Wine Quality Analysis

2025-11-17

Libraries

```
library(readr)
library(naniar)
```

```
## Warning: package 'naniar' was built under R version 4.4.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```
library(tidymodels)
```

```
## Warning: package 'tidymodels' was built under R version 4.4.3

## -- Attaching packages -------------------------------------- tidymodels 1.4.1 --

## v broom        1.0.11     v rsample      1.3.1
## v dials        1.4.2      v tailor       0.1.0
## v dplyr        1.1.4      v tidyr        1.3.1
## v infer        1.0.9      v tune         2.0.1
## v modeldata    1.5.1      v workflows    1.3.0
## v parsnip      1.4.0      v workflowsets 1.1.1
## v purrr        1.2.0      v yardstick    1.3.2
## v recipes      1.3.1

## Warning: package 'broom' was built under R version 4.4.3

## Warning: package 'dials' was built under R version 4.4.3

## Warning: package 'scales' was built under R version 4.4.3

## Warning: package 'infer' was built under R version 4.4.3

## Warning: package 'modeldata' was built under R version 4.4.3

## Warning: package 'parsnip' was built under R version 4.4.3

## Warning: package 'purrr' was built under R version 4.4.3

## Warning: package 'recipes' was built under R version 4.4.3

## Warning: package 'rsample' was built under R version 4.4.3

## Warning: package 'tailor' was built under R version 4.4.3

## Warning: package 'tune' was built under R version 4.4.3

## Warning: package 'workflows' was built under R version 4.4.3

## Warning: package 'workflowsets' was built under R version 4.4.3

## Warning: package 'yardstick' was built under R version 4.4.3
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard()  masks scales::discard()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```r
library(tidyr)
library(DescTools)
```

```
## Warning: package 'DescTools' was built under R version 4.4.3
```

```r
library(tibble)
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

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

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

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

```r
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.4.3
```

```
##
## Attaching package: 'MASS'
```

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

```r
library(effects)
```

```
## Warning: package 'effects' was built under R version 4.4.3
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```r
library(dplyr)
```

Data Import

```r
wine <- read_csv("winequalityN.csv")
```

```
## Rows: 6497 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): type
## dbl (13): ID, fixed_acidity, volatile_acidity, citric_acid, residual_sugar, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Data Set Overview

Structure of the Data Set

Exploring the overall structure of the data set:

```
summary(wine)
```

```
##        ID           type           fixed_acidity    volatile_acidity
##  Min.   :   1   Length:6497        Min.   : 3.800   Min.   :0.0800  
##  1st Qu.:1625   Class :character   1st Qu.: 6.400   1st Qu.:0.2300  
##  Median :3249   Mode  :character   Median : 7.000   Median :0.2900  
##  Mean   :3249                      Mean   : 7.217   Mean   :0.3397  
##  3rd Qu.:4873                      3rd Qu.: 7.700   3rd Qu.:0.4000  
##  Max.   :6497                      Max.   :15.900   Max.   :1.5800  
##                                    NA's   :10       NA's   :8       
##   citric_acid     residual_sugar     chlorides       free_sulfur_dioxide
##  Min.   :0.0000   Min.   : 0.600   Min.   :0.00900   Min.   :  1.00     
##  1st Qu.:0.2500   1st Qu.: 1.800   1st Qu.:0.03800   1st Qu.: 17.00     
##  Median :0.3100   Median : 3.000   Median :0.04700   Median : 29.00     
##  Mean   :0.3187   Mean   : 5.444   Mean   :0.05604   Mean   : 30.53     
##  3rd Qu.:0.3900   3rd Qu.: 8.100   3rd Qu.:0.06500   3rd Qu.: 41.00     
##  Max.   :1.6600   Max.   :65.800   Max.   :0.61100   Max.   :289.00     
##  NA's   :3        NA's   :2        NA's   :2                            
##  total_sulfur_dioxide    density             pH          sulphates     
##  Min.   :  6.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200  
##  1st Qu.: 77.0        1st Qu.:0.9923   1st Qu.:3.110   1st Qu.:0.4300  
##  Median :118.0        Median :0.9949   Median :3.210   Median :0.5100  
##  Mean   :115.7        Mean   :0.9947   Mean   :3.218   Mean   :0.5312  
##  3rd Qu.:156.0        3rd Qu.:0.9970   3rd Qu.:3.320   3rd Qu.:0.6000  
##  Max.   :440.0        Max.   :1.0390   Max.   :4.010   Max.   :2.0000  
##                                        NA's   :9       NA's   :4       
##     alcohol         quality     
##  Min.   : 8.00   Min.   :3.000  
##  1st Qu.: 9.50   1st Qu.:5.000  
##  Median :10.30   Median :6.000  
##  Mean   :10.49   Mean   :5.818  
##  3rd Qu.:11.30   3rd Qu.:6.000  
##  Max.   :14.90   Max.   :9.000  
##
```

```
head(wine)
```

```
## # A tibble: 6 x 14
##      ID type  fixed_acidity volatile_acidity citric_acid residual_sugar
##   <dbl> <chr>         <dbl>            <dbl>       <dbl>          <dbl>
## 1     1 white           7             0.27        0.36           20.7
## 2     2 white           6.3           0.3         0.34            1.6
## 3     3 white           8.1           0.28        0.4             6.9
## 4     4 white           7.2           0.23        0.32            8.5
## 5     5 white           7.2             0.23        0.32            8.5
## 6     6 white           8.1             0.28        0.4             6.9
## # i 8 more variables: chlorides <dbl>, free_sulfur_dioxide <dbl>,
## #   total_sulfur_dioxide <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
## #   alcohol <dbl>, quality <dbl>
```

Checking if there any duplicates in the data set:

```
wine |>
  group_by(ID) |>
  summarise(n = n()) |>
  filter(n > 1L)
```

```
## # A tibble: 0 x 2
## # i 2 variables: ID <dbl>, n <int>
```

Handling Missing Values

Calculating how many missing values this data set contains:

```
sum(is.na(wine))
```

```
## [1] 38
```

Investigating which rows contain missing values:

```
wine |>
  filter(if_any(everything(),is.na))
```

```
## # A tibble: 34 x 14
##       ID type  fixed_acidity volatile_acidity citric_acid residual_sugar
##    <dbl> <chr>         <dbl>            <dbl>       <dbl>          <dbl>
##  1    18 white          NA              0.66         0.48            1.2
##  2    34 white           6.2            0.12         0.34           NA
##  3    55 white           6.8            0.2          0.59            0.9
##  4    87 white           7.2           NA            0.63           11
##  5    99 white           9.8            0.36         0.46           10.5
##  6   140 white           8.1            0.28         0.39            1.9
##  7   175 white          NA              0.27         0.31           17.7
##  8   225 white           6.3            0.495        0.22            1.8
##  9   250 white          NA              0.41         0.14           10.4
## 10   268 white          NA              0.58         0.07            6.9
## # i 24 more rows
## # i 8 more variables: chlorides <dbl>, free_sulfur_dioxide <dbl>,
## #   total_sulfur_dioxide <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
## #   alcohol <dbl>, quality <dbl>
```

Investigating which fields contain missing values:

```
colSums(is.na(wine))
```

```
##                   ID                 type        fixed_acidity
##                    0                    0                   10
##     volatile_acidity          citric_acid       residual_sugar
##                    8                    3                    2
##            chlorides  free_sulfur_dioxide total_sulfur_dioxide
##                    2                    0                    0
##              density                   pH            sulphates
##                    0                    9                    4
##                alcohol              quality
##                      0                    0
```

