Projeto - Red Wine Quality

André Campos da Silva

18 de Janeiro, 2021

Red Wine Quality

Realizar uma análise nos dados e construir um modelo que permita determinar a qualidade do vinho em uma escala de 0 a 10,baseados nas variáveis preditoras Dataset disponível no link - https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009/notebooks (https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009/notebooks)

Os dois conjuntos de dados estão relacionados com as variantes tinto e branco do vinho “Vinho Verde” português. Para mais detalhes, consulte a referência [Cortez et al., 2009]. Devido a questões de privacidade e logística, apenas variáveis físico-químicas (entradas) e sensoriais (a saída) estão disponíveis (por exemplo, não há dados sobre os tipos de uva, marca de vinho, preço de venda do vinho, etc.). Esses conjuntos de dados podem ser vistos como tarefas de classificação ou regressão. As classes são ordenadas e não balanceadas (por exemplo, há muito mais vinhos normais do que excelentes ou ruins)

Dcionario dos dados Input variables (based on physicochemical tests):

1 - fixed acidity - acidez_fixa

2 - volatile acidity - acidez_volátil

3 - citric acid - ácido_cítrico

4 - residual sugar - açúcar_residual

5 - chlorides - cloretos

6 - free sulfur dioxide - dióxido_de_enxofre_livre

7 - total sulfur dioxide - dióxido_de_enxofre_total

8 - density - densidade

9 - pH - pH

10 - sulphates - sulfatos

11 - alcohol - álcool

12 - quality (score between 0 and 10) - qualidade (pontuação entre 0 e 10) - variável target.

Coletando os dados

```
# Pacotes usados no projeto
library('tidyverse')
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library('caret')
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library('ROSE')
```

```
## Loaded ROSE 0.0-3
```

```
library('data.table')
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library('gridExtra')
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library('randomForest')
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library('DMwR')
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library('e1071')
library('rpart')
library('C50')
library("ROCR")
library('caTools')
library('corrplot')
```

```
## corrplot 0.84 loaded
```

```
library('kernlab')
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

Carregando os Dados

```
# Carrego o dataset para análise

df_train <- read_csv('Dados/winequality-red.csv')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   `fixed acidity` = col_double(),
##   `volatile acidity` = col_double(),
##   `citric acid` = col_double(),
##   `residual sugar` = col_double(),
##   chlorides = col_double(),
##   `free sulfur dioxide` = col_double(),
##   `total sulfur dioxide` = col_double(),
##   density = col_double(),
##   pH = col_double(),
##   sulphates = col_double(),
##   alcohol = col_double(),
##   quality = col_double()
## )
```

```
# Crio as colunas com os nomes em portugues
col_names <- c('Acidez_fixa','Acidez_volátil','Ácido_cítrico','Açúcar_residual',
               'Cloretos','Dióxido_de_enxofre_livre','Dióxido_de_enxofre_total',
                 'Densidade','pH','Sulfatos','Álcool','Qualidade')
# Crio as colunas com os nomes em portugues
names(df_train) <- col_names

df_train <- as.data.frame(df_train)

# Verifico as primeiras linhas
head(df_train)
```

```
##   Acidez_fixa Acidez_volátil Ácido_cítrico Açúcar_residual Cloretos
## 1         7.4           0.70          0.00             1.9    0.076
## 2         7.8           0.88          0.00             2.6    0.098
## 3         7.8           0.76          0.04             2.3    0.092
## 4        11.2           0.28          0.56             1.9    0.075
## 5         7.4           0.70          0.00             1.9    0.076
## 6         7.4           0.66          0.00             1.8    0.075
##   Dióxido_de_enxofre_livre Dióxido_de_enxofre_total Densidade   pH Sulfatos
## 1                       11                       34    0.9978 3.51     0.56
## 2                       25                       67    0.9968 3.20     0.68
## 3                       15                       54    0.9970 3.26     0.65
## 4                       17                       60    0.9980 3.16     0.58
## 5                       11                       34    0.9978 3.51     0.56
## 6                       13                       40    0.9978 3.51     0.56
##   Álcool Qualidade
## 1    9.4         5
## 2    9.8         5
## 3    9.8         5
## 4    9.8         6
## 5    9.4         5
## 6    9.4         5
```

Análise Exploratória de Dados

```
# Dimensões dos dados
dim(df_train)
```

```
## [1] 1599   12
```

```
# Tipo dos dados das variáveis
glimpse(df_train)
```

```
## Rows: 1,599
## Columns: 12
## $ Acidez_fixa              <dbl> 7.4, 7.8, 7.8, 11.2, 7.4, 7.4, 7.9, 7.3, 7...
## $ Acidez_volátil           <dbl> 0.700, 0.880, 0.760, 0.280, 0.700, 0.660, ...
## $ Ácido_cítrico            <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.06, ...
## $ Açúcar_residual          <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2, 2....
## $ Cloretos                 <dbl> 0.076, 0.098, 0.092, 0.075, 0.076, 0.075, ...
## $ Dióxido_de_enxofre_livre <dbl> 11, 25, 15, 17, 11, 13, 15, 15, 9, 17, 15,...
## $ Dióxido_de_enxofre_total <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 102, 6...
## $ Densidade                <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978, 0....
## $ pH                       <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.30, ...
## $ Sulfatos                 <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.46, ...
## $ Álcool                   <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, 10.0, 9...
## $ Qualidade                <dbl> 5, 5, 5, 6, 5, 5, 5, 7, 7, 5, 5, 5, 5, 5, ...
```

```
# Sumário estatístico das variáveis
summary(df_train)
```

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

```
# Verifico se existe valores nulos nos dados
sum(is.na(df_train))
```

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

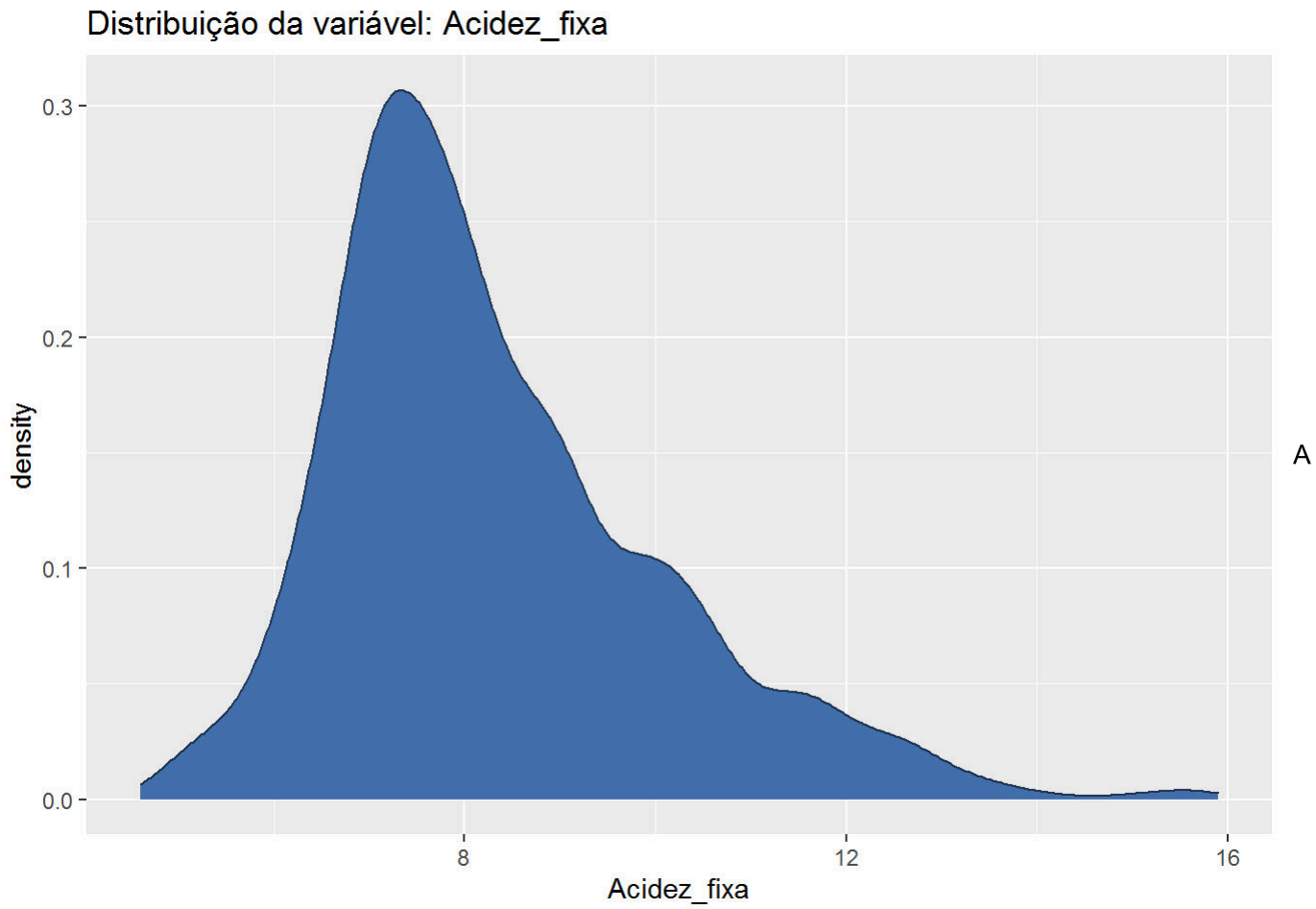
Analise em grafico de cada varável

Funções auxiliares

```
dist_plot <- function(data, col) {

  ggplot() +
    geom_density(aes(data[,col]), fill = '#4271AE', colour = "#1F3552") +
    labs(title = paste('Distribuição da variável:',col), x = col)

}

box_plot <- function(data, col, title, xlab) {

  ggplot() +
    geom_boxplot(aes(x = data[, col]), fill = '#4271AE', colour = "#1F3552") +
    labs(title = paste('BoxPlot da variável:',col), x = col) +
    theme(axis.text.y = element_blank())
}

bar_plot <- function(data, col, title, xlab) {

  ggplot() +
    geom_bar(aes(x = data[, col]), fill = '#4271AE', colour = "#1F3552") +
    labs(title = paste('Grafico de barra da variável: ',col), x = col)

}
```

Acidez_fixa

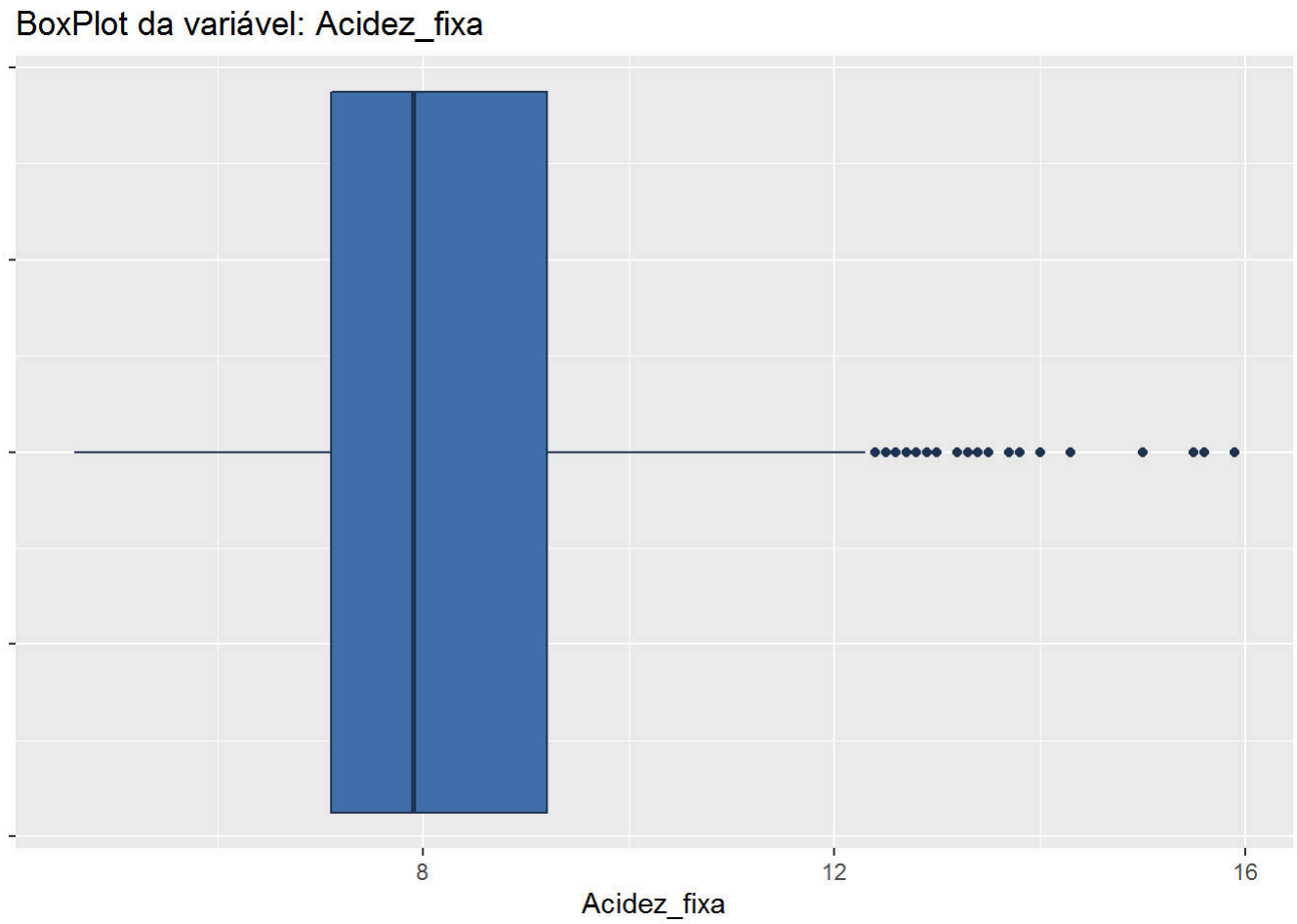
```
dist_plot(df_train, 'Acidez_fixa')
```



A

varável aparente ter uma distribuição assimétrica positiva, dentre os níveis de acidez fixa, a concentração maior fica um pouco abaixo do 8 aproximadamente.

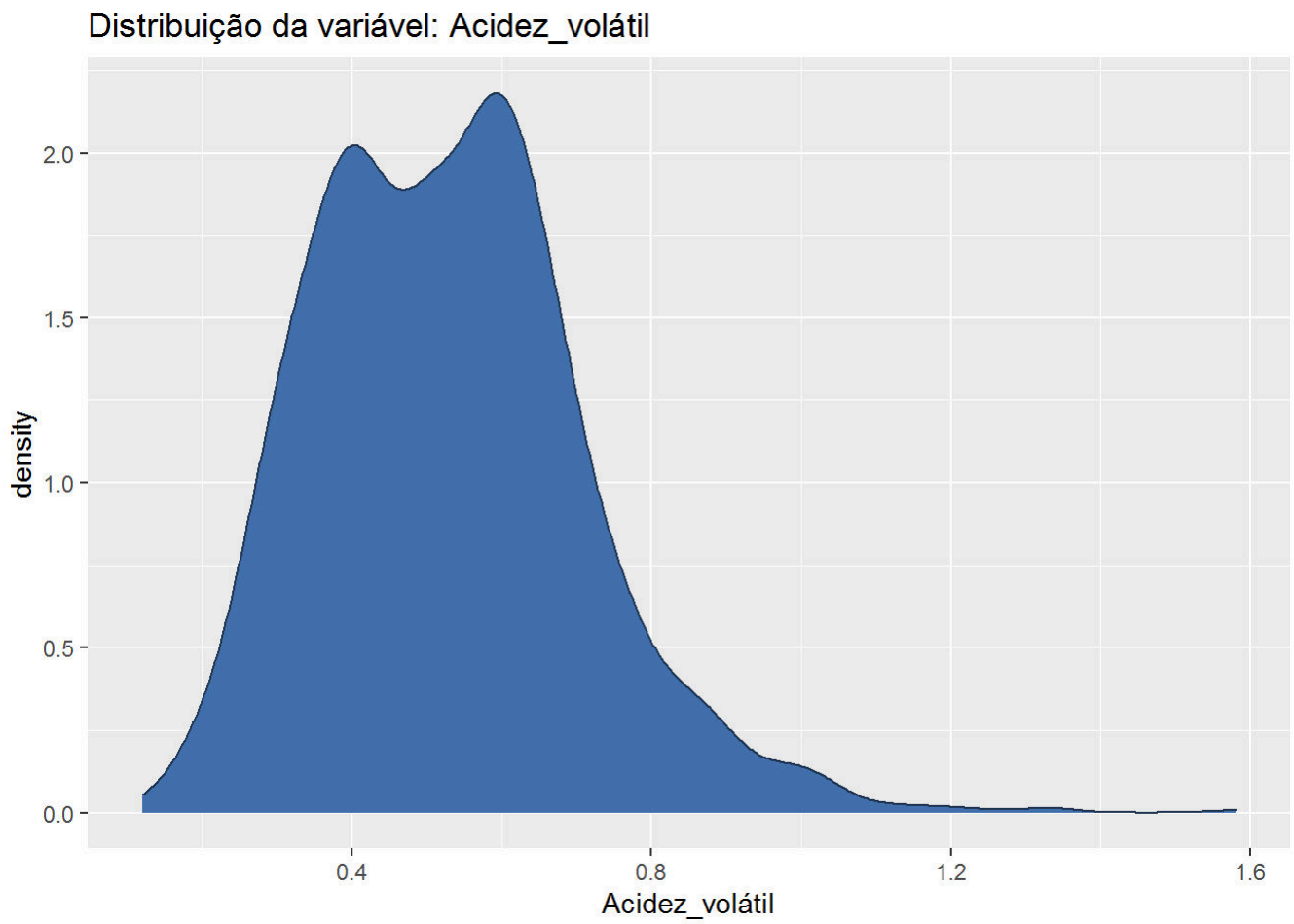
```
box_plot(df_train, 'Acidez_fixa')
```



O boxplot mostra que a varável posse alguns outliers, alguns vinhos aparenta ter uma acidez fixa, muito acima do padrão.

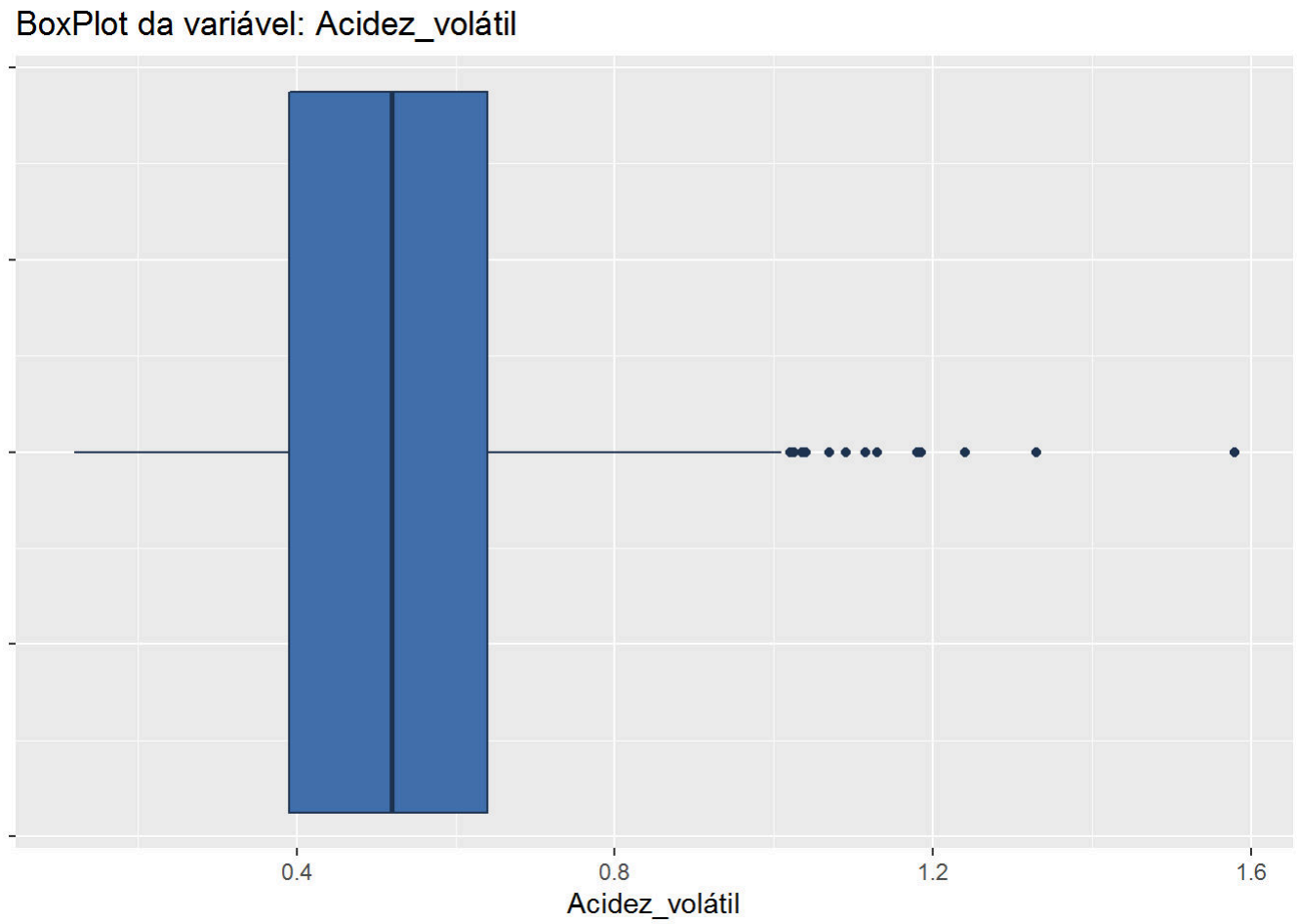
Acidez_volátil

