Forge

2024-10-28

```r
#Importing necessary libraries, importing data, and readying data for analysis
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```r
library(Hmisc)
```

```
##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library(ggplot2)
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha

## The following object is masked from 'package:Hmisc':
##
##     describe
```

```r
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(vioplot)
```

```
## Loading required package: sm

## Package 'sm', version 2.2-6.0: type help(sm) for summary information

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(DescTools)
```

```
##
```

```
## Attaching package: 'DescTools'

## The following objects are masked from 'package:psych':
##
##     AUC, ICC, SD

## The following objects are masked from 'package:Hmisc':
##
##     %nin%, Label, Mean, Quantile
```

```r
library(leaps)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x psych::%+%()       masks ggplot2::%+%()
## x psych::alpha()     masks ggplot2::alpha()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x dplyr::src()       masks Hmisc::src()
## x dplyr::summarize() masks Hmisc::summarize()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
##
## The following objects are masked from 'package:DescTools':
##
##     MAE, RMSE
```

```r
library(e1071)
```

```
##
## Attaching package: 'e1071'
##
## The following object is masked from 'package:Hmisc':
##
##     impute
```

```r
library(rattle)
```

```
## Loading required package: bitops
##
## Attaching package: 'bitops'
##
## The following object is masked from 'package:DescTools':
```

```
##
##     %^%
##
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
##
## Attaching package: 'rattle'
##
## The following object is masked from 'package:sm':
##
##     binning
```

```r
library(dplyr)
library(rpart)
library(kknn)
```

```
##
## Attaching package: 'kknn'
##
## The following object is masked from 'package:caret':
##
##     contr.dummy
```

```r
library(stats)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
##
## The following object is masked from 'package:sm':
##
##     muscle
```

```r
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
##
## The following object is masked from 'package:DescTools':
```

```
##
##     Recode
##
## The following object is masked from 'package:psych':
##
##     logit
```

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:rattle':
##
##     xgboost
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(tidyverse)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
##
## The following object is masked from 'package:DescTools':
##
##     %like%
##
## The following objects are masked from 'package:zoo':
##
##     yearmon, yearqtr
```

```r
library(skimr)
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:rattle':
```

```
##
##     importance
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:psych':
##
##     outlier
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```r
library(tuneRanger)
```

```
## Loading required package: ranger
##
## Attaching package: 'ranger'
##
## The following object is masked from 'package:randomForest':
##
##     importance
##
## The following object is masked from 'package:rattle':
##
##     importance
##
## Loading required package: mlrMBO
## Loading required package: mlr
## Loading required package: ParamHelpers
##
## Attaching package: 'mlr'
##
## The following object is masked from 'package:e1071':
##
##     impute
##
## The following object is masked from 'package:caret':
##
##     train
##
## The following object is masked from 'package:Hmisc':
##
##     impute
##
## Loading required package: smoof
## Loading required package: checkmate
## Loading required package: parallel
## Loading required package: lhs
```

```r
library(VSURF)
```

```
##
## Attaching package: 'VSURF'
##
## The following object is masked from 'package:e1071':
##
##     tune
```

```r
library(foreach)
```

```
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
##
## The following object is masked from 'package:DescTools':
##
##     %:%
```

```r
library(doParallel)
```

```
## Loading required package: iterators
```

```r
setwd("~/Downloads")
redwine <- read.csv("winequality-red.csv")
whitewine <- read.csv("winequality-white.csv")

cat("Creating variable Names Red Wine")
```

```
## Creating variable Names Red Wine
```

```r
redwine_seperated <- str_split_fixed(redwine$fixed.acidity.volatile.acidity.citric.acid.residual.sugar.

redwine_seperated <- data.frame(redwine_seperated)

cat("Creating variable Names White Wine")
```

```
## Creating variable Names White Wine
```

```r
whitewine_seperated <- str_split_fixed(whitewine$fixed.acidity.volatile.acidity.citric.acid.residual.su

whitewine_seperated <- data.frame(whitewine_seperated)



redwine_seperated <- redwine_seperated %>%
  rename(fixed_acidity = 'X1')
redwine_seperated <- redwine_seperated %>%
  rename(volatile_acidity = 'X2')
redwine_seperated <- redwine_seperated %>%
  rename(citric_acid = 'X3')
redwine_seperated <- redwine_seperated %>%
  rename(residual_sugar = 'X4')
redwine_seperated <- redwine_seperated %>%
  rename(chlorides = 'X5')
redwine_seperated <- redwine_seperated %>%
  rename(free_sulfur_dioxide = 'X6')
redwine_seperated <- redwine_seperated %>%
  rename(total_sulfur_dioxide = 'X7')
redwine_seperated <- redwine_seperated %>%
  rename(density = 'X8')
redwine_seperated <- redwine_seperated %>%
  rename(pH = 'X9')
redwine_seperated <- redwine_seperated %>%
  rename(sulphates = 'X10')
redwine_seperated <- redwine_seperated %>%
  rename(alcohol = 'X11')
redwine_seperated <- redwine_seperated %>%
  rename(quality = 'X12')

whitewine_seperated <- str_split_fixed(whitewine$fixed.acidity.volatile.acidity.citric.acid.residual.su

whitewine_seperated <- data.frame(whitewine_seperated)

whitewine_seperated <- whitewine_seperated %>%
  rename(fixed_acidity = 'X1')
whitewine_seperated <- whitewine_seperated %>%
  rename(volatile_acidity = 'X2')
whitewine_seperated <- whitewine_seperated %>%
  rename(citric_acid = 'X3')
whitewine_seperated <- whitewine_seperated %>%
  rename(residual_sugar = 'X4')
whitewine_seperated <- whitewine_seperated %>%
  rename(chlorides = 'X5')
whitewine_seperated <- whitewine_seperated %>%
  rename(free_sulfur_dioxide = 'X6')
whitewine_seperated <- whitewine_seperated %>%
  rename(total_sulfur_dioxide = 'X7')
whitewine_seperated <- whitewine_seperated %>%
  rename(density = 'X8')
whitewine_seperated <- whitewine_seperated %>%
  rename(pH = 'X9')
whitewine_seperated <- whitewine_seperated %>%
  rename(sulphates = 'X10')
whitewine_seperated <- whitewine_seperated %>%
  rename(alcohol = 'X11')
whitewine_seperated <- whitewine_seperated %>%
  rename(quality = 'X12')


redwine_seperated <- apply(redwine_seperated,2,as.numeric)
whitewine_seperated <- apply(whitewine_seperated,2,as.numeric)

redwine_seperated <- data.frame(redwine_seperated)
whitewine_seperated <- data.frame(whitewine_seperated)
```

```r
redwine_seperated$type <- 'red'
whitewine_seperated$type <- 'white'

redwine_seperated$type <- as.factor(redwine_seperated$type)
whitewine_seperated$type <- as.factor(whitewine_seperated$type)

wine <- full_join(redwine_seperated,whitewine_seperated)
```

```
## Joining with `by = join_by(fixed_acidity, volatile_acidity, citric_acid,
## residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, density,
## pH, sulphates, alcohol, quality, type)`
```

```r
#Exploratory Data Analysis Getting Descriptive Statistics for Red and White Wine
#Getting descriptive stats for Red Wine

skim(redwine_seperated)
```

Table 1: Data summary

Name	redwine_seperated
Number of rows	1599
Number of columns	13
Column type frequency:	
factor	1
numeric	12
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
type	0	1	FALSE	1	red: 1599

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
fixed_acidity	0	1	8.32	1.74	4.60	7.10	7.90	9.20	15.90	
volatile_acidity	0	1	0.53	0.18	0.12	0.39	0.52	0.64	1.58	
citric_acid	0	1	0.27	0.19	0.00	0.09	0.26	0.42	1.00	
residual_sugar	0	1	2.54	1.41	0.90	1.90	2.20	2.60	15.50	
chlorides	0	1	0.09	0.05	0.01	0.07	0.08	0.09	0.61	
free_sulfur_dioxide	0	1	15.87	10.46	1.00	7.00	14.00	21.00	72.00	
total_sulfur_dioxide	0	1	46.47	32.90	6.00	22.00	38.00	62.00	289.00	
density	0	1	1.00	0.00	0.99	1.00	1.00	1.00	1.00	
pH	0	1	3.31	0.15	2.74	3.21	3.31	3.40	4.01	
sulphates	0	1	0.66	0.17	0.33	0.55	0.62	0.73	2.00	
alcohol	0	1	10.42	1.07	8.40	9.50	10.20	11.10	14.90	
quality	0	1	5.64	0.81	3.00	5.00	6.00	6.00	8.00	

```r
#Basic Stats
summary(redwine_seperated)
```

```
##  fixed_acidity   volatile_acidity  citric_acid    residual_sugar  
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900  
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900  
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200  
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539  
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600  
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500  
##    chlorides       free_sulfur_dioxide total_sulfur_dioxide    density      
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901  
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956  
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968  
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967  
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978  
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037  
##        pH          sulphates         alcohol         quality       type     
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000   red:1599  
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000             
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000             
##  Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636             
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000             
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000             
```

```r
#Getting descriptive stats for White Wine

skim(whitewine_seperated)
```

Table 4: Data summary

Name	whitewine_seperated
Number of rows	4898
Number of columns	13
Column type frequency:	
factor	1
numeric	12
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
type	0	1	FALSE	1	whi: 4898

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
fixed_acidity	0	1	6.85	0.84	3.80	6.30	6.80	7.30	14.20	
volatile_acidity	0	1	0.28	0.10	0.08	0.21	0.26	0.32	1.10	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
citric_acid	0	1	0.33	0.12	0.00	0.27	0.32	0.39	1.66	
residual_sugar	0	1	6.39	5.07	0.60	1.70	5.20	9.90	65.80	
chlorides	0	1	0.05	0.02	0.01	0.04	0.04	0.05	0.35	
free_sulfur_dioxide	0	1	35.31	17.01	2.00	23.00	34.00	46.00	289.00	
total_sulfur_dioxide	0	1	138.36	42.50	9.00	108.00	134.00	167.00	440.00	
density	0	1	0.99	0.00	0.99	0.99	0.99	1.00	1.04	
pH	0	1	3.19	0.15	2.72	3.09	3.18	3.28	3.82	
sulphates	0	1	0.49	0.11	0.22	0.41	0.47	0.55	1.08	
alcohol	0	1	10.51	1.23	8.00	9.50	10.40	11.40	14.20	
quality	0	1	5.88	0.89	3.00	5.00	6.00	6.00	9.00	

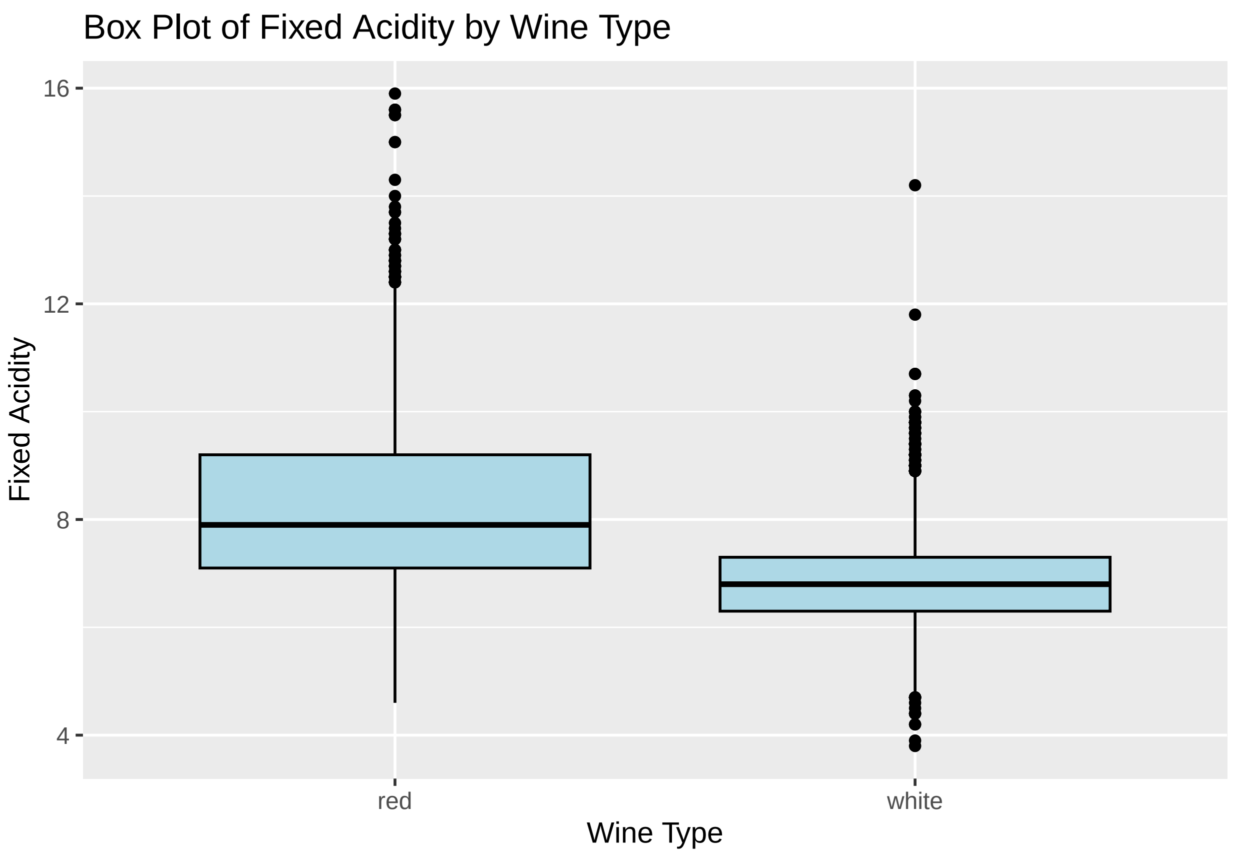
```r
#Basic Stats
summary(whitewine_seperated)
```

```
##  fixed_acidity    volatile_acidity  citric_acid     residual_sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free_sulfur_dioxide total_sulfur_dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##        pH          sulphates         alcohol         quality          type
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000   white:4898
##  1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.000
##  Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

```r
cat("EDA Continued: Exploring Data by viewing distributions,
    and frequency of outliers for each variable")
```

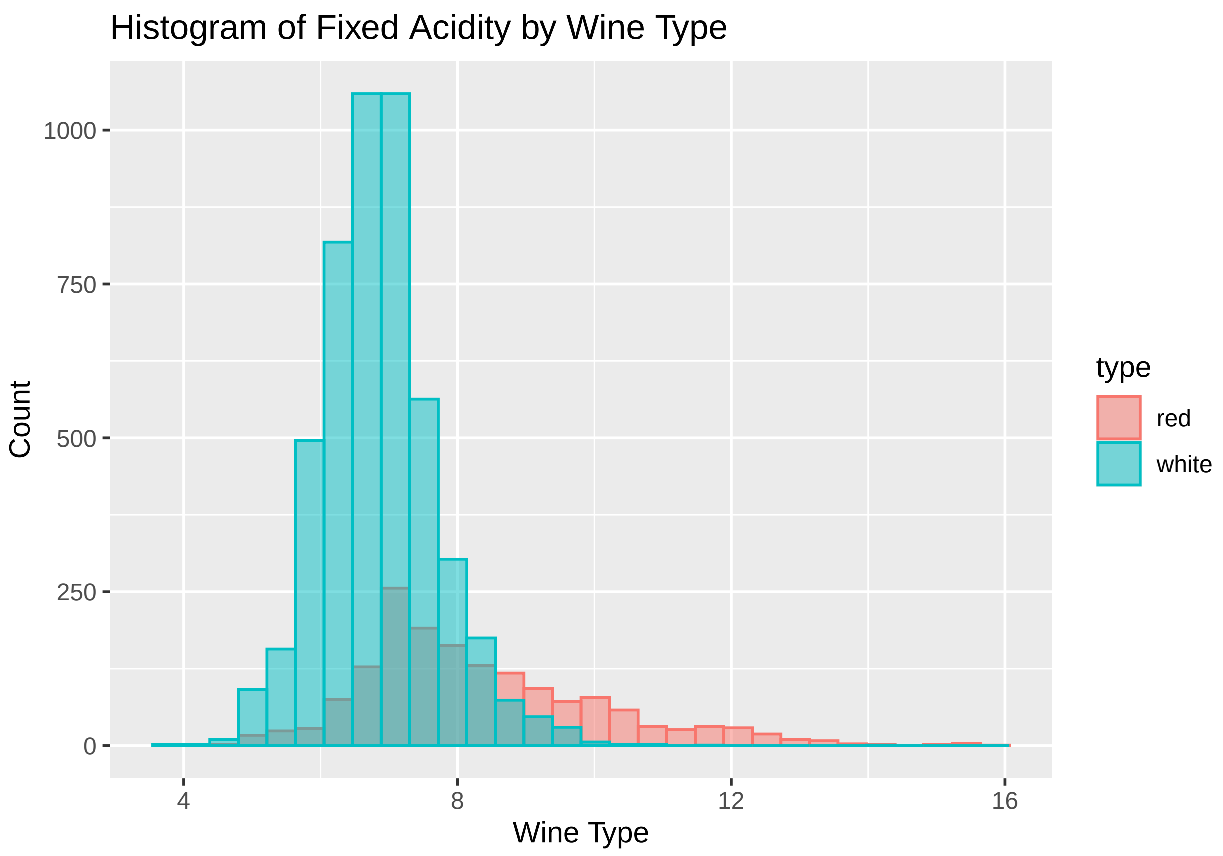
```
## EDA Continued: Exploring Data by viewing distributions,
##     and frequency of outliers for each variable
```

```r
# Box plot Fixed Acidity grouped by Wine Type
ggplot(wine, aes(x = type, y = fixed_acidity)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Fixed Acidity",
       title = "Box Plot of Fixed Acidity by Wine Type")
```

