Statistical Learning Final Report

Alberto Calabrese, Eleonora Mesaglio, Greta d'Amore Grelli

2024-06-10

Contents

1 Introduction

Here Eleonora you can write the introduction of the project describing the scope and the data used.

Thank you Albi, I will. What is our project scope though?

I think that we have to analyze the dataset and perform some statistical analysis on it. We can start by calculating the correlation matrix and then we can visualize the data through histograms, pairplots, barplots and boxplots. Finally, we can perform a regression analysis.

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.

In the upcoming code lines, we import the dataset and generate a summary visualization. This initial step allows us to gain a better understanding of the data structure and the variables involved.

```
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.

```
# 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 some 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.

```
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
```

```
# 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)
```

```r
# Summary of the Caffeine column after cleaning
summary(data_cleaned$Caffeine..mg.)
```

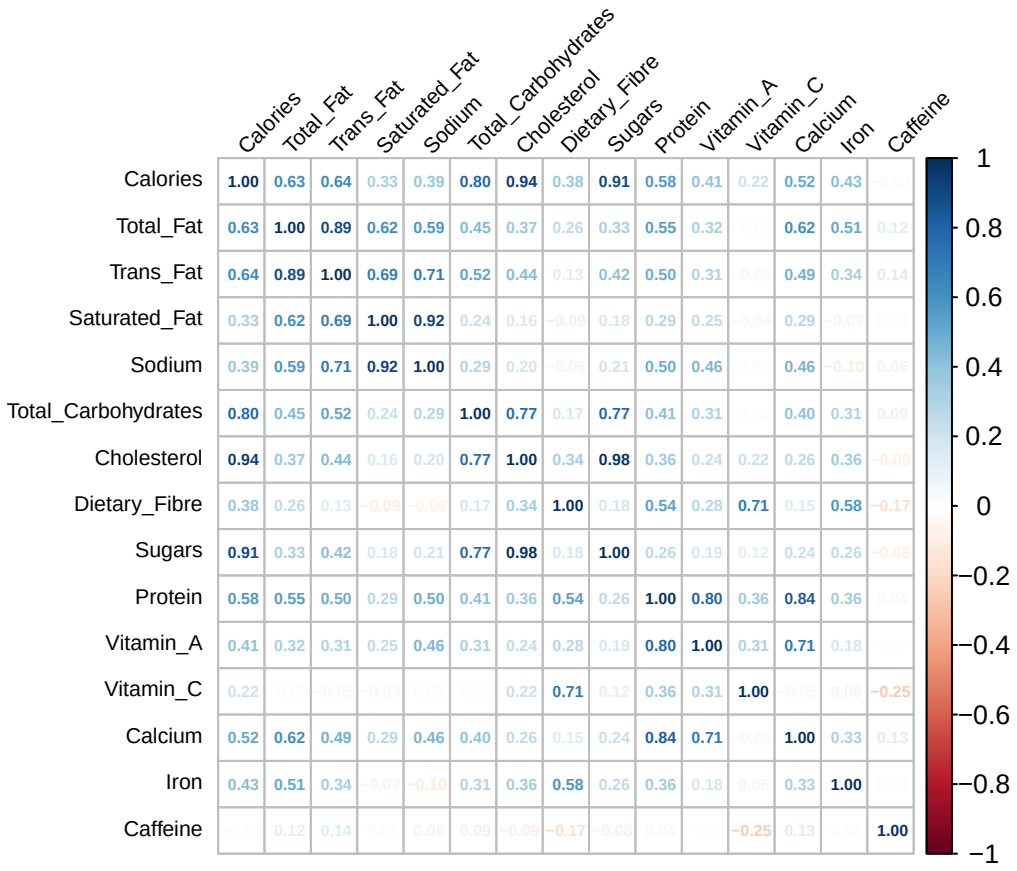
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   70.00   75.00   88.14  130.00  410.00
```

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.



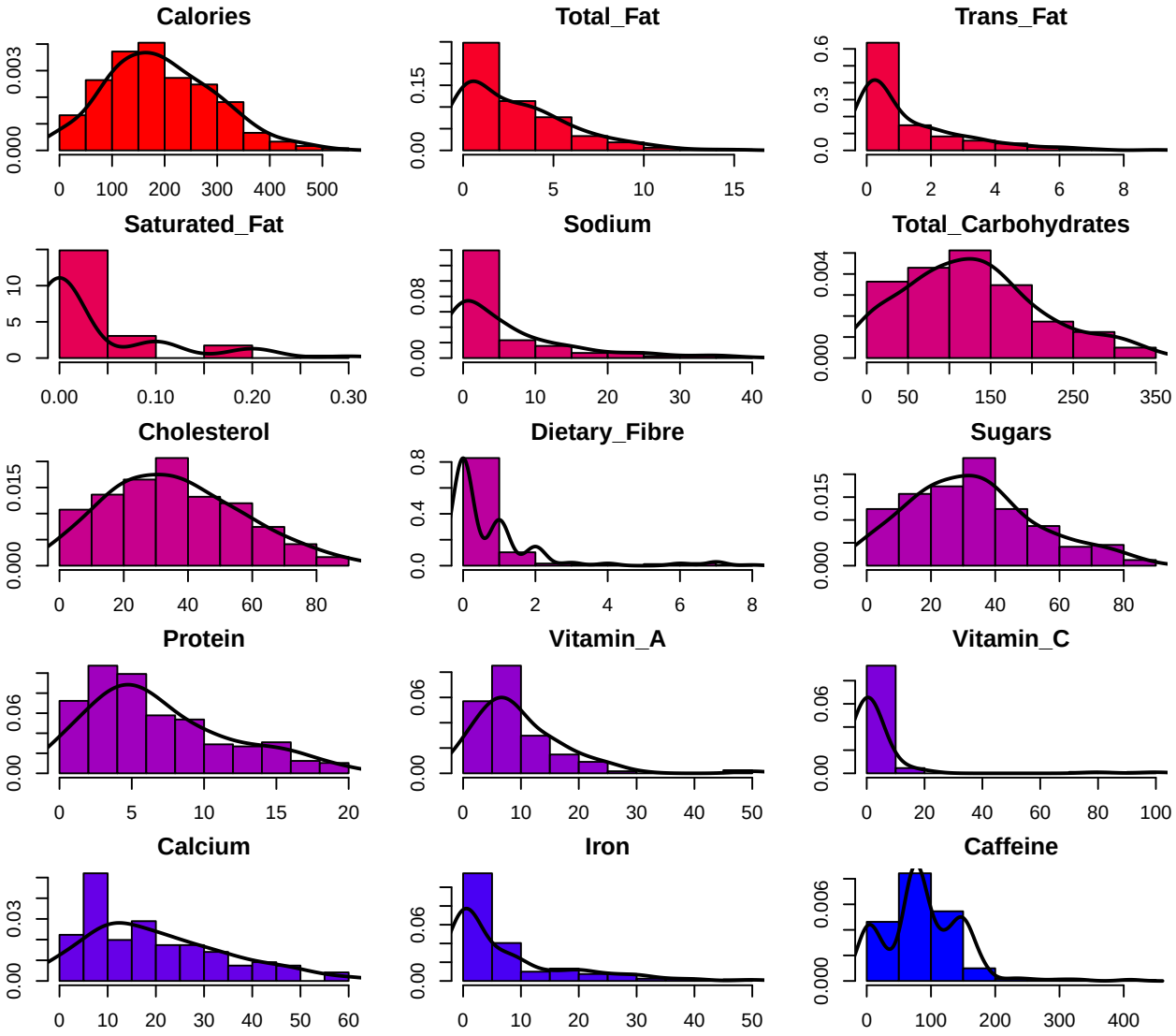
4 Data Visualization

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.

4.2 Pairplot

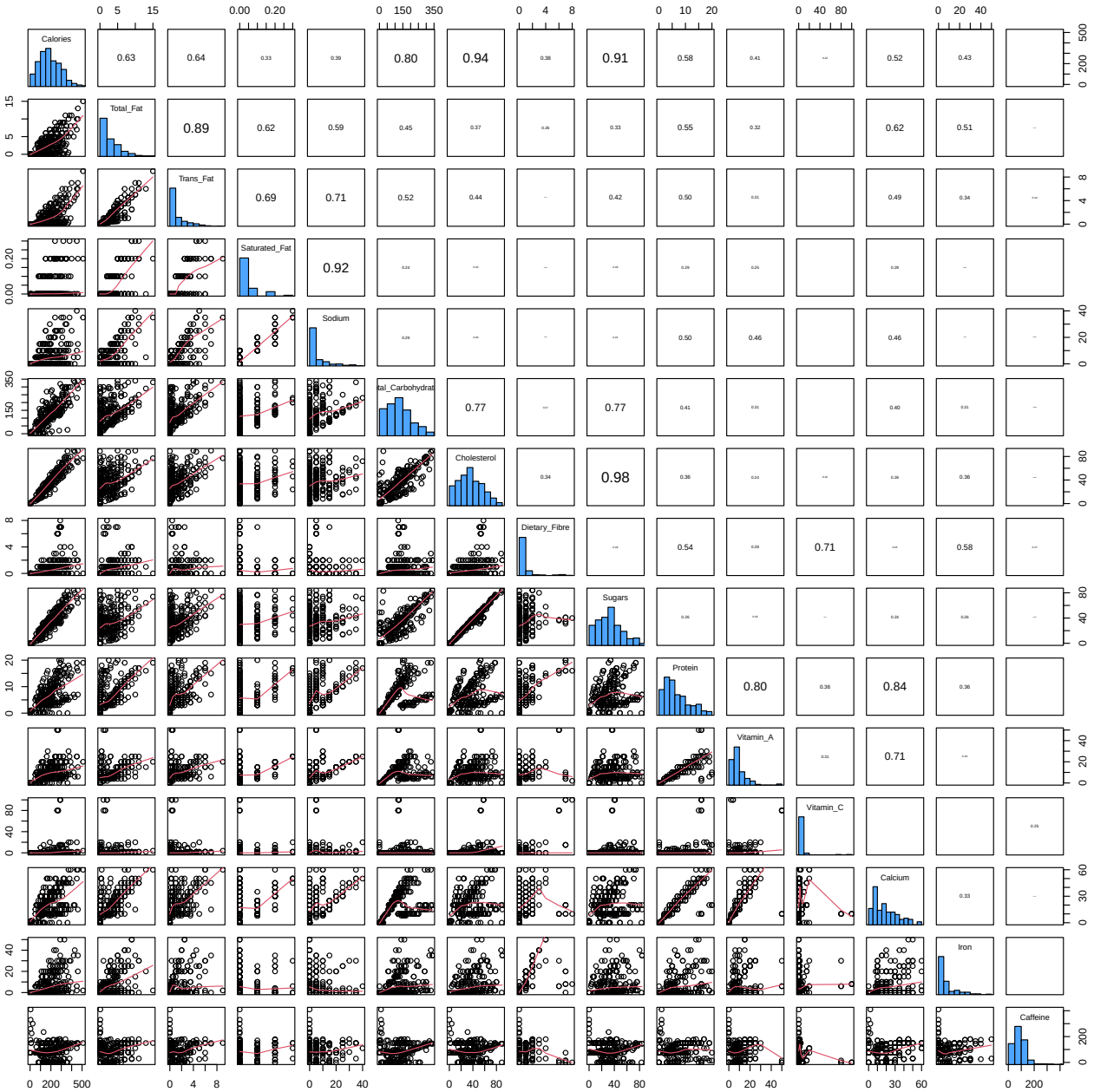
io questi li toglierei. tanto spazio

We will plot a pairplot to visualize the relationship between the variables. The pairplot is a grid of scatterplots that shows the relationship between each pair of variables in the dataset. This visualization helps us to

identify patterns and correlations between the variables.

First of all we have to define the function for the pairplot. We will define a function for the histogram, the correlation and the smooth line.

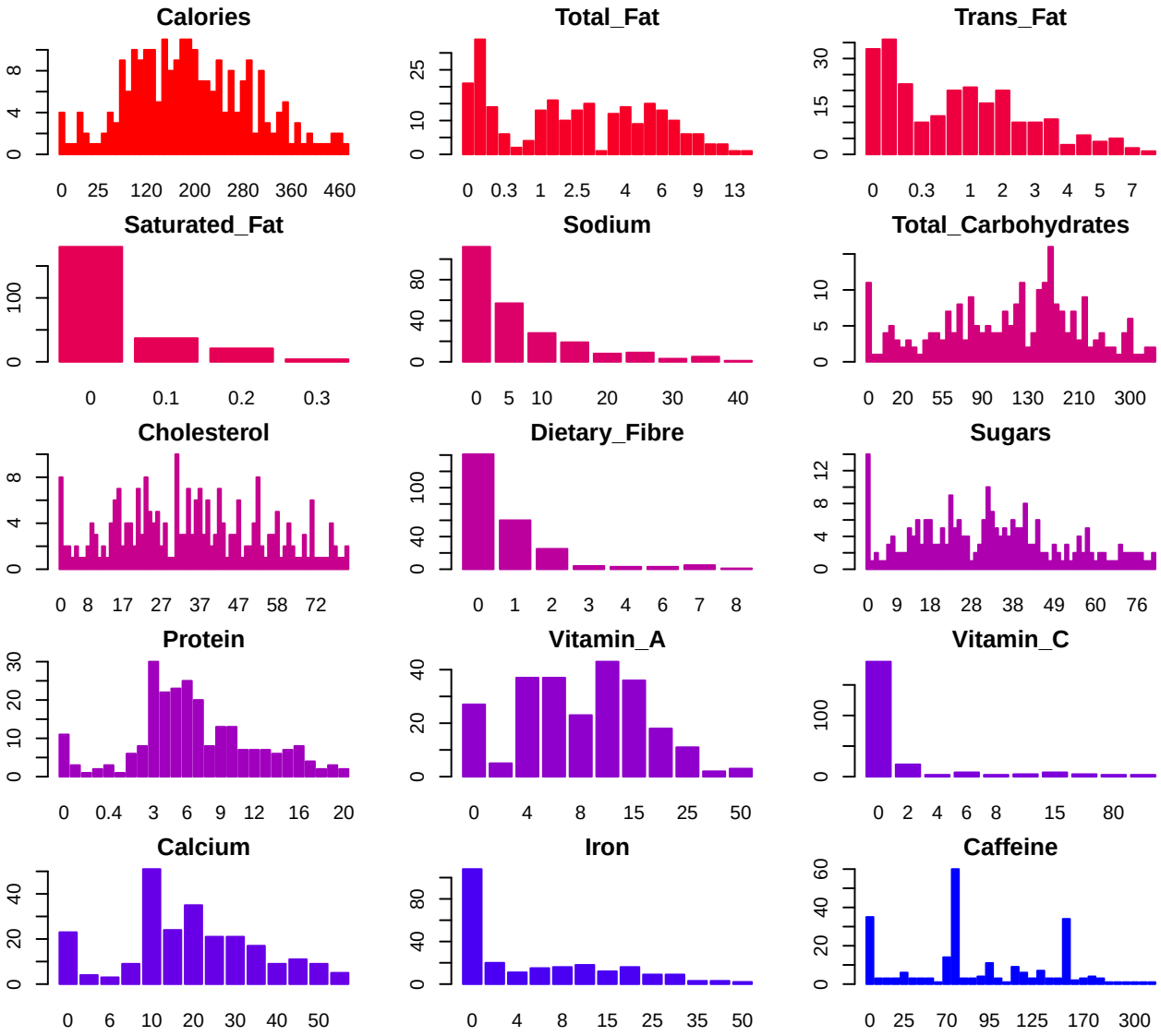
Then we create the pairplot using the defined functions.



