Statistical Learning Final Report

Alberto Calabrese, Eleonora Mesaglio, Greta d'Amore Grelli

2024-06-21

Contents

1 Introduction

In this project, we will conduct a thorough analysis of a dataset of our choice, to gain a full understanding of our data, and we will then build models that enable us to make accurate predictions. The dataset we chose is called *Starbucks Beverage Components* and contains information about the ingredients of Starbucks' beverages.

We will go through several steps, including Data Cleaning, Exploratory Data Analysis (EDA), Regression and Classification Analysis. Indeed, first we will prepare our data for analysis by handling missing values and ensuring that our data is correctly formatted. Once our data is clean, we will proceed to the EDA stage, where we will avail ourself of visual and quantitative methods to understand the structure of our data and the relationships between variables. Finally, we will perform Regression Analysis to understand the relationship between our dependent and independent variables. This will allow us to make predictions about our data and understand the factors that influence our dependent variable, `"Calories"`. Moreover we will implement a Classification Analysis in order to assign various macronutrients to each specific `"Beverage_category"`.

2 Data

The dataset we will analyze in this project is *Starbucks Beverage Components* from Kaggle, that you can find at the following link: https://www.kaggle.com/datasets/henryshan/starbucks.

This data provides a comprehensive guide to the nutritional content of the beverages available on the Starbucks menu. We have a total of 242 samples described by 18 variables. These attributes include the name of the beverage, its categorization and preparation method, the total caloric content and the constituents of the beverage, expressed in grams and milligrams.

```r
data <- read.csv("Data/starbucks.csv", header = TRUE, sep = ",")
```

2.1 Data Transformation

Note that several variables in our dataset, namely `"Vitamin.A....DV."`, `"Vitamin.C....DV."`, `"Calcium....DV."` and `"Iron....DV."`, are represented as percentages. Consequently, the percentage symbol is included in our data. However, when conducting statistical analysis using R, the presence of non-numeric characters such as the percentage symbol can cause complications, interfering with the processing and analysis of the data. Therefore, we proceed to remove it.

Similarly, as R primarily operates on numeric and categorical data, we also convert all the other numerical variables into numeric format.

These preprocessing steps ensure a smooth and efficient analysis, making it easier to explore, visualize, and understand our data.

```r
# Remove percentage sign from the data
data$Vitamin.C....DV. <- as.numeric(gsub("%", "", data$Vitamin.C....DV.))
# Set the other variables as numeric
data$Calories <- as.numeric(data$Calories)
```

2.2 Data Cleaning

Another challenge we have to face is the presence of missing data. Indeed, in `"Caffeine..mg."` column there are 23 `NA` values. This is a common issue in data analysis and needs to be addressed appropriately to ensure the validity of our statistical results.

One way to deal with these unwanted `NA` values is to omit the samples containing them from our study. This guarantees that our analysis is conducted solely on complete and dependable data. Alternatively, we can fill them in with the average or the median of the observed values for that specific attribute. This second method helps to preserve the overall data distribution while addressing the missing data points.

In our work, we opt for the latter approach, replacing `NA` values with the median. This choice is particularly suitable for our data, which is skewed and contains outliers. Indeed, the median, being a measure of central tendency that is not affected by extreme values, provides a more robust replacement in the presence of outliers.

```r
summary(data$Caffeine..mg.)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##    0.00   50.00   75.00   89.52  142.50  410.00      23
```

```r
# Replace NA values with the median
data_cleaned <- data
data_cleaned$Caffeine..mg.[is.na(data_cleaned$Caffeine..mg.)] <- median(
  data_cleaned$Caffeine..mg., na.rm = TRUE)
# Summary of the Caffeine column after cleaning
summary(data_cleaned$Caffeine..mg.)
```

