MTH783P Final Project

Arjun Bayadegere Prabhanna, (230850895)

19 May, 2024

Abstract

This project delves into the realm of time series analysis and forecasting techniques using a dataset comprising average unit prices of avocados across diverse U.S. cities from January 2015 to March 2018. The initial stage of the project's multi-phase execution divides the dataset into subsets for testing and training. The next step is a comprehensive exploratory study that uses summary statistics, visualisation, and outlier detection to explain noteworthy data features and any abnormalities. After the dataset has undergone extensive preprocessing (Box-Cox transformation for variance stabilisation, trend elimination, seasonality, stationarity, autocorrelation tests, etc.), ACF/PACF charts help choose the optimal model. After that, the optimal model is carefully selected, validated by the AIC criterion, and verified using residual analysis to make sure the model is suitable. I ultimately selected the forecast model after calculating the forecasting error using a testing dataset. In the end, the approach provides information about avocado price prediction, which enhances avocado industry decision-making.

Introduction

The avocado market is a dynamic industry where prices are determined by market trends and consumer demand. This study focuses on the average avocado price across US cities from January 2015 to March 2018, using time series analysis and forecasting methodologies. Seasonality and regional variations are two of the many factors that affect avocado pricing. Our goals are to create precise forecasting models and investigate pricing trends through exploratory analysis.

Data Preprocessing and Exploratory Data Analysis

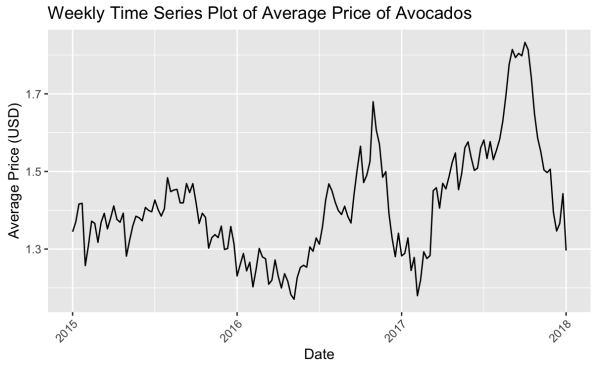
The following steps are involved in processing the training dataset and running an exploratory analysis on it:

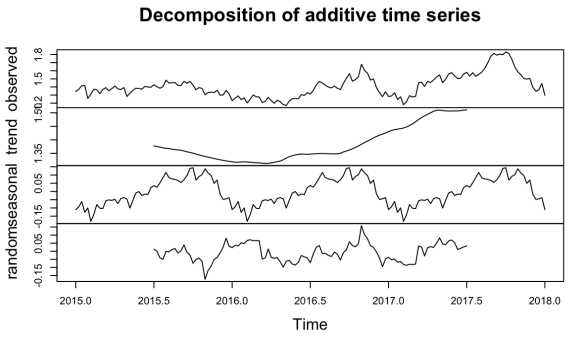
- ***Data Sorting and splitting* -** For the purpose of facilitating time series analysis, the dataset is organised logically and with care to ensure that the Date column has a consistent date format. Spliting the dataset into training and testing subsets to facilitate model evaluation.
- ***Handling Missing Values* -** Applying the mean imputation approach to resolve any missing or NaN values found in the dataset, and then verifying for missing values to confirm the imputation process worked as intended.
- ***Outlier Detection and Handling* -** Employing box plots and other pertinent plots to visually represent the outliers. Afterwards, putting procedures in place to deal with outliers, such as eliminating them or altering the data.
- ***Summary Statistics* -** Computed summary statistics for numerical variables in order to comprehend their distributions, including measures of central tendency and dispersions.
- ***Correlation Analysis* -** Used different plots, like scatterplots, pairwise plots, bar plots, etc., to analyse the correlation between data in order to find potential correlations or dependencies.

Methodologies

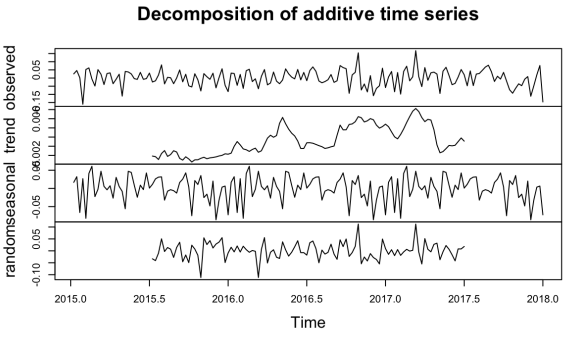
The following steps explain the time series analysis

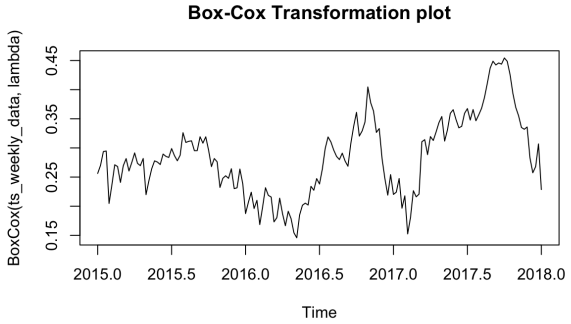
1. ***Visualising the Time Series* -** In order to detect any abnormal structure, the training dataset is first transformed into a time series format and visualised to observe trend, seasonality and cyclic patterns.



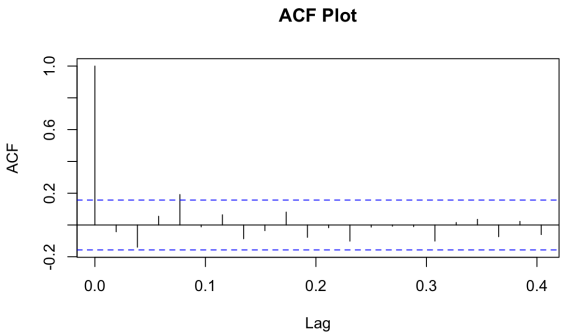


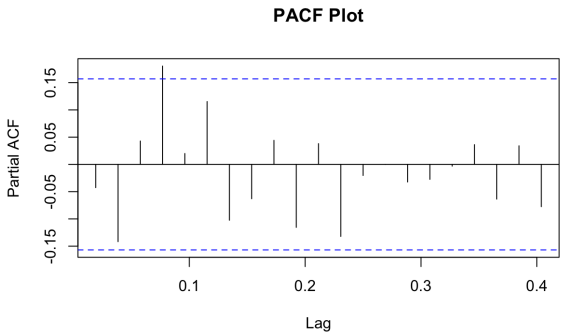
2. ***Stationarising the Time Series -*** By calculating first order differences, the time series is stationarized by eliminating trend and seasonality. To stabilize the variance, the Box-Cox transformation is also used.