ADD COMMENTS ON THE GRAPH

4.3 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. These graphs are commonly used for categorical data, or numerical data that has been binned into categories.



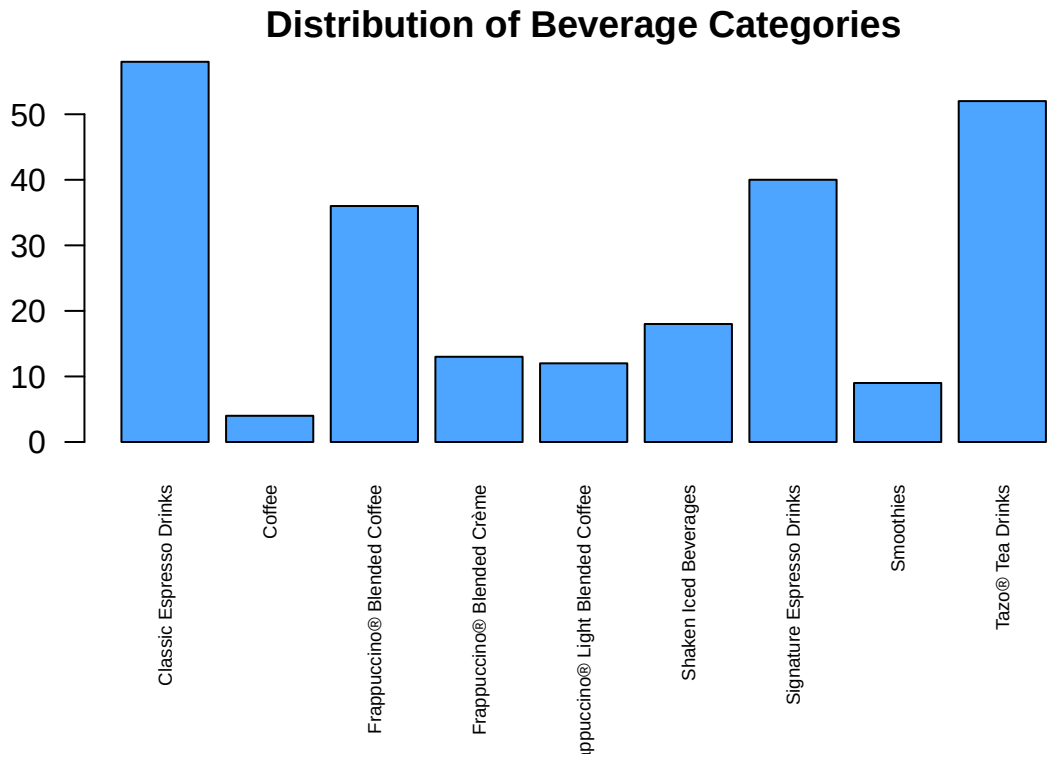
We can deduce some useful information by looking at these plots.

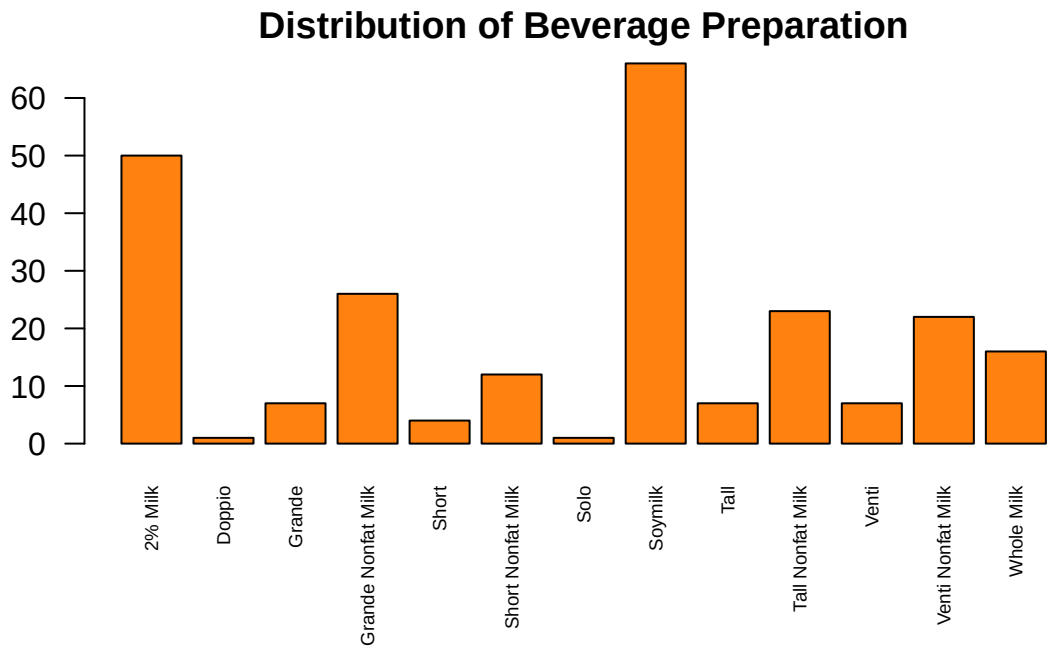
For example, we can notice that variables such as "Saturated_Fat", "Dietary_Fibre", "Vitamin_C", and "Iron" are typically either absent or present in small quantities in the beverages. In particular, the frequency of these variables rapidly diminishes as their levels increase. On the other hand, the 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.

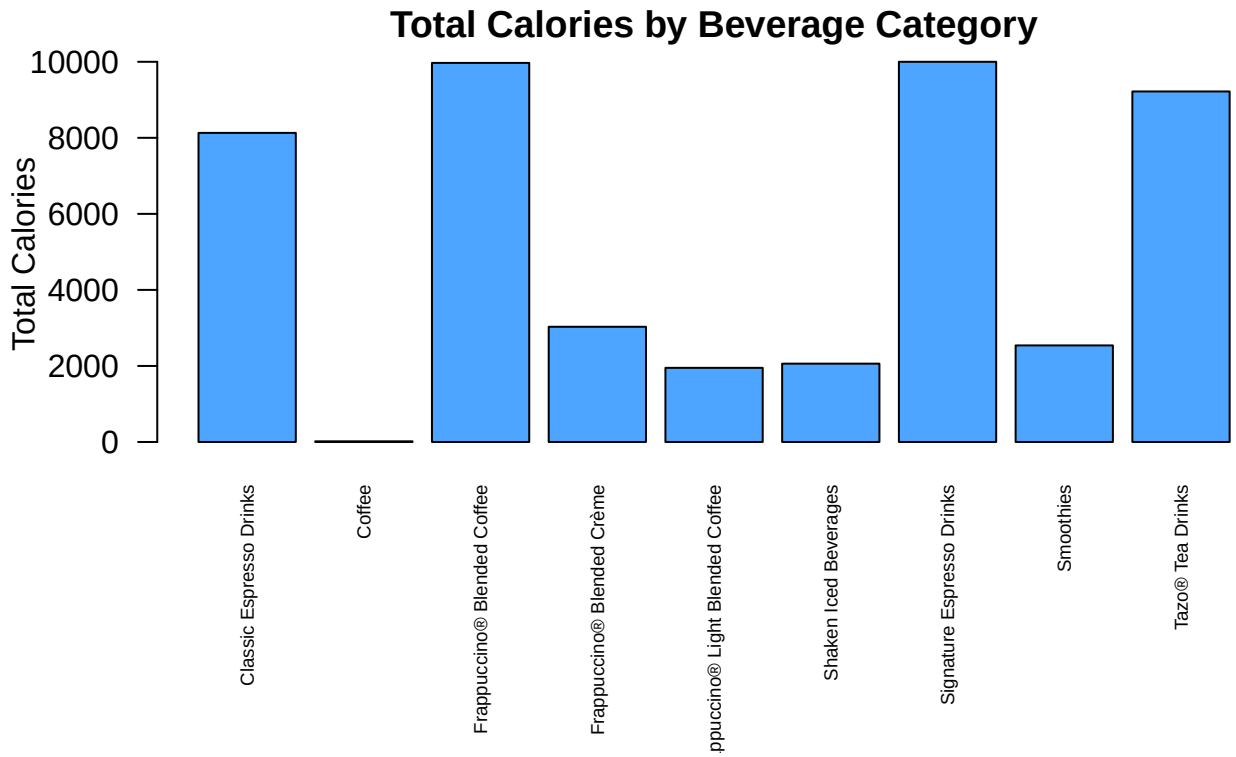
4.3.1 Beverages Barplot

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

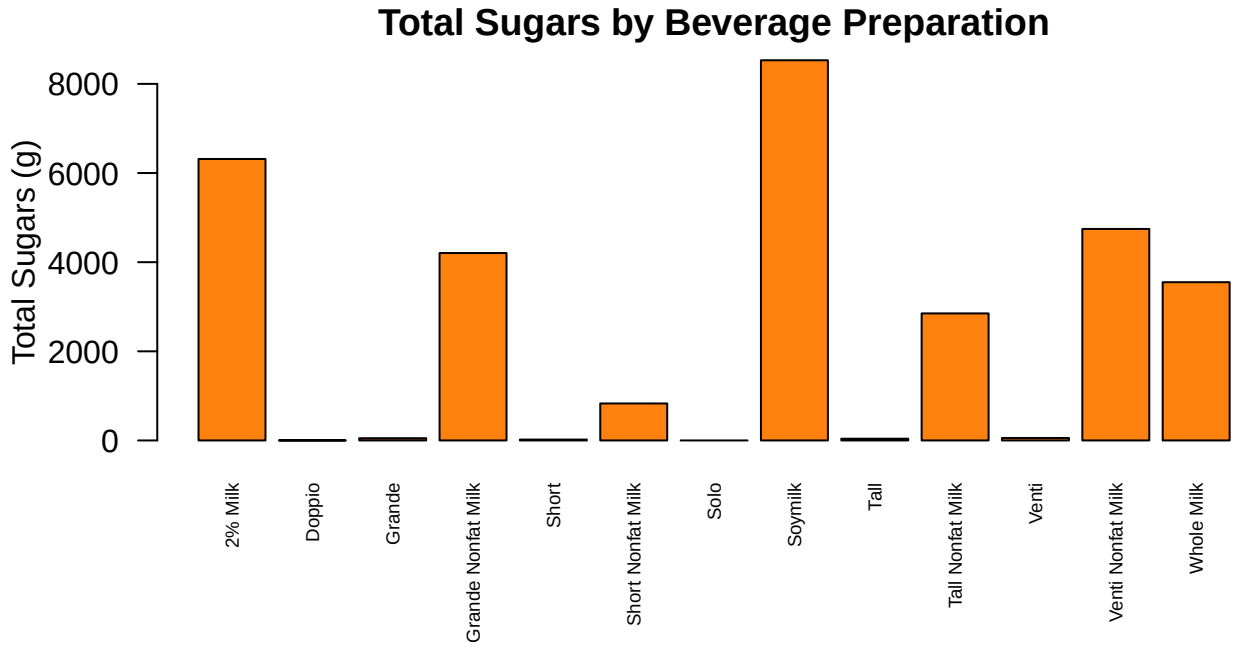




At this point, we aim to compare the total calorie content among different beverage categories. To do so, we first aggregate the data to obtain the total calories for each beverage category. Secondly, we construct a bar plot to visually represent the results.

