutte contre le surapprentissage . . . 4
2.5 Cas d'utilisation et applications pratiques . . . 4

3 Définition et motivation **5**
3.1 Définition . . . 5
3.2 Motivation . . . 5

4 Description des données **5**

5 Prétraitement des données et analyse descriptive **6**
5.1 Chargement des bibliothèques . . . 6
5.2 Chargement des données . . . 6
5.3 Aperçu des données . . . 6
5.4 Vérification de présence de données manquantes . . . 7
5.5 Vérification de la présence des données dupliquées . . . 8
5.6 Points extrêmes . . . 8
5.7 Résumé statistique . . . 8
5.8 Variable d'interêt . . . 9
5.9 Normalisation des données . . . 10
5.10 Analyses univariées . . . 10
5.10.1 fixed.acidity . . . 11
5.10.2 volatile.acidity . . . 12
5.10.3 citric.acid . . . 13
5.10.4 residual.sugar . . . 14
5.10.5 chlorides . . . 15
5.10.6 free.sulfur.dioxide . . . 16
5.10.7 total.sulfur.dioxide . . . 18
5.10.8 density . . . 19
5.10.9 pH . . . 21
5.10.10 sulphates . . . 22
5.10.11 alcohol . . . 23
5.10.12 quality . . . 24
5.11 Analyses multivariées . . . 24

6 Régression logistique **26**
6.1 Sélection des caractéristiques pertinentes . . . 26
6.2 Hypothèses . . . 26
6.2.1 Variable binaire . . . 26
6.2.2 Absence de multicolinéarité . . . 26
6.2.3 Indépendance des observations . . . 27
6.2.4 Absence de valeurs aberrantes . . . 27

6.2.5 Présence de suffisament d'observation . . . 27
6.3 Partitionnement de l'ensemble de données . . . 27
6.4 Réalisation de l'ACP . . . 28
6.5 Ré-échantillonnage . . . 28
6.6 Entrainement du modèle de régression logistique . . . 29
6.7 Analyse, interprétation et évaluation du modèle . . . 30
6.7.1 Résumé statistique . . . 30
6.7.2 Significativité globale du modèle . . . 30
6.7.3 Pouvoir explicatif du modèle . . . 31
6.7.4 Evaluation des performances du modèle . . . 31
6.7.5 Prédiction . . . 31
6.7.6 Matrice de confusion . . . 32
6.8 Conclusion sur le modèle . . . 35

7 Comparaison du modèle **35**
7.1 Modèle de forêt aléatoire (Random Forest) . . . 35
7.1.1 Entrainement du modèle . . . 35
7.1.2 Matrice de confusion . . . 36
7.2 Comparaison . . . 37

8 Conclusion **38**

1 Introduction générale

L'industrie viticole joue un rôle majeur dans l'économie et la culture de nombreux pays tels que le Portugal. La qualité du vin, souvent perçue comme un critère subjectif, repose en réalité sur des facteurs mesurables, tels que ses propriétés physico-chimiques. Dans ce contexte, l'analyse des données permet de mieux comprendre les relations entre ces variables et la qualité perçue du vin.

Grâce aux progrès en **data mining** et en **apprentissage automatique**, il est désormais possible d'extraire des connaissances pertinentes à partir de grandes quantités de données. Le jeu de données **Winequality-white** mis à disposition par l'UCI Machine Learning Repository, offre une base précieuse pour mener ce type d'analyse. Il comprend des mesures objectives (telles que l'acidité, le taux d'alcool ou la densité) ainsi qu'une note de qualité attribuée par des dégustateurs.

L'objectif principal de ce projet est d'appliquer des techniques de data mining pour analyser les facteurs influençant la qualité du vin blanc (variante du vin portugais **Vinho Verde**) et éventuellement construire un modèle prédictif. Pour cela, une série d'étapes méthodologiques sera mise en œuvre : exploration des données, traitement, visualisation, création de nouvelles variables, modélisation et évaluation des résultats. Ce travail s'inscrit dans une démarche à la fois analytique et pratique, visant à mettre en évidence l'utilité des méthodes de science des données dans un domaine concret et apprécié.

Ce rapport est rédigé selon le plan suivante: une introduction générale; un survol de la littérature; une définition et motivation; une description du jeu de données; une description des procédures et des analyses statistiques; description de l'entraînement du modèle; interprétation des résultats du modèle et conclusion; comparaison du modèle avec le modèle de régression forêts aléatoires; conclusion

2 Survol de la littérature

Le survol de la littérature sur la régression logistique a pour objectifs de permettre de mieux comprendre l'évolution des méthodes de machine learning, les techniques populaires et les innovations récentes. Nous explorons les principaux aspects :

2.1 Historiques et concepts de base

Les origines de la régression logistique, modèle statistique, remontent aux années 1940 avec des contributions clés du mathématicien Joseph Berkson. Il a développé et popularisé le modèle en introduisant le terme ***"logit"*** dans son article publié en 1944. Il a également proposé l'utilisation de la fonction de ***probabilité logistique*** en alternative à la fonction de probabilité normale dans les ***bio-essais***. Au fil des années, la régression logistique a été intégrée aux techniques d'apprentissage automatique, permettant de prédire des résultats binaires à partir de données complexes. Aux alentours des années 1980, la régression logistique se généralise dans divers domaines, notamment en épidémiologie et en sciences sociales.

2.2 Autres modèles

Random Forest : appelé régression forêt aléatoire, c'est un algorithme d'apprentissage supervisé proposé par Leo Breiman en 2001, utilisant un ensemble d'arbres de décision pour effectuer des régressions. L'algorithme utilise un échantillonnage avec remplacement (bootstrap) des données d'apprentissage pour construire chaque arbre de décision. Chaque arbre est construit indépendamment des autres et la sélection des caractéristiques est aléatoire. Ce modèle est d'une précision élevée et il est moins sensible au surapprentissage.

SVM (Support Vector Machine ou machine à vecteurs de support) : Les SVMs sont une famille d'algorithmes d'apprentissage automatique qui permettent de résoudre des problèmes tant de classification que de régression ou de détection d'anomalie. Ils sont connus pour leurs solides garanties théoriques, leur grande flexibilité ainsi que leur simplicité d'utilisation même sans grande connaissance de data mining. Ils ont été développés dans les années 1990.

XGBoost : Utilisé pour résoudre des problèmes de classification ou de régression, XGBoost (e**X**trême **G**radient **B**oosting) est un modèle de machine learning. Il s'agit d'un modèle amélioré de l'algorithme

d'amplification de gradient (Gradient Boost) utilisé pour résoudre les problématiques courantes d'entreprises tout en se basant sur une quantité minimale de ressources. Il est souvent utilisé en machine learning pour limiter le nombre d'erreurs dans l'analyse prédictive de données.

2.3 Méthodes d'optimisation et techniques d'entraînement

Les méthodes d'optimisation jouent un rôle crucial dans la réussite des modèles de régression logistique :

- Sélection des caractéristiques : cette partie consiste à décider quelles fonctionnalités sont les plus pertinentes et les plus informatives pour le modèle, tout en éliminant celles qui sont redondantes ou bruyantes. Cela peut permettre d'éviter le surapprentissage. Les méthodes couramment utilisées sont : le test du khi-carré, l'ANOVA et l'information mutuelle ; les méthodes d'encapsulation telles que la sélection directe, l'élimination vers l'arrière et l'élimination récursive des caractéristiques ; les méthodes embarquées comme Lasso, Ridge et Elastic Net.
- Réglage des hyperparamètres : il consiste à trouver les valeurs optimales pour les paramètres qui contrôlent le comportement et la complexité du modèle, tels que le taux d'apprentissage, la force de régularisation et le nombre d'itérations. Les méthodes courantes de réglage des hyperparamètres pour la régression logistique incluent la recherche par grille, la recherche aléatoire et l'optimisation bayésienne.
- Prétraitement des données : c'est un processus de transformation et de nettoyage des données afin de les rendre adaptées et compatibles avec le modèle. Les principales méthodes sont l'imputation, la détection et la suppression des valeurs aberrantes, l'encodage et la mise à l'échelle.
- Equilibrage des classes : L'équilibrage de classe est le processus d'ajustement de la distribution de la variable cible dans les données pour avoir une proportion similaire d'exemples positifs et négatifs, tels que 50 % et 50 %. Les techniques courantes d'équilibrage des classes pour la régression logistique comprennent le suréchantillonnage, le sous-échantillonnage et la pondération.
- Évaluation du modèle : c'est le processus qui consiste à mesurer et à comparer les performances du modèle sur des données inconnues ou nouvelles, telles qu'un jeu de test ou un ensemble d'attente. Les mesures courantes pour la régression logistique incluent l'exactitude, la précision, le rappel et le score F1.

2.4 Régulation et lutte contre le surapprentissage

Le modèle de régression logistique est souvent victime de surapprentissage ou de résultats biaisés. Cela peut être dû à un déséquilibre des classes, une présence élevée de données aberrantes. Nombreuses techniques sont utilisées pour y remédier, telles que :

- **Oversampling**: C'est une technique de rééchantillonnage permettant d'augmenter artificiellement la classe minoritaire par tirage avec remise.
- **Undersampling**: C'est une technique de rééchantillonnage qui consiste à baisser le nombre d'observations de la classe majoritaire.
- **SMOTE**(**S**ynthetic **M**inority **O**ver-sampling **T**echnique): Il s'agit d'une technique de suréchantillonnage qui permet de créer des exemples synthétiques pour la classe minoritaire, utilisant la méthode des k plus proches voisins.

2.5 Cas d'utilisation et applications pratiques

La régression logistique est un modèle de machine learning utilisé dans des situations de Probabilités calibrées (estimer une probabilité d'appartenance à une classe); de classification binaire (ex: "bon vin/mauvais vin", "spam/non spam", "malade/non malade"); de variables explicatives continues ou catégorielles.

La régression logistique est utilisée dans le domaine de la médecine pour prédire le risque de diabète à partir de critères biologiques (oui/non) ; dans le domaine du marketing afin de détecter les clients

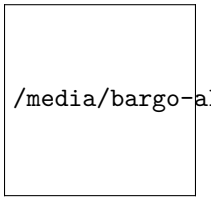


Figure 1: régression linéaire vs logistique

susceptibles d'acheter un produit ; dans l'industrie pour prédire la qualité des produits fabriqués (défectueux/conforme ou bon/mauvais).

3 Définition et motivation

3.1 Définition

La régression logistique est une méthode d'analyse statistique utilisée pour prédire une variable de résultat binaire à partir d'une ou plusieurs variables indépendantes. Contrairement à la régression linéaire, elle est utilisée pour prédire une probabilité binaire.

Le modèle est donné par l'équation suivante: $log(\frac{p}{1-p}) = a_0 + a_1x_1 + ... + a_kx_k$ appelé la fonction **logit** ou p est la probabilité de l'occurence de l'évènement. La probabilité s'obtient à travers la fonction **sigmoïde** donnée par : $P(y = 1) = \frac{1}{1 + exp(-(a_0 + a_1x_1 + ... + a_kx_k))}$

3.2 Motivation

La régression logistique est une méthode simple et rapide ; moins coûteuse en calcul qu'un Random Forest ou un réseau de neurones. Elle est idéale pour un premier benchmark.

Elle permet dans l'interprétation, de comprendre l'impact de chaque variable via les coefficients

La sortie probabiliste est utile pour un seuillage personnalisé.

Elle fonctionne bien sur de petits jeux de données, contrairement au deep learning.

4 Description des données

Le jeu de données **wine+quality** de l'UCI Machine Learning Repository est un ensemble de données lié à la variante du vin rouge et blanc portugais. Les données se composent de propriétés physico-chimiques () de vin blanc au nombre de 11, plus une variable quantifiant la note sensorielle attribuée.

Les données ont été collectées de mai 2004 à février 2007 , fournies par P. Cortez, A. Cerdeira, F. Almeida, T. Matos et J. Reis.

Notre projet se porte sur la variante du vin blanc. Ci-dessous, les caractéristiques de l'ensemble de données:

- **acidité fixe**: La plupart des acides impliqués avec le vin ou fixés ou non volatils (ne s'évaporent pas facilement).
- **acidité volatile**: Quantité d'acide acétique dans le vin.
- **Acide citrique**: Trouvé en petites quantités, l'acide citrique peut ajouter «fraîchir» et arôme aux vins.
- **sucre résiduel** : quantité de sucre restant dans le vin après l'arrêt de la fermentation du vin.
- **Chlorures**: Quantité de sel dans le vin.

- **dioxyde de soufre libre**: Il existe une forme libre de SO2 en équilibre entre le SO2 moléculaire (sous forme de gaz dissous) et l'ion bisulfite.
- **dioxyde de soufre total**: Quantité de formes libres et liées de S02.
- **densité**: Densité du vin
- **pH**: Décrit comment un vin acide ou basique est sur une échelle de 0 (très acide) à 14 (très basique).
- **sulfates**: un additif vinicole pouvant contribuer aux niveaux de dioxyde de soufre (S02).
- **Alcool**: pourcentage d'alcool du vin
- **qualité**: Variable de sortie (basée sur des données sensorielles); score compris entre 0 et 10

5 Prétraitement des données et analyse descriptive

5.1 Chargement des bibliothèques

Nous chargeons les packages nécessaire pour notre travail d'analyse et de modélisation statistique. Ci-dessous les differents packages :

```
# nettoyage de l'environnement de travail
#rm(list=ls())
library(ggplot2)
library(gridExtra)
library(corrplot)
library(dplyr)
library(GGally)
library(caret)
library(themis)
library(ade4)
library(tidymodels)
library(FactoMineR)
library(lmtest)
library(pROC)
library(randomForest)
library(reshape2)
```

5.2 Chargement des données

Nous téléchargeons le jeu de données à travers le lien. (https://archive.ics.uci.edu/ml/datasets/Wine +Quality). Nous importons le jeu de données avec le code suivant:

```
df <- read.csv("~/Data_Mining Project/wine+quality/winequality-white.csv", sep=";")
```

5.3 Aperçu des données

- Nous affichons quelques lignes du jeu de données afin d'avoir un aperçu :

```
##   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
```

- Dimension du jeu de données:

```
## [1] 4898   12
```

Il y a donc 4898 observations dans le jeu de données avec 12 variables.

- Noms des variables:

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

- Structures des variables incluses dans le jeu de données :

```
str(df)
```

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

On remarque que 11 variables sont de type numérique et seule la variable ***quality*** est de type **integer**.

5.4 Vérification de présence de données manquantes

Il est crucial de vérifier la présence de données manquantes et éventuellement les traiter avec la technique appropriée. Dans ce jeu de données, il n'y a pas de données manquantes. Nous vérifions cela avec le code suivant :

```r
colSums(is.na.data.frame(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
```

5.5 Vérification de la présence des données dupliquées

Il est nécessaire également de vérifier s'il n'y a pas de données dupliquées et éventuellement les supprimer de la base de données.

```r
sum(duplicated(df))
```

```
## [1] 937
```

Il y a donc 937 données dupliquées. Nous procédons à leur suppression :

```r
df <- df[!duplicated(df),]
```

5.6 Points extrêmes

Nous vérifions l'existence de données aberrantes :

```
## [1] 1018
```

Il y a donc au total 1018 données aberrantes qui influencent les distributions des différentes variables du jeu de données.

