Predicting the Count of Sample Wine Cases Purchased from Distributors

Teddy Kelly

Table of contents

Motivation

Wine tasting is a very competitive industry with lots of money involved. Knowing which factors and chemical properties contribute the most to wine being purchased is very valuable information for wine companies to understand. In this report, I will construct 6 different regression models that will each predict how many sample cases of wine are purchased based on their chemical properties, judge's scores of how they taste, and public perception of wine labels. Specifically, I will build two linear regressions, 2 Poisson count regressions, and two negative binomial count regressions and select the best one to make these predictions.

Setup

Load in the total data set for the wine data.

```
df <- read.csv("/Users/teddykelly/Downloads/wine-data-1-1.csv")
```

Below is a table of all of the variables included in the wine dataset with an explanation of what each of them mean. Note that I have renamed the variable `TARGET` to `cases_purchased` and the variable `STARS` to `wine_rating` in the dataset for more intuitive interpretations of them.

Table 1: Variable Names and their Meaning

Variable Names	Meaning
Cases Purchased	Number of cases purchased
Fixed Acidity	Fixed acidity of wine
Volatile Acidity	Volatile acid content of wine
Citric Acid	Citric acid Content
Residuals Sugar	Residual sugar of wine
Chlorides	Chloride content of wine
Free Sulfur Dioxide	Sulfur dioxide content of wine
Density	Density of wine
pH	pH level of wine
Sulphates	Sulphate content of wine
Alcohol	Alcohol content of the wine
Label Appeal	Marketing score that indicates the appeal of the label design for consumers.
Acid Index	Total acidity of wine using a weighted average

Variable Names	Meaning
Wine Rating	Wine rating by a team of experts • 4 stars: Excellent • 1 star: Poor

I will first rename all of the variables to make it easier to deduce what each variable means. These new variable names will match exactly what the names are in the variable name and meaning table above.

```
df <- df |> transmute(
  cases_purchased = TARGET,
  fixed_acidity = FixedAcidity,
  volatile_acidity = VolatileAcidity,
  citric_acid = CitricAcid,
  residual_sugar = ResidualSugar,
  chlorides = Chlorides,
  free_sulfur_dioxide = FreeSulfurDioxide,
  density = Density,
  ph = pH,
  sulphates = Sulphates,
  alcohol = Alcohol,
  label_appeal = LabelAppeal,
  acid_index = AcidIndex,
  wine_rating = STARS
)
```

Below, I have split the wine dataset into 80% of the data for training the model and 20% for testing the model.

```
# Creating the evaluation data set using random seeds
set.seed(123)

# Generate a vector of indices for the training set

train_index <- sample(x = nrow(df), size = round(0.8*nrow(df)))

# Create the training and testing sets

train_df <- df[train_index,]
test_df <- df[-train_index,]
```

Now that we have our training and testing datasets, it's time to do some explanatory data analysis on the training data.

1 Data Exploration

1.1 Summary Statistics

```
labels <- c(
  "Cases Purchased",
  "Fixed Acidity",
  "volatile Acidity",
  "Citric Acid",
  "Residual Sugar",
  "Chlorides",
  "Free Sulfur Dioxide",
  "Density",
  "pH Level",
  "Sulphates",
  "Alcohol Content",
  "Label Appeal",
  "Acid Index",
  "Wine Rating"
)

stargazer(train_df, type = "text", covariate.labels = labels, median = T,
          title = "Initial Summary Statistics")
```

```
Initial Summary Statistics
=========================================================================
Statistic              N     Mean  St. Dev.   Min    Median   Max
-------------------------------------------------------------------------
Cases Purchased     10,236 3.026   1.927      0       3       8
Fixed Acidity       10,236 7.090   6.333   -18.000  6.900  34.400
volatile Acidity    10,236 0.327   0.784    -2.790  0.280   3.680
Citric Acid         10,236 0.309   0.863    -3.240  0.310   3.860
Residual Sugar      9,742  5.275   33.941  -127.800 3.900  141.150
Chlorides           9,711  0.055   0.319    -1.170  0.046   1.351
Free Sulfur Dioxide 9,697  30.899 147.309  -546.000 30.000 622.000
Density             10,236 0.994   0.027    0.889   0.995   1.099
```

```
pH Level            9,915  3.212   0.675    0.480   3.200   6.050
Sulphates           9,276  0.529   0.940    -3.130  0.500   4.240
Alcohol Content     9,717  10.497  3.717    -4.700  10.400 26.100
Label Appeal        10,236 -0.011  0.895      -2      0       2
Acid Index          10,236 7.781   1.339      4       8      17
Wine Rating         7,548  2.045   0.905      1       2       4
-------------------------------------------------------------------
```

Summary Statistics Observations

Missing Values

- There are **several** different variables with **missing values**.
  - **residual sugar**
  - **chlorides**
  - **free sulfur dioxide**
  - **pH level**
  - **sulphates**
  - **alcohol**
  - **wine rating**
- We will address these missing values in the data preparation section either through median imputation or deleting the missing observations.

Dependent Variable

- **Cases Purchased**
  - This is the main response variable that represents the number of sample cases that were purchased by the wine distribution companies.
  - The mean value of 3 represents that on average, the wine distribution companies purchased about 3 sample cases of wine for each commercially available wine in the training dataset.

Predictor Variables

It is unclear as to which of the chemical properties of the wine will be most important in predicting how many cases of the wine will be purchased after testing. For this reason, I have summarized the results from the summary statistics for all of the results.

- **Fixed Acidity**

  - The average fixed acidity is about **7** with a relatively **high standard deviation** compared to the rest of the chemical properties.

- **Volatile Acidity**
  - The mean volatile acidity is roughly 0.327 and the median is about 0.28 which indicates a **slight skewness to the right** considering the small scale.

- **Citric Acid**
  - The mean and median citric acid content are almost identical at about 0.31 with a considerably low spread.

- **Residual Sugar**
  - The mean and median are 5.28 and 3.9 respectively and the **standard deviation** is about 34 which is **very high** compared to the rest of the chemical properties.
  - The sugar content in the wines varies considerably more than for the other chemical properties.

- **Chlorides**
  - The **mean and median** values are **very low** around zero with a **small standard deviation**. The chloride levels generally do not change much between the wines which are tested.

- **Free Sulfur Dioxide**
  - The mean and median values are very close at around 30 which is much higher than the rest and this variable has by far the most dispersed data with a **standard deviation of 147**.

- **Density**
  - The **mean and median** density levels are almost **identical at 0.99** and density appears to be very **clustered together**.

- **pH Level**
  - Mean and median pH levels are the same at about 3.2 which indicates a **symmetric** distribution.

- **Sulphates**
  - The mean is slightly greater than the median, indicating **slight right skewness**.

- **Alcohol Content**

  - The mean and median alcohol levels are practically identical at about 10.4, but the max and min values are fairly far from those average values, indicating the data points are **generally dispersed** around the mean.

- **Label Appeal**
  - Negative Scores suggest customers don't like the label design and positive scores suggest customers do like the label design. The mean and median values close to zero represent that **people equally liked and disliked the wine labels**.

- **Acid Index**
  - The average acid index is around **8** with a **low value of 4** and a **high value of 17**.

- **Wine Rating**
  - Perhaps the most important predictor of the sample wines bought, the **mean and median wine ratings are around** 2. The low and high values of 1 and 4 confirm the lowest and highest possible wine ratings.

1.2 Histograms and Bar Graphs

After building a quantitative understanding of the training dataset using the summary statistics, the next step is to understand the data visually. Our first goal is to visualize the distribution of the data using histograms and bar graphs for continuous and discrete variables respectively.

First, let's visualize the distribution of the continuous variables. I have first created a separate data frame only containing those variables and melted them to plot their histograms all together.

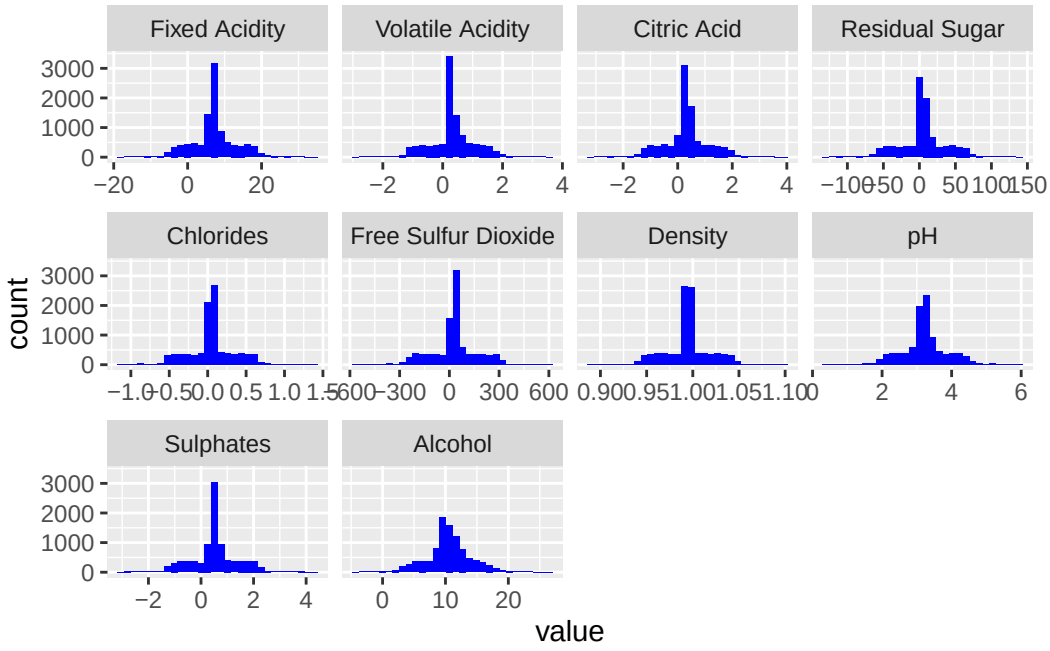
```
train_continuous <- train_df |>
  dplyr::select(fixed_acidity, volatile_acidity, citric_acid,
                residual_sugar, chlorides, free_sulfur_dioxide,
                density, ph, sulphates, alcohol)

continuous_melted <- reshape2::melt(data = train_continuous)

continuous_labels <- c(
  fixed_acidity      = "Fixed Acidity",
  volatile_acidity   = "Volatile Acidity",
  citric_acid        = "Citric Acid",
  residual_sugar     = "Residual Sugar",
```

```
  chlorides           = "Chlorides",
  free_sulfur_dioxide = "Free Sulfur Dioxide",
  density             = "Density",
  ph                  = "pH",
  sulphates           = "Sulphates",
  alcohol             = "Alcohol"
)

ggplot(data = continuous_melted,
       mapping = aes(x = value)) +
  geom_histogram(fill = "blue") +
  facet_wrap(~variable, scales = "free_x",
             labeller = labeller(variable = continuous_labels))
```



Histogram Analysis

- It appears that the distributions of all of the continuous variables are symmetric
- Because of the large number of observations, the distribution appears strange because most of the observations assume the mean/median value of the corresponding variable measure.