```
dist_plot(df_train, 'Acidez_volátil')
```



O gráfico mostra uma que existe dois picos de alta densidade no decorrer dos dados, nos níveis de acidez volátil 0.4 e 0.6 aproximadamente.

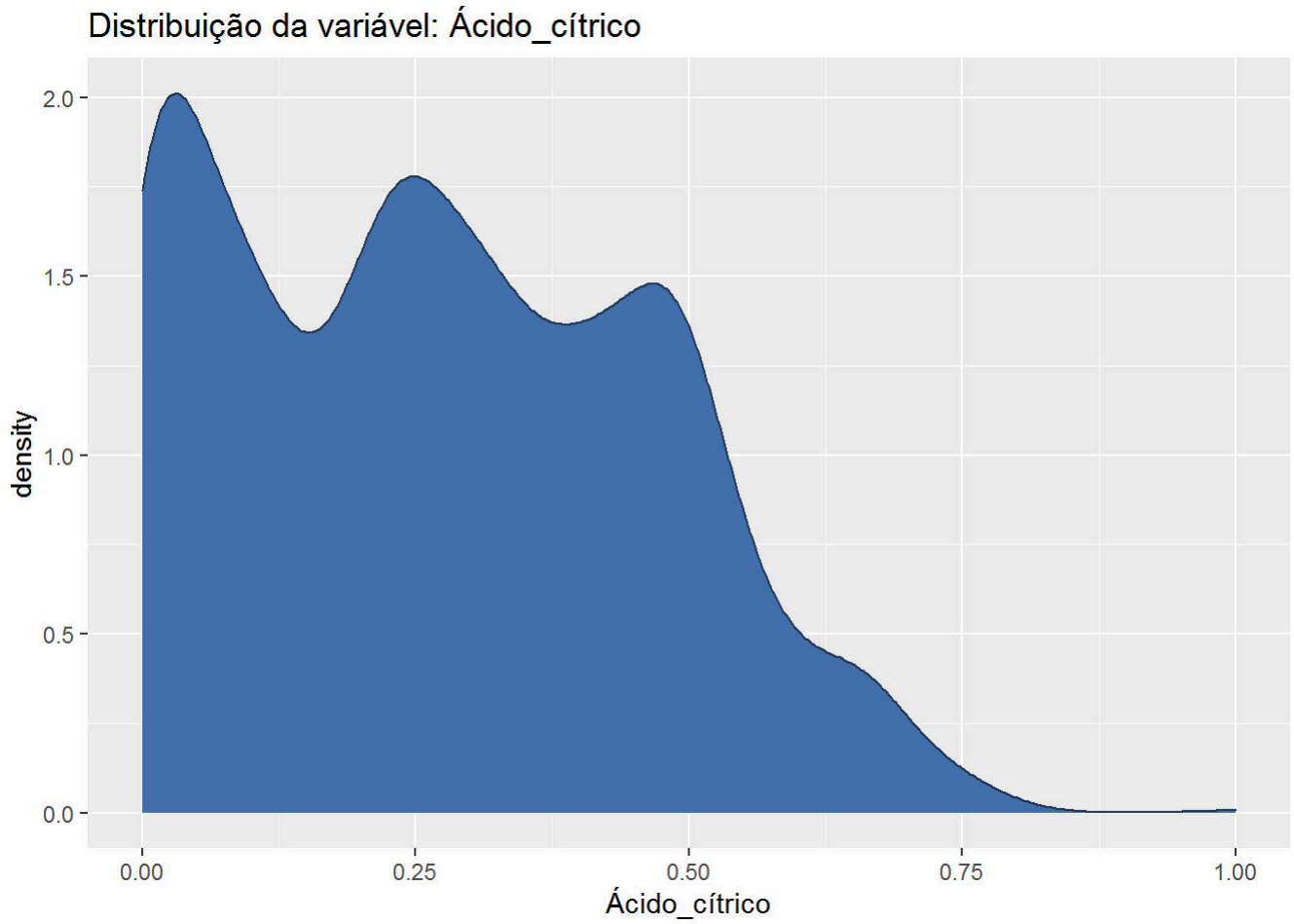
```
box_plot(df_train, 'Acidez_volátil')
```



O gráfico mostra que a poucos vinhos possuem acidez volátil fora dos padrões, percebemos poucos outliers, indicando isso.

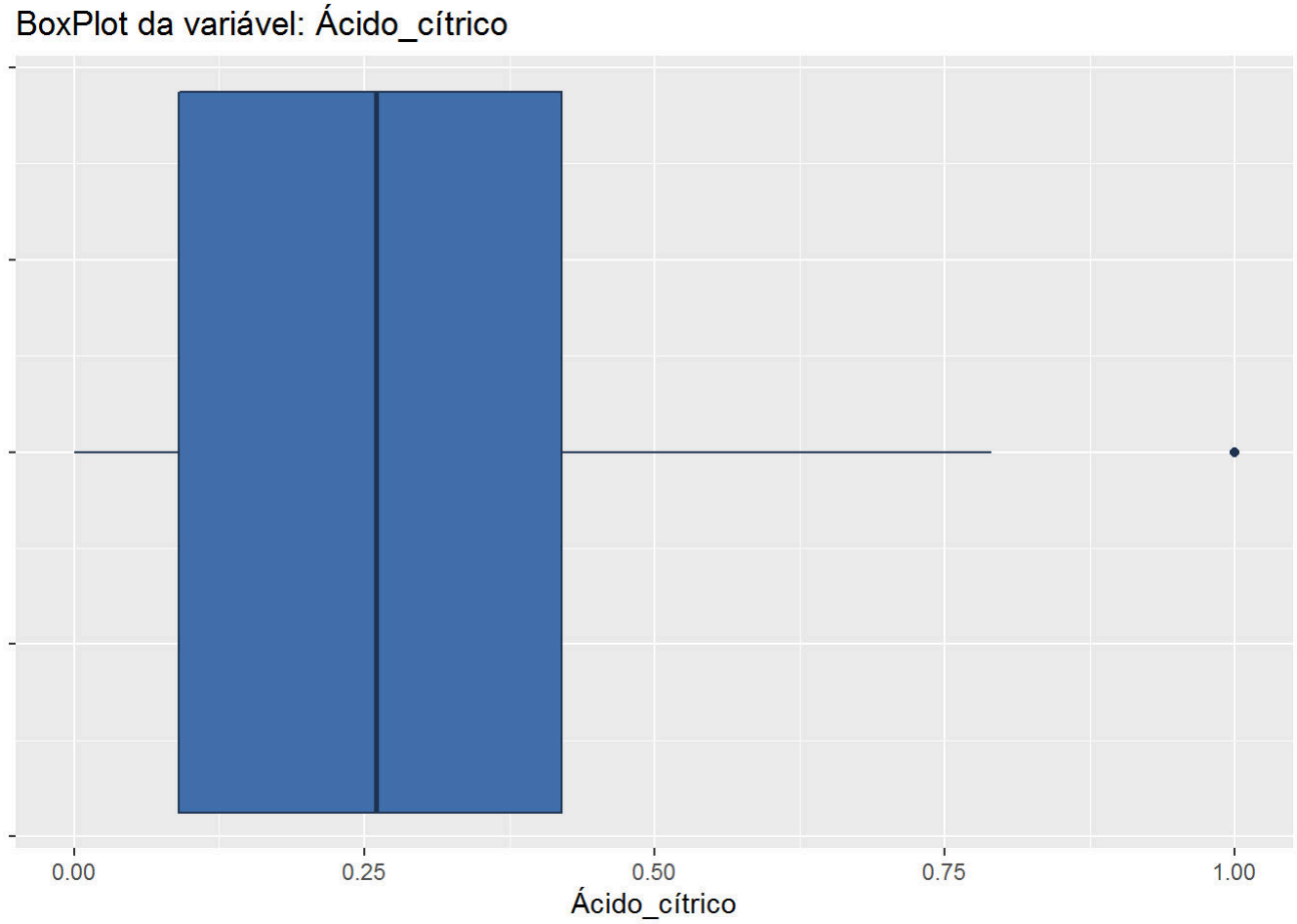
Ácido_cítrico

```
dist_plot(df_train, 'Ácido_cítrico')
```



O gráfico mostra que existe uma distribuição variada com relação ao nível de ácido cítrico, e poucos são os vinhos que possuem o ácido cítrico muito alto.

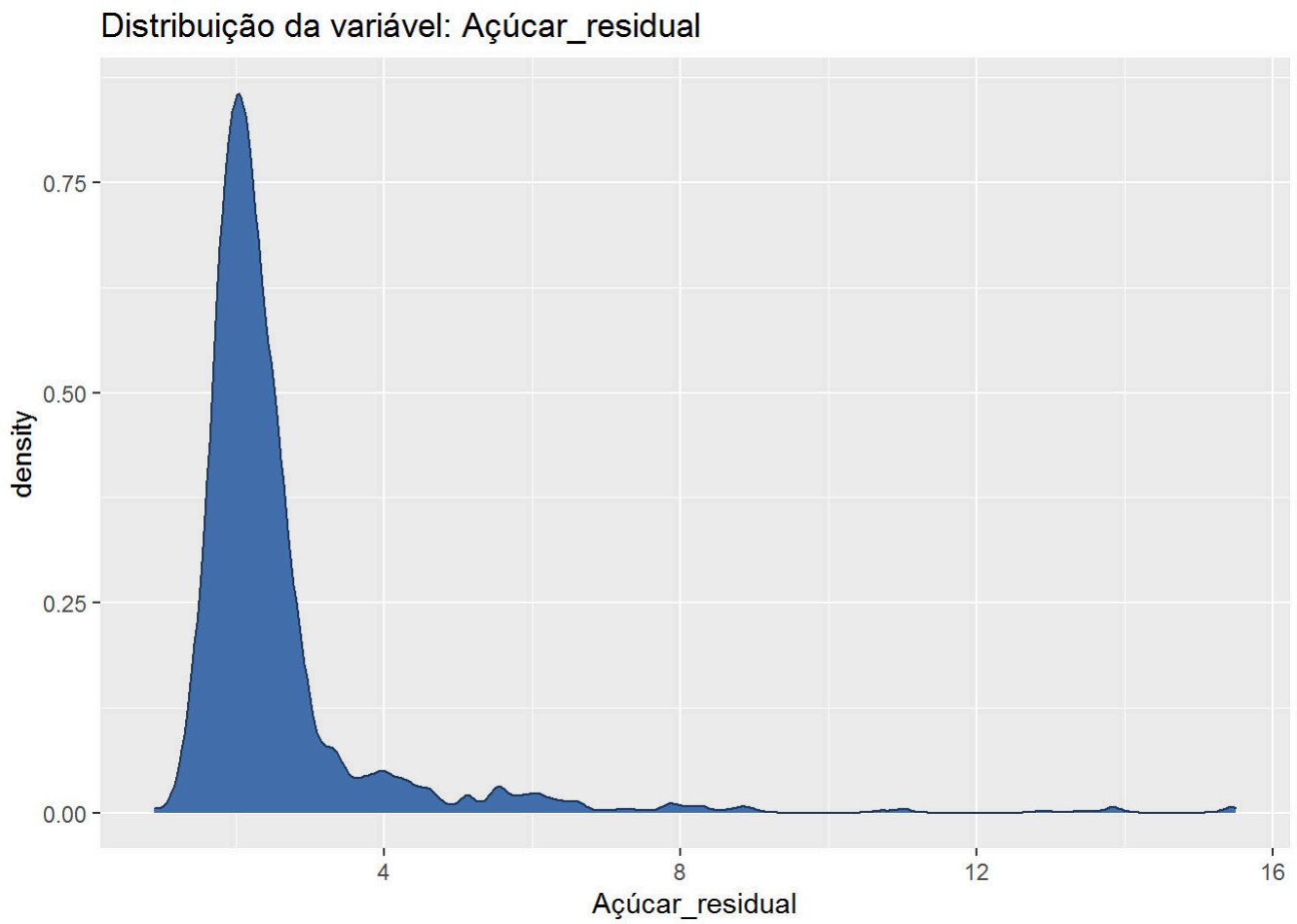
```
box_plot(df_train, 'Ácido_cítrico')
```



O gráfico mostra apenas um valor fora do padrão, indicando que o praticamente todos os vinhos não tem uma diferença expressiva no fator ácido cítrico.

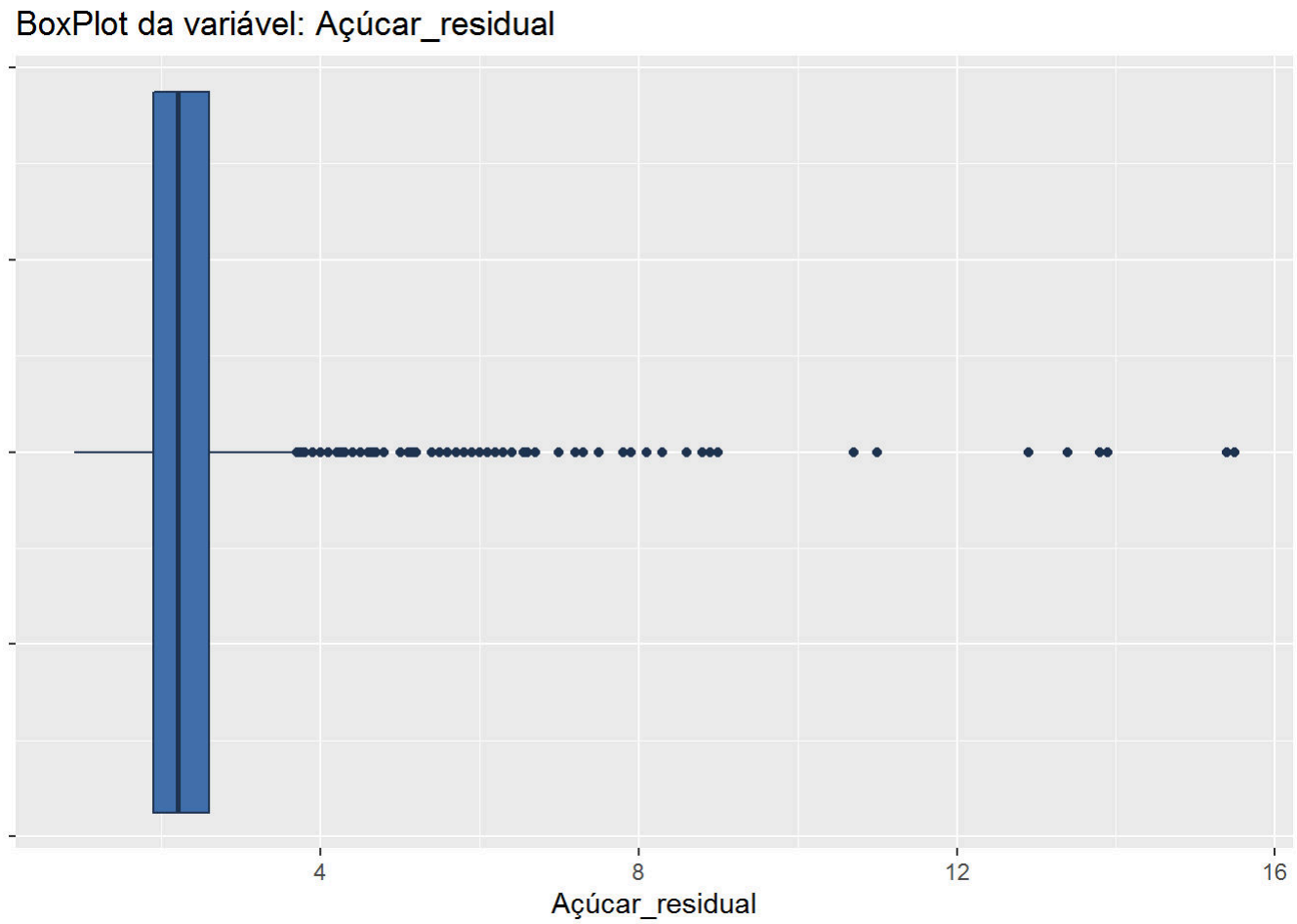
Açúcar_residual

```
dist_plot(df_train, 'Açúcar_residual')
```



O gráfico mostra uma concentração maior no nível de açúcar residual entre 0 ,3, indicando que boa parte dos vinhos possuem um nível de açúcar residual significativo.

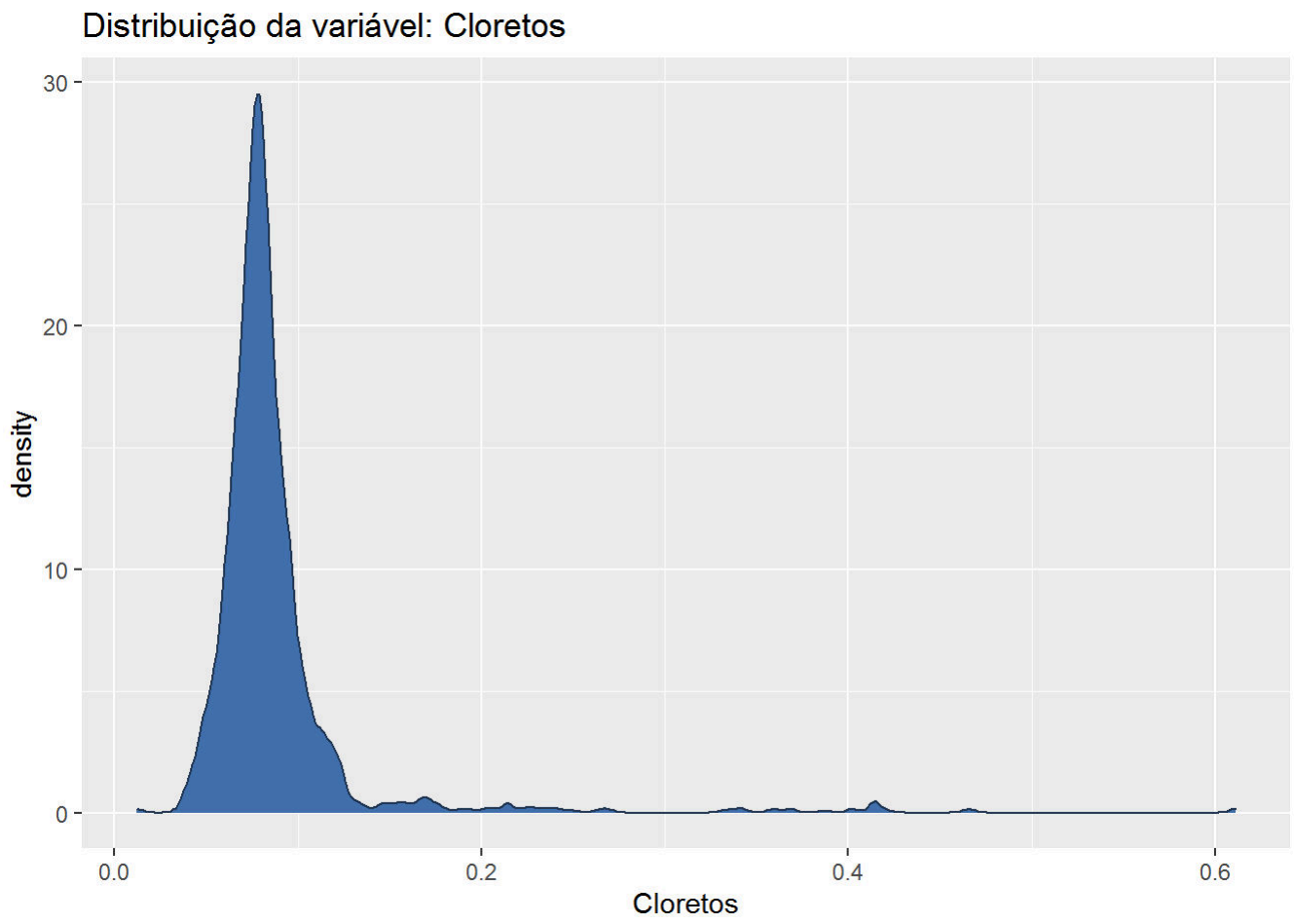
```
box_plot(df_train, 'Açúcar_residual')
```



O gráfico mostra que embora a maioria dos vinhos possua um nível de açúcar residual dento da media, existe alguns que tem um nível maior, chamados outliers.

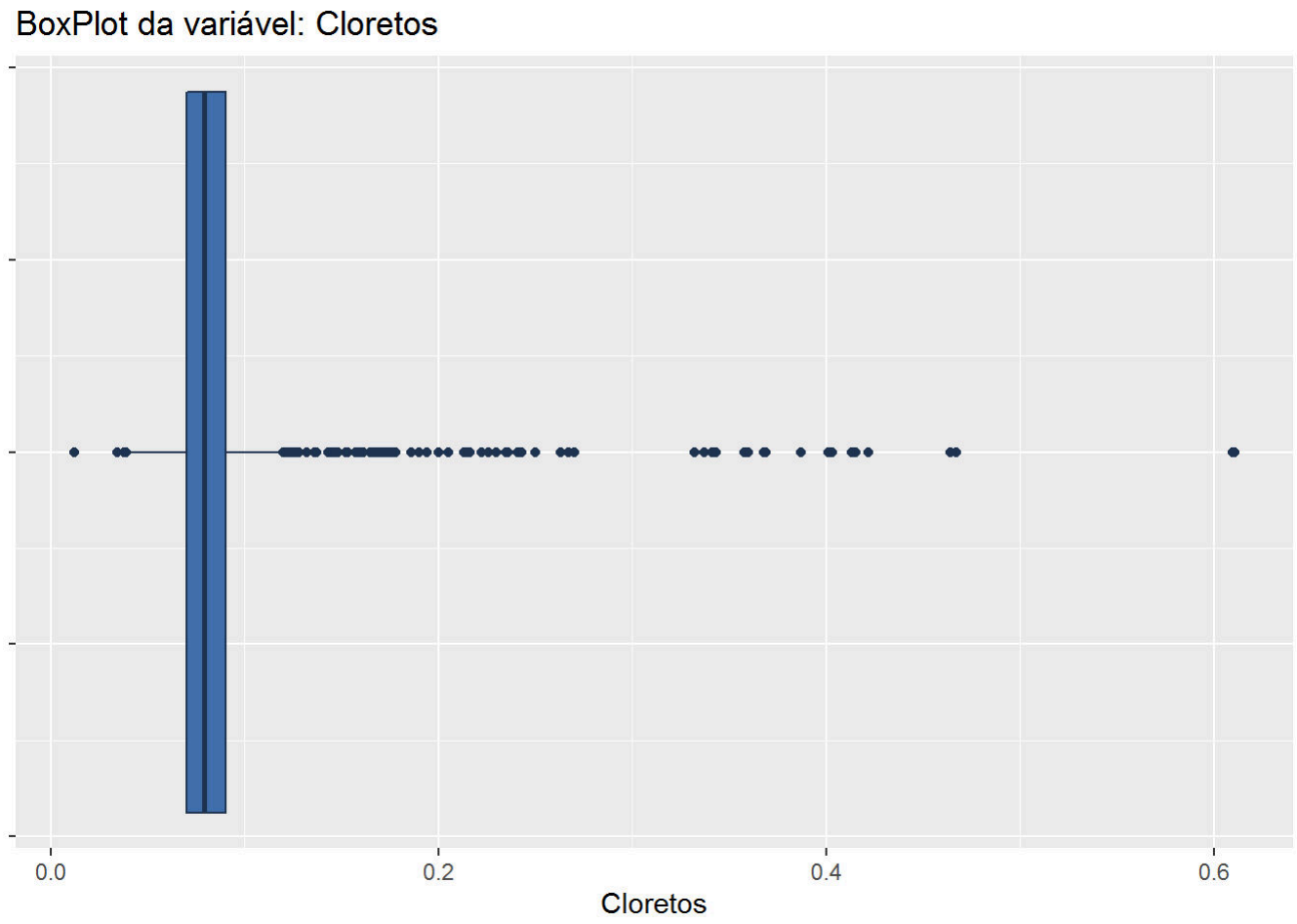
Cloretos