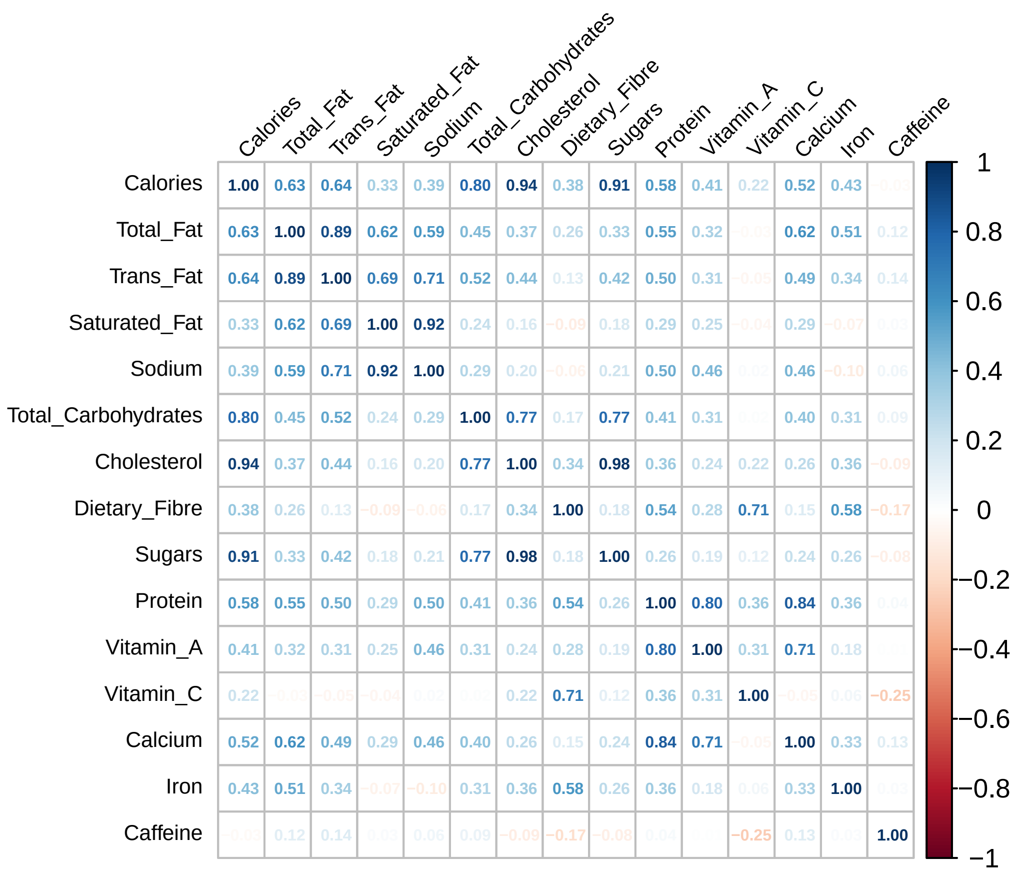
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    0.00   70.00   75.00   88.14  130.00  410.00
```

Lastly, taking in consideration our cleaned data, we renamed the columns by removing dots and units of measure, in order to obtain a more readable dataset.

3 Correlation Analysis

After completing these preliminary preprocessing steps, we calculate the correlation matrix for our dataset. This computation helps us in comprehending the interrelationships among the dataset's variables. In the correlation matrix, a value near to 1 at the ij position indicates a strong positive correlation between the i-th and j-th variables. Conversely, a value close to -1 signifies a strong negative correlation. A value near 0 suggests that the two variables do not significantly influence each other.

Observe that the first three columns of our data are categorical features, thus for these we cannot compute Pearson's correlation coefficient. In the following code lines we remove them to compute and plot such matrix.



	Calories	Total_Fat	Trans_Fat	Saturated_Fat	Sodium	Total_Carbohydrates	Cholesterol	Dietary_Fibre	Sugars	Protein	Vitamin_A	Vitamin_C	Calcium	Iron	Caffeine
Calories	1.00	0.63	0.64	0.33	0.39	0.80	0.94	0.38	0.91	0.58	0.41	0.22	0.52	0.43	−0.03
Total_Fat	0.63	1.00	0.89	0.62	0.59	0.45	0.37	0.26	0.33	0.55	0.32	−0.03	0.62	0.51	0.12
Trans_Fat	0.64	0.89	1.00	0.69	0.71	0.52	0.44	0.13	0.42	0.50	0.31	−0.05	0.49	0.34	0.14
Saturated_Fat	0.33	0.62	0.69	1.00	0.92	0.24	0.16	−0.09	0.18	0.29	0.25	−0.04	0.29	−0.07	0.03
Sodium	0.39	0.59	0.71	0.92	1.00	0.29	0.20	−0.06	0.21	0.50	0.46	0.02	0.46	−0.10	0.06
Total_Carbohydrates	0.80	0.45	0.52	0.24	0.29	1.00	0.77	0.17	0.77	0.41	0.31	0.02	0.40	0.31	0.09
Cholesterol	0.94	0.37	0.44	0.16	0.20	0.77	1.00	0.34	0.98	0.36	0.24	0.22	0.26	0.36	−0.09
Dietary_Fibre	0.38	0.26	0.13	−0.09	−0.06	0.17	0.34	1.00	0.18	0.54	0.28	0.71	0.15	0.58	−0.17
Sugars	0.91	0.33	0.42	0.18	0.21	0.77	0.98	0.18	1.00	0.26	0.19	0.12	0.24	0.26	−0.08
Protein	0.58	0.55	0.50	0.29	0.50	0.41	0.36	0.54	0.26	1.00	0.80	0.36	0.84	0.36	0.04
Vitamin_A	0.41	0.32	0.31	0.25	0.46	0.31	0.24	0.28	0.19	0.80	1.00	0.31	0.71	0.18	[illegible]
Vitamin_C	0.22	−0.03	−0.05	−0.04	0.02	0.02	0.22	0.71	0.12	0.36	0.31	1.00	−0.05	0.06	−0.25
Calcium	0.52	0.62	0.49	0.29	0.46	0.40	0.26	0.15	0.24	0.84	0.71	−0.05	1.00	0.33	0.13
Iron	0.43	0.51	0.34	−0.07	−0.10	0.31	0.36	0.58	0.26	0.36	0.18	0.06	0.33	1.00	0.03
Caffeine	−0.03	0.12	0.14	0.03	0.06	0.09	−0.09	−0.17	−0.08	0.04	[illegible]	−0.25	0.13	0.03	1.00

For example, we can see that `"Calories"` and `"Sugars"` are strongly positively related. In particular at each one-unit increase of calories is associated, in average, a 0.91 increase of cholesterol levels.

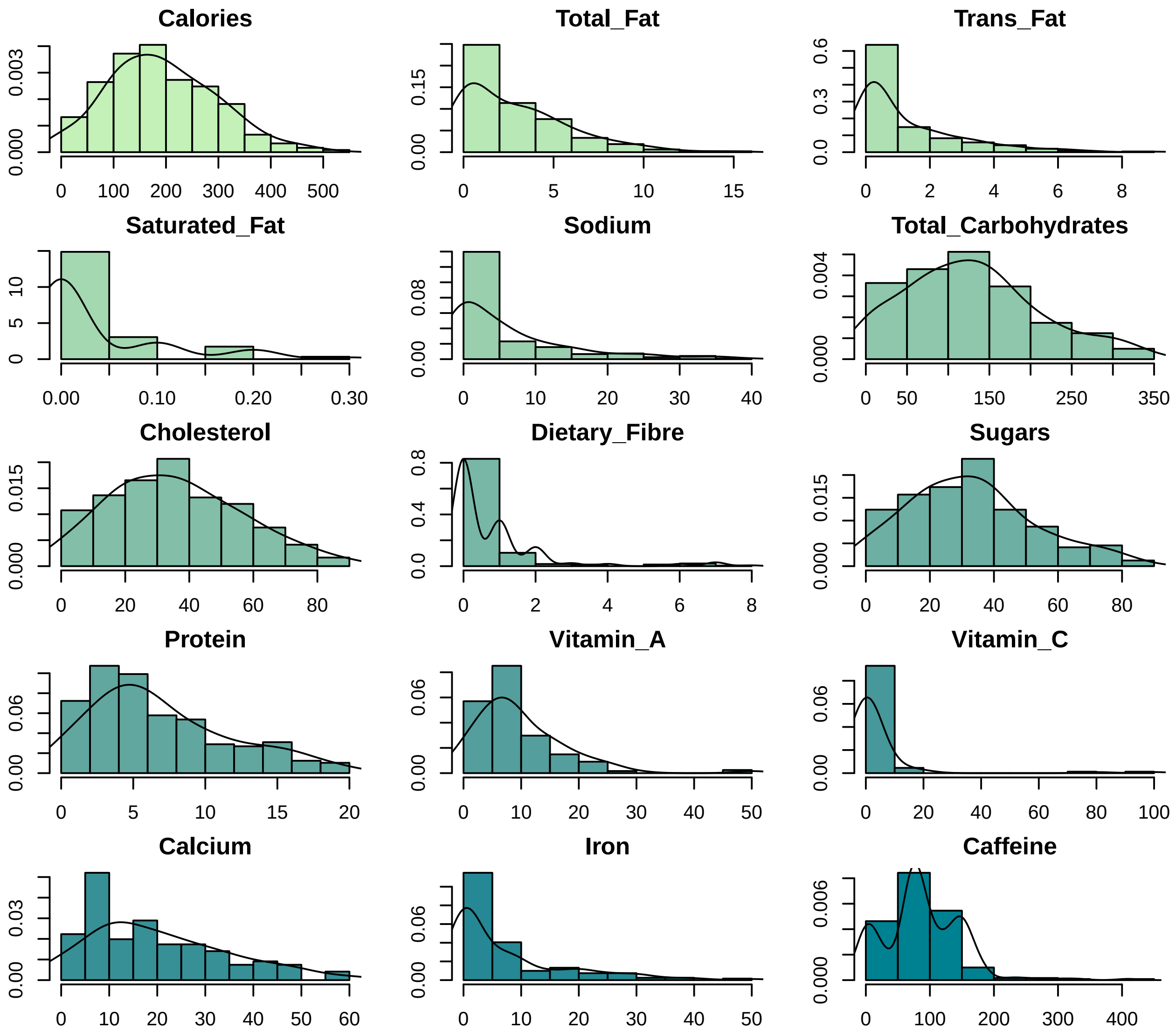
4 Data Visualization

Data visualization is a powerful tool that allows us to uncover patterns, correlations and outliers in our data. It provides visual information on the dataset in our analysis, representing large amounts of data in a clear and comprehensive way and underlining the relationships among them. This enables us to recognize patterns quickly.

So, let us transform our raw data into graphical representations, to gain a more comprehensive understanding of the information at hand.

4.1 Histograms

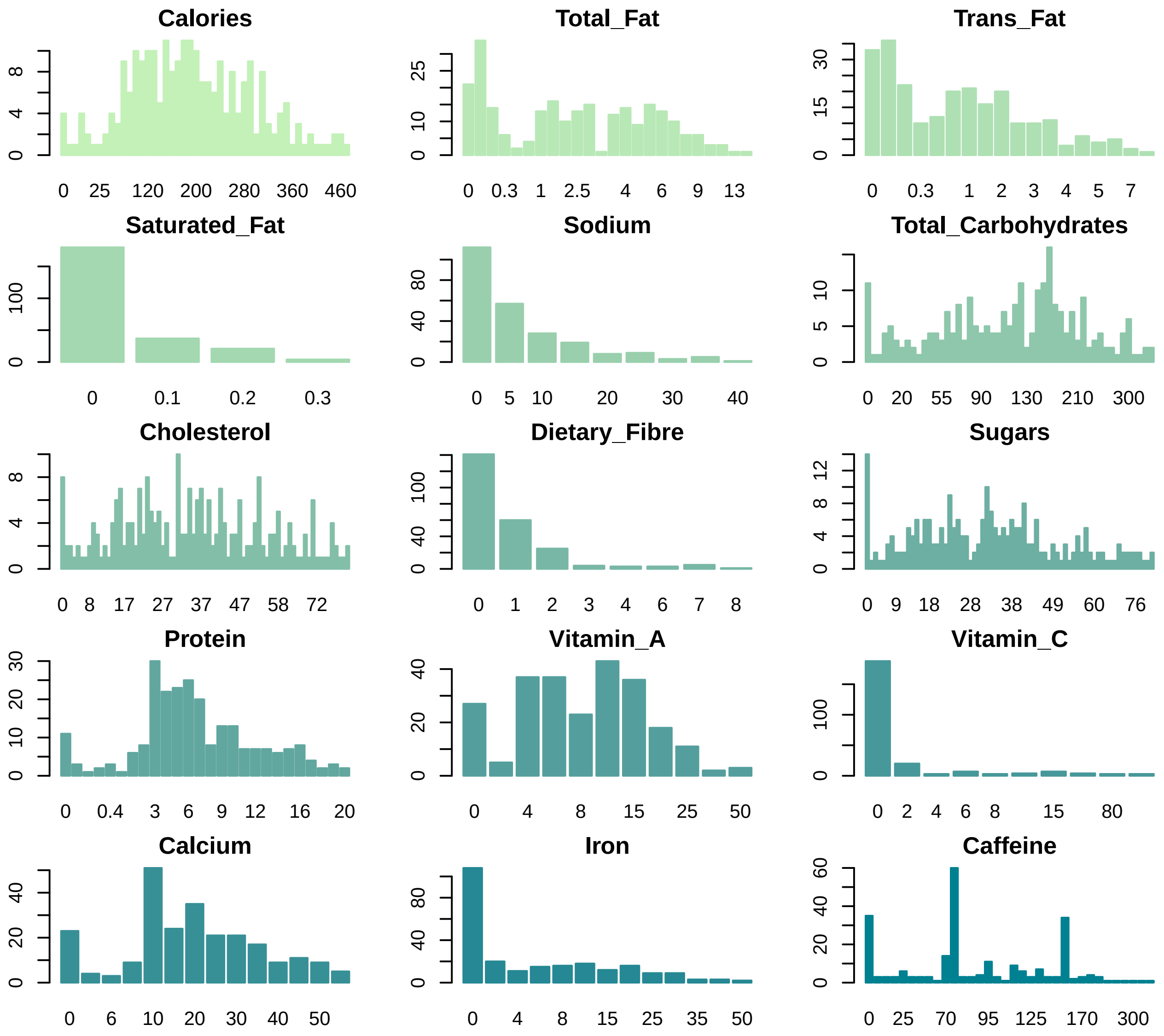
Histograms serve as a graphical interpretation of data distribution. In a histogram, each bar corresponds to the counted frequency within each bin or interval. We introduce these plots to see if our data is normally distributed, skewed, or has outlier values.



By looking at the graphs, we can notice that the variables `"Calories"`, `"Total_Carbohydrates"`, `"Cholesterol"`, and `"Sugars"` exhibit distributions that are nearly normal. Conversely, the distributions of the remaining variables display a noticeable skewness towards the left, confirming that our data is considerably unbalanced.

4.2 Barplot

We will now plot the bar plots for our dataset. The primary use of bar plots is to make comparisons between the amounts of different categories. Indeed, each bar corresponds to a category and the height of the bar represents the frequency or proportion of that category.



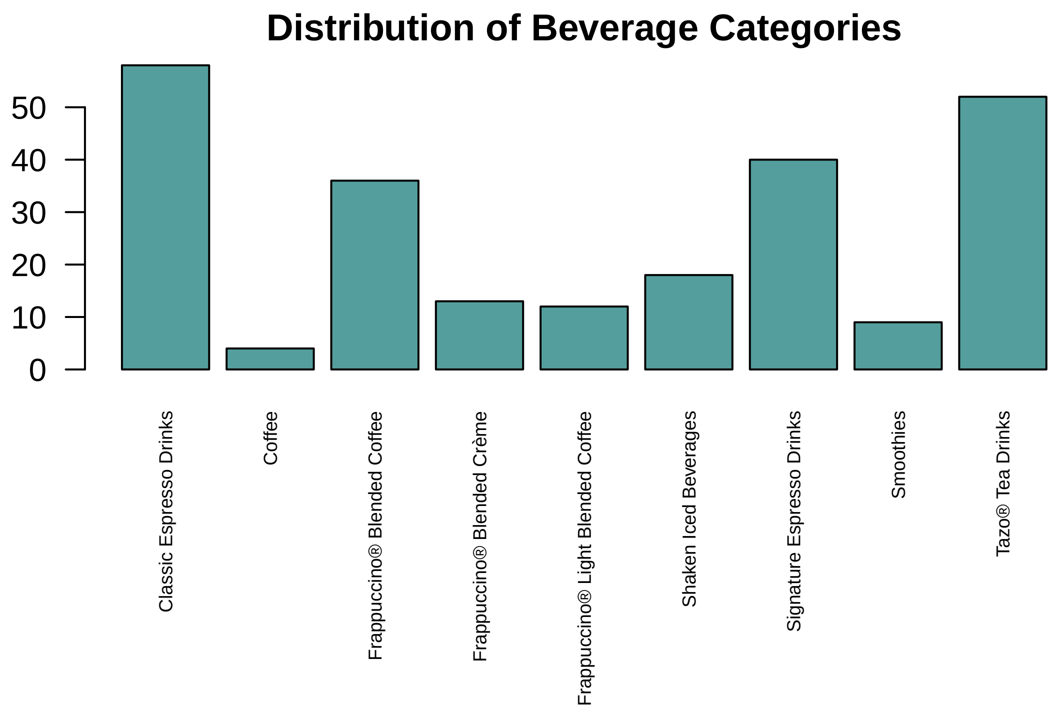
We can deduce some useful information by looking at these plots.

For example, we can notice that variables such as `"Saturated_Fat"`, `"Dietary_Fibre"`, `"Vitamin_C"`, and `"Iron"` are typically either absent or present in small quantities in the beverages. In particular, the frequency of these variables rapidly diminishes as their levels increase. On the other hand, the variables `"Calories"`, `"Total_Fat"`, `"Trans_Fat"`, and `"Total_Carbohydrates"` show a wide range of values across different beverage types, going from high levels in some beverages to minimal amounts in others.

We can further observe that the distribution of `"Vitamin_A"` appears to be more evenly spread among the different levels in various beverages, while instead `"Caffeine"` plot is interesting as it exhibits three distinct peaks in frequency.

4.2.1 Beverages Barplot

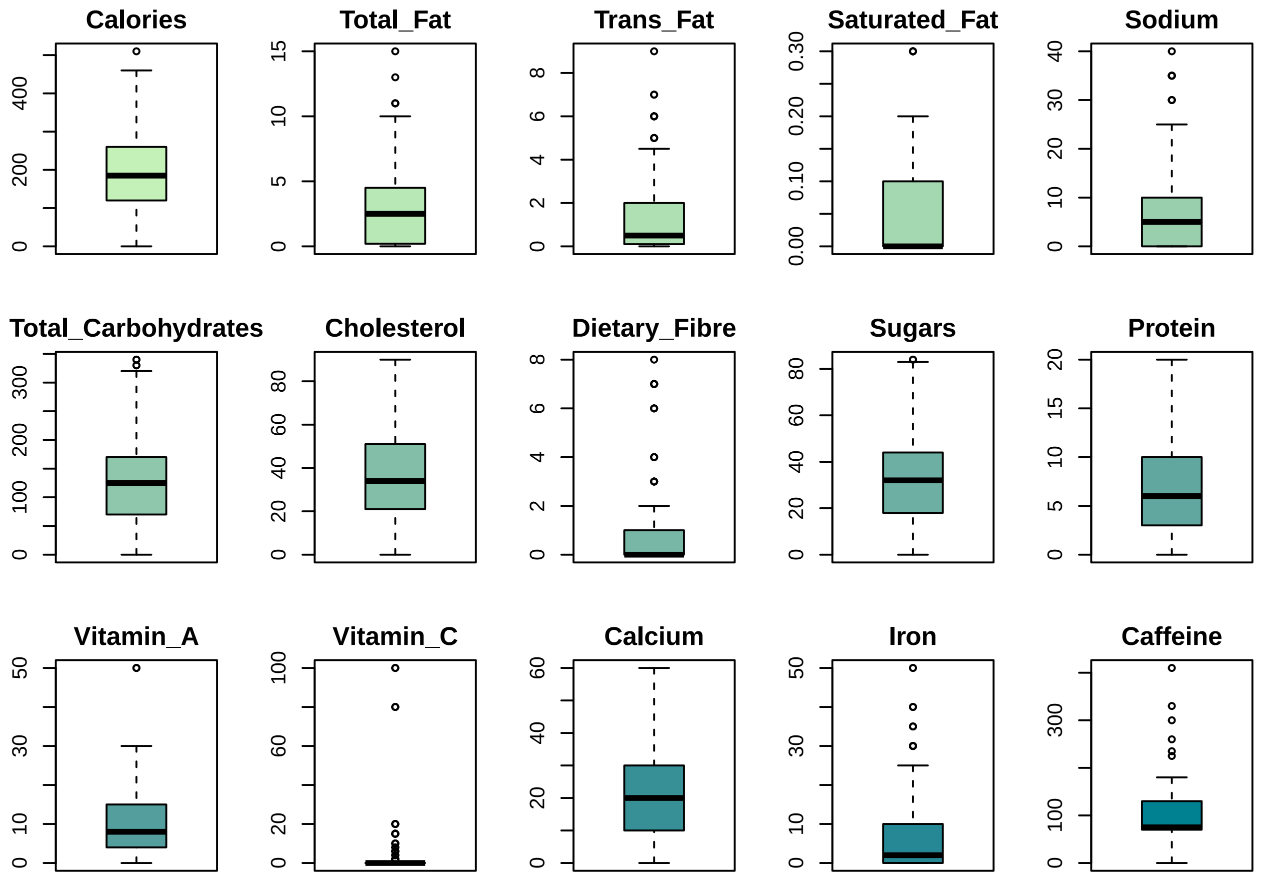
As previously anticipated, bar plots also allows us to see the distribution of categorical variables like "Beverage_category". In this way we can identify the most frequently occurring beverages and their preparation methods.



