551 Project

Jiaqi Sun

Table of contents

1 STATS 551 Project

This is the R Notebook for project of STATS 551 Bayesian Data Analysis at University of Michigan.

In this notebook, I'm going to do Bayesian data analysis on Wine quality dataset from UCI database. A Bayesian hierarchical regression model is used to fit the model. Bayes factor is used to select the variables in the model. Prior predictive and posterior predictive model checking will be applied to check the model. I also made a comparison between Bayesian hierarchical regression model and mixed-effect model, which is the counterpart of mulit-level model in frequentist world.

1.1 Model

I used Bayesian hierarchical regression model to fit the data. Compared to frequentist methods, the Bayesian approach offers several advantages:

- **Prior Information:** Bayesian models allow the integration of prior knowledge or beliefs into the analysis. Here I am going to use weakly informative priors.
- **Uncertainty Quantification:** Bayesian methods provide full posterior distributions of the parameters, offering more information than the point estimation that frequentist methods give.
- **Handling Complexity and Multi-collinearity:** Bayesian methods can handle complex structure data and perform variable selection effectively. In contrast, frequentist methods, especially linear regression, struggle to handle multi-collinearity among predictors, which will lead to unstable estimates.

The hierarchical structure of the model I chose allows **information sharing across different groups**. This sharing enables more robust parameter estimates, especially when some groups have limited data. This is particularly relevant for the wine dataset, where there is a significant imbalance in sample sizes between two categories.

More details of the model setup will be given in the later section.

1.2 Wine Dataset

I chose the wine dataset from UCI Machine learning Repository for this project. It contains chemical and physical measurements of red and white wines derived from three different cultivars grown in the same region of Italy. It also contains the score of each wine assigned by professional wine tasters. Given the dataset includes 11 continuous predictors and a categorical predictor (red or white wine), it is well-suited for exploring Bayesian hierarchical regression models.

First load the necessary packages.

```r
# function to check and load packages
check_and_load <- function(packages) {
  for (pkg in packages) {
    if (!require(pkg, character.only = TRUE)) {
      install.packages(pkg, dependencies = TRUE)
      library(pkg, character.only = TRUE)
    } else {
      library(pkg, character.only = TRUE)
    }
  }
}

packages <- c("dplyr", "ggplot2", "gridExtra", "ggcorrplot", "brms",
              "lme4", "bayesplot")
check_and_load(packages)
```

Load the data sets. The data is separated by ";".

```
data_red <- read.csv("./Stats 551/wine+quality/winequality-red.csv", sep = ";")
data_white <- read.csv("./Stats 551/wine+quality/winequality-white.csv", sep = ";")
```

Check the dimension of the datasets.

```
dim(data_red)
```

```
[1] 1599   12
```

```
dim(data_white)
```

```
[1] 4898   12
```

There are 1599 observations for red wine and 4898 for white wine. The sample size of white wine is three times that of red wine, so it is important to account for this imbalance in the model fitting. Therefore, I chose the Bayesian hierarchical regression model.

1.2.1 Pre-processing

Check if there is any missing data. There is no missing data in either dataset.

```
sum(is.na(data_red))
```

```
[1] 0
```

```
sum(is.na(data_white))
```

```
[1] 0
```

However, the datasets contain duplicated entries. Here I remove the duplicates and check the sample size again. There are 1359 observations for red wine and 3961 for white wine. In total, 5320 observations.

```
data_red <- data_red[!duplicated(data_red), ]
data_white <- data_white[!duplicated(data_white), ]
dim(data_red)
```

```
[1] 1359   12
```

```
dim(data_white)
```

```
[1] 3961   12
```

1.3 Exploratory Data Analysis

1.3.1 Data Description

There are 12 attributes in both data sets. I print the name of the attributes and some example samples.

```
head(data_red)
```

```
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
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
6           7.4             0.66        0.00            1.8     0.075
7           7.9             0.60        0.06            1.6     0.069
  free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
1                  11                   34  0.9978 3.51      0.56     9.4
2                  25                   67  0.9968 3.20      0.68     9.8
3                  15                   54  0.9970 3.26      0.65     9.8
4                  17                   60  0.9980 3.16      0.58     9.8
6                  13                   40  0.9978 3.51      0.56     9.4
7                  15                   59  0.9964 3.30      0.46     9.4
  quality
1       5
2       5
```

```
3         5
4         6
6         5
7         5
```

Here I listed the description of all the attributes in the dataset with the unit.

- `fixed.acidity` (continuous g/dm^3): The non-volatile acids present in wine, primarily tartaric acid. It contributes to the overall acidity and freshness of the wine.
- `volatile.acidity` (continuous g/dm^3): The amount of acetic acid in wine.
- `citric.acid` (continuous g/dm^3): The amount of citric acid in wine.
- `residual.sugar` (continuous g/dm^3): The amount of sugar left in wine after fermentation.
- `chlorides` (continuous g/dm^3): The amount of salt in wine.
- `free.sulfur.dioxide` (continuous mg/dm^3): The amount of sulfur dioxide (SO_2) that is not bound to other molecules in the wine. Free SO_2 acts as an antimicrobial and antioxidant, protecting the wine from spoilage.
- `total.sulfur.dioxide` (continuous mg/dm^3): The total amount of SO_2 in the wine, including both free and bound forms.
- `density` (continuous g/cm^3): The density of the wine.
- `pH` (continuous): The acidity or alkalinity of the wine, ranging from 0 to 14.
- `sulphates` (continuous g/dm^3): The amount of sulphates in the wine.
- `alcohol` (continuous % vol): The alcohol content of the wine measured in percentage.
- `type` (categorical): The additional variable that will be created to represent the type of the wine.
- `quality` (categorical): The subjective score assigned to the wine by experts (on a scale from 0 to 10). Higher scores indicate better perceived quality.

```r
summary(data_red)
summary(data_white)
```

Calculate the standard error of the attribute.

```r
sd_values_red <- apply(data_red, 2, sd)
sd_values_white <- apply(data_white, 2, sd)
sd_values_red
sd_values_white
```

Here I summarized the datasets in the following tables. Since the limitation of table in Markdown, this table may not in the perfect academic style. And due to the length limit, the table is separated into two.

Summary statistics for red wine

Attribute	Red Wine Min	Red Wine Max	Red Wine Median	Red Wine Mean	Standard Deviation
Fixed Acidity	4.60	15.90	7.90	8.31	1.7370
Volatile Acidity	0.12	1.58	0.52	0.53	0.1830
Citric Acid	0.00	1.00	0.26	0.27	0.1955
Residual Sugar	0.90	15.50	2.20	2.52	1.3523
Chlorides	0.012	0.611	0.079	0.088	0.0494
Free Sulfur Dioxide	1.00	72.00	14.00	15.89	10.4473
Total Sulfur Dioxide	6.00	289.00	38.00	46.83	33.4089
Density	0.9901	1.0037	0.9967	0.9967	0.0019
pH	2.740	4.010	3.31	3.31	0.1550
Sulphates	0.33	2.00	0.62	0.6587	0.1707
Alcohol	8.40	14.90	10.20	10.43	1.0821

Summary statistics for white wine

Attribute	White Wine Min	White Wine Max	White Wine Median	White Wine Mean	Standard Deviation
Fixed Acidity	3.80	14.20	6.80	6.84	0.8669
Volatile Acidity	0.08	1.10	0.26	0.28	0.1034
Citric Acid	0.00	1.66	0.32	0.33	0.1224
Residual Sugar	0.60	65.80	4.70	5.92	4.8616
Chlorides	0.009	0.346	0.042	0.046	0.0231
Free Sulfur Dioxide	2.00	289.00	33.00	34.89	17.2100
Total Sulfur Dioxide	9.00	440.00	133.00	137.20	43.1291
Density	0.9871	1.0390	0.9935	0.9938	0.0029
pH	2.720	3.820	3.180	3.195	0.1515
Sulphates	0.2200	1.0800	0.4800	0.4904	0.1135
Alcohol	8.00	14.20	10.40	10.59	1.2171

Based on the summary statistics, several key observations are listed below:

- Red wine has a higher fixed acidity on average (mean 8.31) compared to the white wine (mean 6.84).
- Red wine has a significantly higher average volatile acidity (mean 0.53, median 0.52) compared to white wine (mean 0.28, median 0.26). This suggests that red wines might have more acetic acid.
- White wine has a much higher average residual sugar (mean 5.92) compared to red wine (mean 2.52). This means that white wines tend to be sweeter. What's more, the maximum residual sugar for white wine is 65.8, which is extremely higher than that of red wine (max 15.5). The standard error of residual sugar of white wine is also larger than red wine (4.86 v.s 1.35), which indicates that the sugar content varies widely between samples in white wine.
- White wine has much higher levels of both free sulfur dioxide (mean 34.89 v.s 15.89 in red wine) and total sulfur dioxide (mean 137.2 v.s 46.83 in red wine) than the red wine.
- The density and pH of the wine are basically the same for both wine. However, red wine has a larger range of pH values.

Note that, quality is a categorical variable, the statistics of it were omitted.