3. ***ACF/PACF plots to find optimal parameters -*** In the ACF plot of the differenced time series, significant autocorrelation at a specific lag indicates that the observations in the time series are correlated with their immediate previous values. In the PACF Plot, the significant partial autocorrelation at a specific lag suggests a direct effect of the previous observation on the current observation, after accounting for intervening observations.



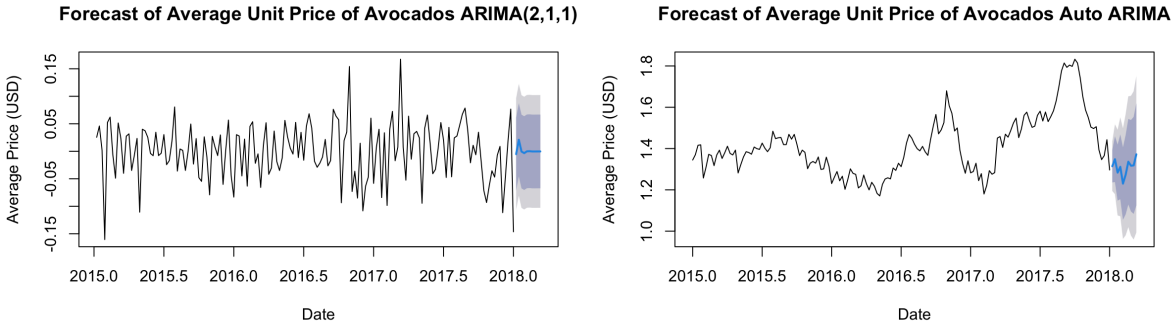


4. ***Model fitting -*** ACF and PACF plots helped identify autocorrelation structures in the differenced series. Through careful examination, optimal parameters (P,Q) were determined to be (1,1). Evaluating multiple ARIMA models, ARIMA(2,1,1) with the lowest AIC was selected. Contrasted with auto-ARIMA, ARIMA(2,1,1) demonstrated

superior fit. However, the best fitting model may not always be the best forecasting model.

5. ***Diagnostic Tests -*** Using the Augmented Dickey-Fuller tests, we find that the auto ARIMA model's residuals have a p-value of 0.01 and the ARIMA(2,1,1) model's residuals have a p-value of 0.028. This suggests that we find the residuals to be stationary and reject the null hypothesis of non-stationarity in both scenarios.

6. ***Model Forecasting -*** Upon applying both models to the training dataset, it is apparent that the ARIMA(2,1,1) model yielded a nearly flat forecast, while the auto arima model provided a good forecast.But, further testing was necessary before reaching any definite findings, even if the forecast was simple to depict using the Auto Arimas model.



7. ***Forecasting model checks -*** Upon visualising the forecasts from both models, additional testing was required to determine which model was the best. To do this, the testing dataset had to be used to calculate predicting errors like MSE and RMSE. In that case, auto arima's forecast error was lower than ARIMA's (2,1,1). In contrast to autoarima, which has MSE and RMSE of 0.04 and 0.213, ARIMA(2,1,1) has MSE and RMSE of 1.632 and 1.277.

Thus, we can conclude that, following a number of model evaluations, Auto Arima is the most appropriate and accurate model for predicting avocado prices in USD.

Conclusion

In conclusion, the industry would be benefited greatly from the helpful information we were able to obtain from our time series study of avocado pricing data. After doing thorough exploratory analysis and fitting the model, we determined that the auto-ARIMA model was the best appropriate for predicting avocado prices. Better supply chain management and pricing plans are made possible by the projected prices, which give stakeholders vital information. Even though our method worked well, there were several drawbacks, notably the reliance on the assumption of stationarity and the impact of outside variables like the state of the economy and weather. For increased accuracy, future studies may focus on adding outside variables to the predicting models.

[END of the REPORT]

```r
# Uncomment the below code to first install the packages
# install.packages("forecast")
# install.packages("ggplot")
# install.packages("fma")
# install.packages("tseries")

# If having any issues, in knitting into
# PDF then please install the below in the console
#tinytex::check_installed()
```

Caution : The images above has been saved and loaded defaultly from macbook and therefore it has a unique number. To properly knit, kinldy choose the defalut path of the image saved in your directory.

```r
# Loading necessary packages
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(ggplot2)
library(fma)
library(tseries)
```

```r
# Loading the file path from directory
file_path <- "~/Downloads/avocado.csv"
avocado_df <- read.csv(file_path)
```

Data Preprocessing and Cleaning

```r
# Data Sorting
# Converting Date column to Date format
avocado_df$Date <- as.Date(avocado_df$Date)
# Extracting year from Date
avocado_df$Year <- format(avocado_df$Date, "%Y")
# Sorting data before splitting
avocado_df <- avocado_df[order(avocado_df$Date), ]
```

```r
# Data Splitting
# Training Set
training_data <- avocado_df[avocado_df$Date <= as.Date("2017-12-31"), ]

# Testing Set
testing_data <- avocado_df[avocado_df$Date >= as.Date("2018-01-01")
                           & avocado_df$Date <= as.Date("2018-03-11"), ]
```

```r
# Checking for missing or NAN values in splitted datasets
# Total number of missing values in the training dataset
train_missing_values <- sum(colSums(is.na(training_data)))
print(train_missing_values)
```

```
## [1] 1620
```

```r
# Total number of missing values in testing dataset
test_missing_values <- sum(colSums(is.na(testing_data)))
print(test_missing_values)
```

```
## [1] 87
```

```r
# Handling missing values by imputation method
# Selecting numeric columns for imputation in training dataset
numeric_cols_train <- sapply(training_data, is.numeric)
training_data_imputed <- training_data
training_data_imputed[,
numeric_cols_train] <- lapply(training_data_imputed[,
numeric_cols_train], function(x) ifelse(is.na(x),
mean(x, na.rm = TRUE), x))

# Select numeric columns for imputation in testing dataset
numeric_cols_test <- sapply(testing_data, is.numeric)
testing_data_imputed <- testing_data
testing_data_imputed[, numeric_cols_test] <- lapply(
testing_data_imputed[,
numeric_cols_test], function(x) ifelse(is.na(x),
mean(x, na.rm = TRUE), x))
```

```r
# Checking for missing values after imputation
# Checking for missing values in imputed training dataset
training_imputed_missing <- colSums(is.na(training_data_imputed))
print(training_imputed_missing)
```