5.7 Résumé statistique

```r
summary(df)
```

```
##  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.: 1.600
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 4.700
##  Mean   : 6.839   Mean   :0.2805   Mean   :0.3343   Mean   : 5.915
##  3rd Qu.: 7.300   3rd Qu.:0.3300   3rd Qu.:0.3900   3rd Qu.: 8.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03500   1st Qu.: 23.00      1st Qu.:106.0        1st Qu.:0.9916
##  Median :0.04200   Median : 33.00      Median :133.0        Median :0.9935
##  Mean   :0.04591   Mean   : 34.89      Mean   :137.2        Mean   :0.9938
##  3rd Qu.:0.05000   3rd Qu.: 45.00      3rd Qu.:166.0        3rd Qu.:0.9957
##  Max.   :0.34600   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.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.180   Median :0.4800   Median :10.40   Median :6.000
##  Mean   :3.195   Mean   :0.4904   Mean   :10.59   Mean   :5.855
##  3rd Qu.:3.290   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

Le résumé statistique permet d'avoir de façon générale la statistique descriptive de chaque variable :

- **fixed.acidity**: L'acidité fixe a une valeur minimale de 0,0800 et une valeur maximale de 1,1000. La moyenne est 6,855, légèrement supérieure à la médiane qui est 6,800. De plus, moins de 25 % des vins ont une acidité fixe inférieure à 6,300, et moins de 75 % ont une acidité fixe inférieure à 7,300.
- **volatile.acidity**: L'acide volatile a une valeur minimale de 3,800 et une valeur maximale de 14,200. La moyenne est 0,2782, légèrement supérieure à la médiane qui est 0,2600. De plus, moins de 25 % des vins ont une acidité volatile inférieure à 0,2100 et moins de 75 % ont une acidité fixe inférieure à 0,3200.
- **citric.acid** : L'acide citrique a une valeur minimale nulle et une valeur maximale de 1,6600. La moyenne est 0,3342, légèrement supérieure à la médiane qui est 0,3200. De plus, moins de 25 % des vins ont un acide citrique inférieur à 0,2700, et moins de 75 % ont une acidité fixe inférieure à 0,3900.
- **residual.sugar**: le sucre résiduel a une valeur minimale 0,600 et une valeur maximale 65,800. La moyenne est 6,391 supérieure à la medianne qui est 5,200 De plus, moins de 25% des vins ont du sucre résiduel inférieure à 1,700 et moins de 75% ont du sucre résiduel inférieure à 9,900.
- **chlorides**: Le chlorure a une valeur minimale de 0,00900 et une valeur maximale de 0,34600. La moyenne est 0,04577 légèrement supérieure à la médiane qui est 0,4300. De plus, moins de 25 % des vins ont des chlorures inférieures à 0,03600, et moins de 75 % ont des chlorures inférieures à 0,0500.
- **free.sulfur.dioxide**: Le dioxyde de soufre libre a une valeur minimale de 2,0 et une valeur maximale de 289,00. La moyenne est 35,31 supérieure à la médiane, qui est 34,00. De plus, moins de 25 % des vins ont du dioxyde de soufre libre inférieur à 23,00, et moins de 75 % ont du dioxyde de soufre libre inférieur à 45,00.
- **total.sulfur.dioxide**: Le dioxyde de soufre total a une valeur minimale de 9,0 et une valeur maximale de 440,00. La moyenne est 138,4 supérieure à la médiane, qui est 134,00. De plus, moins de 25 % des vins ont du dioxyde de soufre total inférieur à 108,0, et moins de 75 % ont du dioxyde de soufre total inférieur à 167,0.
- **density**: La densité a une valeur minimale de 0,9871 et une valeur maximale de 1,0390. La moyenne est 0,9940, légèrement supérieure à la médiane qui est 0,9937. De plus, moins de 25 % des vins ont une densité inférieure à 0,9917 et moins de 75 % ont une densité inférieure à 0,9961.
- **pH**: Le pH a une valeur minimale de 2,720 et une valeur maximale de 3,820. La moyenne est 3,188, légèrement supérieure à la médiane qui est 3,180. De plus, moins de 25 % des vins ont un pH inférieur à 3,090 et moins de 75 % ont un pH inférieur à 3,280.
- **sulphates**: Le sulfate a une valeur minimale de 0,2200 et une valeur maximale de 1,0800. La moyenne est 0,4898, légèrement supérieure à la médiane qui est 0,4700. De plus, moins de 25 % des vins ont des sulfates inférieurs à 0,4100, et moins de 75 % ont des sulfates inférieurs à 0,5500.
- **alcohol**: L'alcool a une valeur minimale de 8,00 et une valeur maximale de 14,20. La moyenne est 10,51, légèrement supérieure à la médiane qui est 10,40. De plus, moins de 25 % des vins ont un alcool inférieur à 9,50, et moins de 75 % ont un alcool inférieur à 11,40.
- **quality**: la qualité a une valeur minimale 3 et une valeur maximale 9 La moyenne est 5,878 inférieure à la médiane, qui est 6. De plus, moins de 25 % des vins ont une qualité inférieure à 5, et moins de 75 % ont une qualité inférieure à 6.

5.8 Variable d'interêt

Nous créons une nouvelle variable binaire qui sera la variable à expliqué du modèle. Cette variable est créer en se basant sur les valeurs de la variable **quality** qui sont les notes sensorielle attribuées aux vins. La condition de création de la variable d'interêt est la suivante: **bon vin"grade<5"/mauvais vin"grade>=5** . Cette condition semble être contre intuitive car, généralement, le vin est de bonne qualité lorsque la note sensorielle (variant entre 0 et 10) est grande.

Néanmoins, nous respecterons cette condition dans notre étude.

Ainsi nous créons la variable **wine_quality** prenant 1 si *quality* < 5 et 0 si *quality* >= 5 dans la ligne de code suivante :

```
# Variable d'interêt
wine_quality <- ifelse(df$quality>=5,0,1)
df$wine_quality <- wine_quality
```

Nous encodons par la suite cette variable en facteur avec le code suivant :

```
# Conversion en facteur
df$wine_quality <- factor(df$wine_quality)
```

Tableau de contingence :

Le tableau de contingence permet de voir la répartition selon la qualité du vin :

```
# table de contingence
contingence_tableau <- table(df$wine_quality)
contingence_tableau
```

```
##
##    0    1
## 3788  173
```

Le tableau de contingence affiche 3788 vins de mauvaise qualité et seulement 173 de bonne qualité.

5.9 Normalisation des données

L'étape de normalisation des données est très importante car elle permet de mettre les valeurs des variables sur une même échelle. Cela permet d'éviter des problèmes de biais dans les modèles.

Nous normalisons les données avec la ligne de code suivante :

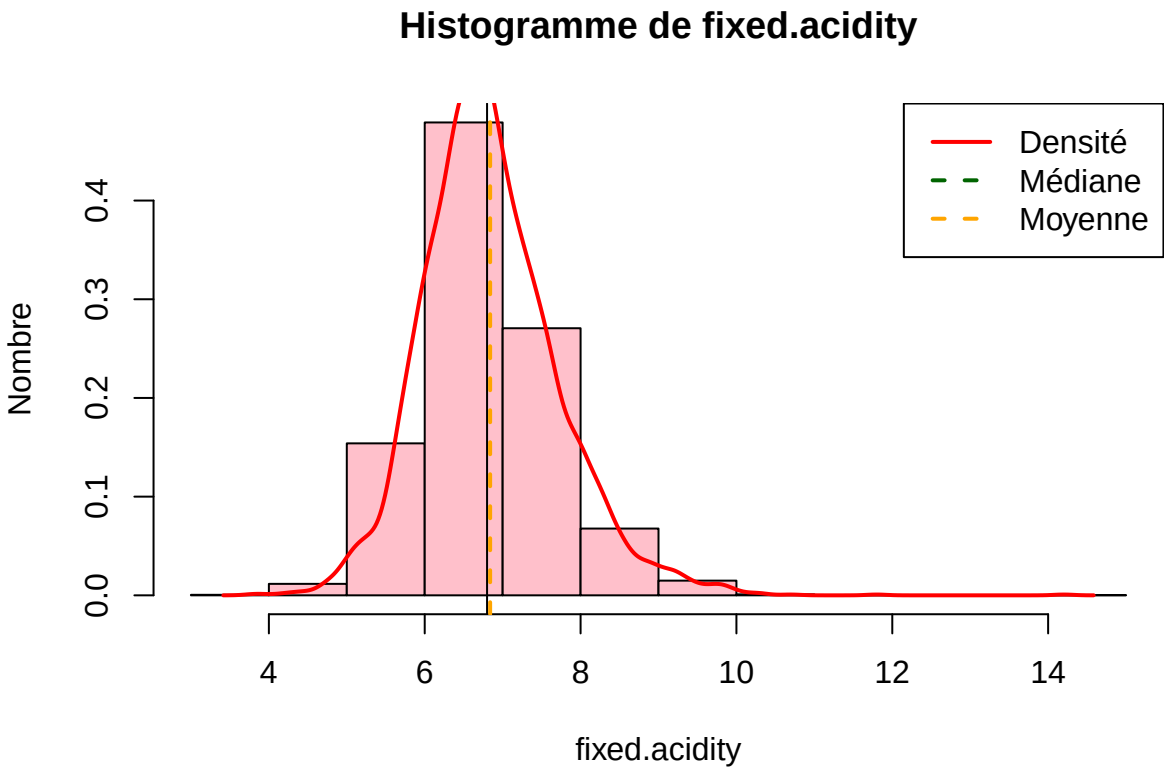
```
# Normalisation
df_norm <- df %>%
  mutate(across(where(is.numeric) & !all_of("wine_quality"), scale))