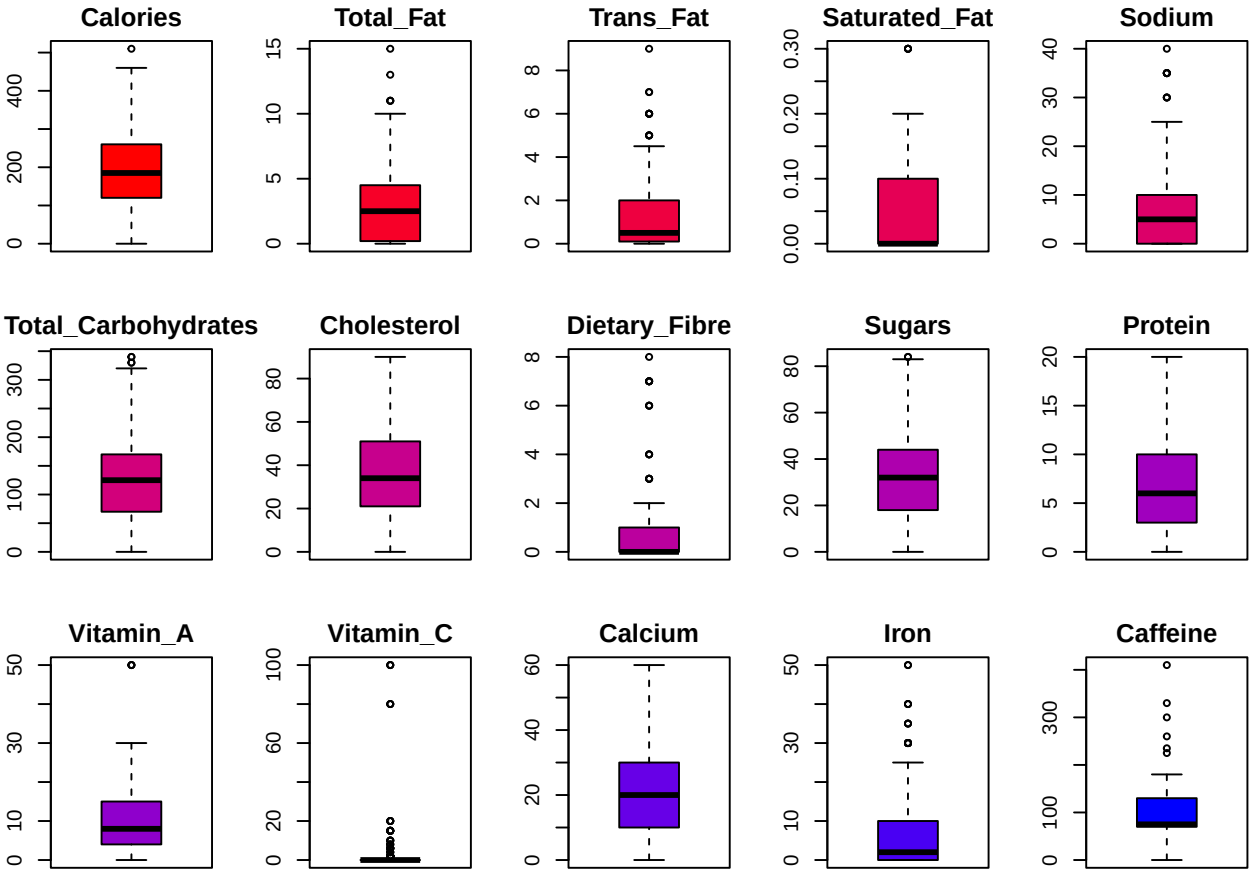


Similarly, we compare the total sugars for each beverage preparation, gathering data to obtain the total sugars for each preparation of beverage and successively creating a bar plot.



4.4 Boxplot

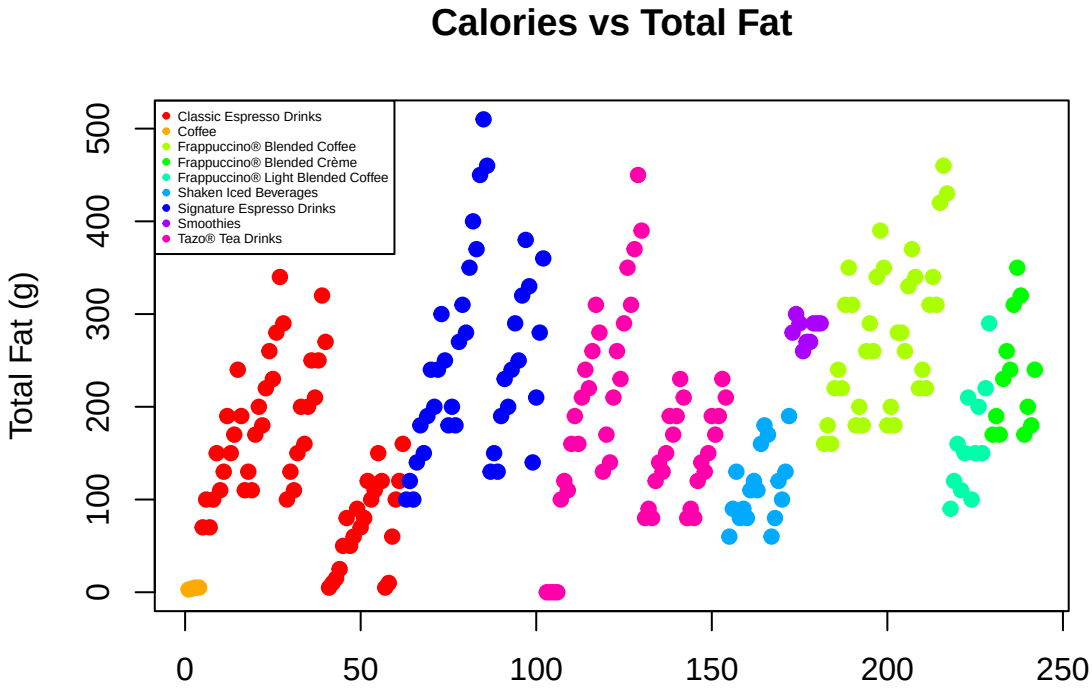
We will plot a boxplot of the data. The boxplot is a graphical representation of the data that displays the distribution of the data, including the median, quartiles, and outliers. This visualization helps us to identify the spread and variability of the data.

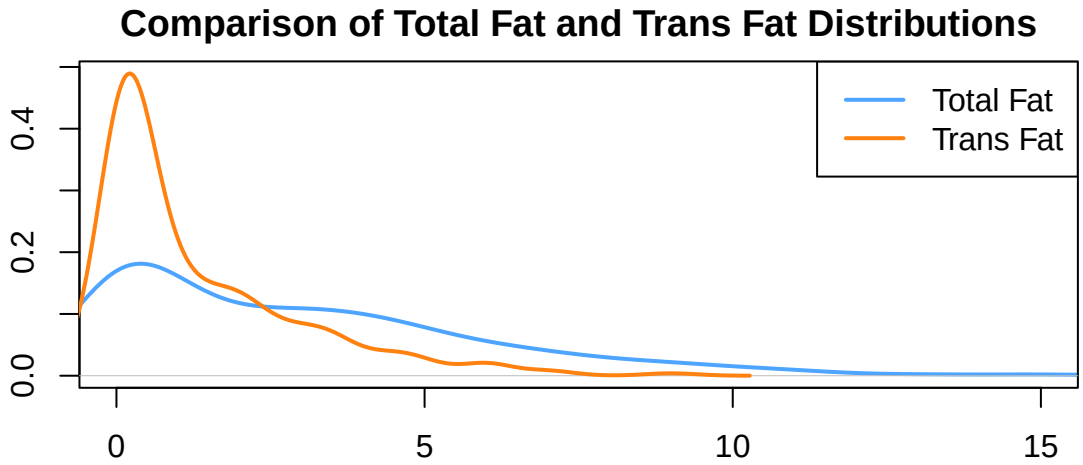


4.5 Scatterplot

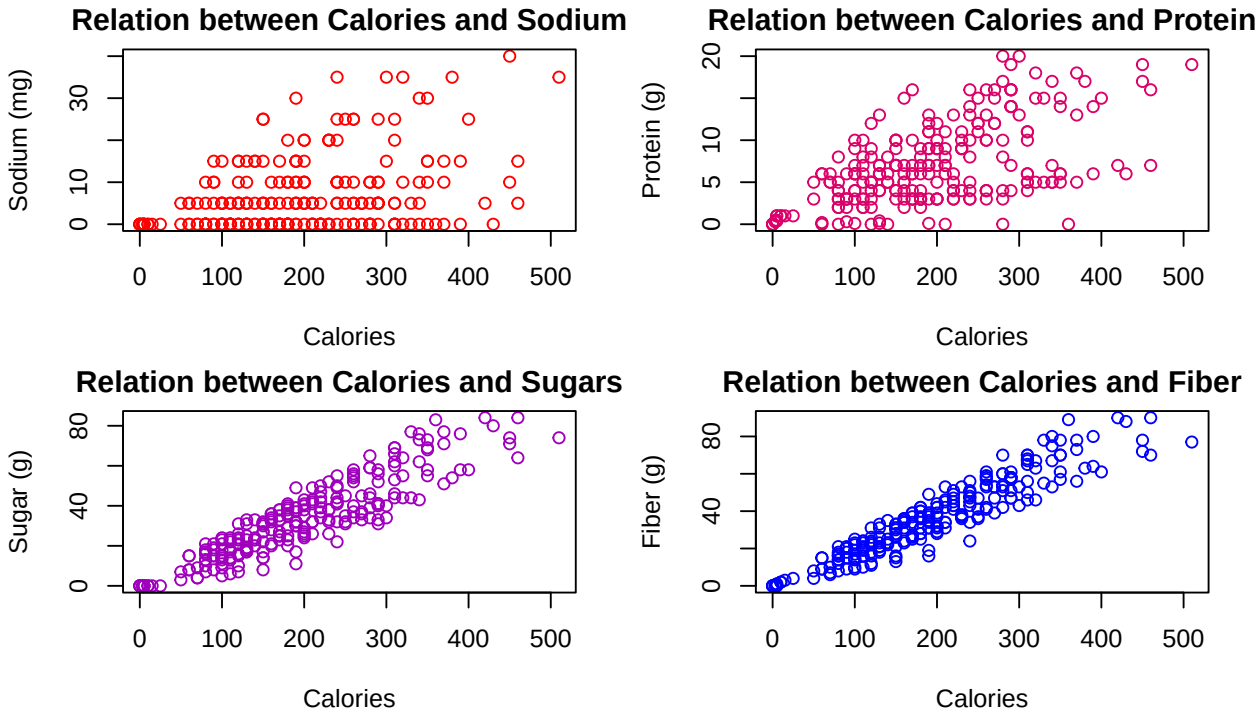
We will plot a scatterplot of the data. The scatterplot is a graphical representation of the data that displays the relationship between two variables. This visualization helps us to identify patterns and correlations between the variables.

We create a scatterplot to compare the amounts of calories and fat for each categories of bevarage. We assign distinct colors to each beverage category and create a legend to identify each category.