Little's Missing Completely at Random (MCAR) Test:

```
mcar_test(wine |> dplyr::select(fixed_acidity:quality))
```

```
## # A tibble: 1 x 4
##   statistic    df p.value missing.patterns
##       <dbl> <dbl>   <dbl>            <int>
## 1      117.   107   0.240               11
```

Is removing missing values going to affect my sample size?

```
sum(is.na(wine)) / nrow(wine) * 100
```

```
## [1] 0.5848853
```

```
# Less than 1% of all observations contain missing values.
```

Performing listwise deletion:

```
wine <- na.omit(wine)
```

Wine Quality

Identifying unique quality scores:

```
unique(wine$quality)
```

```
## [1] 6 5 7 8 4 3 9
```

Transforming quality variable to factor and renaming it to `quality_score`:

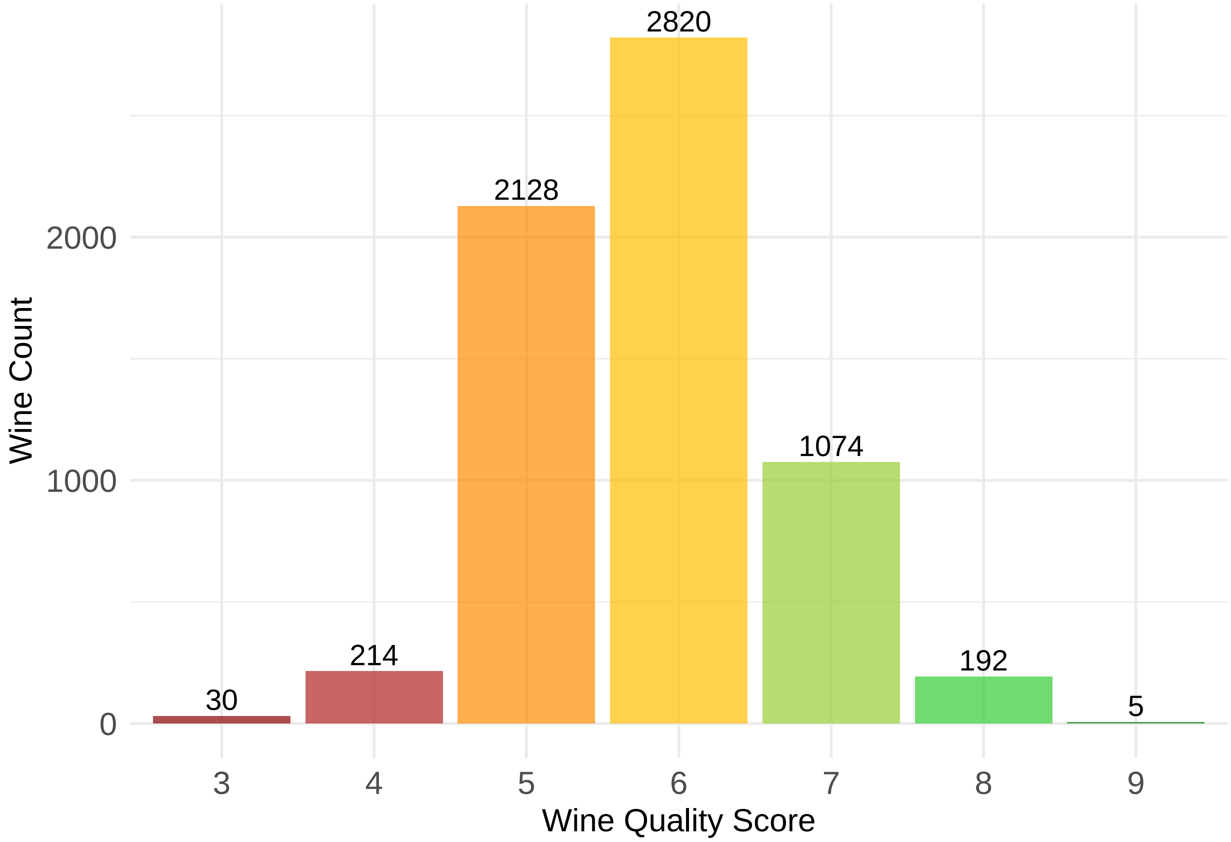
```
# Creating quality levels
quality_levels <- c("3", "4", "5", "6", "7", "8", "9")

wine <- wine |>
  mutate(quality = factor(quality, levels = quality_levels)) |>
  rename(quality_score = quality)

rm(quality_levels)
```

Creating colour scheme for quality scores:

```
clr_score <- c(
  "3" = "#8B0000",
  "4" = "#B22222",
  "5" = "#FF8C00",
  "6" = "#FFBF00",
  "7" = "#9ACD32",
  "8" = "#32CD32",
  "9" = "#008000"
)
```

Exploring distribution of wine quality scores:

```
wine |>
  ggplot(aes(x = quality_score)) +
  geom_bar(
    aes(fill = quality_score),
    alpha = 0.7,
    show.legend = FALSE
    ) +
  geom_text(
    stat = "count",
    aes(label = after_stat(count)),
    vjust = -0.3
    ) +
  theme_minimal() +
  labs(
    x = "Wine Quality Score",
    y = "Wine Count"
    ) +
  scale_fill_manual(
    values = clr_score
  ) +
  theme(
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text    = element_text(size = 12)
  )
```



Creating quality groups:

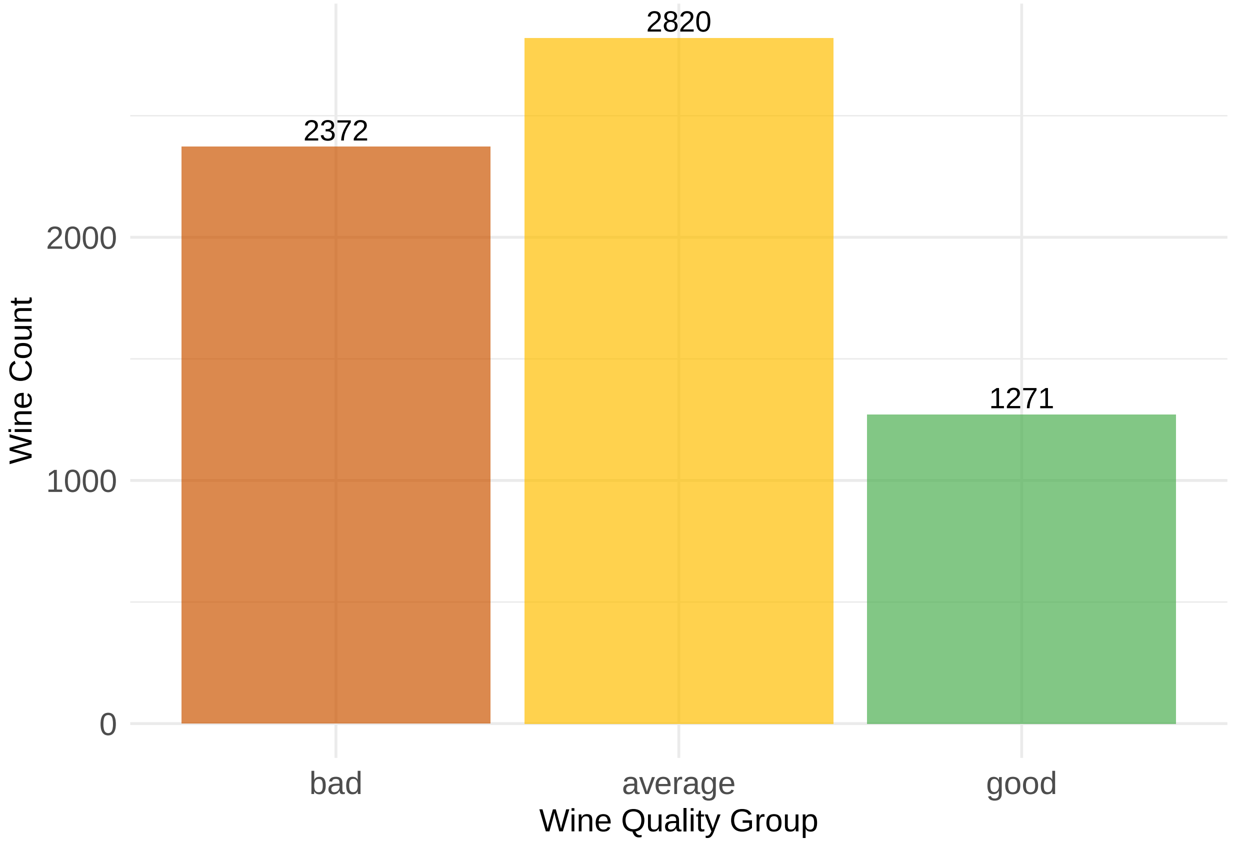
```
wine <- wine |>
  mutate(
    quality_category = case_when(
      quality_score %in% c("3", "4", "5") ~ "bad",
      quality_score %in% c("7", "8", "9") ~ "good",
      quality_score == "6" ~ "average",
    ),
    quality_category = factor(
      quality_category,
      levels = c("bad", "average", "good")
    )
  )
```