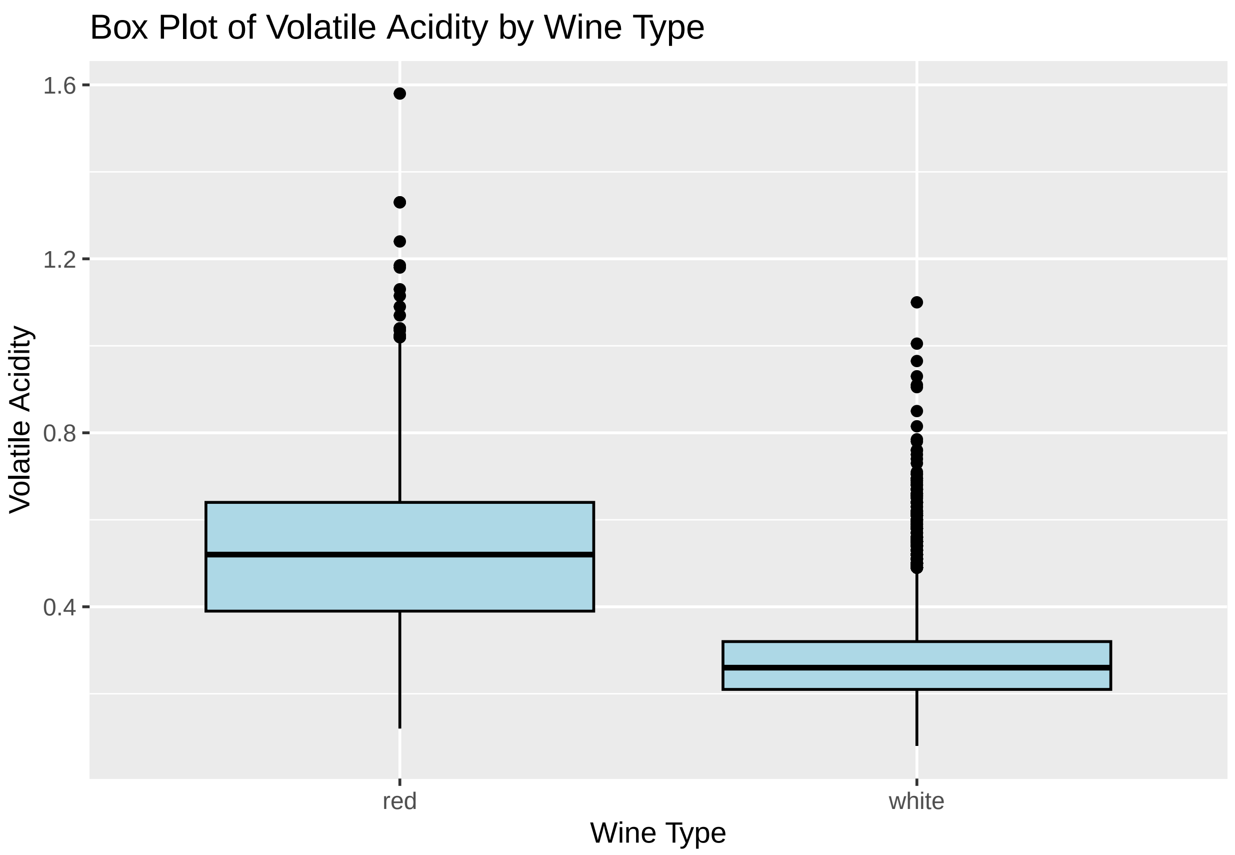


```r
# Histogram of Fixed Acidity with bars colored by Wine Type
ggplot(wine, aes(x = fixed_acidity, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Fixed Acidity by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

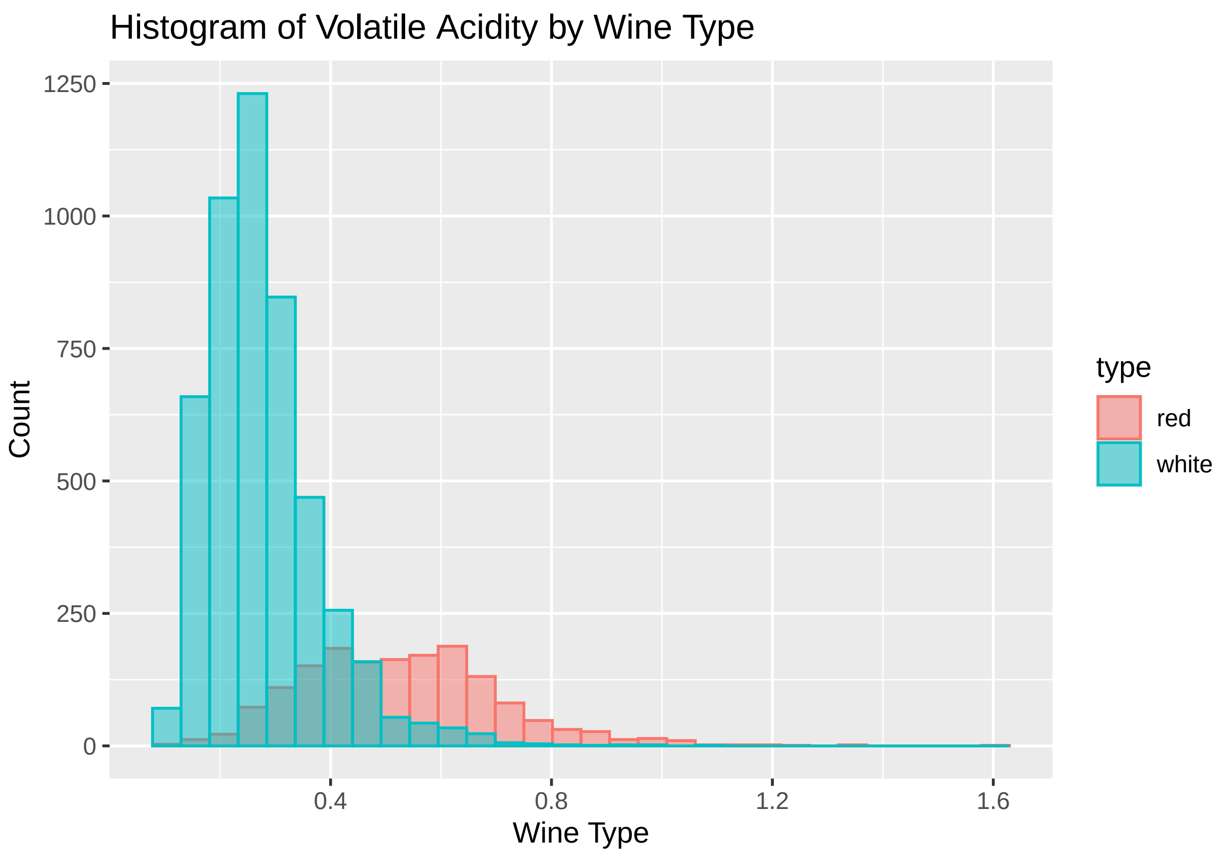


```r
# Box plot Volatile Acidity grouped by Wine Type
ggplot(wine, aes(x = type, y = volatile_acidity)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Volatile Acidity",
       title = "Box Plot of Volatile Acidity by Wine Type")
```

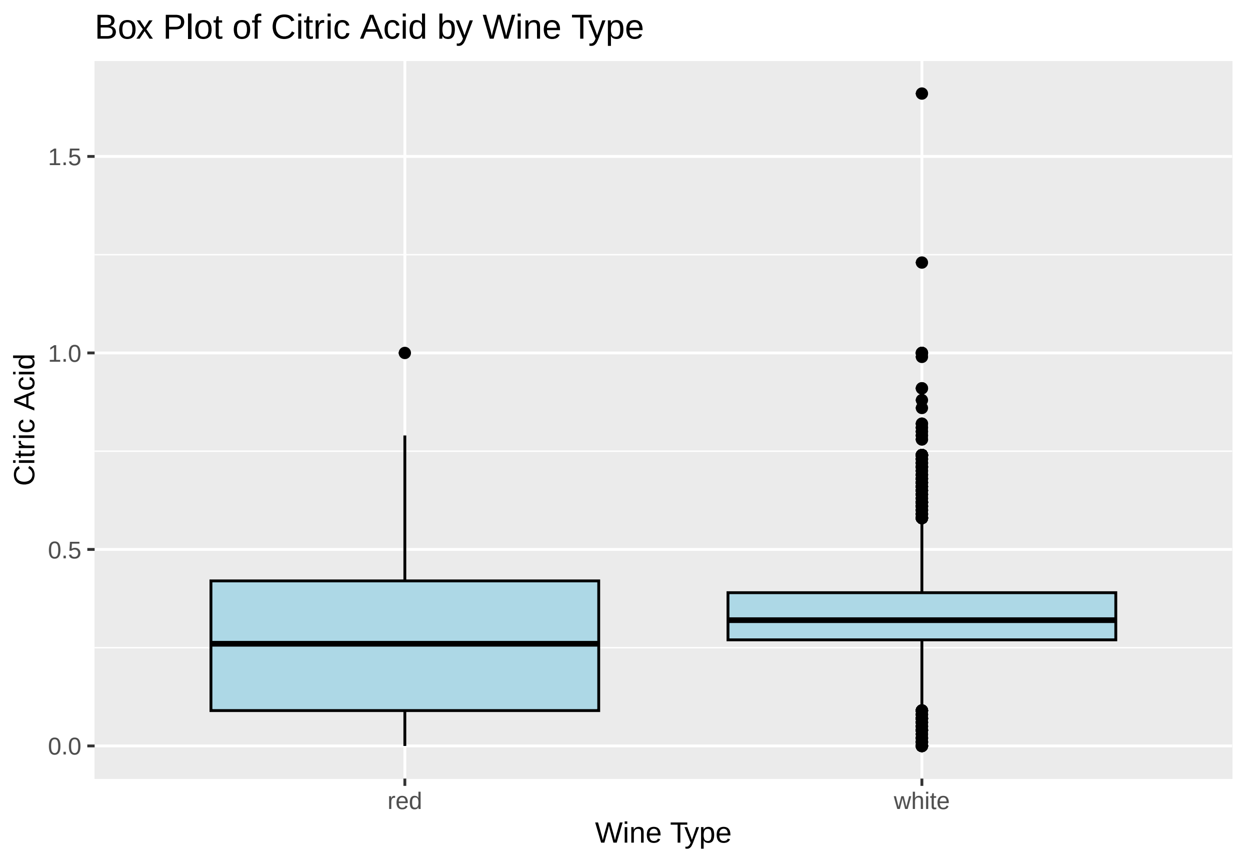


```r
# Histogram of Volatile Acidity  with bars colored by Wine Type
ggplot(wine, aes(x = volatile_acidity, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Volatile Acidity by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

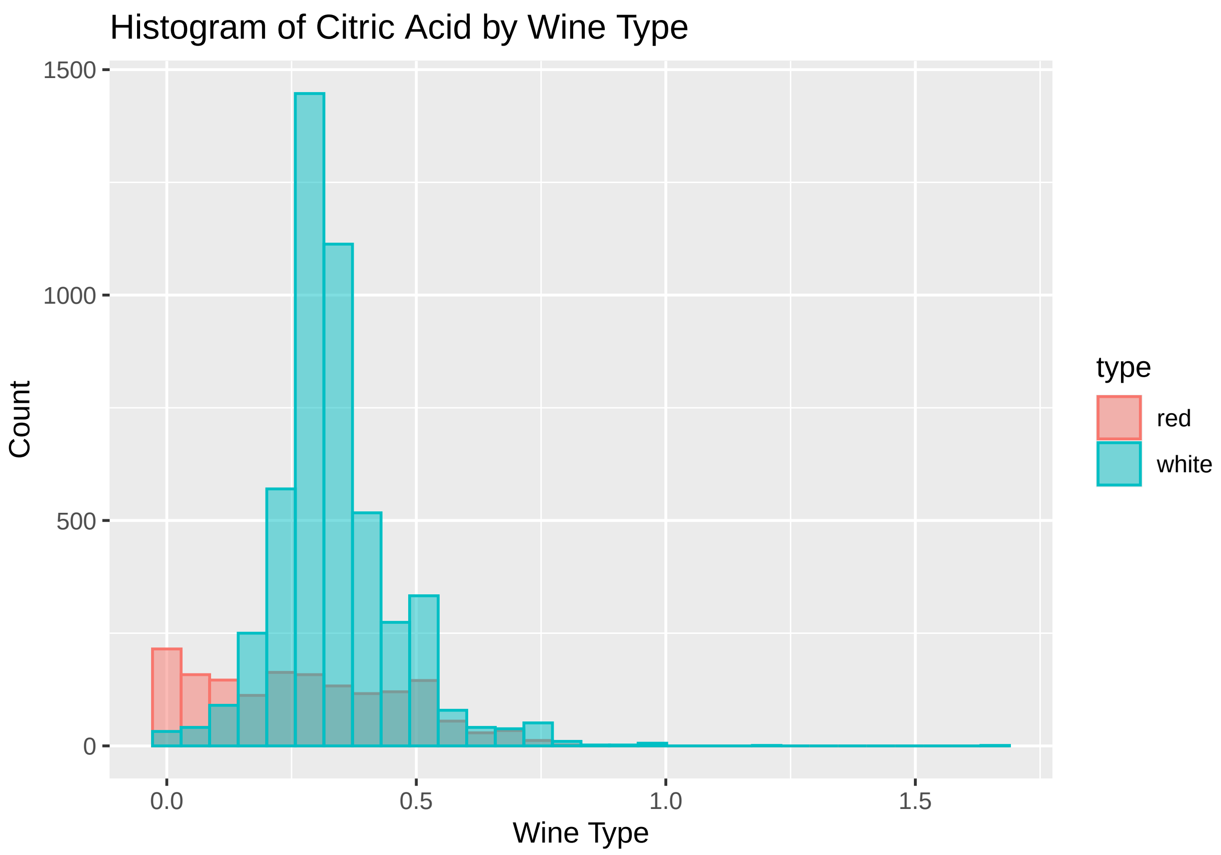


```
# Box plot Citric Acid grouped by Wine Type
ggplot(wine, aes(x = type, y = citric_acid)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Citric Acid",
       title = "Box Plot of Citric Acid by Wine Type")
```

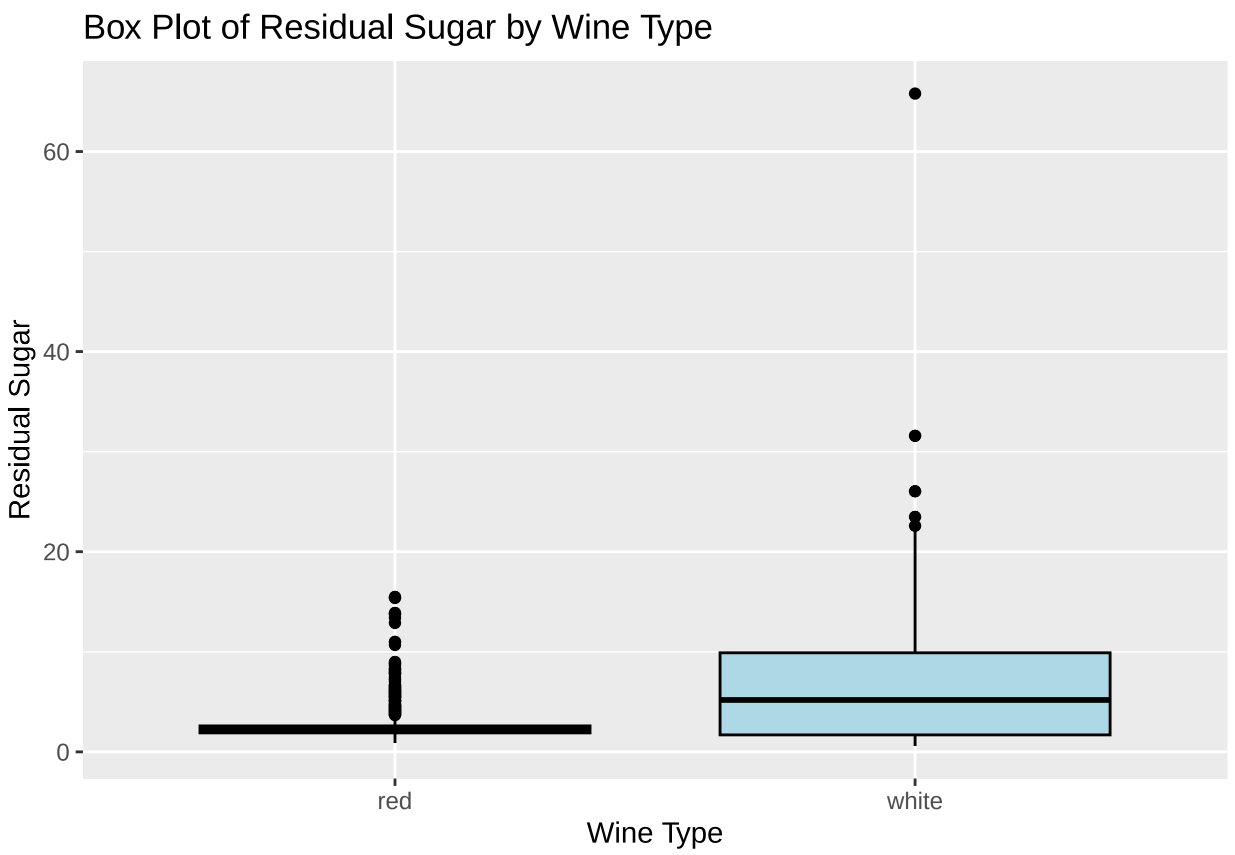


```r
# Histogram of Citric Acid with bars colored by Wine Type
ggplot(wine, aes(x = citric_acid, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Citric Acid by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

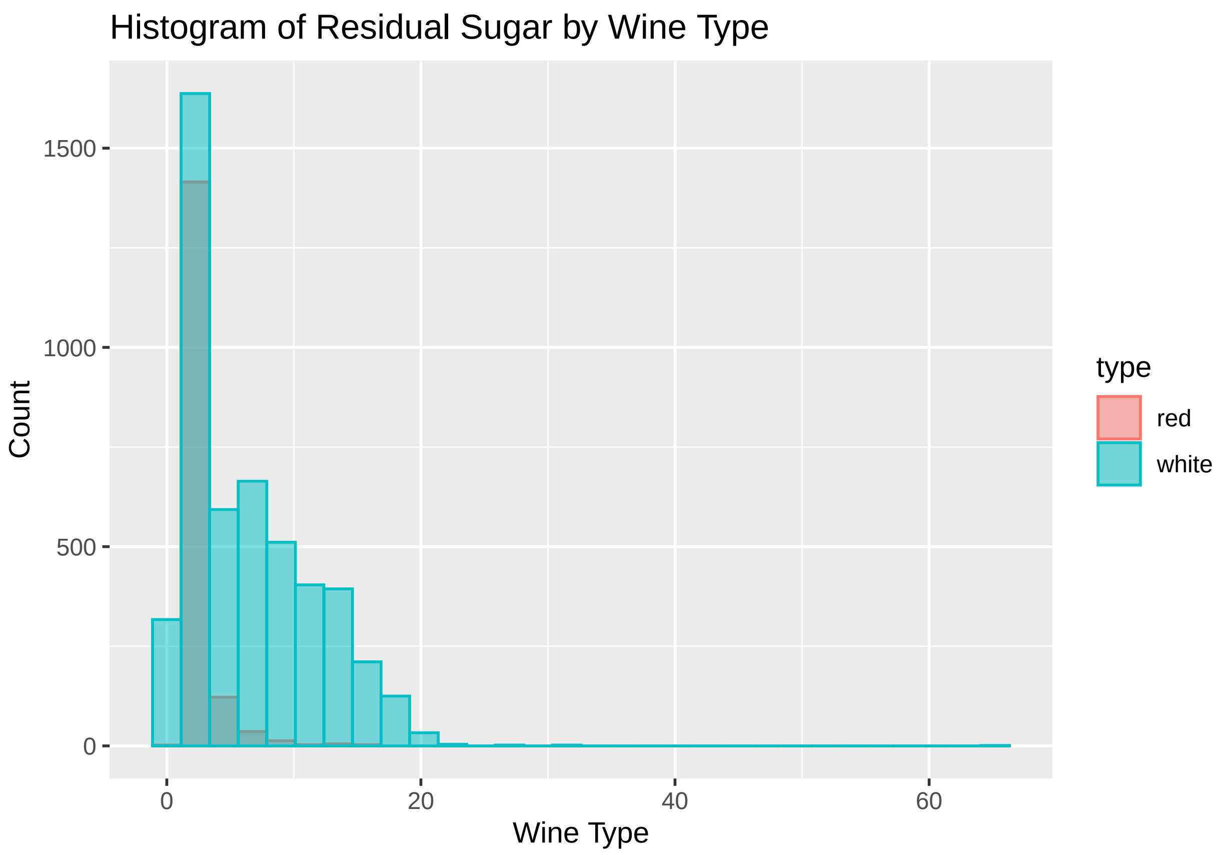