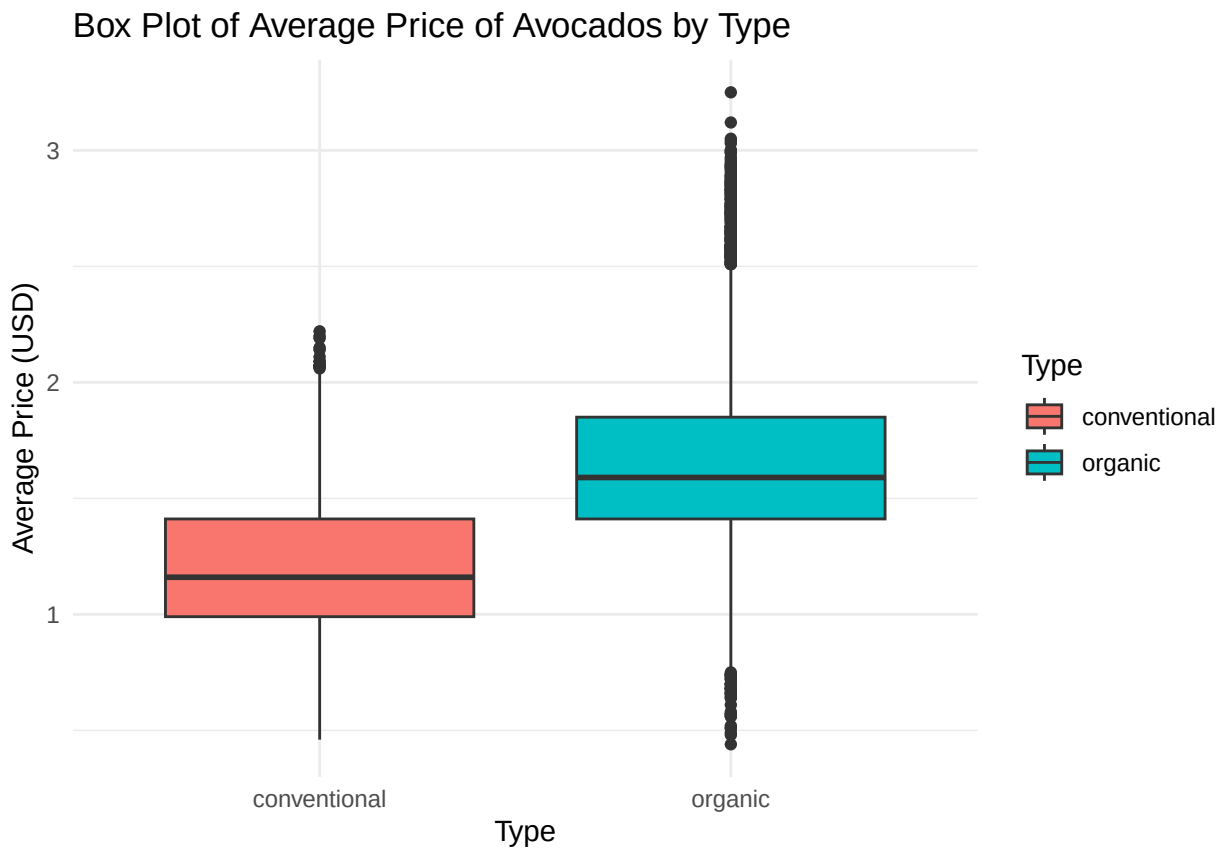
```
##          X.1            X         Date AveragePrice Total.Volume         X4046
##            0            0            0            0            0             0
##         X4225        X4770  Total.Bags  Small.Bags  Large.Bags  XLarge.Bags
##            0            0            0            0            0             0
##         type         year       region         Year
##            0            0            0            0
```

```
# Checking for missing values in imputed testing dataset
testing_imputed_missing <- colSums(is.na(testing_data_imputed))
print(testing_imputed_missing)
```