Create scatterplot to look into relantionship between calories and other variables. We will plot the relationship between calories and sodium, protein, vitamin C and fiber.



TO CHANGE THIS SINCE I CHANGED ONE OF THE CATEGORIES WITH SUGAR
There's increase in every feature with increase in calories. Features like proteins and fiber rapidly increase, instead vitamin and cholesterol more flat growing. Confirmed by correlation coefficients

ADD COMMENTS ON THE GRAPH

5 Regression Analysis

5.1 Linear Regression

Linear regression model to predict the amount of calories based on the amount of the other variables We use the lm() function to fit a linear regression model

5.1.1 Simple Linear Regression

Fit linear simple regression with just one variable on data_cleaned, looking at correlation plot we choose Sugars due to high correlation.

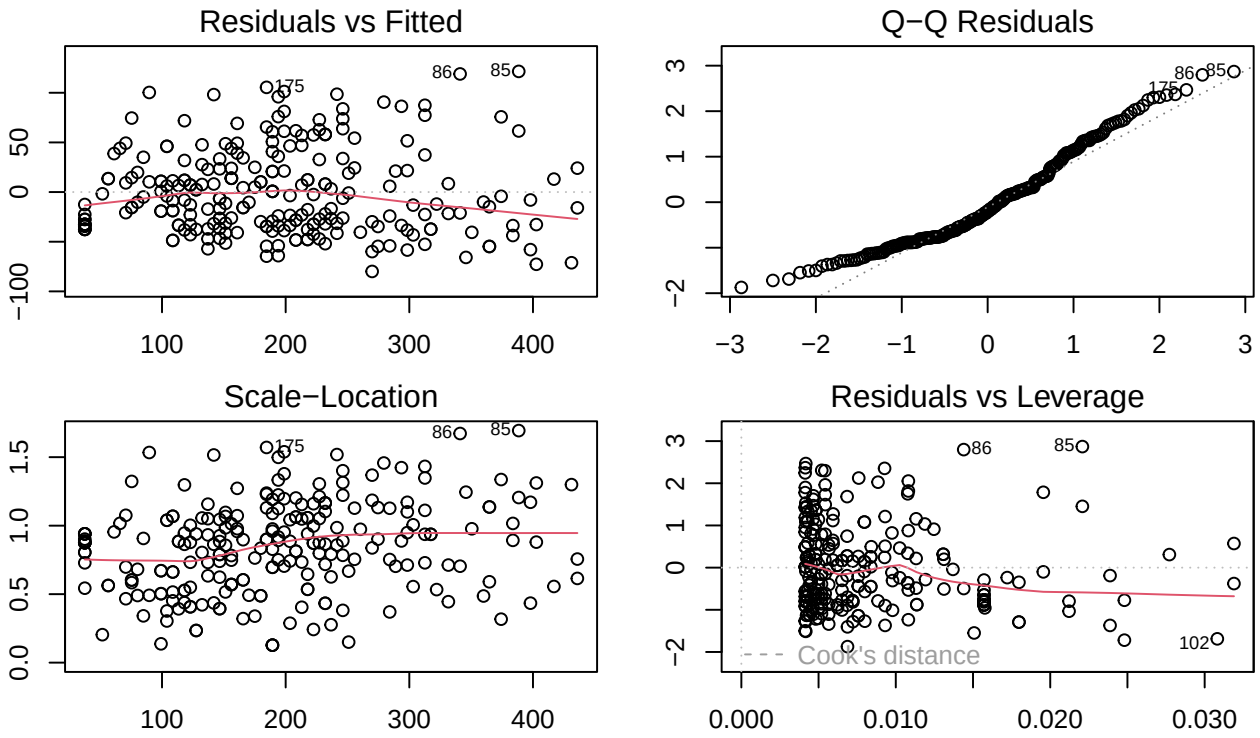
This code will fit a simple linear regression model predicting "Calories" using "Sugars" as the predictor variable and provide a summary of the model.