# Suppression de la variable "quality" au profit de "wine_quality"
df_norm$quality=NULL
```

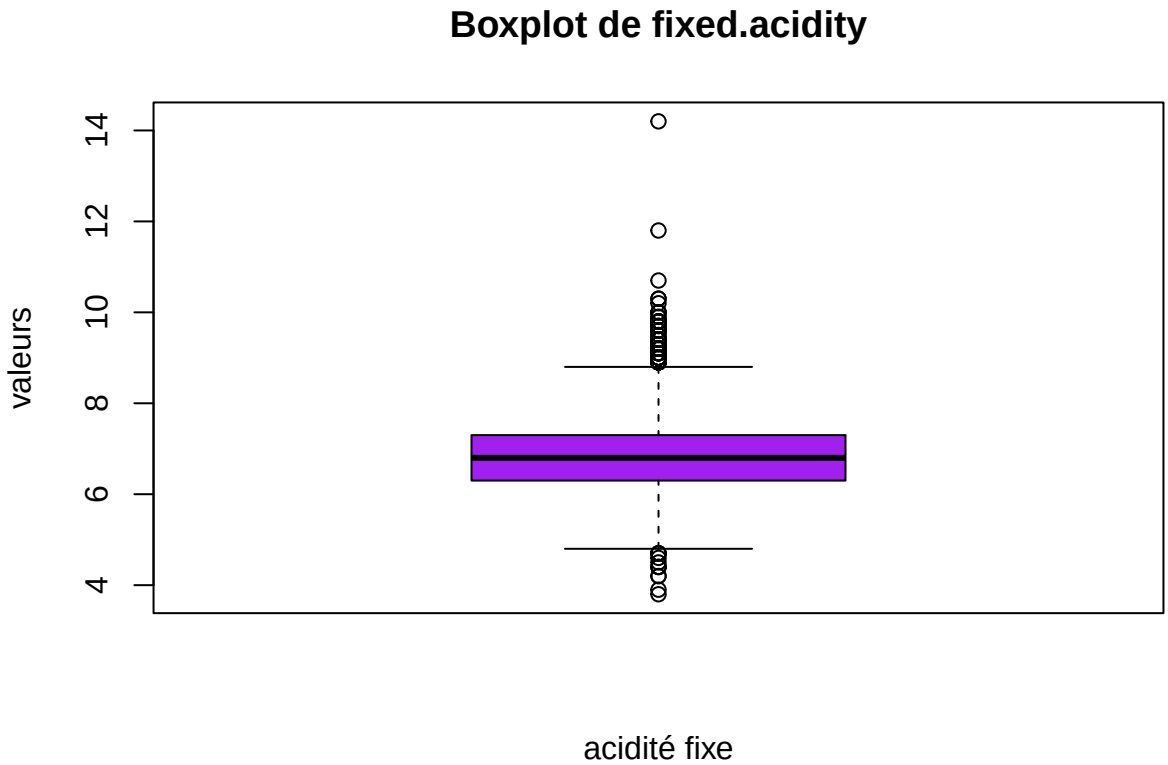
Par la suite, nous utiliserons les données non normalisées pour les analyses statistiques univariées et bivariées.

5.10 Analyses univariées

Après un aperçu un peu général des variables, nous analysons les distributions des variables à travers les histogrammes et les répresentation des boîtes à moustache.

5.10.1 fixed.acidity



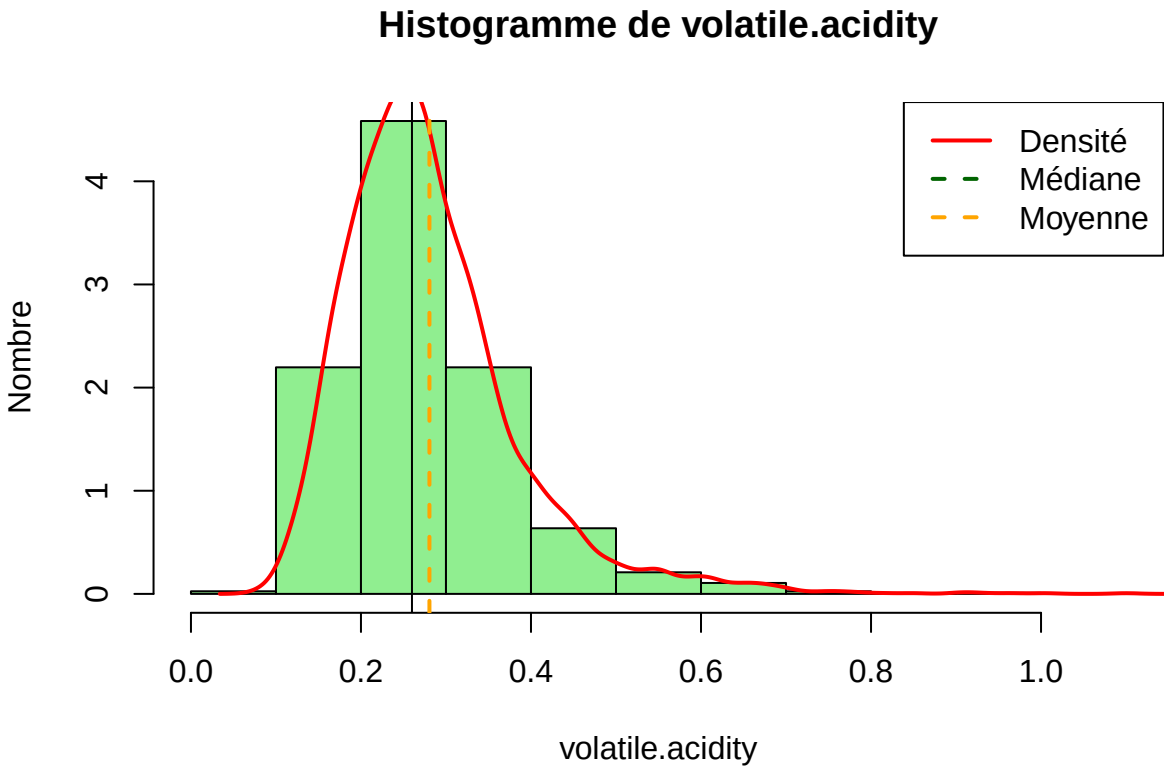


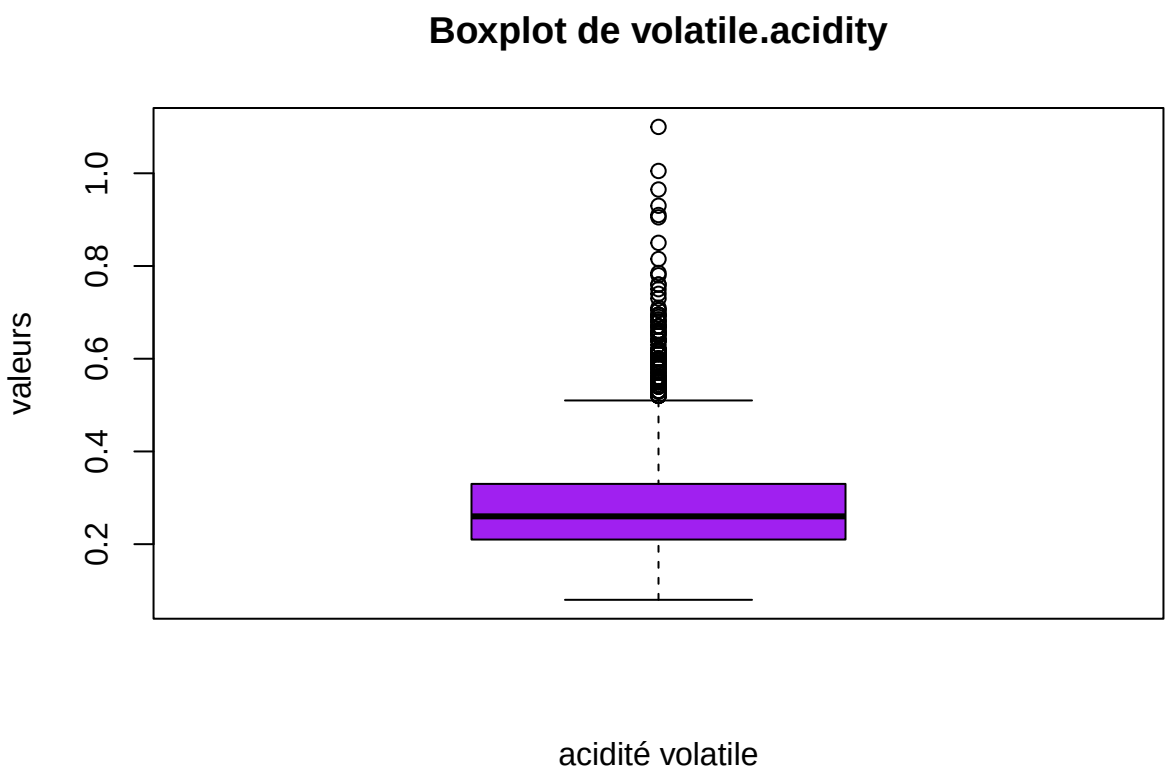
Analyse et interprétation :

L'histogramme de la variable **fixed.acidity** montre une distribution normale.

La représentation de la boîte à moustache montre la présence de valeurs extrêmes. La variabilité des valeurs est de plus faible.

5.10.2 volatile.acidity





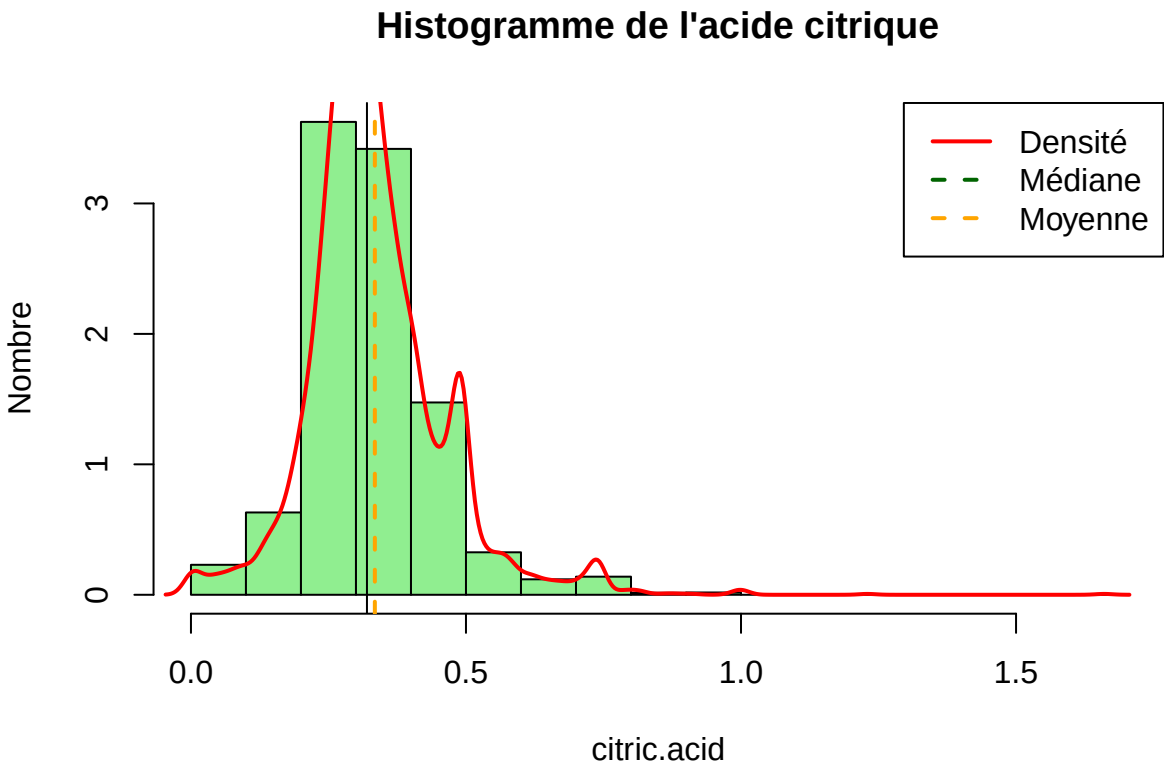
Analyse et interprétation :

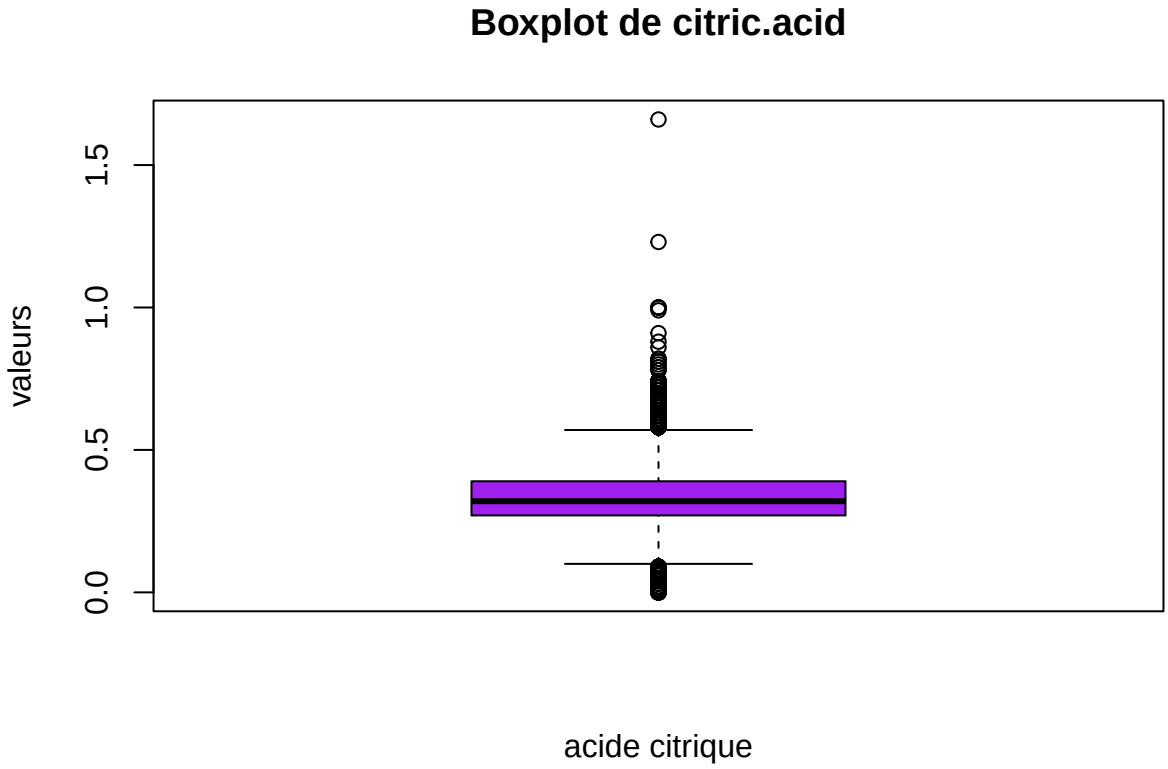
La représentation de l'histogramme de la variable **volatile.acidity** montre une distribution asymétrique à droite. Les données présentent une faible dispersion autour de la moyenne.

La représentation boxplot met en exergue la présence d'un nombre élevé de données aberrantes dans

cette variable et aussi la faible variabilité des données.

5.10.3 citric.acid



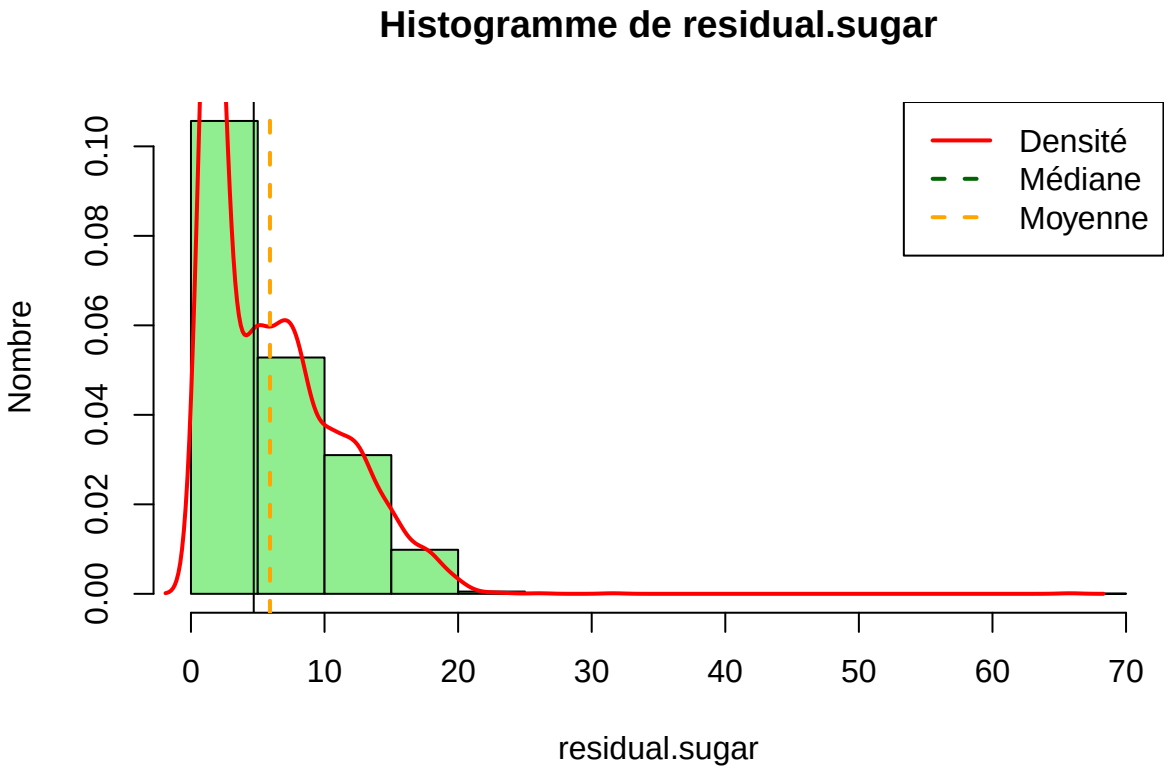


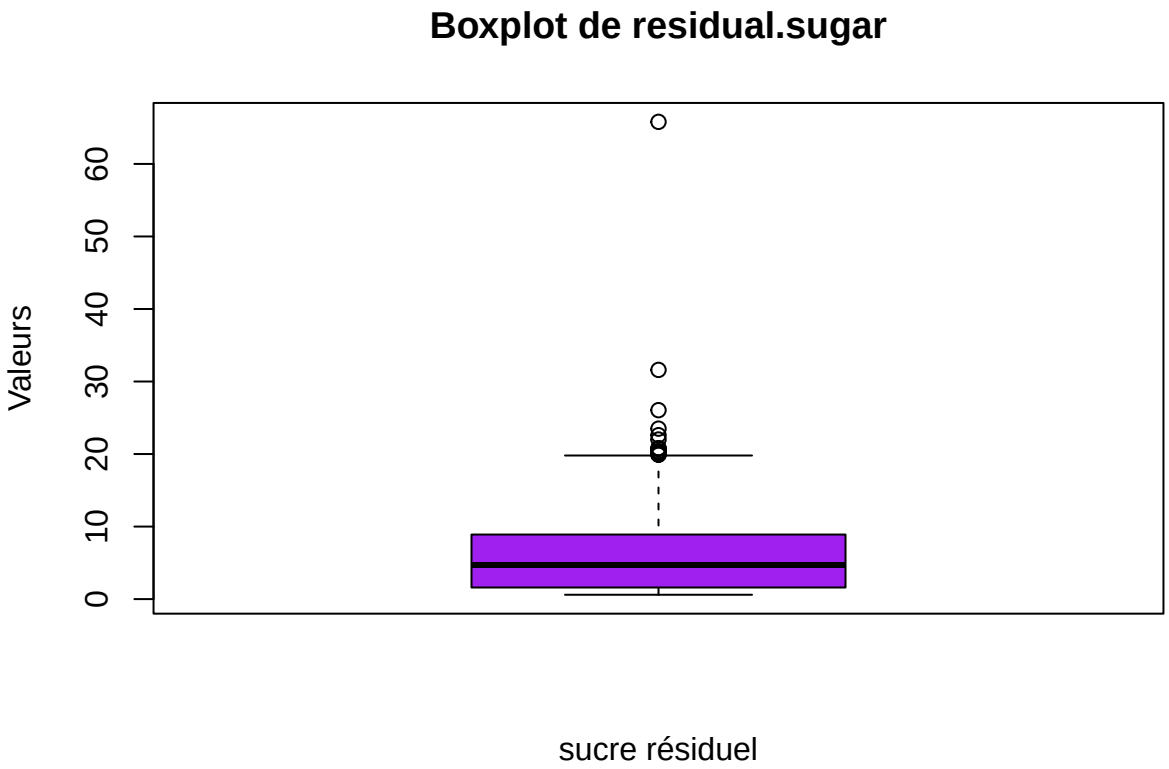
Analyse et interprétation :

L'histogramme de citric.acid montre une distribution légèrement asymétrique à droite.

La représentation du boxplot met en lumière la présence de valeurs aberrantes dans la distribution de l'acide citrique, avec une des valeurs extrêmement grande dépassant 1,5.

5.10.4 residual.sugar



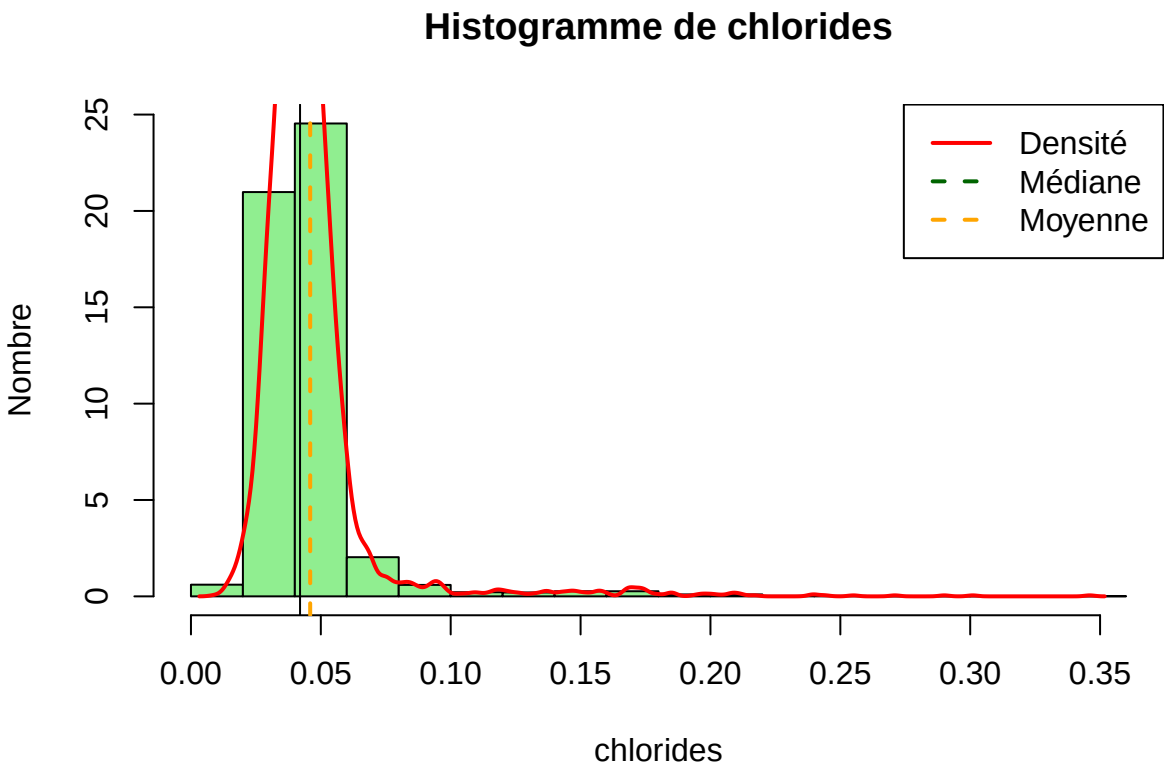


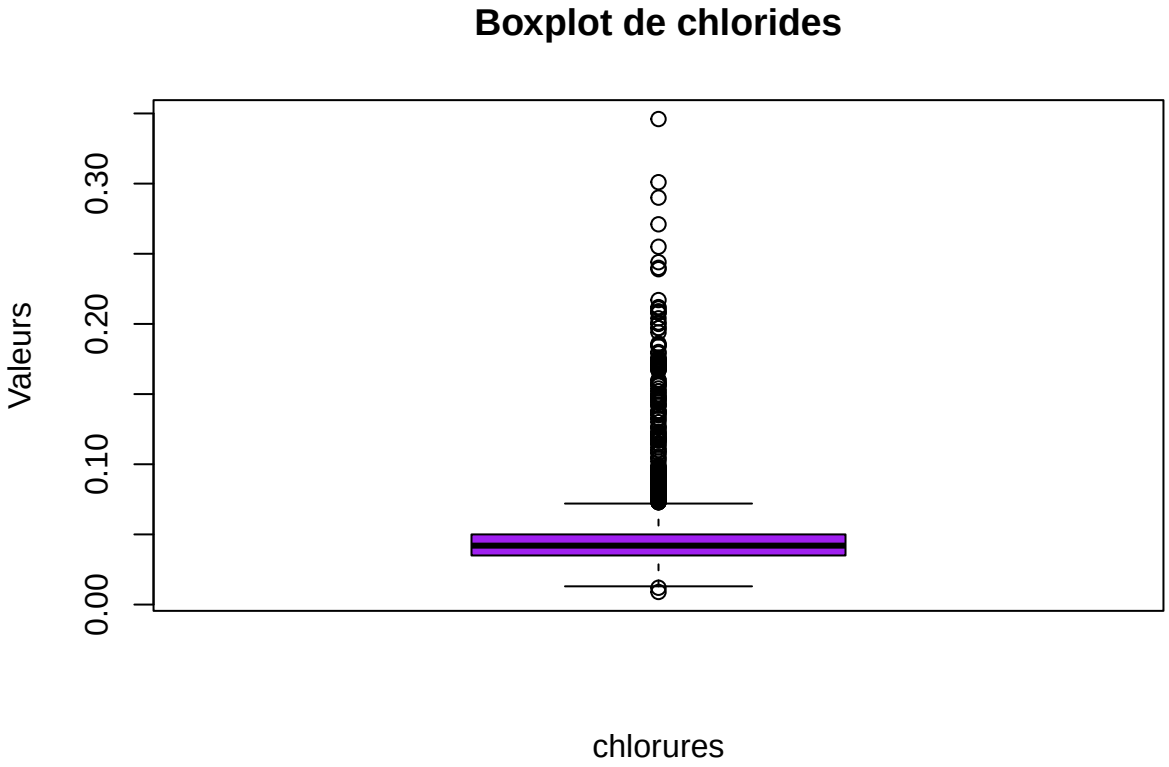
Analyse et interprétation :

L'histogramme de la distribution du sucre résiduel montre une distribution légèrement asymétrique à droite. Une grande majorité des vins ont une composition en sucre résiduel inférieure à 5.

La boîte à moustache de la variable montre qu'il y a des valeurs extrêmes dans la, particulièrement une d'elles qui s'écarte significativement, allant jusqu'au-delà de 60.

5.10.5 chlorides



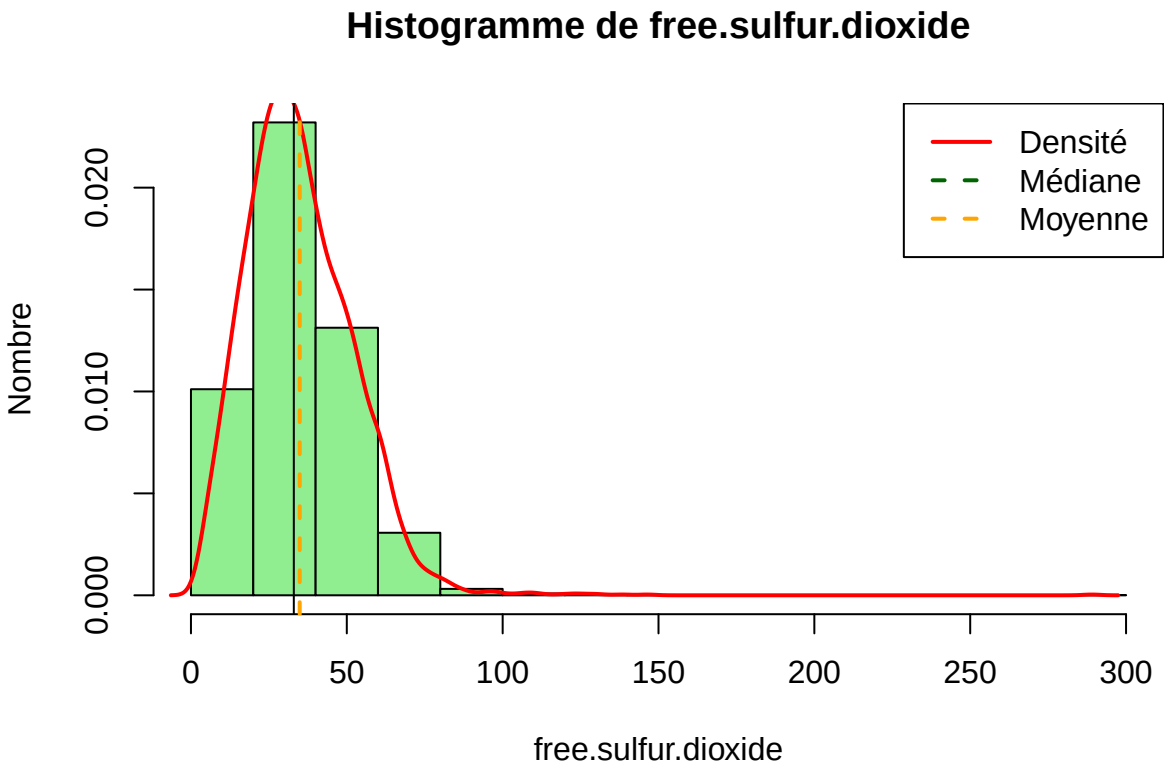


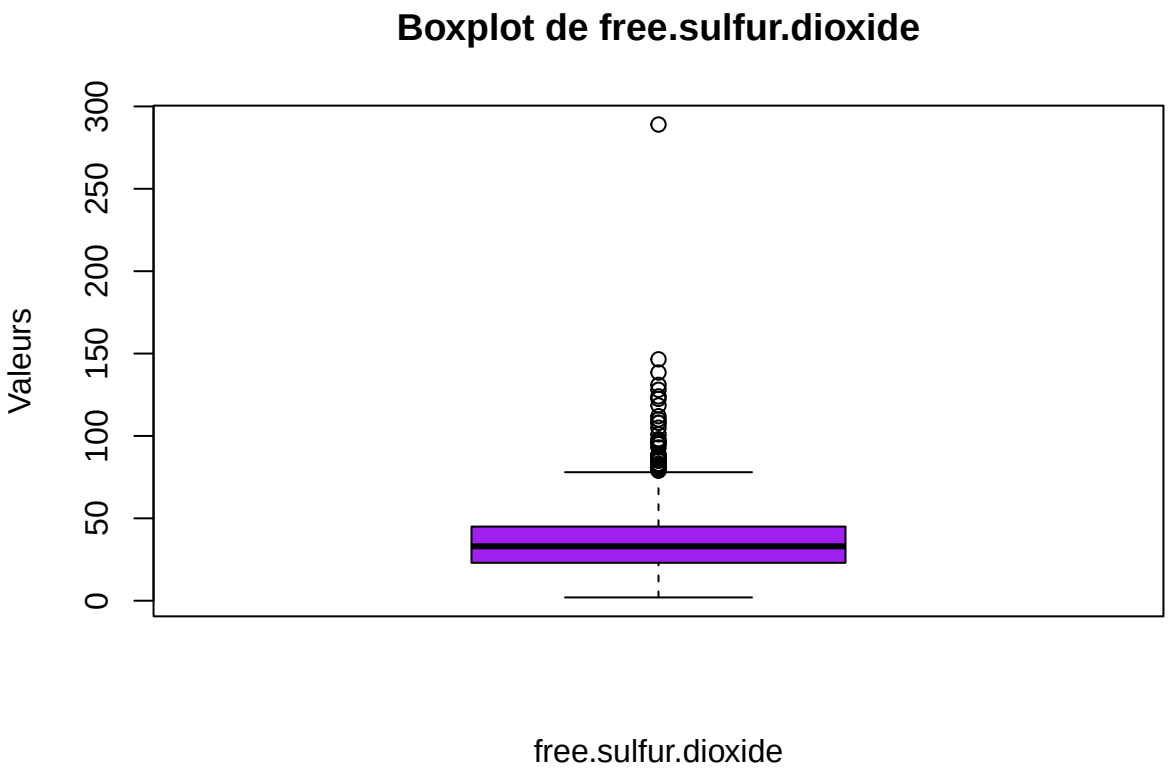
Analyse et interprétation :

L'histogramme de la variable **chlorides** montre une distribution asymétrique à droite.

Le boxplot montre la présence en grand nombre de valeurs extrêmes dans la variable. En plus, la boîte à moustaches est moins large, signifiant que les valeurs de la variable ont une variabilité très faible.

5.10.6 free.sulfur.dioxide



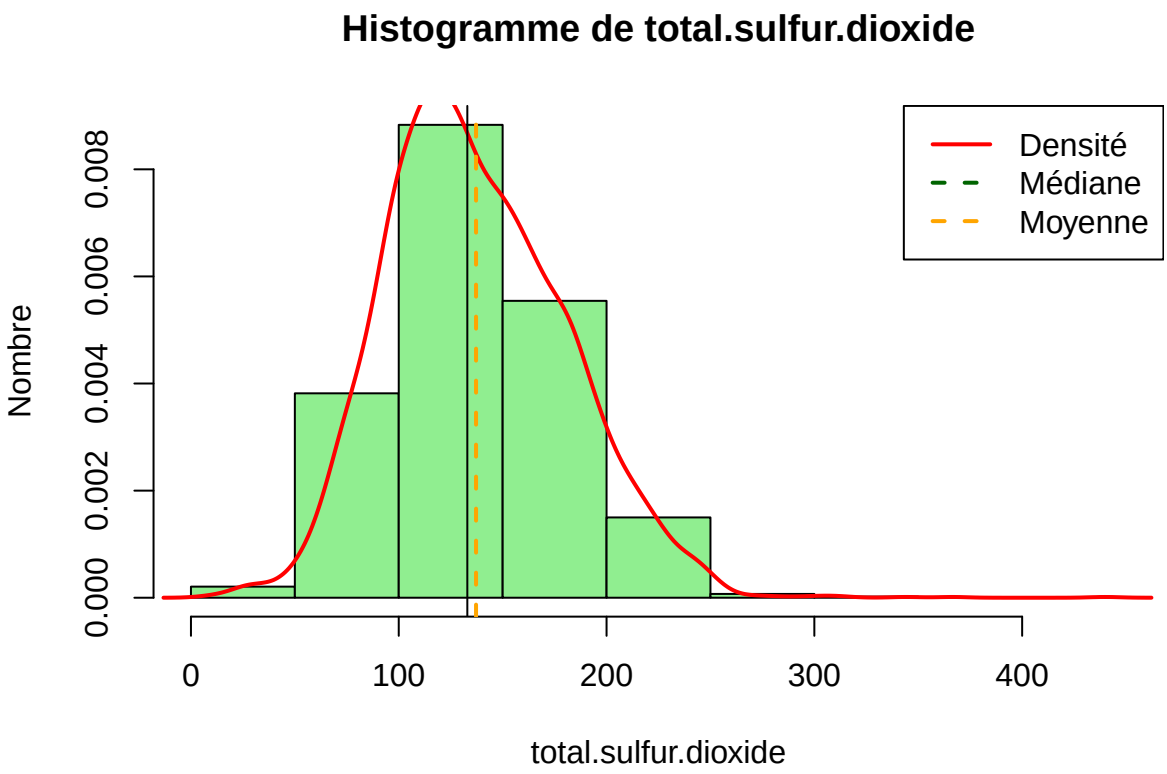