```
dist_plot(df_train, 'Cloretos')
```



O gráfico mostra que o nível de cloreto tem uma concentração maior no valor 0,1 aproximadamente.

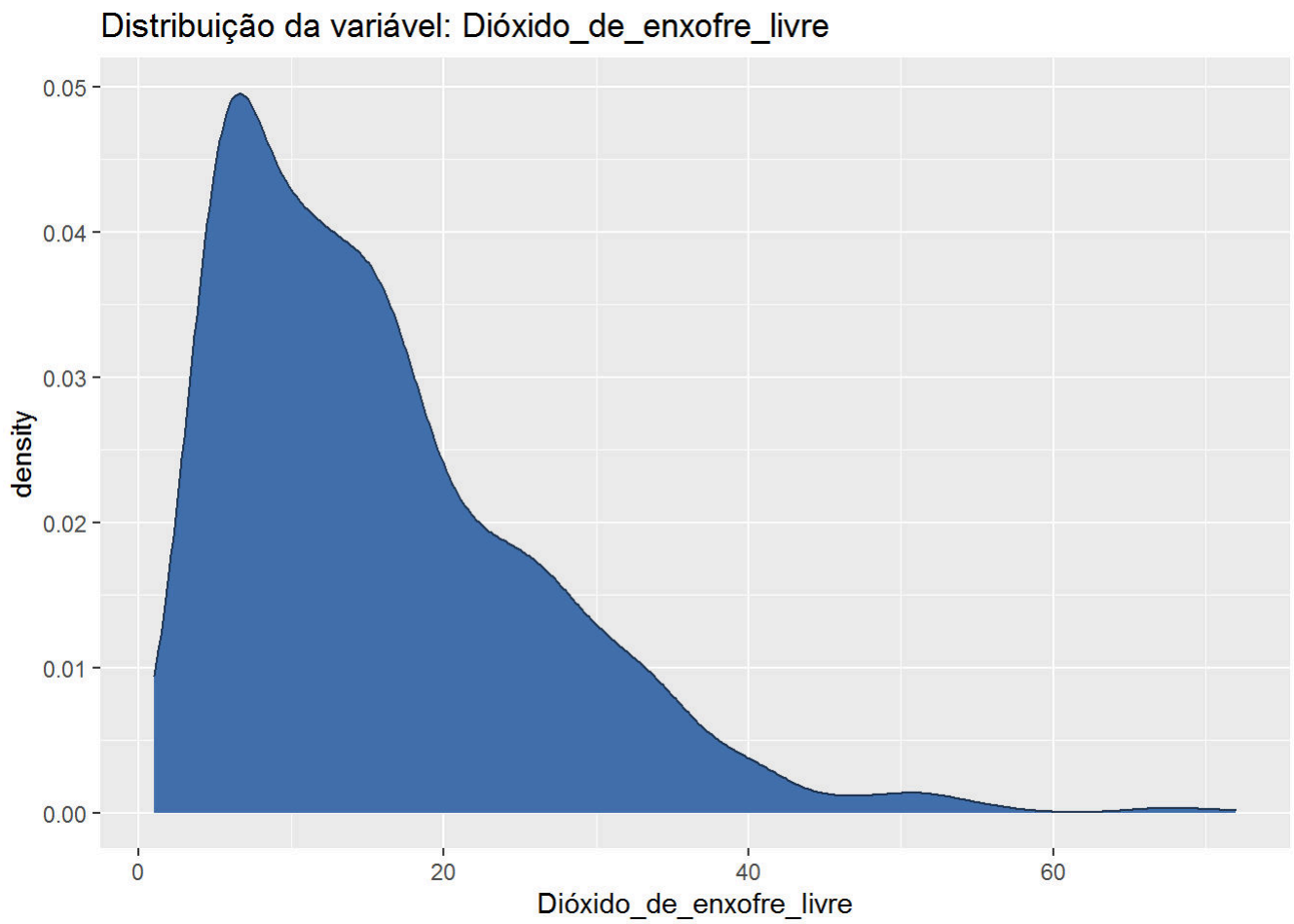
```
box_plot(df_train, 'Cloretos')
```



O gráfico mostra outliers tanto abaixo da média quanto acima, embora a concentração de cloretos fique em torno de 0.1 temos vinhos com concentrações poucos menores como pouco maiores.

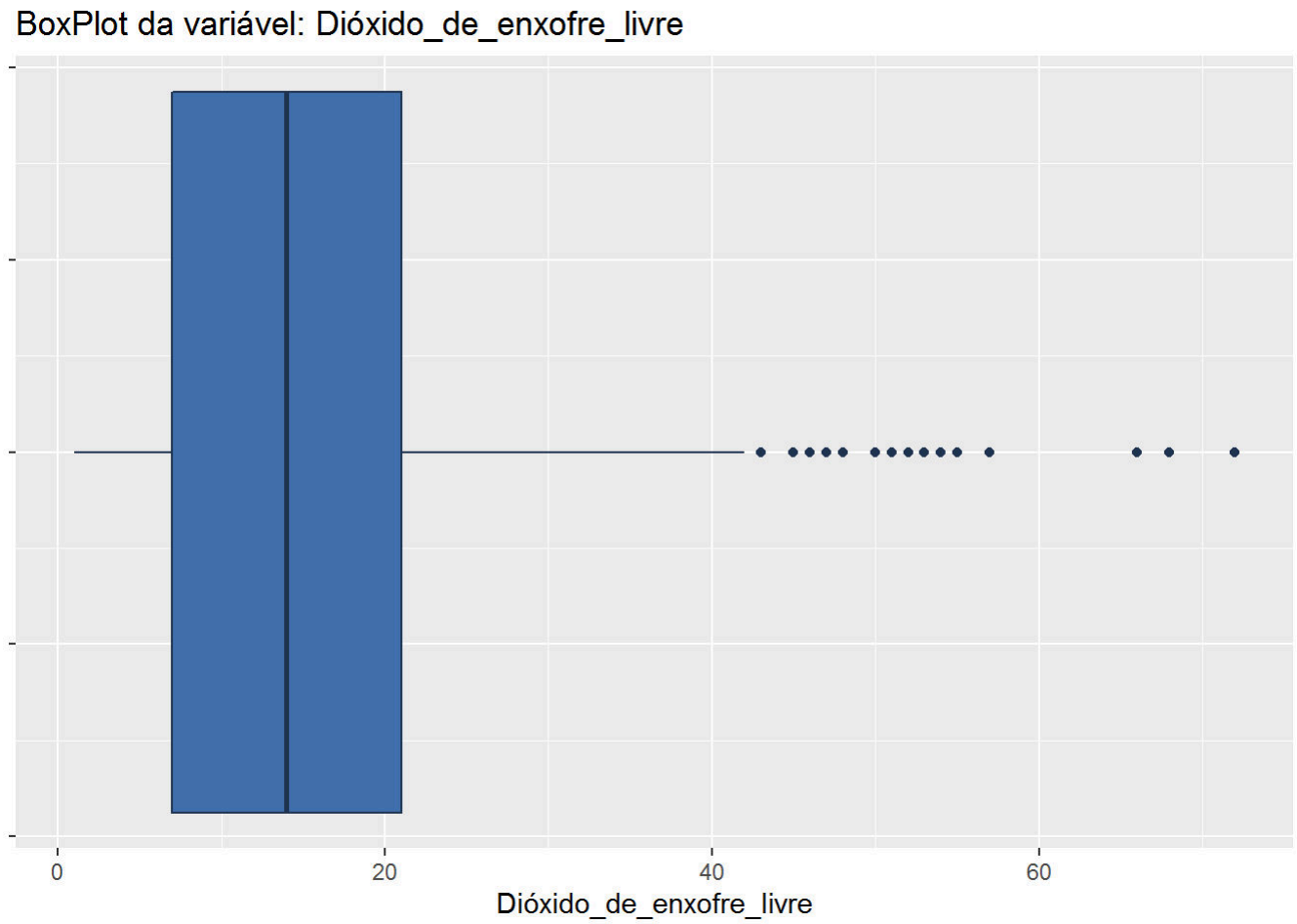
Dióxido_de_enxofre_livre

```
dist_plot(df_train, 'Dióxido_de_enxofre_livre')
```



O gráfico mostra que o nível de dióxido de enxofre livre alto entre os valores 0 e 20 tendo uma diminuição bem acentuada a partir do 20.

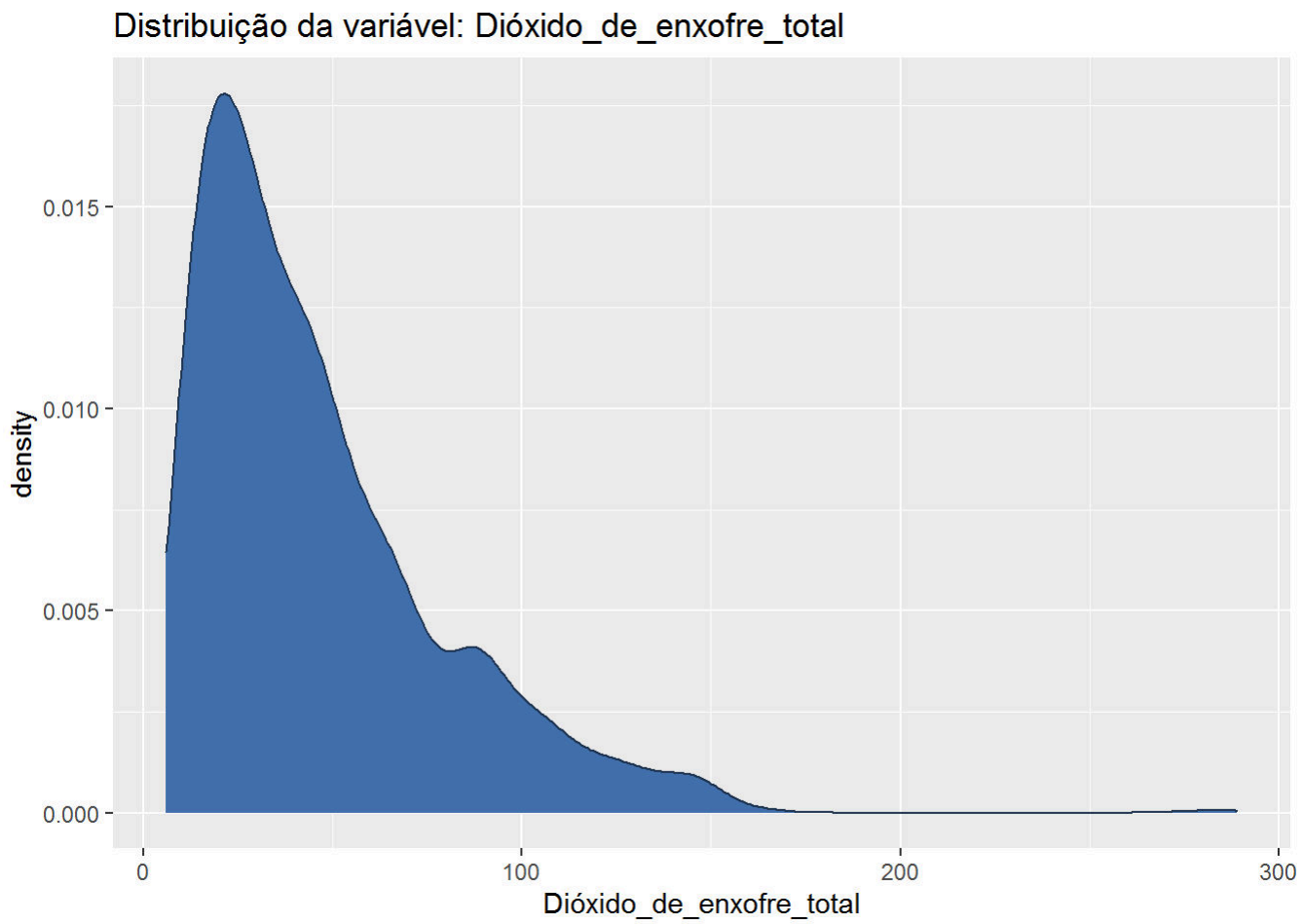
```
box_plot(df_train, 'Dióxido_de_enxofre_livre')
```



O gráfico mostra poucos outliers, em sua maioria os vinhos possuem o nível de dióxido de enxofre livre dentro da média.

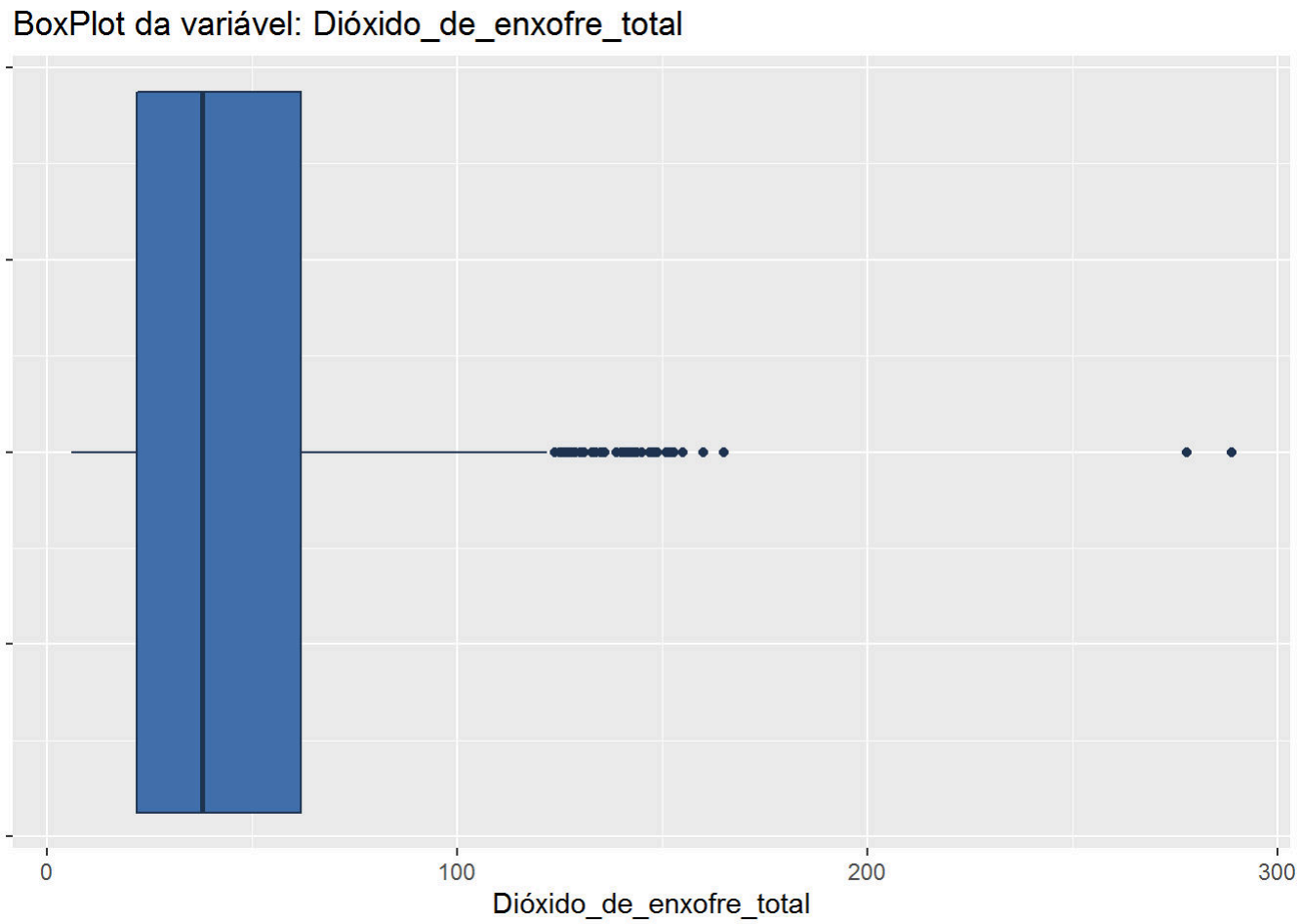
Dióxido_de_enxofre_total

```
dist_plot(df_train, 'Dióxido_de_enxofre_total')
```



O gráfico mostra uma concentração grande de dióxido de enxofre total entre 0 e 30 aproximadamente, caindo gradativamente.

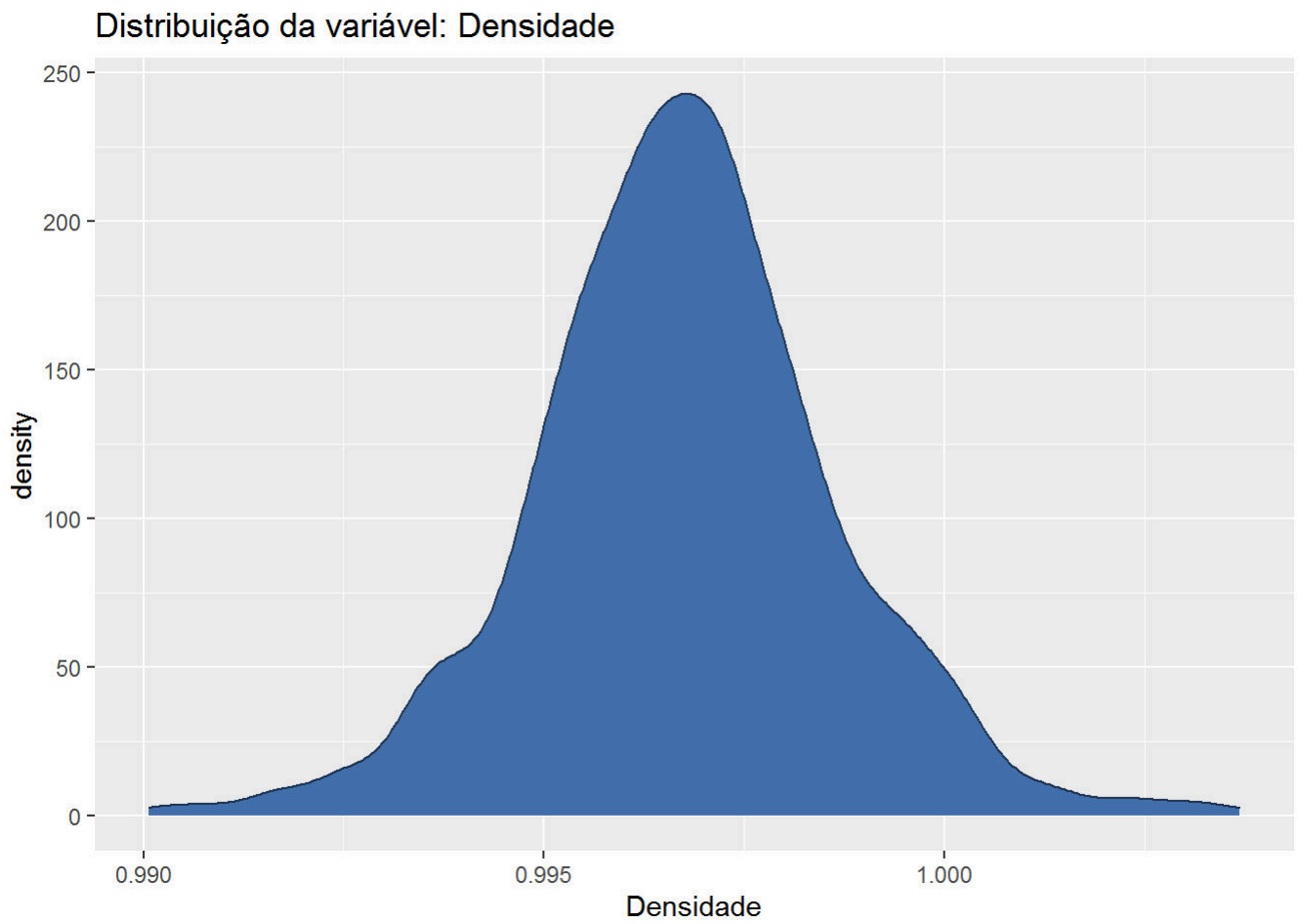
```
box_plot(df_train, 'Dióxido_de_enxofre_total')
```



O gráfico mostra que assim com o dióxido de enxofre livre o total, não possuem muitos outliers, em sua maioria os níveis estão dentro da média.

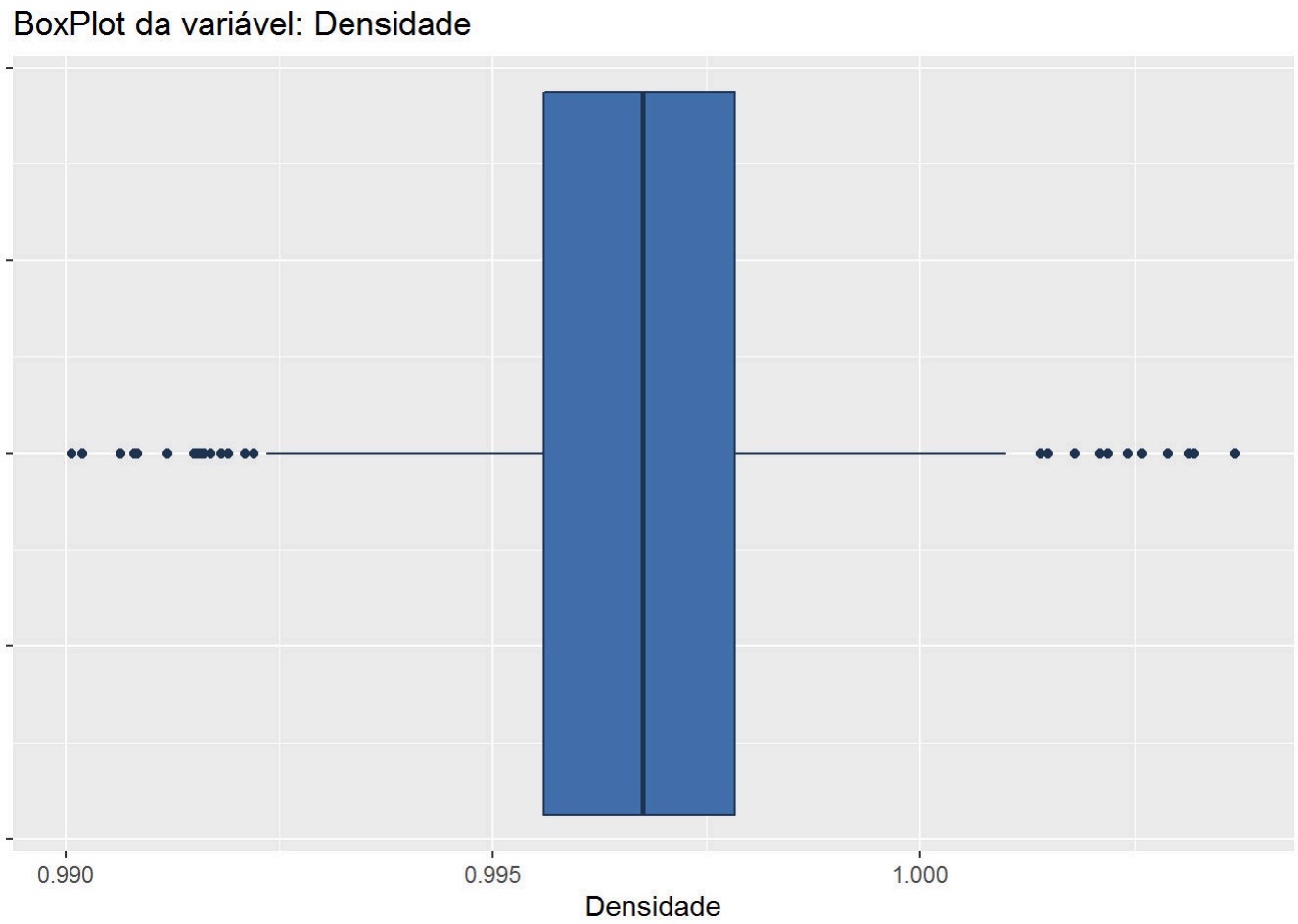
Densidade