```r
# Box plot Residual Sugar grouped by Wine Type
ggplot(wine, aes(x = type, y = residual_sugar)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Residual Sugar",
       title = "Box Plot of Residual Sugar by Wine Type")
```

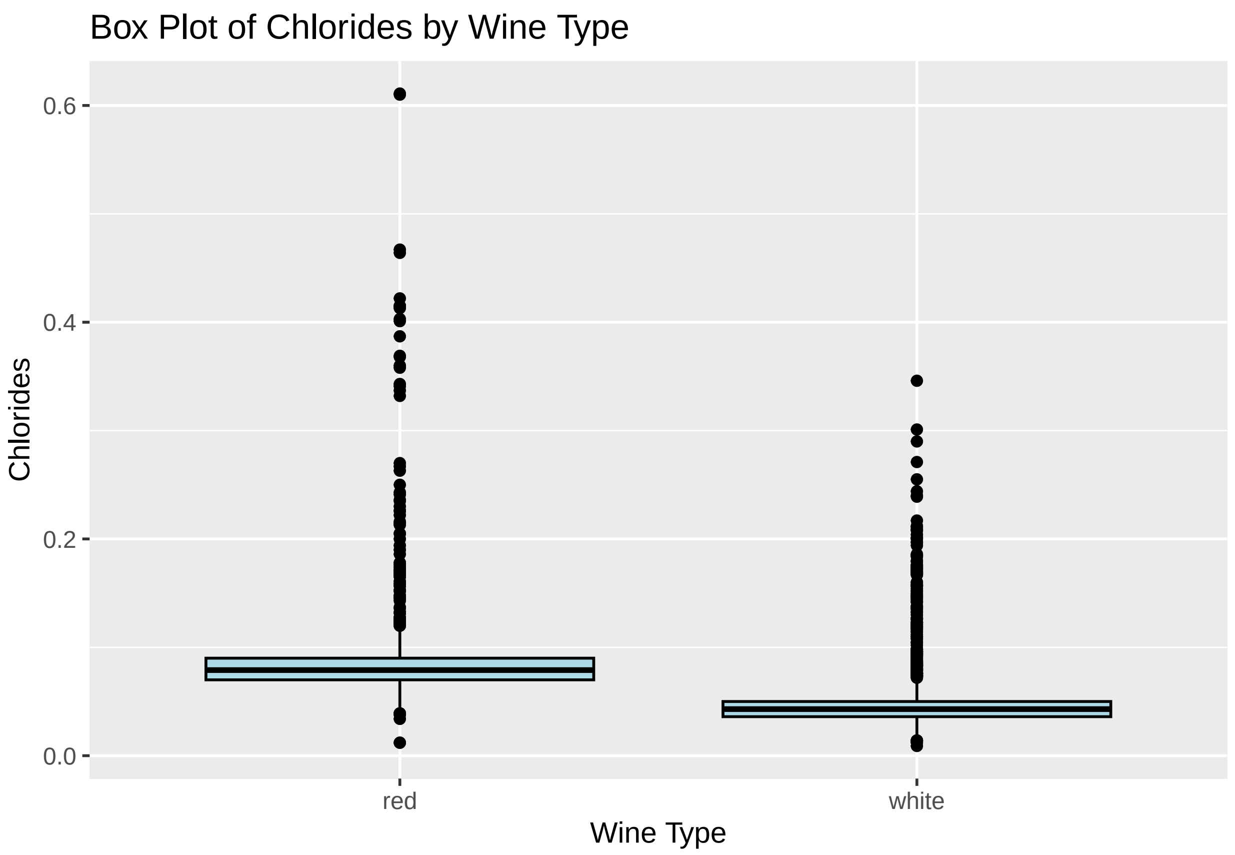


```r
# Histogram of Residual Sugar with bars colored by Wine Type
ggplot(wine, aes(x = residual_sugar, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Residual Sugar by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

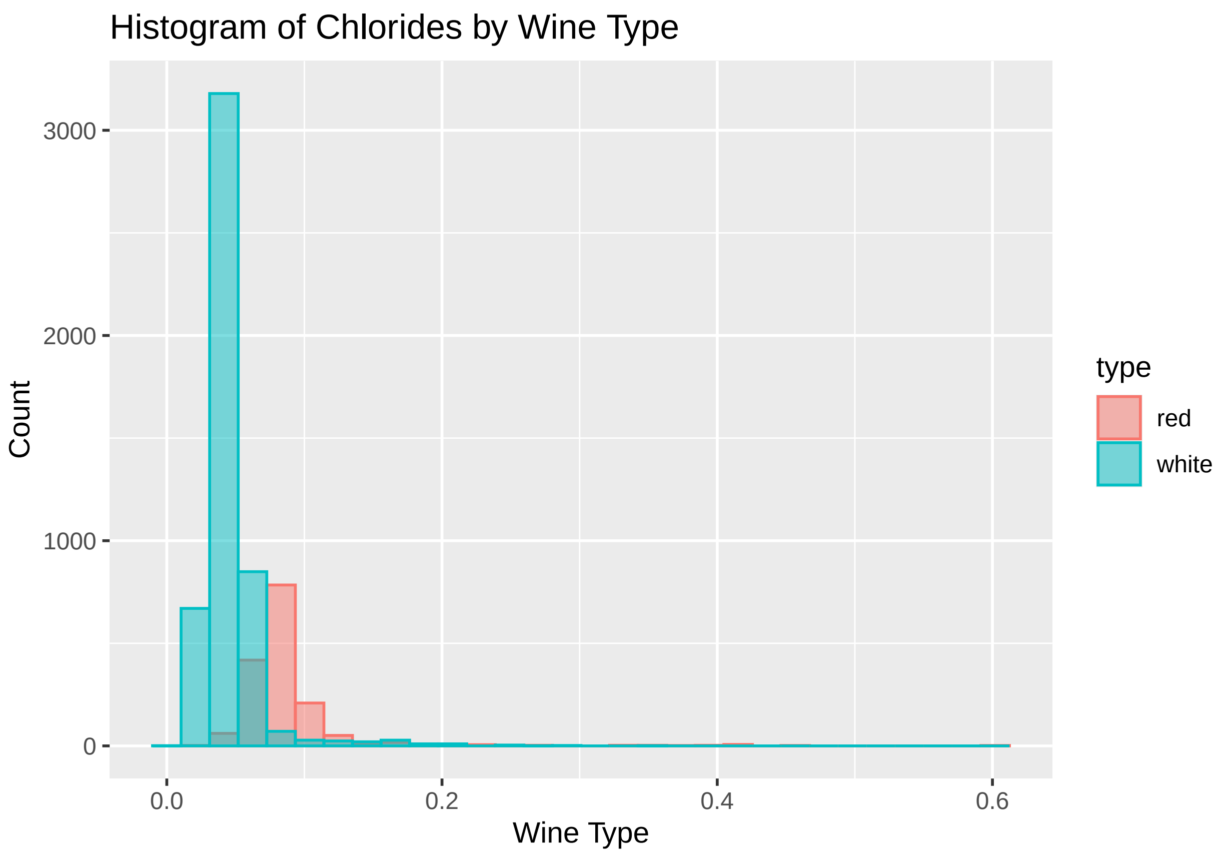


```r
# Box plot Chlorides grouped by Wine Type
ggplot(wine, aes(x = type, y = chlorides)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Chlorides",
       title = "Box Plot of Chlorides by Wine Type")
```

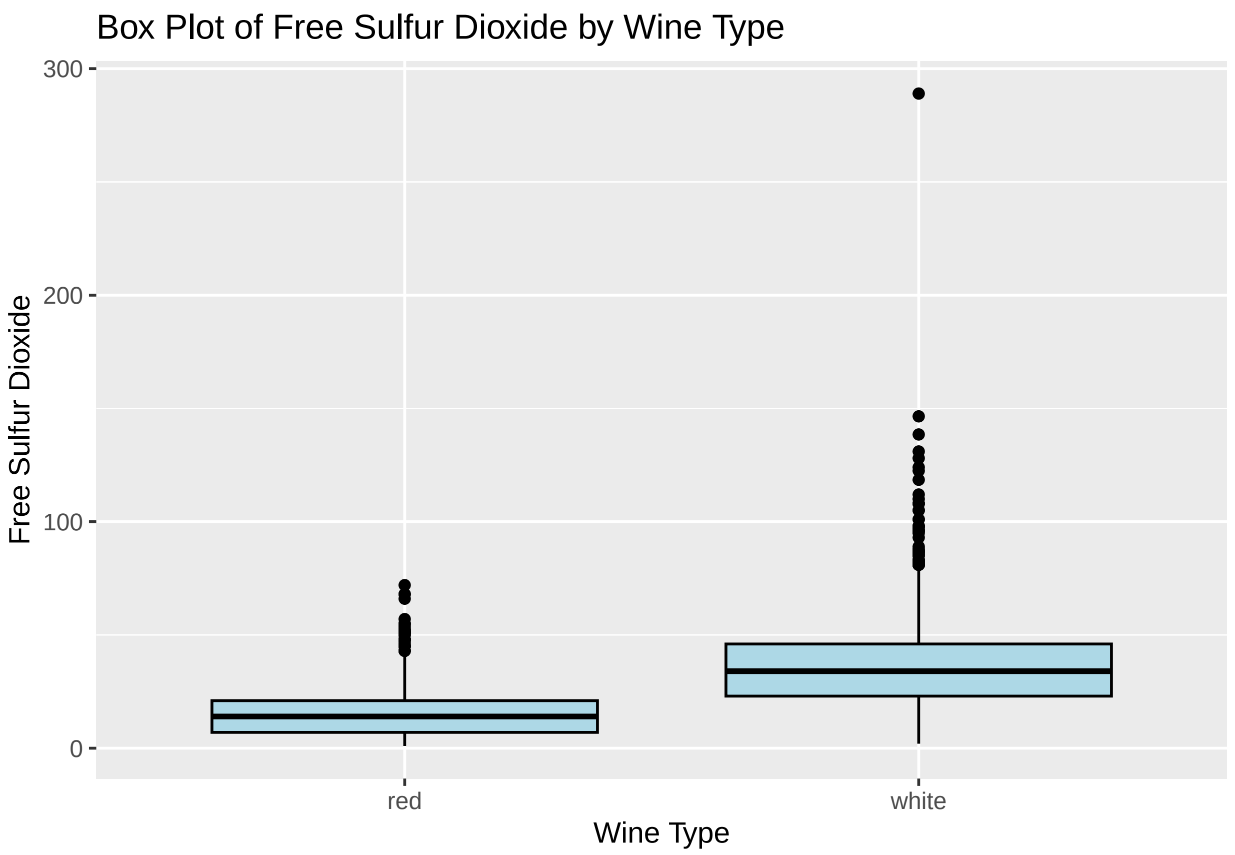


```r
# Histogram of Chlorides with bars colored by Wine Type
ggplot(wine, aes(x = chlorides, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Chlorides by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

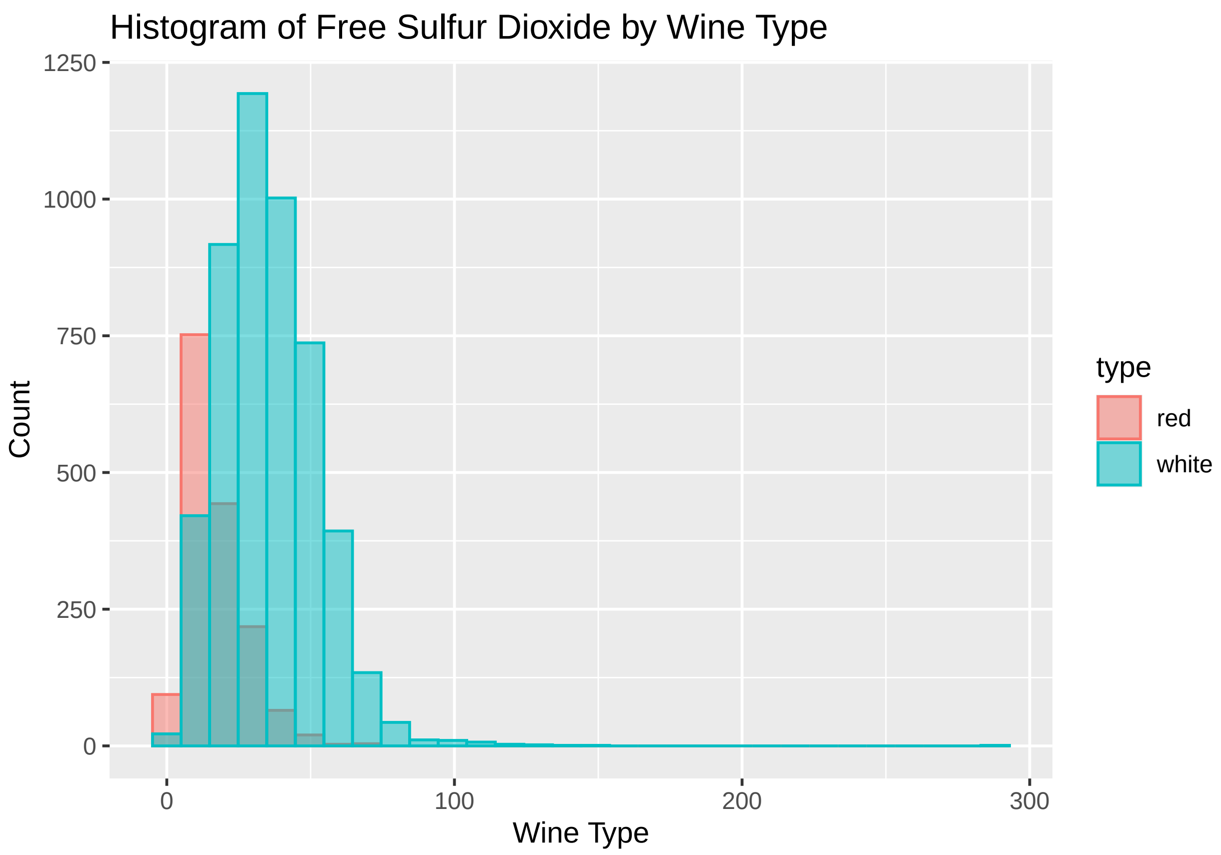


```
# Box plot Free Sulfur Dioxide grouped by Wine Type
ggplot(wine, aes(x = type, y = free_sulfur_dioxide)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Free Sulfur Dioxide",
       title = "Box Plot of Free Sulfur Dioxide by Wine Type")
```

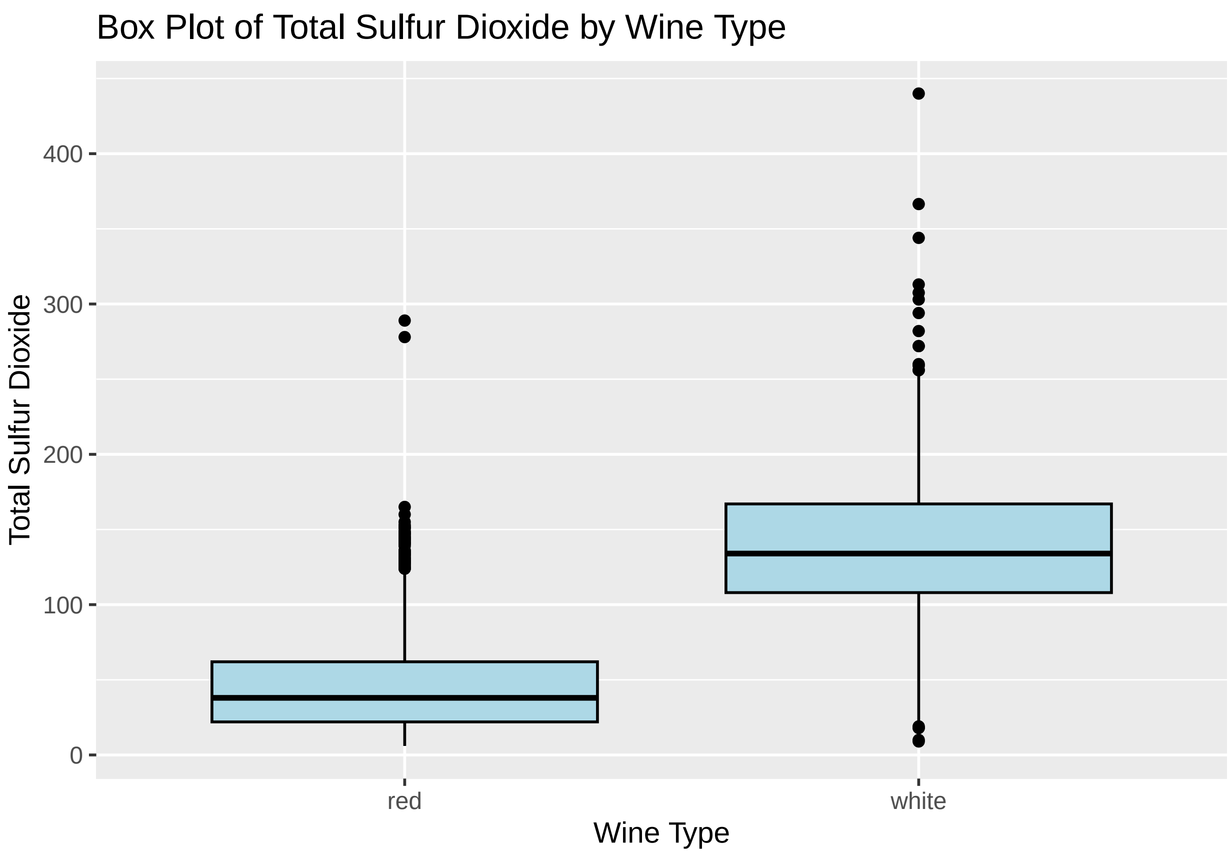