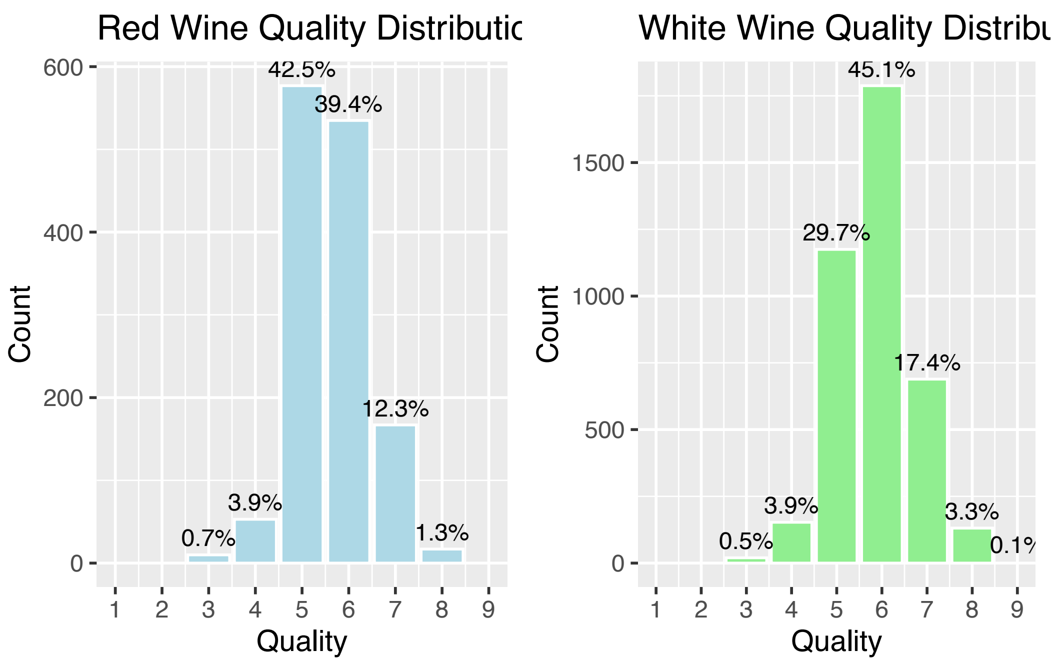
1.3.2 Visualization

Next, I visualize the dataset using plots like histograms, heat maps and box plots.

Draw the histogram of the quality of the wine. The percentage of each score is presented at the top of each bar.

```r
# calculate the probability
data_red_freq <- data_red %>%
  count(quality) %>%
  mutate(percentage = n / sum(n) * 100)

data_white_freq <- data_white %>%
  count(quality) %>%
  mutate(percentage = n / sum(n) * 100)

# hist plot for quality
hist_red <- ggplot(data_red_freq, aes(x = quality, y = n)) +
  geom_bar(stat = "identity", fill = "lightblue", color = "white", na.rm = TRUE) +
  geom_text(aes(label = paste0(round(percentage, 1), "%")),
            vjust = -0.5, size = 3, na.rm = TRUE) +
  labs(title = "Red Wine Quality Distribution", x = "Quality", y = "Count") +
  scale_x_continuous(breaks = seq(1, 9, by = 1), limits = c(1, 9))
```

```r
hist_white <- ggplot(data_white_freq, aes(x = quality, y = n)) +
  geom_bar(stat = "identity", fill = "lightgreen", color = "white", na.rm = TRUE) +
  geom_text(aes(label = paste0(round(percentage, 1), "%")),
            vjust = -0.5, size = 3, na.rm = TRUE) +
  labs(title = "White Wine Quality Distribution", x = "Quality", y = "Count") +
  scale_x_continuous(breaks = seq(1, 9, by = 1), limits = c(1, 9))

grid.arrange(hist_red, hist_white, ncol = 2)
```



The x-axis is the quality of wine, y-axis is the number of observations, and the percentage of each bar is displayed at the top. From the plots, I may conclude that

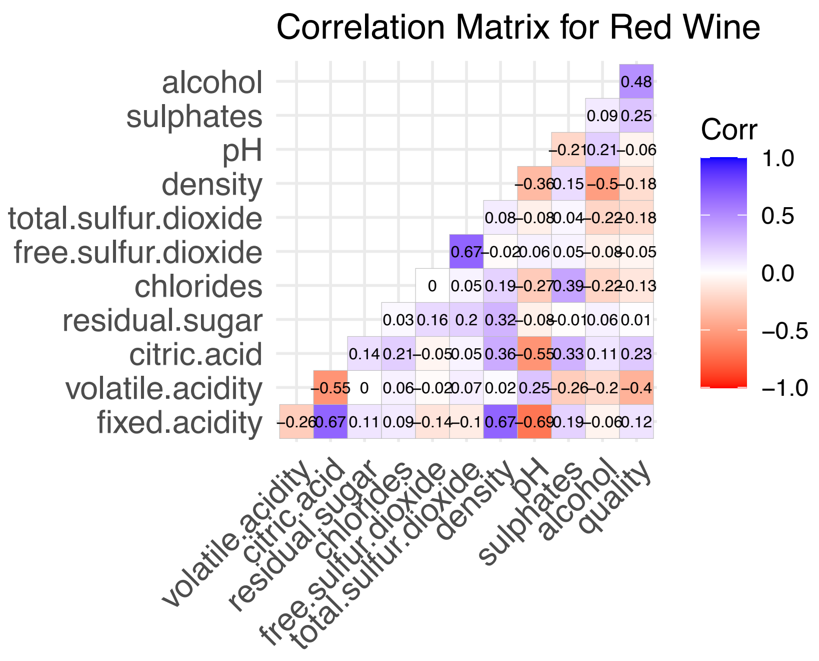
- For red wine, quality 5 is the most frequent, accounting for 42.5% of the red samples. Quality 6 follows closely behind at 39.4%.
- For white wine, quality 6 is the most common, accounting for 45.1%. Quality 5 follows at 29.7%.

Overall, white wine tends to have more high-quality samples.

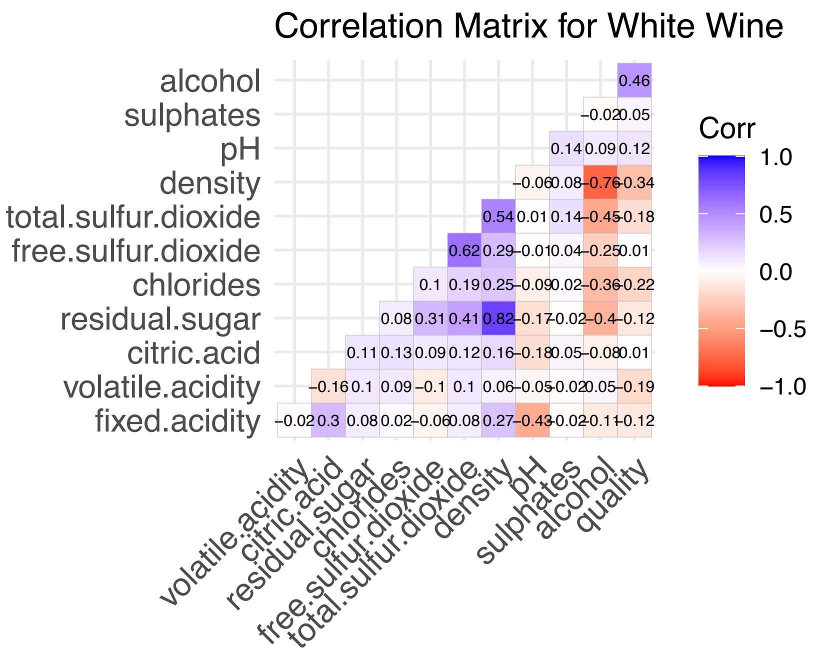
1.3.3 Correlations

To explore the linear relationship between attributes, I plotted the heatmap for correlations between attributes.

```r
# calculate the correlation matrix, make sure the order of columns
common_cols <- intersect(colnames(data_red), colnames(data_white))
data_red <- data_red[, common_cols]
data_white <- data_white[, common_cols]

cor_matrix_red <- cor(data_red)
cor_matrix_white <- cor(data_white)

# correlation heat map
ggcorrplot(cor_matrix_red, hc.order = FALSE, type = "lower", lab = TRUE,
           lab_size = 2,  # adjust the size
           colors = c("red", "white", "blue"),
           title = "Correlation Matrix for Red Wine")
```



```r
ggcorrplot(cor_matrix_white, hc.order = FALSE, type = "lower", lab = TRUE,
           lab_size = 2,
           colors = c("red", "white", "blue"),
           title = "Correlation Matrix for White Wine")
```



The correlation heat map represents the strength and direction of the linear relationship between two attributes. The correlations are between -1 to 1. This is a good way to explore the linear relationship and detect the multi-collinearity problem.