As we an see from the barplot the most frequent beverage categories are Classic Espresso Drinks and Tazo Tea Drinks. The least favorite instead is basic Coffee.

4.3 Boxplot

Boxplots are a type of graphical representation used to display the distribution of a dataset. They provide a visual summary of the data, enabling us to quickly identify key statistical measures such as median, quartiles and outliers. This visualization also helps us to determine the spread and variability of the data.



As we observed earlier, the majority of the graphs exhibits a skewness towards zero, with the exceptions being `"Calories"`, `"Total_Carbohydrates"`, `"Cholesterol"`, and `"Sugars"`.

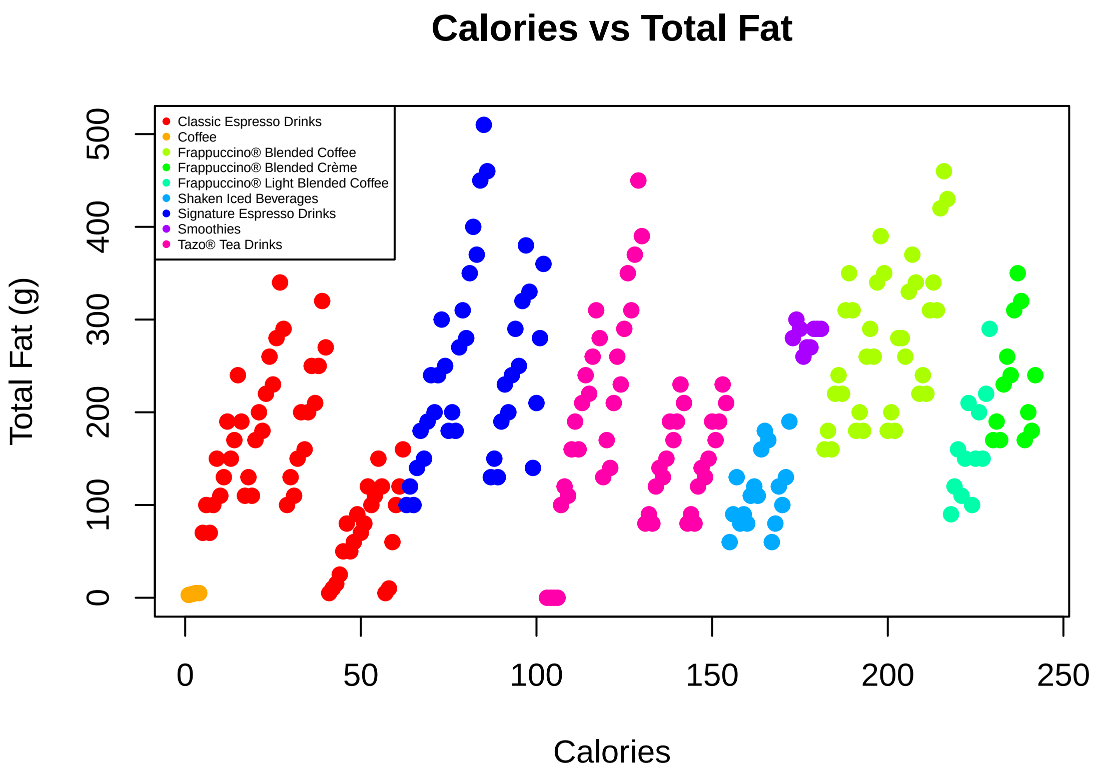
Another aspect that has not been previously highlighted is the presence of outliers. These are notably evident in `"Dietary_Fiber"`, `"Vitamin_C"`, and `"Caffeine"` plots.

Additionally we can highlight that the range of values for each variable varies significantly. For instance, the range for `"Calories"` is $(0, 500)$, whereas for `"Trans_Fat"` is $(0, 8)$.

4.4 Scatterplot

A scatterplot is a type of data visualization that uses dots to represent the values obtained for two different variables - one plotted along the x-axis and the other plotted along the y-axis. Scatterplots are used to observe relationships between variables. This type of graphical representation is crucial in detecting underlying patterns and potential correlations among the variables.

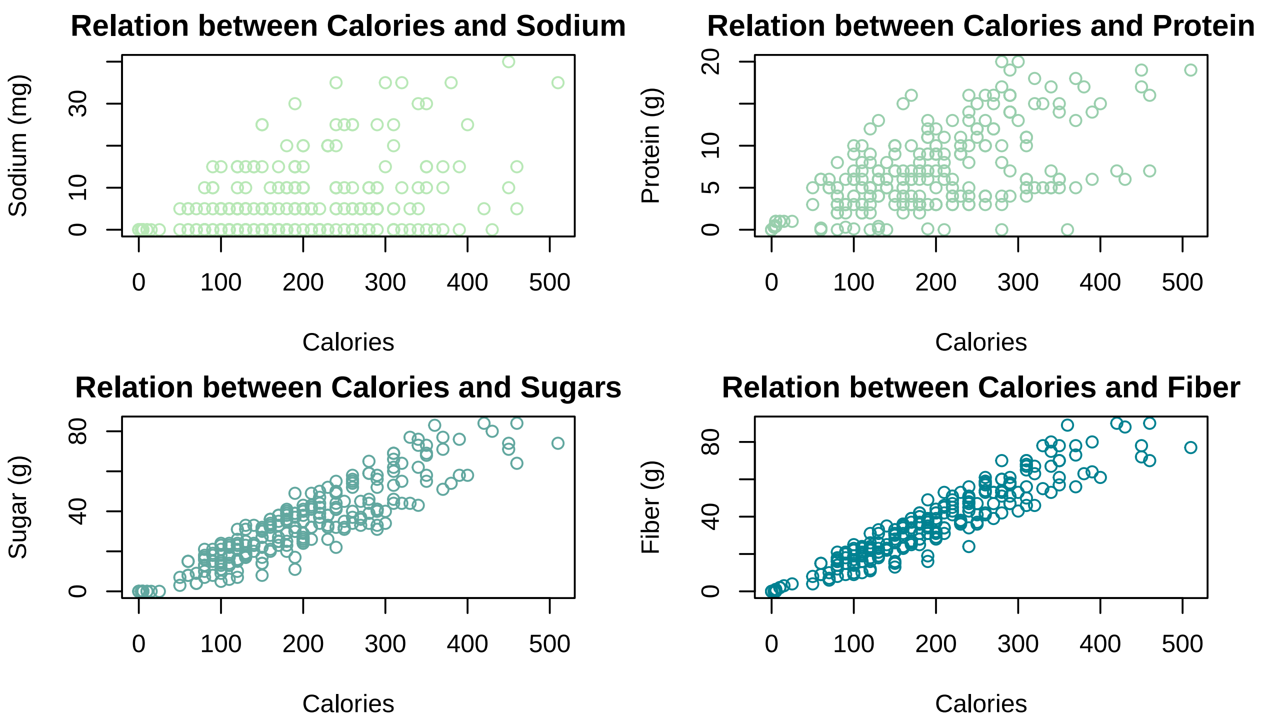
In particular, we place the calorie content and fat levels of various beverage categories side by side for comparison. To make the visualization more intuitive, we assign distinct colors to each beverage category and create a legend to identify each category.



Let us look for any overall pattern or trend in the data points. For all the categories the two variables seem to be related following an almost linear trend, with a positive correlation - as one variable increases, so does the other. However, it is important to note that a very high `"Total_Fat"` value does not necessarily equate to a very high `"Calories"` value - see `"Espresso Signature Drinks"` category.

Additionally we can observe that, given a specific category, it is possible that we find two clusters of data points that follow distinct distibutions. This phenomenon is observed in the `"Classic Espresso Drinks"`, `"Espresso Signature Drinks"` and `"Tazo Tea Drinks"` categories and it is likely attributable to the diverse methods of drink preparation.

Finally, we create some scatterplots to look into relantionship between `"Calories"` and other variables. In particular we focus on `"Sodium"`, `"Protein"`, `"Sugars"` and `"Dietary_Fiber"`.



With an increase in calories, we observe a corresponding rise in all features. However, the rate of increase varies among different features. For instance, `"Sugars"` and `"Dietary_Fiber"` show a steep ascent, indicating a rapid increase with calorie count. On the other hand, `"Sodium"` and `"Protein"` exhibit a more gradual growth, suggesting a slower rate of increase despite the rising calorie content. These observations are further substantiated by the correlation coefficients, which provide a quantitative measure of these relationships. This highlights the complex interplay between calories and various nutritional components in our beverages.

5 Regression Analysis

In this section we will conduct a comprehensive regression analysis on our dataset, exploring our data, constructing different models and lastly performing model selection and validation. Our goal is to build a model that accurately represents the relationships within our data and can provide meaningful predictions. In particular, the variable we want to predict is `"Calories"`. To achieve this, we will first fit a linear regression model, using the subset `data_cleaned` from the original data containing numerical variables only, to predict the amount of calories, based on the amounts of the other variables. We will then compare different models, evaluate their performance by using various metrics such as AIC, BIC, R-squared, and adjusted R-squared, and select the best model for our data.

In our analysis we preferred the models that show low AIC, BIC values and values of R-squared, adjusted R-squared close to 1, as this implies a more accurate fit to the data. We emphasize that these measures should be interpreted in the context of a comparison between models rather than in absolute terms.

5.1 Linear Regression

The simplest form of regression analysis is linear regression, where we predict an outcome variable, in our case`"Calories"`, based on one or more predictor variables. We use the `lm()` function to fit a linear regression model and then we evaluate our model.

5.1.1 Simple Linear Regression

We fit a linear simple regression using just one predictor variable: we choose the variable `Sugars` due to the high correlation, as previously noticed. Indeed this variable seems to be the mostly influential to predict the amount of calories.

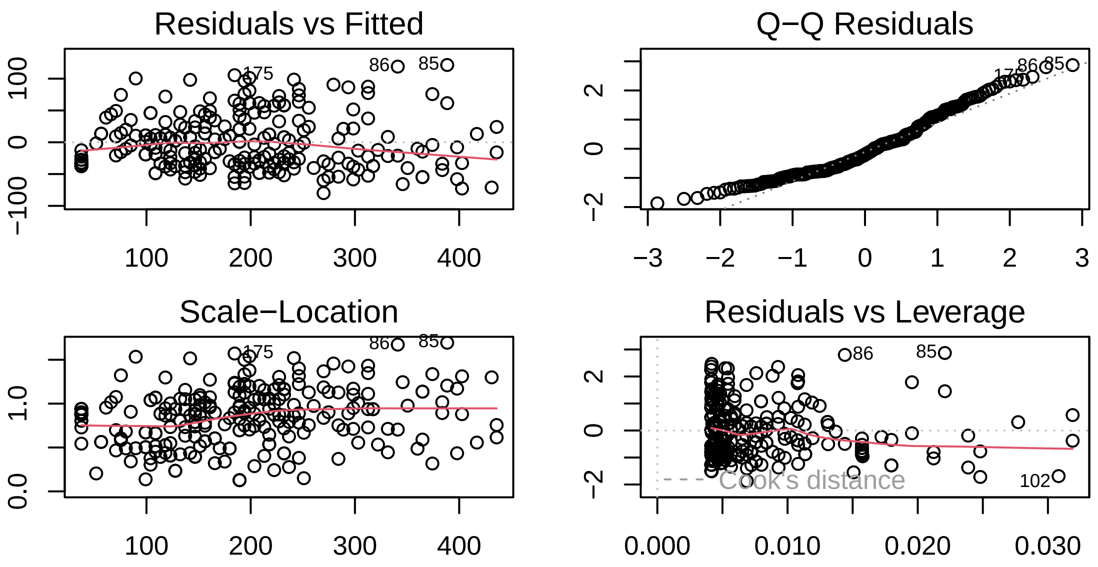
We provide a summary of the code in following lines.