```r
# Histogram of Free Sulfur Dioxide with bars colored by Wine Type
ggplot(wine, aes(x = free_sulfur_dioxide, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Free Sulfur Dioxide by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

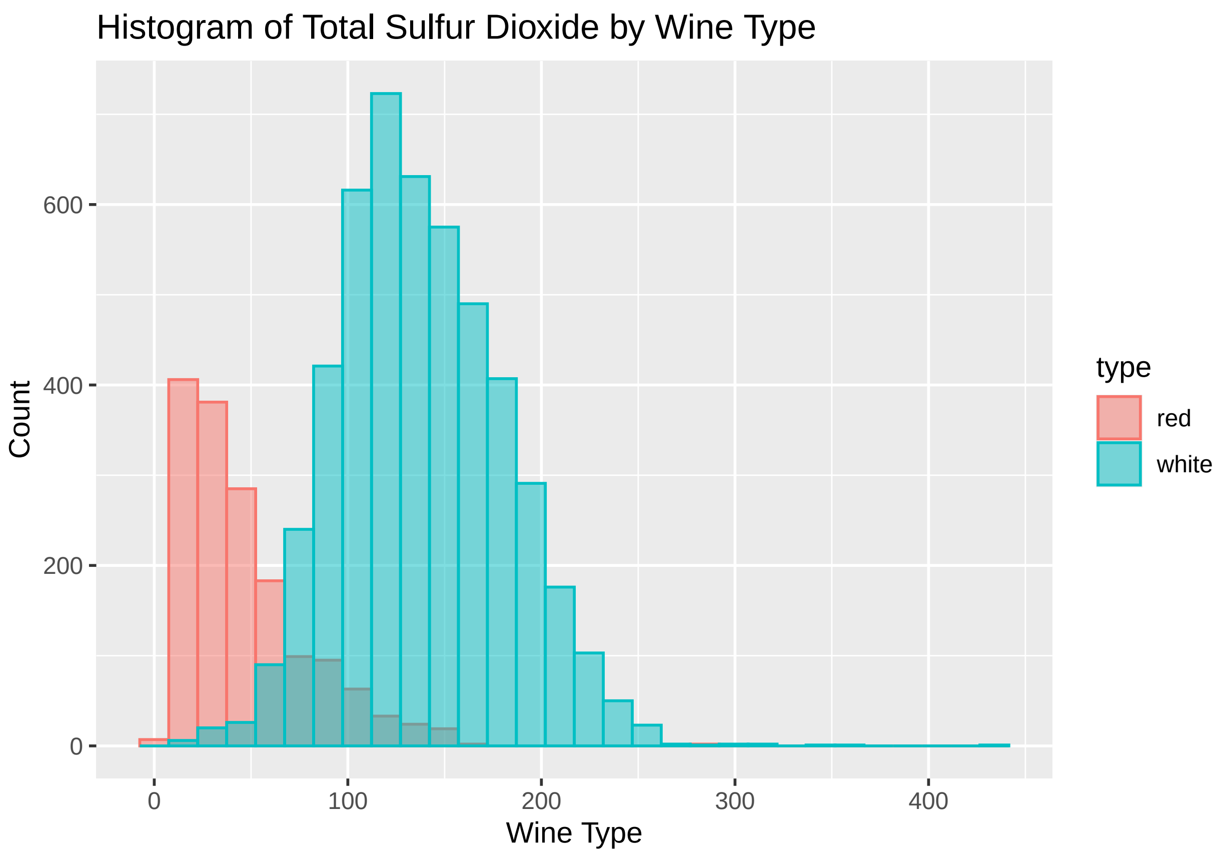


```r
# Box plot Total Sulfur Dioxide grouped by Wine Type
ggplot(wine, aes(x = type, y = total_sulfur_dioxide)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Total Sulfur Dioxide",
       title = "Box Plot of Total Sulfur Dioxide by Wine Type")
```

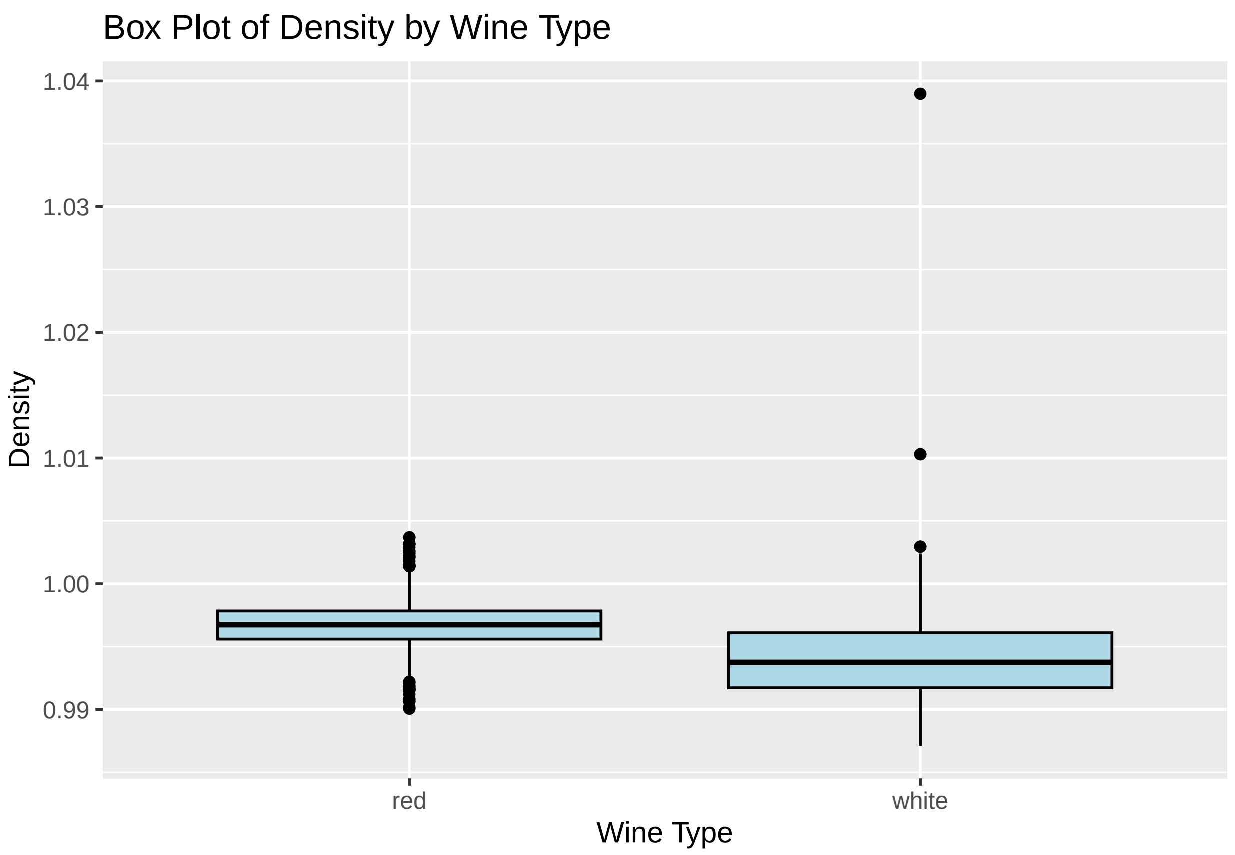


```
# Histogram of Total Sulfur Dioxide with bars colored by Wine Type
ggplot(wine, aes(x = total_sulfur_dioxide, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Total Sulfur Dioxide by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

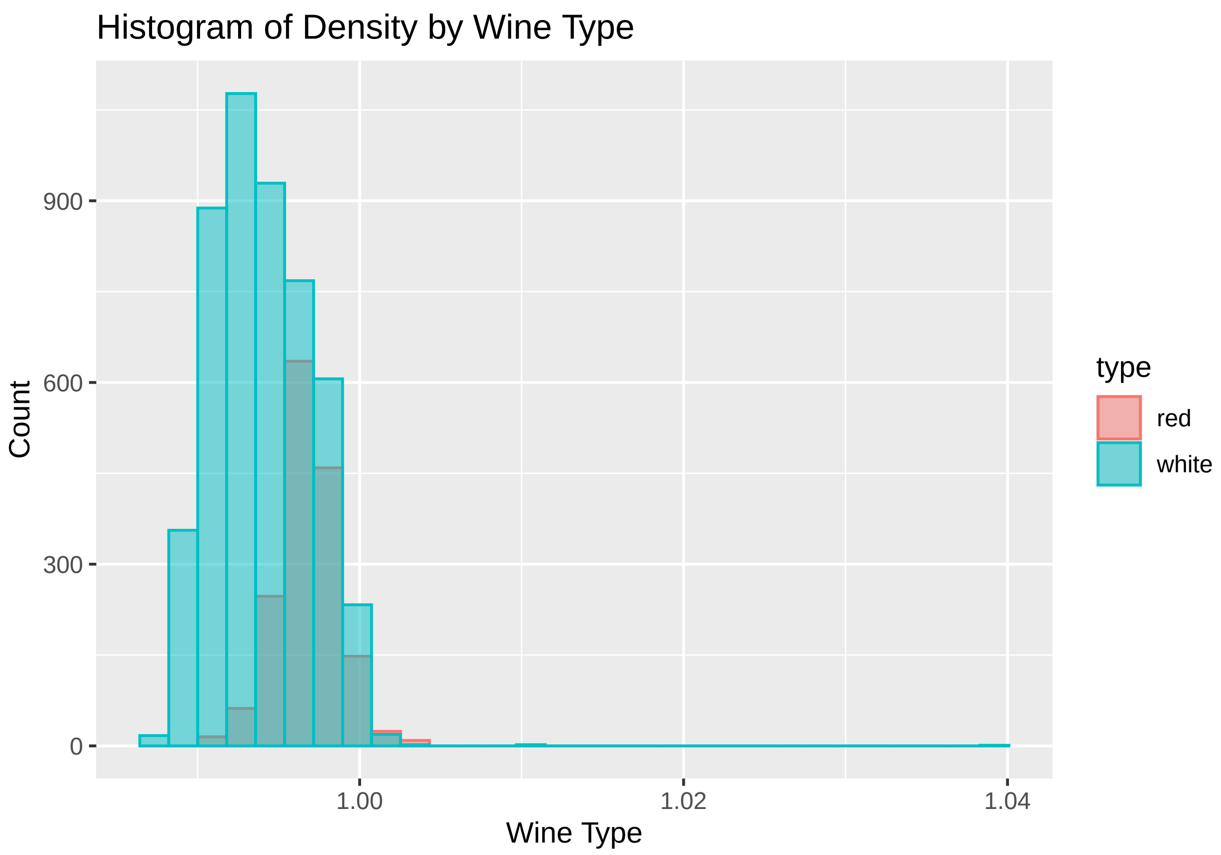


```r
# Box plot Density grouped by Wine Type
ggplot(wine, aes(x = type, y = density)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Density",
       title = "Box Plot of Density by Wine Type")
```

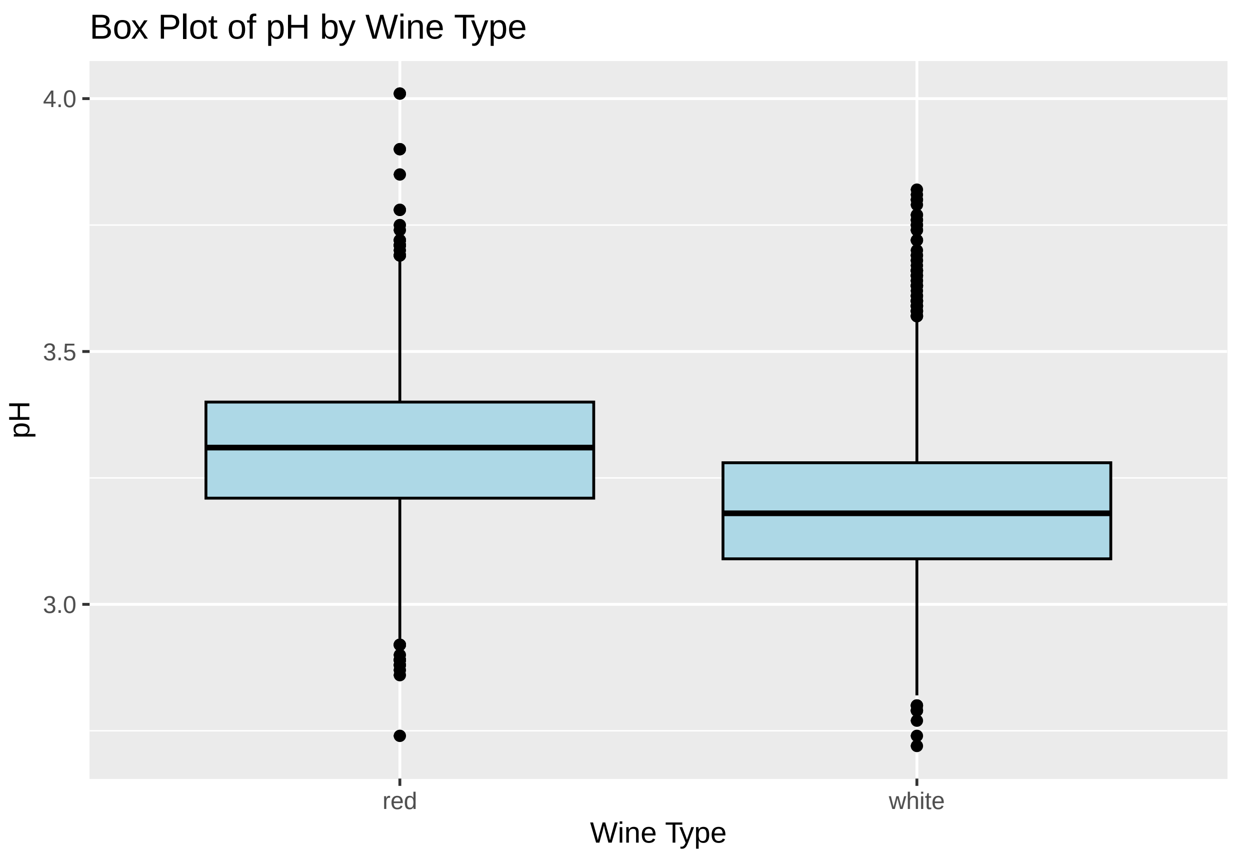


```r
# Histogram of Density with bars colored by Wine Type
ggplot(wine, aes(x = density, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Density by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

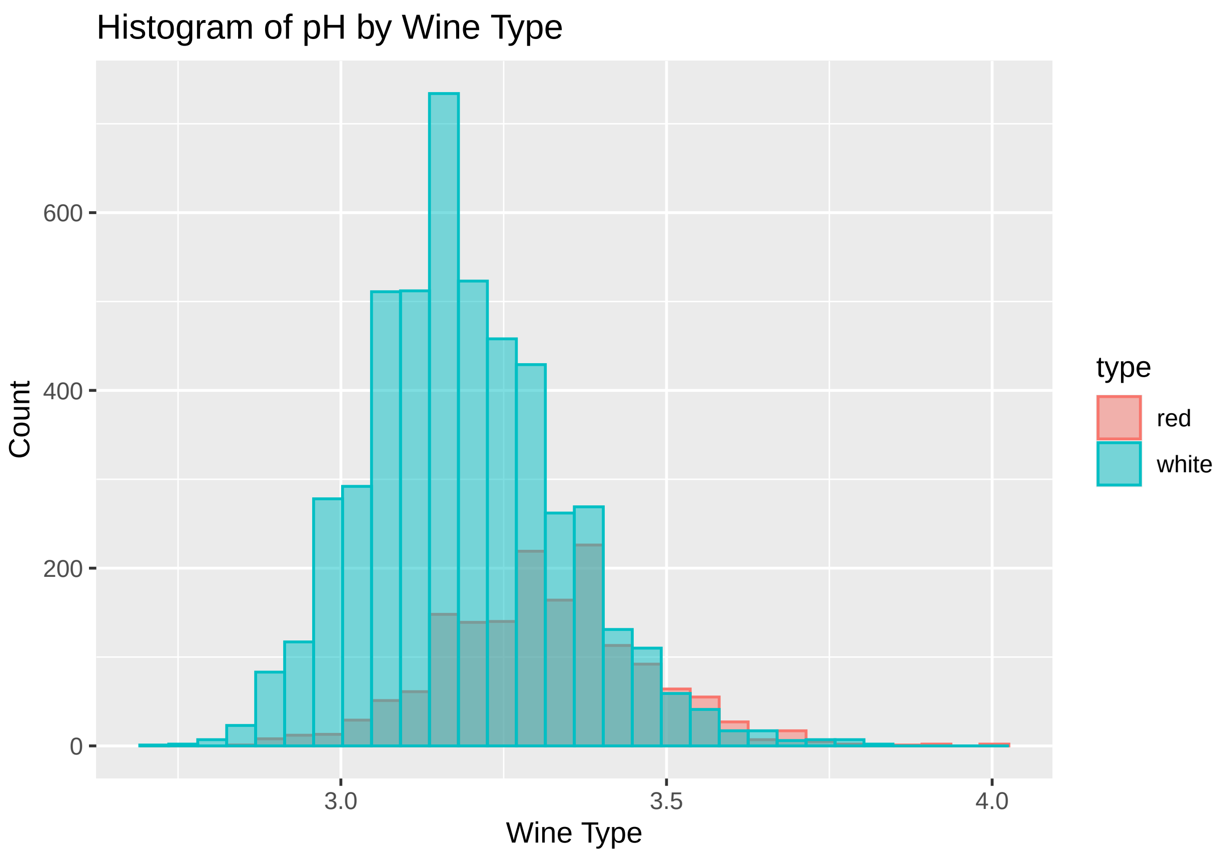


```r
# Box plot pH grouped by Wine Type
ggplot(wine, aes(x = type, y = pH)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "pH", title = "Box Plot of pH by Wine Type")
```