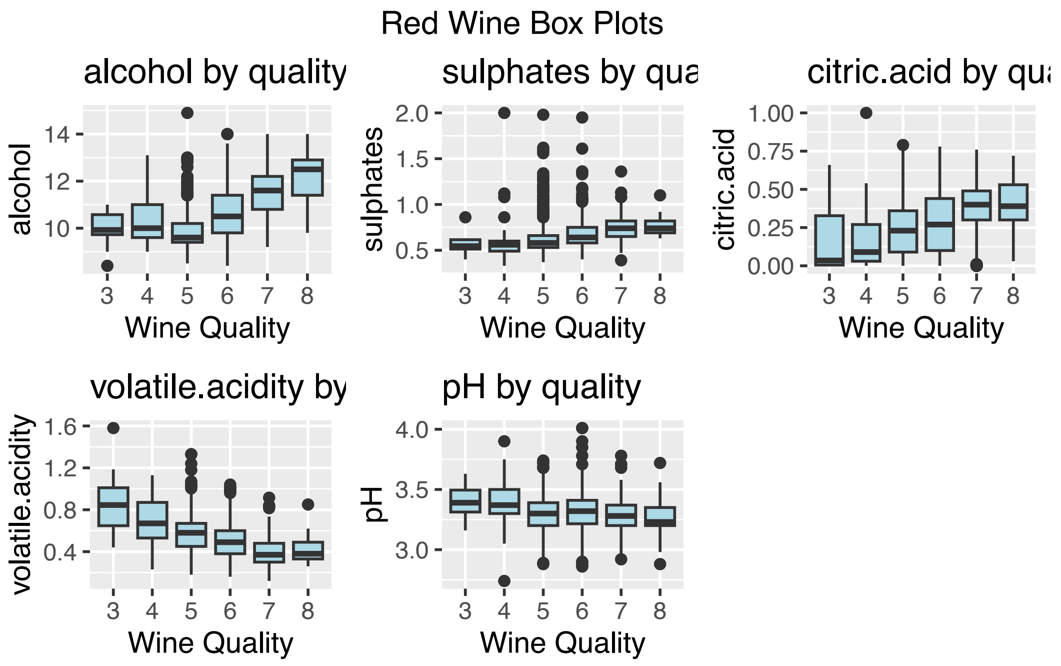
- For red wine, `density` and `fixed.acidity` have a strong positive correlation (0.67). (The attributes `fixed.acidity` and `citric.acid`, `free.sulfur.dioxide` and `total.sulfur.dioxide` also have strong positive correlation, since the pairs of attributes represent similar property or have affiliation. There is a positive correlation between `alcohol` and wine quality (0.48), implying that wines with higher alcohol content tend to have better quality scores.
- For red wine, the attributes related with acid have negative correlation with `pH`, which is consistent with common sense. A moderate negative correlation (-0.40) is seen between `volatile.acidity` and quality, indicating that wines with higher volatile acidity tend to have lower quality scores.
- For white wine, we can also see a positive relation (0.46) between `alcohol` and wine quality. There is a strong positive (0.82) correlation between `residual.sugar` and `density`, indicating that higher residual sugar content results in higher density, which is expected.

Next I visualized the relations between attributes and the quality of wine using box plot. This approach clearly highlights the characteristics of wine with different quality levels. For instance, we can observe how alcohol content tends to increase with higher wine quality, while attributes such as volatile acidity often show a reverse trend, decreasing as quality improves.

```r
# list of attributes to plot against quality
attributes <- c("alcohol", "sulphates", "citric.acid", "volatile.acidity", "pH")

plot_list <- list()

for (attribute in attributes) {
  p <- ggplot(data_red, aes_string(x = "as.factor(quality)", y = attribute)) +
    geom_boxplot(fill = "lightblue") +
    labs(title = paste(attribute, "by quality"),
         x = "Wine Quality", y = attribute)

  plot_list[[attribute]] <- p
}
```

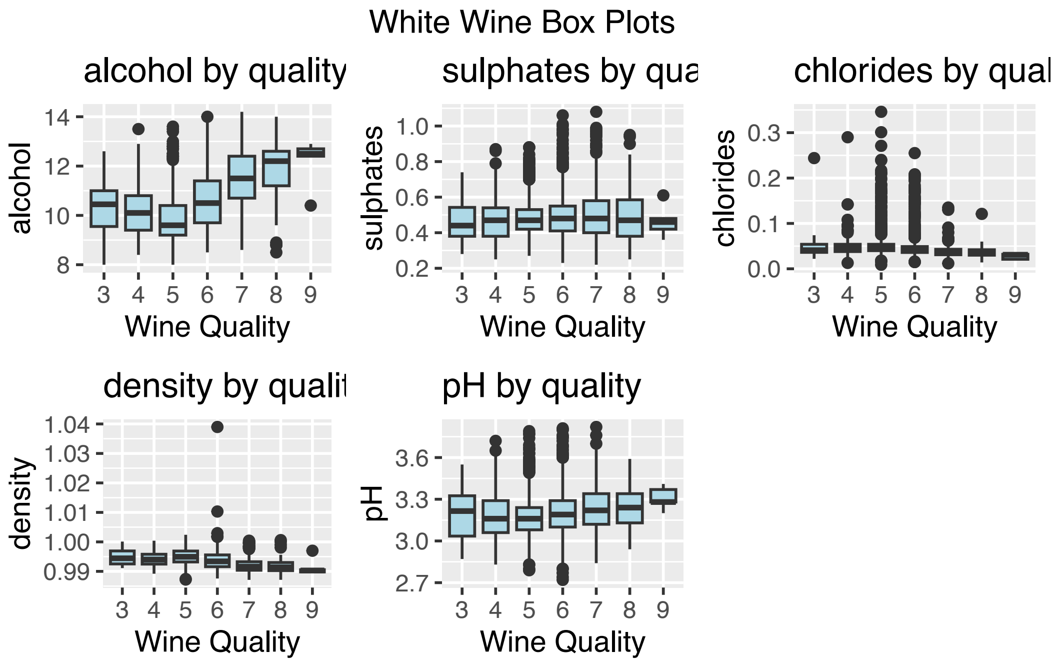
```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
i Please use tidy evaluation idioms with `aes()`.
i See also `vignette("ggplot2-in-packages")` for more information.
```

```r
grid.arrange(grobs = plot_list, ncol = 3, top = "Red Wine Box Plots")
```



For red wine, `alcohol`, `sulphates` and `citric.acid` have positive correlation with quality, while `volatile.acidity` and `pH` have negative correlation. These match with the correlation plot.

```
# list of attributes to plot against quality
attributes <- c("alcohol", "sulphates", "chlorides", "density", "pH")
plot_list <- list()

for (attribute in attributes) {
  p <- ggplot(data_white, aes_string(x = "as.factor(quality)", y = attribute)) +
    geom_boxplot(fill = "lightblue") +
    labs(title = paste(attribute, "by quality"),
         x = "Wine Quality", y = attribute)

  plot_list[[attribute]] <- p
}

grid.arrange(grobs = plot_list, ncol = 3, top = "White Wine Box Plots")
```



For white wine, `alcohol` and `pH` have positive correlation with quality. The relation between `pH` and `quality` is opposite compared with red wine. `chlorides` and `density` have negative correlation with quality.

1.3.4 Summary of Wine Dataset

- White wine tends to have much higher sulfur dioxide, sugar than red wine, and much lower volatile acidity than red wine.
- Overall, white wine tends to have more higher-quality samples than the red wine. The percentage of alcohol in wine has positive correlation with quality in both wine. But the influence of pH has opposite directions in two wines, which is positive in white wine but negative in red wine. Higher volatile acidity reduces the quality of both wine.
- If a model is trained to predict the quality of wine using the attributes in the future, multi-collinearity could be a potential problem. Some attributes have high correlations, such as `free.sulfur.dioxide` and `total.sulfur.sioxide`.
- The data sets are unbalanced since the observations of white wine is 3 times the number of red wine.

1.4 Bayesian Hierarchical Regression and Model Selection

In this section, I will describe the model I used, prior I chose, and the selection of model variables using Bayes factor. Then the model is fitted using `brms` package and perform model checks.

1.4.1 Model Setup

The response `quality` in the dataset is a categorical variable. To simplify the model, I treat it as a continuous score and assume it follows a normal distribution. This assumption allows for the application of regression techniques. Let j represent the wine type, where $j = 1$ for red wine and $j = 2$ for white wine. The Bayesian hierarchical regression model is shown in the following equation.

$$\text{quality}_{ij} = \beta_0 + u_j + \beta^T x_{ij} + \epsilon_{ij}$$

where

- quality_{ij} is the quality of the wine for the i-th observation in the j-th wine type.
- x_{ij} is a vector contains the predictor variables (e.g., `alcohol`, `ph`, etc).
- β_0 is the fixed intercept.
- β is a vector contains the fixed effect coefficients for all predictors.
- u_j is the random effect for the wine type j, capturing the variability between red and white wines.
- ϵ_{ij} is the residual error term, accounting for unexplained variability.

The fixed-effect parameters β_0 and β capture the overall trends or population-level effects. On the other hand, the random effect accounts for group-level variability, allowing the model to adjust for deviations specific to each group. This hierarchical structure enables sharing of information across groups.