```
lm_simple <- lm(Calories ~ Sugars, data = data_cleaned)
kable(data.frame(AIC = AIC(lm_simple), BIC = BIC(lm_simple),
                 R_squared = summary(lm_simple)$r.squared,
                 adj_R_squared = summary(lm_simple)$adj.r.squared),
      caption = "Model evaluation metrics for the simple linear regression model")
```

Table 1: Model evaluation metrics for the simple linear regression model

AIC	BIC	R_squared	adj_R_squared
2509.036	2519.503	0.8275094	0.8267907

```
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
plot(lm_simple)
```



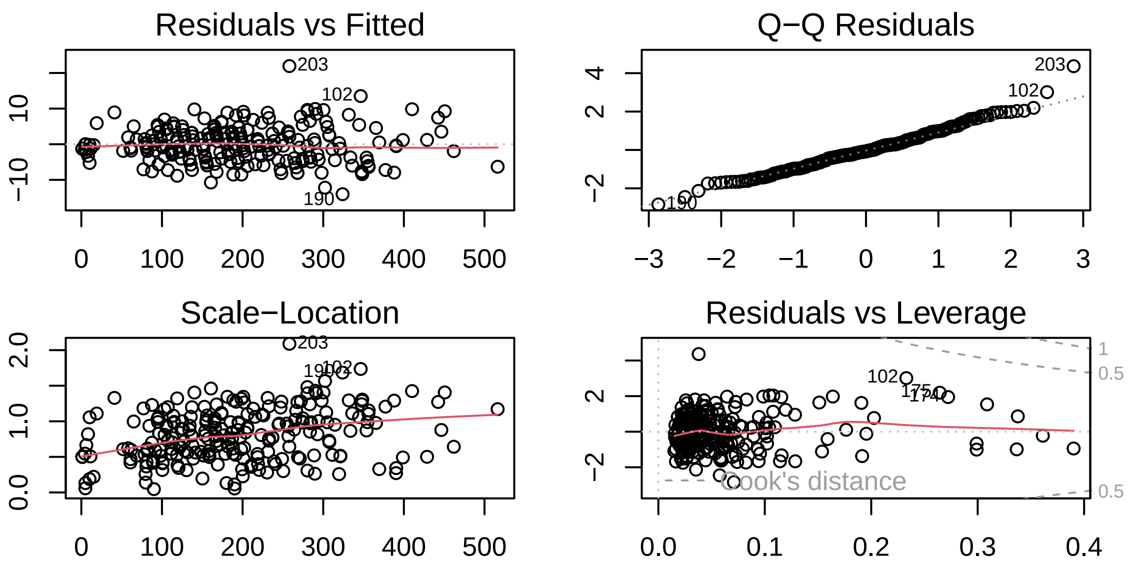
The coefficient 4.7426 for `"Sugars"` indicates that, on average, for every one-unit increase in `"Sugars"`, the predicted value of `"Calories"` increases by approximately 4.7426 units. Both the intercept and the coefficient for our predictor are statistically significant ($p < 0.001$), indicating a strong linear relationship between `"Sugars"` and `"Calories"`. The F-statistic is highly significant ($p < 2.2e-16$), indicating that the overall regression model is statistically significant in explaining the variance in `"Calories"`. This is also confirmed by the value of the adjusted R-squared: we obtain 0.8268, suggesting that roughly 82.68% of the variance in `"Calories"` can be explained by the predictor variable `"Sugars"`.

However, the AIC and BIC values suggest that there might be other models that provide a better fit for the data. Summarizing, the model appears to be too simple to fully capture the complexity of our data. For this reason we try to fit a multiple linear regression model.

5.1.2 Multiple Linear Regression

This method extends simple linear regression, which involves only one predictor, by allowing for the inclusion of multiple predictors, thereby providing a more comprehensive analysis of the factors that influence the dependent variable. In our specific case we predict again `"Calories"` using a full model containing all the other numerical variables.