Creating colour scheme for quality groups:

```
clr_category <- c(
  "bad"     =  "#CC5500",
  "average" =  "#FFBF00",
  "good"    =  "#4CAF50"
)
```

Visualising new quality categories:

```
wine |>
  ggplot(aes(x = quality_category)) +
  geom_bar(
    aes(fill = quality_category),
    alpha = 0.7,
    show.legend = FALSE
    ) +
  geom_text(
    stat = "count",
    aes(label = after_stat(count)),
    vjust = -0.3
    ) +
  theme_minimal() +
  labs(
    x = "Wine Quality Group",
    y = "Wine Count"
    ) +
  scale_fill_manual(
    values = clr_category
  ) +
  theme(
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text    = element_text(size = 12)
  )
```



Wine Type

What are the unique types of wines within the data set?

```r
unique(wine$type)
```

```
## [1] "white" "red"
```

Transforming `type` variable into the factor with two levels: “white” and “red”:

```r
wine <- wine |>
  mutate(type = factor(type))
```

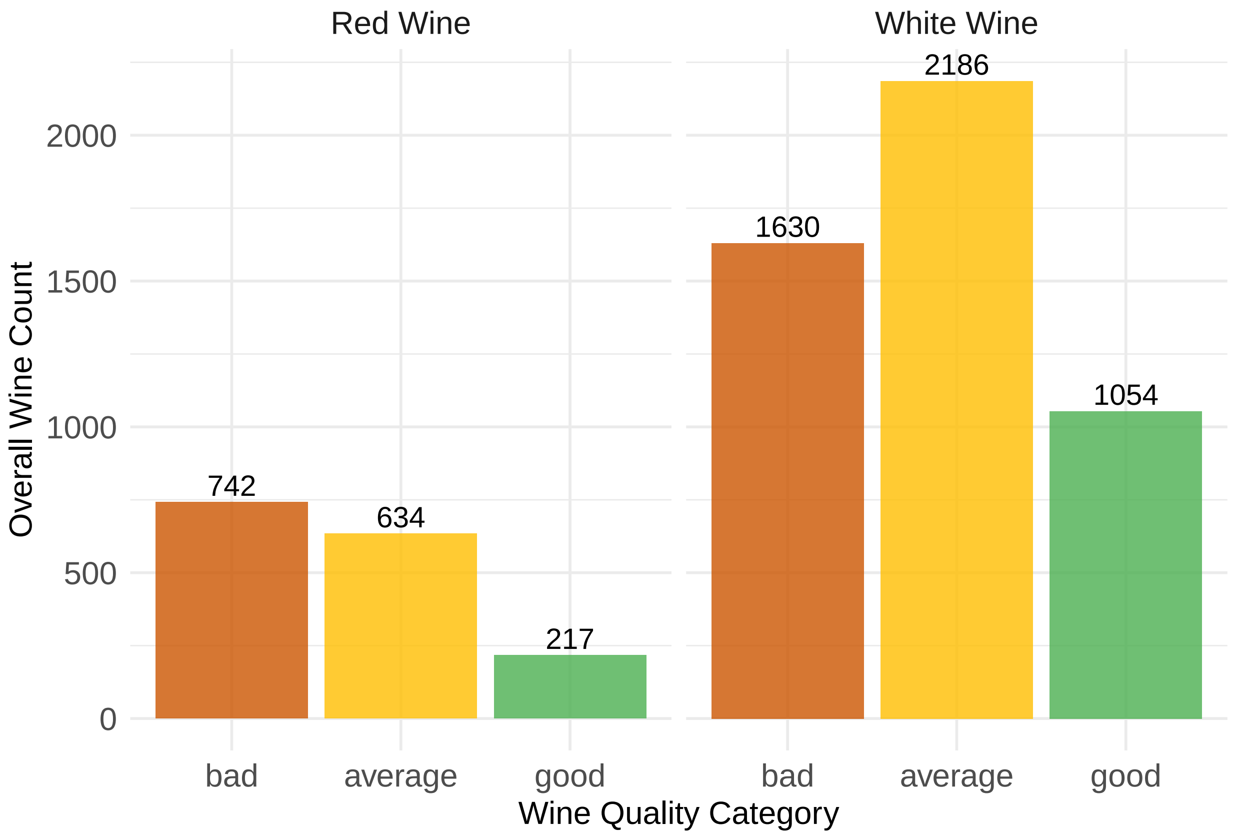
Counting the number of white and red wines within the data set:

```r
wine |>
  group_by(type) |>
  summarise(
    wine_count = n(),
    wine_prop = wine_count/6497
    )
```

```
## # A tibble: 2 x 3
##   type  wine_count wine_prop
##   <fct>      <int>     <dbl>
## 1 red         1593     0.245
## 2 white       4870     0.750
```

Exploring quality group sizes for red and white wines:

```r
wine |>
  mutate(type = if_else(type == "red", "Red Wine", "White Wine")) |>
  ggplot(aes(x = quality_category)) +
  geom_bar(
    aes(fill = quality_category),
    alpha = 0.8,
    show.legend = FALSE
    ) +
  facet_wrap(~type) +
  geom_text(
    stat = "count",
    aes(label = after_stat(count)),
    vjust = -0.3
    ) +
  theme_minimal() +
  scale_fill_manual(values = clr_category) +
  labs(
    x = "Wine Quality Category",
    y = "Overall Wine Count"
    ) +
  theme(
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text    = element_text(size = 12),
    strip.text   = element_text(size = 12)
  )
```



Data Set Split

Splitting data into training and test for red and white wines:

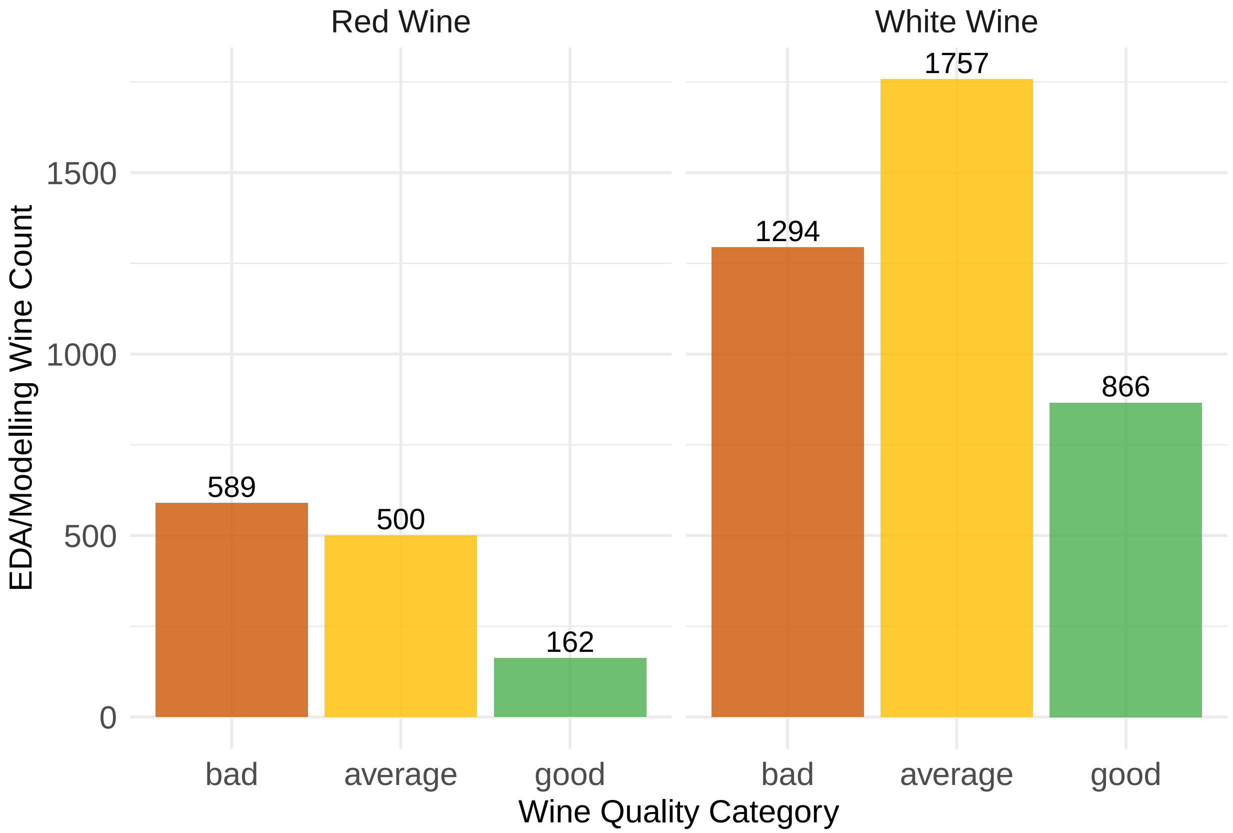
```r
wine$split <- interaction(wine$type, wine$quality_score)

set.seed(123)
split <- initial_split(wine, prop = 0.8, strata = split)

train <- training(split)
test  <- testing(split)

rm(split)
```

Creating separate data sets for each group:

```r
train <- train |>
  dplyr::select(-quality_score, -split)

test <- test |>
  dplyr::select(-quality_score, -split)
```

Visualising final sizes of groups used for EDA/Modelling:

```r
train |>
  mutate(type = if_else(type == "red", "Red Wine", "White Wine")) |>
  ggplot(aes(x = quality_category)) +
  geom_bar(
    aes(fill = quality_category),
    alpha = 0.8,
    show.legend = FALSE
    ) +
  facet_wrap(~type) +
  geom_text(
    stat = "count",
    aes(label = after_stat(count)),
    vjust = -0.3
    ) +
  theme_minimal() +
  scale_fill_manual(values = clr_category) +
  labs(
    x = "Wine Quality Category",
    y = "EDA/Modelling Wine Count"
    ) +
  theme(
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text    = element_text(size = 12),
    strip.text   = element_text(size = 12)
  )
```



Wine Physicochemical Properties

Univariate Analysis

Data set transformation into tidy data, moving all columns that describe the wine's physicochemical properties into one column, with the respective values in another column:

```
tidy_train <- train |>
  pivot_longer(
    cols = fixed_acidity:alcohol,
    names_to = "property_name",
    values_to = "property_value"
  )
```

Renaming each property name:

```
# Creating new property labels:

property_labels <- c(
  alcohol = "Alcohol Percentage by Volume",
  chlorides = "Sodium Chloride (g/L)",
  citric_acid = "Citric Acid (g/L)",
  density = "Density (g/cm3)",
  fixed_acidity = "Fixed Acidity (g/L)",
  free_sulfur_dioxide = "Free Sulphur Dioxide (mg/L)",
  total_sulfur_dioxide = "Total Sulphur Dioxide (mg/L)",
  pH = "pH",
  residual_sugar = "Sugar (g/L)",
  sulphates = "Potassium Sulphate (g/L)",
  volatile_acidity = "Volatile Acidity (g/L)"
)

# Replacing old property labels with the new ones:
tidy_train <- tidy_train |>
  mutate(property_name = property_labels[property_name])
```