1.4.2 Prior Selection

The predictors are normalized to improve model stability and accelerate the convergence of iterative algorithm. Normalization also makes different variables comparable. Additionally, in Bayesian models, variables on different scales will complicate the selection of priors.

```r
# combine the two datasets
data_red$wine_type <- "red"
data_white$wine_type <- "white"
data_wine <- bind_rows(data_red, data_white) %>%
  mutate(wine_type = factor(wine_type, levels = c("red", "white")))

# scale the data
data_wine_scaled <- data_wine
data_wine_scaled[, c("alcohol", "volatile.acidity", "residual.sugar", "chlorides",
                     "free.sulfur.dioxide", "total.sulfur.dioxide", "density",
                     "pH", "sulphates", "fixed.acidity", "citric.acid")] <-
  scale(data_wine[, c("alcohol", "volatile.acidity", "residual.sugar", "chlorides",
                      "free.sulfur.dioxide", "total.sulfur.dioxide", "density",
                      "pH", "sulphates", "fixed.acidity", "citric.acid")])
```

Prior selection

- Prior for fixed intercept β_0: Since the quality score ranges from 0 to 10, I set the mean of prior distribution of the intercept as 5. I chose $\mathrm{N}(5, 2)$ as the prior distribution.
- Prior for fixed effect coefficients β: Since all the predictors are normalized, a same prior can be assigned to all the fixed effect coefficients.I chose $\mathrm{N}(0, 1)$ as the prior distribution.
- Prior for random effect u_j: A $\text{Cauchy}(0, 0.5)$ distribution is used here for its heavier tails.

Due to the lack of additional information, I chose relatively weakly informative priors for the parameters, and let the prior predictive distribution covers the range from 0 to 10 for quality. By doing so, the priors provide regularization while avoiding overly strong assumptions that might bias the results.

```r
# bayesian model formula
full_formula <- quality ~ (1 | wine_type) +
  alcohol + volatile.acidity + residual.sugar + chlorides +
  free.sulfur.dioxide + total.sulfur.dioxide + density +
  pH + sulphates + fixed.acidity + citric.acid

# priors
priors <- c(
  prior(normal(5, 2), class = "Intercept"),  # prior for intercept
  prior(normal(0, 1), class = "b"),          # prior for coefficients
  prior(cauchy(0, 0.5), class = "sd")          # prior for random effect
  )
```

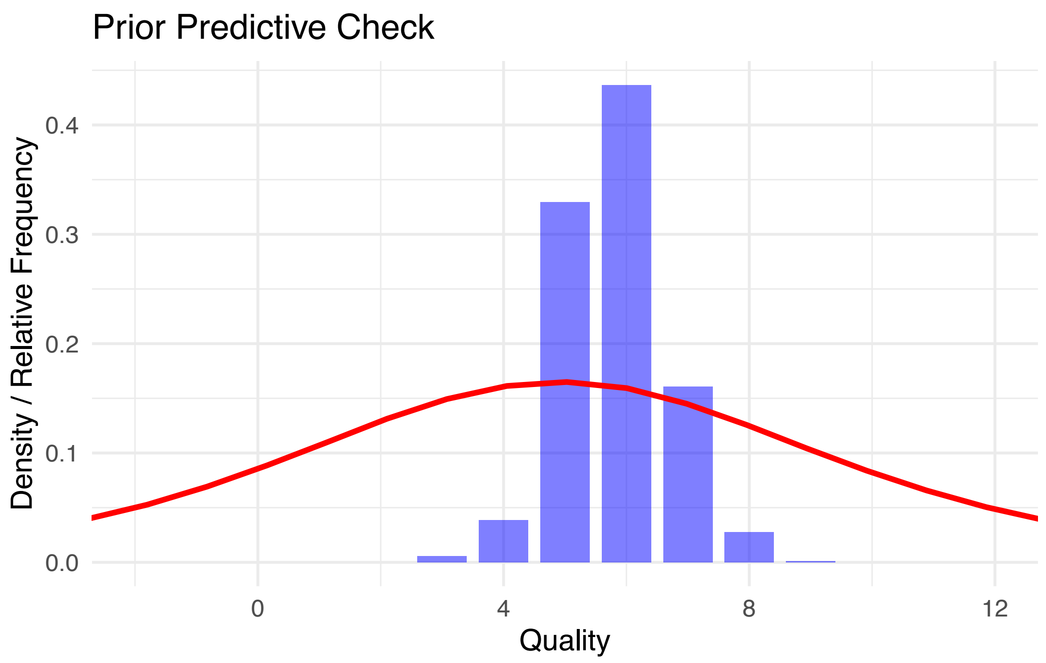
1.4.3 Prior Predictive Check

Fit the model for prior predictive check.

```r
# fit the model for prior predictive check
model_prior <- brm(
  formula = full_formula,
  data = data_wine_scaled,
  prior = priors,
  sample_prior = "only",  # only sample from the prior
  iter = 3000,
  chains = 4,
  cores = 4
)

# get the prior predictive samples
prior_pred_samples <- posterior_predict(model_prior, ndraws = 1000)
```

Perform the prior predictive model checking.

```r
# prior predictive check
ggplot(data_wine, aes(x = quality)) +
  geom_bar(aes(y = after_stat(count / sum(count))),
           fill = "blue", alpha = 0.5, width = 0.8) +  # real data
  geom_density(data = data.frame(prior_pred_samples = as.vector(prior_pred_samples)),
               aes(x = prior_pred_samples, y = after_stat(density)),
               color = "red", linewidth = 1, alpha = 0.5) +  # prior predictive density
  coord_cartesian(xlim = c(-2, 12)) +
  labs(title = "Prior Predictive Check",
       x = "Quality",
       y = "Density / Relative Frequency") +
  theme_minimal()
```



The blue bars represent the histogram of the real data, and the red curve represents the density of the prior predictive dsitribution. As the above figure shows, the prior predictive distribution covers the range of `quality` variable well. The prior is not very informative, therefore, it cannot fit the true data well.

1.4.4 Model Selection

There are many descriptive predictors in the wine dataset, some of which represent similar substances, for example `free.sulfur.dioxide` and `total.sulfur.dioxide`. To address this, model selection is necessary. I used the `brms` package to calculate the Bayes Factor, which is a statistical tool for comparing the relative strength of evidence between models. The Bayes Factor compares the likelihood of the data under two competing hypotheses (models), with higher values indicating stronger evidence for one model over the other. By calculating the Bayes Factor, we can evaluate which model best fits the data while accounting for potential multicollinearity, helping to select a more robust and interpretable model.

Fit the full Bayesian hierarchical regression model as the base line.

```r
full_brms = brm(formula = full_formula,
                data = data_wine_scaled, iter = 3000,
                chains = 4, cores = 4,
                prior = priors,
                save_pars = save_pars(all = TRUE)
                )
```

Here I define a function that can automatically calculate the Bayes Factor of the full model with each "drop-one variable" model.

```r
# function to calculate Bayes Factor by removing one predictor
calculate_bf <- function(full_model, data, predictors) {
  bf_results <- list()

  # loop through each predictor and remove it
  for (predictor in predictors) {
    reduced_formula <- as.formula(paste("quality ~ (1 | wine_type) +",
                                        paste(setdiff(predictors, predictor),
                                              collapse = " + ")))

    # fit the reduced model
    reduced_model <- brm(formula = reduced_formula,
                         data = data, iter = 3000,
                         prior = priors,
                         chains = 4, cores = 4,
                         save_pars = save_pars(all = TRUE))

    # compute the Bayes Factor
    bf <- bayes_factor(full_model, reduced_model)
    bf_results[[predictor]] <- bf
  }
  print(bf)
  return(bf_results)
}

# the list of predictors
predictors <- c("alcohol", "volatile.acidity", "residual.sugar", "chlorides",
                "free.sulfur.dioxide", "total.sulfur.dioxide", "density",
                "pH", "sulphates", "fixed.acidity", "citric.acid")

# perform the selection
bf_results <- calculate_bf(full_brms, data_wine_scaled, predictors)
```

The model comparison results. If the Bayes factor is greater than 1, it means the full model is preferred. The larger the Bayes factor is, the stronger evidence we have to choose the full model. If the Bayes factor is less than 1, the reduced model is preferred.

```
# print the model comparison results
bf_results
```

```
$alcohol
Estimated Bayes factor in favor of full_model over reduced_model: 39019720714314512563991216

$volatile.acidity
Estimated Bayes factor in favor of full_model over reduced_model: 27225889812392391962461835

$residual.sugar
Estimated Bayes factor in favor of full_model over reduced_model: 26764755860749.17578

$chlorides
Estimated Bayes factor in favor of full_model over reduced_model: 0.42870

$free.sulfur.dioxide
Estimated Bayes factor in favor of full_model over reduced_model: 488476395.77497

$total.sulfur.dioxide
Estimated Bayes factor in favor of full_model over reduced_model: 447.21433

$density
Estimated Bayes factor in favor of full_model over reduced_model: 23196404.81867

$pH
Estimated Bayes factor in favor of full_model over reduced_model: 4075570.99364

$sulphates
Estimated Bayes factor in favor of full_model over reduced_model: 1976203423238606.50000

$fixed.acidity
Estimated Bayes factor in favor of full_model over reduced_model: 604.15778

$citric.acid
Estimated Bayes factor in favor of full_model over reduced_model: 0.03398
```

From the above results, the variables `chlorides` and `citric.acid` have very small Bayes factor (0.43 and 0.03 respectively), indicating that we should delete them from the full model.