```r
lm_model <- lm(y ~ ., data = data_num_)
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
plot(lm_model)
```



```r
kable(data.frame(AIC = AIC(lm_model), BIC = BIC(lm_model),
                 R_squared = summary(lm_model)$r.squared,
                 adj_R_squared = summary(lm_model)$adj.r.squared),
      caption = "Model evaluation metrics for the linear regression model")
```

Table 2: Model evaluation metrics for the linear regression model

AIC	BIC	R_squared	adj_R_squared
1494.304	1550.127	0.9976608	0.9975166

We can observe that this model instead has lower AIC and BIC values and a higher R-squared value, 0.997, providing an excellent fit for the data. Notice also that most of the variables coefficients are significant, therefore all the predictors contribute in explaining the dependent variable.

5.1.3 Backward Elimination

Now we apply backward elimination method, performing a selection of predictors. This method starts with all the predictors in the model and then removes the least significant predictor one at a time until all remaining predictors are significant. Each time we remove a variable the method updates all the coefficients, since the relationships between the variables may vary.

```r
backward_model <- step(lm_model, direction = "backward")
```

Table 3: Model evaluation metrics for the linear regression model with backward elimination

AIC	BIC	R_squared	adj_R_squared
1492.616	1544.95	0.9976578	0.9975243

Backward selection only drops the variable `"Saturated_Fat"`, which is not considered significant, while all other variables are maintained, as they play a crucial role in explaining the `"Calories"` variable.

Interpreting the remaining coefficients we obtain useful information on the relationships between the variables: for instance with the increase of `"Trans_Fat"` variable, the calories experience a decline of 2.34 units, by maintaining constants all the other variables.

Comparison between the models:

Table 4: Model comparison

Model	AIC	BIC	R_squared	adj_R_squared
Simple Linear Regression	2509.036	2519.503	0.8275094	0.8267907
Multiple Linear Regression	1494.304	1550.127	0.9976608	0.9975166
Multiple Linear Regression with Backward Elimination	1492.616	1544.950	0.9976578	0.9975243

The multiple linear regression model with backward elimination has the lowest AIC and BIC values, the highest R-squared value, and the highest adjusted R-squared value, indicating that it is the best model for predicting the amount of calories based on the amount of the other variables.

Coefficients and Significance of Variables: Both multiple regression models have very similar coefficients for the variables that were retained. The removal of `"Saturated_Fat"` in the backward model did not significantly affect the estimates of the other coefficients. Indeed, in the full model `"Saturated_Fat"` had a high p-value (0.589), indicating it was not a significant variable. In the backward model, `"Saturated_Fat"` was removed, slightly improving the AIC while keeping all other variables significant.

Overall Performance: Both multiple regression models perform very similarly in terms of R-squared and residual standard error. The backward model is preferable because it has a slightly lower AIC, suggesting it is a more parsimonious model without sacrificing the quality of the fit.

5.1.4 Anova

We now use ANOVA to determine whether there are any statistically significant differences between the means of our multiple regression models.

```
## Analysis of Variance Table
##
## Model 1: y ~ Total_Fat + Trans_Fat + Saturated_Fat + Sodium + Total_Carbohydrates +
##     Cholesterol + Dietary_Fibre + Sugars + Protein + Vitamin_A +
##     Vitamin_C + Calcium + Iron + Caffeine
## Model 2: y ~ Total_Fat + Trans_Fat + Sodium + Total_Carbohydrates + Cholesterol +
##     Dietary_Fibre + Sugars + Protein + Vitamin_A + Vitamin_C +
##     Calcium + Iron + Caffeine
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    227 5964.8
## 2    228 5972.5 -1   -7.6917 0.2927  0.589
```

Degrees of Freedom (Res.Df): The full model has 227 degrees of freedom, while the backward model has 228. This is because we removed one variable from the full model.

Residual Sum of Squares (RSS): The full model has an RSS of 5964.83, while the backward model has an RSS of 5972.55. This indicates that the difference between the two models in terms of residual error is very small.

Sum of Squares (Sum of Sq): The difference between the two models in terms of sum of squares is -7.7235, indicating that the removed variable (`"Saturated_Fat"`) does not significantly contribute to explaining the variability in calories, as saw before.

F-statistic (F): The F value is 0.2937 with a p-value of 0.588. This high p-value indicates that there is no significant difference between the two models. In other words, the reduced model is not significantly worse than the full model.

The ANOVA shows that the removal of the `"Saturated_Fat"` variable does not have a significant impact on the model. This confirms that the model obtained through backward selection is more parsimonious without compromising the quality of the fit. Therefore, the backward model is preferable to the full model.

5.1.5 Multicollinearity

In a multiple regression model the presence of multicollinearity occurs when two or more independent variables are highly correlated, making it challenging to accertain the unique contribution of each predictor variable: changes in one predictor are often accompagned by a change in another

To check for multicollinearity, we calculate the Variance Inflation Factors (VIF), which measures how much the variance of the estimated coefficients is increased due to multicollinearity. Usually a VIF value greater than 10 indicates a problematic amount of multicollinearity.

Table 5: VIF values for the linear regression model

	VIF
Total_Fat	17.863697
Trans_Fat	14.667324
Sodium	4.448925
Total_Carbohydrates	3.419094
Cholesterol	442.886703
Dietary_Fibre	16.896773
Sugars	417.769822
Protein	56.706156
Vitamin_A	4.205667
Vitamin_C	4.288442
Calcium	37.105615
Iron	5.027804
Caffeine	1.176323

As we can see from the *Table X*, we have a problem with multicollinearity, since the VIF values are high for some variables. In particular `"Cholesterol"` and `"Sugars"` have a very large VIF value, meaning that the two variables provide redundant and superflous information. We have to act on the data to solve this problem.

The high values of the VIF could be due to:

- High correlation between variables
- Same information
- Unbalanced data
- Different Measurement Scales: If the variables in the model have significantly different measurement scales such as g and mg this could affect the VIF values.
- Non-linear Relationships: If the relationships between the variables are non-linear, this could also affect the VIF values.

As previously said, we have different unit measures in our dataset, so normalizing the variables might help reduce multicollinearity. We proceed to then standardize the data.

5.1.6 Standardize the data

We have tried different types of standardization to reduce the multicollinearity. First of all we tried standard normalization, which helps to reduce the influence of different measurements scales. The negative value of the AIC (-748.2623) indicates that the standardized linear regression model provides a better compromise between data fit and model complexity compared to the reference model. However, the VIF values are still high, indicating that multicollinearity is still present in the model.

To reduce the problem of high VIF in linear regression, it is generally preferable to use the transformation that includes both log transformation and standardization of the data. So we combined log transformation that reduces the variance of the variables, making the distribution more normal and reducing the impact of outliers and standardization which puts all variables on a common scale, with mean 0 and standard deviation 1.

```r
std_data_log <- scale(log(data_num + 1)) # Standardize the data
std_data_log_df <- as.data.frame(std_data_log) # Set as dataframe
mod_log_tr <- lm(Calories ~ ., data = std_data_log_df)
kable(data.frame(AIC = AIC(mod_log_tr), BIC = BIC(mod_log_tr),
                 R_squared = summary(mod_log_tr)$r.squared,
                 adj_R_squared = summary(mod_log_tr)$adj.r.squared),
      caption = "Model evaluation metrics for the log transformed data")
```

Table 6: Model evaluation metrics for the log transformed data

AIC	BIC	R_squared	adj_R_squared
-53.42411	2.398897	0.9586932	0.9561457

```r
kable(data.frame(VIF = vif(mod_log_tr)),
      caption = "VIF values for the log transformed data")
```

Table 7: VIF values for the log transformed data

	VIF
Total_Fat	12.049669
Trans_Fat	10.577306
Saturated_Fat	4.528080
Sodium	5.817088
Total_Carbohydrates	4.363628
Cholesterol	39.988684
Dietary_Fibre	7.115085
Sugars	38.415586
Protein	31.007121
Vitamin_A	13.647581
Vitamin_C	2.196674
Calcium	25.873742
Iron	4.582594
Caffeine	1.310005

The model has a low AIC and BIC values, the R-squared value is 0.95 so the model is a good fit for the data. Nevertheless, we still have collinearity, so we try to use backward elimination to remove the variables that are unnecessary for the model.