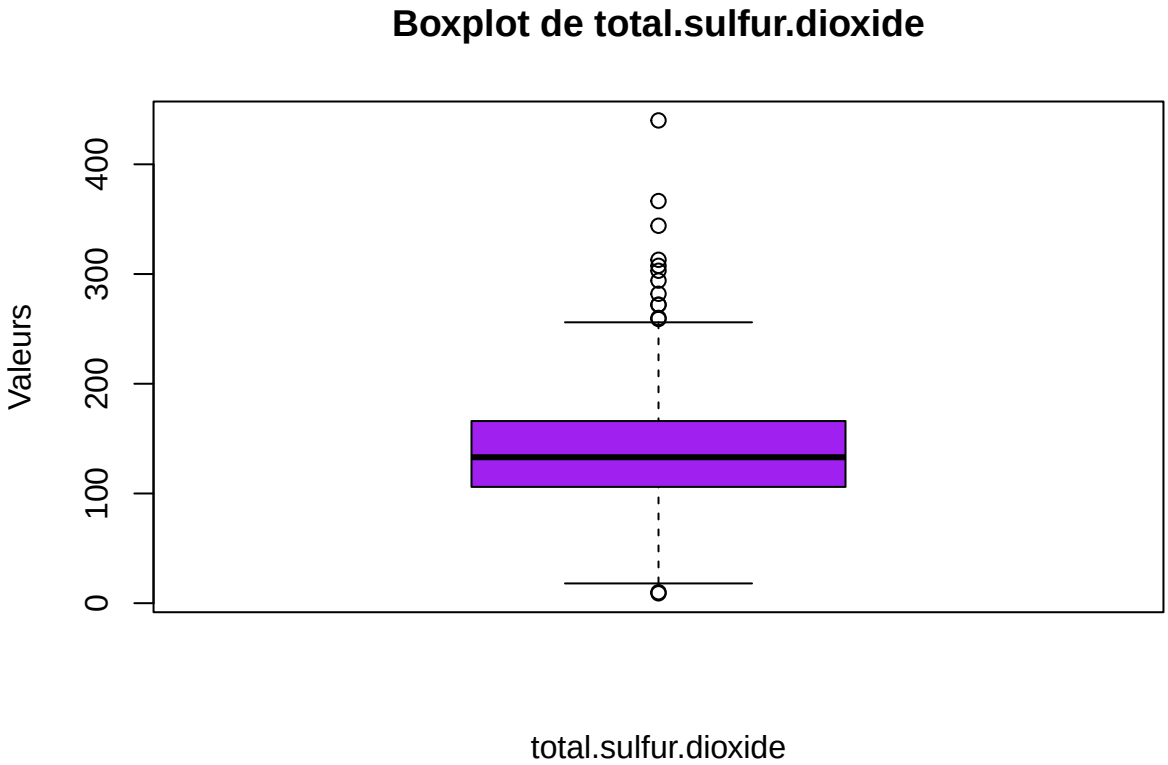
Analyse et interprétation :

L'histogramme de la variable **free.sulfur.dioxide** montre une distribution légèrement asymétrique à droite très proche d'une distribution normale. La majorité des vins ont une valeur en dioxide de soufre libre inférieure à 50.

Le boxplot montre que la variable contient plusieurs valeurs extrêmes. De plus, la variabilité des valeurs est faible puisque la boîte est moins large.

5.10.7 total.sulfur.dioxide





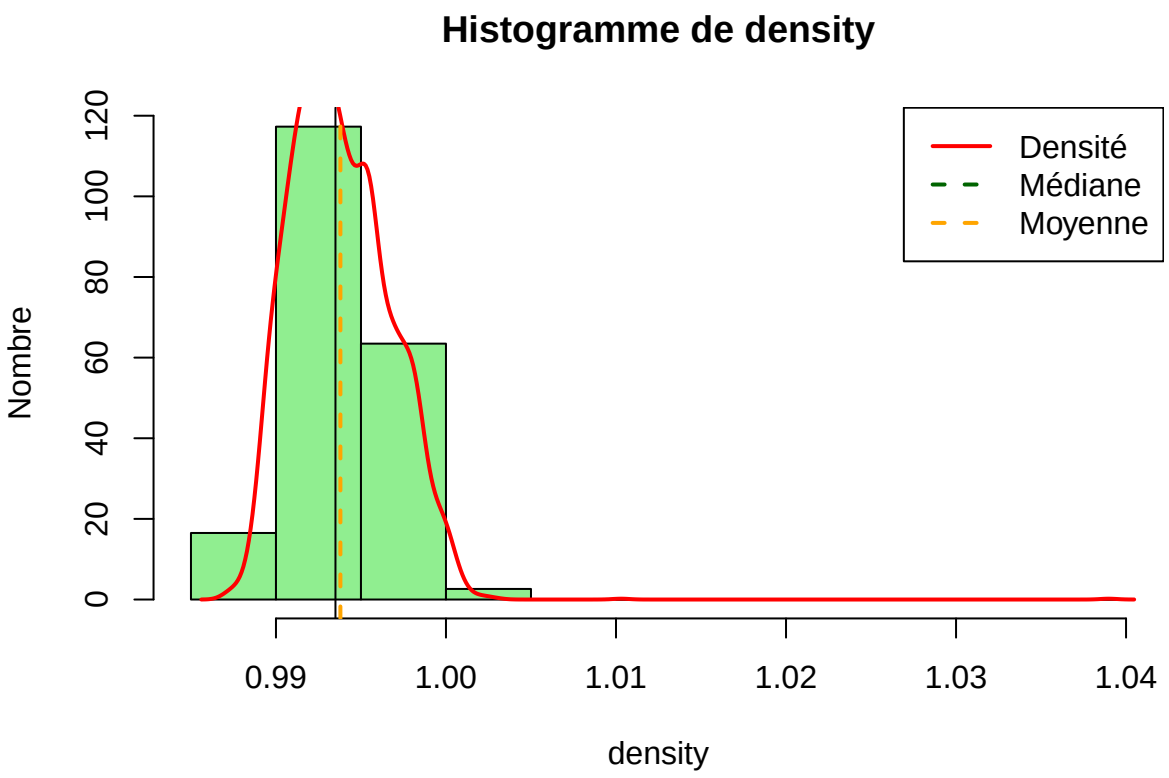
Analyse et interprétation :

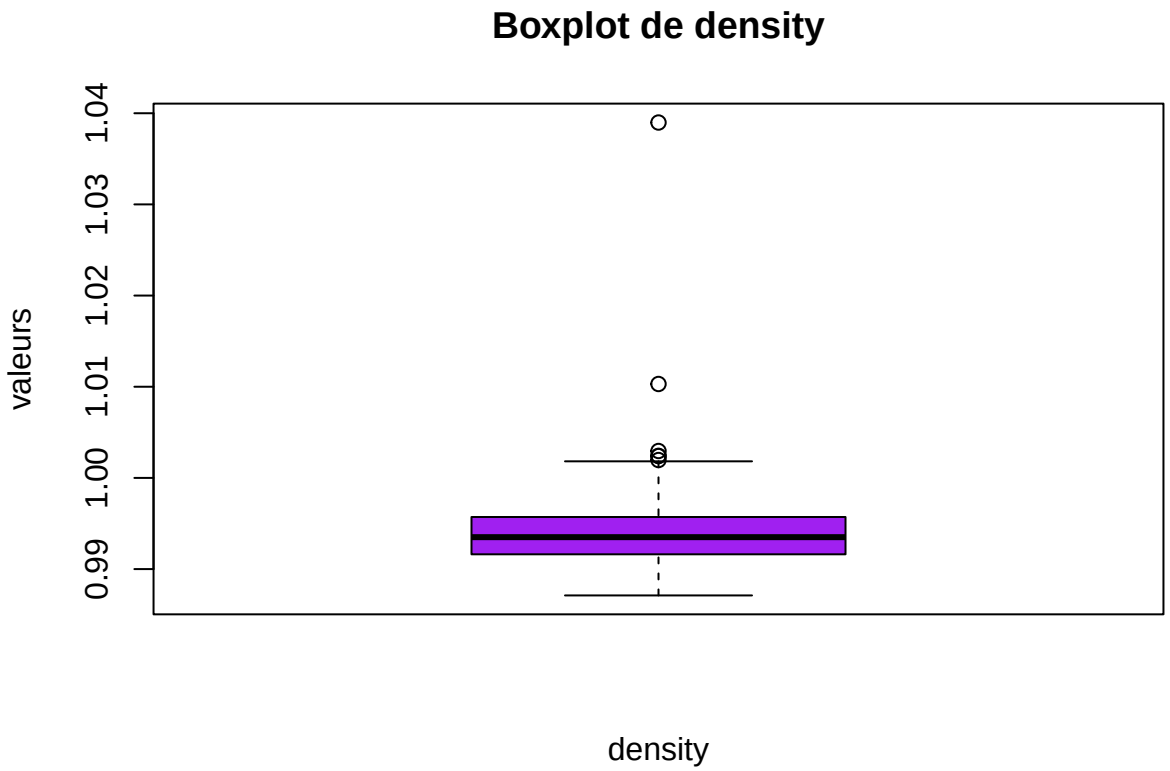
L'histogramme de la distribution du sulfure de dioxide total montre que la variable possède une distribution légèrement asymétrique à droite très proche d'une distribution normale. On obtient un pic de la courbe de normalité entre 100 et 150. Ainsi, la majorité des vins ont une valeur de sulfure de

dioxide total comprise entre 100 et 150.

La représentation de la boîte à moustache de la variable montre la présence de valeurs extrêmes dans la variable. Il y a des valeurs extrêmement grandes et une valeur extrêmement petite. Le boxplot montre également une faible variabilité des valeurs non extrêmes.

5.10.8 density



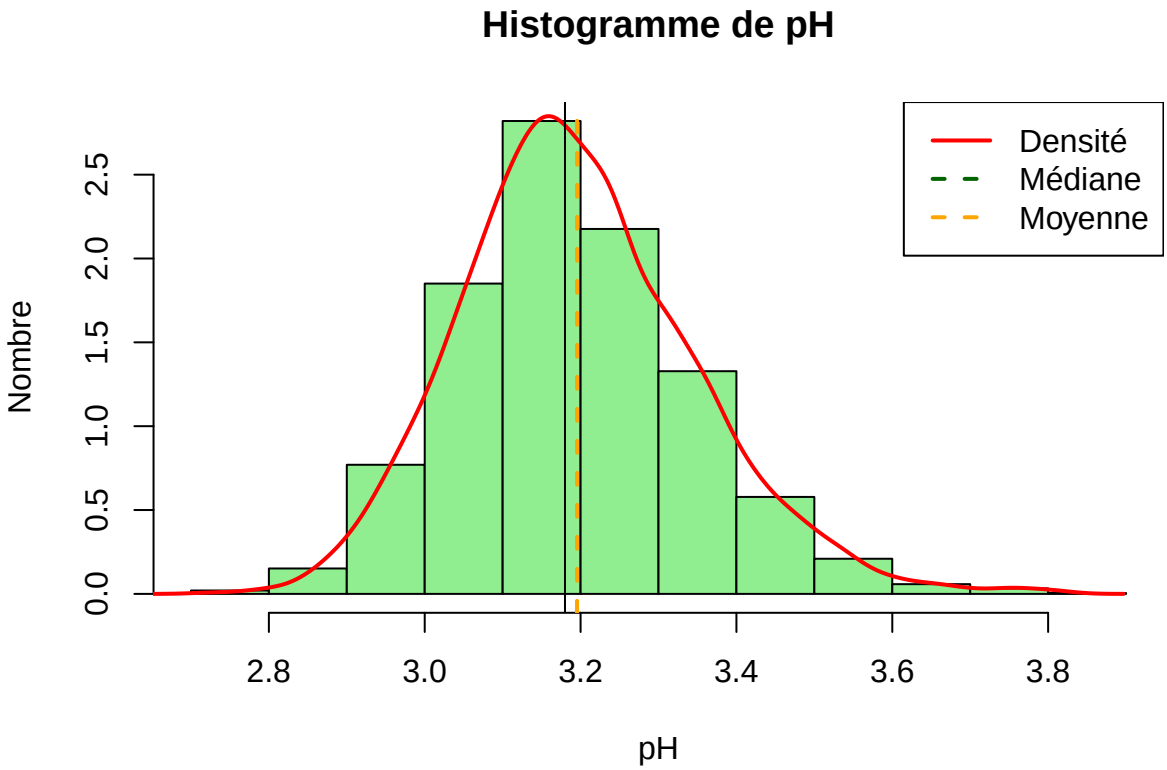


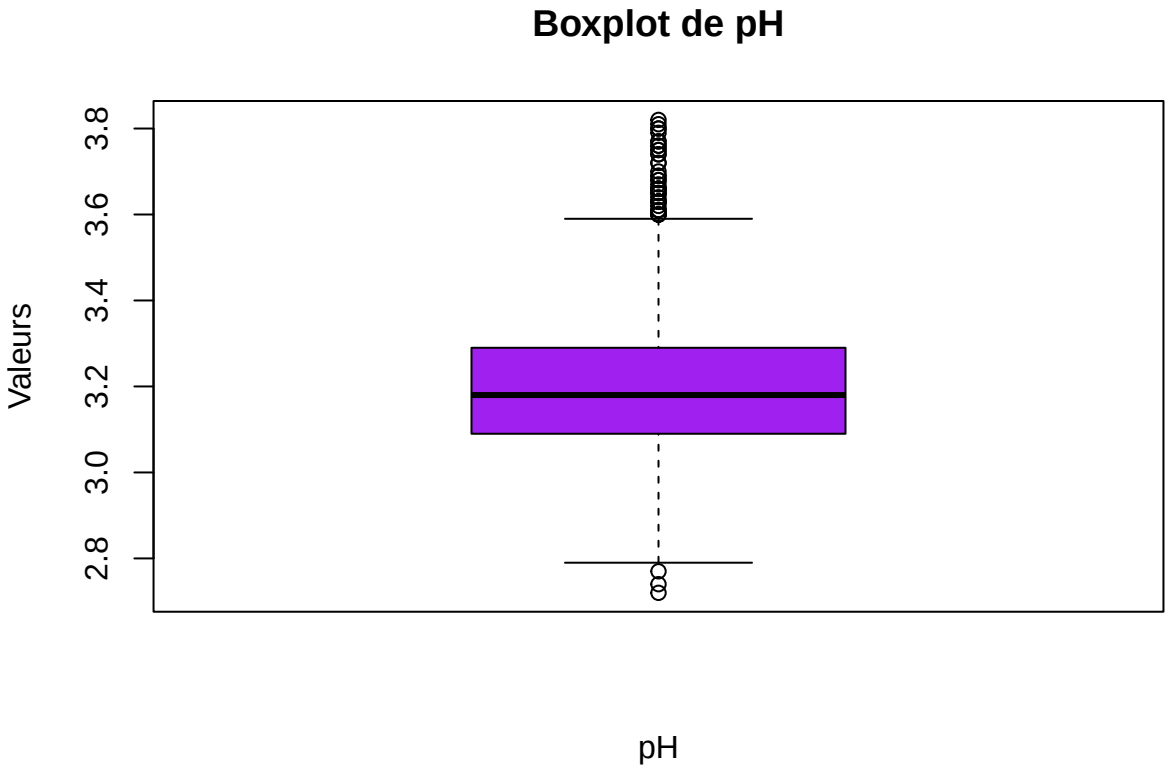
Analyse et interprétation :

La représentation graphique de l'histogramme de la variable **density** met en lumière une distribution légèrement asymétrique à droite très proche d'une distribution normale.

La représentation de la boîte à moustaches met en lumière la présence des valeurs aberrantes dans les données de la variable. Ces valeurs sont spécifiquement des données grandes. Il n'y a pas de données extrêmement petites. De plus, les valeurs non aberrantes ont une faible variabilité.

5.10.9 pH



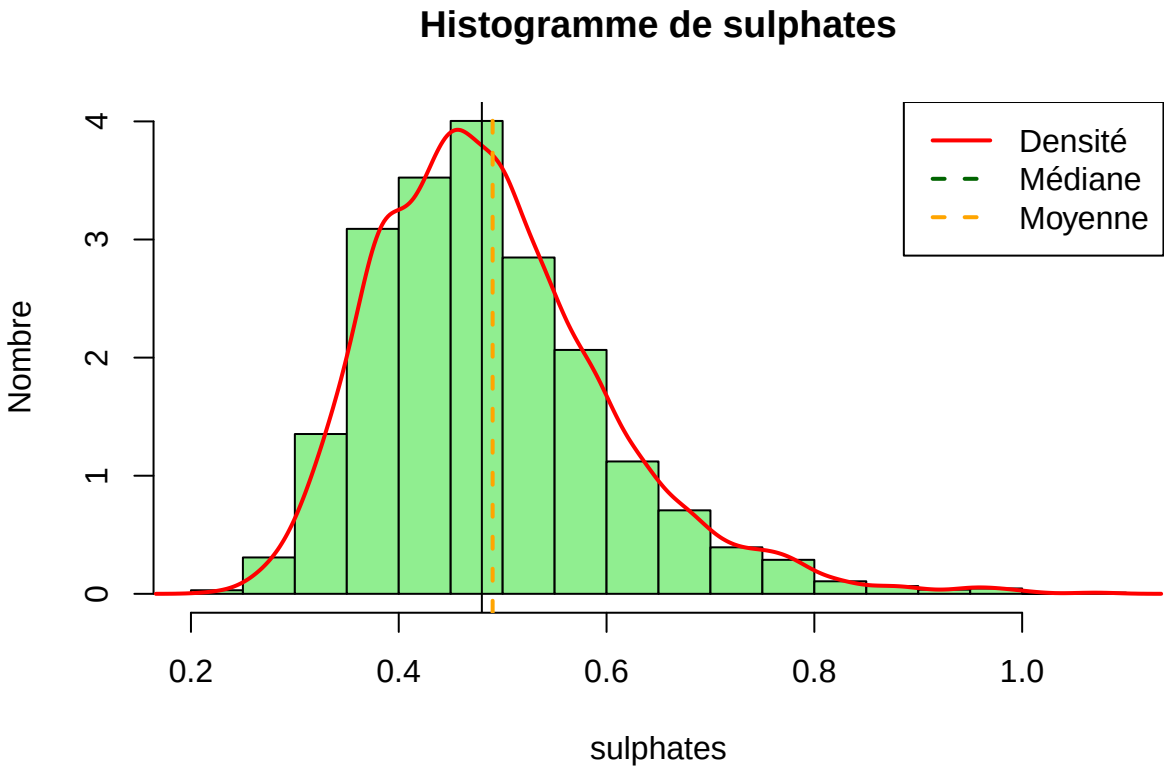


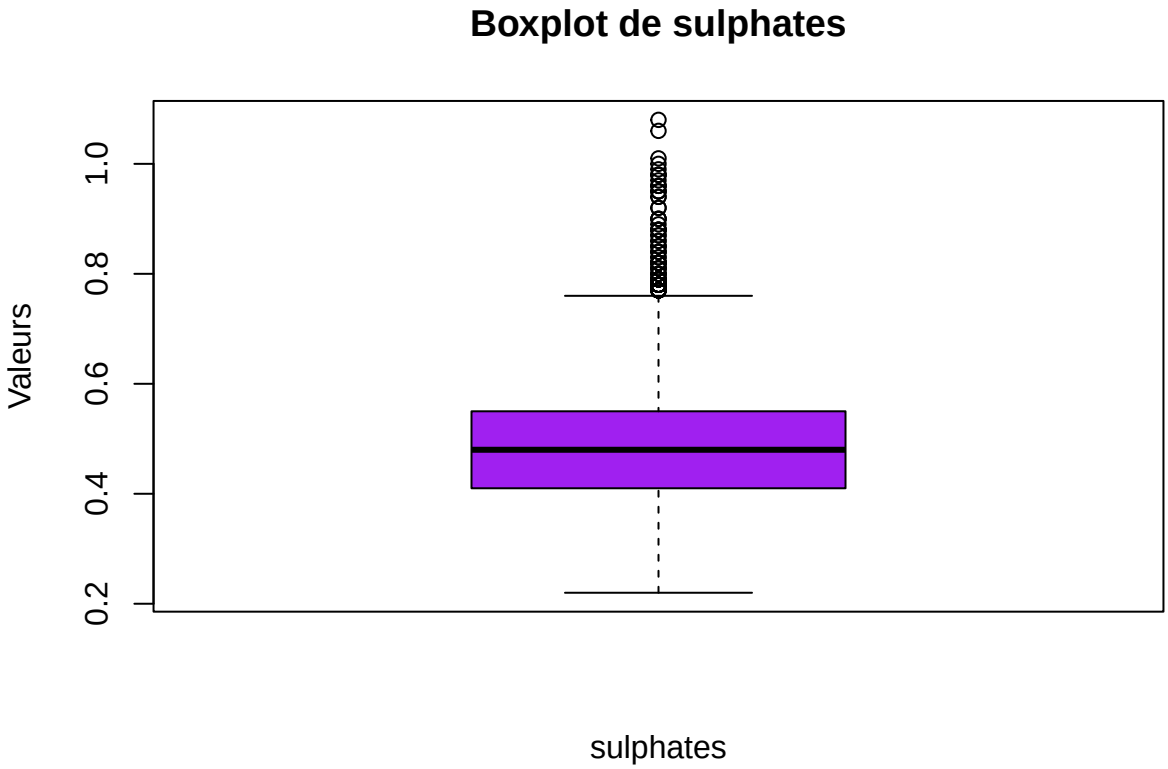
Analyse et interprétation :

L'histogramme du **pH** montre une distribution légèrement asymétrique à droite. La courbe de normalité ressemble à celle de la loi normale.

La représentation boîte à moustache montre la présence de valeurs aberrantes dans la distribution de la variable **pH** avec trois valeurs extrêmement faibles et quelques valeurs extrêmement grandes.

5.10.10 sulphates



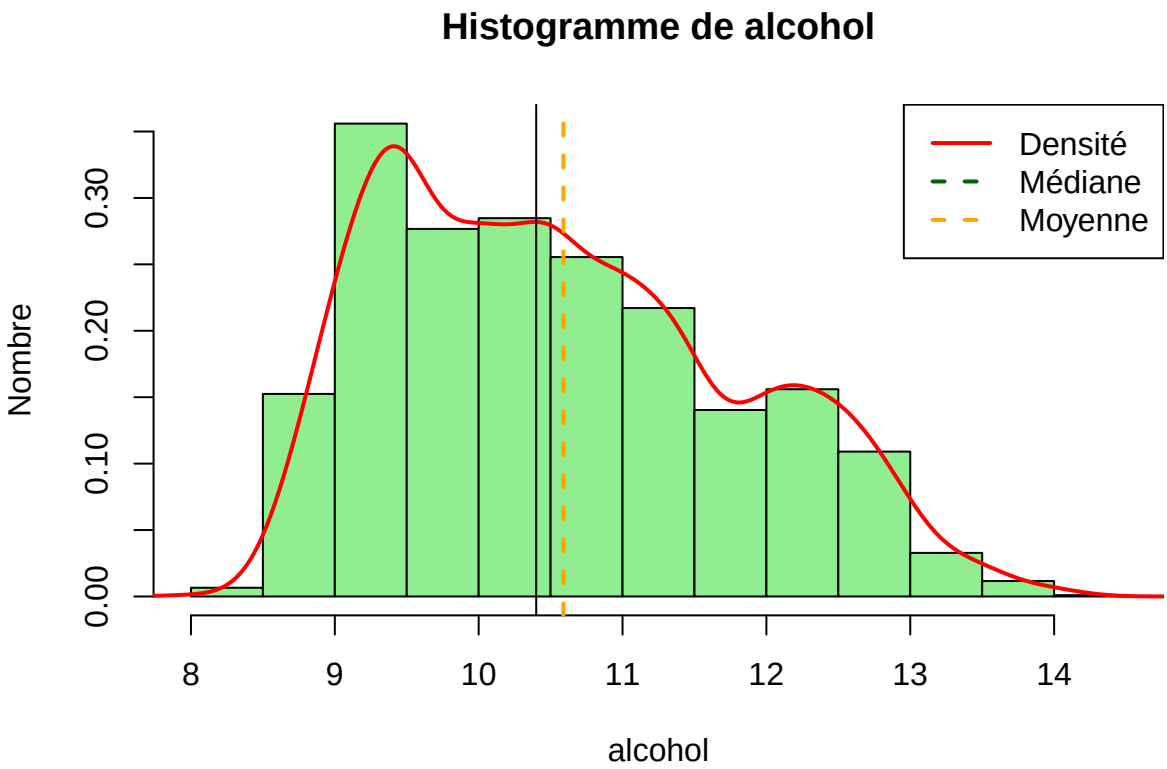


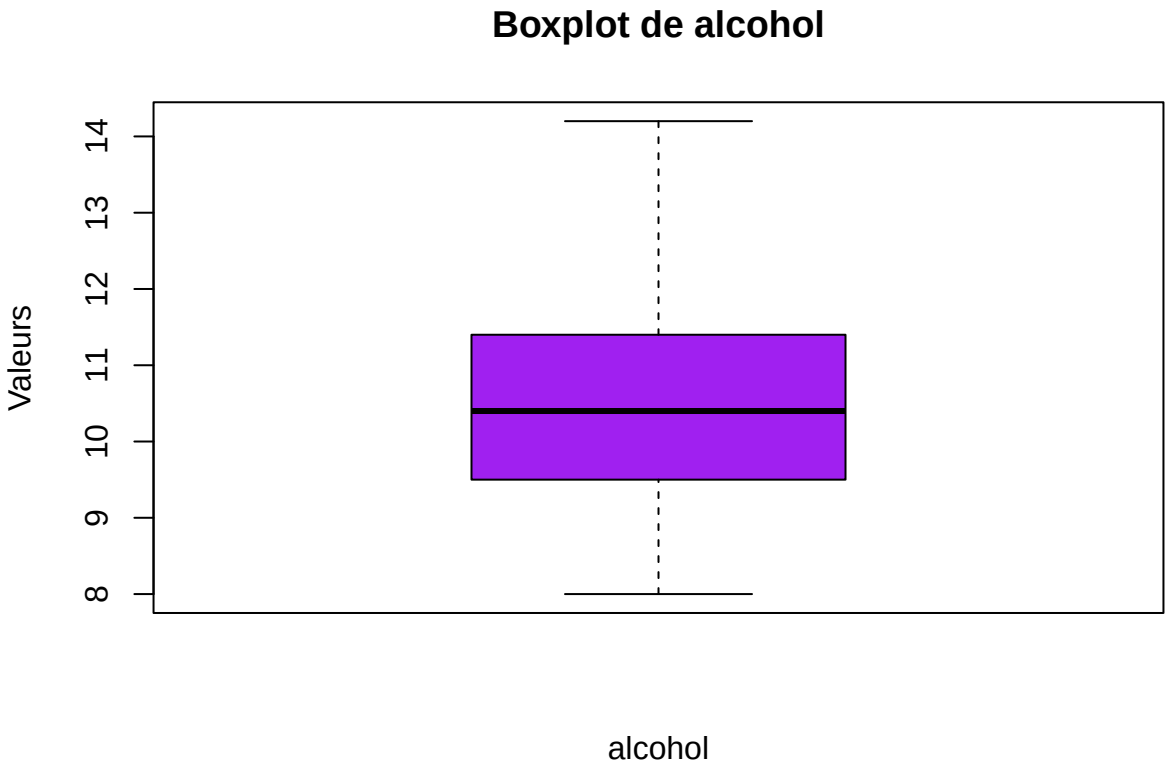
Analyse et interprétation :

L'histogramme de la variable **sulphates** montre une distribution asymétrique à droite.

Le boxplot de **sulphates** montre qu'il y a un grand nombre de valeurs aberantes dans la variable, des valeurs qui sont extrêmement grandes.