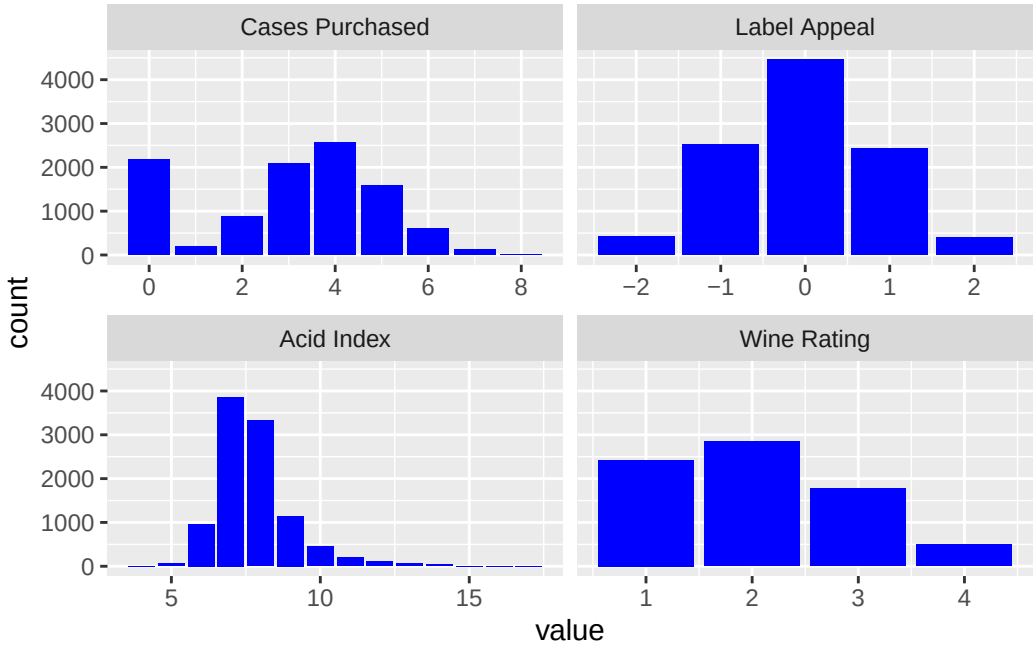
Now, it is time to evaluate the distribution of the discrete numerical variables.

```r
train_discrete <- train_df |> dplyr::select(cases_purchased,
                                            label_appeal,acid_index,
                                            wine_rating)

discrete_melted <- reshape2::melt(data = train_discrete)

discrete_labels <- c(
  cases_purchased = "Cases Purchased",
  label_appeal    = "Label Appeal",
  acid_index      = "Acid Index",
  wine_rating     = "Wine Rating"
)

ggplot(data = discrete_melted,
       mapping = aes(x = value)) +
  geom_bar(fill = "blue") +
  facet_wrap(~variable, scales = "free_x",
             labeller = labeller(variable = discrete_labels))
```



Bar Chart Analysis

- `cases_purchased` - The frequency with the **most amount of sample wine cases** purchased is **4**. However, there were lots of sample wines that did not have any cases

purchased which is why the mean and median sample cases purchased are around 3.

- `label_appeal` - **Most customers appear to be indifferent** to the wine labels with the highest frequency being a marketing score of zero. It also appears that an **equal number of customers** gave **positive label ratings** and **negative label ratings**.
- `acid_idex` - The two highest frequencies of the acid index are 7 and 8 with 7 being the highest.
- `wine_rating` - This is the rating system used by the team of experts with scores ranging from 1 to 4. The rating that receives the most votes for the judges is 2 stars with the majority of the experts rating the wines in the training data as either 1 or 2 stars.

1.3 Correlation Matrix

We will now conduct a preliminary investigation into whether the dependent variable, `cases_purchased` is correlated to any of the independent variables before running any official regressions. We will use a correlation matrix to give a visual representation of how the variables are correlated with each other.

Since there are missing observations for several of the variables, creating a correlation table without addressing those missing values will result in lots of NA values in the table.

Therefore, I have temporarily imputed the NA values with the median values of the corresponding variable names.

```
train_temp <- train_df
```

```
train_temp$residual_sugar[is.na(train_temp$residual_sugar)] <-
  median(train_temp$residual_sugar, na.rm = T)

train_temp$chlorides[is.na(train_temp$chlorides)] <-
  median(train_temp$chlorides, na.rm = T)

train_temp$free_sulfur_dioxide[is.na(train_temp$free_sulfur_dioxide)] <-
  median(train_temp$free_sulfur_dioxide, na.rm = T)

train_temp$ph[is.na(train_temp$ph)] <-
  median(train_temp$ph, na.rm = T)

train_temp$sulphates[is.na(train_temp$sulphates)] <-
  median(train_temp$sulphates, na.rm = T)

train_temp$alcohol[is.na(train_temp$alcohol)] <-
```

```
  median(train_temp$alcohol, na.rm = T)

train_temp$wine_rating[is.na(train_temp$wine_rating)] <-
  median(train_temp$wine_rating, na.rm = T)
```

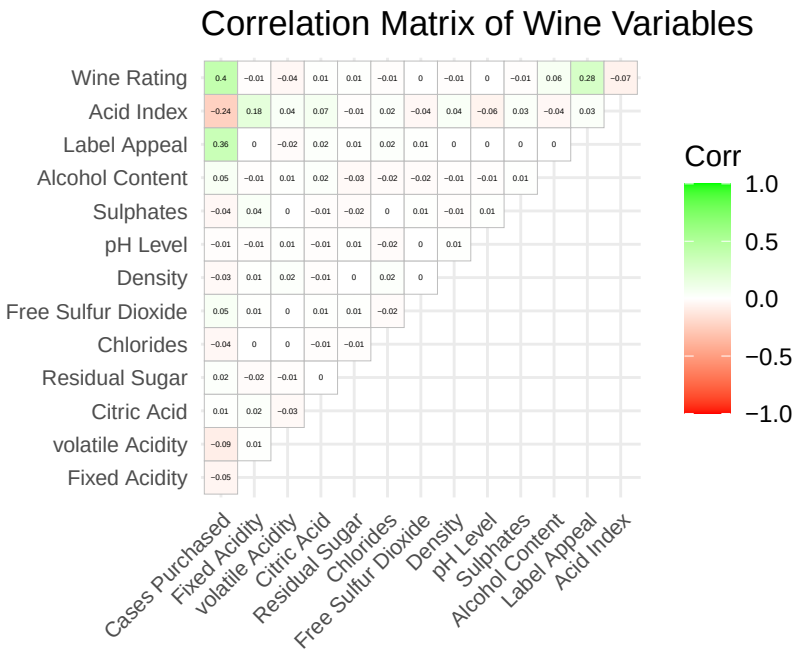
Now with no missing values, we can go ahead and create our correlation matrix to evaluate the initial relationships between the variables.

```
corr_table <- cor(x = train_temp)

colnames(corr_table) <- labels

rownames(corr_table) <- labels

corr_mat <- ggcorrplot(corr = corr_table,
                       method = c("square"),
                       type = c("upper"),
                       title = "Correlation Matrix of Wine Variables",
                       colors = c("red", "white", "green"),
                       lab = TRUE,
                       lab_size = 1,
                       insig = "pch",
                       tl.cex = 8
                       )
corr_mat
```



Correlation Matrix Analysis

There is not much correlation between most of the variables except for a select few. Below I have analyzed the relationships between positively and negatively correlated variables.

Positive Correlation

- **Wine Rating** and **Cases Purchased**
  - Positively correlated with a **correlation of 0.4**. This is the **strongest** positive correlation between any two variables.
  - Indication that **high wine ratings** are associated with **more purchases** of sample cases and low wine ratings with less purchases.
- **Label Appeal** and **Cases Purchased**
  - Positively correlated with a **value of 0.36**. The 2nd strongest correlation with how many cases were sold.
  - Indicates that customers being more receptive to wine labels results in more sample cases being purchased.
- **Acid Index** and **Fixed Acidity**
  - Positively correlated with a **value of 0.1**8.

  - This makes sense because we would expect that wines with a **high wine index** would also have a **high fixed acidity index**.
  - Including both predictors in the final count regression may lead to strong **multicollinearity**.

- **Wine Rating** and **Label Appeal**
  - Positively correlated with a **correlation of 0.28**. This is the highest positive correlation among the independent variables which makes sense because we'd expect that wines with better labels also taste better.
  - There will likely be some **multicollinearity** if both are included in the final count regression, however, since both predictors are **strongly correlated** with the **response variable**, it may be necessary to include both.

Negative Correlation

- **Acid Index** and **Cases Purchased**
  - Negatively correlated with a **correlation of -0.24** which is by far the **strongest negative correlation** between any of the variables.
  - Indication that wines with a **high acid index** are correlated with **less sample cases** of those wines being **purchased**.

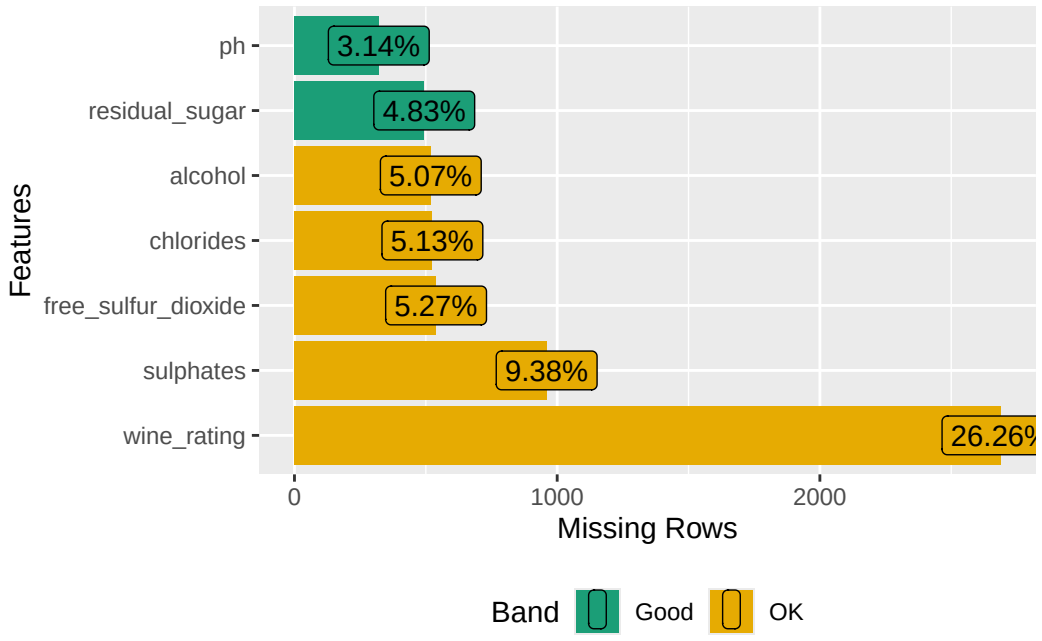
2 Data Preparation

The next step in our data analysis is to clean the training data so it's suitable to build our count regressions which will be used to predict how many sample cases will be purchased. This data cleaning involves addressing any missing values and/or outlier values that may cause bias in our analysis.

2.1 Missing Values

Below, I have visualized all of the missing values in our data.

```
plot_missing(train_df, missing_only = T)
```



About half of the variables contain missing values as we noted previously when analyzing the summary statistics of the training data.

The variables indicated in yellow have missing values in at least 5% of the observations in the training data which may be problematic. What's most concerning is that `wine_rating` has over 26% missing values and it is the most correlated with `cases_purchased` according to the correlation matrix.