```
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

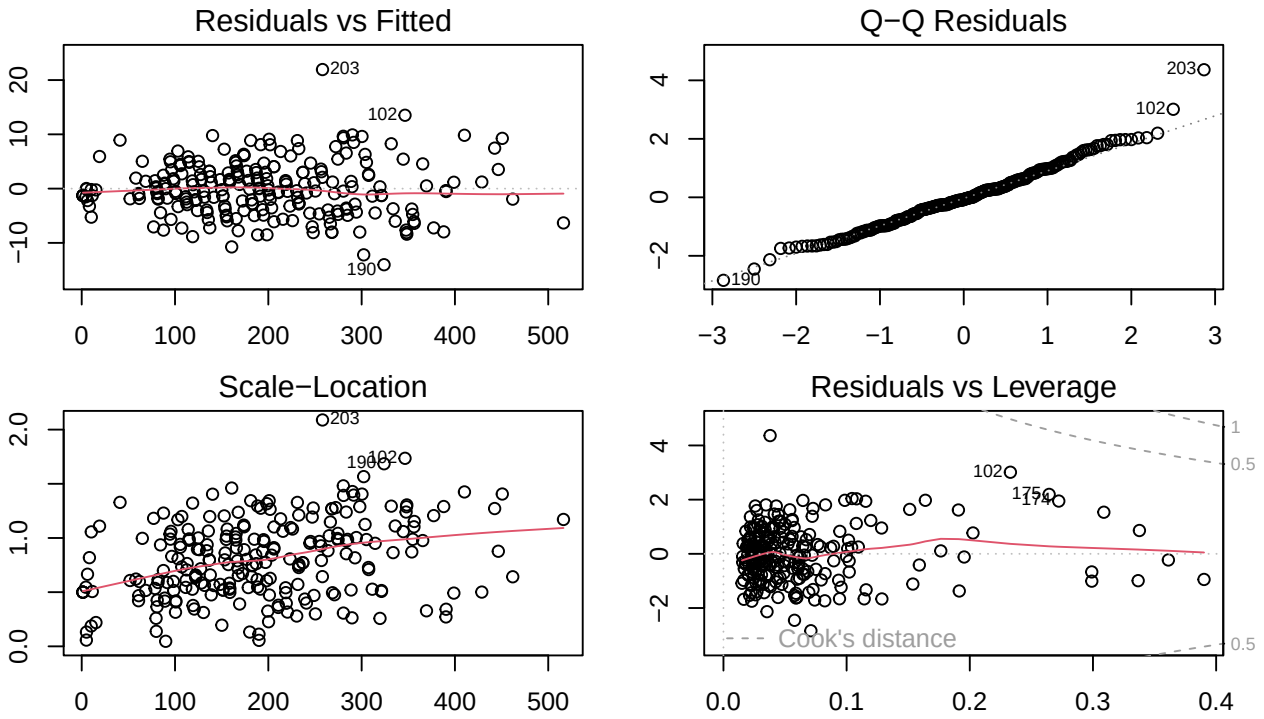
```r
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
plot(lm_simple)
```



The coefficient for "Sugars" (4.7426) indicates that, on average, for every one-unit increase in "Sugars", the predicted "Calories" increases by approximately 4.7426 units. Both the intercept and the coefficient for "Sugars" 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". Model Fit:The adjusted R-squared value (0.8268) indicates that approximately 82.68 of the variance in "Calories" can be explained by the predictor variable "Sugars". Overall, this output suggests that the simple linear regression model provides a statistically significant relationship between "Sugars" and "Calories", with "Sugars" being a strong predictor of "Calories". However, the AIC and BIC values suggest that there might be other models that provide a better fit for the data. Summarizing the model is too simple so it doesn't capture the complexity of the data, so we try to fit a multiple linear regression model.

5.1.2 Multiple Linear Regression

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




```
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

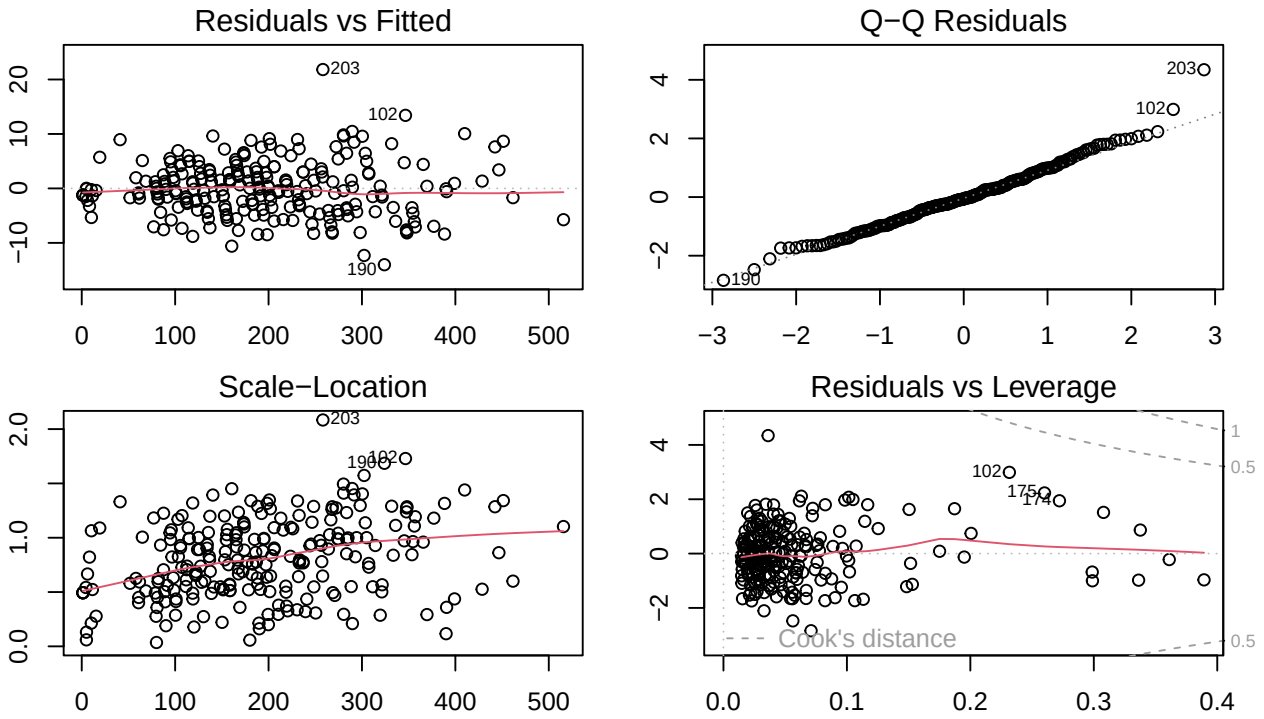
AIC	BIC	R_squared	adj_R_squared
1494.304	1550.127	0.9976608	0.9975166

The model has a low AIC and BIC values, the R-squared value is 0.997 so the model is a good fit for the data.

5.1.3 Backward Elimination

Now we apply the selection of the predictors with the backward elimination method.

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




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

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

The backward selection drops only the variable "Saturated_Fat" since it's not considered significant in explaining the amount of calories mantaining the other variables.

Comarison 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: Both 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.

Significance of Variables: 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 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

Anova comparison between the models

```
anova_results <- anova(lm_model, backward_model)
anova_results
```

```
## 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.

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.

Conclusion: 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

To check for multicollinearity, we calculate the Variance Inflation Factors (VIF) for the variables in the multiple linear regression model, it 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

	VIF
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

However as we can see from the *Table X* e have a problem with multicollinearity, the VIF values are high for some variables, so 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,means that variable contribuite in the same way to predict and explain calories
- Same information
- Data unbalanced
- 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.

In this case, normalizing the variables might help reduce multicollinearity.

5.1.6 Standardize the data

We have tried different kind of standardization to reduce the multicollinearity First at all we tried the scale by standard normalization 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. This is because standardization helps to put all variables on the same scale, reducing the likelihood of multicollinearity. Log transformation: Reduces the variance of the variables, making the distribution more normal and reducing the impact of outliers. Standardization: Puts all variables on a common scale, with mean 0 and standard deviation 1, further reducing multicollinearity.

```
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

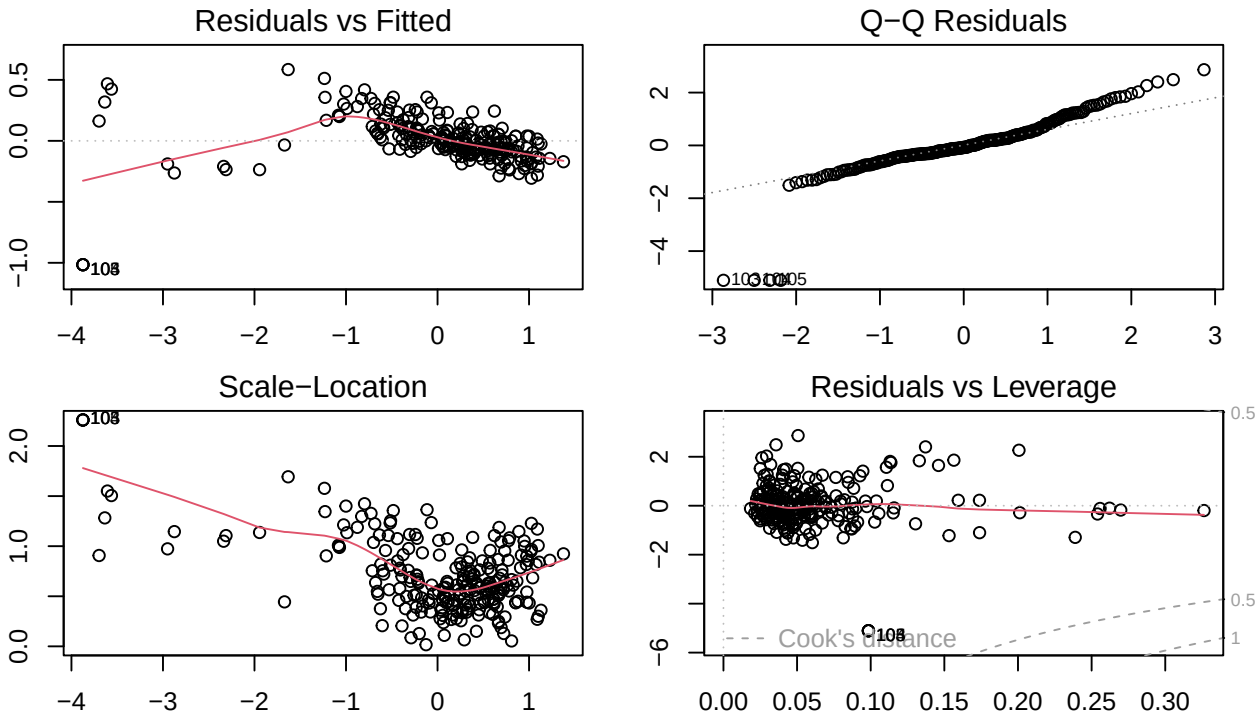
AIC	BIC	R_squared	adj_R_squared
-53.42411	2.398897	0.9586932	0.9561457