5.10.11 alcohol



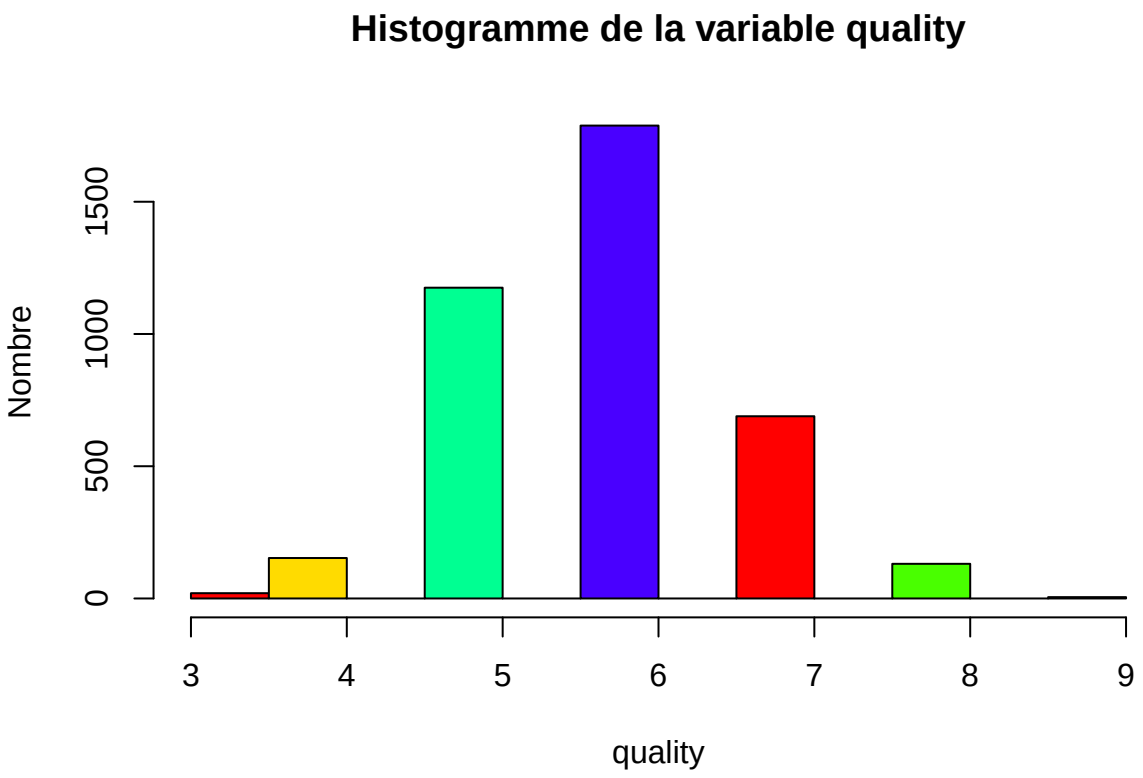


Analyse et interprétation :

L'histogramme de la variable **alcohol** montre une distribution irrégulière.

Le boxplot de la variable **alcohol** montre une distribution sans valeur aberrante. De plus, on observe une grande variabilité des valeurs de la variable puisque la boîte à moustache est large.

5.10.12 quality

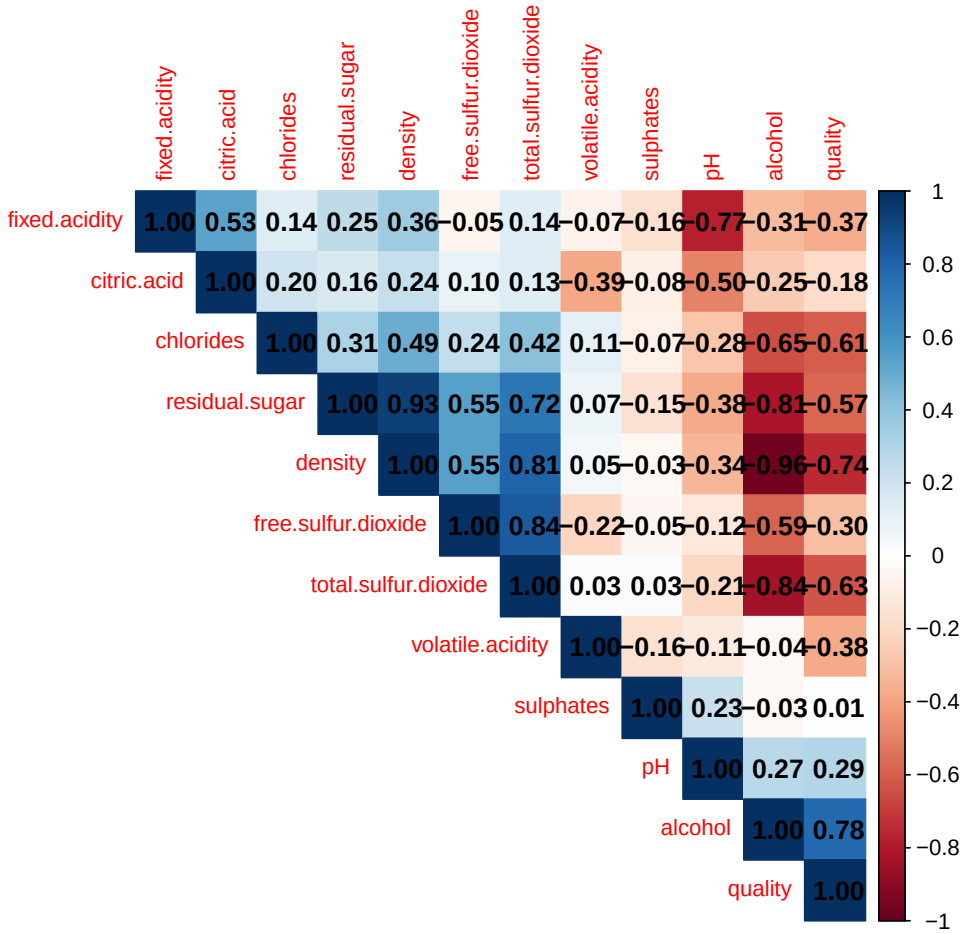


Analyse et interpréatation :

L'histogramme montre que la variable **quality** prend des valeurs discrètes. La plus haute barre se trouve autour de la valeur 6. Il n'y a pas de barre pour les valeurs inférieures à 3 et supérieures à 9. Il n'y a donc pas de vins dont la qualité est jugée inférieure à 3 ou supérieure à 9.

5.11 Analyses multivariées

Nous analysons les relations entre les différentes variables dans le jeu de données à travers la matrice de corrélation donnée ci-dessous :



Cette matrice met en lumière la présence ou non de corrélations négatives et positives entre les caractéristiques du jeu de données. Les variables fortement corrélées sont :

- **fixed.acidity** et **pH** ont une corrélation négative de -0,77 ;
- **chlorides** et **alcohol** ont une corrélation négative de -0,65 ;
- **chlorides** et **quality** ont une corrélation de -0,61 ;
- **residual.sugar** est corrélée de -0,81 avec **alcohol** ;
- **density** est corrélée avec **alcohol** de -0,96 ;
- **density** est corrélée avec **quality** de -0,74 ;
- **total.sulfur.dioxide** et **alcohol** sont corrélées de -0,84 ;
- **total.sulfur.dioxide** et **quality** sont corrélées de -0,63 ;
- **residual.sugar** et **density** ont une corrélation positive de 0,93 ;
- **residual.sugar** et **total.sulfur.dioxide** ont une corrélation positive de 0,72 ;
- **density** et **total.sulfur.dioxide** ont une corrélation positive de 0,81 ;
- **free.sulfur.dioxide** et **total.sulfur.dioxide** ont une corrélation positive de 0,84 ;
- **alcohol** est en corrélation positive avec **quality** de 0,78 .

6 Régression logistique

6.1 Sélection des caractéristiques pertinentes

Nous avons au total 11 variables quantitatives (caractéristiques physico-chimique des vins) dans la base de données normalisées. Il s'agit ici de voir quelles sont les variables pertinentes pour prédire la variable cible qui est la qualité du vin, **wine_quality**. Pour cela, nous réalisons des tests statistiques pour voir l'existence ou non de relations entre les variables quantitatives et la variable cible, et le test statistique convenable est celui de ***Kruskal-Wallis***.

Ci-dessous, la sortie des tests statistiques classée par ordre décroissant des valeur du test :