Imputing about 26% of the observations with the median wine rating score will likely not be the best course of action because `wine_rating` is a discrete variable with a small range of possible values from 1 to 4. Therefore, imputing the missing values with the median wine rating will result in just over 26% of the wine ratings automatically receiving a rating of 2 stars (the median wine rating) which will certainly skew the data.

For this reason, I will **delete** all observations that do **not have a wine rating**. This will still result in having over 7,000 observations which is plenty of data to analyze. However, because omitting all the observations with missing values would **reduce the training data by 50%**, I will impute the variables that have less missing values as the bias resulting from these imputations will be minimal and none of them had any significant correlation with the response variable `cases_purchased`. This will ensure that no observations that do have a wine rating do not get removed from the data since wine rating appears to be the most important predictor of sample cases purchased.

```
train_df <- train_df[!is.na(train_df$wine_rating),]
```

Now that we have removed all of the missing observations for the wine rating column in the data frame, it's time to impute the remaining missing values for the other variables.

```
train_df$residual_sugar[is.na(train_df$residual_sugar)] <-
  median(train_df$residual_sugar, na.rm = T)

train_df$chlorides[is.na(train_df$chlorides)] <-
  median(train_df$chlorides, na.rm = T)

train_df$free_sulfur_dioxide[is.na(train_df$free_sulfur_dioxide)] <-
  median(train_df$free_sulfur_dioxide, na.rm = T)

train_df$ph[is.na(train_df$ph)] <-
  median(train_df$ph, na.rm = T)

train_df$sulphates[is.na(train_df$sulphates)] <-
  median(train_df$sulphates, na.rm = T)

train_df$alcohol[is.na(train_df$alcohol)] <-
  median(train_df$alcohol, na.rm = T)
```

There are no more missing observations left in the data which will enable us to perform count regressions on the training data without any issues. However, there may still be outliers present in the data which could potentially bias the data.

2.2 Outliers

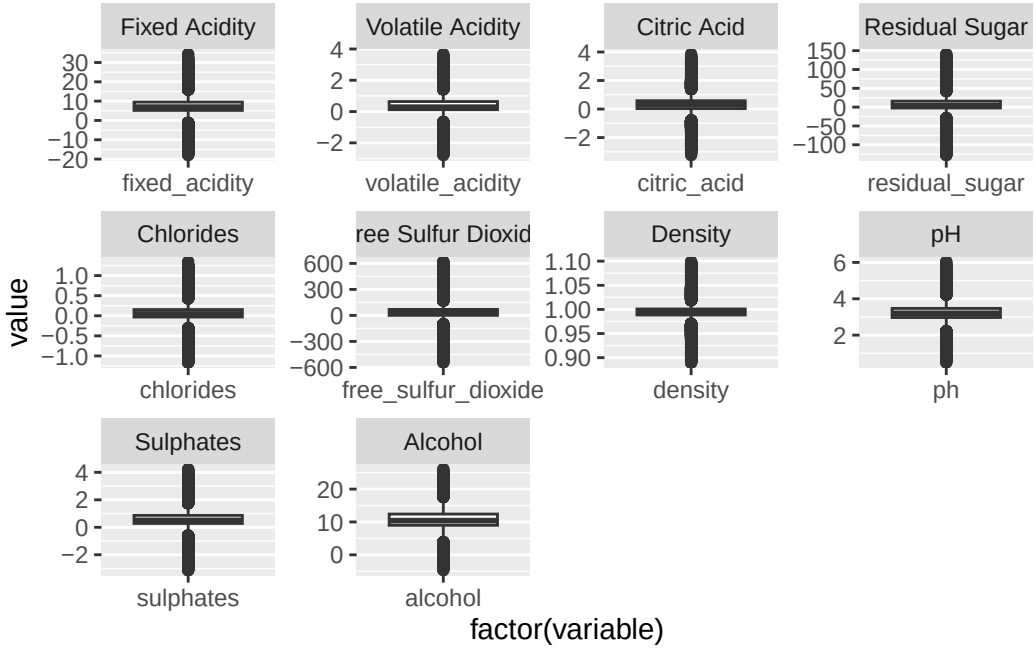
One method I normally use to reduce the influence of outliers is to employ a log transformation on continuous variables. Normally, this will reduce the level of skeweness caused by outliers and result in a distribution that more closely resembles a normal distribution.

However, as we saw with the histograms, none of the continuous variables appear to have any skewness, and more importantly, they take on negative values. This means that applying a log transformation will result in undefined values in the log transformed variables which will cause major problems when running our regressions.

Therefore, I did not perform any variable transformations, as I did not want to disturb the data too much since there are many negative values for the chemical properties of the wines. Also, I did not combine any of the chemical properties together because I do not know enough about

wine properties to accurately combine the correct properties to create meaningful variables for the regressions.

```
# Box plot
ggplot(data = continuous_melted,
       mapping = aes(x = factor(variable), y = value))+
  geom_boxplot()+
  facet_wrap(facets = ~variable, scale = "free",
             labeller = labeller(variable = continuous_labels))
```




```
# Fixed Acidity
# Detect & impute outliers for all numeric variables in train_df

train_df_imputed <- train_df  # copy so you keep the original

numeric_vars <- names(train_df)[sapply(train_df, is.numeric)]

outlier_summary <- data.frame(
  variable = numeric_vars,
  n_outliers = NA_integer_, row.names = labels
)
```

```r
for (v in numeric_vars) {

  x <- train_df[[v]]

  Q1  <- quantile(x, 0.25, na.rm = TRUE)
  Q3  <- quantile(x, 0.75, na.rm = TRUE)
  IQR_val <- Q3 - Q1

  lower_bound <- Q1 - 1.5 * IQR_val
  upper_bound <- Q3 + 1.5 * IQR_val

  # Boolean vector of which values are outliers
  outliers_bool <- x < lower_bound | x > upper_bound

  # Store count in summary table
  outlier_summary[outlier_summary$variable == v, "n_outliers"] <-
    sum(outliers_bool, na.rm = TRUE)

}

# Show outlier counts by variable
outlier_summary$variable <- rownames(outlier_summary)
rownames(outlier_summary) <- NULL
outlier_summary
```

```
              variable n_outliers
1      Cases Purchased          0
2        Fixed Acidity       1536
3     volatile Acidity       1643
4          Citric Acid       1656
5       Residual Sugar       2348
6            Chlorides       2646
7  Free Sulfur Dioxide       2398
8              Density       2330
9              pH Level       1437
10           Sulphates       2198
11     Alcohol Content        650
12        Label Appeal          0
13          Acid Index        487
14         Wine Rating          0
```

Looking at the table above, there are some outliers in all of the continuous numerical variables. However, there are none for any of the discrete numerical variables, `CasesPurchased`, `LabelAppeal`, and `WineRating` which are the most important variables in the dataset. Therefore, I have decided not to delete or impute any of the continuous variables to avoid bias.

2.3 Updated Summary Statistics

Below is an updated version of the summary statistics of the training data after addressing all of the missing values. As we can see, all of the variables in the data have exactly 7,548 observations.

```
stargazer(train_df, type = "text", title = "Update Summary Statistics",
          omit.summary.stat = "N", covariate.labels = labels,
          median = T, notes = "N = 7,548")
```

```
Update Summary Statistics
=============================================================
Statistic             Mean  St. Dev.   Min    Median   Max
-------------------------------------------------------------
Cases Purchased      3.685   1.556      0       4       8
Fixed Acidity        6.923   6.261   -18.000  6.900  32.500
volatile Acidity     0.301   0.781    -2.745  0.270   3.680
Citric Acid          0.309   0.860    -3.160  0.310   3.770
Residual Sugar       5.374   32.838  -127.800 4.500  141.150
Chlorides            0.048   0.308    -1.170  0.045   1.270
Free Sulfur Dioxide 34.064 143.892  -546.000 33.000 622.000
Density              0.994   0.027    0.889   0.994   1.099
pH Level             3.208   0.663    0.480   3.190   6.050
Sulphates            0.508   0.889    -3.130  0.490   4.210
Alcohol Content      10.553  3.644    -4.500  10.500 26.100
Label Appeal         0.044   0.884      -2      0       2
Acid Index           7.647   1.198      5       7      17
Wine Rating          2.045   0.905      1       2       4
-------------------------------------------------------------
N = 7,548
```

3 Build Models

The next step in our data analysis is to build regression models that will accurately make predictions for how many sample cases of wine were purchased using the variables in the

training data.

Since the dependent variable we are attempting to predict, `cases_purchased`, is a discrete variable, we will use count regressions to make these predictions. Specifically, there are two types of count regressions, negative binomial and passion regressions, that we will use. We will construct two regressions for each variation and then compare the results of the regressions to each other. Furthermore, I will also build two multiple linear regression models to compare to the more accurate count regressions.

3.1 Multiple Linear Regressions

We will first construct two multivariate linear regressions that will attempt to predict the number of sample cases of wine purchased for each observation. Note that this is not the ideal model to use since the dependent variable, `CasesPurchased`, is not a continuous variable, but rather a discrete variable. The goal of these regressions is to simply establish a base-line for our regression results which we will then use to compare to the more suitable count regressions.

3.1.1 Multiple Linear Regression 1

The first linear regression I ran contains every variable in the dataset. Below is the estimating equation:

$$\begin{aligned} CasesPurchased_i = \beta_0 + \beta_1 \cdot FixedAcidity_i + \beta_2 \cdot VolatileAcidity + \beta_3 \cdot CitricAcid_i + \\ \beta_4 \cdot ResidualSugar_i + \beta_5 \cdot Chlorides_i + \beta_6 \cdot FreeSulfurDioxide_i + \\ \beta_7 \cdot Density_i + \beta_8 \cdot ph_i + \beta_9 \cdot Sulphates_i + \beta_{10} \cdot Alcohol_i + \\ \beta_{11} \cdot LabelAppeal_i + \beta_{12} \cdot AcidIndex_i + \beta_{13} \cdot WineRating_i + u_i \end{aligned}$$

```
lin_reg1 <- lm(data = train_df,
               formula = cases_purchased ~ .)
```

Variable Selection Process

- I do not have any prior knowledge about which chemical properties most significantly determine the quality of wine, so I thought it would be best to run a preliminary regression containing every variable in the dataset to get an idea of which chemical properties most influence the number of sample cases purchased.
- From our explanatory data analysis, we found that the following three discrete variables have the strongest correlation with `CasesPurchased`:

- `LabelAppeal`: Positive correlation, so there should be a strong positive effect on the number of cases purchased in the regression.
- `AcidIndex`: Negative correlation, so acid index should have a negative effect on cases purchased in the regression.
- `WineRating`: Similar to label appeal, this also has a strong positive correlation and should have a positive effect on the number of sample cases purchased in the regression.

- We will observe the regression results for linear regression 1 in the coefficient interpretation model.
- To obtain the second multivariate linear regression, I will use backward selection using the `stepAIC` command which will select the best model based on the lowest Akaike Information Criterion (AIC).
- We should expect to see `LabelAppeal`, `AcidIndex`, and `WineRating` all remain in the second model since they have the highest correlation with the dependent variable.