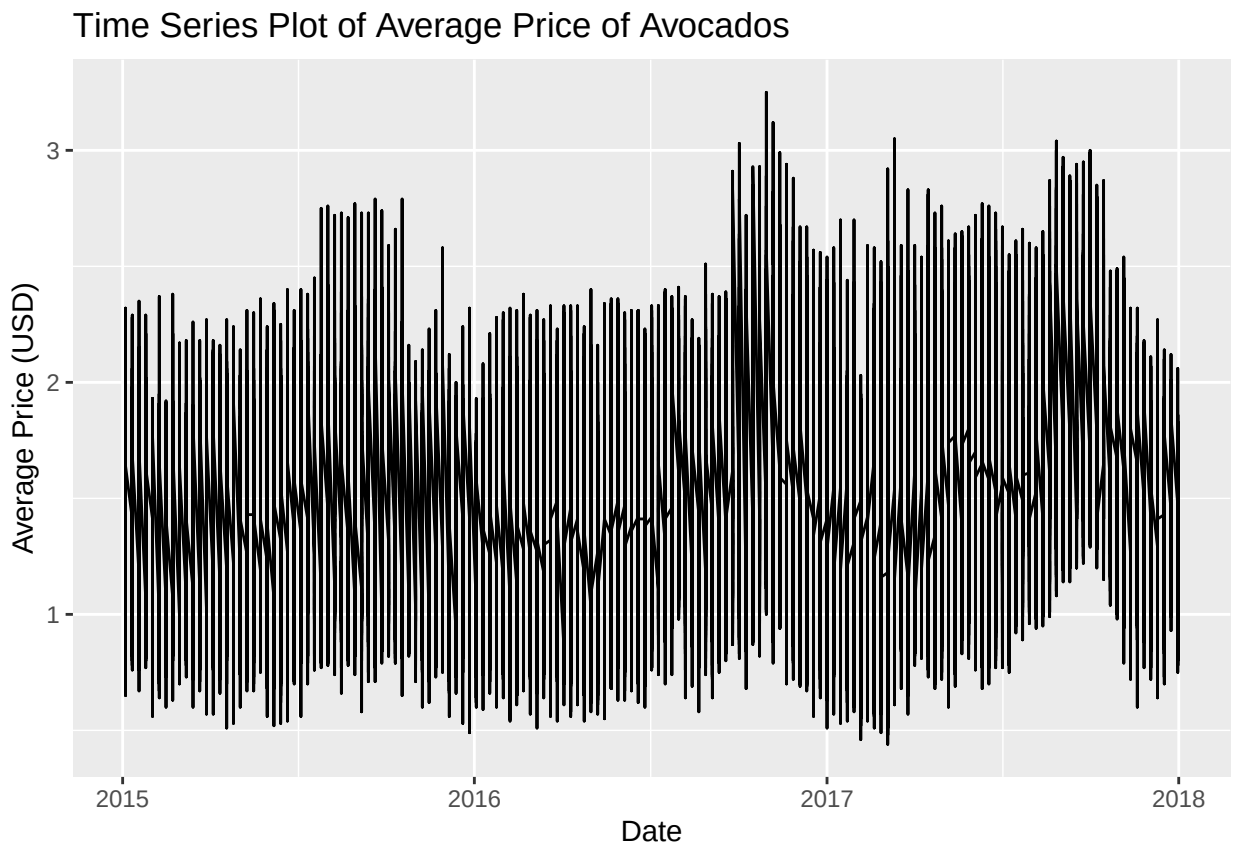
```
##          X.1            X         Date AveragePrice Total.Volume         X4046
##            0            0            0            0            0             0
##         X4225        X4770  Total.Bags  Small.Bags  Large.Bags  XLarge.Bags
##            0            0            0            0            0             0
##         type         year       region         Year
##            0            0            0            0
```

Data Exploration using Training Dataset

```
# Exploring trained dataset explicitly showing outliers
# Box plot of AveragePrice with respect to type
ggplot(training_data_imputed, aes(x = type, y = AveragePrice, fill = type))+
  geom_boxplot() +
  labs(title = "Box Plot of Average Price of Avocados by Type",
  x = "Type", y = "Average Price (USD)",
  fill = "Type") +
  theme_minimal()
```




```r
# Time series plot of AveragePrice with resect to Date
ggplot(training_data_imputed, aes(x = Date, y = AveragePrice)) +
geom_line() + labs(
  title = "Time Series Plot of Average Price of Avocados",
  x = "Date",
  y = "Average Price (USD)")
```

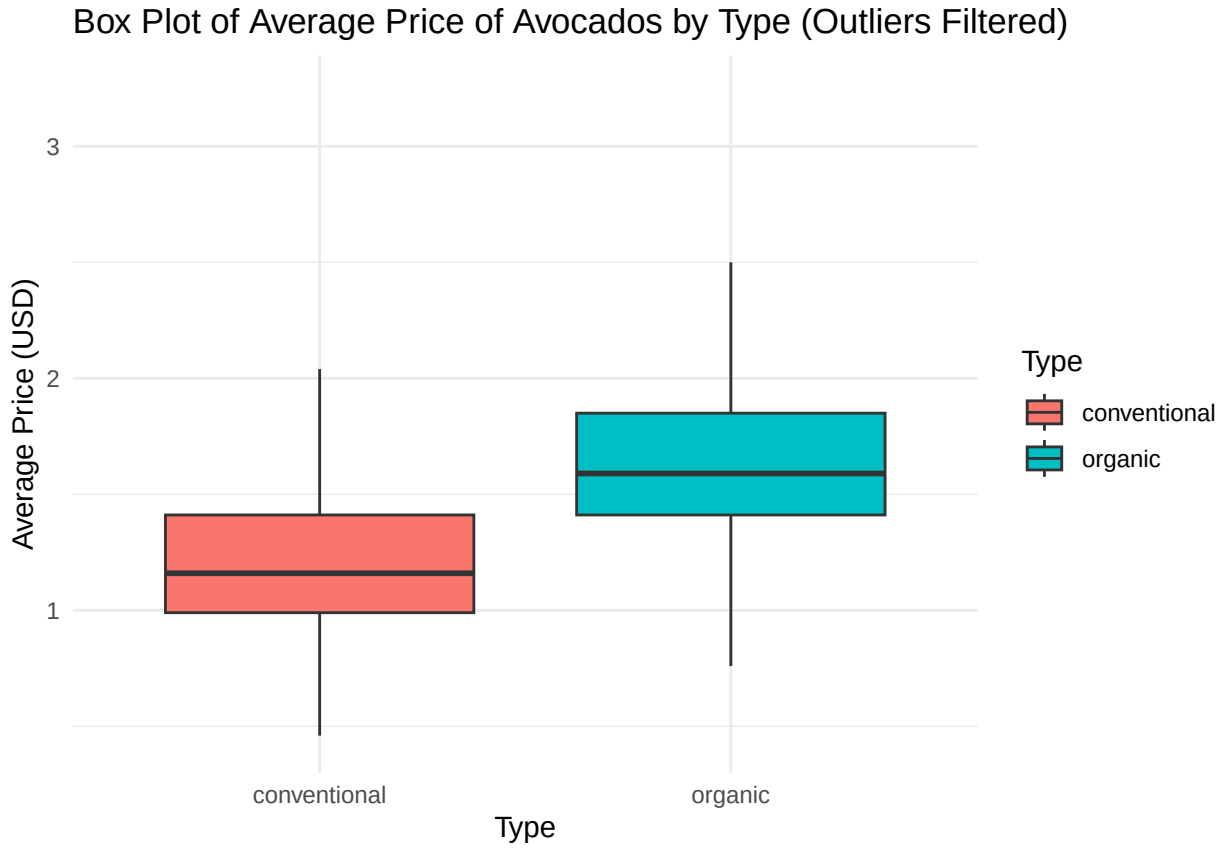



```r
# Handling outliers in the data plots

# Defining IQR to show outliers
lower_percentile <- quantile(training_data_imputed$AveragePrice, 0.005)
upper_percentile <- quantile(training_data_imputed$AveragePrice, 0.995)

# Filtering outliers based on IQR
training_data_filtered <- training_data_imputed[
training_data_imputed$AveragePrice >= lower_percentile &
training_data_imputed$
AveragePrice <= upper_percentile, ]
```

```r
# Box plot of AveragePrice with respect to type
ggplot(training_data_imputed, aes(x = type, y = AveragePrice, fill = type)) +
  geom_boxplot(outlier.shape = NA) +
  labs(
  title = "Box Plot of Average Price of Avocados by Type (Outliers Filtered)",
  x = "Type", y = "Average Price (USD)",
  fill = "Type") +
  theme_minimal()
```