```
##                                       Variable Kruskal_Wallis      P_value
## Kruskal-Wallis chi-squared5   free.sulfur.dioxide      91.961751 8.836806e-22
## Kruskal-Wallis chi-squared1      volatile.acidity      68.918280 1.026291e-16
## Kruskal-Wallis chi-squared3        residual.sugar      16.301844 5.401135e-05
## Kruskal-Wallis chi-squared10             alcohol      14.624145 1.312225e-04
## Kruskal-Wallis chi-squared          fixed.acidity      12.739830 3.579502e-04
## Kruskal-Wallis chi-squared4             chlorides      12.346809 4.417410e-04
## Kruskal-Wallis chi-squared6  total.sulfur.dioxide       8.247726 4.080309e-03
## Kruskal-Wallis chi-squared7               density       7.031047 8.010849e-03
## Kruskal-Wallis chi-squared2           citric.acid       6.300684 1.206914e-02
## Kruskal-Wallis chi-squared9             sulphates       3.073866 7.956006e-02
## Kruskal-Wallis chi-squared8                    pH       1.761868 1.843917e-01
```

Nous remarquons que les p-valeurs sont plus petites que 0,05, sauf pour les variables **sulfates** et **pH.** Seules ces deux dernières variables sont non pertinentes pour prédire la qualité du vin (*wine_quality*).

6.2 Hypothèses

La régression logistique se base également sur des hypothèses statistiques que nous vérifions dans les lignes qui suivent :

6.2.1 Variable binaire

Dans la régression logistique, on souhaite prédire une probabilité. Pour cela, la variable à expliquer se doit d'être binaire. Dans ce projet, nous considérons **0/1** pour la variable cible **wine_quality** .

6.2.2 Absence de multicolinéarité

Dans un modèle de régression logistique, il ne doit pas y avoir de forte corrélation entre les variables explicatives. La multicolinéarité réduit les performance du modèle qui peut avoir de faux résultats car plusieurs variables expliquent en ce moment le même phénomène. Pour le jeu de données qui fait objet de notre étude, il y a de fortes corrélations entre certaines variables d'après la matrice de corrélation réalisée dans la section précedente. Pour rémedier à ce problème, plusieurs option sont disponibles:

- Il est possible de sélectionner dans chaque paire de variables fortement corrélées celle qui est plus corrélée avec la variable cible ;
- il est Il est également possible de réaliser une **An**alyse en **C**omposantes **P**rincipales (ACP) pour réduire la dimensionnalité et avoir des composantes orthogonales (non linéaires) à partir des variables originales.

Nous choisirons la deuxième option qui est de réaliser une ACP sur les données d'entraînement du modèle que nous créerons dans la suite.

6.2.3 Indépendance des observations

Cette hypothèse réclame qu'il y ait une indépendance entre les observations. La dépendance des observations est souvent plus récurrente dans le cas des séries temporelles, ce qui n'est pas le cas pour les données que nous étudions. Donc l'hypothèse est vérifiée.

6.2.4 Absence de valeurs aberrantes

Les valeurs extrêmes peuvent souvent avoir trop d'impact sur le modèle.

6.2.5 Présence de suffisament d'observation

Le modèle de régression logistique a besoin d'un ensemble de données suffisamment grand pour un bon fonctionnement du modèle. Pour le jeu de données soumis à notre étude, nous estimons qu'il y a suffisamment d'observations.

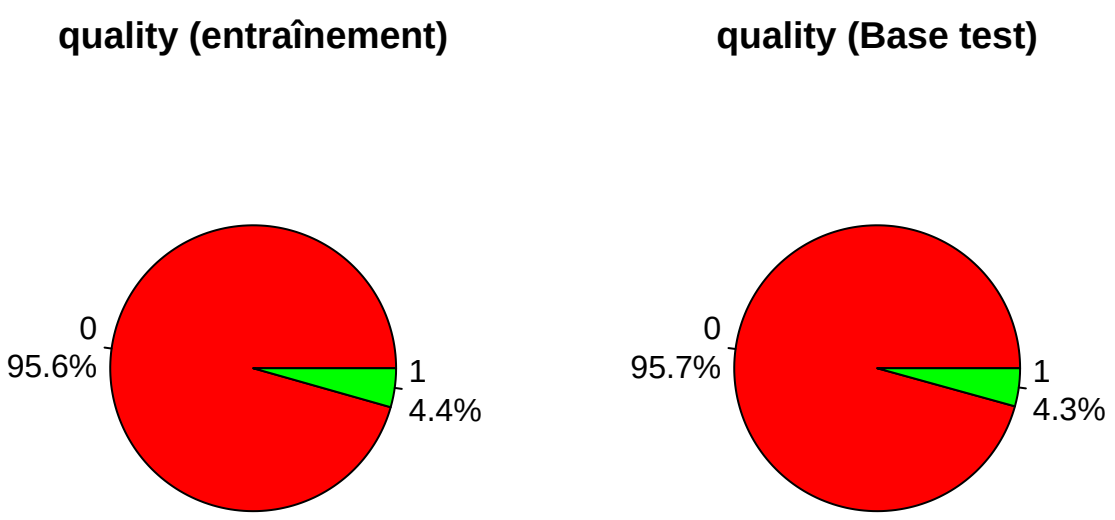
6.3 Partitionnement de l'ensemble de données

Nous divisons l'ensemble de données soumis à notre étude en deux sous-ensembles. Le premier est une base d'entraînement du modèle constituée de 70 % des observations des données initiales, et le deuxième est une base test pour le modèle (30 % des données initiales). Nous réalisons particulièrement un partitionnement stratifié afin d'avoir la même représentativité des classes dans les deux bases.

Ci déssous, le code R permettant de réaliser le partitionnement :