```
dist_plot(df_train, 'Densidade')
```



O gráfico mostra que a densidade aparenta ter uma distribuição normal perfeita.

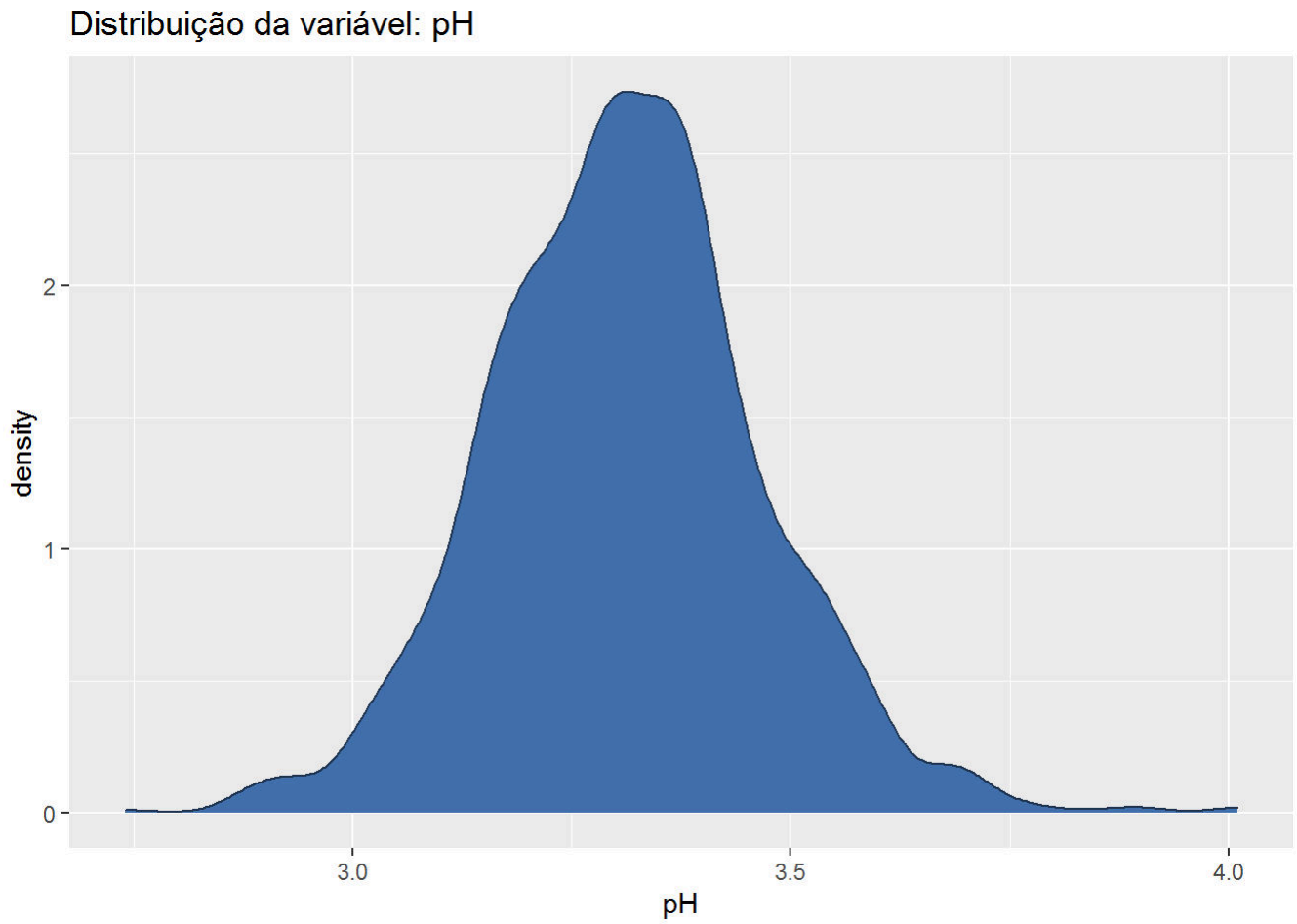
```
box_plot(df_train, 'Densidade')
```



O gráfico mostra que a densidade posse outliers tanto abaixo quanto acima da média, embora poucos, existem vinhos com níveis de densidade abaixo e acima da média.

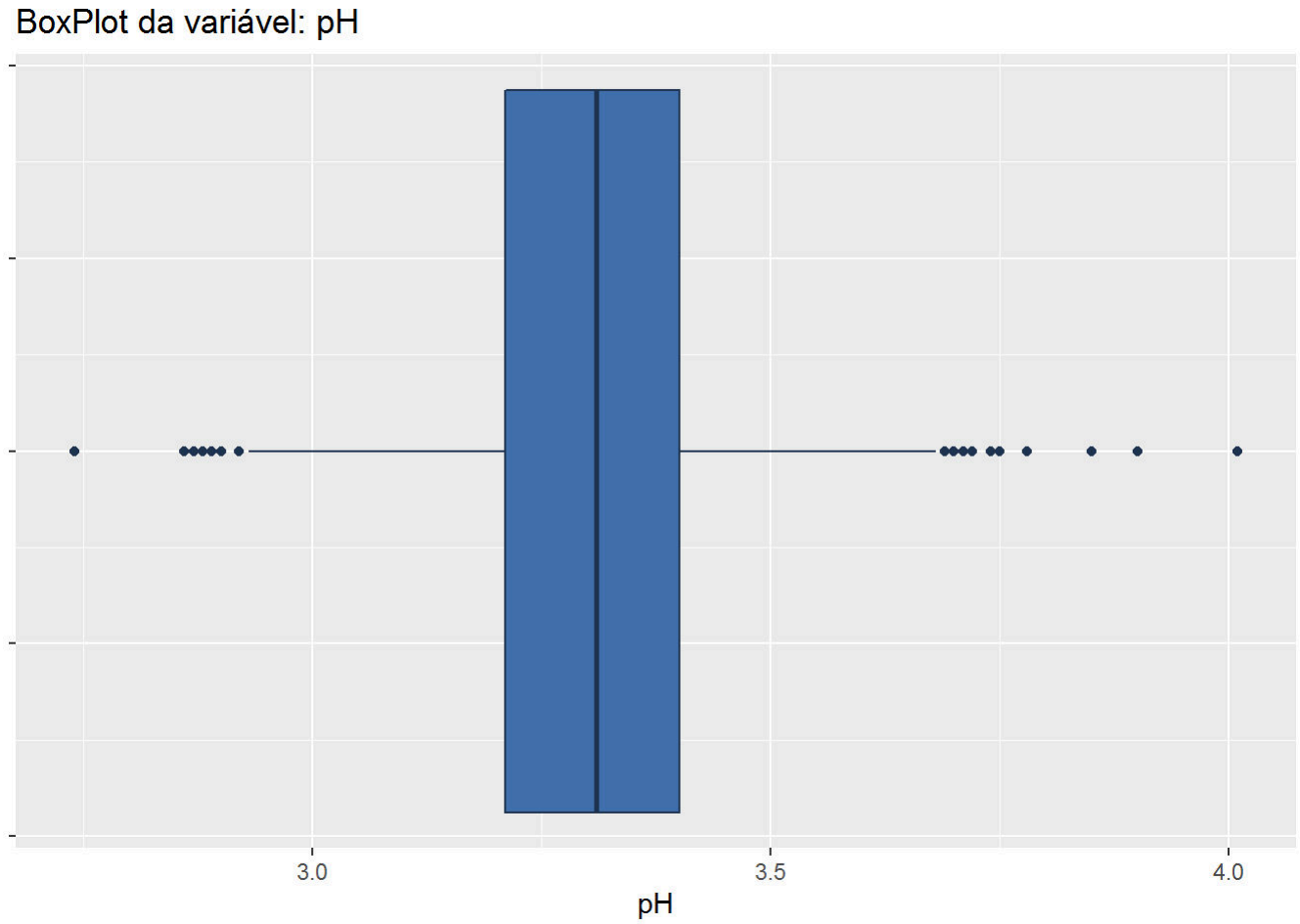
pH

```
dist_plot(df_train, 'pH')
```



O gráfico mostra que a densidade aparenta ter uma distribuição normal perfeita.

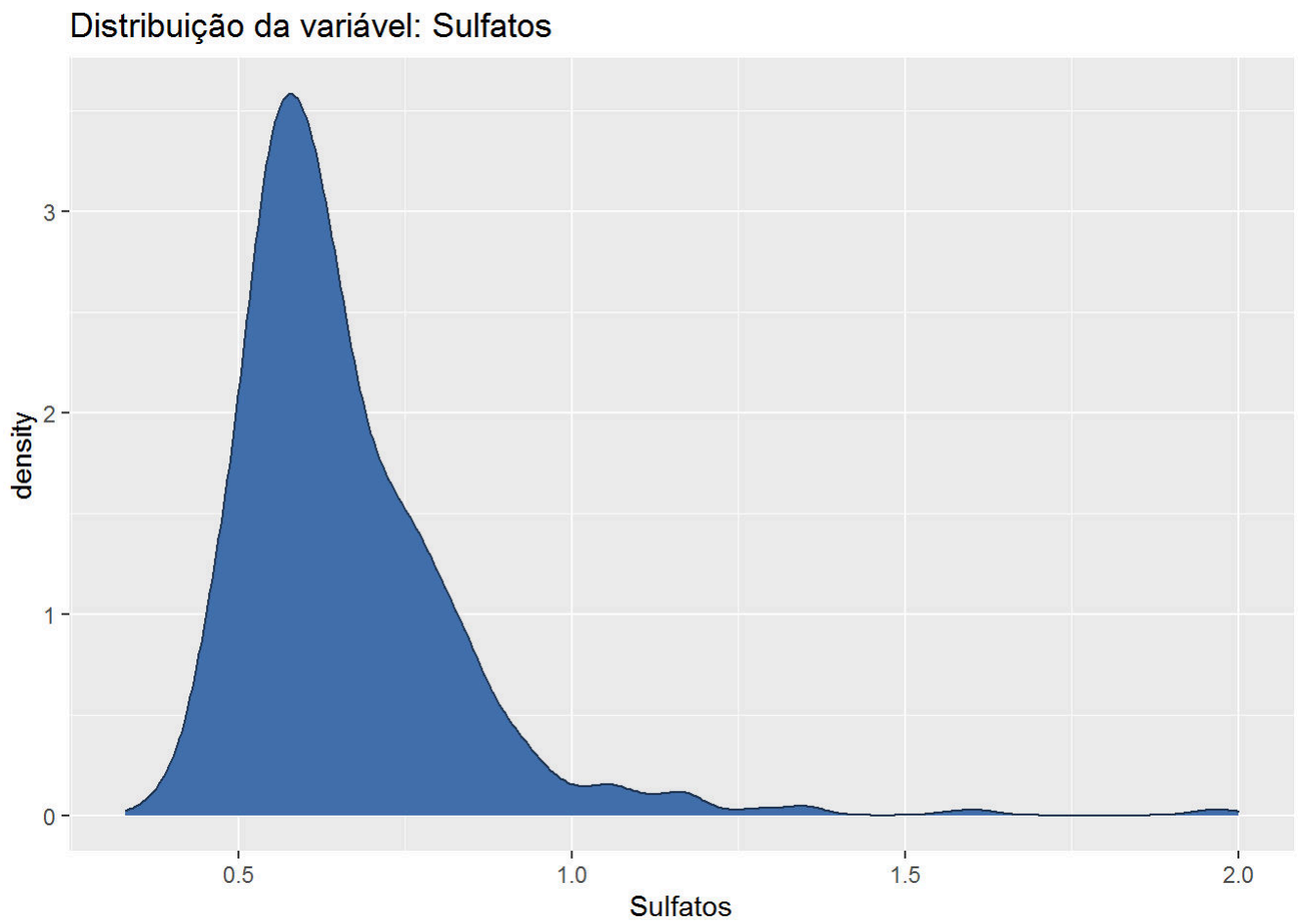
```
box_plot(df_train, 'pH')
```



O gráfico mostra outliers tanto abaixo quanto acima da média, embora poucos, existem vinhos com níveis de pH abaixo e acima da média.

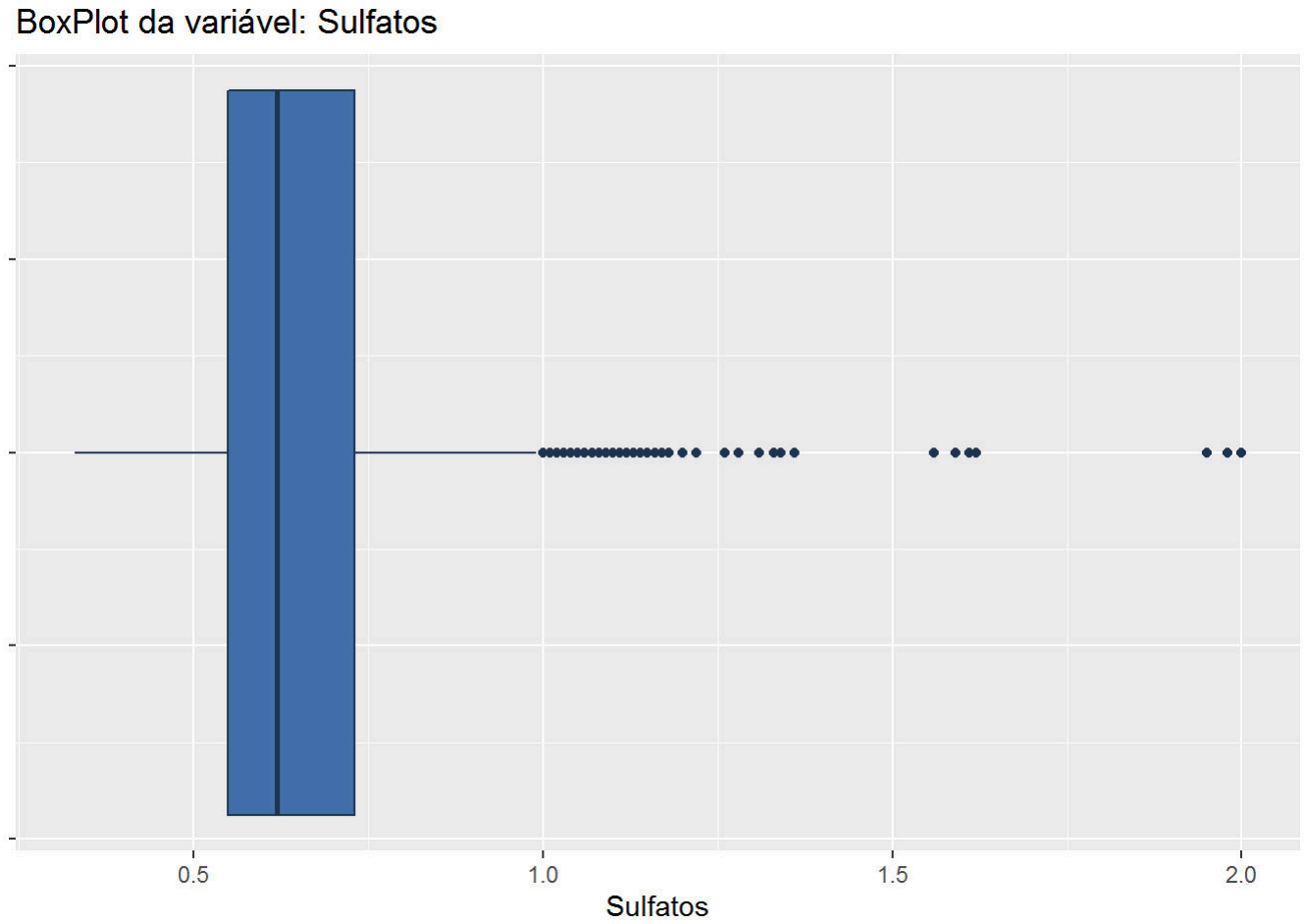
Sulfatos

```
dist_plot(df_train, 'Sulfatos')
```



O gráfico mostra que os níveis de sulfatos nos vinhos possuem um pico entre 0.5 a 0.7 aproximadamente .

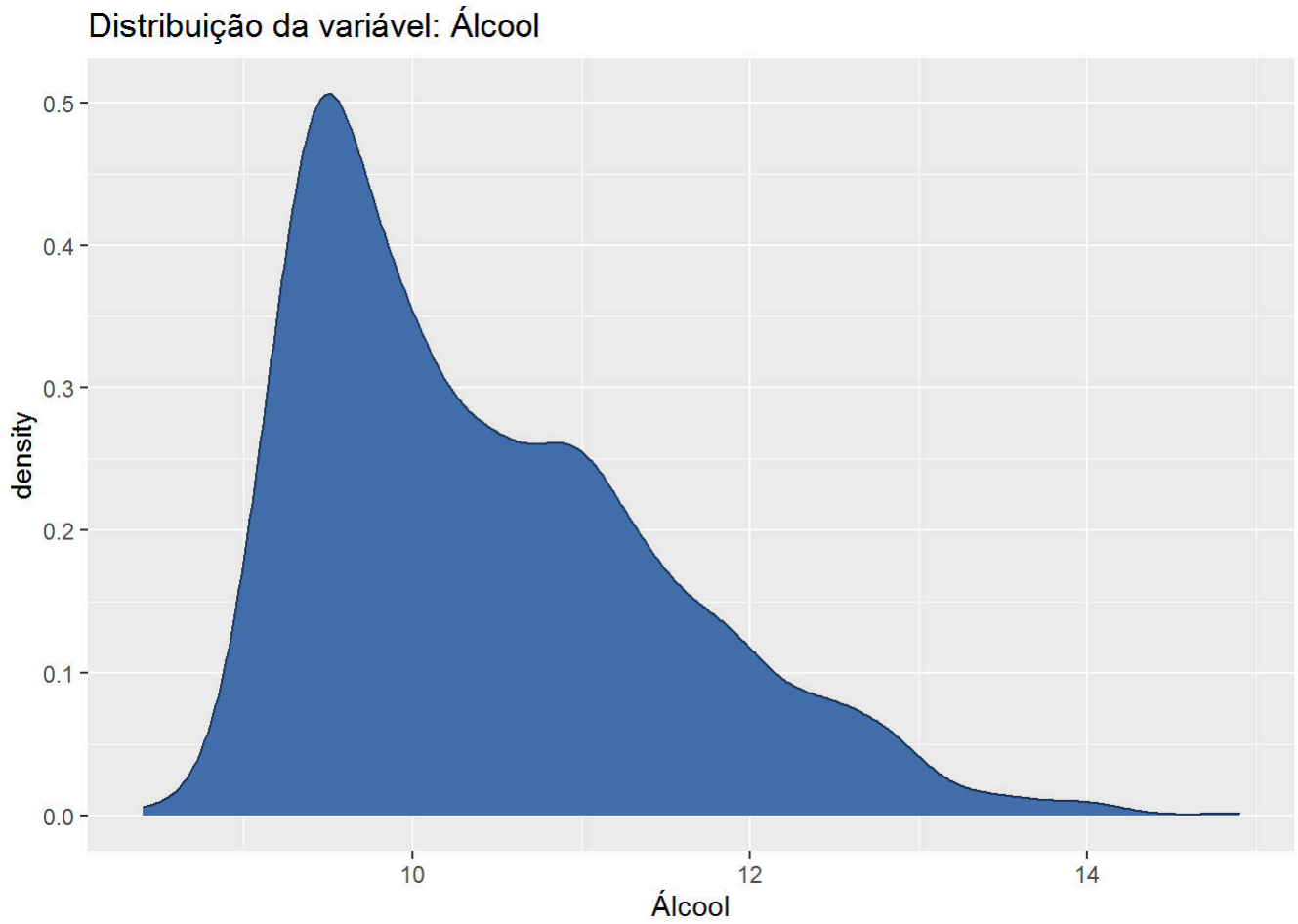
```
box_plot(df_train, 'Sulfatos')
```



O gráfico mostra que os níveis de sulfatos nos vinhos possuem alguns outliers acima da média.

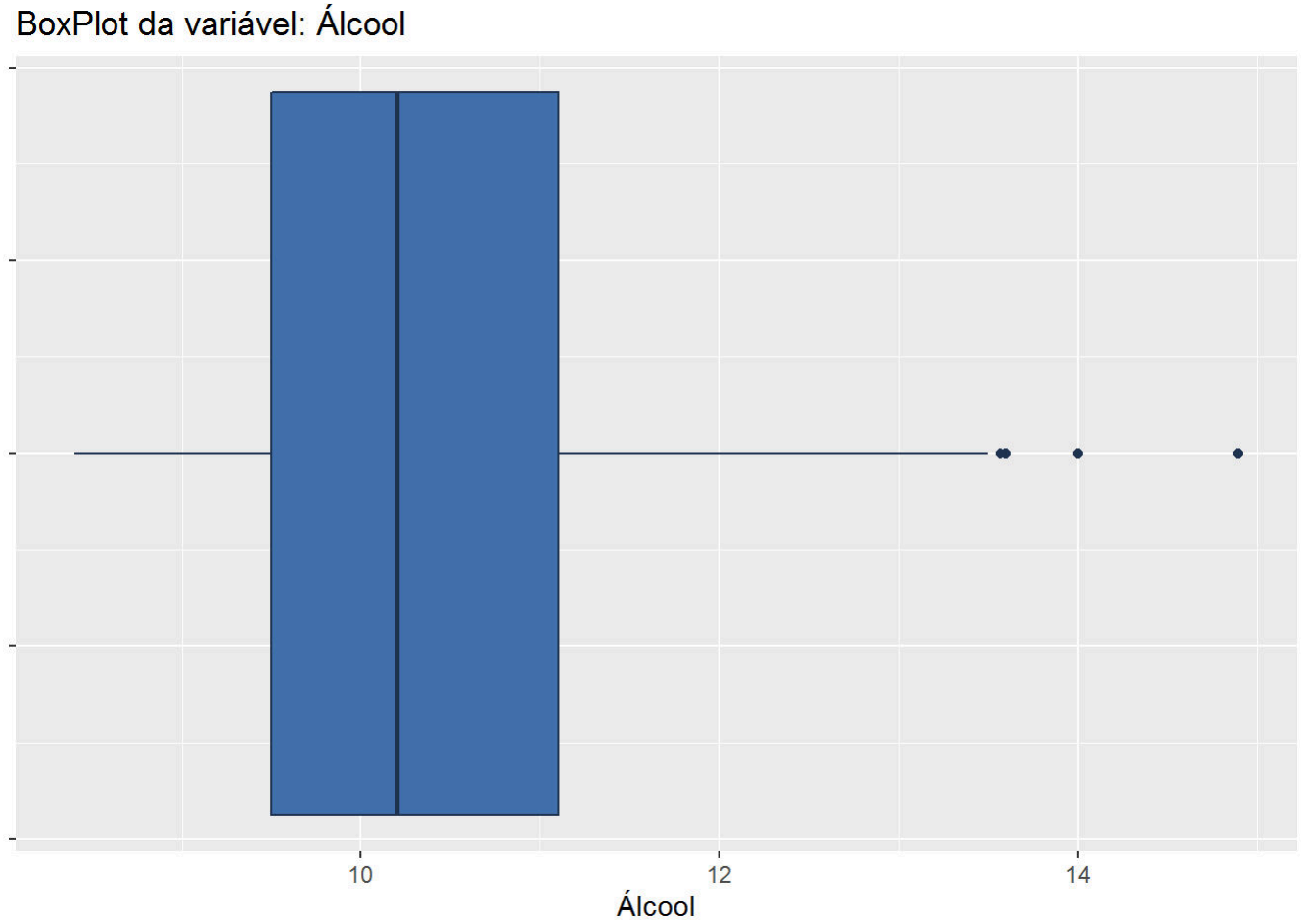
Álcool