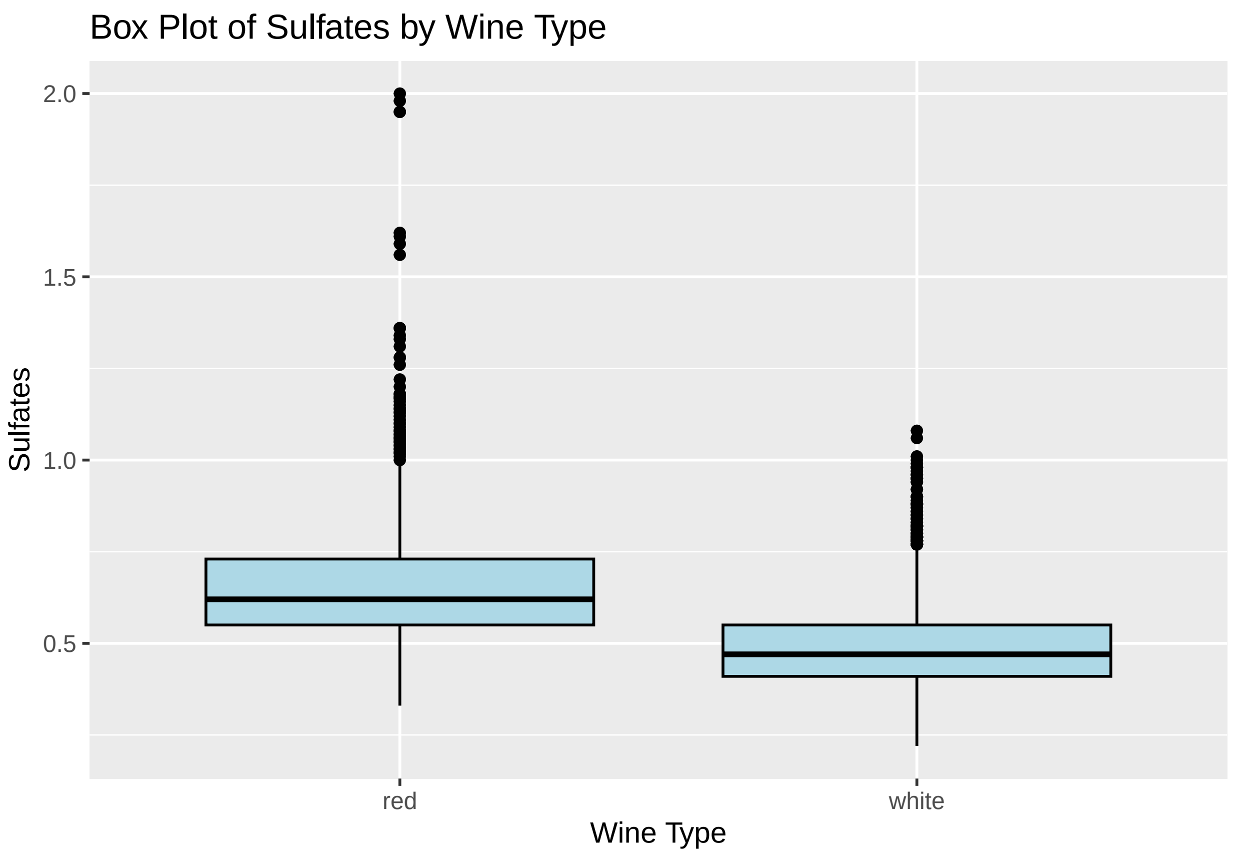


```r
# Histogram of pH with bars colored by Wine Type
ggplot(wine, aes(x = pH, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count", title = "Histogram of pH by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

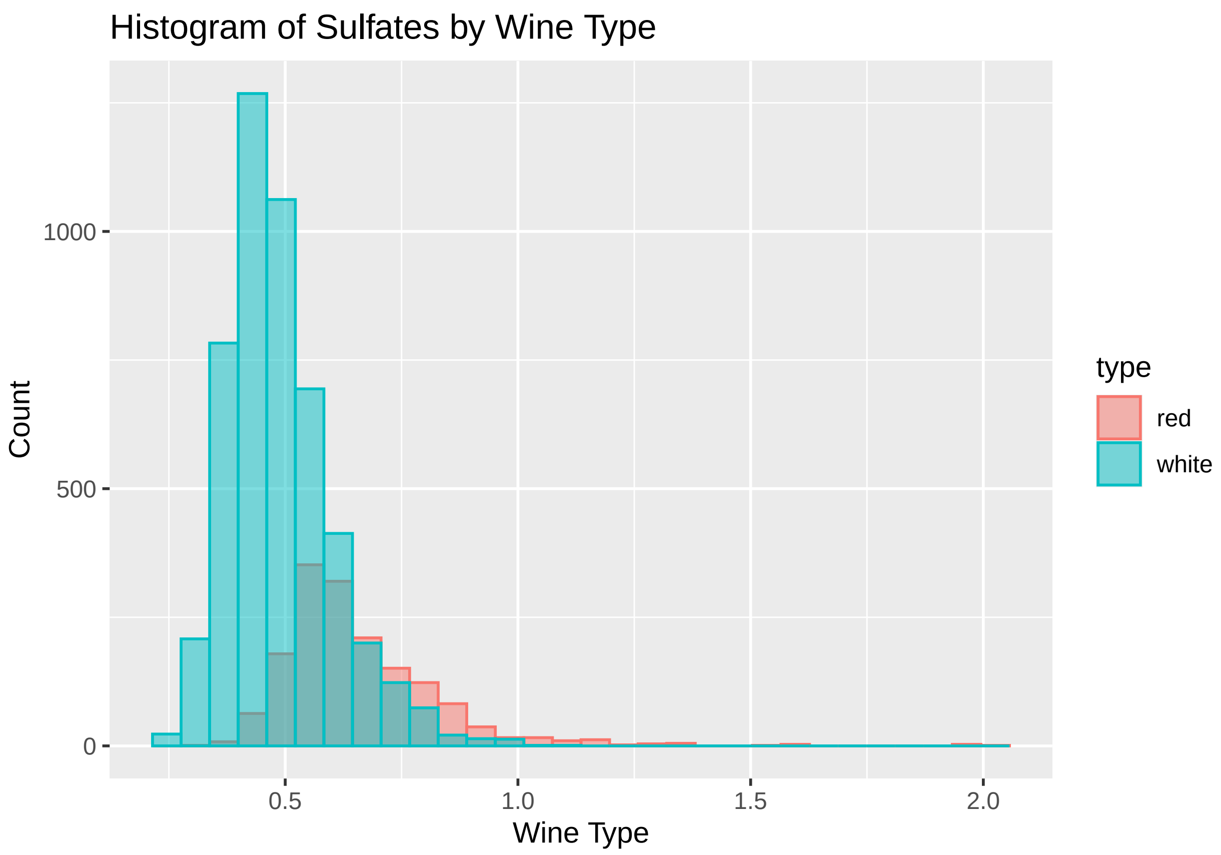


```r
# Box plot Sulfates grouped by Wine Type
ggplot(wine, aes(x = type, y = sulphates)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Sulfates",
       title = "Box Plot of Sulfates by Wine Type")
```

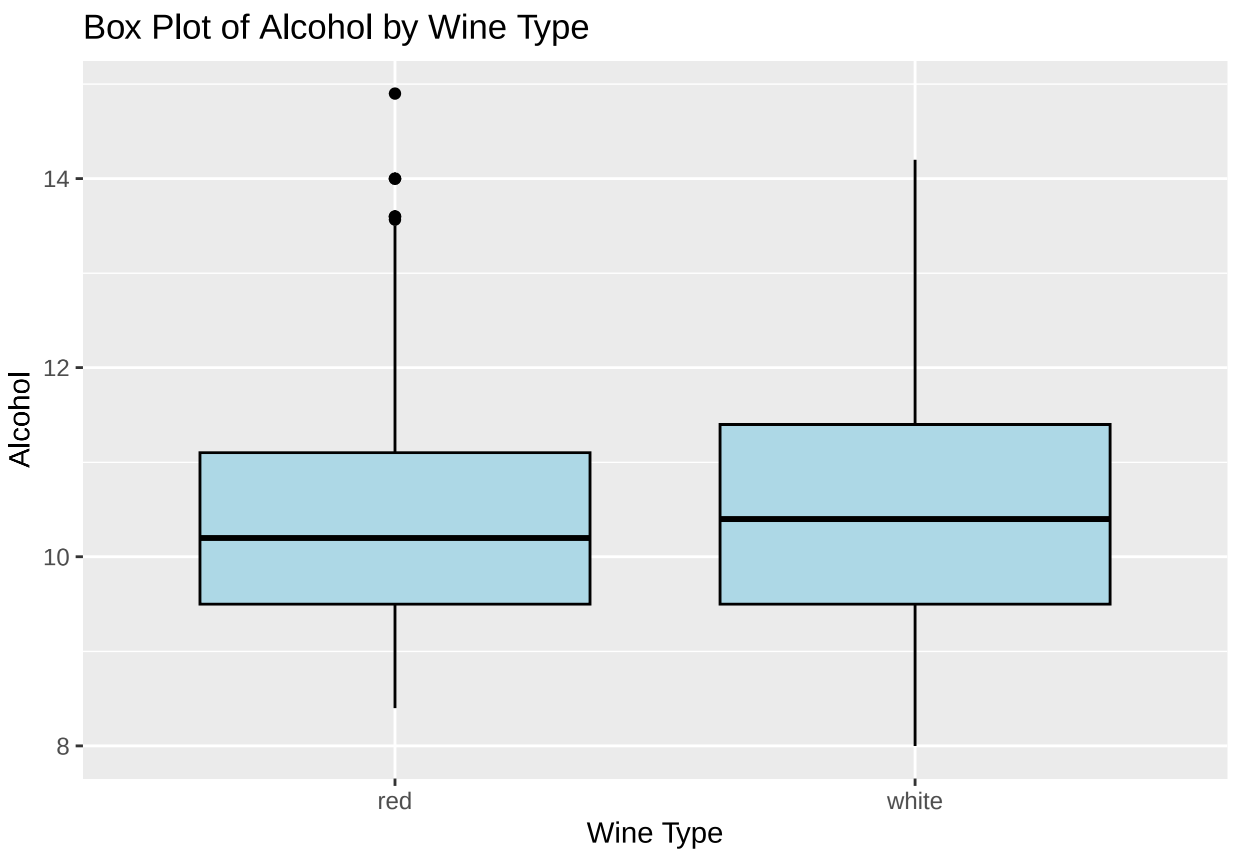


```r
# Histogram of Sulfates with bars colored by Wine Type
ggplot(wine, aes(x = sulphates, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Sulfates by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

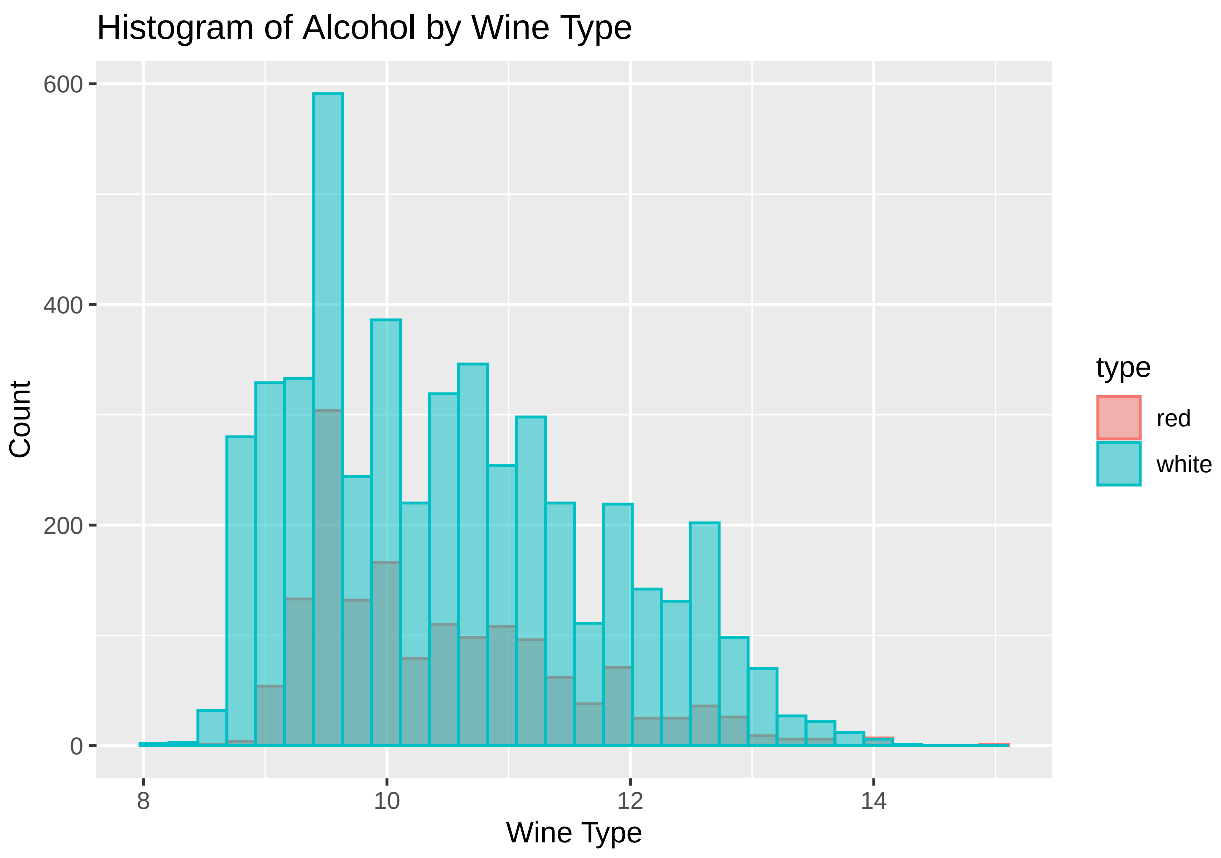


```r
# Box plot Alcohol grouped by Wine Type
ggplot(wine, aes(x = type, y = alcohol)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Alcohol",
       title = "Box Plot of Alcohol by Wine Type")
```

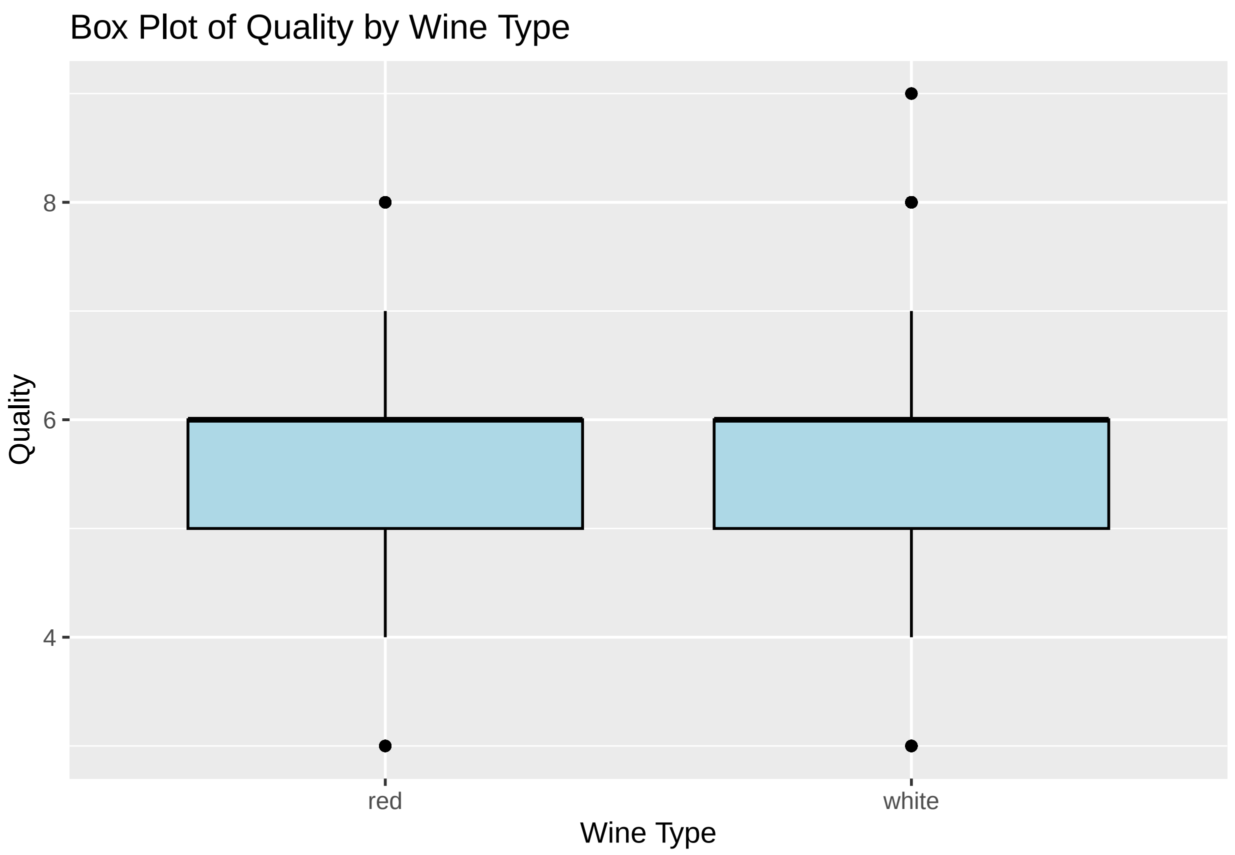


```r
# Histogram of Alcohol with bars colored by Wine Type
ggplot(wine, aes(x = alcohol, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "Histogram of Alcohol by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

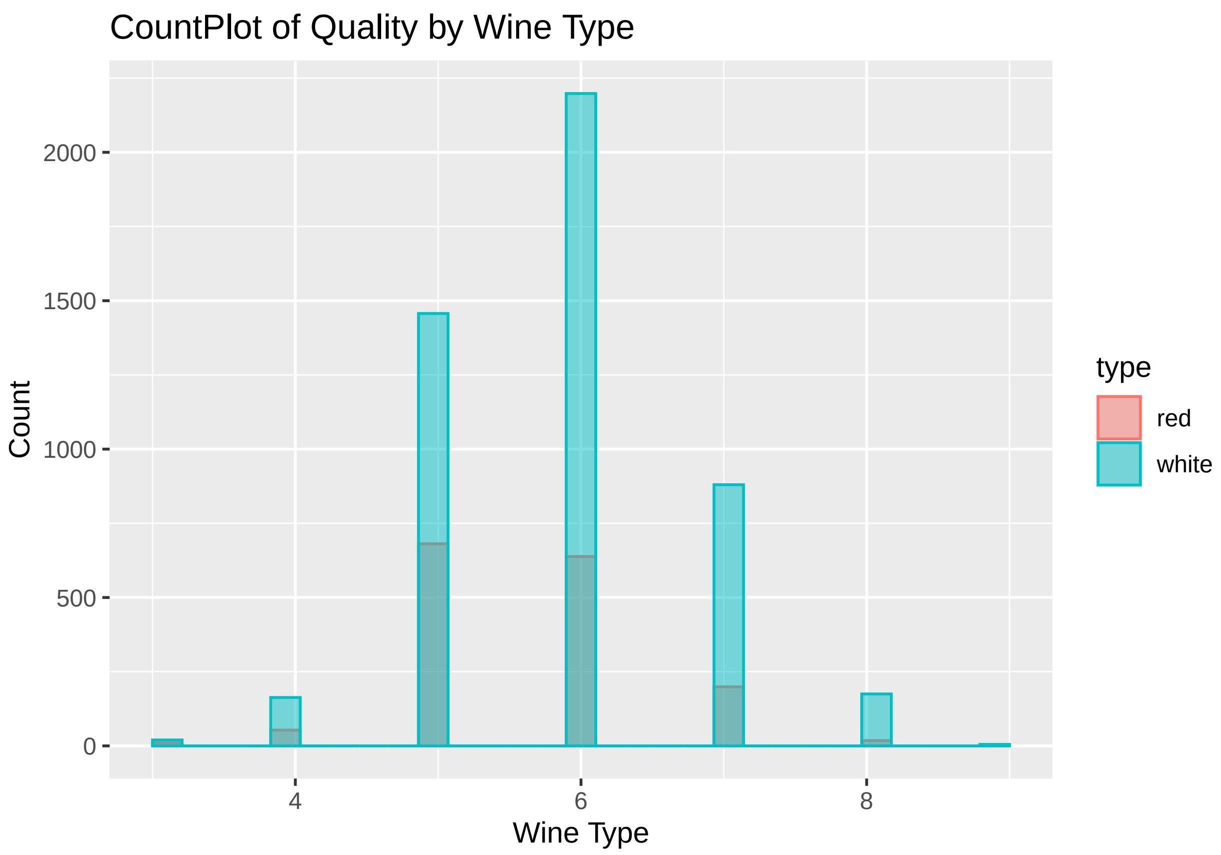