```
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

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



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. However we have still collinearity, so we try to use backward elimination to check if this method will removes the variables that are not significant in the model.

```
kable(data.frame(AIC = AIC(backward_model_log), BIC = BIC(backward_model_log),
                 R_squared = summary(backward_model_log)$r.squared,
                 adj_R_squared = summary(backward_model_log)$adj.r.squared),
      caption = "Model evaluation metrics for the log transformed data
      with backward elimination")
```

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

```
kable(data.frame(VIF = vif(backward_model_log)),
      caption = "VIF values for the log transformed data with backward elimination")
```

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

AIC slightly worst and the VIF values are still high, indicating that multicollinearity is still present in the model. So we try to remove manually the variables that has high VIF values.

```
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

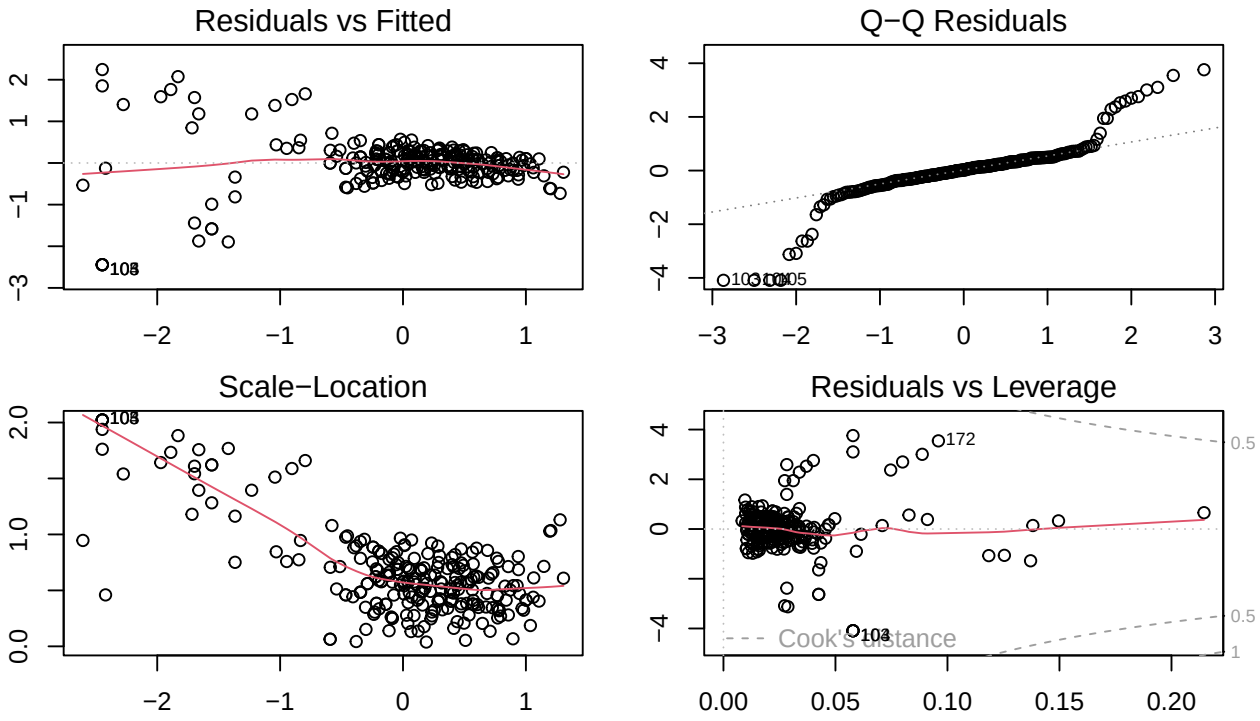
	VIF
Total_Fat	11.902918
Trans_Fat	10.114112
Saturated_Fat	4.466975
Sodium	5.782843
Total_Carbohydrates	3.375194
Dietary_Fibre	7.080360
Protein	26.902392
Vitamin_A	12.396739
Vitamin_C	1.985799
Calcium	25.519022
Iron	4.521552
Caffeine	1.295121

```
kable(data.frame(AIC = AIC(mod_log_tr_backward_2), BIC = BIC(mod_log_tr_backward_2),
                 R_squared = summary(mod_log_tr_backward_2)$r.squared,
                 adj_R_squared = summary(mod_log_tr_backward_2)$adj.r.squared),
      caption = "Model evaluation metrics for the log transformed data with
      backward elimination and manual removal of variables")
```

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

```
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
plot(mod_log_tr_backward_2)
```




```
kable(data.frame(VIF = vif(mod_log_tr_backward_2)),
      caption = "VIF values for the log transformed data with backward
      elimination and manual removal of variables")
```

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

The VIF values are now below 10, indicating that multicollinearity has been reduced in the model. The R-squared value is decreased but it is still good.

Model Diagnostics: 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 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. Sol Robust Methods: Use robust regression methods that do not assume normality of errors

We have tried other trasformation like min-max scaling and robust scaling but not satisfactory due to VIF still to high. Regularization: Using regularization methods such as ridge regression or lasso regression penalizes the coefficients of variables, helping to reduce multicollinearity.

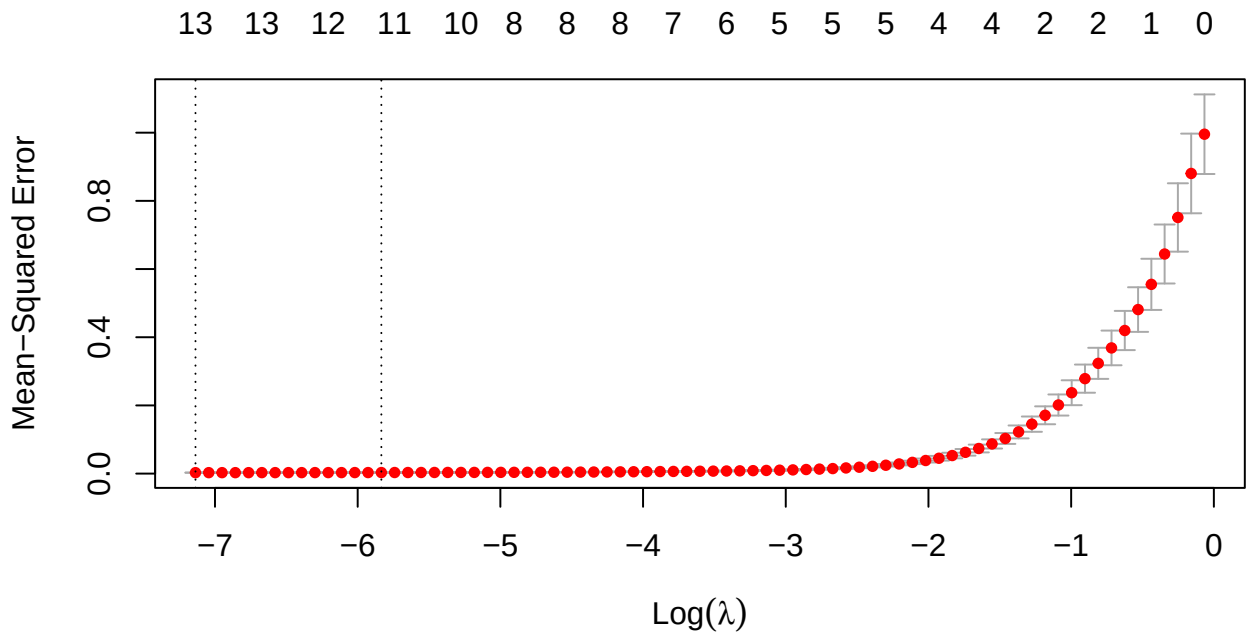
5.2 Lasso Regression

We use the glmnet package to fit a lasso regression model. Lasso regression is a type of linear regression that uses L1 regularization to penalize the coefficients of the model. This helps to prevent overfitting and select the most important features in the data.

First we standardize the data and then we fit the lasso regression model using the cv.glmnet() function. We use cross-validation to select the optimal lambda value for the model. The lambda value that minimizes the mean squared error (MSE) is selected as the optimal lambda value. The optimal lambda value is 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 from the model, it reduces the number of predictors and thereby reduces multicollinearity. Lasso tends to produce sparse solutions, meaning it drives many coefficients to exactly zero. When variables are removed from the model, the multicollinearity among predictors decreases, leading to lower VIF values. It performs automatic features selection by shrinking some coefficients to zero. This feature selection process inherently 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)
```