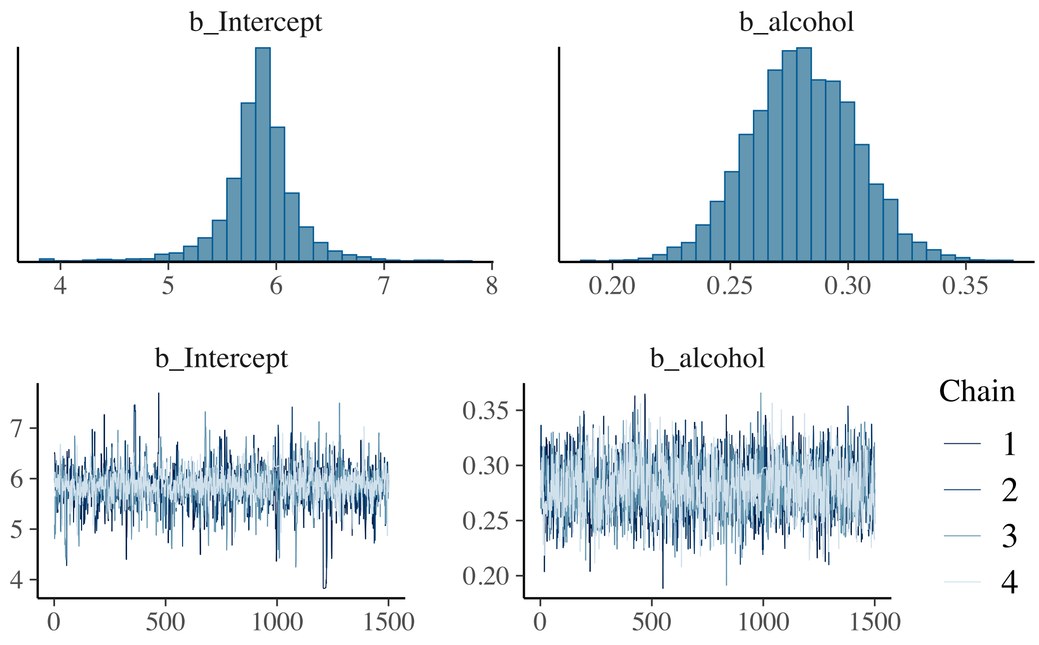
The rest Bayes factors are very large (all greater than 100). This means they contribute a lot to the regression model.

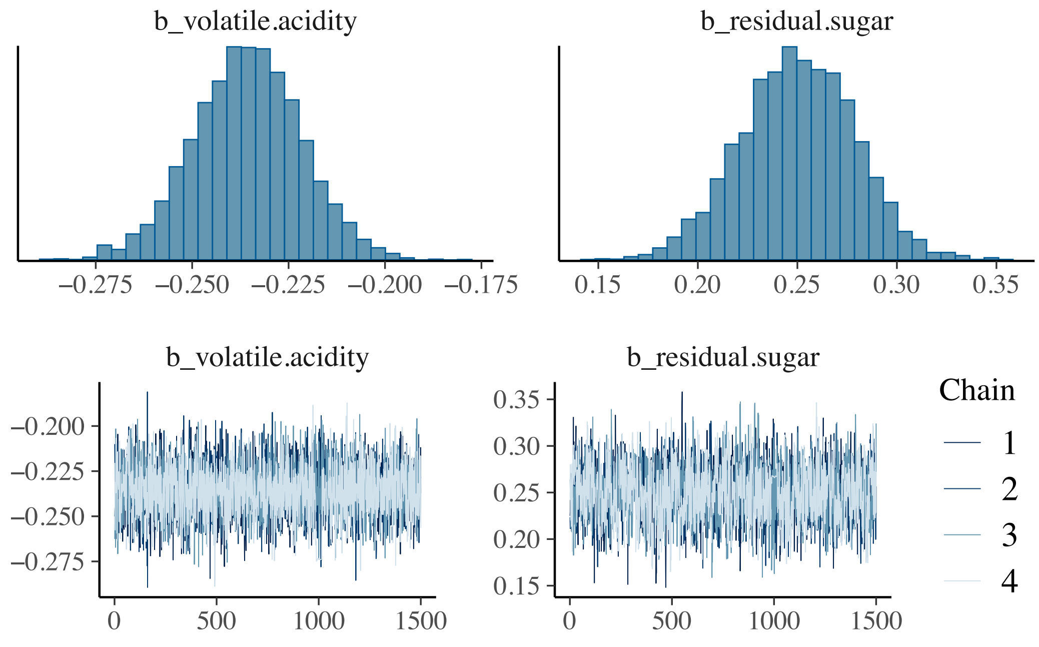
Here we fit the final Bayesian hierarchical regression model according to the above model selection result.

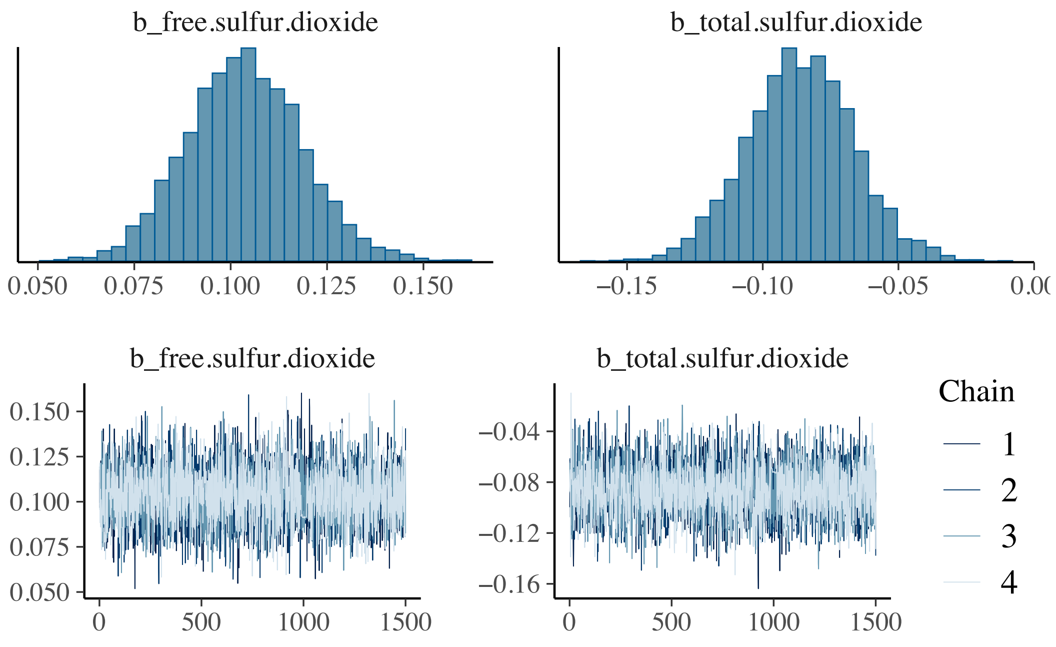
```r
# final model formula
final_model_formula <- quality ~ (1 | wine_type) + alcohol +
  volatile.acidity + residual.sugar + free.sulfur.dioxide +
  total.sulfur.dioxide + density + pH + sulphates + fixed.acidity

# fit the final model
final_brms = brm(formula = final_model_formula,
                data = data_wine_scaled, iter = 3000,
                chains = 4, cores = 4,
                prior = priors,
                save_pars = save_pars(all = TRUE),
                control = list(adapt_delta = 0.95)
                )
```

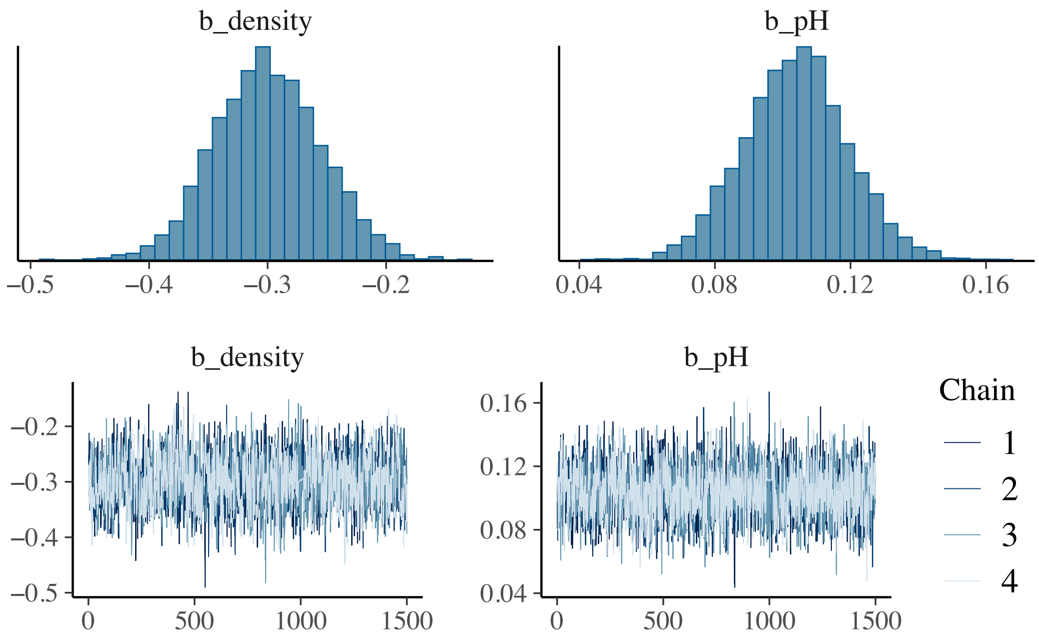
Check the histograms and trace plots of the simulation. Since the `plot` function in `brms` package cannot control how many plots in one figure, here I define my own function to plot them.