```r
# Box plot Quality grouped by Wine Type
ggplot(wine, aes(x = type, y = quality)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(x = "Wine Type", y = "Quality",
       title = "Box Plot of Quality by Wine Type")
```



```
# CountPlot of Quality with bars colored by Wine Type
ggplot(wine, aes(x = quality, fill = type,colour = type)) +
  geom_histogram(alpha = 0.5, position = "identity") +
  labs(x = "Wine Type", y = "Count",
       title = "CountPlot of Quality by Wine Type")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

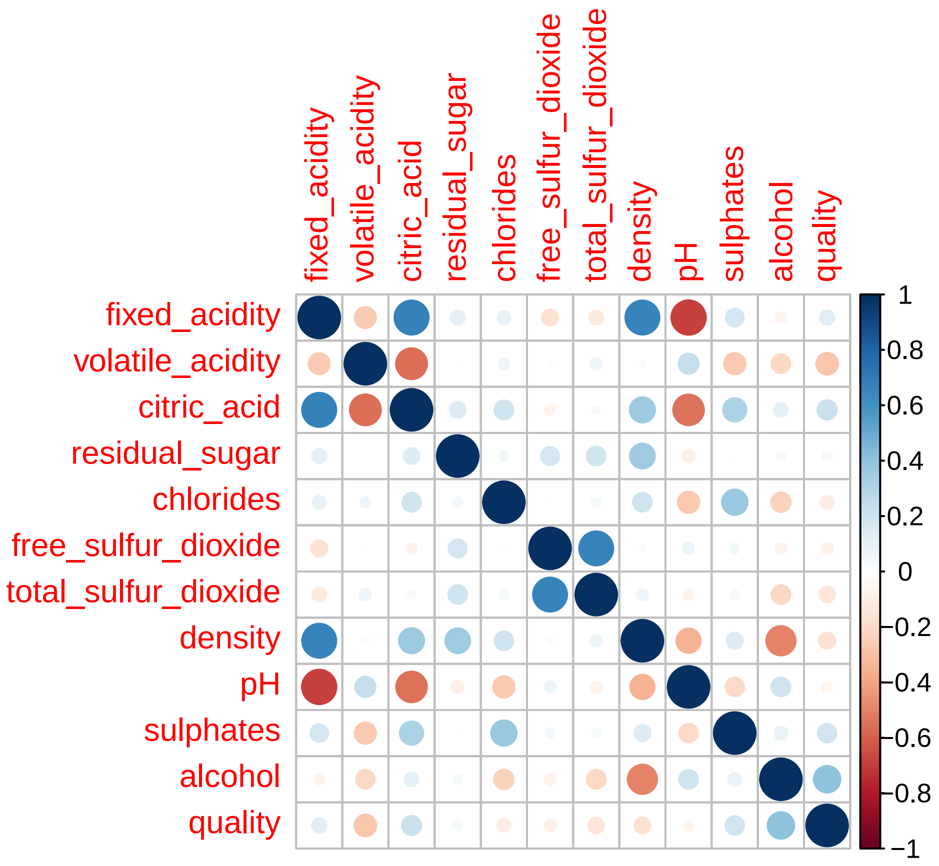


```r
cat("Examining the CountPlot of Wine Type, the purpose of this investigation
    since less than 25% of wines in both red and white wines have high wine
    ratings greater than 7, Grade A wine is defined as those with an rating of 7
    or greater. The purpose of this investigation is to predict Grade A wine
    and be able to define characteristics of Grade A wine")
```

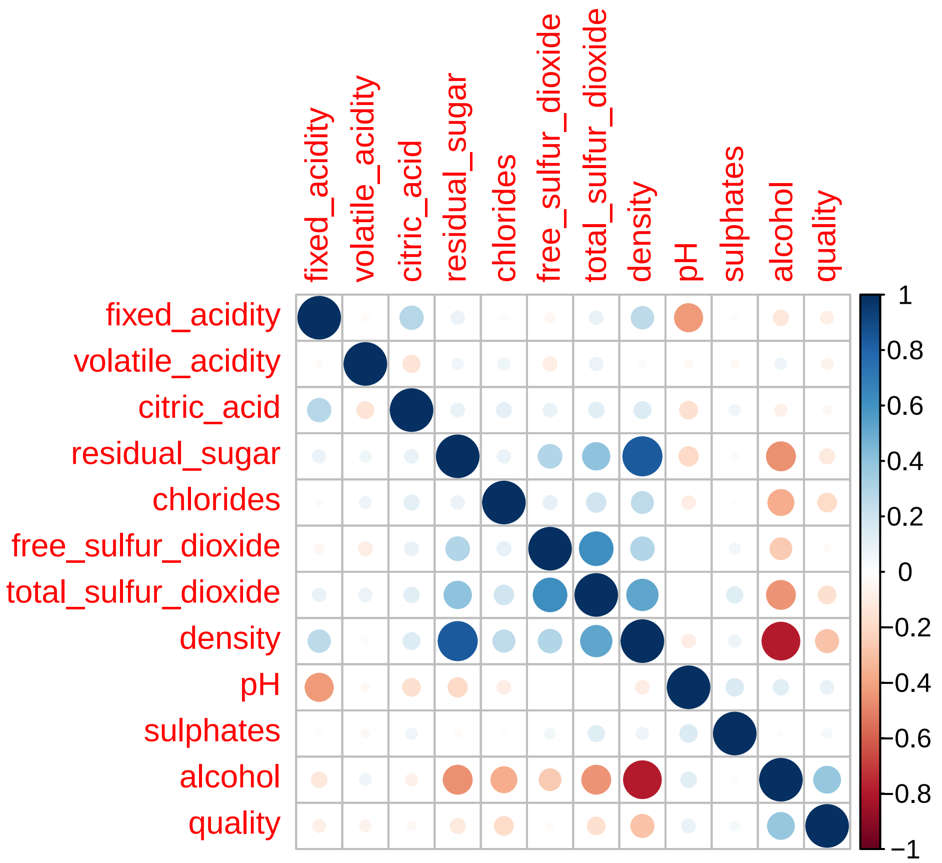
```
## Examining the CountPlot of Wine Type, the purpose of this investigation
##     since less than 25% of wines in both red and white wines have high wine
##     ratings greater than 7, Grade A wine is defined as those with an rating of 7
##     or greater. The purpose of this investigation is to predict Grade A wine
##     and be able to define characteristics of Grade A wine
```

```r
redwine_seperated$quality <- ifelse(redwine_seperated$quality >= 7,1,0)

whitewine_seperated$quality <- ifelse(whitewine_seperated$quality >= 7,1,0)
```

```r
#EDA Continued
#Checking Correlation of all Variables with correlation plot

#Correlation Plot of all red wine variables
corrplot(cor(redwine_seperated[1:12]))
```



```
#Correlation Plot of all white wine variables
corr_plot_white <- corrplot(cor(whitewine_seperated[1:12]))
```



```
corr_plot_white
```

```
## $corr
##                      fixed_acidity volatile_acidity citric_acid residual_sugar
## fixed_acidity           1.00000000      -0.02269729  0.28918070     0.08902070
## volatile_acidity       -0.02269729       1.00000000 -0.14947181     0.06428606
## citric_acid             0.28918070      -0.14947181  1.00000000     0.09421162
## residual_sugar          0.08902070       0.06428606  0.09421162     1.00000000
## chlorides               0.02308564       0.07051157  0.11436445     0.08868454
## free_sulfur_dioxide    -0.04939586      -0.09701194  0.09407722     0.29909835
## total_sulfur_dioxide    0.09106976       0.08926050  0.12113080     0.40143931
## density                 0.26533101       0.02711385  0.14950257     0.83896645
## pH                     -0.42585829      -0.03191537 -0.16374821    -0.19413345
## sulphates              -0.01714299      -0.03572815  0.06233094    -0.02666437
## alcohol                -0.12088112       0.06771794 -0.07572873    -0.45063122
## quality                -0.08074763      -0.06722490 -0.03532976    -0.11708539
##                        chlorides free_sulfur_dioxide total_sulfur_dioxide
## fixed_acidity         0.02308564       -0.0493958591          0.091069756
## volatile_acidity      0.07051157       -0.0970119393          0.089260504
## citric_acid           0.11436445        0.0940772210          0.121130798
## residual_sugar        0.08868454        0.2990983537          0.401439311
## chlorides             1.00000000        0.1013923521          0.198910300
## free_sulfur_dioxide   0.10139235        1.0000000000          0.615500965
## total_sulfur_dioxide  0.19891030        0.6155009650          1.000000000
## density               0.25721132        0.2942104109          0.529881324
## pH                   -0.09043946       -0.0006177961          0.002320972
## sulphates             0.01676288        0.0592172458          0.134562367
## alcohol              -0.36018871       -0.2501039415         -0.448892102
## quality              -0.18311811       -0.0234132186         -0.162202045
##                          density            pH   sulphates     alcohol
## fixed_acidity         0.26533101 -0.4258582910 -0.01714299 -0.12088112
## volatile_acidity      0.02711385 -0.0319153683 -0.03572815  0.06771794
## citric_acid           0.14950257 -0.1637482114  0.06233094 -0.07572873
## residual_sugar        0.83896645 -0.1941334540 -0.02666437 -0.45063122
## chlorides             0.25721132 -0.0904394560  0.01676288 -0.36018871
## free_sulfur_dioxide   0.29421041 -0.0006177961  0.05921725 -0.25010394
## total_sulfur_dioxide  0.52988132  0.0023209718  0.13456237 -0.44889210
## density               1.00000000 -0.0935914935  0.07449315 -0.78013762
## pH                   -0.09359149  1.0000000000  0.15595150  0.12143210
## sulphates             0.07449315  0.1559514973  1.00000000 -0.01743277
## alcohol              -0.78013762  0.1214320987 -0.01743277  1.00000000
## quality              -0.28387080  0.0935104200  0.04741019  0.38513160
##                          quality
## fixed_acidity        -0.08074763
## volatile_acidity     -0.06722490
## citric_acid          -0.03532976
## residual_sugar       -0.11708539
## chlorides            -0.18311811
## free_sulfur_dioxide  -0.02341322
## total_sulfur_dioxide -0.16220205
## density              -0.28387080
## pH                    0.09351042
## sulphates             0.04741019
## alcohol               0.38513160
## quality               1.00000000
##
## $corrPos
##                 xName                yName  x  y          corr
## 1           fixed_acidity        fixed_acidity  1 12  1.0000000000
## 2           fixed_acidity     volatile_acidity  1 11 -0.0226972901
## 3           fixed_acidity          citric_acid  1 10  0.2891806977
## 4           fixed_acidity       residual_sugar  1  9  0.0890207014
## 5           fixed_acidity            chlorides  1  8  0.0230856437
## 6           fixed_acidity  free_sulfur_dioxide  1  7 -0.0493958591
## 7           fixed_acidity total_sulfur_dioxide  1  6  0.0910697562
## 8           fixed_acidity              density  1  5  0.2653310138
## 9           fixed_acidity                   pH  1  4 -0.4258582910
## 10          fixed_acidity            sulphates  1  3 -0.0171429850
## 11          fixed_acidity              alcohol  1  2 -0.1208811232
## 12          fixed_acidity              quality  1  1 -0.0807476338
## 13       volatile_acidity        fixed_acidity  2 12 -0.0226972901
## 14       volatile_acidity     volatile_acidity  2 11  1.0000000000
## 15       volatile_acidity          citric_acid  2 10 -0.1494718106
## 16       volatile_acidity       residual_sugar  2  9  0.0642860601
## 17       volatile_acidity            chlorides  2  8  0.0705115715
## 18       volatile_acidity  free_sulfur_dioxide  2  7 -0.0970119393
## 19       volatile_acidity total_sulfur_dioxide  2  6  0.0892605036
## 20       volatile_acidity              density  2  5  0.0271138455
## 21       volatile_acidity                   pH  2  4 -0.0319153683
## 22       volatile_acidity            sulphates  2  3 -0.0357281469
## 23       volatile_acidity              alcohol  2  2  0.0677179428
## 24      volatile_acidity             quality  2  1 -0.0672248954
## 25           citric_acid       fixed_acidity  3 12  0.2891806977
## 26           citric_acid    volatile_acidity  3 11 -0.1494718106
## 27           citric_acid         citric_acid  3 10  1.0000000000
## 28           citric_acid       residual_sugar  3  9  0.0942116243
## 29           citric_acid           chlorides  3  8  0.1143644484
## 30           citric_acid  free_sulfur_dioxide  3  7  0.0940772210
## 31           citric_acid total_sulfur_dioxide  3  6  0.1211307977
## 32           citric_acid             density  3  5  0.1495025706
## 33           citric_acid                  pH  3  4 -0.1637482114
## 34           citric_acid           sulphates  3  3  0.0623309403
## 35           citric_acid             alcohol  3  2 -0.0757287301
## 36           citric_acid             quality  3  1 -0.0353297624
## 37        residual_sugar       fixed_acidity  4 12  0.0890207014
## 38        residual_sugar    volatile_acidity  4 11  0.0642860601
## 39        residual_sugar         citric_acid  4 10  0.0942116243
## 40        residual_sugar       residual_sugar  4  9  1.0000000000
## 41        residual_sugar           chlorides  4  8  0.0886845359
## 42        residual_sugar  free_sulfur_dioxide  4  7  0.2990983537
## 43        residual_sugar total_sulfur_dioxide  4  6  0.4014393112
## 44        residual_sugar             density  4  5  0.8389664549
## 45        residual_sugar                  pH  4  4 -0.1941334540
## 46        residual_sugar           sulphates  4  3 -0.0266643659
## 47        residual_sugar             alcohol  4  2 -0.4506312220
## 48        residual_sugar             quality  4  1 -0.1170853851
## 49             chlorides       fixed_acidity  5 12  0.0230856437
## 50             chlorides    volatile_acidity  5 11  0.0705115715
## 51             chlorides         citric_acid  5 10  0.1143644484
## 52             chlorides       residual_sugar  5  9  0.0886845359
## 53             chlorides           chlorides  5  8  1.0000000000
## 54             chlorides  free_sulfur_dioxide  5  7  0.1013923521
## 55             chlorides total_sulfur_dioxide  5  6  0.1989102996
## 56             chlorides             density  5  5  0.2572113204
## 57             chlorides                  pH  5  4 -0.0904394560
## 58             chlorides           sulphates  5  3  0.0167628837
## 59             chlorides             alcohol  5  2 -0.3601887121
## 60             chlorides             quality  5  1 -0.1831181101
## 61   free_sulfur_dioxide       fixed_acidity  6 12 -0.0493958591
## 62   free_sulfur_dioxide    volatile_acidity  6 11 -0.0970119393
## 63   free_sulfur_dioxide         citric_acid  6 10  0.0940772210
## 64   free_sulfur_dioxide       residual_sugar  6  9  0.2990983537
## 65   free_sulfur_dioxide           chlorides  6  8  0.1013923521
## 66   free_sulfur_dioxide  free_sulfur_dioxide  6  7  1.0000000000
## 67   free_sulfur_dioxide total_sulfur_dioxide  6  6  0.6155009650
## 68   free_sulfur_dioxide             density  6  5  0.2942104109
## 69   free_sulfur_dioxide                  pH  6  4 -0.0006177961
## 70   free_sulfur_dioxide           sulphates  6  3  0.0592172458
## 71   free_sulfur_dioxide             alcohol  6  2 -0.2501039415
## 72   free_sulfur_dioxide             quality  6  1 -0.0234132186
## 73  total_sulfur_dioxide       fixed_acidity  7 12  0.0910697562
## 74  total_sulfur_dioxide    volatile_acidity  7 11  0.0892605036
## 75  total_sulfur_dioxide         citric_acid  7 10  0.1211307977
## 76  total_sulfur_dioxide       residual_sugar  7  9  0.4014393112
## 77  total_sulfur_dioxide           chlorides  7  8  0.1989102996
## 78  total_sulfur_dioxide  free_sulfur_dioxide  7  7  0.6155009650
## 79  total_sulfur_dioxide total_sulfur_dioxide  7  6  1.0000000000
## 80  total_sulfur_dioxide              density  7  5  0.5298813239
## 81  total_sulfur_dioxide                   pH  7  4  0.0023209718
## 82  total_sulfur_dioxide            sulphates  7  3  0.1345623669
## 83  total_sulfur_dioxide              alcohol  7  2 -0.4488921021
## 84  total_sulfur_dioxide              quality  7  1 -0.1622020454
## 85               density        fixed_acidity  8 12  0.2653310138
## 86               density     volatile_acidity  8 11  0.0271138455
## 87               density          citric_acid  8 10  0.1495025706
## 88               density       residual_sugar  8  9  0.8389664549
## 89               density            chlorides  8  8  0.2572113204
## 90               density  free_sulfur_dioxide  8  7  0.2942104109
## 91               density total_sulfur_dioxide  8  6  0.5298813239
## 92               density              density  8  5  1.0000000000
## 93               density                   pH  8  4 -0.0935914935
## 94               density            sulphates  8  3  0.0744931485
## 95               density              alcohol  8  2 -0.7801376214
## 96               density              quality  8  1 -0.2838707967
## 97                    pH        fixed_acidity  9 12 -0.4258582910
## 98                    pH     volatile_acidity  9 11 -0.0319153683
## 99                    pH          citric_acid  9 10 -0.1637482114
## 100                   pH       residual_sugar  9  9 -0.1941334540
## 101                   pH            chlorides  9  8 -0.0904394560
## 102                   pH  free_sulfur_dioxide  9  7 -0.0006177961
## 103                   pH total_sulfur_dioxide  9  6  0.0023209718
## 104                   pH              density  9  5 -0.0935914935
## 105                   pH                   pH  9  4  1.0000000000
## 106                   pH            sulphates  9  3  0.1559514973
## 107                   pH              alcohol  9  2  0.1214320987
## 108                   pH              quality  9  1  0.0935104200
## 109            sulphates        fixed_acidity 10 12 -0.0171429850
## 110            sulphates     volatile_acidity 10 11 -0.0357281469
## 111            sulphates          citric_acid 10 10  0.0623309403
## 112            sulphates       residual_sugar 10  9 -0.0266643659
## 113            sulphates            chlorides 10  8  0.0167628837
## 114            sulphates  free_sulfur_dioxide 10  7  0.0592172458
## 115            sulphates total_sulfur_dioxide 10  6  0.1345623669
## 116            sulphates              density 10  5  0.0744931485
## 117            sulphates                   pH 10  4  0.1559514973
## 118            sulphates            sulphates 10  3  1.0000000000
## 119            sulphates              alcohol 10  2 -0.0174327719
## 120            sulphates              quality 10  1  0.0474101902
## 121              alcohol        fixed_acidity 11 12 -0.1208811232
## 122              alcohol     volatile_acidity 11 11  0.0677179428
## 123              alcohol          citric_acid 11 10 -0.0757287301
## 124              alcohol       residual_sugar 11  9 -0.4506312220
## 125              alcohol            chlorides 11  8 -0.3601887121
## 126              alcohol  free_sulfur_dioxide 11  7 -0.2501039415
## 127              alcohol total_sulfur_dioxide 11  6 -0.4488921021
## 128              alcohol              density 11  5 -0.7801376214
## 129              alcohol                   pH 11  4  0.1214320987
## 130              alcohol            sulphates 11  3 -0.0174327719
## 131              alcohol              alcohol 11  2  1.0000000000
## 132              alcohol              quality 11  1  0.3851316042
## 133              quality        fixed_acidity 12 12 -0.0807476338
## 134              quality     volatile_acidity 12 11 -0.0672248954
## 135              quality          citric_acid 12 10 -0.0353297624
## 136              quality       residual_sugar 12  9 -0.1170853851
## 137              quality            chlorides 12  8 -0.1831181101
## 138              quality  free_sulfur_dioxide 12  7 -0.0234132186
## 139              quality total_sulfur_dioxide 12  6 -0.1622020454
## 140              quality              density 12  5 -0.2838707967
## 141              quality                   pH 12  4  0.0935104200
## 142              quality            sulphates 12  3  0.0474101902
## 143              quality              alcohol 12  2  0.3851316042
## 144              quality              quality 12  1  1.0000000000
##
## $arg
## $arg$type
## [1] "full"
```

```
cat("Since summary statistics showed that variables were on different scales
    and there was a lot of difference observed between variables must apply
    scaling method to normalize data.")
```

```
## Since summary statistics showed that variables were on different scales
##     and there was a lot of difference observed between variables must apply
##     scaling method to normalize data.
```

```
#Applying Min-max scaling
min_max_scaling_white <- preProcess(whitewine_seperated[1:11], method = "range")
white_wine_scaled <- predict(min_max_scaling_white,whitewine_seperated)
white_wine_scaled$quality <- as.factor(white_wine_scaled$quality)

min_max_scaling_red <- preProcess(redwine_seperated[1:11], method = "range")
red_wine_scaled <- predict(min_max_scaling_red,redwine_seperated)
red_wine_scaled$quality <- as.factor(red_wine_scaled$quality)



redwine_randomforest_columns12 <- c("fixed_acidity", "volatile_acidity",
                                    "citric_acid","residual_sugar","chlorides",
                                    "free_sulfur_dioxide","total_sulfur_dioxide",
                            "density","pH","sulphates", "alcohol", "quality")
whitewine_randomforest_columns12 <- c("fixed_acidity", "volatile_acidity",
                                      "citric_acid","residual_sugar","chlorides",
                                      "free_sulfur_dioxide","total_sulfur_dioxide",
                            "density","pH","sulphates", "alcohol", "quality")

red_wine_scaled <- red_wine_scaled[, redwine_randomforest_columns12,
                                   drop = FALSE ]
white_wine_scaled <- white_wine_scaled[, whitewine_randomforest_columns12,
                                       drop = FALSE ]
```

```
cat("Splitting Data into train/test")
```

```
## Splitting Data into train/test
```

```
# Randomly shuffling the data and dividing into train/test
white_wine_indexes <- sample(2, nrow(white_wine_scaled),
```

```r
                             replace = TRUE, prob = c(0.8,0.2))
white_wine_train <- white_wine_scaled[white_wine_indexes==1,]
white_wine_test <- white_wine_scaled[white_wine_indexes==2,]


red_wine_indexes <- sample(2, nrow(red_wine_scaled),
                           replace = TRUE, prob = c(0.8,0.2))
red_wine_train <- red_wine_scaled[red_wine_indexes==1,]
red_wine_test <- red_wine_scaled[red_wine_indexes==2,]



# Set up 30 random train/test splits for white and red wine data
set.seed(123) # for reproducibility

# Generate indexes for 30 iterations
white_wine_indexes_list <- replicate(31, sample(2,
                                                nrow(white_wine_scaled),
                                                replace = TRUE,
                                                prob = c(0.8, 0.2)),
                                     simplify = FALSE)

red_wine_indexes_list <- replicate(31, sample(2, nrow(red_wine_scaled),
                                              replace = TRUE,
                                              prob = c(0.8, 0.2)),
                                   simplify = FALSE)

# Vectorized approach with lapply
white_wine_train_list <- lapply(white_wine_indexes_list, function(index) white_wine_scaled[index == 1,
white_wine_test_list <- lapply(white_wine_indexes_list, function(index) white_wine_scaled[index == 2, ]

red_wine_train_list <- lapply(red_wine_indexes_list, function(index) red_wine_scaled[index == 1, ])
red_wine_test_list <- lapply(red_wine_indexes_list, function(index) red_wine_scaled[index == 2, ])


cat("Since data is very unbalanced with Grade A Wine
    representing less than 25% of respective wine types
    randomly sampling with replacement fom original
    data to synthetically replicate minority class
    of Grade A Wine in both red wine and white wine
    data so that model can pick up complex relationships")
```

```
## Since data is very unbalanced with Grade A Wine
##     representing less than 25% of respective wine types
##     randomly sampling with replacement fom original
##     data to synthetically replicate minority class
##     of Grade A Wine in both red wine and white wine
##     data so that model can pick up complex relationships
```

```r
# Define oversampling functions
oversample_data_red <- function(my_data) {
  data <- my_data
  return(ovun.sample(quality ~ ., data = data, method = "over", N = 2150)$data)
}
```

```r
oversample_data_white <- function(my_data) {
  data <- my_data
  return(ovun.sample(quality ~ ., data = data, method = "over", N = 6150)$data)
}


# Applying oversampling to all training sets
oversampled_red_wine_train_list <- lapply(red_wine_train_list,
                                          oversample_data_red)

oversampled_white_wine_train_list <- lapply(white_wine_train_list,
                                            oversample_data_white)
```