```r
backward_model_log <- step(mod_log_tr, direction = "backward")
```

Table 8: Model evaluation metrics for the log transformed data with backward elimination

AIC	BIC	R_squared	adj_R_squared
-63.78109	-32.38065	0.9580667	0.9568123

Table 9: VIF values for the log transformed data with backward elimination

	VIF
Total_Fat	5.823786
Trans_Fat	5.161252
Total_Carbohydrates	3.390622
Cholesterol	36.628698
Dietary_Fibre	1.851534
Sugars	33.948827
Protein	2.976444

Now the AIC value is slightly worst and the VIF values are still high, indicating that multicollinearity is still present in the model. Therefore we try to manually remove the variables that have high VIF values, which, as we can see from the table, are `"Cholesterol"` and `"Sugars"`.

```r
mod_log_tr_updated <- lm(Calories ~ . - Cholesterol - Sugars,
                         data = std_data_log_df)
kable(data.frame(VIF = vif(mod_log_tr_updated)),
      caption = "VIF values for the log transformed data with
      manual removal of variables")
```

Table 10: VIF values for the log transformed data with manual removal of variables

	VIF
Total_Fat	11.902918
Trans_Fat	10.114112
Saturated_Fat	4.466975
Sodium	5.782843
Total_Carbohydrates	3.375194
Dietary_Fibre	7.080360
Protein	26.902392
Vitamin_A	12.396739
Vitamin_C	1.985799
Calcium	25.519022
Iron	4.521552
Caffeine	1.295121

```r
mod_log_tr_backward_2 <- step(mod_log_tr_updated, direction = "backward")
```

Table 11: Model evaluation metrics for the log transformed data with backward elimination and manual removal of variables

AIC	BIC	R_squared	adj_R_squared
460.5536	488.4651	0.6309185	0.6214952

Table 12: VIF values for the log transformed data with backward elimination and manual removal of variables

	VIF
Trans_Fat	1.491986
Total_Carbohydrates	2.649737
Protein	10.478245
Vitamin_A	8.772514
Vitamin_C	1.241962
Iron	1.304571

Finally, VIF values are all below or around 10, implying that multicollinearity has been reduced in the model. We furthermore notice that the R-squared value has decreased, but it is still a good value.

We perform a model diagnostic by doing a Shapiro-Wilk test.

Model diagnostic: Non-normal residuals suggest that some assumptions of linear regression might be violated. Specifically, the assumption of normality of the residuals is not met, this can affect the validity of hypothesis tests on the coefficients and predictions.

```
shapiro.test((residuals(mod_log_tr_backward_2)))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  (residuals(mod_log_tr_backward_2))
## W = 0.82171, p-value = 5.816e-16
```

Given that the p-value is significantly smaller than 0.05, we reject the null hypothesis. This indicates that the residuals of the model `mod_log_tr_backward_2` do not follow a normal distribution. In this case, W is quite a bit lower than 1, suggesting the residuals deviate from normality. As a strategy to overcome this issue we should use a robust regression model, which does not require the assumption of normality of errors.

We have tried other trasformation like min-max scaling and robust scaling but these were not satisfactory due to VIF values that were still to high. After this data manipulation we changed our approach by using regularization methods such as ridge regression or lasso regression. These methods penalizes the variable coefficients, helping to reduce multicollinearity.

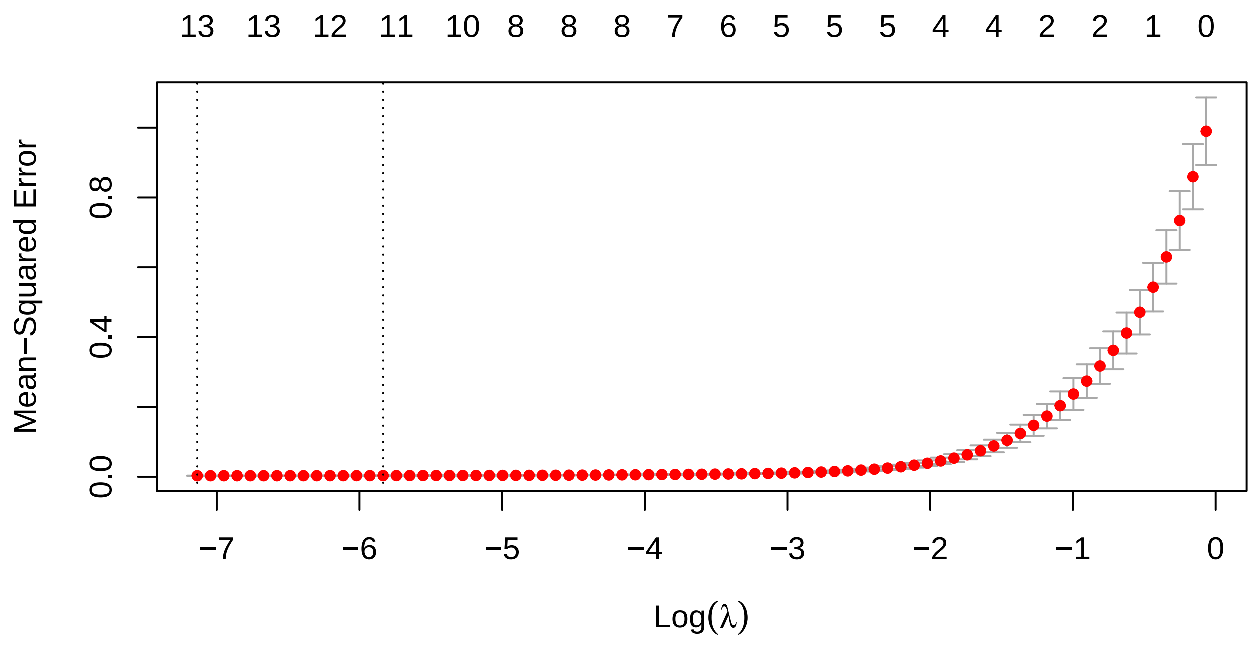
5.2 Lasso Regression

We use the `glmnet` package to fit a lasso regression model. Lasso regression, a type of linear regression that employs L1 regularization, penalizes the model's coefficients. This approach helps prevent overfitting and identifies the most significant features in the data.

First, we standardize the data and then fit the lasso regression model using the `cv.glmnet` function. Cross-validation is employed to select the optimal lambda value for the model. The lambda value that minimizes the mean squared error (MSE) is chosen as the optimal value, which is then used to fit the final lasso regression model.

Lasso regression tends to shrink the coefficients of less important variables towards zero, effectively performing variable selection. By eliminating irrelevant variables, it reduces the number of predictors and, consequently, multicollinearity. This reduction in variables decreases multicollinearity among predictors, resulting in lower VIF values. The automatic feature selection inherent in lasso regression removes redundant variables and reduces multicollinearity in the model.

```
std_data <- as.data.frame(scale(data_num)) # Standardize the data
mod_lasso <- cv.glmnet(x = as.matrix(std_data[, -1]),
                       y = std_data$Calories, alpha = 1, standardize = FALSE)
par(mfrow = c(1, 1))
plot(mod_lasso, xvar = "lambda", label = TRUE)
```



By setting some coefficients to zero (such as `"Saturated_Fat"`), lasso regression aids in feature selection, thereby reducing the model's complexity. The remaining non-zero coefficients indicate the variables that significantly contribute to predicting calories. The signs and magnitudes of these coefficients illustrate the direction and strength of their relationships with the target variable (`"Calories"`). For example, a one-unit increase in `"Sodium"`, holding all other variables constant, is associated with a decrease of approximately 0.021 calories.

The `cv.glmnet` function performs cross-validation to determine the lambda value that minimizes the prediction error, identified as `lambda.min`. In our case, we found that the optimal lambda value is equal to 1.

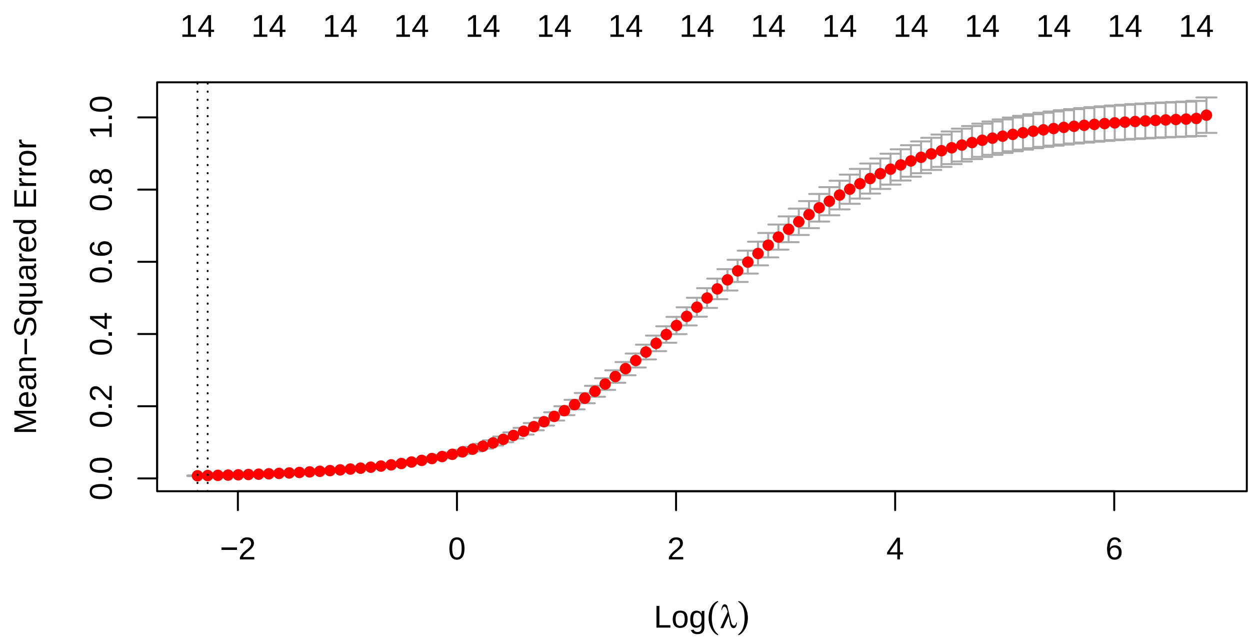
To further evaluate the model's performance, metrics such as R-squared and Mean Squared Error (MSE) should be considered.

5.3 Ridge Regression

We also tried ridge regression to reduce multicollinearity. Like lasso regression, ridge regression is a regularization technique that introduces a penalty term, L2 regularization, to shrink coefficients towards zero without eliminating them entirely, meaning that all variables can remain in the model. Unlike lasso, which can zero out some coefficients and thus perform variable selection, ridge regression retains all variables, making it suitable for reducing multicollinearity without excluding any variables. Lasso may be preferred when selecting a subset of the most relevant variables is desired.

We use the `glmnet` package to fit a ridge regression model. The ridge regression model is fit using the `cv.glmnet()` function, which incorporates cross-validation to determine the optimal lambda value. Similar to lasso, the best lambda value, which minimizes the error, was found to be equal to 1. The interpretation of the coefficients is the same as in lasso regression.