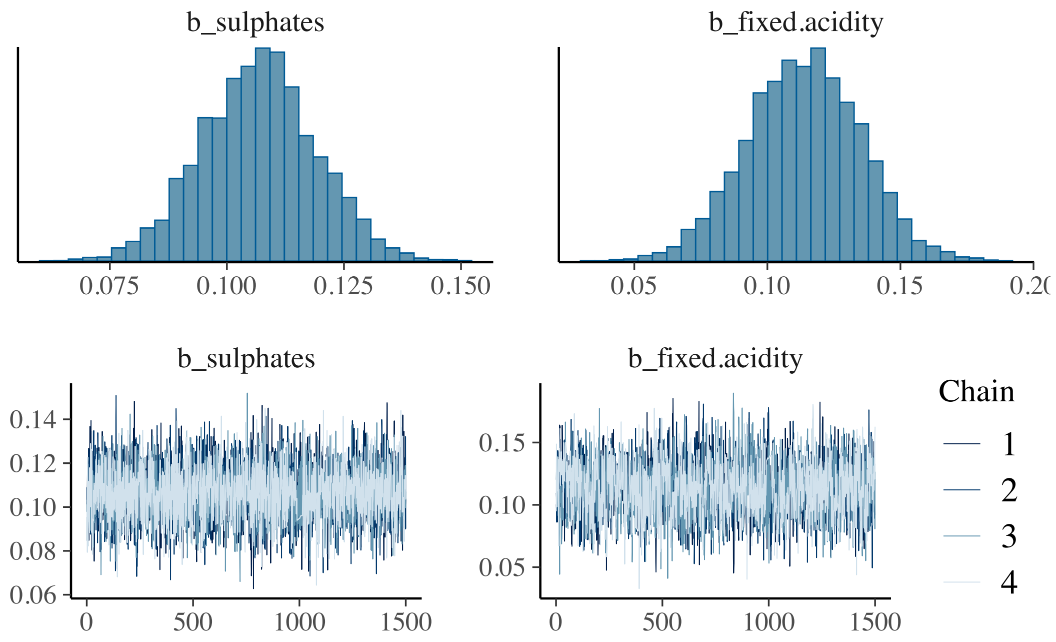
```r
# check convergence
posterior_samples <- as.array(final_brms)

# define function to
plot_two_variables_with_chains <- function(var1, var2, posterior_samples) {
  selected_samples <- posterior_samples[, , c(var1, var2)]

  # histogram
  hist_plot <- bayesplot::mcmc_hist(
    selected_samples,
    facet_args = list(ncol = 2),
    bins = 30
  )

  # trace plot
  trace_plot <- bayesplot::mcmc_trace(
    selected_samples,
    facet_args = list(ncol = 2),
    size = 0.5,
  )

  grid.arrange(hist_plot, trace_plot, nrow = 2)
}

variables <- dimnames(posterior_samples)$variable[1:14]
for (i in seq(1, length(variables), by = 2)) {
  var1 <- variables[i]
  var2 <- ifelse(i + 1 <= length(variables), variables[i + 1], NA)
  if (!is.na(var2)) {
    plot_two_variables_with_chains(var1, var2, posterior_samples)
  }
}
```

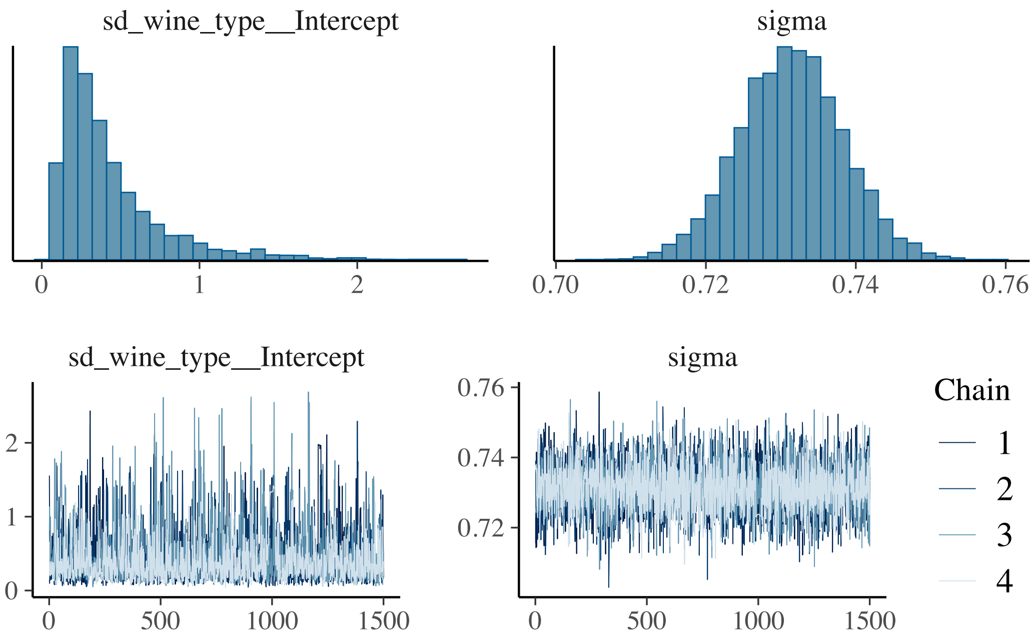


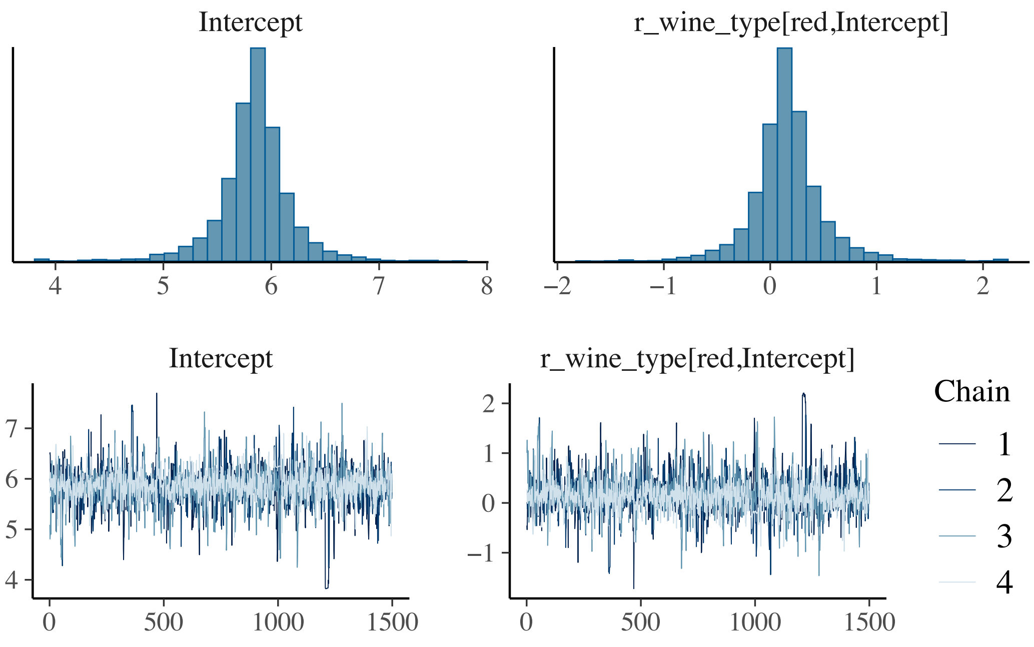

b_volatile.acidity
−0.275
−0.250
−0.225
−0.200
−0.175
b_residual.sugar
0.15
0.20
0.25
0.30
0.35
b_volatile.acidity
−0.200
−0.225
−0.250
−0.275
0
500
1000
1500
b_residual.sugar
0.35
0.30
0.25
0.20
0.15
0
500
1000
1500
Chain
1
2
3
4


b_free.sulfur.dioxide
0.050
0.075
0.100
0.125
0.150
b_total.sulfur.dioxide
−0.15
−0.10
−0.05
0.0
b_free.sulfur.dioxide
0.150
0.125
0.100
0.075
0.050
0
500
1000
1500
b_total.sulfur.dioxide
−0.04
−0.08
−0.12
−0.16
0
500
1000
1500
Chain
1
2
3
4


b_density
b_pH
−0.5
−0.4
−0.3
−0.2
0.04
0.08
0.12
0.16
b_density
b_pH
Chain
1
2
3
4
0
500
1000
1500


b_sulphates
b_fixed.acidity
0.075
0.100
0.125
0.150
0.05
0.10
0.15
b_sulphates
b_fixed.acidity
Chain
1
2
3
4
0.06
0.08
0.10
0.12
0.14
0
500
1000
1500





The trace plots for each parameter show that all chains overlap substantially and exhibit no obvious trends, indicating good mixing among chains. The model has converged.