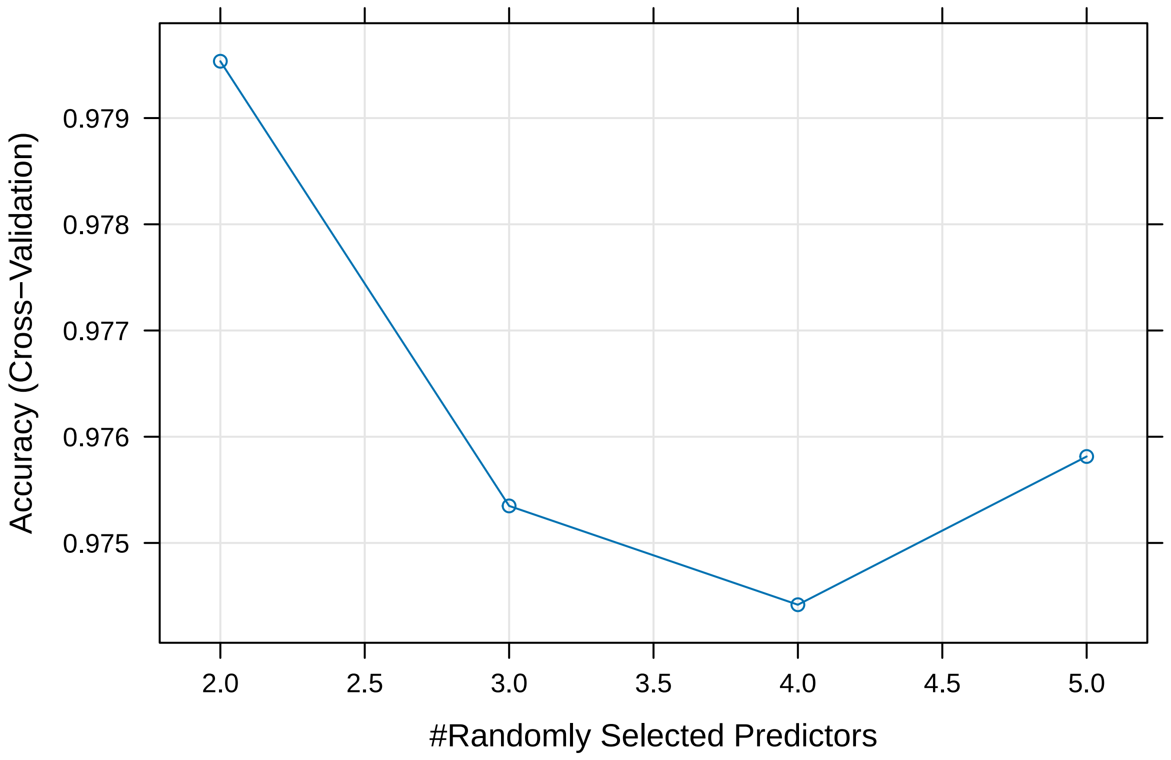
```r
#Random Forest Model for predicting Grade A Red Wine

cat("Calling extra sample storing in data frame and using cross validation and
    grid search to find optimal parameters")
```

```
## Calling extra sample storing in data frame and using cross validation and
##     grid search to find optimal parameters
```

```r
red_wine_rf_extra <- oversampled_red_wine_train_list[[31]]

# Define the control for grid search with 10-fold cross-validation
train_control <- trainControl(method = "cv", number = 10)

# Define the grid of hyper-parameters to tune
tune_grid <- expand.grid(
  mtry = c(2,3,4,5))

cat("Training the Random Forest model using grid search and
    10-fold cross-validation for Red Wine")
```

```
## Training the Random Forest model using grid search and
##     10-fold cross-validation for Red Wine
```

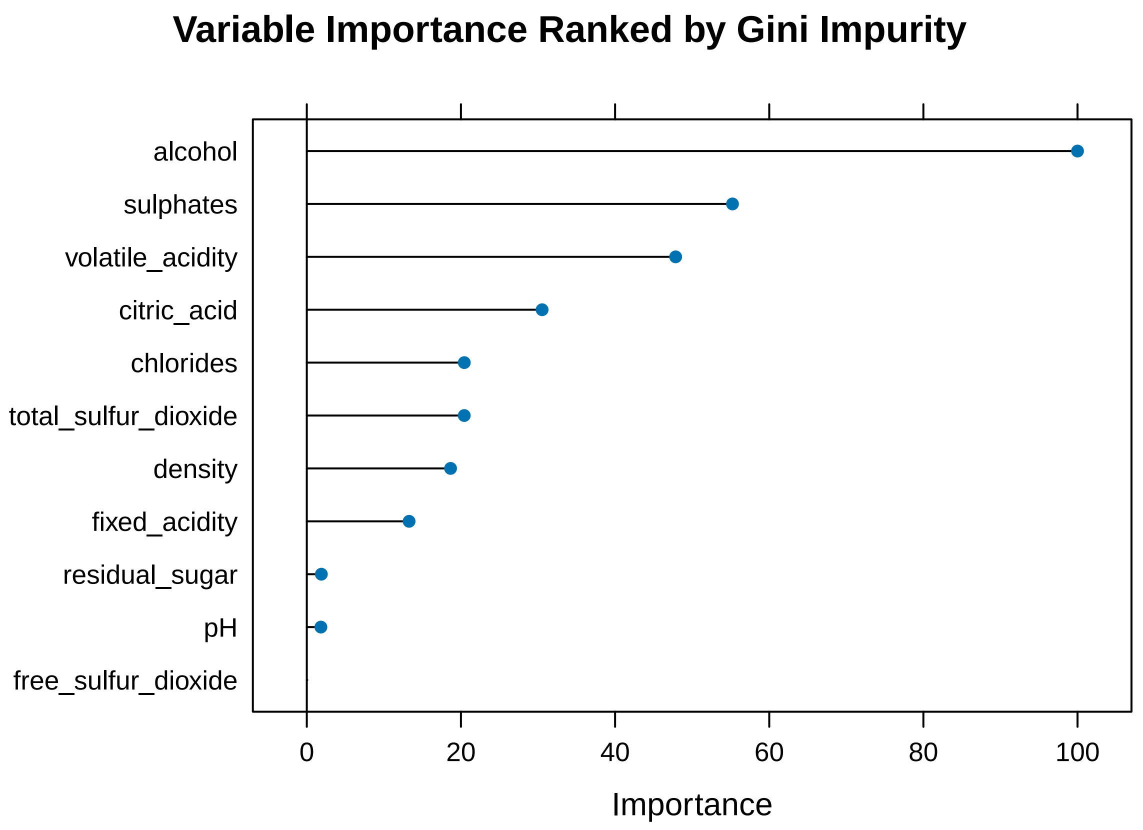
```r
rf_gridsearch_red <- caret::train(quality ~ .,
                        red_wine_rf_extra,
                        method = "rf",
                        trControl = train_control,
                        tuneGrid = tune_grid,
                        importance = TRUE)
plot(rf_gridsearch_red)
```



```
cat("This plot shows that optimal number of variables
    to try at every node split is 2")
```

```
## This plot shows that optimal number of variables
##     to try at every node split is 2
```

```
#Variable Importance Plot of model predicting Grade A red wine
rf_gridsearch_red_importance <- varImp(rf_gridsearch_red, type = 2)
plot(rf_gridsearch_red_importance,
     main = "Variable Importance Ranked by Gini Impurity")
```



```
cat("This plot shows that the elbow of the importance plot is at the fifth most
    important variable so the remaining variables are dropped from future model,
    these variables are: total_sulfur_dioxide, density, fixed_acidity,
    residual_sugar, pH, free_sulfur_dioxide")
```

```
## This plot shows that the elbow of the importance plot is at the fifth most
##     important variable so the remaining variables are dropped from future model,
##     these variables are: total_sulfur_dioxide, density, fixed_acidity,
##     residual_sugar, pH, free_sulfur_dioxide
```

```
drop_columns <- c("total_sulfur_dioxide","density","fixed_acidity",
                  "residual_sugar","pH","free_sulfur_dioxide")

oversampled_red_wine_train_list <- lapply(oversampled_red_wine_train_list, function(df) {
  df %>% dplyr::select(-all_of(drop_columns))
})


cat("Examining the grid-search's plot it shows the optimal number of variables
    to randomly sample from at every node split is 2, now applying Random Forest
    Model with optimal parameter 30 times, since this is very time consuming
    using parallel processing")
```

```
## Examining the grid-search's plot it shows the optimal number of variables
##     to randomly sample from at every node split is 2, now applying Random Forest
##     Model with optimal parameter 30 times, since this is very time consuming
##     using parallel processing
```

```r
#Creating empty lists
accuracy_vector_red <- numeric(length(1:30))
conf_mat_list_red <- vector("list",length(1:30))
variable_importance_list_red <- vector("list",length(1:30))

tune_grid2 <- expand.grid(mtry = 2)

#initializing parallel processing
num_cores <- detectCores() - 2
cl <- makePSOCKcluster(num_cores)
registerDoParallel(cl)


results <- foreach (i = 1:length(oversampled_red_wine_train_list),
                    .packages = c("caret", "dplyr")) %dopar% {
# Training the Random Forest model with 30 times
  rf_model_red <- caret::train(
    quality ~ .,
    data = oversampled_red_wine_train_list[[i]],
    method = "rf",
    tuneGrid = tune_grid2,
    importance = TRUE
  )

#Confusion Matrix of final model predicting Grade A red wine
predictions_red <- predict(rf_model_red, newdata = red_wine_test_list[[i]])
confusion_mat <- confusionMatrix(predictions_red, red_wine_test_list[[i]]$quality)
#conf_mat_list_red[[i]] <- confusion_mat

accuracy_vector_red[i] <- confusion_mat$overall['Accuracy']

var_importance <- varImp(rf_model_red, type = 2)
variable_importance_list_red[[i]] <- var_importance

list(
confusion_matrix = confusion_mat,
accuracy = confusion_mat$overall['Accuracy'],
variable_importance = var_importance
)
}
stopCluster(cl)

for (i in 1:length(results)) {
  conf_mat_list_red[[i]] <- results[[i]]$confusion_matrix
  accuracy_vector_red[i] <- results[[i]]$accuracy
  variable_importance_list_red[[i]] <- results[[i]]$variable_importance
}


cat("Creating 95% Confidence Interval for Accuracy of Model
    predicting Grade A red wine")
```

```
## Creating 95% Confidence Interval for Accuracy of Model
##     predicting Grade A red wine
```