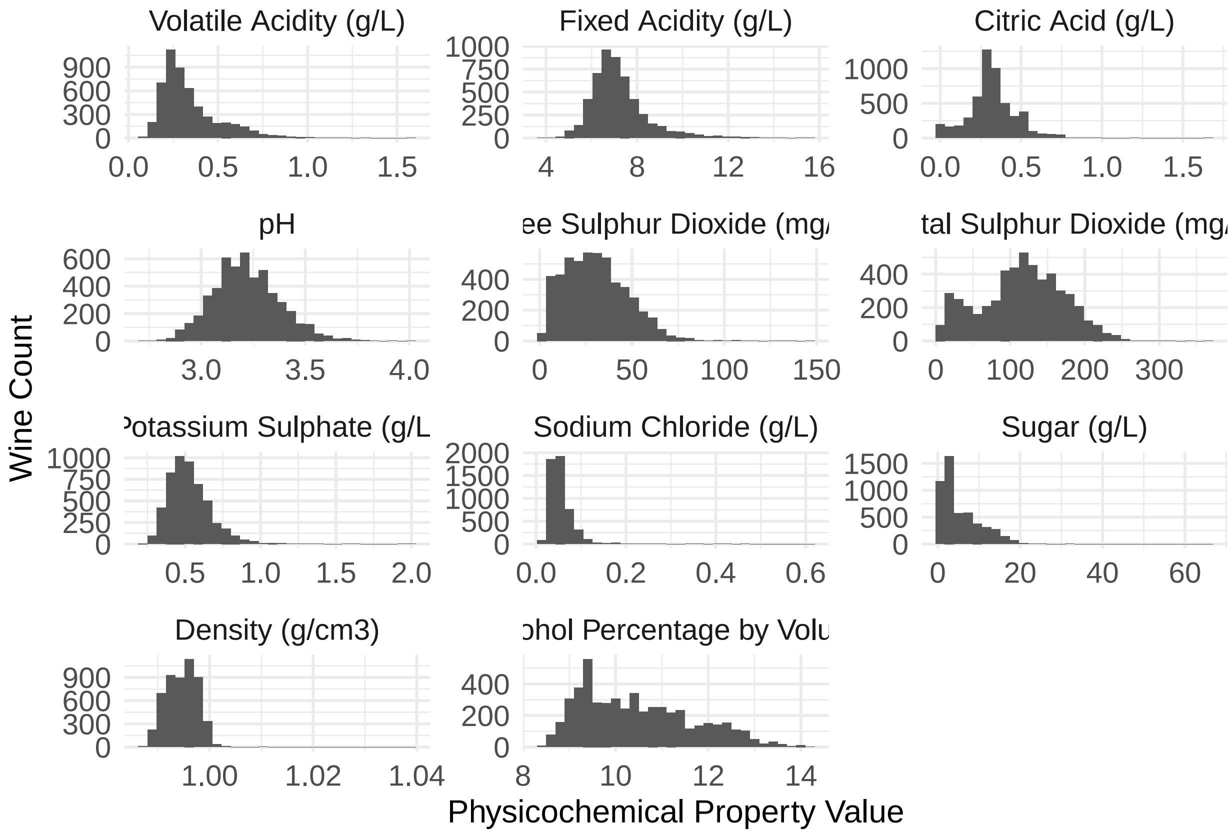
Transforming `property_name` column into a factor:

```
# Creating property levels:
# (I want them in specific order - e.g., variables that relate to
# acidity grouped together).

property_order <- c(
                     "Volatile Acidity (g/L)",
                     "Fixed Acidity (g/L)",
                     "Citric Acid (g/L)",
                     "pH",
                     "Free Sulphur Dioxide (mg/L)",
                     "Total Sulphur Dioxide (mg/L)",
                     "Potassium Sulphate (g/L)",
                     "Sodium Chloride (g/L)",
                     "Sugar (g/L)",
                     "Density (g/cm3)",
                     "Alcohol Percentage by Volume"
                     )

tidy_train <- tidy_train |>
  mutate(
    property_name = factor(property_name, levels = property_order)
  )
```

Visualising data distribution for each property (histograms):

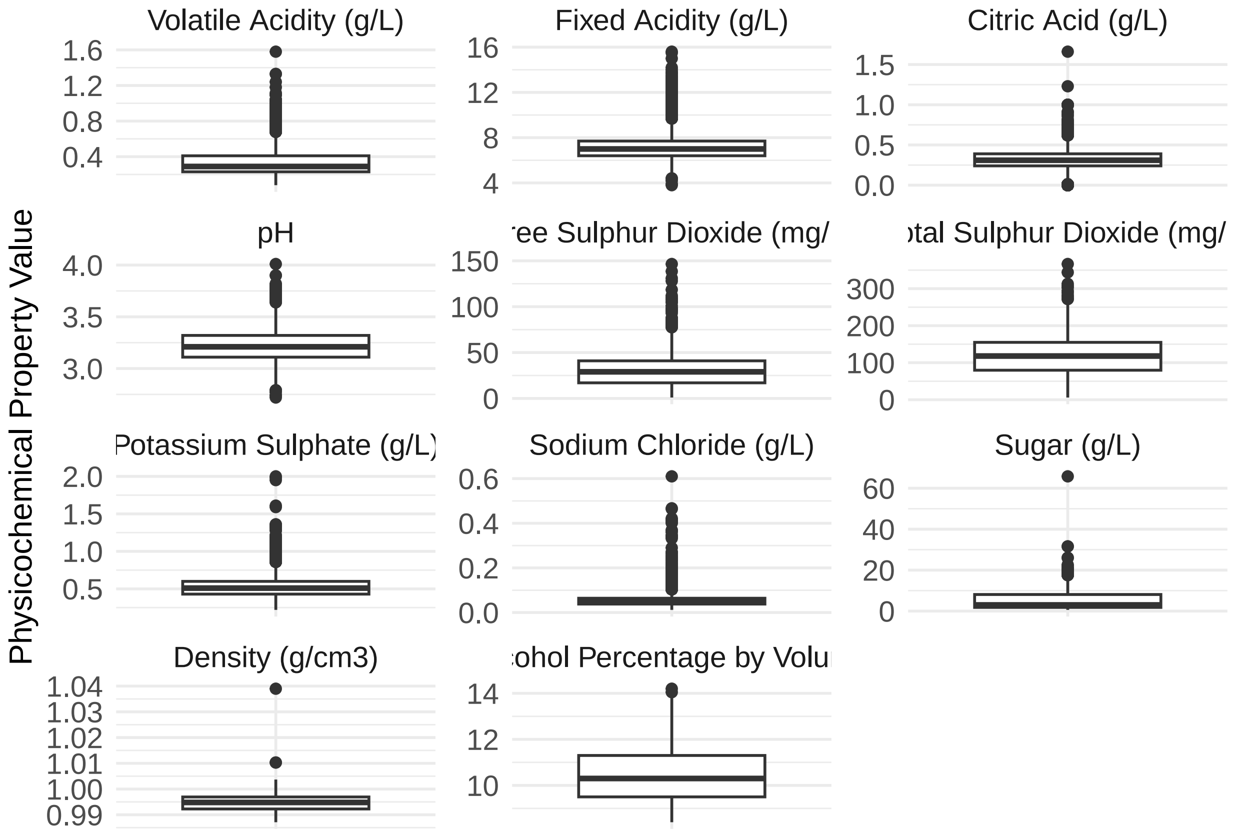
```
tidy_train |>
  ggplot(aes(x = property_value)) +
  geom_histogram() +
  facet_wrap(
    ~property_name,
    scales = "free",
    nrow = 4
  ) +
  labs(
    y = "Wine Count",
    x = "Physicochemical Property Value"
  ) +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text    = element_text(size = 11),
    strip.text   = element_text(size = 11)
  )
```

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



Visualising data distribution for each property (boxplots):

```
tidy_train |>
  ggplot(aes(x = factor(1), y = property_value)) +
  geom_boxplot(width = 0.7) +
  facet_wrap(
    ~property_name,
    scales = "free_y",
    nrow = 4
  ) +
  labs(
    y = "Physicochemical Property Value"
  ) +
  theme_minimal() +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_text(size = 12),
    axis.text.y  = element_text(size = 11),
    axis.text.x  = element_blank(),
    strip.text   = element_text(size = 11)
  )
```



Capping Outliers

I capped outliers for red and white wines separately:

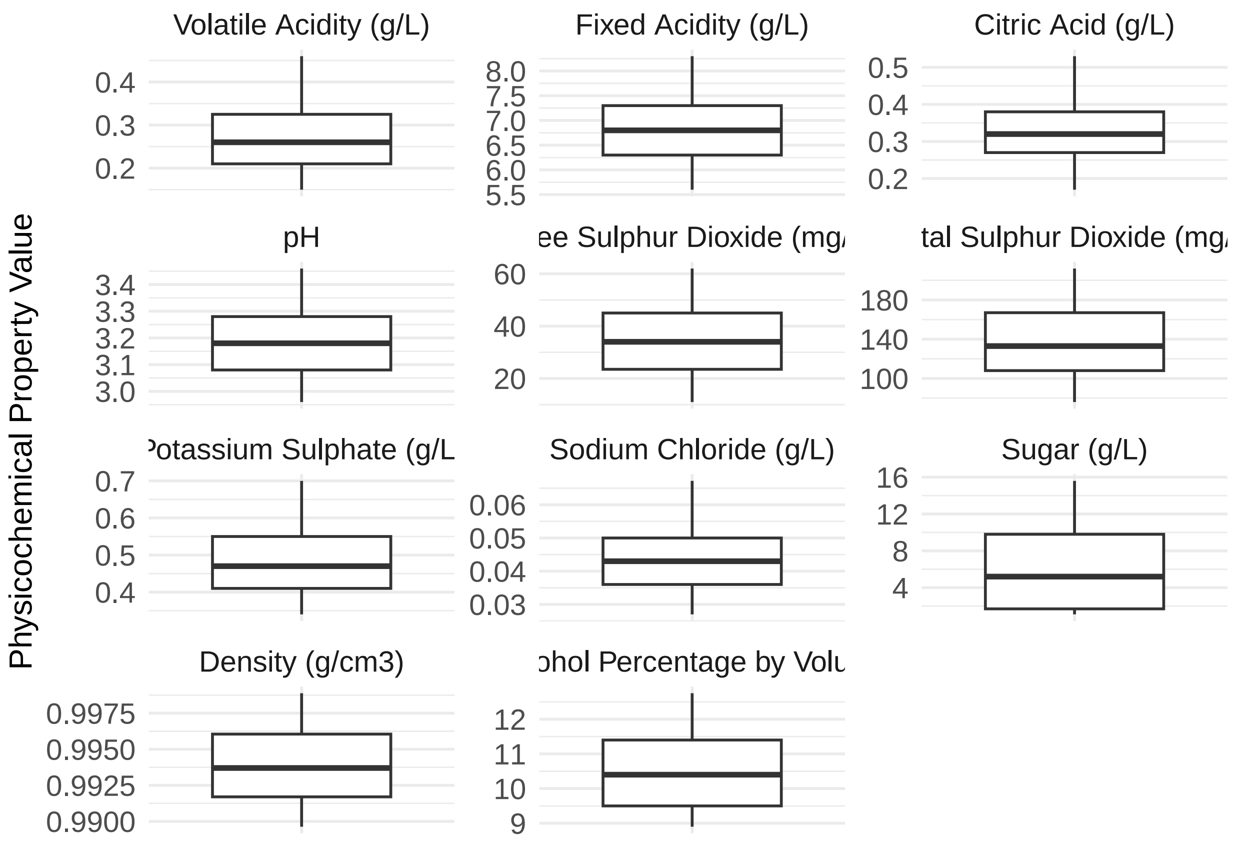
```r
ws_train <- train |>
  group_by(type) |>
  mutate(across(
    fixed_acidity:alcohol,
    ~Winsorize(.,val = quantile(., probs = c(0.05, 0.95)))
    )
  ) |>
  ungroup()
```

Creating a tidy version of the winsorized data set:

```r
ws_tidy_train <- ws_train |>
  pivot_longer(
    cols = fixed_acidity:alcohol,
    names_to = "property_name",
    values_to = "property_value"
  )

ws_tidy_train <- ws_tidy_train |>
  mutate(property_name = property_labels[property_name])

ws_tidy_train <- ws_tidy_train |>
  mutate(
    property_name = factor(property_name, levels = property_order)
  )
```

Investigating how winsorization affected outliers:

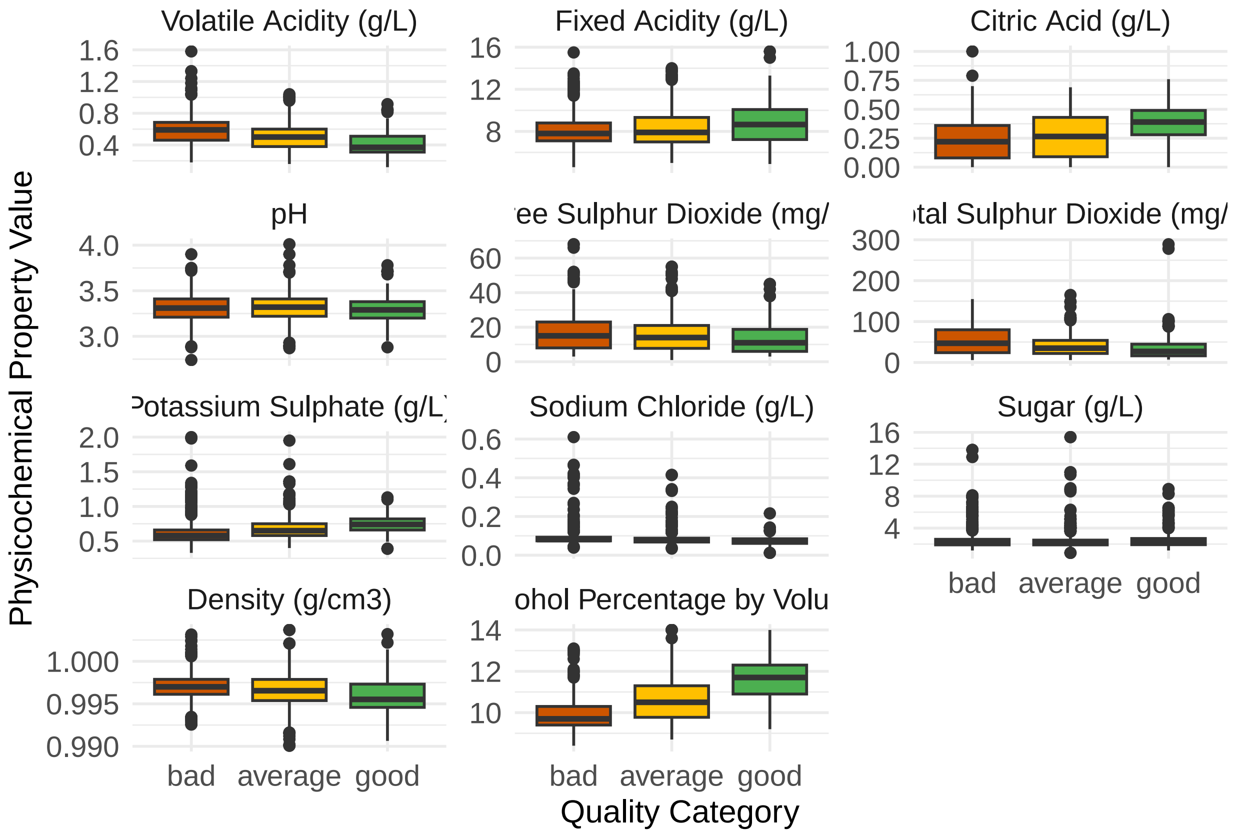
```
ws_tidy_train |>

   # Select which group of wine - "red" or "white" - to investigate:
  filter(type == "white") |>
  ggplot(aes(x = factor(1), y = property_value)) +
  geom_boxplot(width = 0.7) +
  facet_wrap(
    ~property_name,
    scales = "free_y",
    nrow = 4
  ) +
  labs(
    y = "Physicochemical Property Value"
  ) +
  theme_minimal() +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_text(size = 12),
    axis.text.y  = element_text(size = 11),
    axis.text.x  = element_blank(),
    strip.text   = element_text(size = 11)
  )
```