Below, I have run the `stepAIC` command on the first linear regression to obtain the second regression using backward variable selection.

```
stepAIC(object = lin_reg1,
        direction = "backward")
```

3.1.2 Multiple Linear Regression 2

Based on the results from performing backward selection, the estimating equation with the lowest AIC is the following:

$$\begin{aligned} CasesPurchased_i = \beta_0 + \beta_1 \cdot VolatileAcidity_i + \beta_2 \cdot chlorides_i + \\ \beta_3 \cdot FreeSulfurDioxide_i + \beta_4 \cdot density_i + \\ \beta_5 \cdot alcohol_i + \beta_6 \cdot LabelAppeal_i + \beta_7 \cdot AcidIndex_i + \\ \beta_8 \cdot WineRating_i + u_i \end{aligned}$$

```
lin_reg2 <- lm(formula = cases_purchased ~ volatile_acidity + chlorides +
    free_sulfur_dioxide + density + alcohol + label_appeal +
    acid_index + wine_rating, data = train_df)
```

- We can see that the second model is shorter than the first and that all three of the main regressors from the first regression have remained. (label appeal, acid index, and wine rating)

- We will compare the regression results of these two linear models in the linear coefficient interpretations section and evaluate the reliability of their models by performing residual analysis, heteroskedasticity, and multicollinearity tests on them both.
- We will now shift our attention to constructing the count regression models.

3.2 Poisson Count Regressions

Now that we have our preliminary multiple linear regressions, we can now begin to build our count regressions to accurately predict the number of sample cases purchased for each observation. We will start by building two Poisson regressions.

For Poisson regressions, the key assumption is that the **conditional mean and variance** of the dependent variable (number of sample cases purchased are the **same**.

3.2.1 Poisson Regression 1

The first Poisson count regression will contain all of the variables in the dataset, and its estimating equation is seen below:

$$\begin{aligned} log(\mu_i) = \beta_0 + \beta_1 \cdot FixedAcidity_i + \beta_2 \cdot VolatileAcidity + \beta_3 \cdot CitricAcid_i + \\ \beta_4 \cdot ResidualSugar_i + \beta_5 \cdot Chlorides_i + \beta_6 \cdot FreeSulfurDioxide_i + \\ \beta_7 \cdot Density_i + \beta_8 \cdot ph_i + \beta_9 \cdot Sulphates_i + \beta_{10} \cdot Alcohol_i + \\ \beta_{11} \cdot LabelAppeal_i + \beta_{12} \cdot AcidIndex_i + \beta_{13} \cdot WineRating_i \end{aligned}$$

where μ_i is the mean count of the number of sample cases purchased.

```
poisson_reg1 <- glm(data = train_df,
                    formula = cases_purchased ~.,
                    family = poisson(link = "log"))
```

Variable Selection Process

- Similar to the process used for the linear models, the first Poisson regression incorporates all of the chemical property variables.
- Despite the fact that the same exact variables are used here as in the first linear model, this regression model will yield **different results** because of the inherently different structure.

- Notice the $log(\mu_i)$ in the estimating equation. This log transformation ensures that the mean count of the number of cases purchased is always positive. This then influences how we interpret this model's coefficient estimates because Poisson regressions follow a log-linear structure.
- We will also incorporate backward selection to select the best possible Poisson model based on the AIC value. The second Poisson model will likely use different variables than were used in the second linear model because the first linear and Poisson models are structurally different even though they use the same variables.

StepAIC

```
stepAIC(object = poisson_reg1,
        direction = "backward")
```

3.2.2 Poisson Regression 2

$$\begin{aligned} log(\mu_i) = \beta_0 + \beta_1 \cdot VolatileAcidity + \beta_2 \cdot Chlorides_i + \beta_3 \cdot FreeSulfurDioxide_i + \\ \beta_4 \cdot Alcohol_i + \beta_5 \cdot LabelAppeal_i + \beta_6 \cdot AcidIndex_i + \\ \beta_7 \cdot WineRating_i \end{aligned}$$

```
poisson_reg2 <- glm(formula = cases_purchased ~ volatile_acidity + chlorides +
    free_sulfur_dioxide + alcohol + label_appeal + acid_index +
    wine_rating, family = poisson(link = "log"), data = train_df)
```

- The second Poisson model contains 7 of the original 13 variables from the first model.
- Notice that label appeal, acid index, and wine rating are all still included with the second model even after backward selection. Just as with the linear model, these three variables should have the greatest effect on the count of sample cases purchased.
- We will compare the results of these Poisson models with another type of count model, the negative binomial count model.

3.3 Negative Binomial Count Regressions

The next step is to build two negative binomial regressions which will handle any over-dispersion (variance is greater than the mean) from the data and compare the results with the Poisson regressions.

3.3.1 Negative Binomial Regression 1

Regression 1

The first negative binomial regression will include all of the variables that are included in the wine dataset. The **estimating equation is exactly as seen with the Poisson model 1**, expect the negative binomial model takes into account for overdispersion where the variance of the count of sample wines purchased may be greater than the mean count:

$$Var(CasesPurchased_i) = \mu_i + \alpha \cdot \mu_i^2$$

where α is a constant. If $\alpha > 0$, then the variance will be greater than the mean and there is overdispersion. However, if $\alpha = 0$, then the $\alpha \cdot \mu_i^2$ term will cancel out and the variance will be precisely the mean, resulting in the original Poisson model.

```
nb_reg1 <- glm.nb(data = train_df,
                  formula = cases_purchased ~.)
```

```
Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
control$trace > : iteration limit reached
Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
control$trace > : iteration limit reached
```

Variable Selection Process

- The variables included in the first negative binomial model are precisely the same as the variables for the first linear and Poisson models.
- If there is **no overdispersion** in the data (the conditional mean and variance of the number of sample cases purchased is the same), then the results for the negative binomial model will be exactly the same as the Poisson model.
- If the negative binomial model yields different results than the Poisson regression, then the variance is greater than the mean in the data, and using the Poisson regression is not ideal since it assumes equal mean and variance of the dependent variable.
- I have also used backward selection to select the variables for the second negative binomial model. Note that if the conditional mean is equal to the variance of the dependent variable, then the second negative binomial model will contain exactly the same variables as the second Poisson model.

StepAIC

```
stepAIC(object = nb_reg1,
        direction = "backward")
```

3.3.2 Negative Binomial Regression 2

After applying backward selection on the preliminary negative binomial regression, I have run a second negative binomial regression.

StepAIC actually outputs the same exact remaining variables as used for the second Poisson model. Therefore, the estimating equation will be precisely the same as the second Poisson model with the addition of the variance condition to account for possible overdispersion:

$$Var(CasesPurchased_i) = \mu_i + \alpha \cdot \mu_i^2$$

As specified for the first negative binomial model, the variance of the number of sample cases purchased will become equal to the mean if the constant $\alpha = 0$.

```
nb_reg2 <- glm.nb(formula = cases_purchased ~ volatile_acidity + chlorides +
    free_sulfur_dioxide + alcohol + label_appeal + acid_index +
    wine_rating, data = train_df, init.theta = 141259.0943, link = log)
```

- As we can see, the model above contains precisely the same variables as the second Poisson model, indicating that there is no overdispersion in the data and that using a Poisson model in this scenario would be appropriate.
- We will investigate this more thoroughly in the count regression interpretations section and analyze if the coefficient results are indeed the same.
- I will create a third negative binomial model only containing the three most important regressors so we have at least one negative binomial model that yields different coefficient estimates than the Poisson models.

3.3.3 Negative Binomial Regression 3

As we will see in section 3.5, the coefficient estimates for the second Poisson and negative binomial regressions are identical, indicating that the mean and the variance are precisely the same. Therefore, I have also included a third negative binomial regression using different input variables to obtain different results.

In our explanatory data analysis, we determined that `LabelAppeal`, `WineRating`, and `AcidIndex` have the strongest correlation with how many sample cases of wine were purchased. Therefore, I have decided to remove any irrelevant variables that may reduce the precision of the coefficient estimates of the key variables.

Therefore, the estimating equation for the third estimating equation is the following:

$$log(\mu_i) = \beta_0 + \beta_1 \cdot LabelAppeal_i + \beta_2 \cdot AcidIndex_i + \beta_3 \cdot WineRating_i$$

where

$$Var(CasesPurchased_i) = \mu_i + \alpha \cdot \mu_i^2$$

Since the first two estimating equations yield the same exact results as the two Poisson models, then for this equation, it's likely that α will also equal zero, meaning no overdispersion, and thus a Poisson model of the same form would yield the same results produced from this estimating equation.

```
nb_reg3 <- glm.nb(formula = cases_purchased ~ label_appeal + acid_index +
                    wine_rating,
                  data = train_df)
```

3.4 Linear Regressions Coefficient Interpretations

```
labels1 <- c(
  "Fixed Acidity",
  "Volatile Acidity",
  "Citric Acid",
  "Residual Sugar",
  "Chlorides",
  "Free Sulfur Dioxide",
  "Density",
  "pH Level",
  "Sulphates",
  "Alcohol Content",
  "Label Appeal",
  "Acid Index",
  "Wine Rating"
)
stargazer(lin_reg1, lin_reg2, type = "text",
          title = "Linear Regression Regression Comparison",
          covariate.labels = labels1)
```

Linear Regression Regression Comparison

	Dependent variable:	
	cases_purchased	
	(1)	(2)
Fixed Acidity	0.0004	
	(0.002)	
Volatile Acidity	-0.088***	-0.089***
	(0.017)	(0.017)
Citric Acid	0.021	
	(0.015)	
Residual Sugar	-0.0001	
	(0.0004)	
Chlorides	-0.131***	-0.131***
	(0.043)	(0.043)
Free Sulfur Dioxide	0.0003***	0.0003***
	(0.0001)	(0.0001)
Density	-0.742	-0.746
	(0.496)	(0.496)
pH Level	-0.015	
	(0.020)	
Sulphates	-0.009	
	(0.015)	
Alcohol Content	0.016***	0.016***
	(0.004)	(0.004)
Label Appeal	0.657***	0.657***
	(0.016)	(0.016)
Acid Index	-0.162***	-0.161***
	(0.011)	(0.011)

```
Wine Rating                   0.726***                  0.726***
                              (0.016)                    (0.016)

Constant                      4.047***                  3.990***
                              (0.506)                    (0.501)

-------------------------------------------------------------------------
Observations                   7,548                      7,548
R2                             0.455                      0.455
Adjusted R2                    0.454                      0.454
Residual Std. Error     1.149 (df = 7534)          1.149 (df = 7539)
F Statistic         484.329*** (df = 13; 7534) 786.874*** (df = 8; 7539)
=========================================================================
Note:                                          *p<0.1; **p<0.05; ***p<0.01
```

Coefficient Interpretations

I will use the second linear model derived from performing backward selection to discuss the sign, economic magnitude, and statistical significance of the important variables.

Label Appeal

- Sign
  - **Positive (+)**
  - This is the **expected coefficient sign** because it makes sense that wines that receive a more positive reaction to their labels will be more likely to be purchased.
- Economic magnitude
  - The coefficient is **0.657**.
  - Since this is linear regression, this means that for a one unit increase in a wine's label appeal, the number of sample cases purchased will increase by about 0.657 cases, holding all else constant.
  - This is a fairly significant result in terms of magnitude because an increase in label appeal leads to almost a whole other sample case of wine purchased.
- Statistical Significance
  - **Statistically significant** below the $\alpha = 0.01$ level, implying that we are confident that this estimate is not a result of random chance,

Wine Rating

- Sign
  - **positive (+)**
  - This is the **expected coefficient sign** because wine that is more highly rated by experts will be more likely to have sample cases purchased.
- Economic Magnitude
  - The coefficient is **0.726**.
  - For a one unit increase in wine rating (1 additional star), the number of sample cases will increase on average by about 0.726 cases, holding all else constant.
  - This result is even more economically significant than the result for label appeal.
- Statistical Significance
  - **Statistically significant** below the $\alpha = 0.01$ level
  - We are confident that this is the true estimate and that it likely did not occur by random chance.