Here I summarize the model.

```r
# summary of the model parameters
summary(final_brms)
```

```
Warning: There were 80 divergent transitions after warmup. Increasing
adapt_delta above 0.95 may help. See
http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: quality ~ (1 | wine_type) + alcohol + volatile.acidity + residual.sugar + free.sulfu
   Data: data_wine_scaled (Number of observations: 5320)
  Draws: 4 chains, each with iter = 3000; warmup = 1500; thin = 1;
         total post-warmup draws = 6000

Multilevel Hyperparameters:
~wine_type (Number of levels: 2)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.43      0.34     0.09     1.41 1.00     1717     1096

Regression Coefficients:
                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
Intercept                5.84      0.34     5.09     6.51 1.01     1954
alcohol                  0.28      0.02     0.24     0.33 1.00     2265
volatile.acidity        -0.24      0.01    -0.26    -0.21 1.00     3086
residual.sugar           0.25      0.03     0.19     0.30 1.00     2195
free.sulfur.dioxide      0.10      0.01     0.08     0.13 1.00     3829
total.sulfur.dioxide    -0.09      0.02    -0.12    -0.05 1.00     3577
density                 -0.30      0.04    -0.39    -0.21 1.00     1996
pH                       0.10      0.02     0.07     0.13 1.00     2781
sulphates                0.11      0.01     0.08     0.13 1.00     4795
fixed.acidity            0.11      0.02     0.07     0.15 1.00     2259
                     Tail_ESS
Intercept                1701
alcohol                  3335
volatile.acidity         1518
residual.sugar           3691
free.sulfur.dioxide      4192
total.sulfur.dioxide     4091
density                  3111
pH                       3918
sulphates                4280
```

```
fixed.acidity            3536

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     0.73      0.01     0.72     0.75 1.00     5691     4317

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

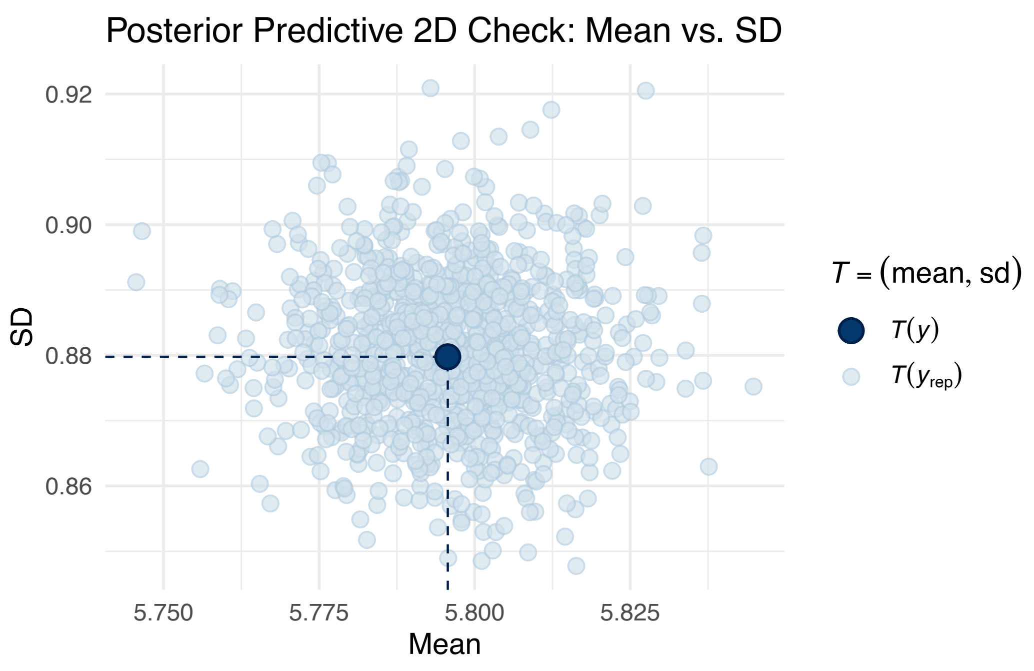
From the above summary one may conclude that

- The `Rhat` is exactly 1 for all the parameters, indicating good convergence across chains.
- Both `Bulk_ESS` and `Tail_ESS` are reasonably large for all parameters, ensuring that the parameter estimates are reliable.
- `alcohol`, `volatile.acidity`, `residual.sugar`, `free.sulfur.dioxide`, `total.sulfur.dioxide`, `density`, `pH`, `sulphates`, `fixed.acidity` all the predictors have significant effects on wine quality, as indicated by the 95% credible intervals not crossing zero.
- `volatile.acidity` and `density` have negative effects on the wine quality, and `total.sulfur.dioxide` has smaller negative effects (-0.08). While others like `alcohol`, `residual.sugar`, etc., have positive effects.

1.4.5 Posterior Predictive check

Next I make a scatter plot for mean and standard deviation of the posterior samples.

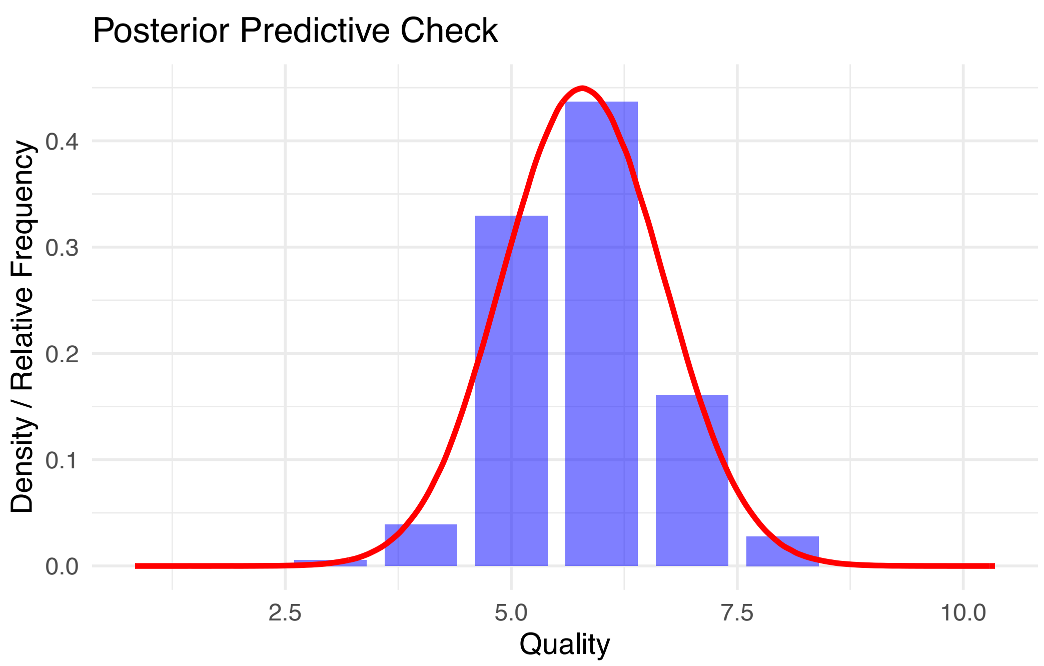
```r
posterior_samples <- posterior_predict(final_brms, ndraws = 1000)

# scatter plot for mean and standard deviation of the posterior samples.
ppc_stat_2d(y = data_wine$quality, yrep = posterior_samples, stat = c("mean", "sd")) +
  labs(title = "Posterior Predictive 2D Check: Mean vs. SD",
       x = "Mean",y = "SD") +
  theme_minimal()
```



The mean and standard deviation of the real data lie exactly in the center of the scatter plot.

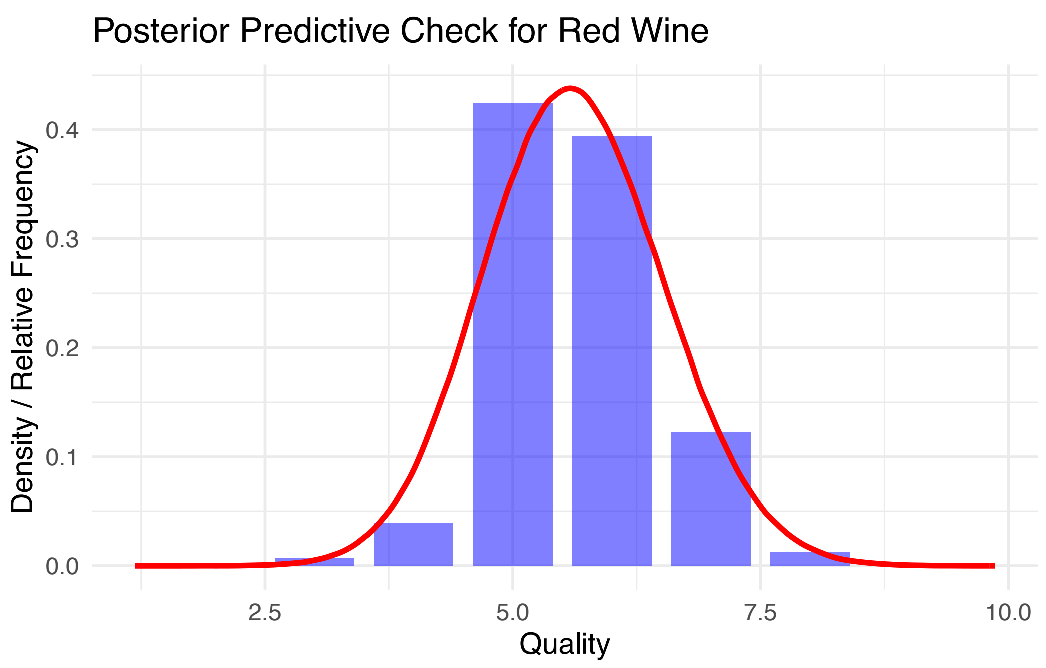
Here I plot the posterior predictive distribution density over the histogram of the real data.