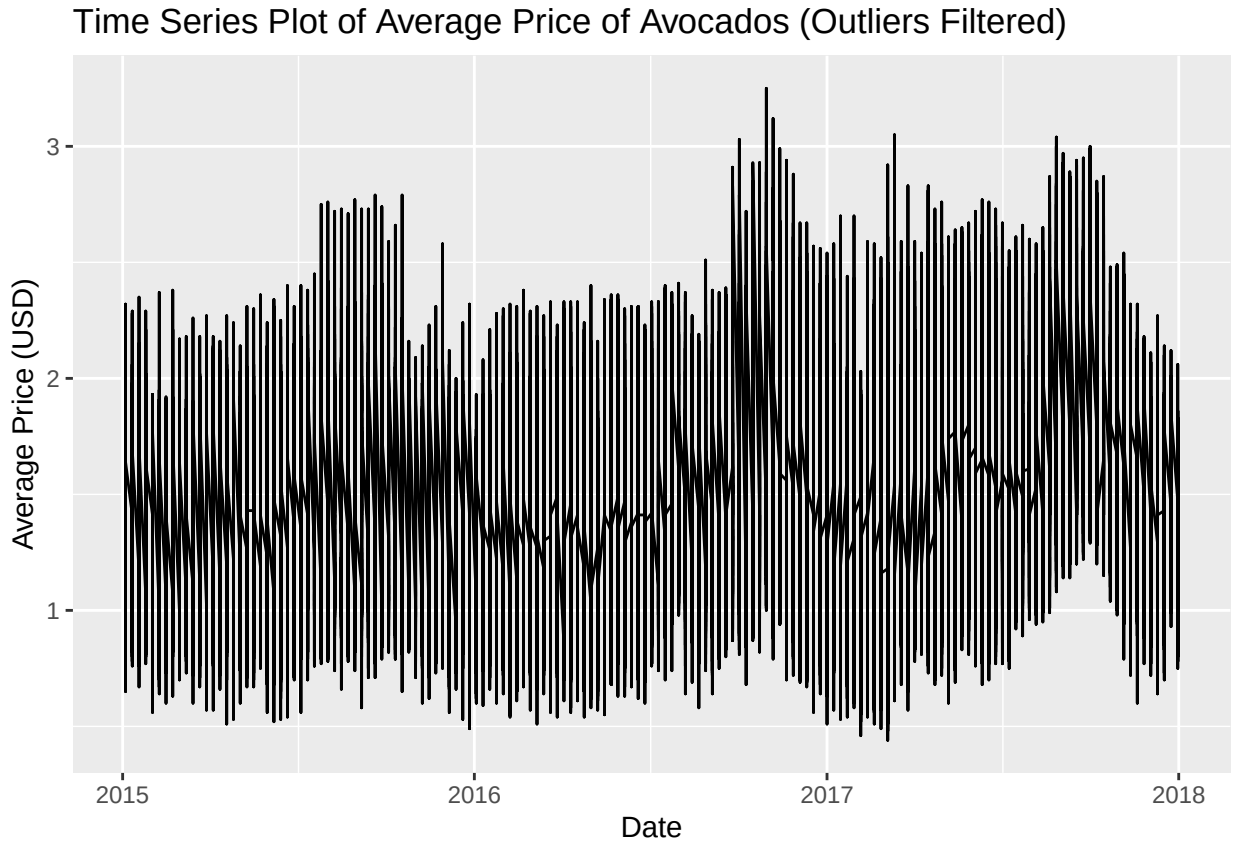



```r
# Handling outliers in the time series plot
# Calculating the mean and standard deviation of AveragePrice
mean_price <- mean(training_data_imputed$AveragePrice)
sd_price <- sd(training_data_imputed$AveragePrice)

# Defining the threshold for outliers
threshold <- 3

# Filtering out outliers
training_data_ts_filtered <- training_data_imputed[abs(
  training_data_imputed$AveragePrice - mean_price) <
    threshold * sd_price, ]
```

```r
# Time Series plot of Average Price with respect to Date
ggplot(training_data_imputed, aes(x = Date, y = AveragePrice)) +
  geom_line() +
  labs(
  title = "Time Series Plot of Average Price of Avocados (Outliers Filtered)",
  x = "Date",
  y = "Average Price (USD)")
```




```
# Summary Statistics for numerical variables
summary(training_data_imputed[, numeric_cols_train])
```

```
##       X.1              X          AveragePrice    Total.Volume
##  Min.   :    1   Min.   : 0.00   Min.   :0.440   Min.   :      85
##  1st Qu.: 4239   1st Qu.:13.00   1st Qu.:1.120   1st Qu.:   10460
##  Median : 8477   Median :26.00   Median :1.411   Median :  104849
##  Mean   : 8801   Mean   :25.66   Mean   :1.411   Mean   :  834110
##  3rd Qu.:13363   3rd Qu.:39.00   3rd Qu.:1.640   3rd Qu.:  423186
##  Max.   :17601   Max.   :52.00   Max.   :3.250   Max.   :61034457
##      X4046             X4225             X4770          Total.Bags
##  Min.   :       0   Min.   :       0   Min.   :      0.0   Min.   :       0
##  1st Qu.:     855   1st Qu.:    3012   1st Qu.:      0.0   1st Qu.:    4558
##  Median :    8263   Median :   29162   Median :    184.6   Median :   38480
##  Mean   :  288245   Mean   :  293666   Mean   :  23233.0   Mean   :  228965
##  3rd Qu.:  110508   3rd Qu.:  149877   3rd Qu.:   6444.3   3rd Qu.:  105615
##  Max.   :22743616   Max.   :20470573   Max.   :2546439.1   Max.   :16298296
##    Small.Bags         Large.Bags        XLarge.Bags            year
##  Min.   :       0   Min.   :      0   Min.   :     0.0   Min.   :2015
##  1st Qu.:    2466   1st Qu.:    113   1st Qu.:     0.0   1st Qu.:2015
##  Median :   25200   Median :   2479   Median :     0.0   Median :2016
```

```
##  Mean   :  174844   Mean   :  51202   Mean   :  2918.6   Mean   :2016
##  3rd Qu.:   79805   3rd Qu.:  20433   3rd Qu.:    96.4   3rd Qu.:2017
##  Max.   :12567156   Max.   :4324231   Max.   :551693.7   Max.   :2017
```

```r
# Summary Statistics for categorical variables
summary(training_data_imputed$type)
```

```
##    Length     Class      Mode
##     16953 character character
```

```r
summary(training_data_imputed$region)
```