Acid Index

- Sign
  - **Negative (-)**
  - Suggests that wine with a low acid index tends to be purchased more.
- Economic Magnitude
  - The coefficient estimate is **-0.161**
  - For every additional unit increase in a wine's acid index, on average, the number of sample cases purchased by distributors decreases by about 0.161 cases, holding all else constant.
  - This is not as economically significant of a result as for label appeal and wine rating which makes sense because its negative correlation with sample cases purchased was not as strong in magnitude as label appeal and wine rating.
- Statistical Significance
  - **Statistically significant** at the $\alpha = 0.01$ level
  - This coefficient estimate is likely to not have come from random chance.

3.5 Poisson and Negative Binomial Count Regression Interpretations

This section will compare the coefficient estimates of the second models for the Poisson and negative binomial count regressions. These models both have lower AIC levels than the first regressions, so I have not included the full models in this section to make the coefficient analysis more straightforward.

```
labels2 <- c(
  "Volatile Acidity",
  "Chlorides",
  "Free Sulfur Dioxide",
  "Alcohol Content",
  "Label Appeal",
  "Acid Index",
  "Wine Rating"
)
stargazer(poisson_reg2, nb_reg2, nb_reg3, type = "text",
          title = "Count Regression Comparison",
          covariate.labels = labels2)
```

```
Count Regression Comparison
================================================================================
                                        Dependent variable:
                    ---------------------------------------------------------
                                          cases_purchased
                      Poisson                      negative
                                                   binomial
                        (1)                (2)                    (3)
--------------------------------------------------------------------------------
Volatile Acidity     -0.024***          -0.024***
                      (0.008)            (0.008)

Chlorides             -0.036*            -0.036*
                      (0.019)            (0.019)

Free Sulfur Dioxide   0.0001*            0.0001*
                     (0.00004)          (0.00004)

Alcohol Content       0.004**            0.004**
                      (0.002)            (0.002)
```

Label Appeal	0.180***	0.180***	0.180***
	(0.007)	(0.007)	(0.007)
Acid Index	-0.047***	-0.047***	-0.048***
	(0.005)	(0.005)	(0.005)
Wine Rating	0.186***	0.186***	0.188***
	(0.007)	(0.007)	(0.007)
Constant	1.199***	1.199***	1.243***
	(0.048)	(0.048)	(0.044)
Observations	7,548	7,548	7,548
Log Likelihood	-13,573.050	-13,574.110	-13,585.180
theta		141,259.400 (218,247.900)	140,639.200 (217,227.300)
Akaike Inf. Crit.	27,162.100	27,164.220	27,178.360
Note:			*p<0.1; **p<0.05; ***p<0.01

As we can see, the coefficient estimates for the Poisson and negative binomial count regressions are identical and the log likelihood and AIC values are almost the same as well. This indicates that there is no sign of overdispersion (the variance is greater than the mean), and therefore the Poisson count regression is sufficient to use in this case.

As noted in section 3.3.3, I included another negative binomial regression which only includes the three most correlated predictors of the number of sample cases purchased. The coefficient estimates of the included predictors did not change nearly at all, however, the AIC value did increase which suggests that it is not as well-balanced of a model as the Poisson or negative binomial regressions also in the regression table.

I will still provide a multiclass confusion matrix for the shortened negative binomial model and evaluate its prediction accuracy in comparison to the Poisson model 2.

Coefficient Interpretations (Poisson Regression)

Since there is no evidence of overdispersion, I will interpret the coefficient for the Poisson count regression. Also, I will create a new model for the negative binomial distribution to obtain a different model.

Label Appeal

- Sign
  - **Positive (+)**

  - Similar to the linear regression, this is the **expected coefficient sign** because if consumers are more receptive to the label design of a wine bottle, we can expect that more sample cases will be purchased by a wine distribution company.

- Economic magnitude
  - The coefficient estimate is **0.18.**
  - This is a **count regression**, so this means that an additional unit increase in a wine's label appeal score leads to an **increase in the expected count** by **factor** of $e^{0.18}$ or about **1.19**.
  - It's more difficult to determine if this is an economically significant result than it was for the linear regression because of the coefficient interpretation.

- Statistical Significance
  - **Statistically significant** under the $\alpha = 0.01$ significance level
  - We are confident that this estimate likely did not occur by random chance alone.

Wine Rating

- Sign
  - **Positive(+)**
  - This is the **expected coefficient sign** because it makes sense that the number of sample cases purchased will increase if its wine rating increases.

- Economic Magnitude
  - The coefficient estimate is **0.186**.
  - Every additional star a wine receives results in the expected count of sample wines purchased to increase by a factor of $e^{0.186} = 1.204$ holding all else constant. This means that the expected count of sample wines purchased to increase by about 20.4%.
  - The economic magnitude is greater than for label appeal which makes sense if we refer back to our correlation matrix in section 1 where win rating had the highest positive correlation with cases purchased.

- Statistical Significance
  - **Statistically significant** under the $\alpha = 0.01$
  - We are confident that the coefficient estimate did not occur by random chance alone.

Acid Index

- Sign
  - **Negative (-)**
- Economic Magnitude
  - The coefficient estimate is **-0.047**.
  - A one unit increase in the acid index of wine results in the expected count of sample cases purchased by wine distribution companies to decrease by a factor of about 0.95 holding all else constant.
  - Alternatively, this means that the expected count of sample cases purchased resulting from a one unit increase in the acid index decreases by about 4.59%.
  - This doesn't seem like a lot, but if the acid index of a particular wine increases by a lot, then we can deduce that the number of sample cases purchased will decrease by a fairly significant amount.
- Statistical Significance
  - **Statistically significant** at the $\alpha = 0.01$ significance level.
  - The coefficient result likely did not occur due to random chance and we can be confident with the estimate.

4 Select Models

In this section, we will be evaluating the model performance of the linear and count regressions we have. For the linear regressions, I will use standard residual analysis and will compare each of their F-statistics, MSE, and R-squared values.

For the count regressions, I will make classification predictions on the training data and build a multiclass confusion matrix to test the accuracy of both the Poisson and negative binomial regressions. As we saw in the previous section, the AIC value for the Poisson regression was lower than the negative binomial regression, indicating a more sound model, however I will still check their prediction accuracy before making a final selection.

4.1 Linear Regression Diagnostic Tests

```r
diagnostic_table <- data.frame(
  model = c("Model 1", "Model 2"),
  MSE = c(mean(lin_reg1$residuals**2), mean(lin_reg2$residuals**2)),
  R_squared = c((summary(lin_reg1))$adj.r.squared,
                (summary(lin_reg2))$adj.r.squared),
  f_stat = c((summary(lin_reg1))$fstatistic[1],
             (summary(lin_reg2))$fstatistic[1])
)

diagnostic_table |> kable(digits = 3,
                          caption = "Comparison of Diagnostic Results",
                          col.names = c("Model", "MSE",
                                        "R Squared", "F-Statistic"))
```

Table 2: Comparison of Diagnostic Results

Model	MSE	R Squared	F-Statistic
Model 1	1.318	0.454	484.329
Model 2	1.319	0.454	786.874

Mean Squared Error

- The mean squared error values are **almost identical** with the full linear model actually having a slightly lower MSE value. This indicates that the predictions for the full model may be slightly closer to the actual values, although not by very much.

R Squared

- The adjusted r-squared values are **exactly the same** in both of the linear models.
- For both models, approximately 45.5% of the variation in the number of sample wines purchased can be explained by the models which is a fairly strong r-squared value.

F-statistic

- The f-statistic is much **higher** for the **second linear model** which suggests that model 2 is generally more statistically significant than model 1.
- Both f-static values are relatively high and statistically significant which means that there is likely a relationship between the predictors and the number of sample wine cases purchased.

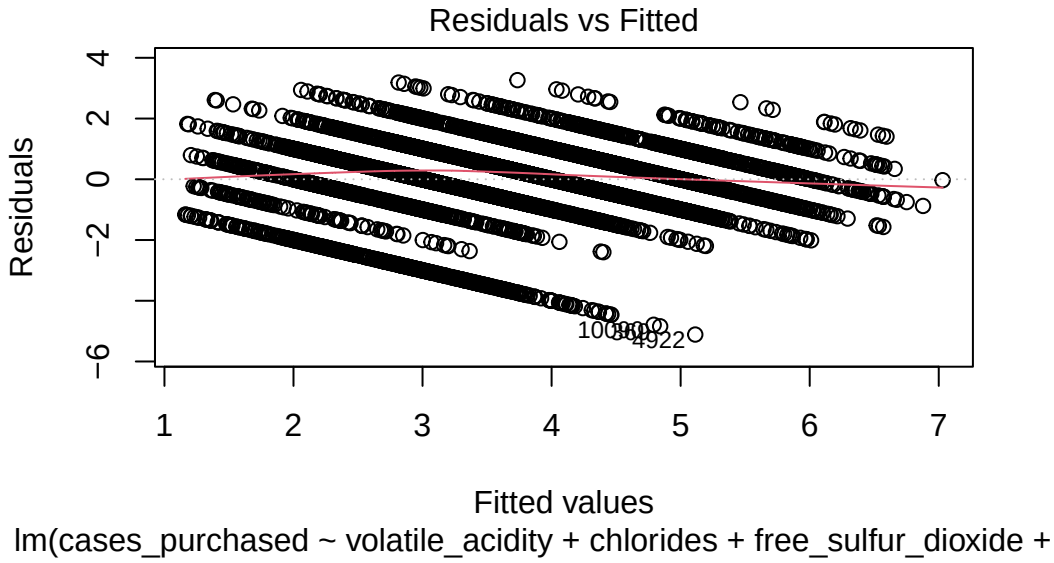
Diagnostic Testing Conclusion

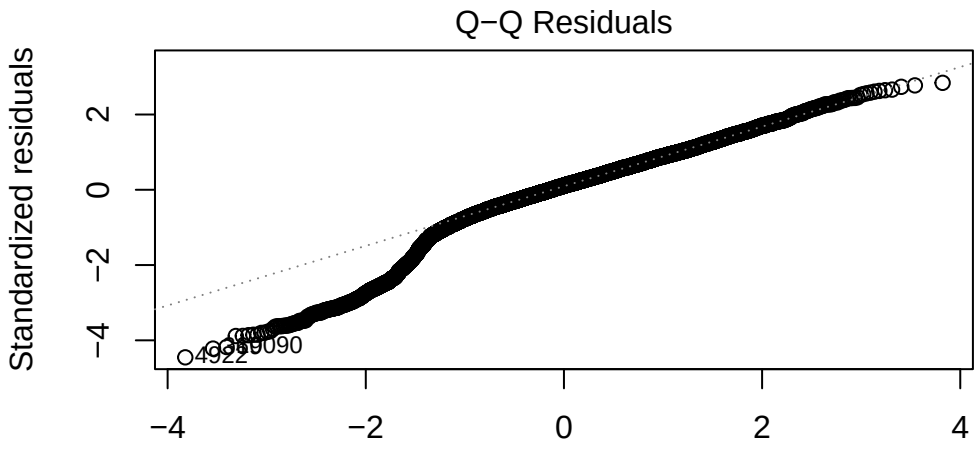
- Based on the large disparity in the f-statistic value between the models, **model 2** appears to be the strongest and most reliable linear regression model. I will have to further investigate how these models uphold the Gauss markov assumptions through residual analysis to make a clear decision of which linear model is the best.
- Also, I will still have to evaluate the prediction accuracy of the count regression models to determine the best overall model for prediction on the evaluation dataset.