```r
# Définition de la proportion de données à garder dans l'ensemble d'entraînement
proportion_entrainement <- 0.7

# Création d'indices pour un partitionnement stratifié
set.seed(123) # Pour la reproductibilité
indices_entrainement <- createDataPartition(df_norm$wine_quality,
                                            p = proportion_entrainement,
                                            list = FALSE)

# Création des ensembles d'entraînement et de test
data_entrainement <- df_norm[indices_entrainement, ]
data_test <- df_norm[-indices_entrainement, ]
```

Diagrammes circulaire de la répartition de la qualité du vin dans les deux bases



Dimensions des bases:

```
## [1] 2774   12
```

```
## [1] 1187   12
```

L'affichage des dimensions des nouvelles bases donne 2774 observations pour la base d'entrainement et 1187 observations pour la base test.

6.4 Réalisation de l'ACP

Nous réalisons une ACP uniquement sur les données d'entraînement afin d'avoir un modèle plus ou moins performant.

Ci-dessous le code R:

```
var.keep <- c("fixed.acidity", "volatile.acidity", "citric.acid", "residual.sugar", "chlorides",
              "free.sulfur.dioxide","total.sulfur.dioxide","density",
              "pH","sulphates","alcohol")

pca=PCA(data_entrainement[,var.keep], scale.unit=TRUE, graph=F);

vp=pca$eig;

seuil=80;
nb=max(which(vp[,3]< seuil));

pca=PCA(df_norm[,var.keep],ncp=nb, scale.unit=TRUE, graph=FALSE);
pca_df=pca$ind$coord;

nb
```

```
## [1] 5
```

L'acp donne 5 composantes principales.

6.5 Ré-échantillonnage

```
##
##    0    1
## 3788  173
```

Dans la base d'entrainement, nous remarquons la présence d'un déséquilibre significatif entre les classes. 3788 observations dans la classe "0" (mauvais vins) contre 173 pour la classe "1" (bons vins). Cela peut affecter la performance du modèle et, pour résoudre ce problème, nous réalisons un ré-échantillonnage en utilisant la méthode **SMOTE** qui crée des échantillons synthétiques pour augmenter la classe minoritaire qui est celle des vins de bonne qualité. Nous l'appliquons sur la base d'entrainement, puis transformons la base test afin d'éviter les fuites de données.

```
# Création de la recette avec SMOTE appliqué au données d'entrainement
recette_smote <- recipe(wine_quality~.,data = df_norm)%>%
  step_pca(all_predictors(),num_comp = 5) %>%
  step_smote(wine_quality,over_ratio = 0.8)

# Entrainement de la recette
recette_prep <- prep(recette_smote)

# Application de la recette
data_entrainement_smote <- bake(recette_prep,new_data=NULL)
```

```r
# Application au testset (sans SMOTE, mais avec le même PCA)
data_test_prepare <- bake(recette_prep,new_data = data_test)
```

Aperçu des nouveaux données

```
## # A tibble: 6 x 6
##      PC1     PC2     PC3     PC4     PC5 wine_quality
##    <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <fct>
## 1 -3.86   0.511   0.906   1.19    0.161  0
## 2  0.512 -0.436   0.346  -1.06   -0.404  0
## 3 -0.284  1.17    0.116  -0.241  -0.105  0
## 4 -1.58  -0.0628 -0.0804  0.530  -0.805  0
## 5 -0.215 -0.886   1.28    0.0770 -0.250  0
## 6  0.456  1.36   -0.804  -0.150  -0.0831 0
```

```
## # A tibble: 6 x 6
##   wine_quality    PC1     PC2     PC3     PC4     PC5
##   <fct>         <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 0            -0.284  1.17    0.116  -0.241  -0.105
## 2 0             0.456  1.36   -0.804  -0.150  -0.0831
## 3 0             1.61   0.0193 -1.32   -0.0921  0.0797
## 4 0             2.84  -0.0392  1.80   -0.141   1.02
## 5 0             0.408 -0.805  -0.614  -0.486  -0.839
## 6 0            -1.36   1.39   -0.0772 -0.250   0.852
```

```r
# Vérification des dimention de la distribution
table(data_entrainement$wine_quality)  # Avant smote
```

```
##
##    0    1
## 2652  122
```

```r
table(data_entrainement_smote$wine_quality)  # Après smote
```

```
##
##    0    1
## 3788 3030
```

Le ré-échantillonnage donne 3788 observations pour la classe des vins de mauvaise qualité et 3030 observations des vins de bonne qualité.

Dimension de la nouvelle base d'entrainement :

```r
# Dimension du nouveau jeu de données d'entrainement
dim(data_entrainement_smote)
```

```
## [1] 6818    6
```

Le nouveau jeu de données d'entrainement est composé de 6818 observations et 6 variables.

6.6 Entrainement du modèle de régression logistique

Nous entrainons le modèle de régression logistique sur les données d'entrainement avec la fonction **glm()**

```r
# Entraînement du modèle de régression logistique avec glm
modele_logistique <- glm(wine_quality ~ ., data = data_entrainement_smote, family = binomial)
```

6.7 Analyse, interprétation et évaluation du modèle

6.7.1 Résumé statistique

Le résumé statistique du modèle de régression logistique donne une synthèse complète des résultats obtenu

Ci-dessous, le résumé du modèle :

```r
# Afficher le résumé du modèle
summary(modele_logistique)
```

```
##
## Call:
## glm(formula = wine_quality ~ ., family = binomial, data = data_entrainement_smote)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.58246    0.02924 -19.921  < 2e-16 ***
## PC1         -0.05566    0.01546  -3.601 0.000317 ***
## PC2          0.22571    0.01936  11.660  < 2e-16 ***
## PC3          0.57245    0.02296  24.930  < 2e-16 ***
## PC4         -0.42522    0.03029 -14.037  < 2e-16 ***
## PC5          0.09951    0.02967   3.354 0.000795 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9367.3  on 6817  degrees of freedom
## Residual deviance: 8112.1  on 6812  degrees of freedom
## AIC: 8124.1
##
## Number of Fisher Scoring iterations: 4
```

Analyse et interprétation :

Les 5 composantes ont toutes des p-valeurs inférieures à 0,05. Les coefficients de ces composantes sont donc significatifs dans la prédiction de la variable **wine_quality** . Par exemple, la composante une (PC1) a un coefficient négatif, cela signifie que cette composante contribue à diminuer la valeur de la probabilité, contrairement aux composantes 2, 3 ou 5 qui ont des coefficients de signe positif.

6.7.2 Significativité globale du modèle

Evaluons la significativité globale du modèle à travers le test du **rapport de vraisemblance**.

```
## Likelihood ratio test
##
## Model 1: wine_quality ~ PC1 + PC2 + PC3 + PC4 + PC5
## Model 2: wine_quality ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   6 -4056.0
## 2   1 -4683.7 -5 1255.2  < 2.2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La p-valeur est inférieure à 0.05, donc le modèle dans son ensemble est statistiquement significatif.

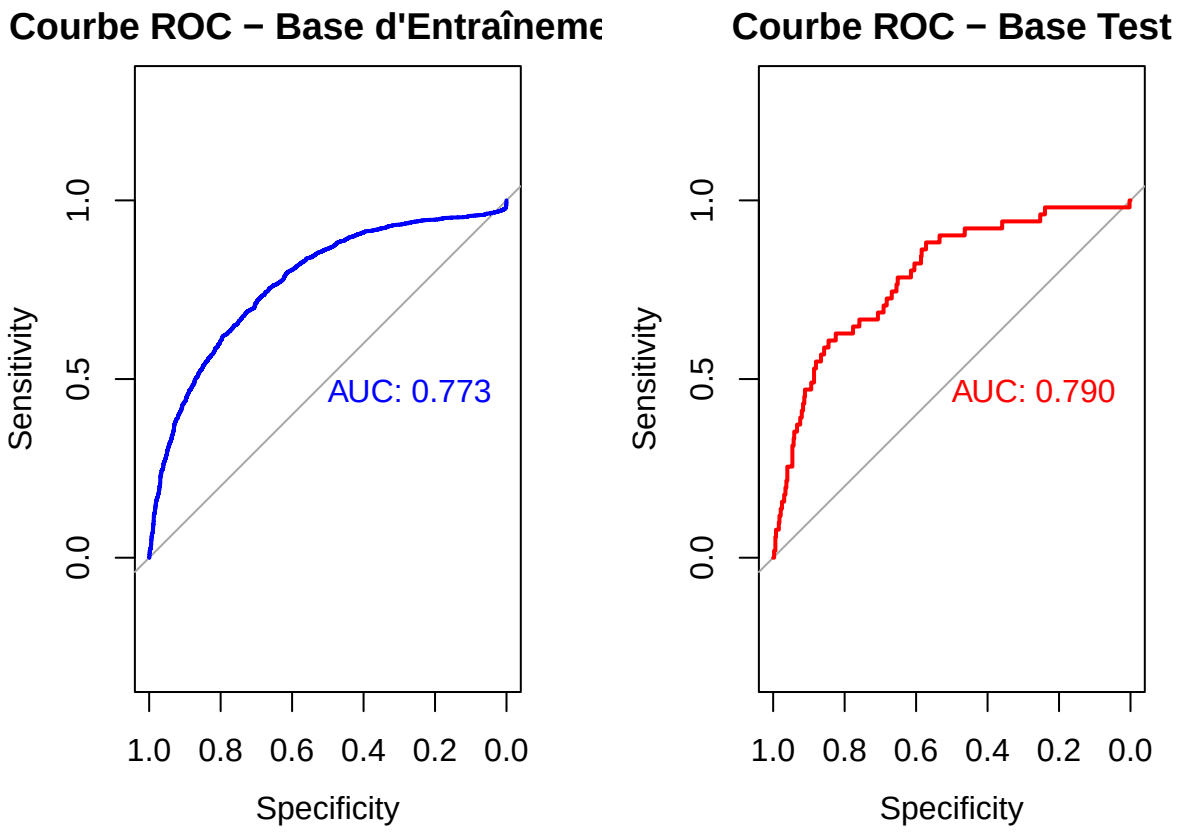
6.7.3 Pouvoir explicatif du modèle

```
## [1] 0.1340001
```

Le coefficient R^2 de McFadden permet d'évaluer le pouvoir explicatif du modèle de régression logistique. Ce modèle de régression logistique donne un coefficient qui est 0.134. Celà montre que le modèle a un faible pouvoir explicatif.

6.7.4 Evaluation des performances du modèle

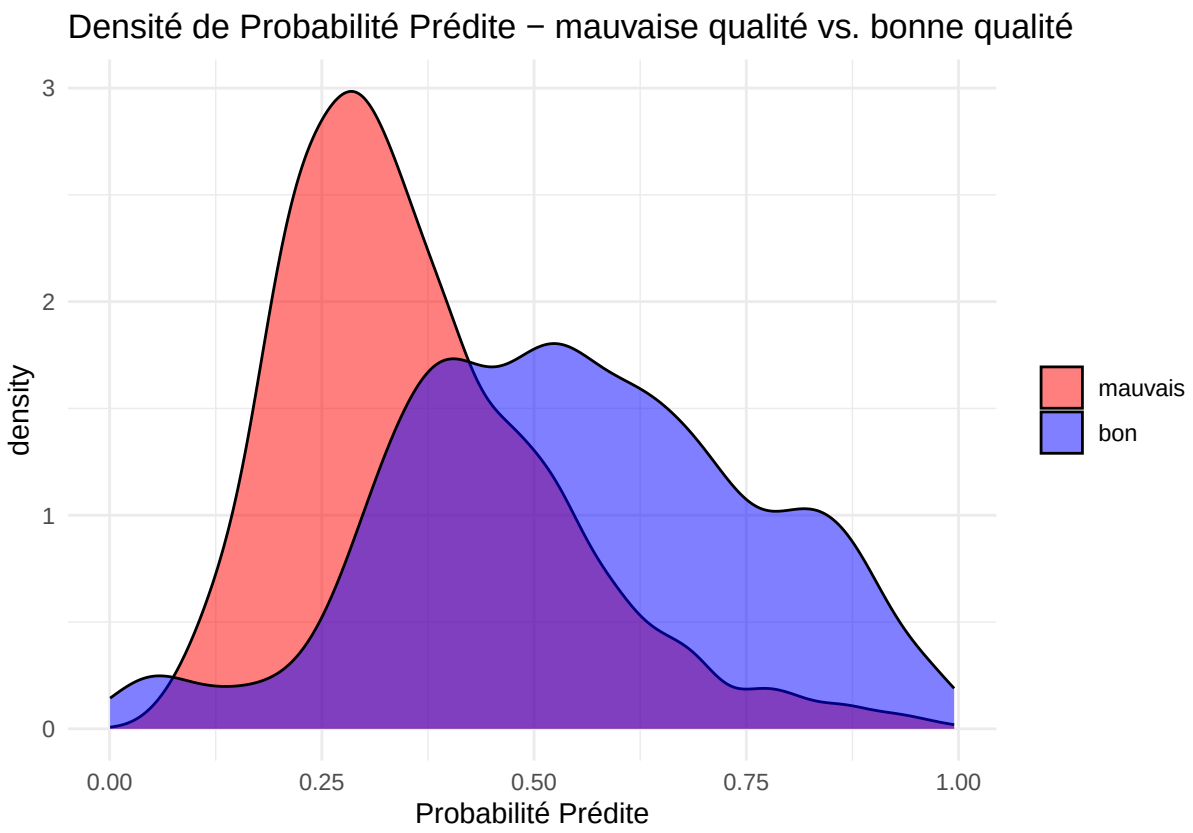
La représentation de la courbe de ROC pour les deux bases de données permet d'évaluer la performance du modèle de régression logistique.



L'aire sous la courbe de ROC(AUC) est de 0,773 dans la base d'entraînement et 0,79 dans la base test. On peut dire que le modèle apprend bien. En effet, dans la base d'entraînement, sur 10 observations, le modèle arrive à bien classer 7 d'entre eux. Dans la base test, il arrive à bien classer 8 d'entre eux. De plus, la legère augmentation de l'AUC dans la base test montre qu'il n'y a pas du surapprentissage du modèle.

6.7.5 Prédiction

Il s'agit de prédire l'appartenance ou non à une classe à partir de la probabilité. Pour cela, nous construisons d'abord les courbes de normalité des deux classes afin de pouvoir trouver un seuil de prédiction.