Multivariate Analysis

```
# Select if you want to visualise tidy data set (i.e., "tidy_train")
# or winsorized data set (i.e., "ws_tidy_train")

tidy_train |>

  # Select which group of wine - "red" or "white" - to investigate:
  filter(type == "red") |>

  ggplot(aes(x = quality_category, y = property_value)) +
  geom_boxplot(
    aes(fill = quality_category),
    show.legend = FALSE
  ) +
  facet_wrap(
    ~property_name,
    scales = "free_y",
    nrow = 4
  ) +
  labs(
    y = "Physicochemical Property Value",
    x = "Quality Category"
  ) +
  theme_minimal() +
```

```r
  scale_fill_manual(values = clr_category) +
  theme(
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text    = element_text(size = 11),
    strip.text   = element_text(size = 11)
  )
```



Checking for multicollinearity using Variance Inflation Factor (VIF):

```r
lm_model <- lm(
  fixed_acidity ~ .,

  # Select if you want to visualise regular or winsorized data set
  # And which group of wine - "red" or "white" - to investigate:
  data = dplyr::select(ws_train |> filter(type == "white"), fixed_acidity:alcohol)
  )

vif_values <- vif(lm_model)
vif_values
```

```
##     volatile_acidity          citric_acid       residual_sugar
##             1.118959             1.140817             7.156704
##            chlorides  free_sulfur_dioxide total_sulfur_dioxide
##             1.514900             1.837052             2.336314
##              density                   pH            sulphates
##            16.181059             1.220517             1.100082
##              alcohol
##             5.784256
```

```
rm(list = c("lm_model", "vif_values"))
```

Wine Type Selection

To avoid code repetition select which group of wine – "red" or "white" – to investigate further:

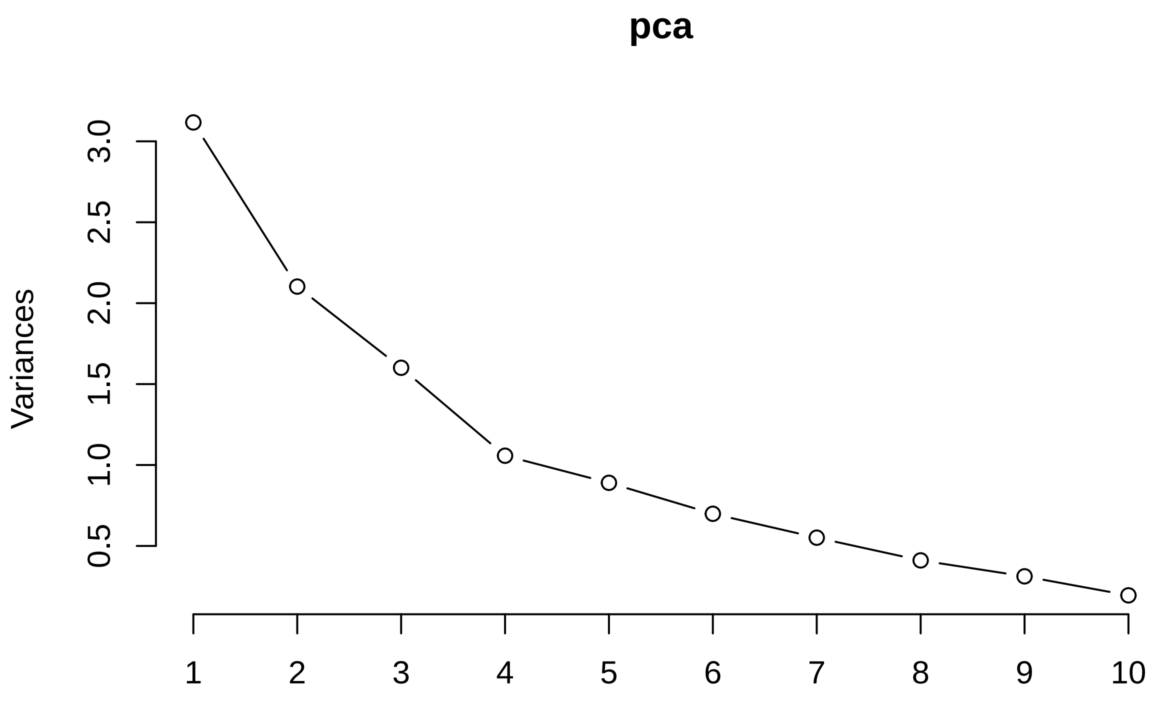
```
# input either "red" or "white"

tp_wine <- "red"
```

The code below first runs principal component analysis and then fits ordinal logistic regression on the specified wine type.

Principal Component Analysis

```
pca <- prcomp(
  dplyr::select(ws_train |>  filter(type == tp_wine), fixed_acidity:alcohol),
  scale. = TRUE
)
```

```
plot(pca, type = 'l')
```



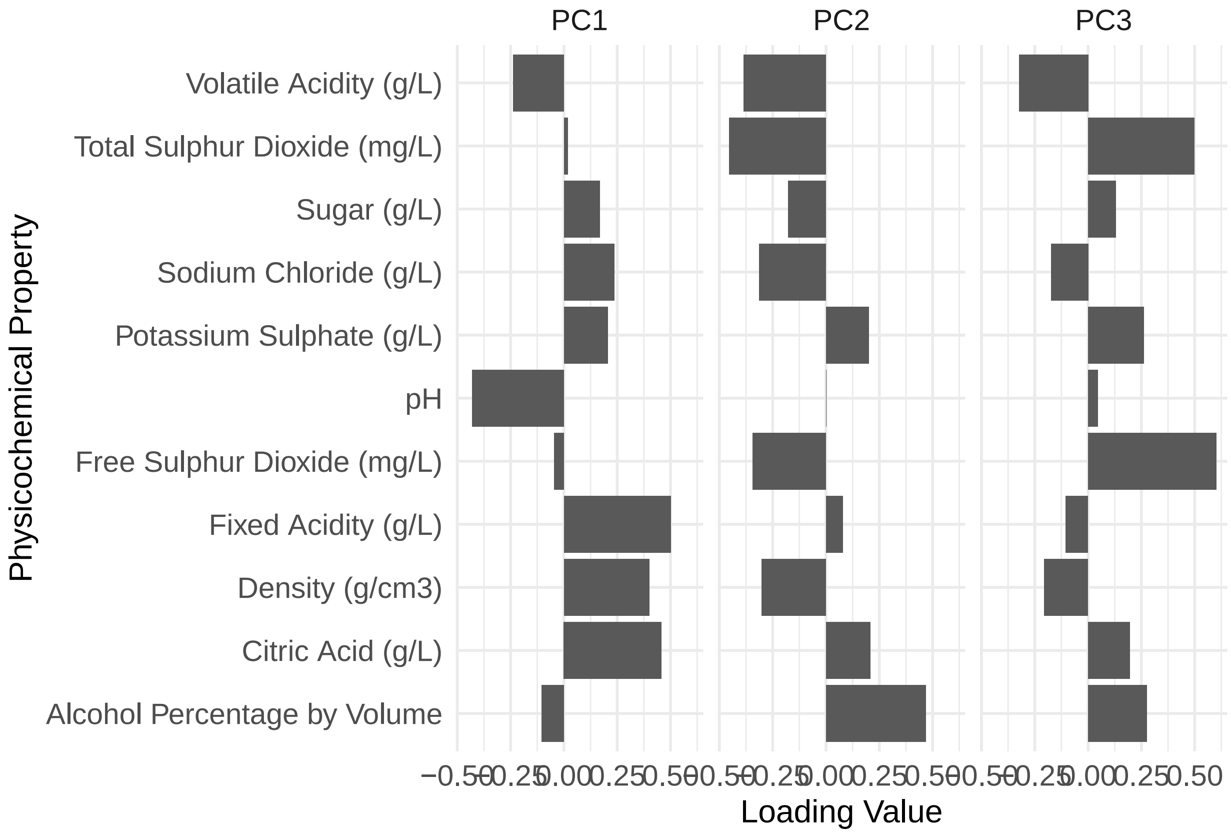
```r
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6    PC7
## Standard deviation     1.7654 1.4500 1.2652 1.02831 0.94342 0.83587 0.7423
## Proportion of Variance 0.2833 0.1911 0.1455 0.09613 0.08091 0.06352 0.0501
## Cumulative Proportion  0.2833 0.4745 0.6200 0.71613 0.79705 0.86056 0.9107
##                            PC8     PC9   PC10    PC11
## Standard deviation     0.64085 0.55875 0.4413 0.25524
## Proportion of Variance 0.03734 0.02838 0.0177 0.00592
## Cumulative Proportion  0.94800 0.97638 0.9941 1.00000
```

```r
loadings <-
  pca$rotation |>
  as.data.frame() |>
  rownames_to_column("property_name")
```