4.1.1 Residual Analysis Plots

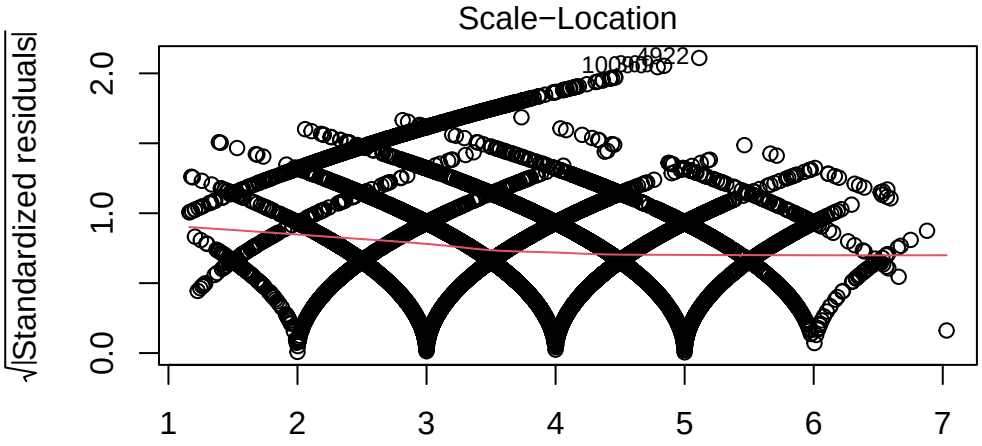
Below I have plotted the four residual plots for the second linear model. The residual plots for model 1 are identical so I only included model 2 here for simplicity.

```
plot(lin_reg2)
```

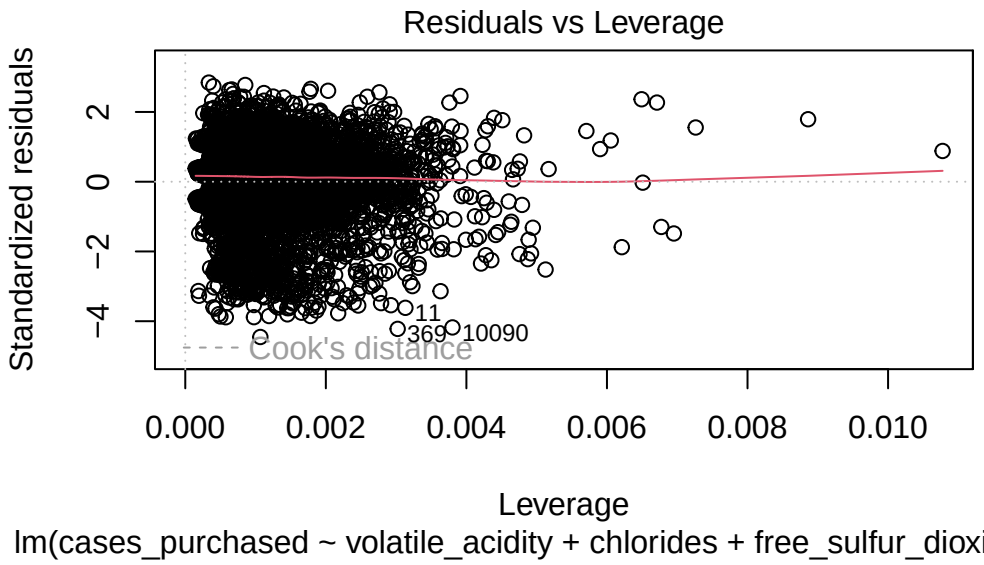




lm(cases_purchased ~ volatile_acidity + chlorides + free_sulfur_dioxide +



lm(cases_purchased ~ volatile_acidity + chlorides + free_sulfur_dioxide +



Residuals VS Fitted Values

- Clearly, we can see that the residual values are not randomly distributed around zero and that the variance of the residuals follow a clear downward sloping trend as the fitted values increase. Clear sign of heteroskedasticity.

Normal Q-Q Plot

- The residuals follow a relatively normal distribution for positive theoretical quantiles, however the residuals near the lower tail shift off of the normal distribution fairly significantly.

Scale Location Plot

- The residuals are clearly not spread equally among the predicted values which is a clear violation of homoskedasticity.
- I have also run a White's Test below to confirm the presence of heteroskedasticity in both linear models.

Residuals vs Leverage Plot

- There is no indication of any leverage points in the data.

White's Test

Below is a comparison of the White's test results for both linear model 1 and 2.

```r
library(skedastic)

wt1 <- white(mainlm = lin_reg1)
wt2 <- white(mainlm = lin_reg2)

white_tests <- rbind(wt1, wt2)
white_tests <- add_column(white_tests, Model = c("Model 1", "Model 2"),
                          .before = 1)

kable(white_tests)
```

Model	statistic	p.value	parameter	method	alternative
Model 1	894.4523	0	26	White's Test	greater
Model 2	882.3044	0	16	White's Test	greater

The null hypothesis for a White's Test is that there is homoskedasticity in a regression model. Since the **p-values** for both model 1 and 2 are **exactly zero**, then we can reject the null hypothesis which confirms what we found from the residuals plots that there is **strong evidence** to suggest **heteroskedasticity**.

Since there is such strong heteroskedasticity present, the statistically significant results that we found for the linear models may be unreliable since not having constant variance among residuals affects the standard errors. Therefore, these models may not be the best for making reliable predictions on the evaluation data, however we will also analyze the homoskedasticity assumption for the count regressions.

Multicollinearity Testing with Variance inflation Factor

```r
vif1 <- data.frame(Variable = c("Volatile Acidity", "Chlorides",
                     "Free Sulfur Dioxide", "Density", "Alcohol",
                     "Label Appeal", "Acid Index", "Wine Rating"),
                   VIF = vif(lin_reg2),
                   row.names = NULL)

kable(vif1)
```

Variable	VIF
Volatile Acidity	1.003559
Chlorides	1.001043
Free Sulfur Dioxide	1.001287

Variable	VIF
Density	1.002089
Alcohol	1.008457
Label Appeal	1.129241
Acid Index	1.014251
Wine Rating	1.141363

Below displays that none of the variables in the linear model 2 have high VIF values, indicating that there is no multicollinearity present in the regression model.

4.1.2 Predictions with Linear Model

Below, I have used the **second linear model** that I created to generate predictions for the number of sample cases purchased in the training dataset. I will compare these results with the predictions of the count models in the next section.

```
train_df$lin_initial <- predict(lin_reg2)

train_df$lin_pred <- round(train_df$lin_initial)
```

Since I have **rounded** the linear model predictions to the **nearest whole number**, I will also create a **multiclass confusion matrix** to evaluate its prediction accuracy and compare this with the count models.

Since the minimum number of sample cases predicted to be purchased is 1 and the maximum amount is only 7 for the linear model, I will have to specify the levels 0 through 8 for the linear model so the confusion matrix will work properly.

```
library(caret)
com_levels <- 0:8
confusionMatrix(factor(train_df$cases_purchased, levels = com_levels),
                factor(train_df$lin_pred, levels = com_levels),
                positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5    6    7    8
         0    0   20  175  258  100    3    0    0    0
         1    0   19   65   20    0    0    0    0    0
```

```
         2    0   15  318  228   40    0    0    0    0
         3    0   11  349  841  481   32    0    0    0
         4    0    2   77  690 1229  331   18    0    0
         5    0    0   15  159  688  528  121    3    0
         6    0    0    0   13  137  268  163    9    0
         7    0    0    0    0    9   37   54    9    0
         8    0    0    0    0    0    1    9    3    0

Overall Statistics

               Accuracy : 0.4116
                 95% CI : (0.4005, 0.4228)
    No Information Rate : 0.3556
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2424

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity                NA 0.283582  0.31832   0.3807   0.4579  0.44000
Specificity           0.92634 0.988638  0.95679   0.8365   0.7701  0.84468
Pos Pred Value             NA 0.182692  0.52912   0.4907   0.5236  0.34875
Neg Pred Value             NA 0.993552  0.90197   0.7655   0.7202  0.88863
Prevalence            0.00000 0.008877  0.13235   0.2927   0.3556  0.15898
Detection Rate        0.00000 0.002517  0.04213   0.1114   0.1628  0.06995
Detection Prevalence  0.07366 0.013778  0.07962   0.2271   0.3109  0.20058
Balanced Accuracy          NA 0.636110  0.63755   0.6086   0.6140  0.64234
                     Class: 6 Class: 7 Class: 8
Sensitivity           0.44658 0.375000       NA
Specificity           0.94055 0.986709 0.998278
Pos Pred Value        0.27627 0.082569       NA
Neg Pred Value        0.97097 0.997984       NA
Prevalence            0.04836 0.003180 0.000000
Detection Rate        0.02160 0.001192 0.000000
Detection Prevalence  0.07817 0.014441 0.001722
Balanced Accuracy     0.69356 0.680855       NA
```

Linear Model Confusion Matrix Analysis

IMPORTANT NOTE:

- The confusion matrix that R creates when using the `confusionMatrix()` command under the `caret` package lays out the matrix so that Reference refers to the rows and Prediction refers to the columns.
- However, I noticed that, for whatever reason, the confusion matrix metrics are calculated as if Reference refers to the columns and Prediction refers to the rows.
- Therefore, the values for **sensitivity** and **positive predicted** value are **swapped**, and the values for **specificity** and **negative predicted** value are also **swapped**.
- Note that this does not affect the total accuracy of the models, so **accuracy** and **classification error rate** will be the same as they are seen for all of the models.
- For this first linear model, I will use logic to deduce the sensitivity, specificity, and precision for the class with the highest percentage, but then for the count models, I will just swap the corresponding values to obtain the correct percentages for each metric to reduce clutter.

Accuracy

- The accuracy of the linear model is **41.16%.**
- It accurately predicts the correct number of sample cases purchased for only 41.16% of the wines.
- This does not seem like a very good prediction accuracy, but considering that it is a linear regression model and not intrinsically a classification model, this is not too bad.

Classification Error Rate

- 1-41.18 = **58.82%**.
- The linear model inaccurately predicts the number of sample cases purchased for 58.84% of the wines.

Sensitivity (True Positive Rate)

- As noted earlier, the sensitivity, specificity, and positive predicted values as seen above are not accurate. In particular, sensitivity and positive predicted value are swapped. So, the class with the highest true positive rate should be **class 4** with a value of **52.36%** since that is the highest "positive predicted value" as depicted from the confusion matrix metrics.
- I will solve for the true positive rate for class 4 manually to confirm this.

```
# number of observations where cases purchased equals 4
length(which(train_df$cases_purchased==4))
```

```
[1] 2347
```

- We can see that there are 2,347 observations in the training dataset where the number wines with 4 sample cases purchased is exactly 4. If we add up the number of times Reference is 4 in the confusion matrix above, we also get 2,347.

```
2 + 77 + 690 + 1229 + 331 + 18
```

```
[1] 2347
```

- Also, according to the confusion matrix, the linear model accurately predicted 4 sample cases of wine to be purchased 1,229 times. Therefore, calculating the true positive rate, we get $1229/2347 = 0.5236$ as seen below:

```
1229/2347
```

```
[1] 0.5236472
```

- This confirms the sensitivity value we found above from swapping the positive predicted value for sensitivity. Therefore, the linear correctly classified **52.36%** of the wine that actually had 4 sample cases purchased.