```
dist_plot(df_train, 'Álcool')
```



O gráfico mostra que os níveis de álcool nos vinhos possuem uma boa distribuição, apenas com um pico maior no nível de valor 8 aproximadamente.

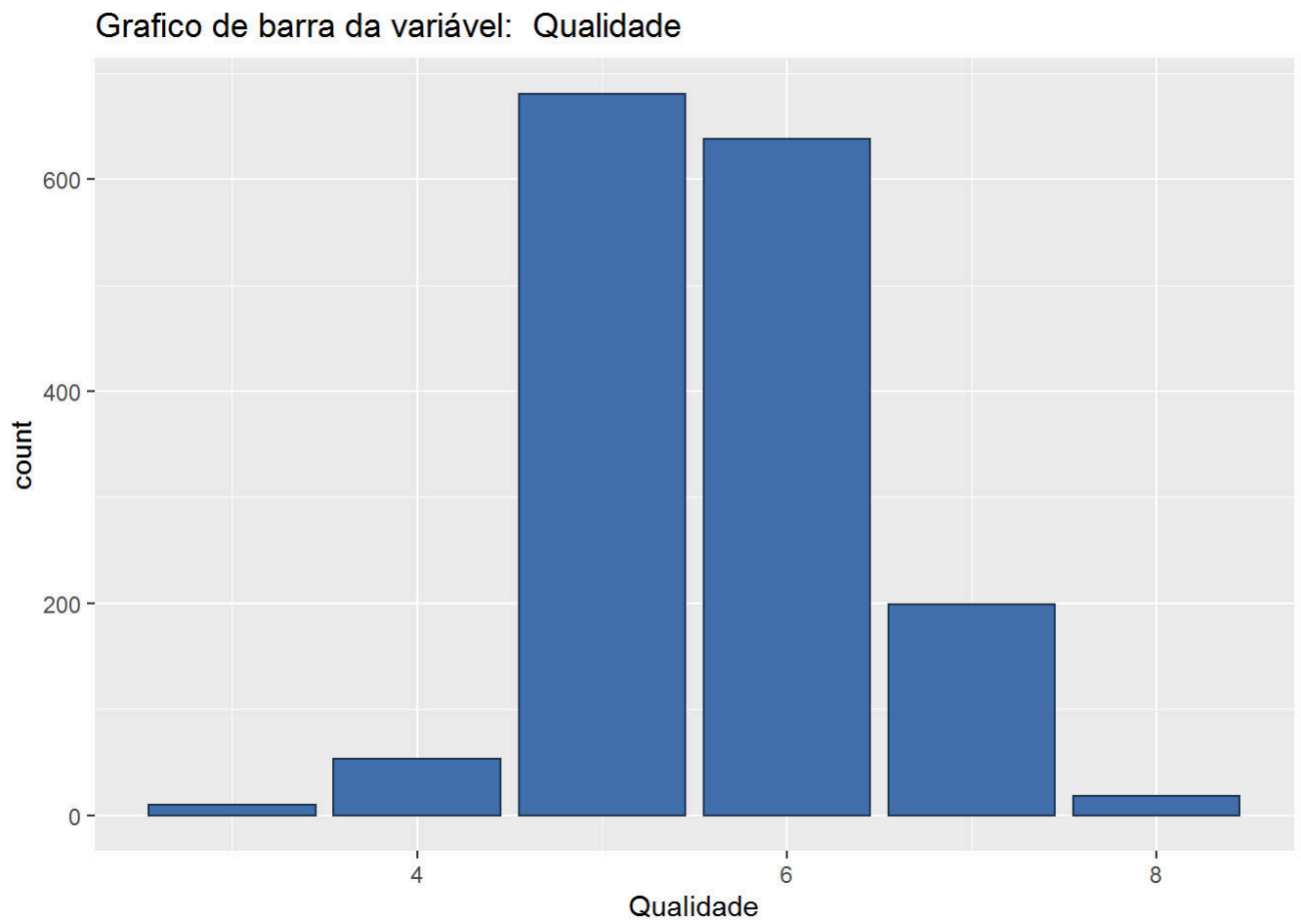
```
box_plot(df_train, 'Álcool')
```



O gráfico mostra que os níveis de álcool nos vinhos possuem pouquíssimos outliers, isso se dá pela grande destruição vista no gráfico anterior, indicando que os níveis de álcool nos vinhos não possuem tanta diferença entre um vinho e outro apenas com algumas exceções.

Qualidade

```
bar_plot(df_train, 'Qualidade')
```



Podemos ver que os dados realmente não estão balanceados, temos muitos mais vinhos com qualidade nível 5 e 6 do que os demais, indicando que a maioria dos vinhos estão em um nível médio para bom.

Gráfico com o nível Max e Min de cada componente do vinho agrupado pela sua qualidade.

Acidez_fixa

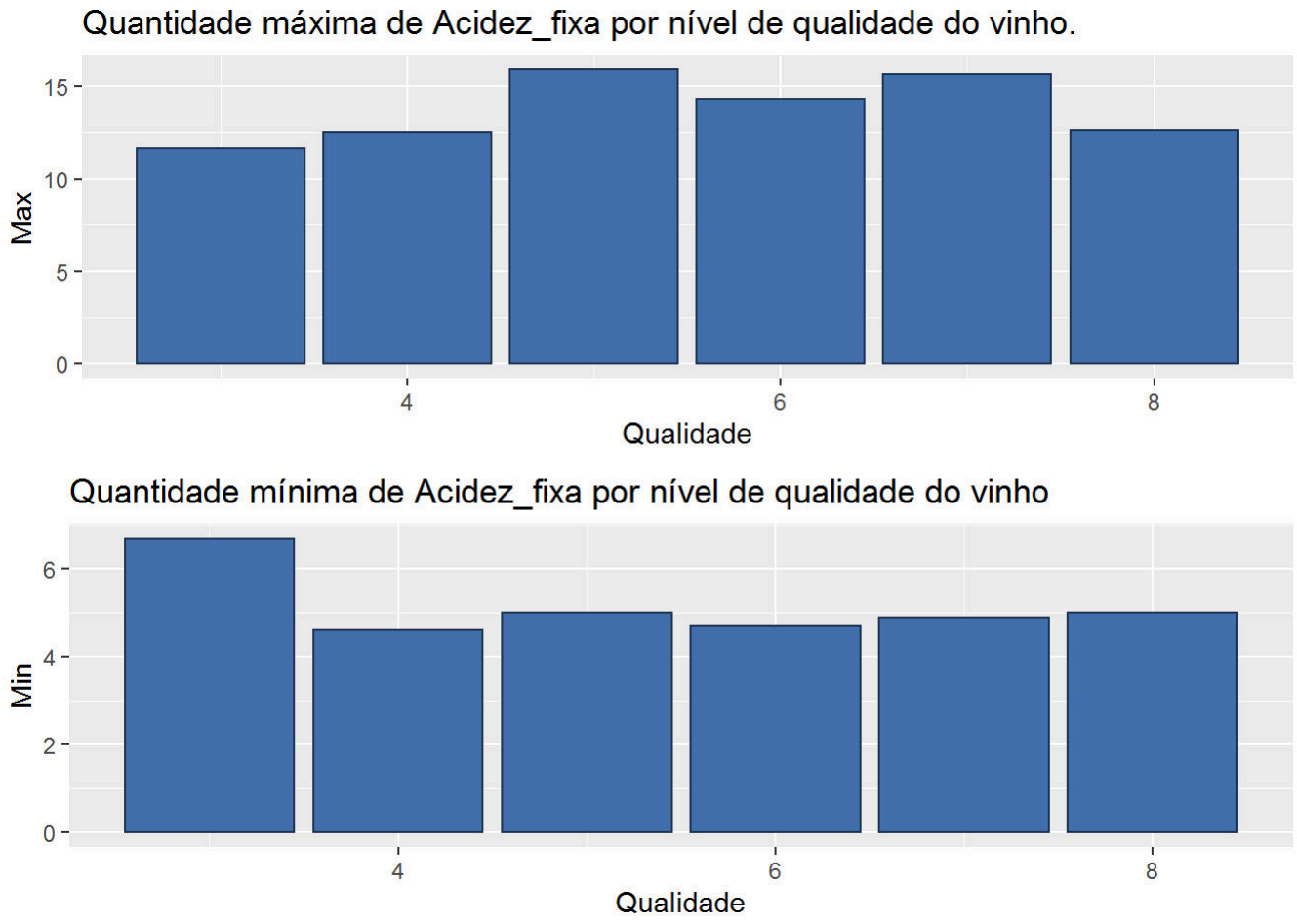
```
plot1 <- df_train %>%
        select(Acidez_fixa,Qualidade)%>%
        group_by(Qualidade)%>%
        summarise(Max= max(Acidez_fixa))%>%
        ggplot() +
        geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271A
E") +
        labs( title = paste('Quantidade máxima de Acidez_fixa por nível de qualidade do vinho.'
))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Acidez_fixa,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Acidez_fixa))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Acidez_fixa por nível de qualidade do vinho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```



O gráfico mostra uma que para um vinho ser considerado ótimo o nível de acidez fixa não pode ser nem muito alto nem muito baixo, considerando o vinho de nível 8 o melhor nível de acidez gira em torno de 4 a 10 aproximadamente.

Acidez_volátil

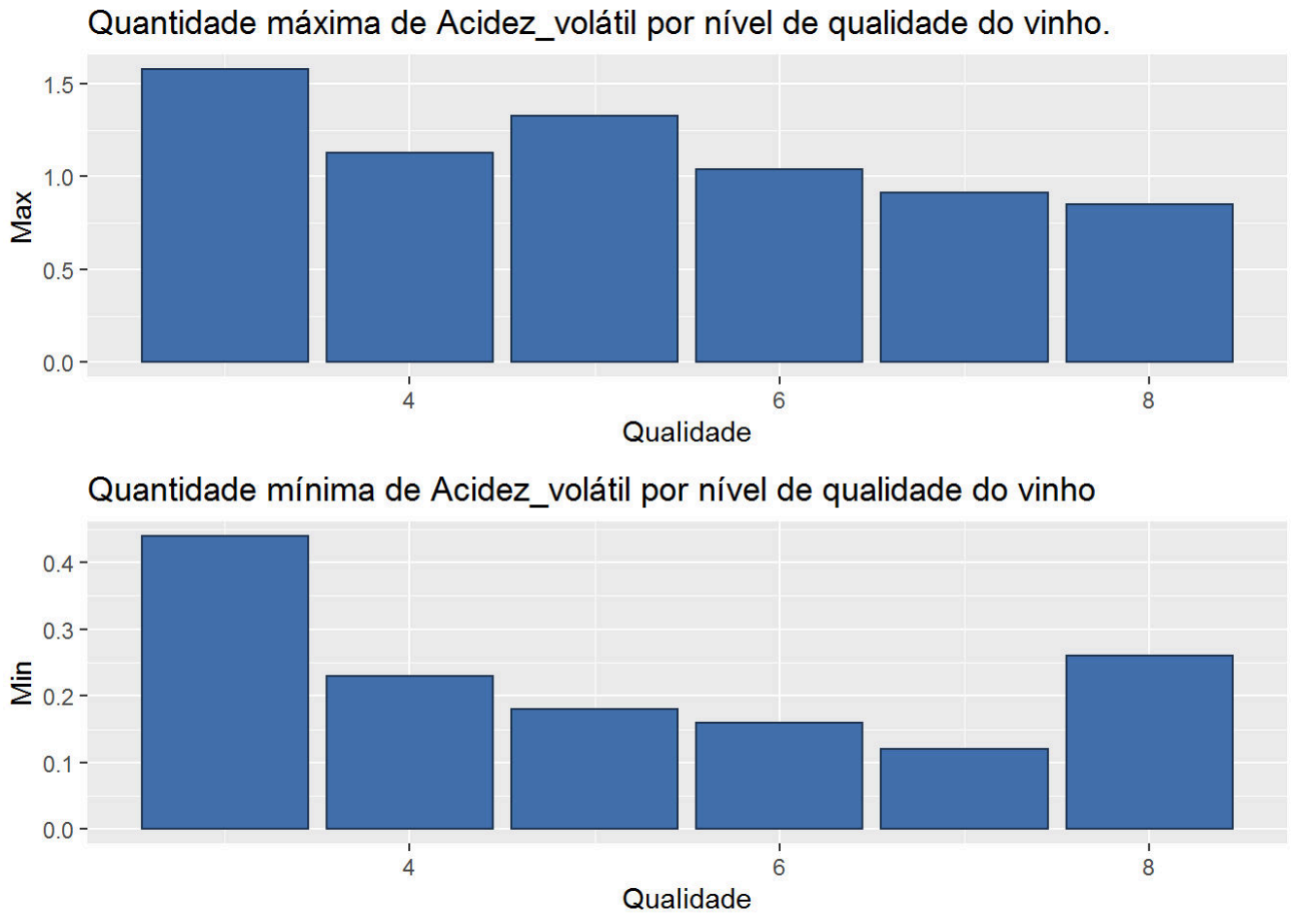
```
plot1 <- df_train %>%
  select(Acidez_volátil,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(Acidez_volátil))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de Acidez_volátil por nível de qualidade do vinho.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Acidez_volátil,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Acidez_volátil))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Acidez_volátil por nível de qualidade do vinho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```



O gráfico mostra assim como a acidez fixa para um vinho ser considerado ótimo o nível de acidez volátil não pode ser nem muito alto nem muito baixo, considerando o vinho de nível 8 o melhor nível de acidez volátil gira em torno de 0.2 a 1 aproximadamente.

Ácido_cítrico

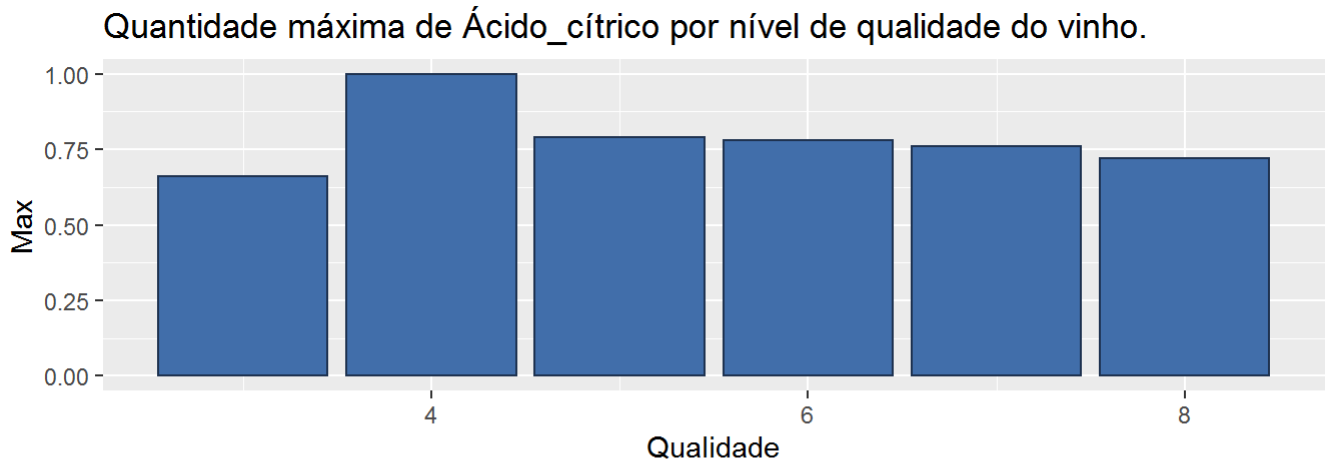
```
plot1 <- df_train %>%
  select(Ácido_cítrico,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(Ácido_cítrico))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de Ácido_cítrico por nível de qualidade do vinho.'))
```

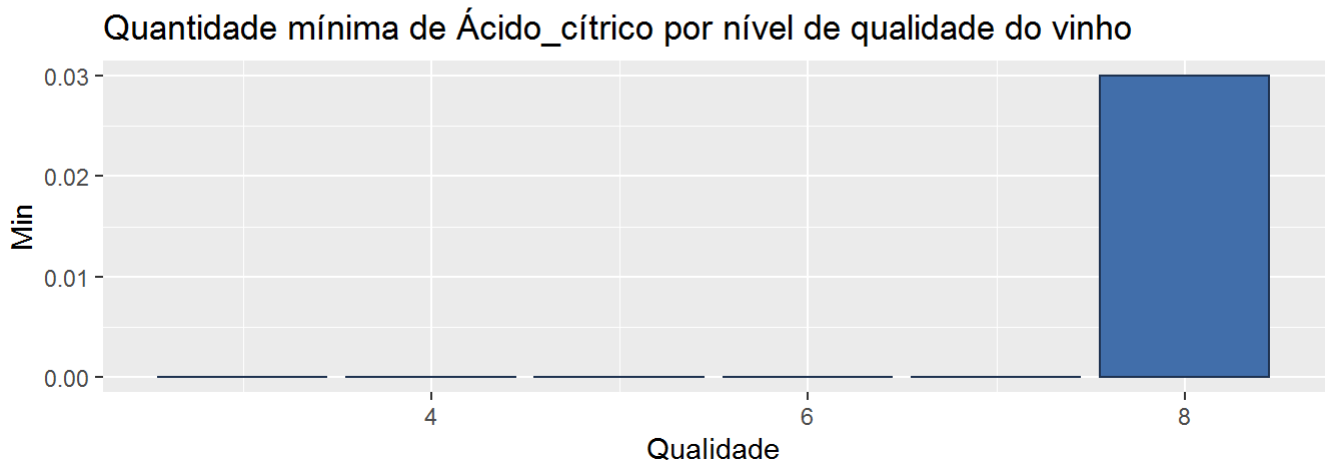
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Ácido_cítrico,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Ácido_cítrico))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Ácido_cítrico por nível de qualidade do vinho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```





O gráfico que vinhos de qualidade 7 e inferior praticamente não possuem nível mínimo de Ácido critico, e o melhor vinho de novel 8 está entre o que tem menor nível máximo.

Açúcar_residual

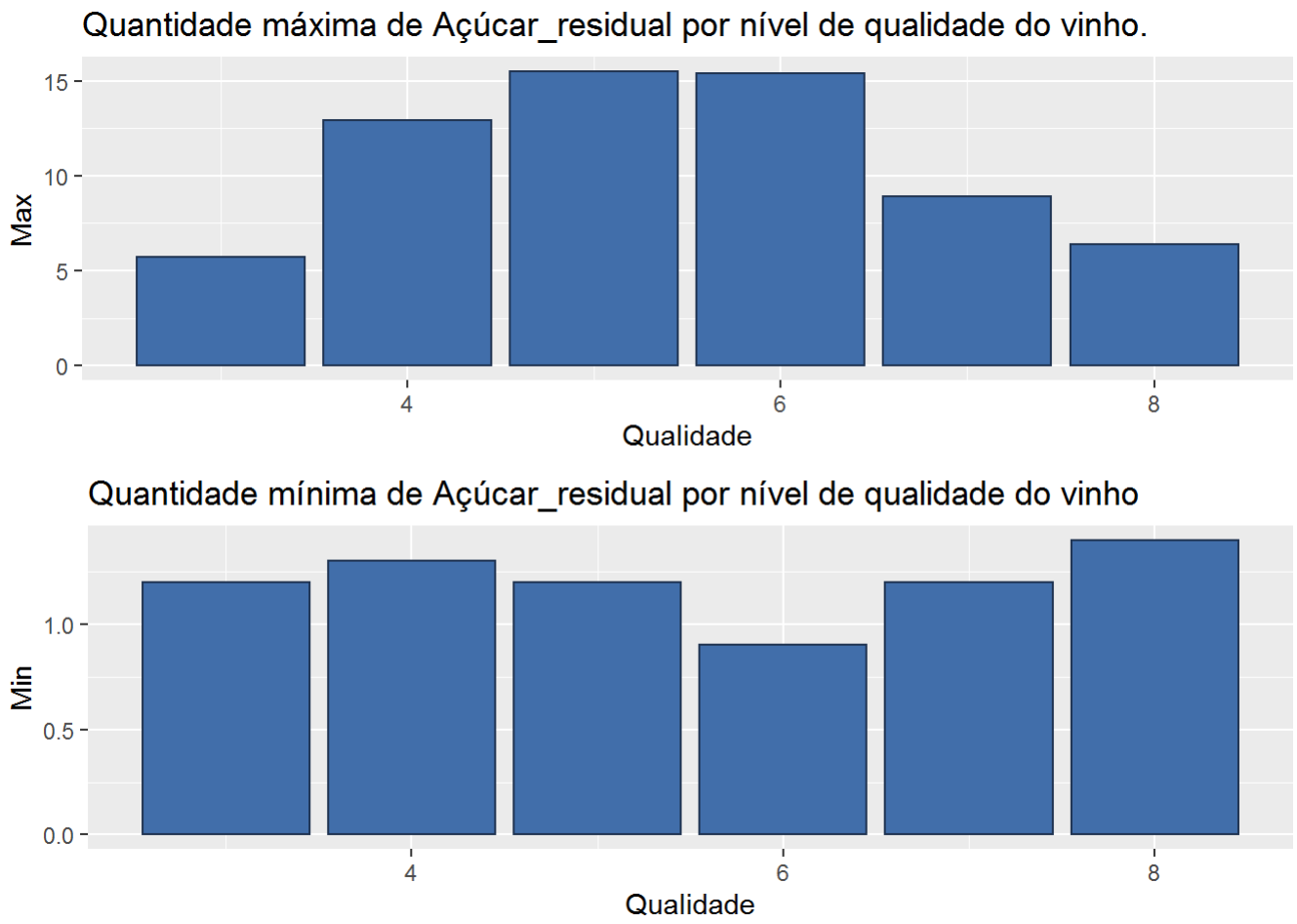
```
plot1 <- df_train %>%
  select(Açúcar_residual,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(Açúcar_residual))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de Açúcar_residual por nível de qualidade do vinho.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Açúcar_residual,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Açúcar_residual))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Açúcar_residual por nível de qualidade do vinho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```



O gráfico mostra que a quantidade de açúcar residual para um nível considerado de qualidade ótima deve estar entre 1 e 5 aproximadamente.