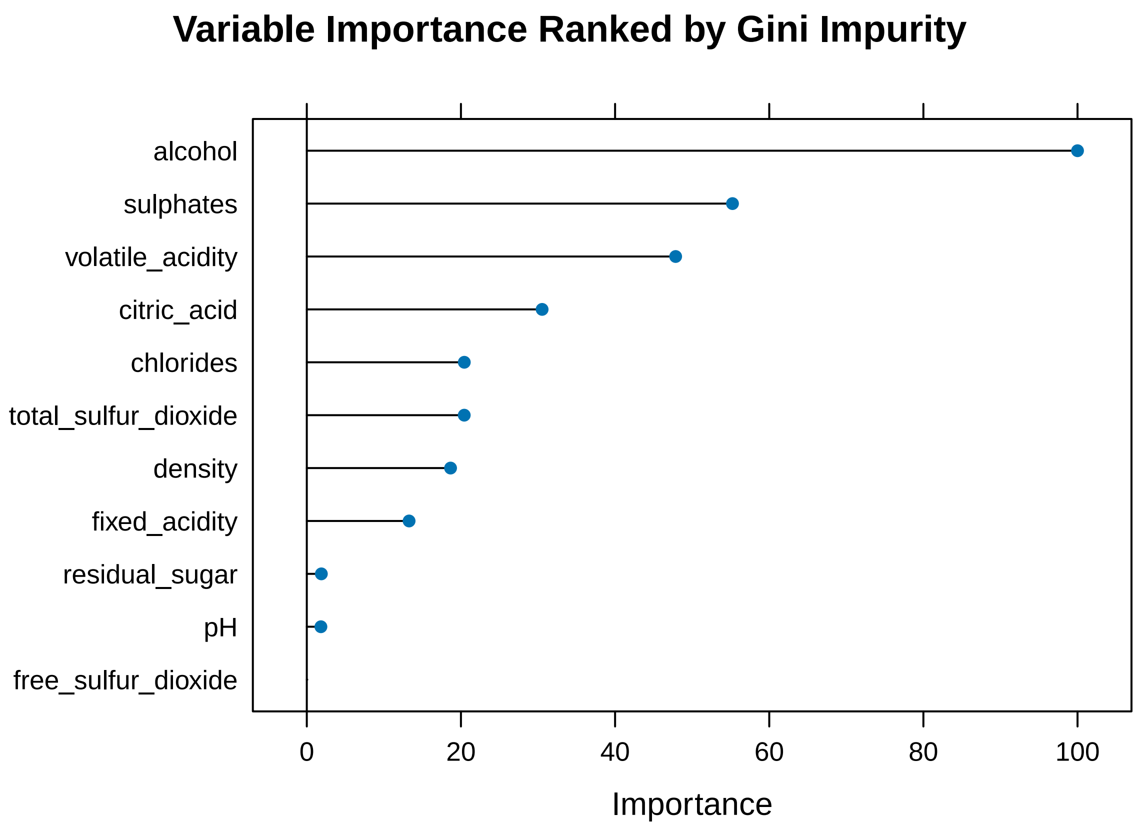
```r
mean_red2_vec  <- mean(accuracy_vector_red)

#standard error
std_error_red <- sd(accuracy_vector_red) / sqrt(length(accuracy_vector_red))

#critical t value for 95% CI
critical_value_red <- qt(0.975, df = length(accuracy_vector_red) - 1)

#confidence interval
lower_ci_red <- mean_red2_vec - (critical_value_red * std_error_red)
upper_ci_red <- mean_red2_vec + (critical_value_red * std_error_red)

# 95% CI
cat("95% Confidence Interval Predicting Grade A Red Wine: [", lower_ci_red, ", ", upper_ci_red, "]\n")
```

```
## 95% Confidence Interval Predicting Grade A Red Wine: [ 0.8906711 ,  0.9034908 ]
```

```r
#Finding Index of accuracy value closest to mean
closest_index_red <- which.min(abs(accuracy_vector_red - mean_red2_vec))


#Confusion Matrix of Model closest to mean accuracy
print(conf_mat_list_red[closest_index_red])
```

```
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 265  22
##          1  12  32
##
##                Accuracy : 0.8973
##                  95% CI : (0.8594, 0.9278)
##     No Information Rate : 0.8369
##     P-Value [Acc > NIR] : 0.001137
##
##                   Kappa : 0.5935
##
##  Mcnemar's Test P-Value : 0.122713
##
##             Sensitivity : 0.9567
##             Specificity : 0.5926
##          Pos Pred Value : 0.9233
##          Neg Pred Value : 0.7273
##              Prevalence : 0.8369
##          Detection Rate : 0.8006
##    Detection Prevalence : 0.8671
##       Balanced Accuracy : 0.7746
##
##        'Positive' Class : 0
##
```

```r
#Variable Importance Plot of model
plot(rf_gridsearch_red_importance,
     main = "Variable Importance Ranked by Gini Impurity")
```



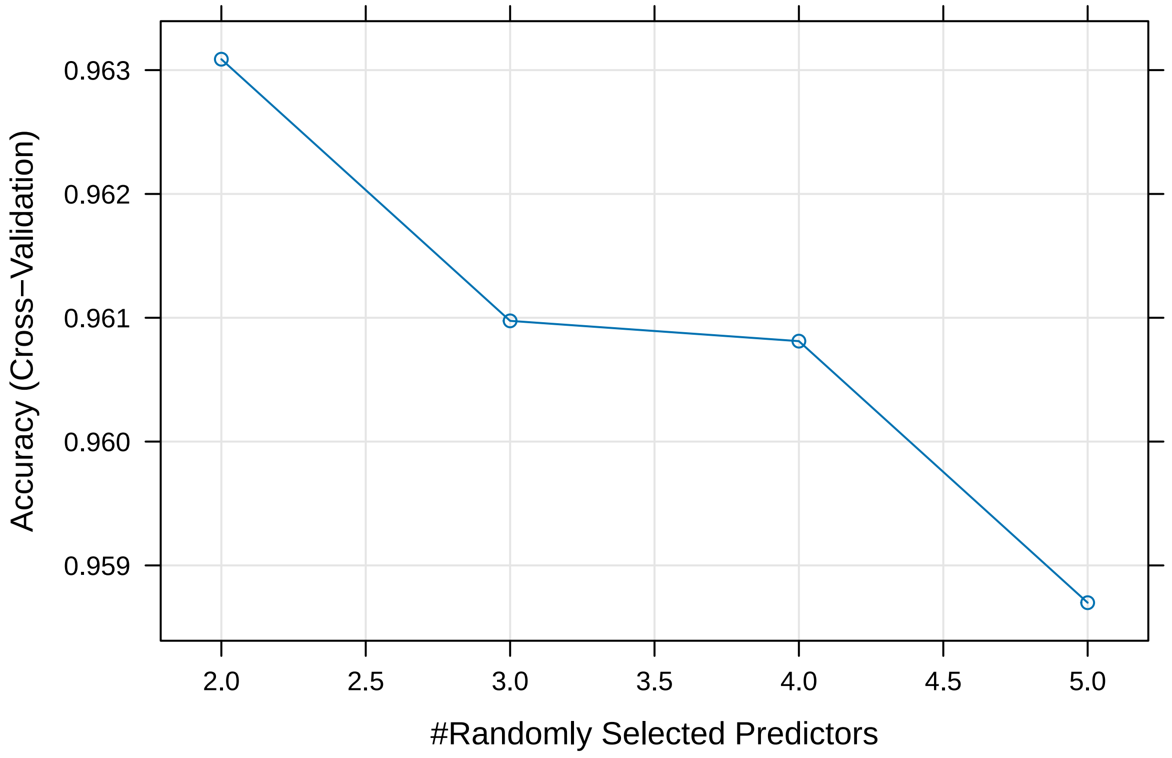
```r
unregister_dopar <- function() {
  env <- foreach:::.foreachGlobals
  rm(list=ls(name=env), pos=env)
}
unregister_dopar()

#Random Forest Model for predicting Grade A White Wine

cat("Calling extra sample storing in data frame and using cross validation and
    grid search to find optimal parameters")
```

```
## Calling extra sample storing in data frame and using cross validation and
##     grid search to find optimal parameters
```

```r
white_wine_rf_extra <- oversampled_white_wine_train_list[[31]]

# Define the control for grid search with 10-fold cross-validation
train_control <- trainControl(method = "cv", number = 10)

# Define the grid of hyper-parameters to tune
tune_grid <- expand.grid(mtry = c(2,3,4,5))

cat("Training the Random Forest model using grid search and
    10-fold cross-validation for White Wine")
```

```
## Training the Random Forest model using grid search and
##     10-fold cross-validation for White Wine
```

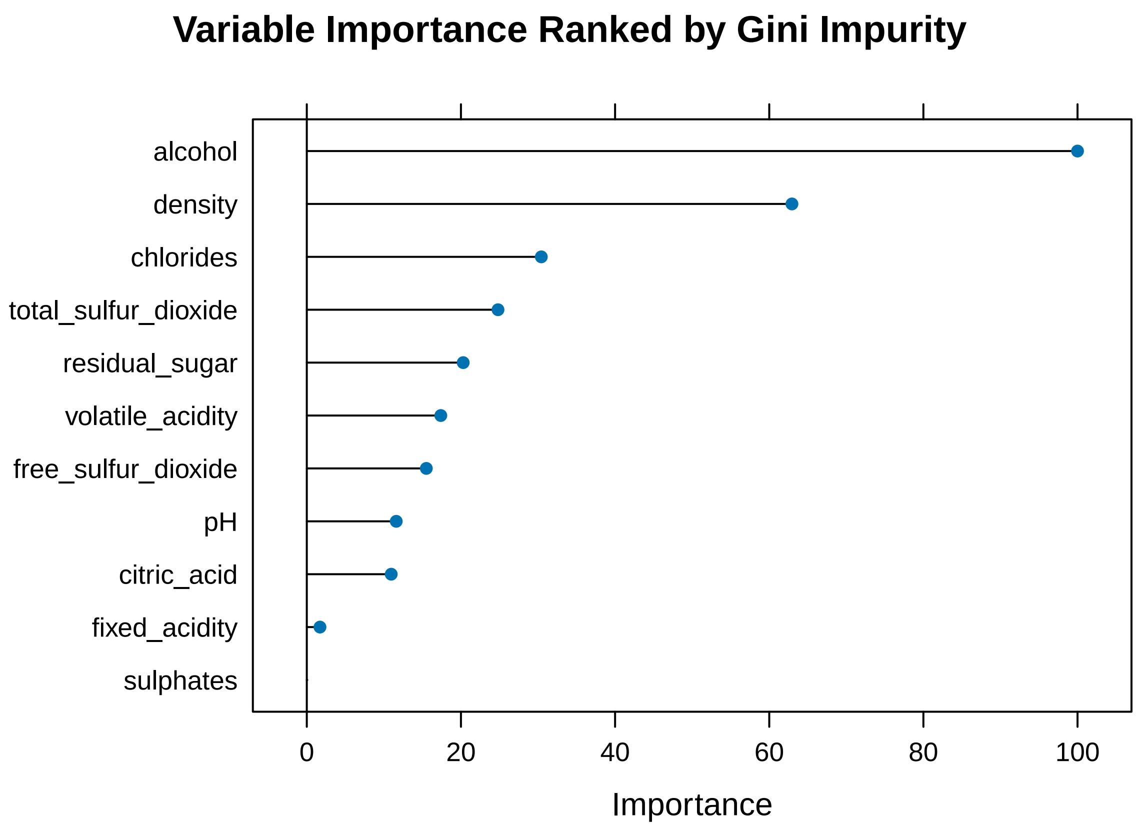
```r
rf_gridsearch_white <- caret::train(quality ~ .,
                                    white_wine_rf_extra,
                                    method = "rf",
                                    trControl = train_control,
                                    tuneGrid = tune_grid,
                                    importance = TRUE)
plot(rf_gridsearch_white)
```



```r
cat("This plot shows that optimal number of variables to
    try at every node split is 2")
```

```
## This plot shows that optimal number of variables to
##     try at every node split is 2
```

```r
#Variable Importance Plot of model predicting Grade A white wine
rf_gridsearch_white_importance <- varImp(rf_gridsearch_white, type = 2)
plot(rf_gridsearch_white_importance,
     main = "Variable Importance Ranked by Gini Impurity")
```



```
cat("This plot shows that the elbow of the importance plot is at the fifth most
    important variable so the remaining variables are dropped from future model,
    these variables are: volatile_acidity, free_sulfur_dioxide, pH, citric_acid,
    fixed_acidity, sulphates")
```

```
## This plot shows that the elbow of the importance plot is at the fifth most
##     important variable so the remaining variables are dropped from future model,
##     these variables are: volatile_acidity, free_sulfur_dioxide, pH, citric_acid,
##     fixed_acidity, sulphates
```

```
drop_columns <- c("volatile_acidity","free_sulfur_dioxide", "pH", "citric_acid",
                  "fixed_acidity", "sulphates")

oversampled_white_wine_train_list <- lapply(oversampled_white_wine_train_list,
                                            function(df) {
  df %>% dplyr::select(-all_of(drop_columns))
})


cat("Examining the grid-search's plot it shows the optimal number of variables
    to randomly sample from at every node split is 2, now applying Random Forest
    Model with optimal parameter 30 times, since this is very time consuming
    using parallel processing")
```

```
## Examining the grid-search's plot it shows the optimal number of variables
##     to randomly sample from at every node split is 2, now applying Random Forest
##     Model with optimal parameter 30 times, since this is very time consuming
##     using parallel processing
```

```r
#Creating empty lists
accuracy_vector_white <- numeric(length(1:30))
conf_mat_list_white <- vector("list",length(1:30))
variable_importance_list_white <- vector("list",length(1:30))

tune_grid2 <- expand.grid(mtry = 2)

#initializing parallel processing
num_cores <- detectCores() - 2
cl <- makePSOCKcluster(num_cores)
registerDoParallel(cl)


results <- foreach (i = 1:length(oversampled_white_wine_train_list),
                    .packages = c("caret", "dplyr")) %dopar% {
# Training the Random Forest model with 30 times
  rf_model_white <- caret::train(
    quality ~ .,
    data = oversampled_white_wine_train_list[[i]],
    method = "rf",
    tuneGrid = tune_grid2,
    importance = TRUE
  )

#Confusion Matrix of final model pwhiteicting Grade A white wine
predictions_white <- predict(rf_model_white, newdata = white_wine_test_list[[i]])
confusion_mat <- confusionMatrix(predictions_white,
                                white_wine_test_list[[i]]$quality)
#conf_mat_list_white[[i]] <- confusion_mat

accuracy_vector_white[i] <- confusion_mat$overall['Accuracy']

var_importance <- varImp(rf_model_white, type = 2)
variable_importance_list_white[[i]] <- var_importance

list(
confusion_matrix = confusion_mat,
accuracy = confusion_mat$overall['Accuracy'],
variable_importance = var_importance
)
}
stopCluster(cl)

for (i in 1:length(results)) {
  conf_mat_list_white[[i]] <- results[[i]]$confusion_matrix
  accuracy_vector_white[i] <- results[[i]]$accuracy
  variable_importance_list_white[[i]] <- results[[i]]$variable_importance
}


cat("Creating 95% Confidence Interval for Accuracy of Model predicting
    Grade A white wine")
```

```
## Creating 95% Confidence Interval for Accuracy of Model predicting
##     Grade A white wine
```

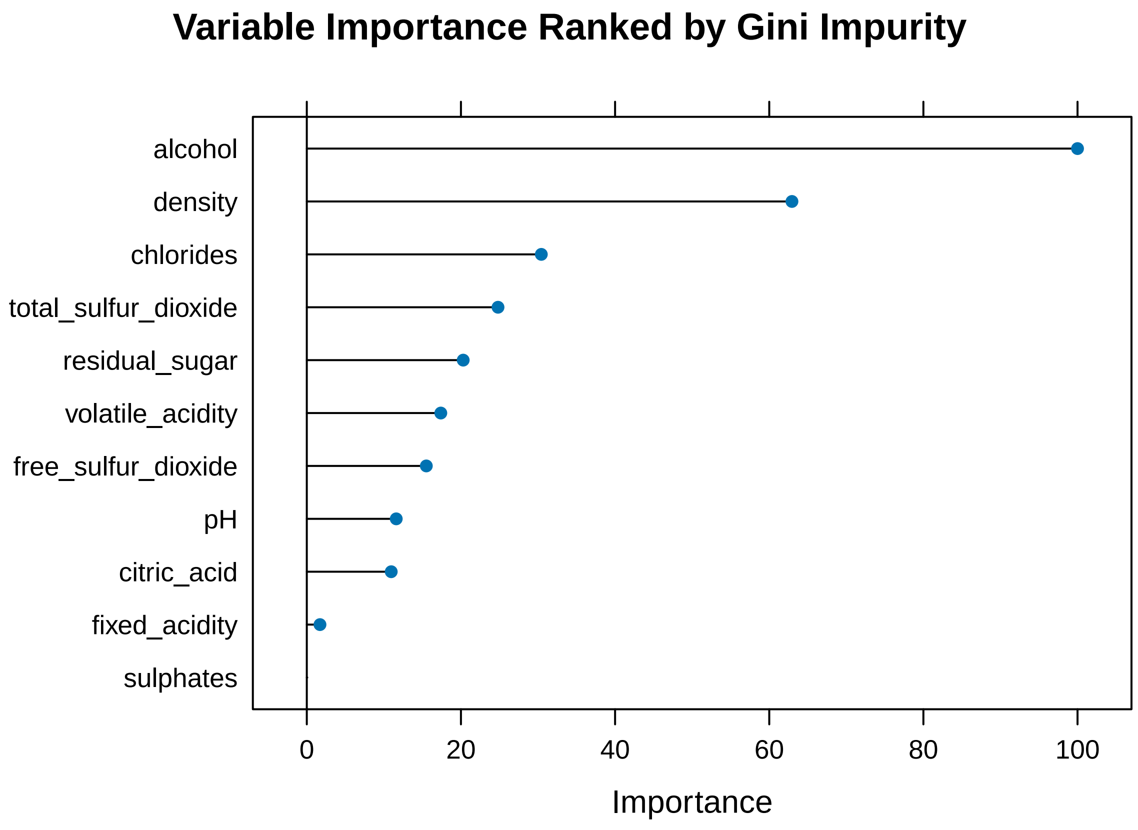
```
mean_white2_vec  <- mean(accuracy_vector_white)

#standard error
std_error_white <- sd(accuracy_vector_white) / sqrt(length(accuracy_vector_white))

#critical t value for 95% CI
critical_value_white <- qt(0.975, df = length(accuracy_vector_white) - 1)

#confidence interval
lower_ci_white <- mean_white2_vec - (critical_value_white * std_error_white)
upper_ci_white <- mean_white2_vec + (critical_value_white * std_error_white)

# 95% CI
cat("95% Confidence Interval Pwhiteicting Grade A white Wine: [", lower_ci_white, ", ", upper_ci_white,
```

```
## 95% Confidence Interval Pwhiteicting Grade A white Wine: [ 0.8584045 ,  0.8654989 ]
```

```
#Finding Index of accuracy value closest to mean
closest_index_white <- which.min(abs(accuracy_vector_white - mean_white2_vec))


#Confusion Matrix of Model closest to mean accuracy
print(conf_mat_list_white[closest_index_white])
```

```
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 704  66
##          1  72 155
##
##                Accuracy : 0.8616
##                  95% CI : (0.8386, 0.8824)
##     No Information Rate : 0.7783
##     P-Value [Acc > NIR] : 1.703e-11
##
##                   Kappa : 0.6027
##
##  Mcnemar's Test P-Value : 0.6704
##
##             Sensitivity : 0.9072
##             Specificity : 0.7014
##          Pos Pred Value : 0.9143
##          Neg Pred Value : 0.6828
##              Prevalence : 0.7783
##          Detection Rate : 0.7061
##    Detection Prevalence : 0.7723
##       Balanced Accuracy : 0.8043
##
##        'Positive' Class : 0
##
```

```r
#Variable Importance Plot of model
plot(rf_gridsearch_white_importance, main = "Variable Importance Ranked by Gini Impurity")
```


Variable Importance Ranked by Gini Impurity
alcohol
density
chlorides
total_sulfur_dioxide
residual_sugar
volatile_acidity
free_sulfur_dioxide
pH
citric_acid
fixed_acidity
sulphates
Importance