Specificity (True negative Rate)

- Using similar logic as for sensitivity, interpreting the negative predicted values for class 4 as the specificity values, the class with the lowest specificity I **Class 4** with a true negative rate of **72.02%**.
- I will solve for the true negative rate manually to confirm this. The total number of observations that did not have 4 sample cases purchased is 5,201 as seen below:

```
# Number of observations where cases purchased does not equal 4
length(which(train_df$cases_purchased!=4))
```

```
[1] 5201
```

- Therefore, the number of true negatives that the linear model predicted for class 4 will be 5,201 minus the number of times the model incorrectly predicted 4 sample cases purchased. According to the confusion matrix, the model inaccurately predicted 4 cases purchased 1,455 times as seen below.

```
100+40+481+688+137+9
```

```
[1] 1455
```

- Subtracting 1,455 from 5,201 gives us 3,746 which represents the true negatives. Then the true negative rate is the following $3746/5201 = 0.7202$ as seen below:

```
3746/5201
```

```
[1] 0.7202461
```

- This confirms the specificity value we found for class 4 earlier from interpreting the negative predicted value as the specificity.
- The true negative rate of 72.02% for class 4 means that for the wines where there were not 4 sample cases purchased, the Poisson model only classifies 72% of those observations in another class other than 4 cases purchased.

Precision (Positive Predicted Value)

- The class with the highest positive predicted value should be **class 4** with a value of **45.79%** since that is the highest "sensitivity" as depicted from the confusion matrix metrics.
- I will solve for the positive predicted value to confirm this finding.

```
length(which(train_df$lin_pred == 4))
```

```
[1] 2684
```

- We can see that there are 2,684 observations in the training data where the linear model predicted the number of sample cases of wine purchased to be 4. Also, adding up the number of times where Prediction is 4 in the confusion matrix we also get 2,684:

```
 100+40+481+1229+688+137+9
```

```
[1] 2684
```

- Finally, since the confusion matrix shows that the linear correctly predicted 4 sample cases of wine being purchased 1,229 times, calculating the positive predicted value, we get $1229/2684 = 0.4579$ as seen below:

```
1229/2684
```

```
[1] 0.4578987
```

- This confirms the same precision value we found earlier from swapping the sensitivity with the precision.
- This means that out of the wines that the linear model predicted to have 4 sample cases purchased, almost **46**% of those classifications were accurate, meaning 46% of those wines actually did have 4 sample cases purchased.

4.2 Count Regression Multiclass Confusion Matrix

Now, I will evaluate the accuracy and reliability of the count regressions.

The first step is to make classification predictions for how many sample cases were purchased by wine distribution companies for each observation. Here, I have used the second Poisson count regression to make predictions on the number of sample wine cases bought.

```
train_df$pois_log <- predict(poisson_reg2, type = "response")

# Round these predictions to the nearest whole number
train_df$pois_pred <- round(train_df$pois_log)
```

The Poisson predicted counts range from 2 through 9, meaning that the model does not predict any of the observations to have only 1 or zero sample cases purchased, and that it actually predicts some observations to have 9 cases purchased. This does not match the levels of 0 through 8 for the actual cases purchased, so I have to set common factor levels to get the `ConfusionMatrix` command to work.

4.2.1 Poisson Regression Multiclass Confusion Matrix

Below is the multiclass confusion matrix for the second Poisson count regression.

```
common_levels <- 0:9

confusionMatrix(factor(train_df$cases_purchased, levels = common_levels),
                factor(train_df$pois_pred, levels = common_levels),
                positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5    6    7    8    9
         0    0    0  163  311   79    3    0    0    0    0
         1    0    0   79   25    0    0    0    0    0    0
         2    0    0  274  301   26    0    0    0    0    0
         3    0    0  289 1045  362   18    0    0    0    0
         4    0    0   55  921 1094  250   27    0    0    0
         5    0    0   10  230  674  434  147   16    3    0
         6    0    0    0   24  141  230  147   39    9    0
         7    0    0    0    0    9   29   48   18    4    1
         8    0    0    0    0    0    0    3    7    3    0
         9    0    0    0    0    0    0    0    0    0    0

Overall Statistics

               Accuracy : 0.3994
                 95% CI : (0.3884, 0.4106)
    No Information Rate : 0.3785
    P-Value [Acc > NIR] : 9.731e-05

                  Kappa : 0.2271

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity                NA       NA  0.31494   0.3658   0.4587   0.4502
Specificity           0.92634  0.98622  0.95103   0.8574   0.7573   0.8360
Pos Pred Value             NA       NA  0.45591   0.6097   0.4661   0.2867
Neg Pred Value             NA       NA  0.91421   0.6894   0.7518   0.9122
Prevalence            0.00000  0.00000  0.11526   0.3785   0.3160   0.1277
Detection Rate        0.00000  0.00000  0.03630   0.1384   0.1449   0.0575
Detection Prevalence  0.07366  0.01378  0.07962   0.2271   0.3109   0.2006
Balanced Accuracy          NA       NA  0.63299   0.6116   0.6080   0.6431
                     Class: 6 Class: 7  Class: 8  Class: 9
Sensitivity           0.39516 0.225000 0.1578947 0.0000000
Specificity           0.93827 0.987815 0.9986718 1.0000000
Pos Pred Value        0.24915 0.165138 0.2307692       NaN
Neg Pred Value        0.96766 0.991666 0.9978766 0.9998675
Prevalence            0.04928 0.010599 0.0025172 0.0001325
```

```
Detection Rate         0.01948 0.002385 0.0003975 0.0000000
Detection Prevalence   0.07817 0.014441 0.0017223 0.0000000
Balanced Accuracy      0.66671 0.606407 0.5782833 0.5000000
```

4.2.2 Negative Binomial Multiclass Confusion Matrix (Shortened Model)

I have followed the same procedure as highlighted above with the Poisson regression prediction, but this time using the third negative binomial regression.

```
train_df$nb_log <- predict(nb_reg3, type = "response")

train_df$nb_pred <- round(train_df$nb_log)

confusionMatrix(factor(train_df$cases_purchased, levels = common_levels),
                factor(train_df$nb_pred, levels = common_levels),
                positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5    6    7    8    9
         0    0    0  171  299   83    3    0    0    0    0
         1    0    0   82   22    0    0    0    0    0    0
         2    0    0  316  261   24    0    0    0    0    0
         3    0    0  328 1000  368   18    0    0    0    0
         4    0    0   79  905 1078  260   25    0    0    0
         5    0    0   16  227  675  453  122   18    3    0
         6    0    0    0   24  143  230  147   29   17    0
         7    0    0    0    0    7   34   46   14    8    0
         8    0    0    0    0    0    0    3    7    3    0
         9    0    0    0    0    0    0    0    0    0    0

Overall Statistics

               Accuracy : 0.3989
                 95% CI : (0.3878, 0.4101)
    No Information Rate : 0.3627
    P-Value [Acc > NIR] : 4.53e-11

                  Kappa : 0.2284
```

```
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity                NA       NA  0.31855   0.3652   0.4533  0.45391
Specificity           0.92634  0.98622  0.95653   0.8516   0.7545  0.83802
Pos Pred Value             NA       NA  0.52579   0.5834   0.4593  0.29921
Neg Pred Value             NA       NA  0.90269   0.7021   0.7500  0.90968
Prevalence            0.00000  0.00000  0.13143   0.3627   0.3151  0.13222
Detection Rate        0.00000  0.00000  0.04187   0.1325   0.1428  0.06002
Detection Prevalence  0.07366  0.01378  0.07962   0.2271   0.3109  0.20058
Balanced Accuracy          NA       NA  0.63754   0.6084   0.6039  0.64596
                     Class: 6 Class: 7  Class: 8 Class: 9
Sensitivity           0.42857 0.205882 0.0967742       NA
Specificity           0.93851 0.987299 0.9986697        1
Pos Pred Value        0.24915 0.128440 0.2307692       NA
Neg Pred Value        0.97183 0.992741 0.9962840       NA
Prevalence            0.04544 0.009009 0.0041070        0
Detection Rate        0.01948 0.001855 0.0003975        0
Detection Prevalence  0.07817 0.014441 0.0017223        0
Balanced Accuracy     0.68354 0.596591 0.5477219       NA
```

4.2.3 Confusion Matrix Analysis

- As discussed in the confusion matrix analysis for the linear model, the values seen for the positive predicted values are to be interpreted as the sensitivities and the values seen as the negative predicted values are to be interpreted as the specificity values.
- To save space, I will not outline the calculations necessary to obtain these values, but will instead just swap the corresponding metrics.

Accuracy

- Poisson (**39.94%)**
- Negative Binomial (39.89%)
- Both of these count regression models have low prediction accuracy, but the Poisson Model has a slightly higher prediction accuracy by about 0.05 percentage points.

Classification Error Rate

- Poisson: (1-39.94) = **60.06%**

- Negative Binomial: (1-39.49) = 60.11%
- Both classification error rates are high, but the Poisson regression has a slightly lower classification error rate.

Sensitivity (True Positive Rate)

- Poisson
  - Highest detection of true positives is for observations where 3 sample cases were purchased (**60.97%**)
- Negative Binomial
  - Highest detection of true positives if for observations where also 5 sample cases were purchased (58.34%)
- Both models do a poor job of detecting true positives and neither model was able to accurately predict observations where only 1 or no sample cases were purchased.

Specificity (True Negative Rate)

- Poisson
  - The lowest accuracy for detecting true negatives is for class 3 with an accuracy of 68.94%
  - Therefore, for observations where there were not 3 sample cases purchased, the Poisson model only classifies 69% of those observations in another class other than 3 cases purchased.
- Negative Binomial
  - The lowest accuracy for detecting true negatives is also for class 3 with a true positive rate of 70.21%.
  - A similar interpretation as seen for the Poisson model follows here.
- Both models were very successful in detecting true negatives, that is, for the observations that are not in a certain class for the number of sample cases purchased, the models do a good job of predicting that the number of cases purchased is not in the specific class.

Precision (Positive Predicted Value)

- Poisson
  - Does a decent job for classes 4 and 5 with a maximum precision of 45.87% for class 4.

  - Out of the observations that the Poisson model predicted to have 4 sample cases purchased, about 46% of those wines actually did have 3 sample cases purchased.

- Negative Binomial
  - Similar interpretation as for the Poisson model but the highest positive predicted value is for class 5 at 45.39%.
  - Out of the observations in which the negative binomial model predicted to have 5 sample cases purchased, about 45.39% of those wines actually did have 3 sample cases purchased.

- On average, the sensitivity for the two models is generally better than their precision.
- Note that both of these models did generate at least one instance of predicting 9 sample cases to be purchased even though the maximum number of sample cases purchased in the training data is only 8.