Investigating the first 3 principal components:

```r
loadings |>
  pivot_longer(
    cols = PC1:PC11,
    names_to = "PC_number",
    values_to = "PC_value"
  ) |>
  mutate(property_name = property_labels[property_name]) |>
  filter(PC_number %in% c("PC1", "PC2", "PC3")) |>
  ggplot(aes(y = PC_value, x = property_name)) +
  geom_col() +
  coord_flip() +
  theme_minimal() +
  facet_wrap(~PC_number) +
  labs(
    y = "Loading Value",
    x = "Physicochemical Property"
  ) +
  theme(
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    axis.text    = element_text(size = 11),
    strip.text   = element_text(size = 11)
  )
```



```
scores_train <-
  pca$x |>
  as.data.frame() |>
  cbind(train |> filter(type == tp_wine))
```

Ordinal Logistic Regression

Building Models

```
model= polr(
  quality_category ~ PC1 + PC2 + PC3 ,
  data = scores_train,
  Hess = TRUE
  )
```

```
summary(model)
```

```
## Call:
## polr(formula = quality_category ~ PC1 + PC2 + PC3, data = scores_train,
##     Hess = TRUE)
##
## Coefficients:
##      Value Std. Error t value
## PC1 0.1114    0.03282   3.396
## PC2 0.8620    0.04908  17.566
## PC3 0.4147    0.04834   8.580
```

```
##
## Intercepts:
##              Value   Std. Error t value
## bad|average  -0.2315  0.0661    -3.5039
## average|good  2.5284  0.1068    23.6638
##
## Residual Deviance: 1999.315
## AIC: 2009.315
```

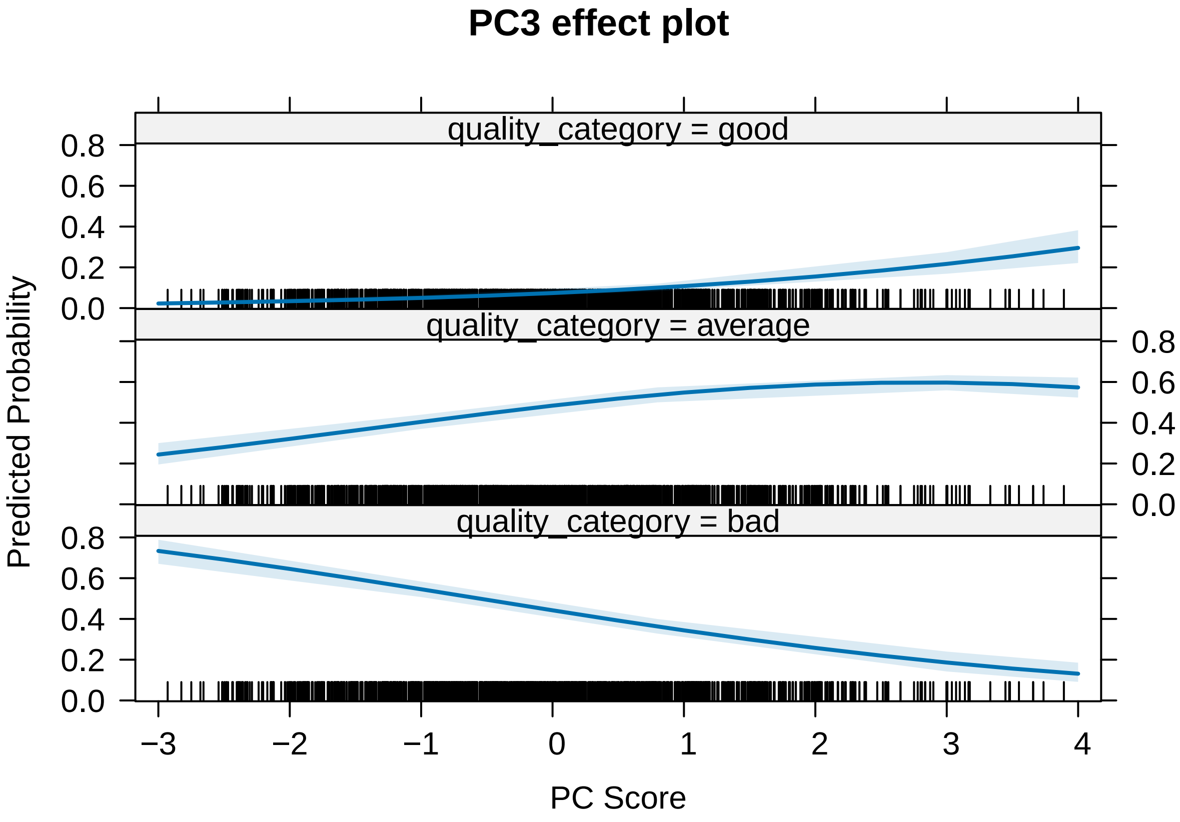
Estimating OR and confidence intervals:

```
exp(cbind(OR = coef(model), confint(model)))
```

```
## Waiting for profiling to be done...
```

```
##           OR    2.5 %   97.5 %
## PC1 1.117877 1.048332 1.192310
## PC2 2.367988 2.154051 2.611206
## PC3 1.513991 1.377936 1.665568
```

```
plot(
  Effect(focal.predictors = "PC3", model),
  xlab = "PC Score",
  ylab = "Predicted Probability"
)
```



Testing Models

Calculating 5th and 95th percentile cutoffs:

```
winsor_cutoffs <- apply(
  dplyr::select(train |>  filter(type == tp_wine), fixed_acidity:alcohol),
  2,
  function(x) quantile(x, probs = c(0.05, 0.95))
)
```

Applying winsorization to the test data:

```
ws_test <- test |>
  dplyr::select(fixed_acidity:alcohol) |>
  mutate(across(
    fixed_acidity:alcohol,
    ~pmin(
      pmax(
        .x,
        winsor_cutoffs[1, cur_column()]
      ),
    winsor_cutoffs[2, cur_column()]
    ))
  )
```

Transforming the test data using PCA:

```
scores_test <- as.data.frame(predict(pca, newdata = ws_test)[, 1:3])

scores_test$quality_category <- test$quality_category
```

Testing model predictions:

```
pred_test <- predict(model, newdata = scores_test, type = "class")

table(pred_test, scores_test$quality_category)
```

```
##
## pred_test bad average good
##   bad     257     151   31
##   average 230     398  183
##   good      2      14   29
```

```
mean(pred_test == scores_test$quality_category)
```

```
## [1] 0.5281853
```