```r
mod_ridge <- cv.glmnet(x = as.matrix(std_data[, -1]),
                       y = std_data$Calories, alpha = 0, standardize = FALSE)
plot(mod_ridge, xvar = "lambda", label = TRUE)
```



5.4 Model Comparison

We compare the linear regression, lasso regression, and ridge regression models to select the best model for predicting the amount of calories based on the amount of the other variables. We evaluate the models using the R-squared value, and the Mean Squared Error (MSE) for each model.

Table 13: R-squared values for the models

Model	R_squared
Linear Regression	0.9976608
Lasso Regression	0.9975756
Ridge Regression	0.9941815

The R-squared value resulted from Lasso regression, approximately 0.998, indicates that the model explains about 99.76% of the variance in the `"Calories"` variable. This suggests a very strong fit, as the model is capturing almost all the variability in the target variable.

The value obtained from Ridge regression is slightly worse but still very high.

5.5 Model Evaluation

We evaluate the performance of the linear regression, lasso regression, and ridge regression models using the mean squared error (MSE). The MSE is a measure of the average squared difference between the predicted and actual values. Lower values of the MSE indicate better performance of the model.

Table 14: MSE values for the models

Model	MSE
Linear Regression	24.6481166
Lasso Regression	0.0024158
Ridge Regression	0.0066477

We choose the model with the highest R-squared value and the lowest MSE as the best model for predicting the amount of calories based on the amount of the other variables. The best model is the lasso because it has the highest value for R^2 and lowest MSE and it is the most robust model.

The MSE of Lasso, approximately 0.0024, shows a very low average squared difference between the observed actual outcomes and the outcomes predicted by the model. This suggests that the model's predictions are very close to the actual values, implying an high accuracy.

The variables `"Total_Fat"`, `"Trans_Fat"`, `"Sodium"`, `"Total_Carbohydrates"`, `"Cholesterol"`, `"Dietary_Fibre"`, `"Sugars"`, `"Protein"`, `"Vitamin_A"`, `"Vitamin_C"`, `"Calcium"`, `"Iron"`, and `"Caffeine"` have coefficients of significant magnitudes, suggesting that these variables are important for predicting calories. Overall, the absence of coefficients with very large magnitudes and the presence of coefficients close to zero for some variables suggest that the Lasso model may have helped mitigate multicollinearity and select only the most important variables for predicting calories.

5.6 Cross Validation

Cross validation is a technique used to evaluate the performance of a model. It involves splitting the data into training and testing sets, fitting the model using the training set, and evaluating the model using the testing set. We decided to split the data with 80% of examples for training and 20% for testing.

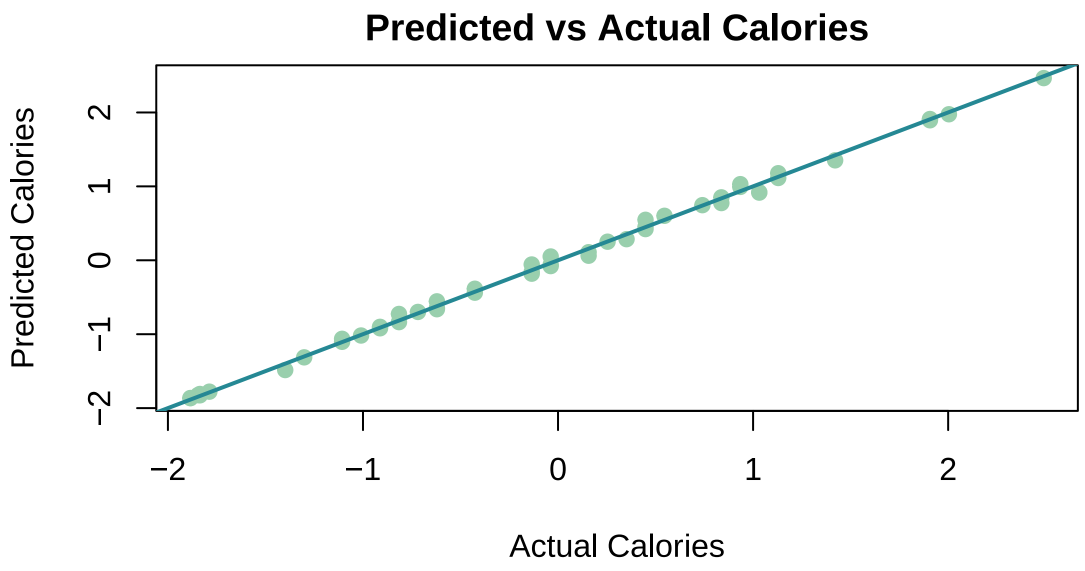
```r
set.seed(123)
train_index <- sample(1:nrow(std_data), 0.8 * nrow(std_data))
train_data <- std_data[train_index, ]
test_data <- std_data[-train_index, ]
mod_lasso_train <- cv.glmnet(x = as.matrix(train_data[, -1]), y = train_data$Calories,
                             alpha = 1, standardize = FALSE)
```

We evaluate the model using the testing set, making predictions using the test set and calculating the mean squared error and the root mean squared error to assess the model's accuracy.

```r
lasso_pred_test <- predict(mod_lasso_train, s = "lambda.min",
                           newx = as.matrix(test_data[, -1]))
lasso_r_squared_test <- cor(lasso_pred_test, test_data$Calories)^2
lasso_mse_test <- mean((lasso_pred_test - test_data$Calories)^2)
```

Overall, the high R-squared value and low MSE on the test data suggest that the Lasso regression model has learned effectively from the training data and generalizes well to unseen examples.

```r
accuracy_lm <- 1 - (lasso_mse_test / var(test_data$Calories))
par(mfrow = c(1, 1), mar = c(4, 4, 2, 2))
plot(test_data$Calories, lasso_pred_test, xlab = "Actual Calories", col = "#99cfad",
     ylab = "Predicted Calories", main = "Predicted vs Actual Calories", pch = 19)
abline(0, 1, col = "#258894", lwd = 2)
```



```r
kable(data.frame(Accuracy = accuracy_lm, MSE = lasso_mse_test,
                 R_squared = lasso_r_squared_test),
      caption = "Model evaluation metrics on the test data")
```

Table 15: Model evaluation metrics on the test data

	Accuracy	MSE	R_squared
lambda.min	0.9979473	0.0026283	0.9979454

As we can see from *Table X* the R-squared value is 0.997, indicating that the model explains 99% of the variance in the data and the MSE is 0.002628338, indicating that the model has a low error rate. The accuracy of the model is 0.9979473, showing that the model is able to predict the amount of calories with high accuracy. In the plot of the predicted values against the actual values on the test data, the points are close to the diagonal line, indicating that the model is making accurate predictions.

6 Classification Analysis

6.1 Logistic Regression

Logistic regression is a statistical model used to predict the outcome of a binary categorical dependent variable based on one or more independent variables. Since logistic regression is tipycally used for classification task and our variable `"Calories"` ,that we want to predict is continous random variable, we have to traspose the problem into a classification one by making the variable binary. In order to do that we classify foods into two categories: “low calorie” and “high calories” defining a threshold to distinguish between the two classes.

We’ve tried with normal data, standardize data, and log trasformation since again it helps to reduce multicollinearity and we find out that the best is with log trasformed data. Looking at the summary and the plot of the variable, we notice that calories follow a semi-gaussian distribution both median and mean are reasonable approach to use, since they are close to each other. However we choose median, as the treshold is less sensitive to outliers and skewness in our data. It ensures that half the data points are classified as `"low calories"` and the other half as `"high calories"`, providing balanced classes.