4.2.4 Count Regression Heteroskedasticity White's Test

Below, I ran a White's test for the second poisson and third negative binomial model, and stored their results in a table.

```
pois_wt <- white(mainlm = poisson_reg2)
nb_wt <- white(mainlm = nb_reg3)

count_white_tests <- rbind(pois_wt, nb_wt)
count_white_tests <- add_column(count_white_tests,
                                Model = c("Poisson Model 2",
                                          "Negative Binomial Model 3"),
                                .before = 1)
kable(count_white_tests)
```

Model	statistic	p.value	parameter	method	alternative
Poisson Model 2	1135.047	0	14	White's Test	greater
Negative Binomial Model 3	1105.249	0	6	White's Test	greater

Similarly to the linear regression models, the table above shows that both the Poisson and negative binomial count models have a p-value of zero, meaning we can reject the null hypothesis of homoskedasticty. Therefore, there is a strong indication of heteroskedasticty which indicates unreliable conclusions of the statistical significance of our estimates, potentially leading to misleading predictions on the testing data.

4.2.5 Selecting the Best Model

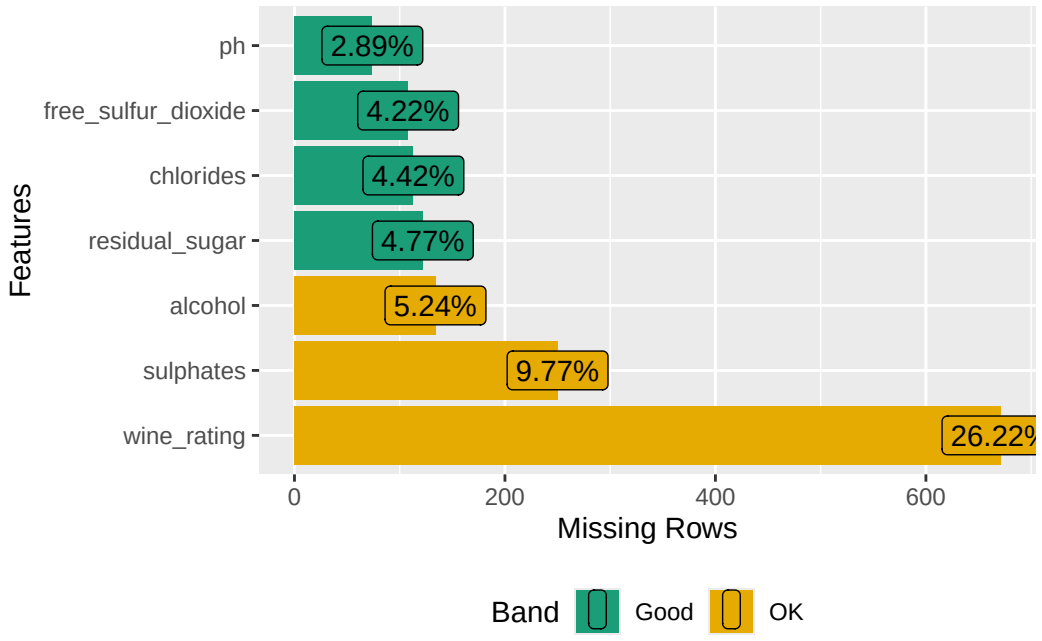
After implementing diagnostic testing for the linear and count models, I have decided to select the **second linear model** to make predictions on the testing data. This may seem counterintuitive since the dependent variable is the number of sample cases purchased which is a discrete variable. However, after rounding the predictions of the linear model to the nearest whole number and building a multiclass confusion matrix, it ended up having the highest prediction accuracy at **41.18%**, as well as generally having higher sensitivity, specificity, and positive predicated values across the classes than the count models. Also, even though the linear model showed clear signs of heteroskedasticity in the residual plots, the count models were shown to also have heteroskedasticity in the White's Test. Furthermore, the linear model had a very high F-statistic value which suggests that the entire model is significantly significant in predicting the number of sample wine cases purchased.

4.3 Predictions on Testing Data

Before making predictions on the testing wine data, I must address the NA values that are present so it is compatible to receive predictions from the linear model.

Since I deleted any observation that had a missing value for the wine rating, I will also delete any observation with a missing wine rating for the testing data. Furthermore, since there were a large number of other variables that had missing values, I imputed those values with their corresponding median values to prevent too much loss of data. To be consistent, I will also impute any missing values not in the `WineRating` column with their corresponding median value.

```
plot_missing(test_df, missing_only = T)
```




```
# Delete missing wine rating values
test_df <- test_df[!is.na(test_df$wine_rating),]

# Imputing other missing values with median values of corresponding variables
test_df$residual_sugar[is.na(test_df$residual_sugar)] <-
  median(test_df$residual_sugar, na.rm = T)

test_df$chlorides[is.na(test_df$chlorides)] <-
  median(test_df$chlorides, na.rm = T)

test_df$free_sulfur_dioxide[is.na(test_df$free_sulfur_dioxide)] <-
  median(test_df$free_sulfur_dioxide, na.rm = T)

test_df$ph[is.na(test_df$ph)] <-
  median(test_df$ph, na.rm = T)

test_df$sulphates[is.na(test_df$sulphates)] <-
  median(test_df$sulphates, na.rm = T)

test_df$alcohol[is.na(test_df$alcohol)] <-
  median(test_df$alcohol, na.rm = T)
```

Now that the data is cleaned and all of the missing values have been addressed, we can finally

make predictions using the linear regression model on the testing data.

I first made normal predictions using the linear regression model and then I rounded the predictions to the nearest whole number. I will use these rounded values to construct the multiclass confusion matrix for the evaluation data.

```
# initial predictions using the linear model
test_df$lin_initial <- predict(lin_reg2, newdata = test_df)

# Rounding the predictions to the nearst whole number
test_df$lin_pred <- round(test_df$lin_initial)

min(test_df$lin_pred)
```

```
[1] 1
```

```
max(test_df$lin_pred)
```

```
[1] 7
```

The minimum and maximum number of sample cases predicted by the linear model are again 1 and 7 respectively which means we must identify the common levels that must be used for the confusion matrix to be accurate.

```
confusionMatrix(factor(test_df$cases_purchased, levels = com_levels,
                       ordered = TRUE),
                factor(test_df$lin_pred, levels = com_levels, ordered = TRUE))
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3   4   5   6   7   8
         0   0   4  41  78  16   1   0   0   0
         1   0   1  11   2   0   0   0   0   0
         2   0   3  84  54  13   1   0   0   0
         3   0   0  85 205 140  10   0   0   0
         4   0   0  20 180 299  66   5   0   0
         5   0   0   2  39 187 130  41   0   0
         6   0   0   0   3  44  52  42   2   0
         7   0   0   0   0   0   7  15   3   0
         8   0   0   0   0   0   0   2   0   0
```

```
Overall Statistics

               Accuracy : 0.4047
                 95% CI : (0.3824, 0.4272)
    No Information Rate : 0.3702
    P-Value [Acc > NIR] : 0.001113

                  Kappa : 0.2311

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0  Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity                NA 0.1250000  0.34568   0.3654   0.4278  0.48689
Specificity           0.92585 0.9930851  0.95684   0.8229   0.7721  0.83405
Pos Pred Value             NA 0.0714286  0.54194   0.4659   0.5246  0.32581
Neg Pred Value             NA 0.9962647  0.90825   0.7541   0.6965  0.90799
Prevalence            0.00000 0.0042373  0.12871   0.2971   0.3702  0.14142
Detection Rate        0.00000 0.0005297  0.04449   0.1086   0.1584  0.06886
Detection Prevalence  0.07415 0.0074153  0.08210   0.2331   0.3019  0.21133
Balanced Accuracy          NA 0.5590426  0.65126   0.5942   0.5999  0.66047
                     Class: 6 Class: 7 Class: 8
Sensitivity           0.40000 0.600000       NA
Specificity           0.94335 0.988317 0.998941
Pos Pred Value        0.29371 0.120000       NA
Neg Pred Value        0.96390 0.998926       NA
Prevalence            0.05561 0.002648 0.000000
Detection Rate        0.02225 0.001589 0.000000
Detection Prevalence  0.07574 0.013242 0.001059
Balanced Accuracy     0.67168 0.794158       NA
```

Confusion Matrix Interpretations

- The confusion matrix for the testing data has the same problems as I mentioned with the linear model on the training data.
- Therefore, I will again interpret the values that are **displayed** as the **positive predicted values** as the **true positive rates and vice versa**, and also the values seen as the **negative predicted values** as the **true negative rates**.

Accuracy

- The accuracy of the linear model is **40.47%.**
- It accurately predicts the correct number of sample cases purchased for only 40.47% of the wines.
- This classification accuracy is slightly less than the linear model achieved for the training data, but it is still higher than the prediction accuracy of any of the count regression models.

Classification Error Rate

- 1-41.18 = **59.53%**.
- The linear model inaccurately predicts the number of sample cases purchased for 59.53% of the wines.

Sensitivity (True Positive Rate)

- Interpreting the positive predicted values as the true positive rates, the class with the highest sensitivity is **class 2** with a true positive rate of **54.19%**.
- This means that out of the wines who actually had 2 sample cases purchased, the linear model correctly detected 54%% of those wines having 2 sample cases purchased.

Specificity (True negative Rate)

- The specificity values are all very high across all of the classes but the lowest specificity value occurs in **class 4** with a true negative rate of about **69.65%**.
- This means that out of the wines that did not have 4 sample cases purchased, the linear model correctly predicted almost 70% of those wines as not having 4 sample cases purchased.

Precision (Positive Predicted Value)

- The class in which the linear model had the highest precision is **class 7** with a positive predicted value of **60%**.
- This means that out of the wines that the linear model predicted to have 7 sample cases purchased, exactly 60% of those classifications were accurate, meaning 60% of those wines actually did have 7 sample cases purchased.

4.4 Conclusion

- In this project, we looked at a dataset that contained about 12,000 different commercially available wines along with various chemical properties of each of the wines and how they were rated in wine sampling.
- I trained 6 different regression models on the training data.
  - 2 multivariate linear regression models
  - 2 Poisson Count Regressions
  - 2 Negative Binomial Count Regressions
- After comparing the linear and count models using confusion matrix metrics, I determined that **linear model 2** was the best model for prediction on the evaluation data set. This model contained 8 of the 13 original variables and the following are 7 of those variables that were found to be statistically significantly associated with determining the number of sample cases of wine purchased after sampling them at a significance level of $\alpha = 0.01$:
  - Volatile Acidity ($\beta = -0.089$)
  - Chlorides ($\beta = -0.131$)
  - Free Sulfur Dioxide ($\beta = 0.0003$)
  - Alcohol Content ($\beta = 0.016$)
  - Label Appeal ($\beta = 0.657$)
  - Acid Index ($\beta = -0.161$)
  - Wine Rating ($\beta = 0.726$)
- Specifically, our explanatory data analysis showed label appeal and wine rating having the two strongest positive correlations and acid index having the strongest negative correlation with the number of sample cases purchased. These findings are corroborated with the signs and economic magnitude of their respective coefficient estimates from the linear regression as seen above.
- This linear model had higher prediction accuracies than both of the count models and specifically, it had a prediction accuracy of about **40.47%**. This means that it correctly classified the number of sample cases of wine purchased for 40.47% of the wines in the testing data.
- The linear model used for prediction in the testing data did have strong signs of heteroskedasticty which may have negatively affected the prediction accuracy of the model.