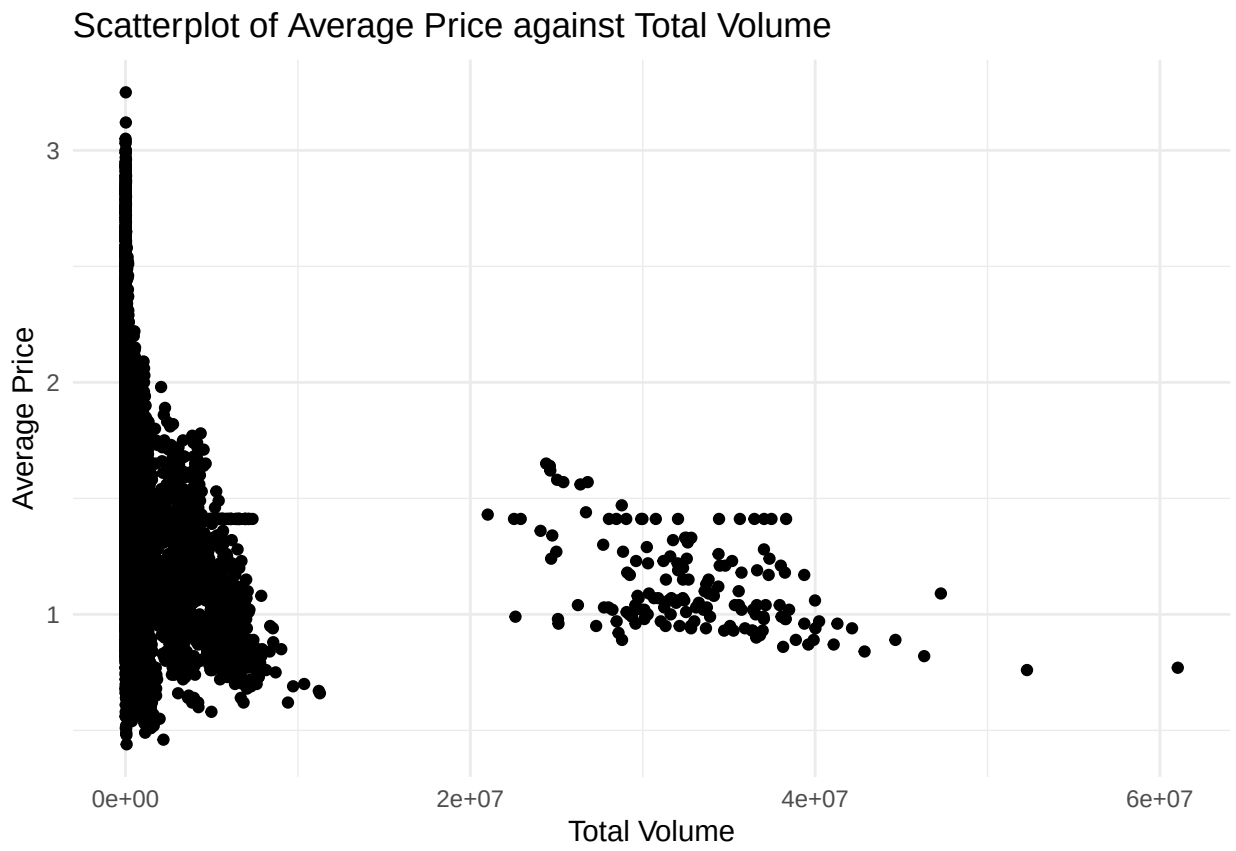
```
##    Length     Class      Mode
##     16953 character character
```

```r
# Correlation Analysis
correlation_matrix <- cor(training_data_imputed[, numeric_cols_train])
print(correlation_matrix)
```