```r
y <- std_data_log_df$Calories
calories_median <- median(y)
std_data_log_df$Calorie_Class <- ifelse(std_data_log_df$Calories > calories_median, 1, 0)
table(std_data_log_df$Calorie_Class)
```

```
##
##   0   1
## 121 121
```

```r
logistic_model <- glm(Calorie_Class ~ ., data = std_data_log_df[,-1], family = binomial)
kable(data.frame(AIC = AIC(logistic_model), BIC = BIC(logistic_model),
                 Residual_deviance = logistic_model$deviance,
                 Null_deviance = logistic_model$null.deviance,
                 R_squared = 1 - logistic_model$deviance / logistic_model$null.deviance),
      caption = "Model evaluation metrics for the logistic regression model")
```

Table 16: Model evaluation metrics for the logistic regression model

AIC	BIC	Residual_deviance	Null_deviance	R_squared
69.42364	121.7577	39.42364	335.4832	0.882487

The statistically significant coefficient of the model are `"Cholesterol"` at the 0.01 level and `"Total_Fat"` at the 0.05 level. The residual deviance is much smaller than the null deviance, indicating that the model with predictors explains more variability than the null model. The AIC is 69.424, suggesting that the model has reasonable fit.

Number of Fisher Scoring Iterations: Indicates the number of iterations performed by the Fisher scoring algorithm during model fitting. In this case, it took 11 iterations.

Now we use the model to make predictions on the data and evaluate its performance using the confusion matrix, accuracy, precision, recall, and F1-score.

Table 17: Confusion matrix for the logistic regression model

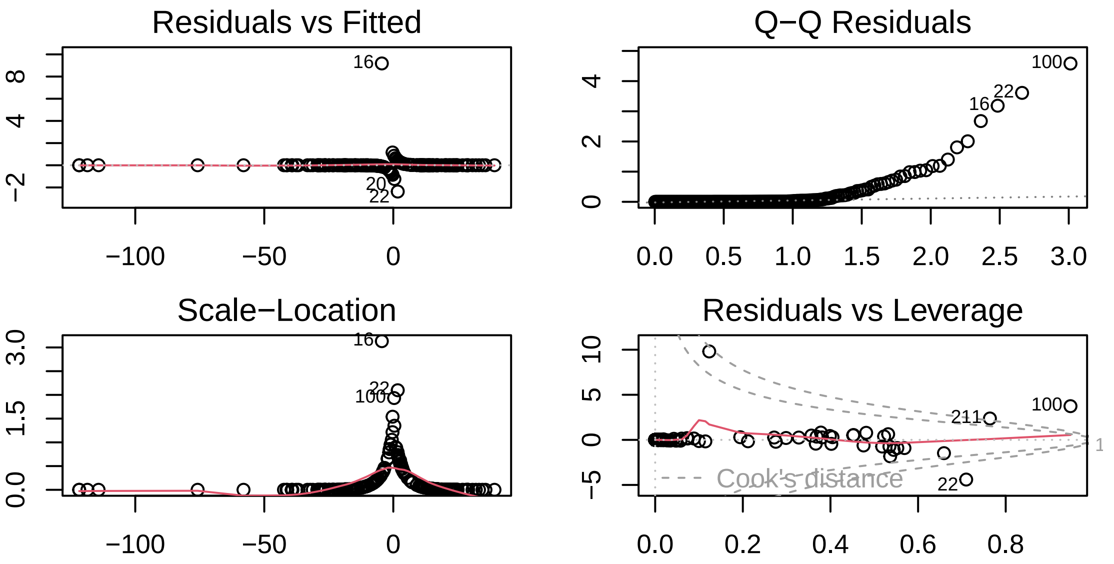
	0	1
0	22	2
1	2	23

Table 18: Model evaluation metrics for the logistic regression model

Accuracy	Precision	Recall	F1_Score
0.9183673	0.92	0.92	0.92

The model's accuracy of approximately 91.8% indicates it is performing well overall in classifying the calorie content correctly and it generalizes well on the test set.

```r
par(mfrow = c(2,2), mar = c(2, 2, 2, 2))
plot(logistic_model_train)
```



Residuals vs Fitted: This plot shows the Pearson residuals against the fitted values. Ideally, there should be no clear pattern, indicating that the model is well-fitted. However, the presence of data points at extreme values (far from 0) suggests potential issues with model fit or outliers.

Normal Q-Q Plot: The Q-Q plot compares the standardized deviance residuals to a theoretical normal distribution. Significant deviations from the straight line suggest that the residuals are not normally distributed, which can indicate potential problems with the model. In this case, the data points deviate from the line, particularly at the higher quantiles, indicating that the residuals are not perfectly normally distributed. The Q-Q plot indicates that the residuals are not perfectly normally distributed, which is expected in logistic regression but still worth noting.

Scale-Location Plot (Spread-Location Plot): This plot shows the square root of the standardized residuals against the fitted values. The red line helps to identify trends. Ideally, the points should be randomly scattered without a clear pattern. Here, we see some clustering and trends at extreme fitted values, suggesting heteroscedasticity or non-constant variance.

Residuals vs Leverage: This plot shows standardized residuals against leverage, highlighting influential data points. The dashed lines represent Cook's distance. Points outside these lines indicate influential observations that have a significant impact on the model. In this plot, several points, especially at higher leverage values, fall outside the dashed lines, indicating they are influential.

6.2 LDA

Linear Discriminant Analysis (LDA) is a statistical technique primarily used for classification and dimensionality reduction. Its main goal is to find a linear combination of features that best separates two or more classes of objects or events.LDA seeks to maximize the separation between classes by minimizing the variance within classes and maximizing the variance between classes.

In the context of our dataset, LDA is useful for classifying different categories of beverages (`"Beverage_category"`). LDA is particularly useful for classification problems where the goal is to assign observations to one of several classes. In our case, we want to classify beverages into different categories based on all numeric variables.

Some advantages of LDA:

- **Dimensionality Reduction:** Even if your dataset has a moderate number of variables, LDA can help reduce the dimensionality of the problem, making classification more manageable and reducing the risk of overfitting.

- **Interpretability:** LDA produces coefficients that are easily interpretable and can provide insights into how each variable contributes to the classification of beverages.

We use the MASS package to fit an LDA model to predict the calorie class based on the other variables in the dataset. We then evaluate the model using the confusion matrix, accuracy, precision, recall, and F1-score.

```
target_var <- data_cleaned$Beverage_category
lda_model <- lda(Beverage_category ~ ., data = train_data)
```

By fitting an LDA model, we can classify new beverage entries based on their numeric attributes.

1. The prior probabilities indicate the proportion of each beverage category in the training dataset. This shows the initial likelihood of each category before considering any predictors.
2. These are the mean values of each numeric predictor for each beverage category. They provide insights into the average nutritional and compositional profile of each beverage type.
   - Calories: Varies widely, with Coffee having the lowest (4.67) and Smoothies the highest (282.22).
   - Total Fat, Trans Fat, Saturated Fat: Reflect the fat content differences across beverage types.
   - Sodium, Total Carbohydrates, Cholesterol: Provide further details on the nutritional content.
   - Dietary Fibre, Sugars, Protein: Highlight the beverage's dietary fiber, sugar, and protein content.
   - Vitamin A, Vitamin C, Calcium, Iron, Caffeine: Show the average micronutrient and caffeine levels.
3. The coefficients for each linear discriminant function (LD1 to LD8) indicate the importance and direction of each predictor in distinguishing between beverage categories.

- LD1: Accounts for 75.36% of the variation between categories. Major contributing factors include Total Fat, Saturated Fat, and Sugars, suggesting these are crucial in separating the beverage types.
- LD2: Accounts for 17.12% of the variation. Trans Fat and Dietary Fibre are significant contributors. Subsequent linear discriminants (LD3 to LD8) account for progressively smaller proportions of the variation.

4. The proportion of trace values indicates how much of the total variability in the dataset is captured by each linear discriminant. LD1 captures the majority (75.36%), followed by LD2 (17.12%), with the remaining discriminants capturing much smaller amounts of variability.

After applying the Linear Discriminant Analysis (LDA) model to classify beverage categories in our dataset, we assessed its performance using a confusion matrix and calculated the accuracy.

```
## [1] "Accuracy: 93.88 %"
```

The model made 4 errors out of 49 predictions: 2 false negatives and 2 false positives. This suggests that while the model performs well, there are occasional misclassifications, which could be due to overlap in the predictor values between certain beverage categories.

An accuracy of 91.84% indicates that the LDA model is very effective in classifying the beverage categories based on the given predictors. This high accuracy suggests that the nutritional and compositional variables we included are strong discriminators of the different beverage types.

Given the high accuracy, this LDA model can be reliably used for classifying new beverage entries in our dataset. It can help automate the categorization process, ensuring consistent and accurate classification based on the nutritional content. This model can assist in various aspects such as inventory management, nutritional labeling, and marketing strategies by accurately categorizing beverages into predefined categories based on their nutritional attributes.

Overall:

Classification Power: The LDA model effectively uses the nutritional and compositional variables to classify beverages into their respective categories. The first few linear discriminants (especially LD1 and LD2) capture most of the variability, indicating that the model is likely to be quite effective in distinguishing between beverage types.

Important Predictors: Total Fat, Saturated Fat, and Sugars are particularly important in differentiating between beverage categories. This suggests that these nutritional factors are key characteristics that define different types of beverages in your dataset.

Group Means Analysis: By examining the group means, you can see which beverage types are higher or lower in certain nutrients. For instance, Smoothies have the highest calories and protein, whereas Coffee has the lowest in both.

Practical Application: This LDA model can now be used to classify new beverage entries based on their numeric attributes. It can help in identifying which category a new beverage falls into, which is useful for inventory management, nutritional analysis, and consumer information.

7 Conclusion

During our analysis, we employed various regression and classification methods to gain a better understanding of our dataset and to obtain results that are applicable in the real world. Our results highlighted that the best models are the Lasso regression model for regression tasks and Logistic regression for classification tasks, with accuracies of, respectively, 0.9975756 and 0.9183673.

Given these results, we propose using these models as tools for companies developing new beverages for the market. Such a tool is particularly useful for predicting the calorie content of their products. This is especially pertinent in the United States, where obesity and eating disorders are widespread. A tool like this could contribute to the creation of healthier products, thereby helping to address these issues.

Since we intend to provide this tool to an external company, we opted to design it to be as straightforward as possible to facilitate easy interpretation of the results.