Cloretos

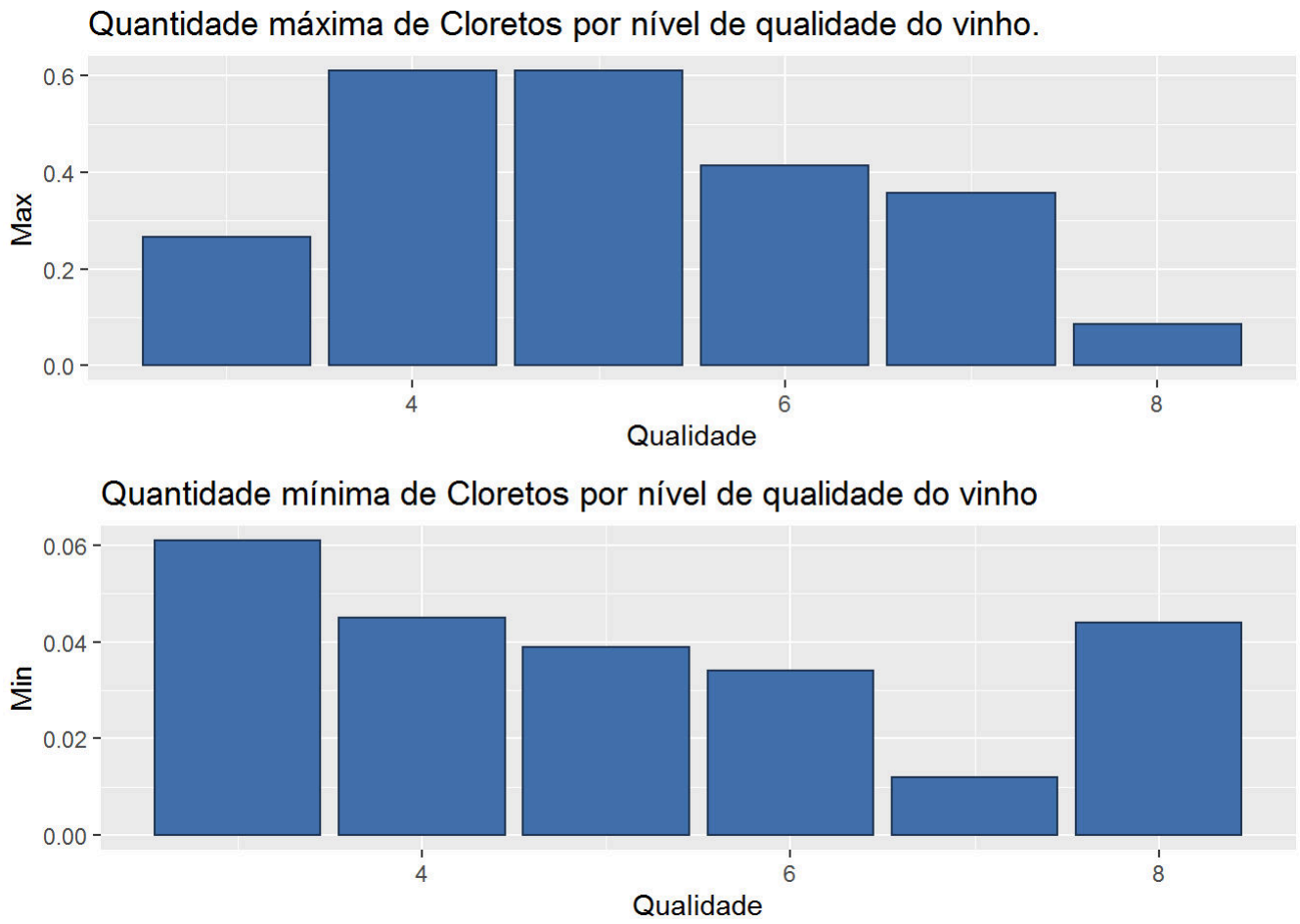
```
plot1 <- df_train %>%
  select(Cloretos,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(Cloretos))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de Cloretos por nível de qualidade do vinho.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Cloretos,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Cloretos))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Cloretos por nível de qualidade do vinho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```



O gráfico mostra que o nível de cloreto entre 0.04 a 0.2 é o nível ideal para um vinho de qualidade ótima, e que uma quantidade grande pode prejudicar na qualidade.

Dióxido_de_enxofre_livre

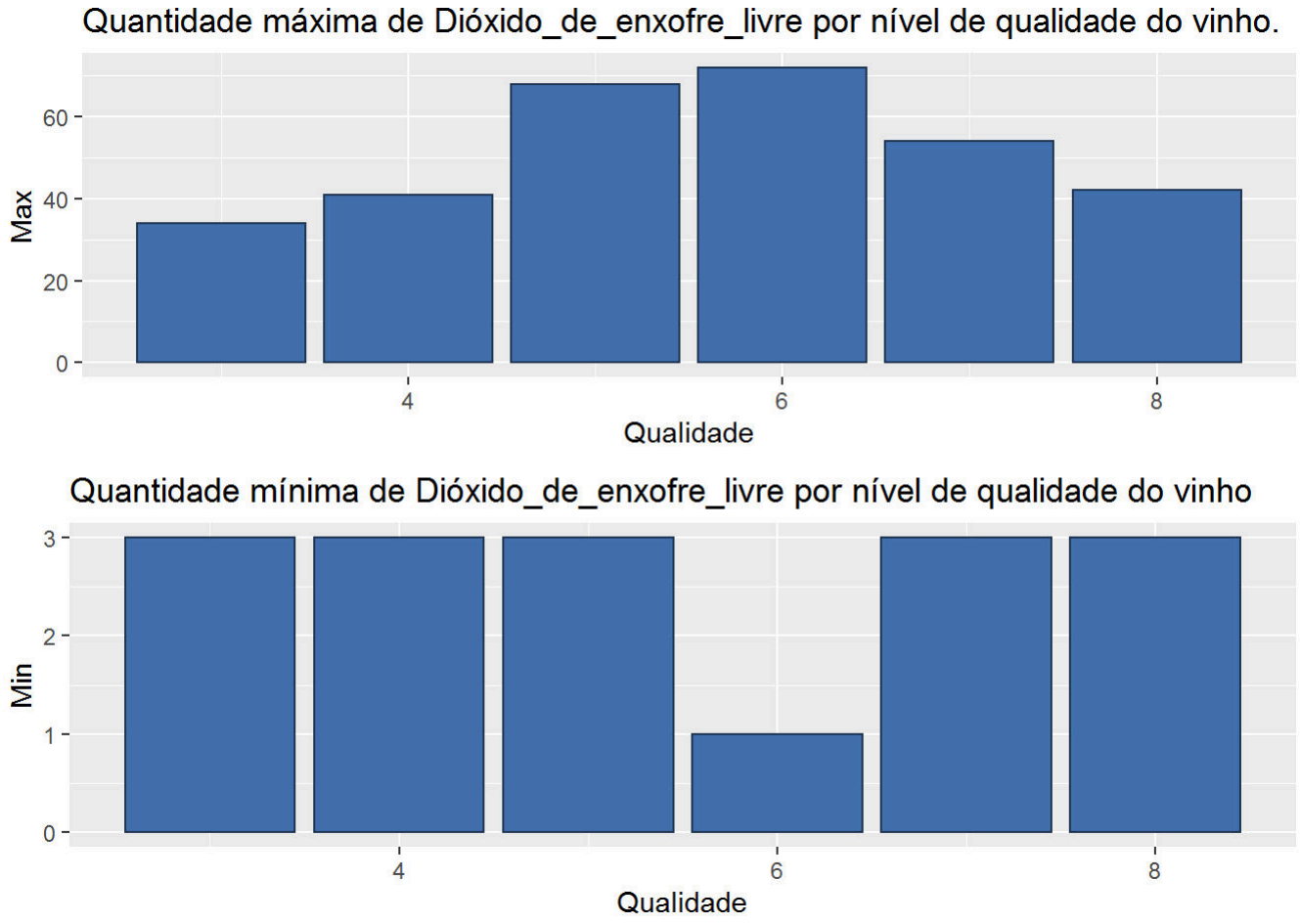
```
plot1 <- df_train %>%
  select(Dióxido_de_enxofre_livre,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(Dióxido_de_enxofre_livre))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de Dióxido_de_enxofre_livre por nível de qualidade do v
inho.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Dióxido_de_enxofre_livre,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Dióxido_de_enxofre_livre))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Dióxido_de_enxofre_livre por nível de qualidade do v
inho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```



O gráfico mostra que o nível de enxofre livre entre 3 a 40 aproximadamente é o nível ideal para um vinho de qualidade ótima, e que uma quantidade grande pode prejudicar na qualidade. Embora mostre que o nível de enxofre livre não é um fator aparentemente tão relevante, considerando que o vinho de pior qualidade tem uma relação de mínimo e máximo bem parecida com o de melhor qualidade.

Dióxido_de_enxofre_total

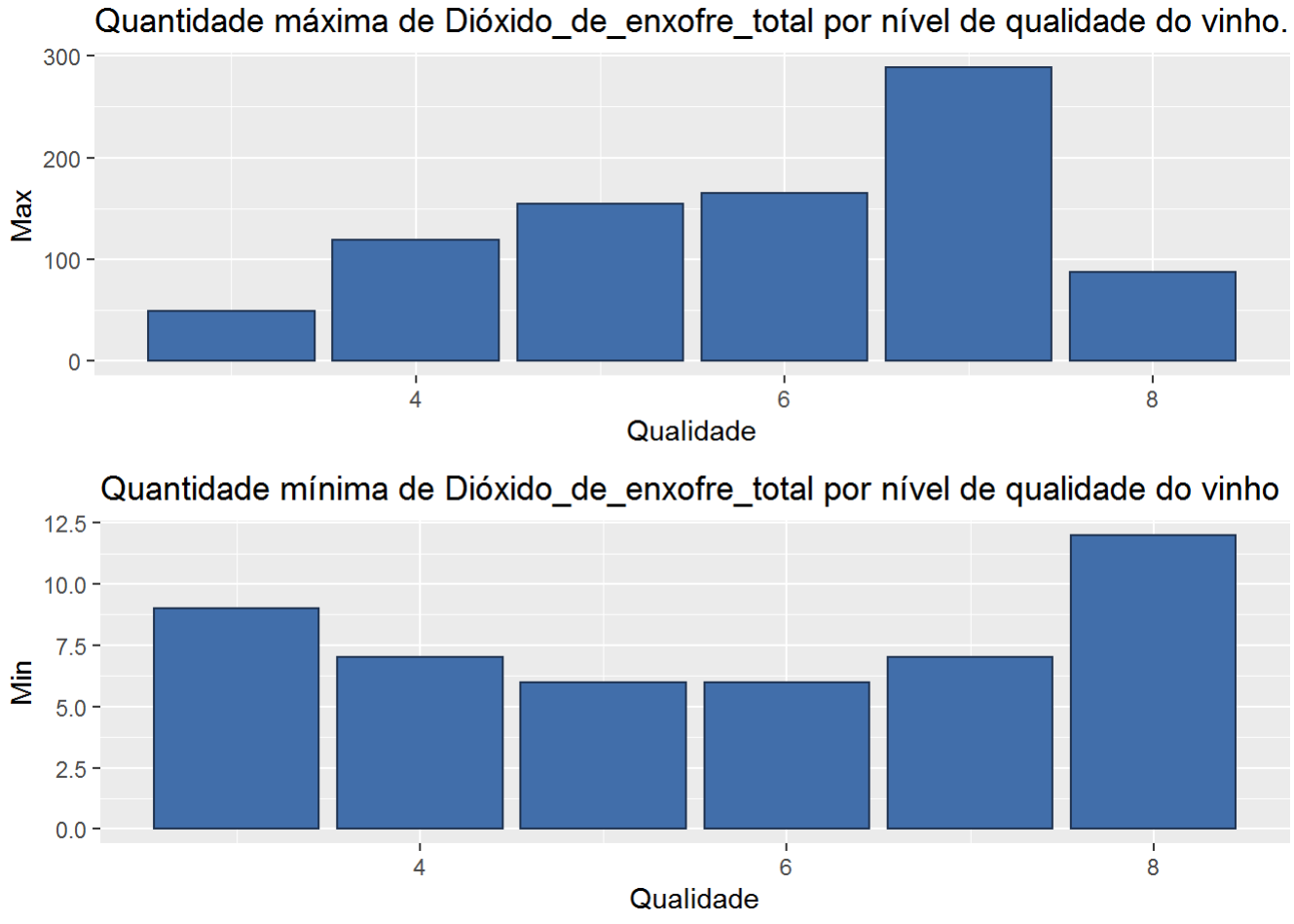
```
plot1 <- df_train %>%
  select(Dióxido_de_enxofre_total,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(Dióxido_de_enxofre_total))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de Dióxido_de_enxofre_total por nível de qualidade do v
inho.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Dióxido_de_enxofre_total,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Dióxido_de_enxofre_total))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Dióxido_de_enxofre_total por nível de qualidade do v
inho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```



O gráfico mostra que o nível de enxofre total entre 12 a 100 aproximadamente é o nível ideal para um vinho de qualidade ótima, assim como o enxofre livre o total não é um fator aparentemente tão relevante, considerando que o vinho de pior qualidade tem uma relação de mínimo e máximo bem parecida com o de melhor qualidade.

Densidade

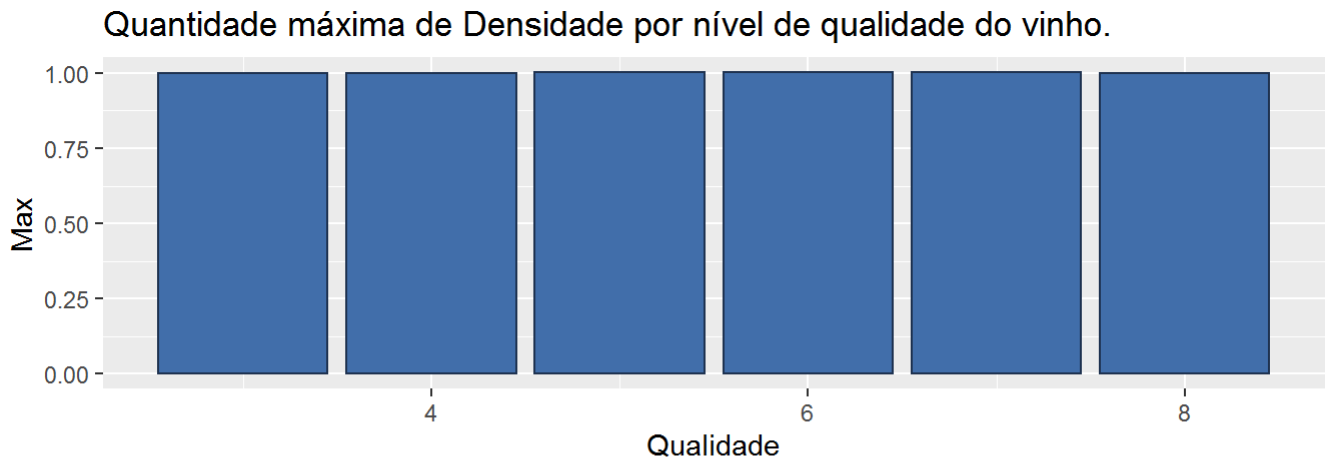
```
plot1 <- df_train %>%
  select(Densidade,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(Densidade))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de Densidade por nível de qualidade do vinho.'))
```

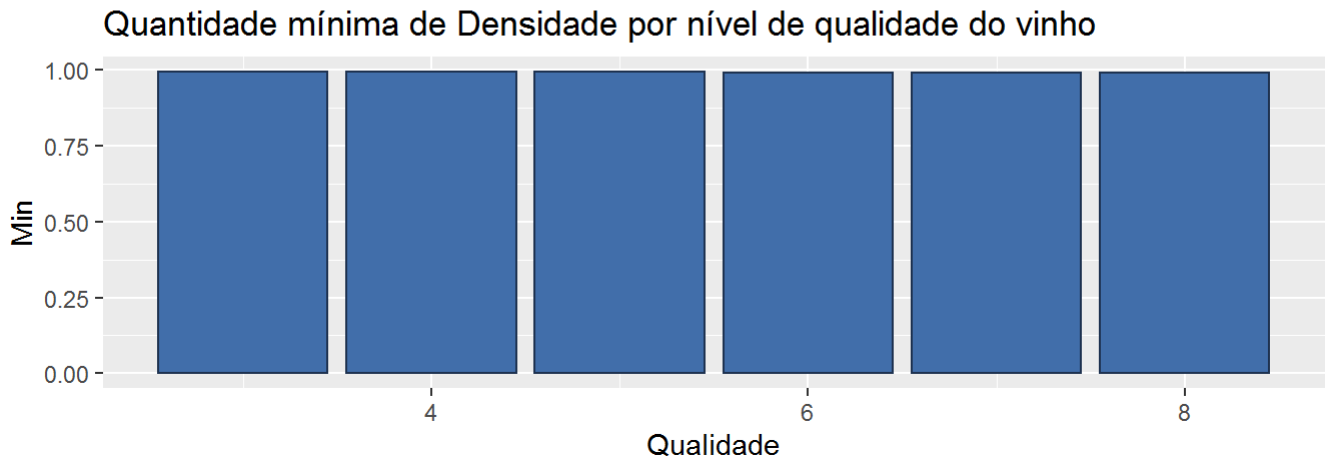
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Densidade,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Densidade))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Densidade por nível de qualidade do vinho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```





O nível máximo e mínimo de densidade é exatamente igual para os níveis de qualidade dos vinhos apresentados

pH

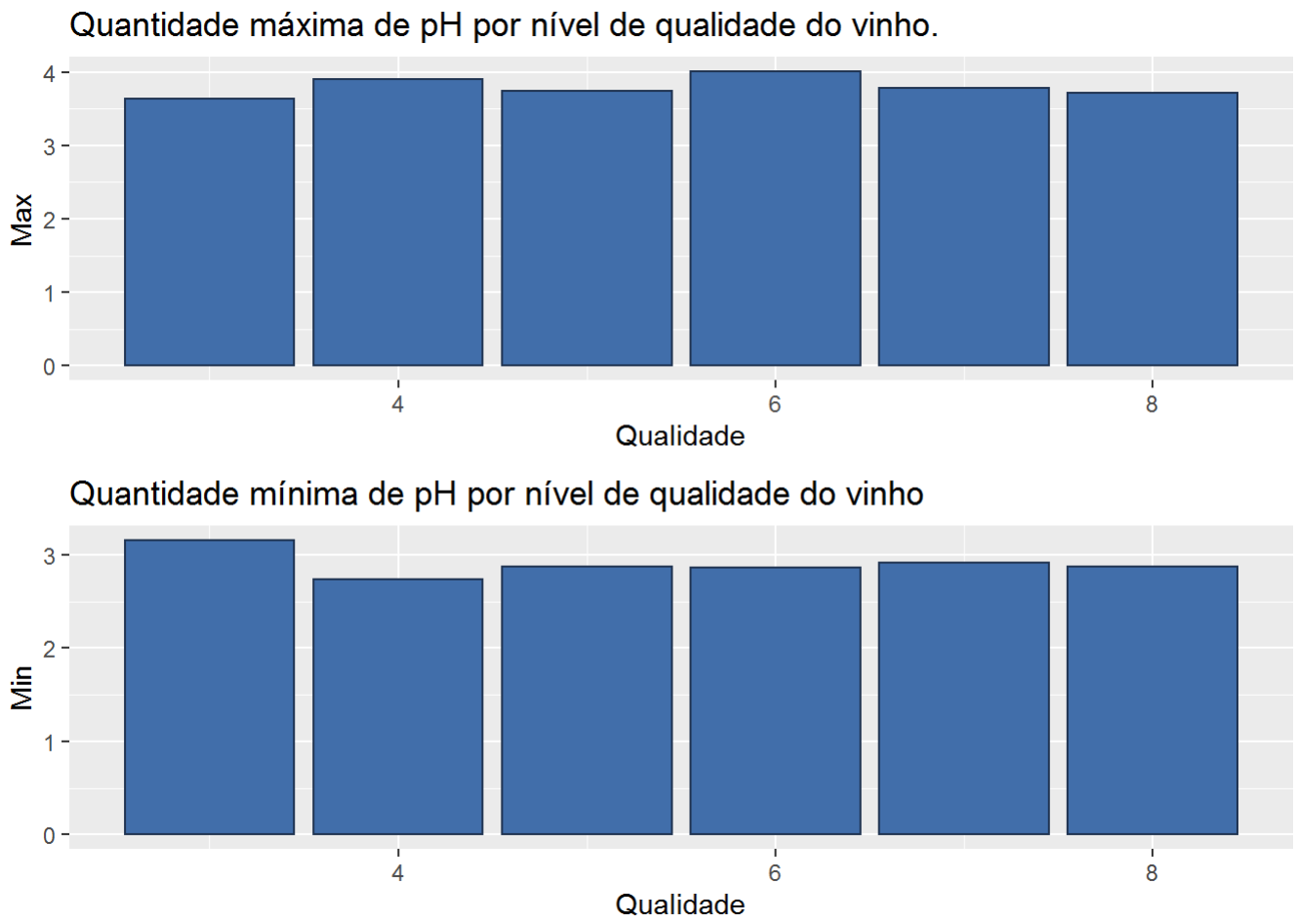
```
plot1 <- df_train %>%
  select(pH,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(pH))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de pH por nível de qualidade do vinho.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(pH,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(pH))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de pH por nível de qualidade do vinho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```



O nível máximo e mínimo de pH é praticamente igual para os níveis de qualidade dos vinhos apresentados.