```r
# get the posterior predictive samples
posterior_samples <- posterior_predict(final_brms, ndraws = 1000)

posterior_df <- data.frame(posterior_samples = as.vector(posterior_samples))

# posterior predictive check
ggplot() +
  geom_bar(data = data_wine, aes(x = quality, y = after_stat(count / sum(count))),
           fill = "blue", alpha = 0.5, width = 0.8, stat = "count") +
  geom_density(data = posterior_df, aes(x = posterior_samples, y = after_stat(density)),
               color = "red", linewidth = 1) +
  labs(title = "Posterior Predictive Check",
       x = "Quality",
       y = "Density / Relative Frequency") +
  theme_minimal()
```

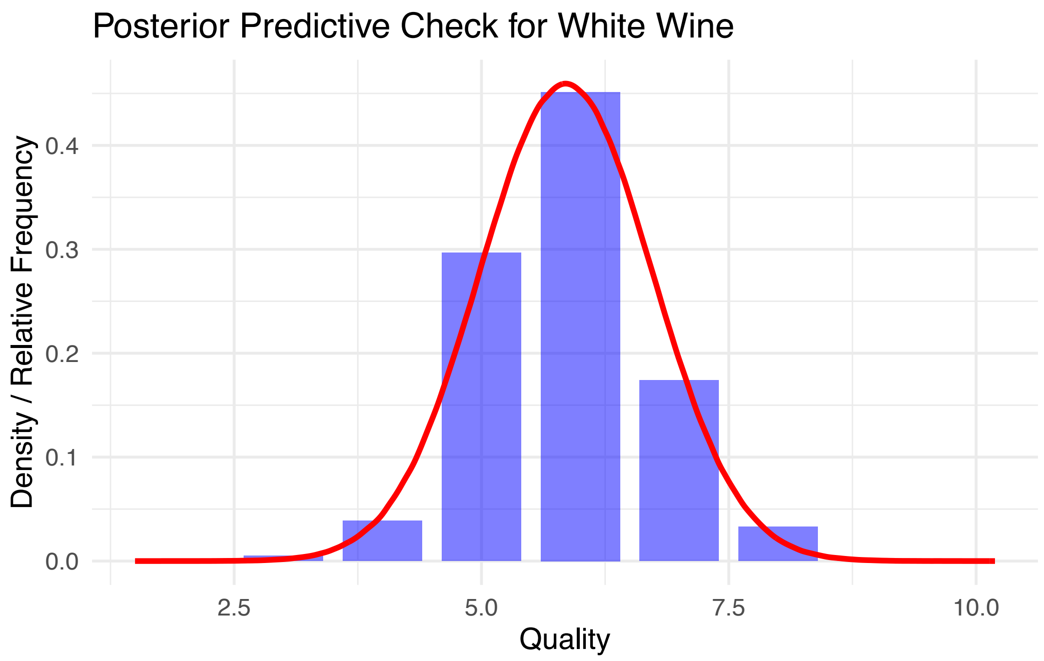


Plot the posterior predictive check for each type of wine to check the fit of each sub group.

```
# get the posterior predictive samples for each wine
posterior_samples_red_wine <- posterior_predict(final_brms, ndraws = 1000, newdata = data_wi
posterior_samples_white_wine <- posterior_predict(final_brms, ndraws = 1000, newdata = data_

posterior_df_red <- data.frame(posterior_samples = as.vector(posterior_samples_red_wine))
posterior_df_white <- data.frame(posterior_samples = as.vector(posterior_samples_white_wine)

# real data
data_wine_red <- data_wine[data_wine$wine_type == "red", ]
data_wine_white <- data_wine[data_wine$wine_type == "white", ]

# posterior predictive check for red wine
ggplot() +
  geom_bar(data = data_wine_red, aes(x = quality, y = after_stat(count / sum(count))),
           fill = "blue", alpha = 0.5, width = 0.8, stat = "count") +
  geom_density(data = posterior_df_red, aes(x = posterior_samples, y = after_stat(density)),
               color = "red", linewidth = 1) +
  labs(title = "Posterior Predictive Check for Red Wine",
       x = "Quality",
       y = "Density / Relative Frequency") +
  theme_minimal()
```



```r
# posterior predictive check for white wine
ggplot() +
  geom_bar(data = data_wine_white, aes(x = quality, y = after_stat(count / sum(count))),
           fill = "blue", alpha = 0.5, width = 0.8, stat = "count") +
  geom_density(data = posterior_df_white, aes(x = posterior_samples, y = after_stat(density)
               color = "red", linewidth = 1) +
  labs(title = "Posterior Predictive Check for White Wine",
       x = "Quality",
       y = "Density / Relative Frequency") +
  theme_minimal()
```



The three plots above demonstrate that the model fits the data exceptionally well, both for the entire dataset and for each wine category individually. The posterior predictive density aligns closely with the histograms of the real data, indicating a high degree of overlap. This provides strong evidence of the model's excellent fit to the data.

1.5 Comparison With Frequentist Mixed-Effect Model

Here I fit a `lmer` model to compare the Bayesian hierarchical regression model with frequentist approach counterpart: mixed effect model. Both approaches are hierarchical in structure, meaning they model data with multiple levels of variation (such as group-level and individual-level effects). However, the Bayesian approach estimates the full posterior distribution of the parameters. We also have the credible interval for parameters from the distribution. While the frequentist mixed-effect model provides point estimates for each parameters, which is estimated by methods like Restricted Maximum Likelihood Estimation (RMLE). Confidence intervals may be used to express uncertainty, but they are not distributions of parameters.

```
# fit the mixed effect model
frequentist_model <- lmer(formula = final_model_formula, data = data_wine_scaled)

# summarize the results
summary(frequentist_model)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula:
quality ~ (1 | wine_type) + alcohol + volatile.acidity + residual.sugar +
    free.sulfur.dioxide + total.sulfur.dioxide + density + pH +
    sulphates + fixed.acidity
   Data: data_wine_scaled

REML criterion at convergence: 11831.5

Scaled residuals:
    Min      1Q  Median      3Q     Max
-5.4452 -0.6150 -0.0449  0.6421  4.2147

Random effects:
 Groups    Name        Variance Std.Dev.
 wine_type (Intercept) 0.04306  0.2075
 Residual              0.53494  0.7314
Number of obs: 5320, groups:  wine_type, 2

Fixed effects:
                     Estimate Std. Error t value
(Intercept)           5.86598    0.14777  39.695
alcohol               0.28087    0.02289  12.273
volatile.acidity     -0.23543    0.01376 -17.115
residual.sugar        0.24980    0.02869   8.706
free.sulfur.dioxide   0.10375    0.01501   6.912
total.sulfur.dioxide -0.08634    0.01990  -4.339
density              -0.29908    0.04497  -6.651
pH                    0.10389    0.01550   6.703
sulphates             0.10666    0.01226   8.700
fixed.acidity         0.11364    0.02177   5.221

Correlation of Fixed Effects:
            (Intr) alcohl vltl.c rsdl.s fr.sl. ttl.s. densty pH     slphts
alcohol     -0.035
volatl.cdty -0.037 -0.112
residul.sgr  0.049 -0.746  0.090
fr.slfr.dxd -0.022  0.066  0.172 -0.137
ttl.slfr.dx  0.052  0.027 -0.145  0.047 -0.589
density     -0.052  0.869 -0.125 -0.909  0.107 -0.132
pH           0.010 -0.546  0.075  0.582 -0.035  0.025 -0.598
sulphates   -0.012 -0.268  0.159  0.270  0.010 -0.086 -0.281  0.119
fixed.acdty  0.019 -0.669  0.183  0.709 -0.002  0.058 -0.780  0.717  0.163
```

All the fixed effects are statistically significant (high absolute t-values), indicating that the predictors contribute meaningfully to explain the wine quality. The signs of the parameters (positive or negative) are consistent with the results from the Bayesian hierarchical regression model. Additionally, the estimated parameter values are quite similar to those from the Bayesian model. Given our sufficiently large dataset (5320 observations in total) and the use of weakly informative priors, it is expected that the results align closely with those of the frequentist approach.

When we have prior information or the sample size is small, Bayesian models are generally preferred as the incorporate prior knowledge to improve the estimation. When the sample size is large, both model is able to obtain good estimation. However, frequentist approach is usually faster in computation, as Bayesian methods involve complex iterative computations, such as Markov Chain Monte Carlo (MCMC) sampling.