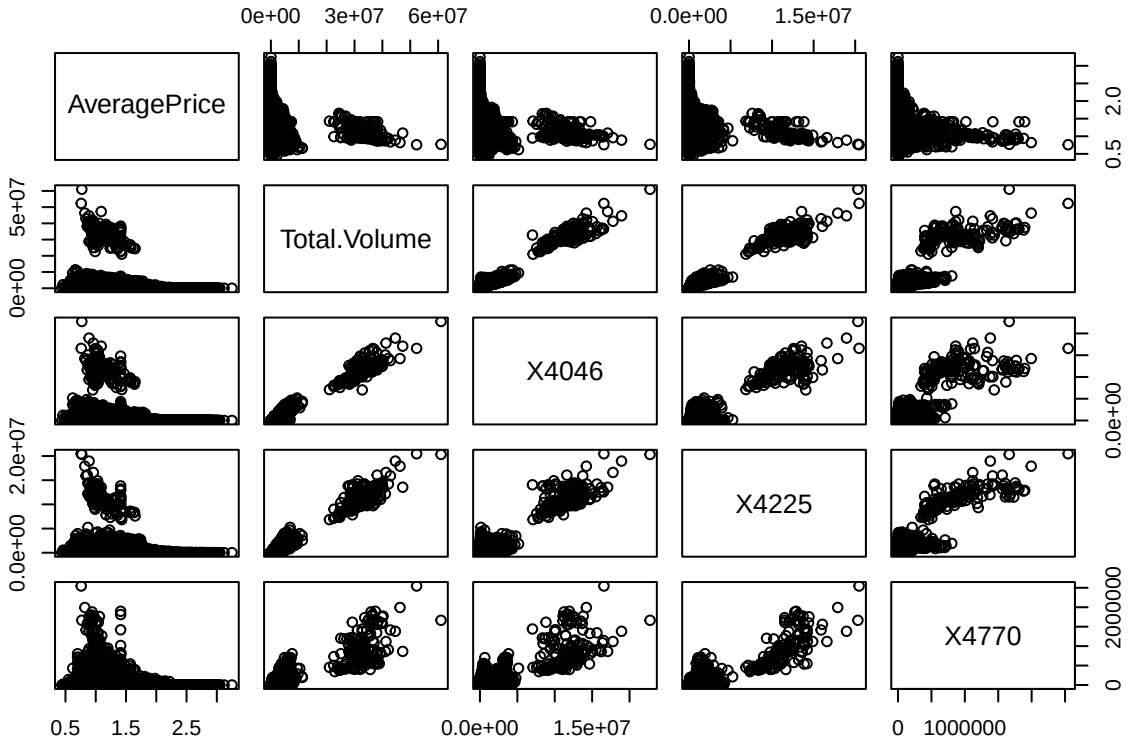
```
##                      X.1            X AveragePrice Total.Volume        X4046
## X.1           1.000000000  0.008577148    0.5706789 -0.173580099 -0.170996217
## X             0.008577148  1.000000000   -0.1533114  0.022384666  0.024850320
## AveragePrice  0.570678857 -0.153311414    1.0000000 -0.183243791 -0.197618539
## Total.Volume -0.173580099  0.022384666   -0.1832438  1.000000000  0.976985461
## X4046        -0.170996217  0.024850320   -0.1976185  0.976985461  1.000000000
## X4225        -0.188450732  0.023435838   -0.1652497  0.975662918  0.926614641
## X4770        -0.185884517  0.040440390   -0.1712809  0.880816177  0.839864305
## Total.Bags   -0.136819083  0.013121517   -0.1677187  0.961728693  0.915641047
## Small.Bags   -0.144339891  0.014768256   -0.1651554  0.966319951  0.921361332
## Large.Bags   -0.103294847  0.007525878   -0.1654856  0.879275833  0.834103103
## XLarge.Bags  -0.104603394  0.002782666   -0.1059909  0.724849913  0.673739588
## year          0.445553214  0.013456371    0.1353715  0.009805297 -0.005120501
##                    X4225       X4770  Total.Bags  Small.Bags   Large.Bags
## X.1          -0.18845073 -0.18588452 -0.13681908 -0.14433989 -0.103294847
## X             0.02343584  0.04044039  0.01312152  0.01476826  0.007525878
## AveragePrice -0.16524968 -0.17128094 -0.16771867 -0.16515539 -0.165485626
## Total.Volume  0.97566292  0.88081618  0.96172869  0.96631995  0.879275833
## X4046         0.92661464  0.83986430  0.91564105  0.92136133  0.834103103
## X4225         1.00000000  0.89296149  0.90823276  0.91850656  0.812539510
## X4770         0.89296149  1.00000000  0.80550611  0.81490450  0.713754318
## Total.Bags    0.90823276  0.80550611  1.00000000  0.99474828  0.943721721
## Small.Bags    0.91850656  0.81490450  0.99474828  1.00000000  0.905246247
```

```
## Large.Bags    0.81253951  0.71375432  0.94372172  0.90524625  1.000000000
## XLarge.Bags   0.67219745  0.67619141  0.78313093  0.78495709  0.691422835
## year         -0.01491759 -0.03570635  0.06538593  0.05751791  0.083472152
##                XLarge.Bags         year
## X.1          -0.104603394  0.445553214
## X             0.002782666  0.013456371
## AveragePrice -0.105990909  0.135371519
## Total.Volume  0.724849913  0.009805297
## X4046         0.673739588 -0.005120501
## X4225         0.672197450 -0.014917590
## X4770         0.676191408 -0.035706354
## Total.Bags    0.783130933  0.065385931
## Small.Bags    0.784957092  0.057517909
## Large.Bags    0.691422835  0.083472152
## XLarge.Bags   1.000000000  0.077871295
## year          0.077871295  1.000000000
```