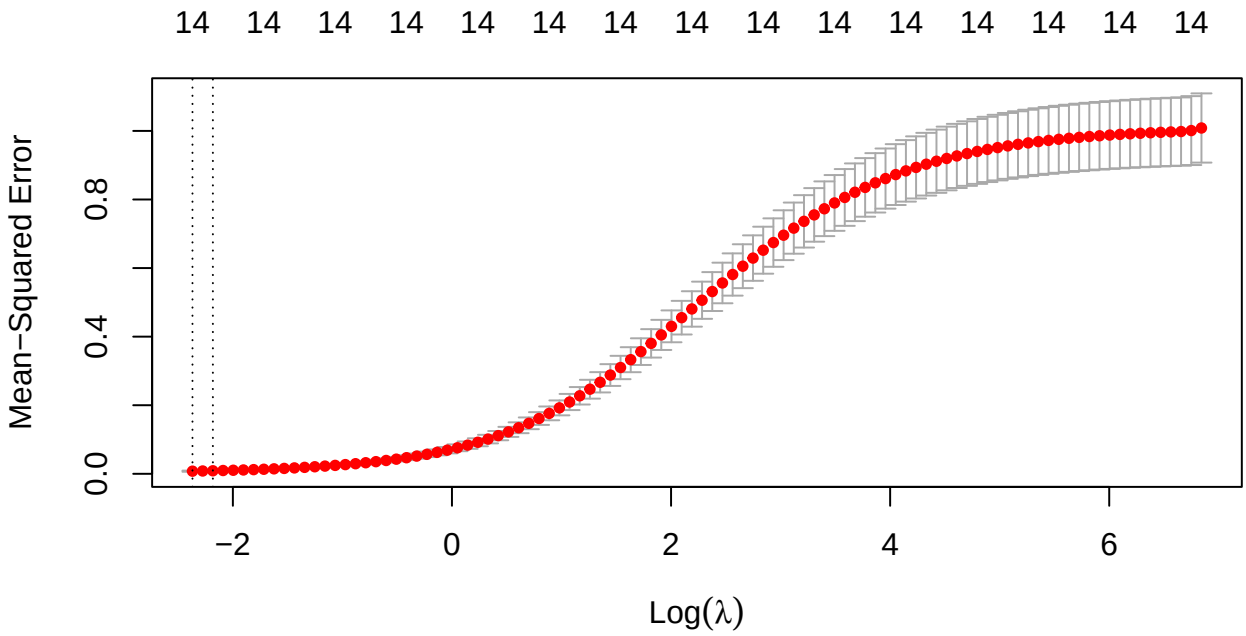


The lasso regression model selects the most important features in the data and penalizes the coefficients of the model. The model has a low AIC and BIC values, the R-squared value is 0.99 so the model is a good fit for the data.

5.3 Ridge Regression

We use the glmnet package to fit a ridge regression model. Ridge regression is a type of linear regression that uses L2 regularization to penalize the coefficients of the model. This helps to prevent overfitting and reduce the impact of collinearity in the data.

```
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)
```



The ridge regression model reduces the impact of collinearity in the data and penalizes the coefficients of the model. The model has a low AIC and BIC values, the R-squared value is 0.99 so the model is a good fit for the data.

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.

The R-squared value is a measure of how well the model fits the data, it ranges from 0 to 1, with higher values indicating a better fit

```
lasso_pred <- predict(mod_lasso, s = "lambda.min", newx = as.matrix(std_data[, -1]))
lasso_r_squared <- cor(lasso_pred, std_data$Calories)^2
ridge_pred <- predict(mod_ridge, s = "lambda.min",  newx = as.matrix(std_data[, -1]))
ridge_r_squared <- cor(ridge_pred, std_data$Calories)^2
kable(data.frame(Model = c("Linear Regression", "Lasso Regression","Ridge Regression"),
                 R_squared = c(summary(lm_model)$r.squared,
                               lasso_r_squared, ridge_r_squared)),
      caption = "R-squared values for the models")
```

Table 13: R-squared values for the models

Model	R_squared
Linear Regression	0.9976608
Lasso Regression	0.9975756
Ridge Regression	0.9941815

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.

```
linear_pred <- predict(lm_model, newdata = data_num)
linear_mse <- mean((linear_pred - data_num$Calories)^2)
lasso_mse <- mean((lasso_pred - std_data$Calories)^2)
ridge_mse <- mean((ridge_pred - std_data$Calories)^2)
kable(data.frame(Model = c("Linear Regression", "Lasso Regression", "Ridge Regression"),
                 MSE = c(linear_mse, lasso_mse, ridge_mse)),
      caption = "MSE values for the models")
```

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 lowest value for R^2 and MSE and it is the most robust model.

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. This process is repeated multiple times to ensure that the model is robust and generalizes well to new data.

We split the data into training and testing sets, fit the lasso regression model using the training set.

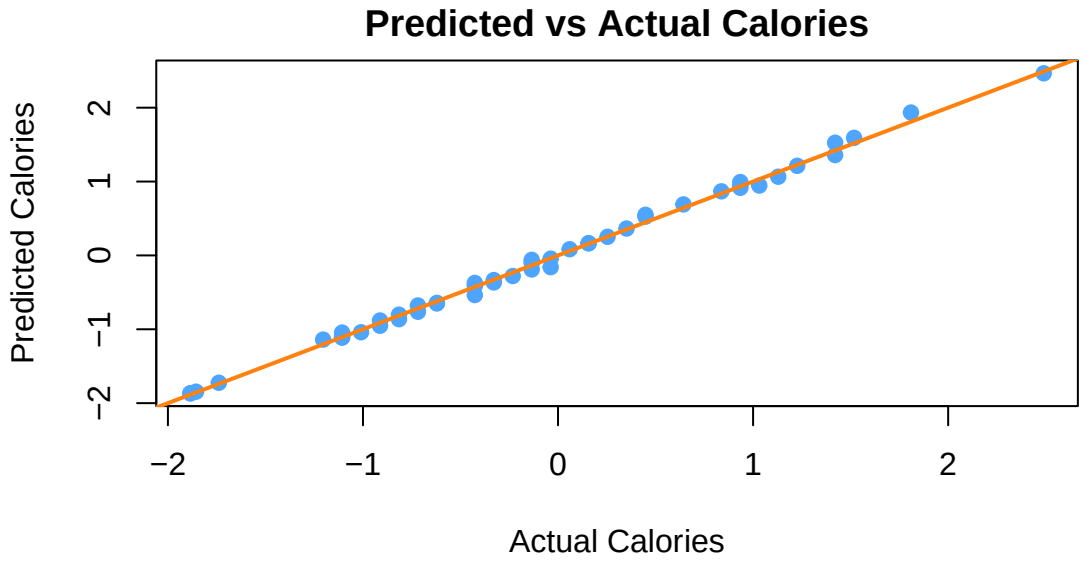
```
train_index <- sample(1:nrow(std_data), 0.8 * nrow(std_data))
train_data <- std_data[train_index, ]
test_data <- std_data[-train_index, ]
mod_lasso_train <- cv.glmnet(x = as.matrix(train_data[, -1]), y = train_data$Calories,
                             alpha = 1, standardize = FALSE)
```

We evaluate the model using the testing set. We make predictions using the testing set and calculate the mean squared error and the root mean squared error to assess the model's accuracy.

```
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)
```

The R-squared value and MSE are used to evaluate the performance of the model on the test data.

```
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 = "#4ea5ff",
     ylab = "Predicted Calories", main = "Predicted vs Actual Calories", pch = 19)
abline(0, 1, col = "#ff810f", lwd = 2)
```




```
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	0.9973488	0.0027738	0.9973759

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, indicating that the model is able to predict the amount of calories with high accuracy. The plot shows 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.