Sulfatos

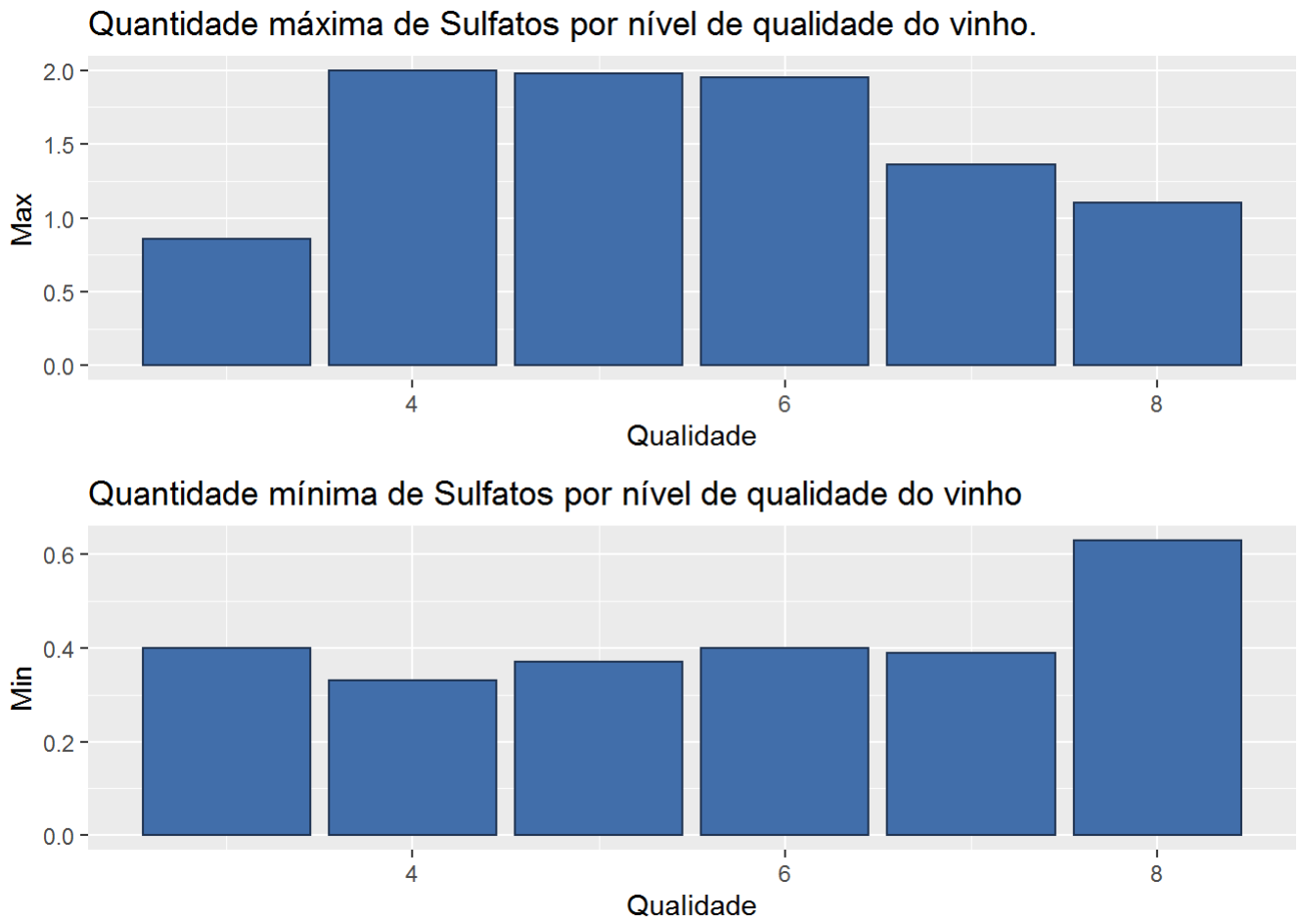
```
plot1 <- df_train %>%
  select(Sulfatos,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(Sulfatos))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de Sulfatos por nível de qualidade do vinho.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Sulfatos,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Sulfatos))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Sulfatos por nível de qualidade do vinho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```



O gráfico mostra um nível de sulfato ideal de sulfato para um vinho de melhor qualidade entre 0.6 a 1.0 aproximadamente, e que um nível muito alto de sulfato pode interferir negativamente na sua qualidade.

Álcool

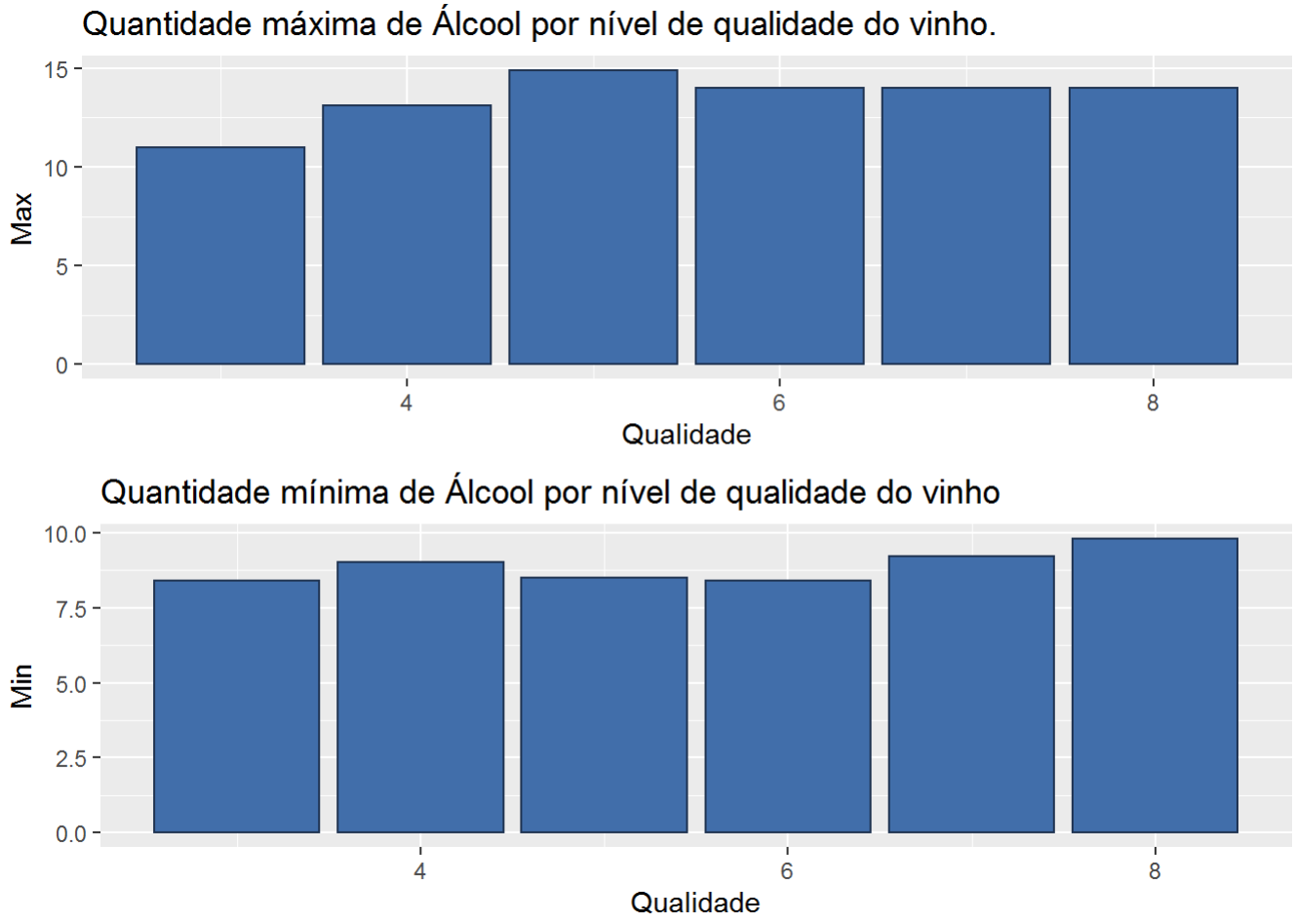
```
plot1 <- df_train %>%
  select(Álcool,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Max= max(Álcool))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Max),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade máxima de Álcool por nível de qualidade do vinho.'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
plot2 <- df_train %>%
  select(Álcool,Qualidade)%>%
  group_by(Qualidade)%>%
  summarise(Min= min(Álcool))%>%
  ggplot() +
  geom_bar(aes (x = Qualidade, y= Min),stat = "identity",color = "#1F3552", fill = "#4271AE") +
  labs( title = paste('Quantidade mínima de Álcool por nível de qualidade do vinho'))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
grid.arrange(plot1, plot2)
```

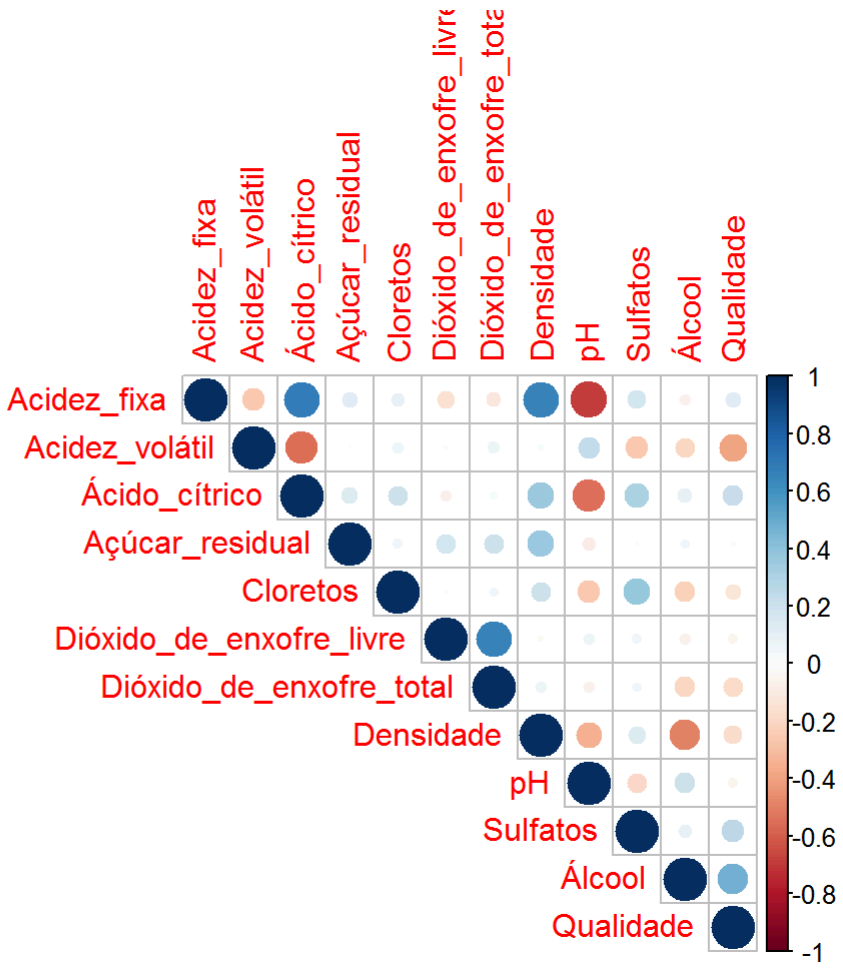


O gráfico mostra que vinhos de melhores qualidades tende a ter um teor de álcool maior geralmente entre 10 a 15. Já os vinhos de pior qualidade o contrário.

Correlação das variáveis

Faço um grafico de correlação com o corrplot

```
corelacao <- cor(df_train, method = )
corrplot(corelacao, type = "upper")
```



O gráfico de correlação mostra que as variáveis acidez volátil, acidez cítrico, álcool, dióxido de enxofre total e densidade são as com melhores correlação com a variável target qualidade.

Tratamento dos dados

Converterei a target que são 6 valores para 3 onde 3,4 valerão 1 com label ruim, 4,5 valerão 5 com label médio, e 7,8 valerão 10 com label excelente.

```
# Crio a função que fara a conversão
cria_label <- function(data){
  if (data == 3 || data == 4){
    data = 1
  }
  else if  (data == 5 || data == 6){
    data = 5
  }
  else if  (data == 7 || data == 8){
    data = 10
  }
}
```

```
# Crio a nova target para a Qualidade
df_train$Qualidade <- sapply(df_train$Qualidade, cria_label)
# Converto a variável targe para factor.
df_train$Qualidade <- as.factor(df_train$Qualidade)
glimpse(df_train)
```

```
## Rows: 1,599
## Columns: 12
## $ Acidez_fixa              <dbl> 7.4, 7.8, 7.8, 11.2, 7.4, 7.4, 7.9, 7.3, 7...
## $ Acidez_volátil           <dbl> 0.700, 0.880, 0.760, 0.280, 0.700, 0.660, ...
## $ Ácido_cítrico            <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.06, ...
## $ Açúcar_residual          <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2, 2....
## $ Cloretos                 <dbl> 0.076, 0.098, 0.092, 0.075, 0.076, 0.075, ...
## $ Dióxido_de_enxofre_livre <dbl> 11, 25, 15, 17, 11, 13, 15, 15, 9, 17, 15,...
## $ Dióxido_de_enxofre_total <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 102, 6...
## $ Densidade                <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978, 0....
## $ pH                       <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.30, ...
## $ Sulfatos                 <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.46, ...
## $ Álcool                   <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, 10.0, 9...
## $ Qualidade                <fct> 5, 5, 5, 5, 5, 5, 5, 10, 10, 5, 5, 5, 5, 5...
```

```
# Separo as variáveis preditoras da target.
features_train <- df_train[1:11]
Qualidade <- as.factor(df_train$Qualidade)
head(features_train)
```

```
##   Acidez_fixa Acidez_volátil Ácido_cítrico Açúcar_residual Cloretos
## 1         7.4           0.70          0.00             1.9    0.076
## 2         7.8           0.88          0.00             2.6    0.098
## 3         7.8           0.76          0.04             2.3    0.092
## 4        11.2           0.28          0.56             1.9    0.075
## 5         7.4           0.70          0.00             1.9    0.076
## 6         7.4           0.66          0.00             1.8    0.075
##   Dióxido_de_enxofre_livre Dióxido_de_enxofre_total Densidade   pH Sulfatos
## 1                       11                       34    0.9978 3.51     0.56
## 2                       25                       67    0.9968 3.20     0.68
## 3                       15                       54    0.9970 3.26     0.65
## 4                       17                       60    0.9980 3.16     0.58
## 5                       11                       34    0.9978 3.51     0.56
## 6                       13                       40    0.9978 3.51     0.56
##   Álcool
## 1    9.4
## 2    9.8
## 3    9.8
## 4    9.8
## 5    9.4
## 6    9.4
```

```
head(Qualidade)
```

```
## [1] 5 5 5 5 5 5
## Levels: 1 5 10
```

Normalização dos dados

As variáveis preditoras estão em escalas diferentes e isso para os modelos não é interessante, irei usar a função min max scaler e a scaler para criar dois tipos de dados para fazer a seleção das variáveis e treinar os modelos.

```
# Min Max Scaler
normalize = function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

features_trainMM <- sapply(features_train, normalize)
head(features_trainMM)
```

```
##      Acidez_fixa Acidez_volátil Ácido_cítrico Açúcar_residual  Cloretos
## [1,]   0.2477876      0.3972603          0.00      0.06849315 0.1068447
## [2,]   0.2831858      0.5205479          0.00      0.11643836 0.1435726
## [3,]   0.2831858      0.4383562          0.04      0.09589041 0.1335559
## [4,]   0.5840708      0.1095890          0.56      0.06849315 0.1051753
## [5,]   0.2477876      0.3972603          0.00      0.06849315 0.1068447
## [6,]   0.2477876      0.3698630          0.00      0.06164384 0.1051753
##      Dióxido_de_enxofre_livre Dióxido_de_enxofre_total Densidade        pH
## [1,]                0.1408451              0.09893993 0.5675477 0.6062992
## [2,]                0.3380282              0.21554770 0.4941263 0.3622047
## [3,]                0.1971831              0.16961131 0.5088106 0.4094488
## [4,]                0.2253521              0.19081272 0.5822320 0.3307087
## [5,]                0.1408451              0.09893993 0.5675477 0.6062992
## [6,]                0.1690141              0.12014134 0.5675477 0.6062992
##       Sulfatos    Álcool
## [1,] 0.1377246 0.1538462
## [2,] 0.2095808 0.2153846
## [3,] 0.1916168 0.2153846
## [4,] 0.1497006 0.2153846
## [5,] 0.1377246 0.1538462
## [6,] 0.1377246 0.1538462
```

```
# scale

features_trainS <- sapply(features_train, scale)
head(features_trainS)
```

```
##      Acidez_fixa Acidez_volátil Ácido_cítrico Açúcar_residual    Cloretos
## [1,]  -0.5281944      0.9615758     -1.391037     -0.45307667 -0.24363047
## [2,]  -0.2984541      1.9668271     -1.391037      0.04340257  0.22380518
## [3,]  -0.2984541      1.2966596     -1.185699     -0.16937425  0.09632273
## [4,]   1.6543385     -1.3840105      1.483689     -0.45307667 -0.26487754
## [5,]  -0.5281944      0.9615758     -1.391037     -0.45307667 -0.24363047
## [6,]  -0.5281944      0.7381867     -1.391037     -0.52400227 -0.26487754
##      Dióxido_de_enxofre_livre Dióxido_de_enxofre_total  Densidade         pH
## [1,]              -0.46604672              -0.3790141 0.55809987  1.2882399
## [2,]               0.87236532               0.6241680 0.02825193 -0.7197081
## [3,]              -0.08364328               0.2289750 0.13422152 -0.3310730
## [4,]               0.10755844               0.4113718 0.66406945 -0.9787982
## [5,]              -0.46604672              -0.3790141 0.55809987  1.2882399
## [6,]              -0.27484500              -0.1966174 0.55809987  1.2882399
##         Sulfatos     Álcool
## [1,] -0.57902538 -0.9599458
## [2,]  0.12891007 -0.5845942
## [3,] -0.04807379 -0.5845942
## [4,] -0.46103614 -0.5845942
## [5,] -0.57902538 -0.9599458
## [6,] -0.57902538 -0.9599458
```

```
## Crio um Data frame para cada dado nromalizado com a variavel targe.

# Dados Min Max Scaler
df_trainMM <- as.data.frame(features_trainMM)
df_trainMM['Qualidade'] <- Qualidade
head(df_trainMM)
```

```
##   Acidez_fixa Acidez_volátil Ácido_cítrico Açúcar_residual  Cloretos
## 1   0.2477876      0.3972603          0.00      0.06849315 0.1068447
## 2   0.2831858      0.5205479          0.00      0.11643836 0.1435726
## 3   0.2831858      0.4383562          0.04      0.09589041 0.1335559
## 4   0.5840708      0.1095890          0.56      0.06849315 0.1051753
## 5   0.2477876      0.3972603          0.00      0.06849315 0.1068447
## 6   0.2477876      0.3698630          0.00      0.06164384 0.1051753
##   Dióxido_de_enxofre_livre Dióxido_de_enxofre_total Densidade        pH
## 1                0.1408451               0.09893993 0.5675477 0.6062992
## 2                0.3380282               0.21554770 0.4941263 0.3622047
## 3                0.1971831               0.16961131 0.5088106 0.4094488
## 4                0.2253521               0.19081272 0.5822320 0.3307087
## 5                0.1408451               0.09893993 0.5675477 0.6062992
## 6                0.1690141               0.12014134 0.5675477 0.6062992
##    Sulfatos    Álcool Qualidade
## 1 0.1377246 0.1538462         5
## 2 0.2095808 0.2153846         5
## 3 0.1916168 0.2153846         5
## 4 0.1497006 0.2153846         5
## 5 0.1377246 0.1538462         5
## 6 0.1377246 0.1538462         5
```

```
# Dados Scaler
df_trainS <- as.data.frame(features_trainS)
df_trainS['Qualidade'] <- Qualidade
head(df_trainS)
```

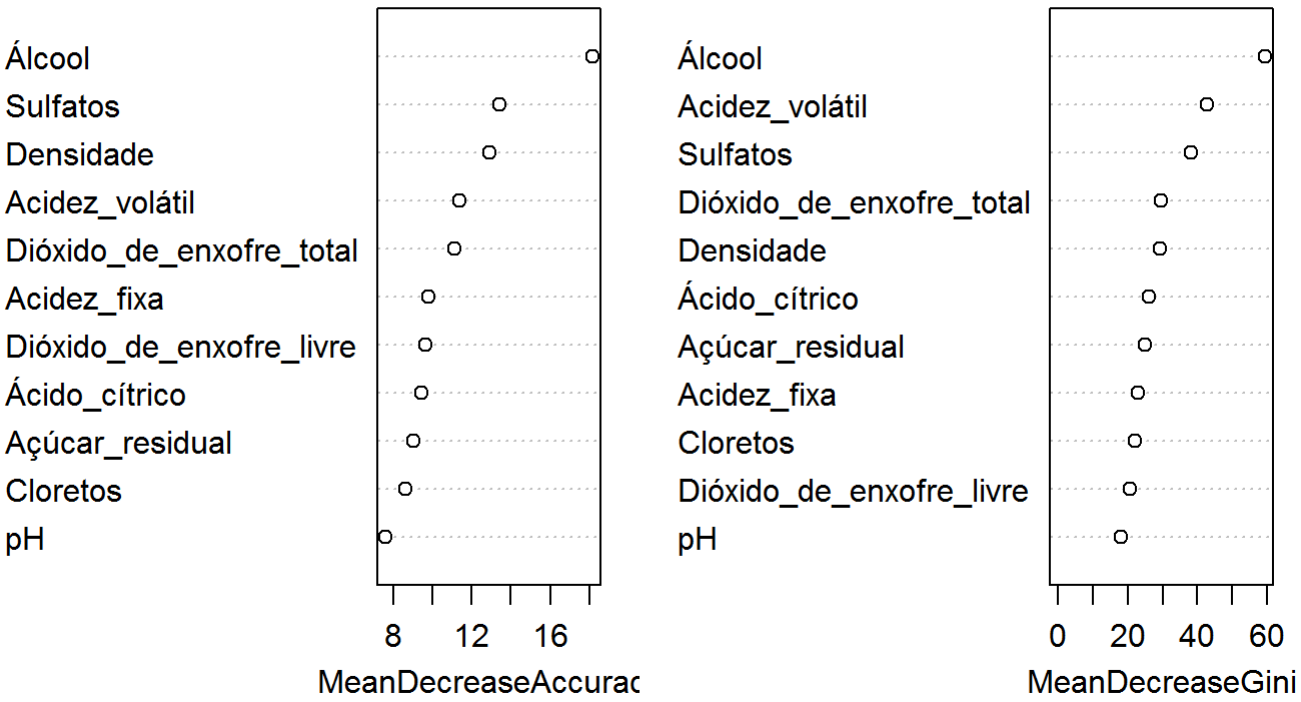
```
##   Acidez_fixa Acidez_volátil Ácido_cítrico Açúcar_residual    Cloretos
## 1  -0.5281944      0.9615758     -1.391037     -0.45307667 -0.24363047
## 2  -0.2984541      1.9668271     -1.391037      0.04340257  0.22380518
## 3  -0.2984541      1.2966596     -1.185699     -0.16937425  0.09632273
## 4   1.6543385     -1.3840105      1.483689     -0.45307667 -0.26487754
## 5  -0.5281944      0.9615758     -1.391037     -0.45307667 -0.24363047
## 6  -0.5281944      0.7381867     -1.391037     -0.52400227 -0.26487754
##   Dióxido_de_enxofre_livre Dióxido_de_enxofre_total  Densidade         pH
## 1              -0.46604672               -0.3790141 0.55809987  1.2882399
## 2               0.87236532                0.6241680 0.02825193 -0.7197081
## 3              -0.08364328                0.2289750 0.13422152 -0.3310730
## 4               0.10755844                0.4113718 0.66406945 -0.9787982
## 5              -0.46604672               -0.3790141 0.55809987  1.2882399
## 6              -0.27484500               -0.1966174 0.55809987  1.2882399
##      Sulfatos     Álcool Qualidade
## 1 -0.57902538 -0.9599458         5
## 2  0.12891007 -0.5845942         5
## 3 -0.04807379 -0.5845942         5
## 4 -0.46103614 -0.5845942         5
## 5 -0.57902538 -0.9599458         5
## 6 -0.57902538 -0.9599458         5
```

Feature Selection (Seleção de Variáveis)

Usarei o modelo randomForest e para seleção das melhores variáveis para cada features normalizada.

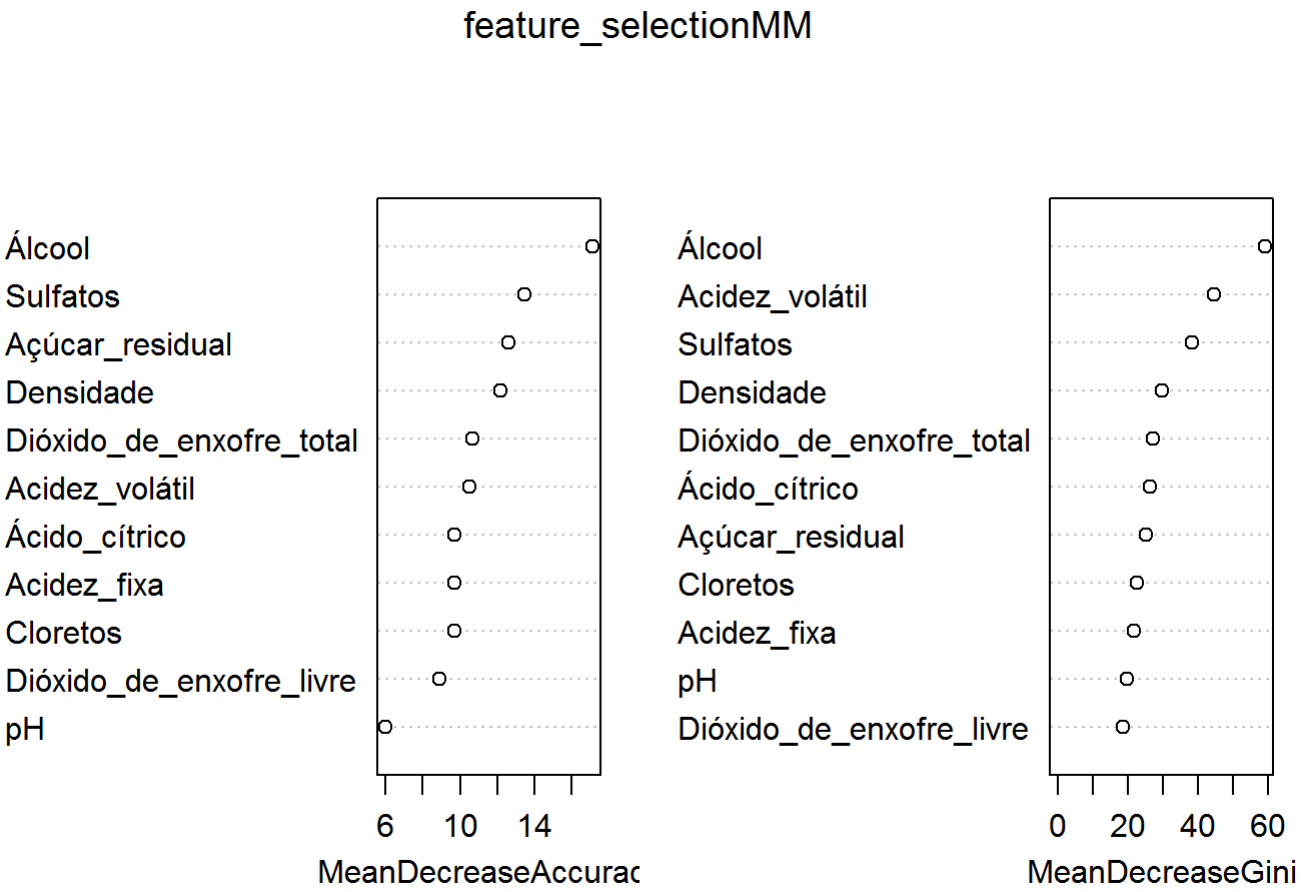
Dados Min Max Scaler

```
#Random Forest Dados
feature_selectionMM <- randomForest(Qualidade ~ .,
                                    data = df_trainMM,
                                    ntree = 100, nodesize = 10, importance = T)
varImpPlot(feature_selectionMM)
```



Dados Scaler

```
#Random Forest Dados

feature_selectionMM <- randomForest(Qualidade ~ .,
                                    data = df_trainS,
                                    ntree = 100, nodesize = 10, importance = T)
varImpPlot(feature_selectionMM)
```




```
# Ambos os modelos e tipos de dados tiveram o mesmo resultado determinando as melhores variáveis.
best_var <- c('Álcool','Acidez_volátil','Sulfatos','Dióxido_de_enxofre_total','Cloretos', 'Densidade', 'Qualidade')
```

```
# Seleciono apenas as variáveis mais importantes e a target em um novo DF para o treinamento dos modelos.
# Faço para os dados no Min Max Scaler
best_df_trainMM <- df_trainMM[best_var]
best_df_trainMM$Qualidade <- as.factor(best_df_trainMM$Qualidade)
# Faço para os dados no Scaler
best_df_trainS <- df_trainS[best_var]
best_df_trainS$Qualidade <- as.factor(best_df_trainS$Qualidade)
```

Split dos dados

Crio os dados de treino e teste em cada escala para o treinamento dos algoritmos.