Choisissons un seuil de 0.40 pour la prédiction.

```
# Prédire les classes en utilisant un seuil de probabilité de 0.4 pour la base d'entraînement
seuil <- 0.4
predictions_train <- ifelse(probas_train >= seuil, 1, 0)
predictions_train <- factor(predictions_train, levels = c(0, 1))

# Créer la matrice de confusion pour la base d'entraînement
confusion_matrix_train <- confusionMatrix(predictions_train, data_entrainement_smote$wine_quality)

# Prédire les classes en utilisant un seuil de probabilité de 0.5 pour la base de test
predictions_test <- ifelse(probas_test >= seuil, 1, 0)
predictions_test <- factor(predictions_test, levels = c(0, 1))
# Créer la matrice de confusion pour la base de test
confusion_matrix_test <- confusionMatrix(predictions_test, data_test_prepare$wine_quality)
```

6.7.6 Matrice de confusion

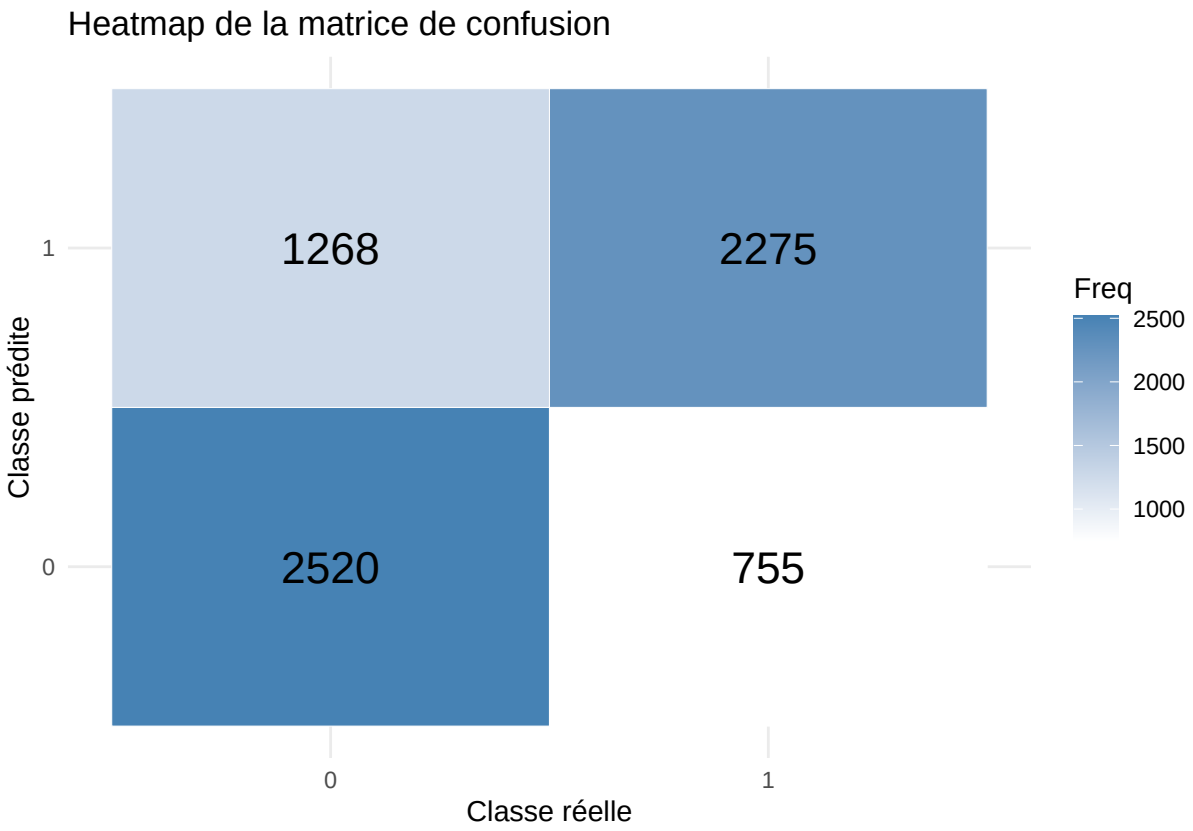
La matrice de confusion du modèle permet d'avoir des information importantes sur celui-ci telles que la sensibilité, la précision, la spécificité, le F1 Score et bien d'autres métriques. Ces metriques permettent d'évaluer le modèle et le comparer avec d'autres modèles.

Matrice de confusion de la base d'entrainement :

```
#confusion_matrix_train
confusion_matrix_train
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    0    1
##          0 2520  755
##          1 1268 2275
##
##                Accuracy : 0.7033
##                  95% CI : (0.6923, 0.7141)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4092
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6653
##             Specificity : 0.7508
##          Pos Pred Value : 0.7695
##          Neg Pred Value : 0.6421
##              Prevalence : 0.5556
##          Detection Rate : 0.3696
##    Detection Prevalence : 0.4803
##       Balanced Accuracy : 0.7080
##
##        'Positive' Class : 0
##
```



Analyse :

La matrice de confusion sur la base d'entrainement donne plusieurs informations qui permettent d'evaluer le niveau d'apprentissage du modèle. La représentation du heatmap donne:

- 2520 vrais positifs: le modèle classe 2520 vins comme de mauvaises qualités sachant qu'ils sont de

mauvaises qualité

- 2275 vins sont classés comme étant de bonnes qualités sachant qu'ils sont réellement de mauvaises qualités
- le modèle classe 1268 vins de mauvaises qualités comme étant de bonnes qualités
- le modèle classe 755 vins de bonnes qualités comme étant de mauvaises qualités

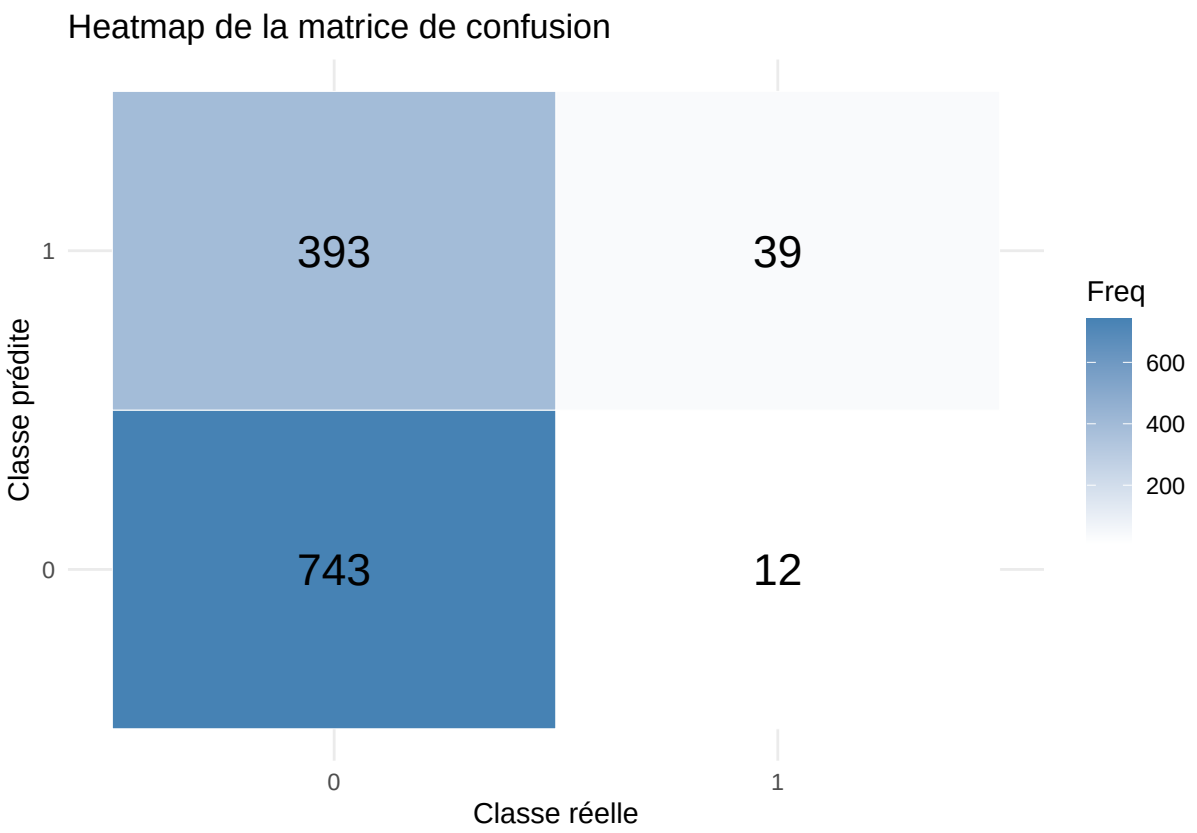
En général, les vins correctement classés sont nettement supérieur à ceux mal classé par le modèle.

Matrice de confusion de la base test

La matrice de confusion pour la base test permet d'évaluer la capacité de généralisation du modèle et de detecter éventuellement la présence de surapprentissage

```
confusion_matrix_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 743  12
##          1 393  39
##
##                Accuracy : 0.6588
##                  95% CI : (0.631, 0.6858)
##     No Information Rate : 0.957
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0917
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.65405
##             Specificity : 0.76471
##          Pos Pred Value : 0.98411
##          Neg Pred Value : 0.09028
##              Prevalence : 0.95703
##          Detection Rate : 0.62595
##    Detection Prevalence : 0.63606
##       Balanced Accuracy : 0.70938
##
##        'Positive' Class : 0
##
```



Analyse:

- Le modèle classe dans l'ensemble test, 743 vins de mauvaises qualités comme étant de mauvaises qualité
- 39 vins sont classé comme étant de bonne qualité coincidantt avec leurs classe réelle
- 393 vins sont classés comme de bonne qualités alors qu'ils sont de mauvaises qualités
- 12 vins sont classé comme étant de mauvaises qualité alors qu'il sont de bonnes qualité

6.8 Conclusion sur le modèle

Le modèle de régression logistique construit pour la prédiction de la qualité du vin blanc est performant. En effet, il n'y a pas de surrapprentissage ou de sous apprentissage. Le modèle apprend moins mais se généralise un peu plus.

7 Comparaison du modèle

Comparons le modèle de régression logistique construit avec le modèle de régression forêts aléatoires (Random Forest)

7.1 Modèle de forêt aléatoire (Random Forest)

7.1.1 Entrainement du modèle

Entrainons le modèle avec 10 arbres de décisions

```
set.seed(123)
model_rf <- randomForest(wine_quality ~ ., data = data_entrainement_smote, ntree = 10,
                         importance = TRUE)
```

7.1.2 Matrice de confusion

Base d'entrainement :

```
# Evaluation de la matrice de confusion
rf_cofusion_matrix_train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3775    8
##          1   13 3022
##
##                Accuracy : 0.9969
##                  95% CI : (0.9953, 0.9981)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9938
##
##  Mcnemar's Test P-Value : 0.3827
##
##             Sensitivity : 0.9966
##             Specificity : 0.9974
##          Pos Pred Value : 0.9979
##          Neg Pred Value : 0.9957
##              Prevalence : 0.5556
##          Detection Rate : 0.5537
##    Detection Prevalence : 0.5549
##       Balanced Accuracy : 0.9970
##
##        'Positive' Class : 0
##
```

Le modèle classe avec 21 erreurs dans la base d'entraînement. Il s'agit de 8 vins de bonnes qualités considérés comme mauvaises et 13 vins de mauvaise qualité classés comme de bonne qualité

Données test :

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1134    0
##          1    2   51
##
##                Accuracy : 0.9983
##                  95% CI : (0.9939, 0.9998)
##     No Information Rate : 0.957
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9799
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.9982
```

```
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9623
##              Prevalence : 0.9570
##          Detection Rate : 0.9553
##    Detection Prevalence : 0.9553
##       Balanced Accuracy : 0.9991
##
##        'Positive' Class : 0
##
```

Le modèle classe les vins dans leurs classes d'origine avec seulement 2 erreurs. Il s'agit de deux vins de mauvaises qualités classés comme de bonnes qualités.

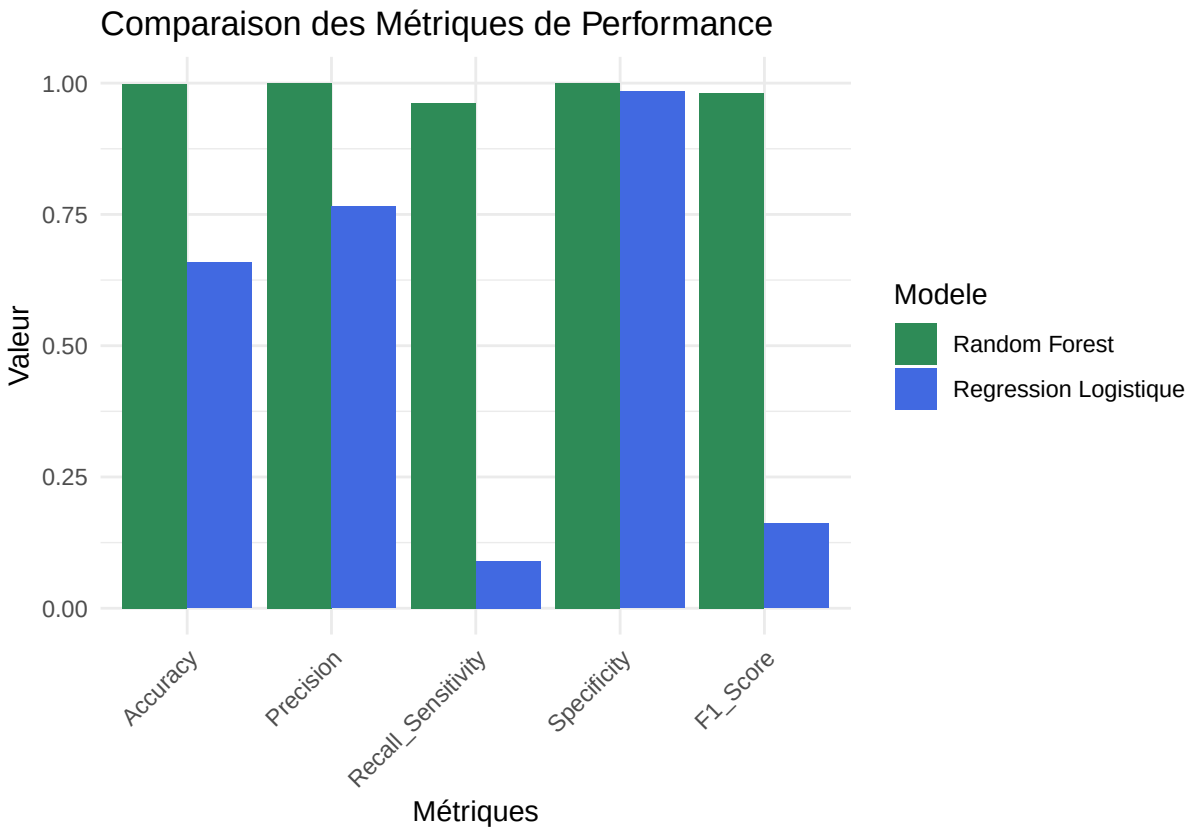
7.2 Comparaison

Tableau des métriques des deux modèles :

```
## [1] "=== COMPARAISON DES MODELES ==="
```

```
##                  Modele Accuracy Precision Recall_Sensitivity Specificity
## 1         Random Forest   0.9983    1.0000             0.9623      1.0000
## 2 Regression Logistique   0.6588    0.7647             0.0903      0.9841
##   F1_Score
## 1   0.9808
## 2   0.1615
```

Visualisation graphique



On remarque que les valeurs de tous les métrique (l'accuracy , la précision ,le recall_sensitivity, la

spécificité et le f1_score) obtenu avec le modèle de forêt aléatoires sont supérieurs à ceux du modèle de régression logistique.

Meilleur modèle selon les metriques :

```
## [1] "\n=== MEILLEUR MODELE PAR METRIQUE ==="
```

```
##      Metrique Meilleur_Modele
## 1    Accuracy   Random Forest
## 2   Precision   Random Forest
## 3      Recall   Random Forest
## 4 Specificity   Random Forest
## 5    F1_Score   Random Forest
```

Conculsion :

A travers la comparaison des différentes métriques des matrices de confusion des deux modèles, nous venons à la conclusion que le modèle de régression forêt aléatoires (Random Forest) est plus meilleur dans la prédiction de la qualité du vin que le modèle de régression logistique.

8 Conclusion

En resumé, dans cette étude, nous avons exploré le jeu de données ***wine+quality*** afin d'analyser les facteurs influençant la qualité du vin et de construire un modèle prédictif à l'aide de la *régression logistique*. Bien que ce modèle ait permis d'obtenir des résultats satisfaisants, certaines limites subsistent, notamment la linéarité entre les prédicteurs et la variable log-odds, la présence de données extrêmes. A cet effet, nous avons comparer la performance du modèle à celle de régression forêt aléatoire et avons trouver que le plus performant est celui de forêt aléatoire.

En perspective, plusieurs axes d'approfondissement peuvent être envisagés. Sur le plan méthodologique, l'utilisation de modèles d'apprentissage plus complexes, comme les forêts aléatoires ou les méthodes d'ensemble, pourrait améliorer la précision des prédictions. De même, l'intégration de techniques d'optimisation et de validation croisée permettrait de renforcer la robustesse du modèle.

D'un point de vue applicatif, ces travaux pourraient déboucher sur le développement d'un outil prédictif simple d'utilisation à destination des producteurs de vin, leur permettant d'évaluer la qualité probable d'un échantillon à partir de ses caractéristiques chimiques.

Ce projet nous a permit de nous familiariser avec les techniques de fouilles de données ainsi que la construction-évaluation de modèles statistiques.

Bibliographie

[1] Rev Mal Respir 2005 ; 22 : 159-62

[2] Wang, J., Chen, Q., & Chen, Y. (2004, August). RBF kernel based support vector machine with universal approximation and its application. In International Symposium on Neural Networks (pp. 512-517). Springer, Berlin, Heidelberg.

[3] P. Cortez, A. Cerdeira, F. Almeida, T. Matos et J. Reis. Modélisation des préférences vitivinicoles par l'extraction de données à partir de propriétés physico-chimiques. Decision Support Systems, Elsevier, 47(4):547-553, 2009.

[4] F Duyme, JJ Claustriaux - Notes de Statistique et d'Informatique, 2006 - orbi.uliege.be