```
# Min Max Scaler
splitMM = sample.split(best_df_trainMM$Densidade, SplitRatio = 0.80)
trainMM = subset(best_df_trainMM, splitMM == TRUE)
testMM = subset(best_df_trainMM, splitMM == FALSE)


# Scaler
splitS = sample.split(best_df_trainS$Densidade, SplitRatio = 0.80)
trainS = subset(best_df_trainS, splitS == TRUE)
testS = subset(best_df_trainS, splitS == FALSE)
```

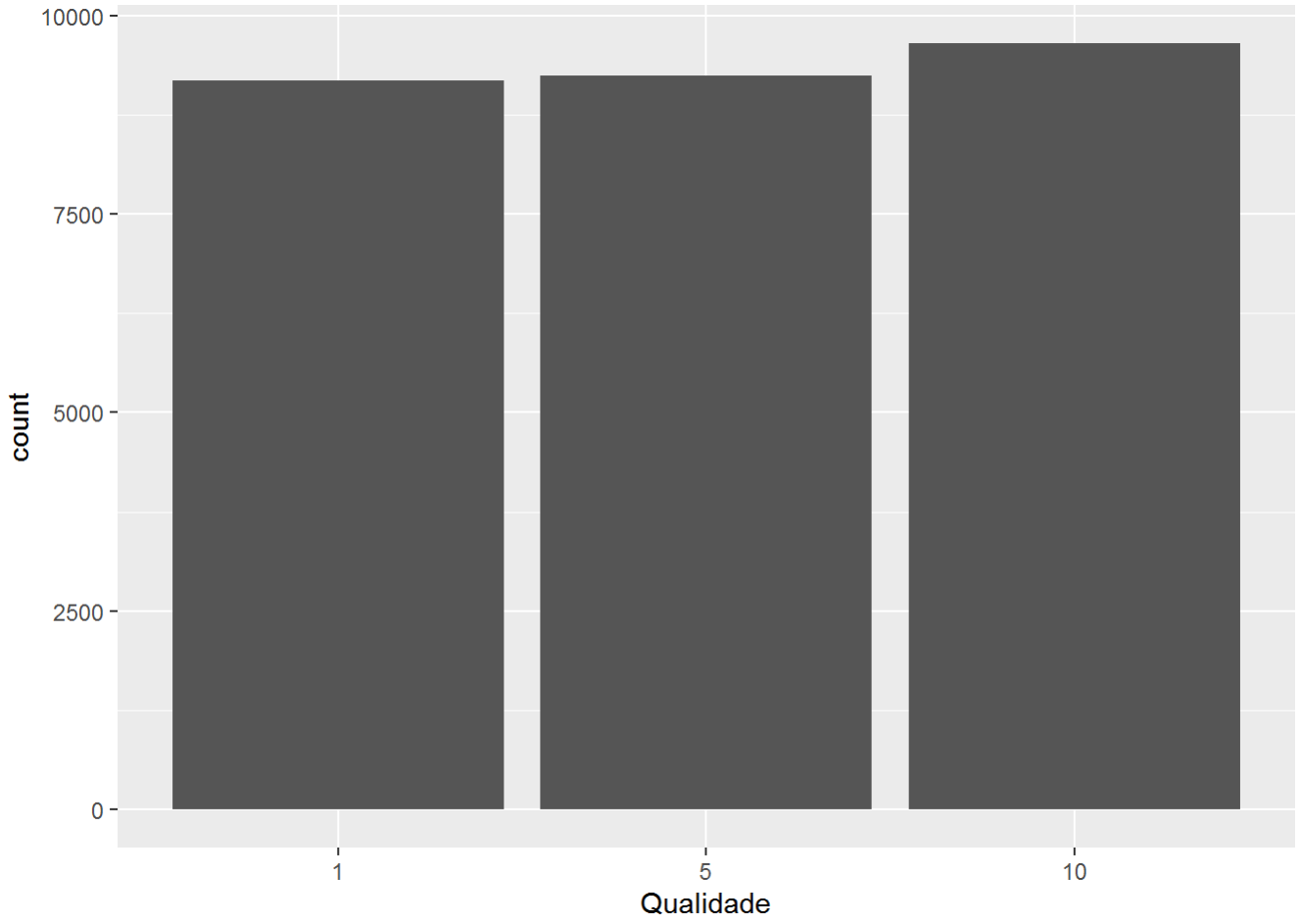
Balanceamento dos dados

Como já vimos temos que balancear a target em cada DF, pois está muito desbalanceada e pode tendenciar os modelos de aprendizagem. Passo duas vezes o smote, pois temos muitos poucos dados para treinar com ele eu aumento a quantidade dos dados e deixo balanceado.

```
# Balanceando  a TrainMM
trainMM <- SMOTE(Qualidade ~ .,data =trainMM,perc.over = 10000, perc.under = 120)
trainMM <- SMOTE(Qualidade ~ .,data =trainMM,perc.over = 1000, perc.under = 210)
table(trainMM$Qualidade)
```

```
##
##    1    5   10
## 9178 9239 9647
```

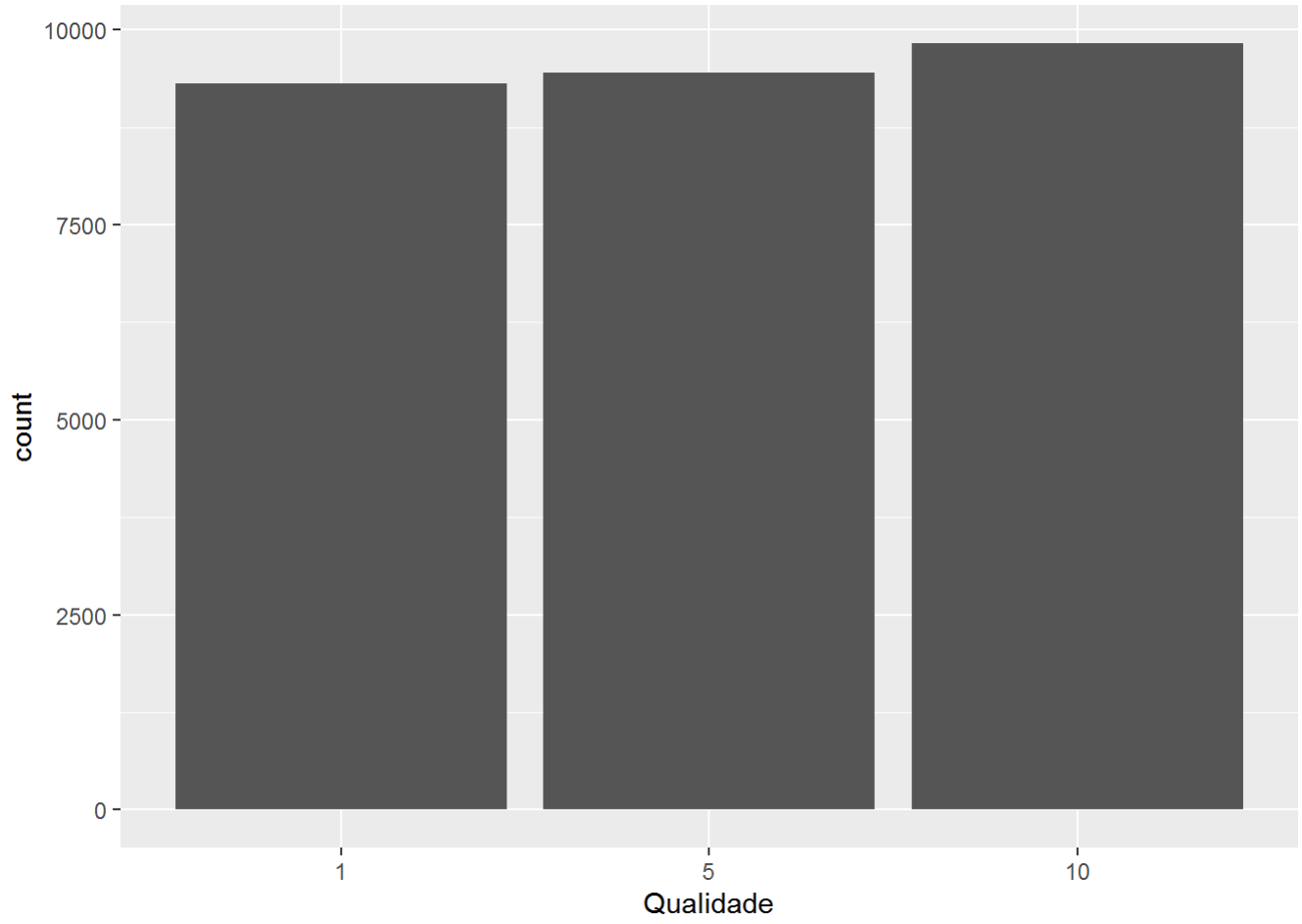
```
ggplot(trainMM, aes(x = Qualidade)) + geom_bar()
```




```
# Balanceando  a TrainS
trainS <- SMOTE(Qualidade ~ .,data =trainS,perc.over = 10000, perc.under = 120)
trainS <- SMOTE(Qualidade ~ .,data =trainS,perc.over = 1000, perc.under = 210)
table(trainS$Qualidade)
```

```
##
##    1    5   10
## 9310 9443 9823
```

```
ggplot(trainS, aes(x = Qualidade)) + geom_bar()
```



Algoritmos de aprendizagem

Dados com o Min Max Scaler

```
# Modelo  com o KSVM  library(kernlab)

modelo_v1 <- ksvm(Qualidade ~ .
                  ,data= trainMM,type="C-bsvc", kernel = "rbfdot")
previsao_v1 <- predict(modelo_v1, testMM)
confusionMatrix(previsao_v1,testMM$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    7  32   2
##         5    3 173  14
##         10   1  21  24
##
## Overall Statistics
##
##                Accuracy : 0.7365
##                  95% CI : (0.6804, 0.7874)
##     No Information Rate : 0.8159
##     P-Value [Acc > NIR] : 0.9996
##
##                   Kappa : 0.358
##
##  Mcnemar's Test P-Value : 1.07e-05
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.63636   0.7655   0.60000
## Specificity           0.87218   0.6667   0.90717
## Pos Pred Value        0.17073   0.9105   0.52174
## Neg Pred Value        0.98305   0.3908   0.93074
## Prevalence            0.03971   0.8159   0.14440
## Detection Rate        0.02527   0.6245   0.08664
## Detection Prevalence  0.14801   0.6859   0.16606
## Balanced Accuracy     0.75427   0.7161   0.75359
```

```
# Modelo  com o RandomForest library('randomForest')
modelo_v2 <- randomForest(Qualidade ~ .
                  ,data= trainMM,ntree = 100,
                  nodesize = 10,method="repeatedcv",
                  number=15, repeats=10)
previsao_v2 <- predict(modelo_v2, testMM)
confusionMatrix(previsao_v2,testMM$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    6  16   0
##         5    5 201  18
##         10   0   9  22
##
## Overall Statistics
##
##                Accuracy : 0.8267
##                  95% CI : (0.7769, 0.8694)
##     No Information Rate : 0.8159
##     P-Value [Acc > NIR] : 0.3544
##
##                   Kappa : 0.46
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.54545   0.8894   0.55000
## Specificity           0.93985   0.5490   0.96203
## Pos Pred Value        0.27273   0.8973   0.70968
## Neg Pred Value        0.98039   0.5283   0.92683
## Prevalence            0.03971   0.8159   0.14440
## Detection Rate        0.02166   0.7256   0.07942
## Detection Prevalence  0.07942   0.8087   0.11191
## Balanced Accuracy     0.74265   0.7192   0.75601
```

```
# Modelo svm do pacote library(e1071)
modelo_v3 <- svm(Qualidade ~ .
                          ,data= trainMM,type = 'C-classification',kernel = 'radial',
                          cost = 10, scale = FALSE,gamma = 0.1)
previsao_v3 <- predict(modelo_v3, testMM)
confusionMatrix(previsao_v3,testMM$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    9  56   2
##         5    2 129   9
##         10   0  41  29
##
## Overall Statistics
##
##                Accuracy : 0.6029
##                  95% CI : (0.5426, 0.6609)
##     No Information Rate : 0.8159
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2667
##
##  Mcnemar's Test P-Value : 1.096e-15
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.81818   0.5708    0.7250
## Specificity           0.78195   0.7843    0.8270
## Pos Pred Value        0.13433   0.9214    0.4143
## Neg Pred Value        0.99048   0.2920    0.9469
## Prevalence            0.03971   0.8159    0.1444
## Detection Rate        0.03249   0.4657    0.1047
## Detection Prevalence  0.24188   0.5054    0.2527
## Balanced Accuracy     0.80007   0.6776    0.7760
```

```
# Modelo   naiveBayes do  pacote library(e1071)
modelo_v4 <- naiveBayes(Qualidade ~ .
                        ,data= trainMM, laplace=3)
previsao_v4 <- predict(modelo_v4, testMM, type="class")
confusionMatrix(previsao_v4,testMM$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    9  64   4
##         5    2 117   9
##         10   0  45  27
##
## Overall Statistics
##
##                Accuracy : 0.5523
##                  95% CI : (0.4917, 0.6119)
##     No Information Rate : 0.8159
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2207
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.81818   0.5177   0.67500
## Specificity           0.74436   0.7843   0.81013
## Pos Pred Value        0.11688   0.9141   0.37500
## Neg Pred Value        0.99000   0.2685   0.93659
## Prevalence            0.03971   0.8159   0.14440
## Detection Rate        0.03249   0.4224   0.09747
## Detection Prevalence  0.27798   0.4621   0.25993
## Balanced Accuracy     0.78127   0.6510   0.74256
```

```
# Modelo com o C5.0 e library(C50)
modelo_v5 <- C5.0(Qualidade ~ .
                        ,data= trainMM,trials = 100)
previsao_v5 <- predict(modelo_v5, testMM)
confusionMatrix(previsao_v5,testMM$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    5  14   0
##         5    6 204  17
##         10   0   8  23
##
## Overall Statistics
##
##                Accuracy : 0.8375
##                  95% CI : (0.7887, 0.879)
##     No Information Rate : 0.8159
##     P-Value [Acc > NIR] : 0.1981
##
##                   Kappa : 0.4801
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.45455   0.9027   0.57500
## Specificity           0.94737   0.5490   0.96624
## Pos Pred Value        0.26316   0.8987   0.74194
## Neg Pred Value        0.97674   0.5600   0.93089
## Prevalence            0.03971   0.8159   0.14440
## Detection Rate        0.01805   0.7365   0.08303
## Detection Prevalence  0.06859   0.8195   0.11191
## Balanced Accuracy     0.70096   0.7258   0.77062
```

```
# Modelo com o library(rpart)
modelo_v6 <- rpart(Qualidade ~ .
                       ,data= trainMM,control = rpart.control(cp = .0005), method = 'class')
previsao_v6 <- predict(modelo_v6, testMM, type = 'class')
confusionMatrix(previsao_v6,testMM$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    8  33   0
##         5    2 180  19
##         10   1  13  21
##
## Overall Statistics
##
##                Accuracy : 0.7545
##                  95% CI : (0.6995, 0.804)
##     No Information Rate : 0.8159
##     P-Value [Acc > NIR] : 0.9957
##
##                   Kappa : 0.3605
##
##  Mcnemar's Test P-Value : 1.69e-06
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.72727   0.7965   0.52500
## Specificity           0.87594   0.5882   0.94093
## Pos Pred Value        0.19512   0.8955   0.60000
## Neg Pred Value        0.98729   0.3947   0.92149
## Prevalence            0.03971   0.8159   0.14440
## Detection Rate        0.02888   0.6498   0.07581
## Detection Prevalence  0.14801   0.7256   0.12635
## Balanced Accuracy     0.80161   0.6923   0.73296
```

Dados com o Scaler

```
# Modelo  com o KSVM  library(kernlab)

modelo_v7 <- ksvm(Qualidade ~ .
                  ,data= trainS,type="C-bsvc", kernel = "rbfdot")
previsao_v7 <- predict(modelo_v7, testS)
confusionMatrix(previsao_v7,testS$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    8  39   0
##         5    5 162   7
##         10   0  34  22
##
## Overall Statistics
##
##                Accuracy : 0.6931
##                  95% CI : (0.6352, 0.7469)
##     No Information Rate : 0.8484
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2993
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.61538   0.6894   0.75862
## Specificity           0.85227   0.7143   0.86290
## Pos Pred Value        0.17021   0.9310   0.39286
## Neg Pred Value        0.97826   0.2913   0.96833
## Prevalence            0.04693   0.8484   0.10469
## Detection Rate        0.02888   0.5848   0.07942
## Detection Prevalence  0.16968   0.6282   0.20217
## Balanced Accuracy     0.73383   0.7018   0.81076
```

```
# Modelo  com o RandomForest library('randomForest')
modelo_v8 <- randomForest(Qualidade ~ .
                          ,data= trainS,ntree = 100 ,
                          nodesize = 10,method="repeatedcv",
                          number=15, repeats=10)
previsao_v8 <- predict(modelo_v8, testS)
confusionMatrix(previsao_v8,testS$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    3  14   0
##         5   10 207   9
##         10   0  14  20
##
## Overall Statistics
##
##                Accuracy : 0.8303
##                  95% CI : (0.7808, 0.8726)
##     No Information Rate : 0.8484
##     P-Value [Acc > NIR] : 0.8224
##
##                   Kappa : 0.4191
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.23077   0.8809    0.6897
## Specificity           0.94697   0.5476    0.9435
## Pos Pred Value        0.17647   0.9159    0.5882
## Neg Pred Value        0.96154   0.4510    0.9630
## Prevalence            0.04693   0.8484    0.1047
## Detection Rate        0.01083   0.7473    0.0722
## Detection Prevalence  0.06137   0.8159    0.1227
## Balanced Accuracy     0.58887   0.7142    0.8166
```

```
# Modelo svm do pacote library(e1071)
modelo_v9 <- svm(Qualidade ~ .
                 ,data= trainS,type = 'C-classification',
                 kernel = 'radial',cost = 10, scale = FALSE)
previsao_v9 <- predict(modelo_v9, testS)
confusionMatrix(previsao_v9,testS$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    5  34   0
##         5    8 172   8
##         10   0  29  21
##
## Overall Statistics
##
##                Accuracy : 0.7148
##                  95% CI : (0.6577, 0.7672)
##     No Information Rate : 0.8484
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2847
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.38462   0.7319   0.72414
## Specificity           0.87121   0.6190   0.88306
## Pos Pred Value        0.12821   0.9149   0.42000
## Neg Pred Value        0.96639   0.2921   0.96476
## Prevalence            0.04693   0.8484   0.10469
## Detection Rate        0.01805   0.6209   0.07581
## Detection Prevalence  0.14079   0.6787   0.18051
## Balanced Accuracy     0.62791   0.6755   0.80360
```

```
# Modelo   naiveBayes do  pacote library(e1071)
modelo_v10 <- naiveBayes(Qualidade ~ .
                         ,data= trainS, laplace=1)
previsao_v10 <- predict(modelo_v10, testS, type="class")
confusionMatrix(previsao_v10,testS$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    9  74   1
##         5    3 101   1
##         10   1  60  27
##
## Overall Statistics
##
##                Accuracy : 0.4946
##                  95% CI : (0.4342, 0.5551)
##     No Information Rate : 0.8484
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1989
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.69231   0.4298   0.93103
## Specificity           0.71591   0.9048   0.75403
## Pos Pred Value        0.10714   0.9619   0.30682
## Neg Pred Value        0.97927   0.2209   0.98942
## Prevalence            0.04693   0.8484   0.10469
## Detection Rate        0.03249   0.3646   0.09747
## Detection Prevalence  0.30325   0.3791   0.31769
## Balanced Accuracy     0.70411   0.6673   0.84253
```

```
# Modelo com o C5.0 e library(C50)
modelo_v11 <- C5.0(Qualidade ~ .
                       ,data= trainS,trials = 100)
previsao_v11 <- predict(modelo_v11, testS)
confusionMatrix(previsao_v11,testS$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    3  10   0
##         5   10 211  10
##         10   0  14  19
##
## Overall Statistics
##
##                Accuracy : 0.8412
##                  95% CI : (0.7927, 0.8821)
##     No Information Rate : 0.8484
##     P-Value [Acc > NIR] : 0.6681
##
##                   Kappa : 0.4283
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.23077   0.8979   0.65517
## Specificity           0.96212   0.5238   0.94355
## Pos Pred Value        0.23077   0.9134   0.57576
## Neg Pred Value        0.96212   0.4783   0.95902
## Prevalence            0.04693   0.8484   0.10469
## Detection Rate        0.01083   0.7617   0.06859
## Detection Prevalence  0.04693   0.8339   0.11913
## Balanced Accuracy     0.59645   0.7108   0.79936
```

```
# Modelo com o library(rpart)
modelo_v12 <- rpart(Qualidade ~ .
                    ,data= trainS,control = rpart.control(cp = .0005), method = 'class')
previsao_v12 <- predict(modelo_v12, testS, type = 'class')
confusionMatrix(previsao_v12,testS$Qualidade)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   5  10
##         1    4  40   1
##         5    9 173   8
##         10   0  22  20
##
## Overall Statistics
##
##                Accuracy : 0.7112
##                  95% CI : (0.6539, 0.7639)
##     No Information Rate : 0.8484
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2681
##
##  Mcnemar's Test P-Value : 5.488e-06
##
## Statistics by Class:
##
##                      Class: 1 Class: 5 Class: 10
## Sensitivity           0.30769   0.7362    0.6897
## Specificity           0.84470   0.5952    0.9113
## Pos Pred Value        0.08889   0.9105    0.4762
## Neg Pred Value        0.96121   0.2874    0.9617
## Prevalence            0.04693   0.8484    0.1047
## Detection Rate        0.01444   0.6245    0.0722
## Detection Prevalence  0.16245   0.6859    0.1516
## Balanced Accuracy     0.57619   0.6657    0.8005
```

Considerações Finais

Com a pouca quantidade de dados, a acurácia fica um pouco comprometida pois, o modelo não tem tantos dados para aprender, mesmo assim tivemos um bom aproveitamento com os modelos randomForest e o C5.0, tanto nos dados Min Max Scaler quanto no Scaler, escolheria algum desse modelos para os novos dados.

Obrigado! Entre em contato comigo acessando meu portifolio (https://campos1989.github.io/ (https://campos1989.github.io/)) no menu contato!