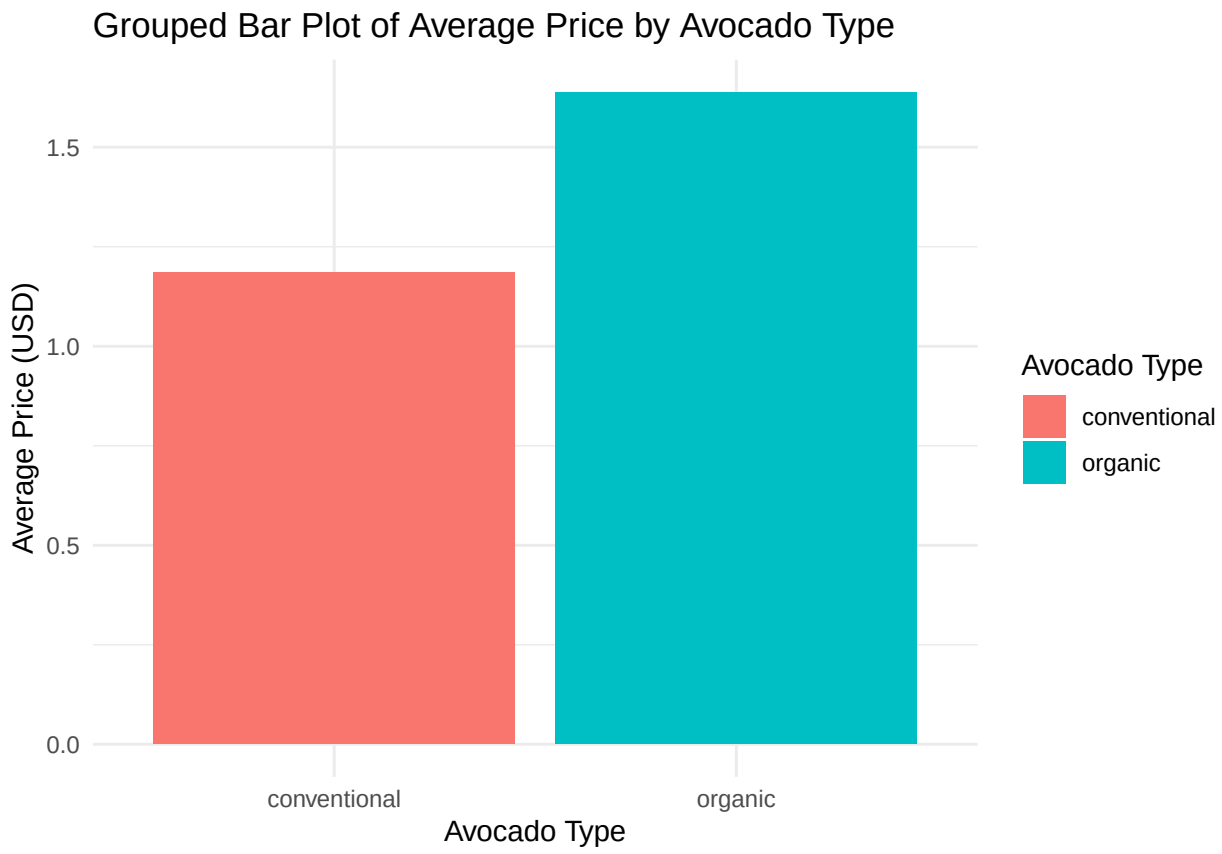
```r
# Scatterplot of AveragePrice against Total Volume
ggplot(training_data_imputed, aes(x = Total.Volume, y = AveragePrice)) +
  geom_point() +
  labs(title = "Scatterplot of Average Price against Total Volume",
       x = "Total Volume", y = "Average Price") +
  theme_minimal()
```



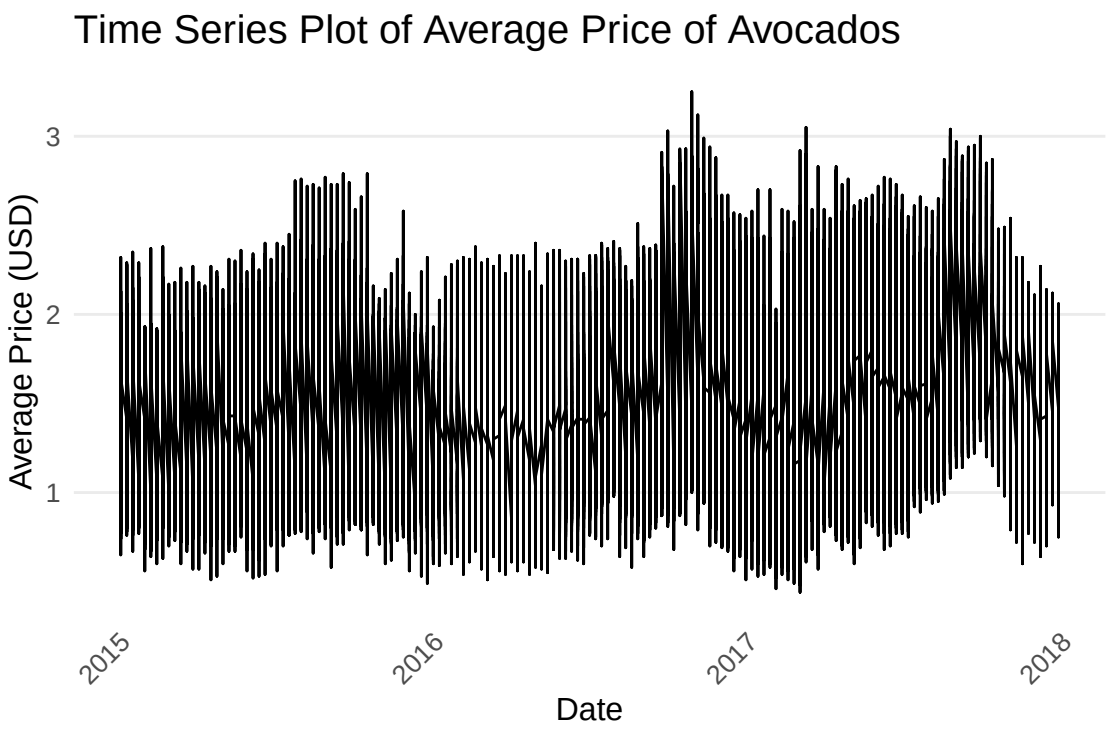

```
# Pairwise plot of selected variables
pairs(~ AveragePrice + Total.Volume + X4046 + X4225 + X4770,
      data = training_data_imputed)
```




```r
# Grouped bar plot for categorical variables
ggplot(training_data_imputed, aes(
  x = type, y = AveragePrice, fill = type))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge") +
  labs(title = "Grouped Bar Plot of Average Price by Avocado Type",
  x = "Avocado Type", y = "Average Price (USD)",
  fill = "Avocado Type") +
  theme_minimal()
```




```r
# Time Series plot of Average Price of Avocados
ggplot(training_data_imputed, aes(x = Date, y = AveragePrice)) +
  geom_line() +
  labs(title = "Time Series Plot of Average Price of Avocados",
       x = "Date", y = "Average Price (USD)") +
  theme_minimal() +
  theme(plot.title = element_text(size = 15)) +
  theme(axis.text.x = element_text(size = 10, angle = 45, hjust = 1)) +
  theme(axis.text.y = element_text(size = 10)) +
  theme(axis.title = element_text(size = 12)) +
  theme(plot.margin = margin(1, 1, 1, 1, "cm")) +
  theme(panel.grid.minor = element_blank()) +
  theme(panel.grid.major.x = element_blank())
```



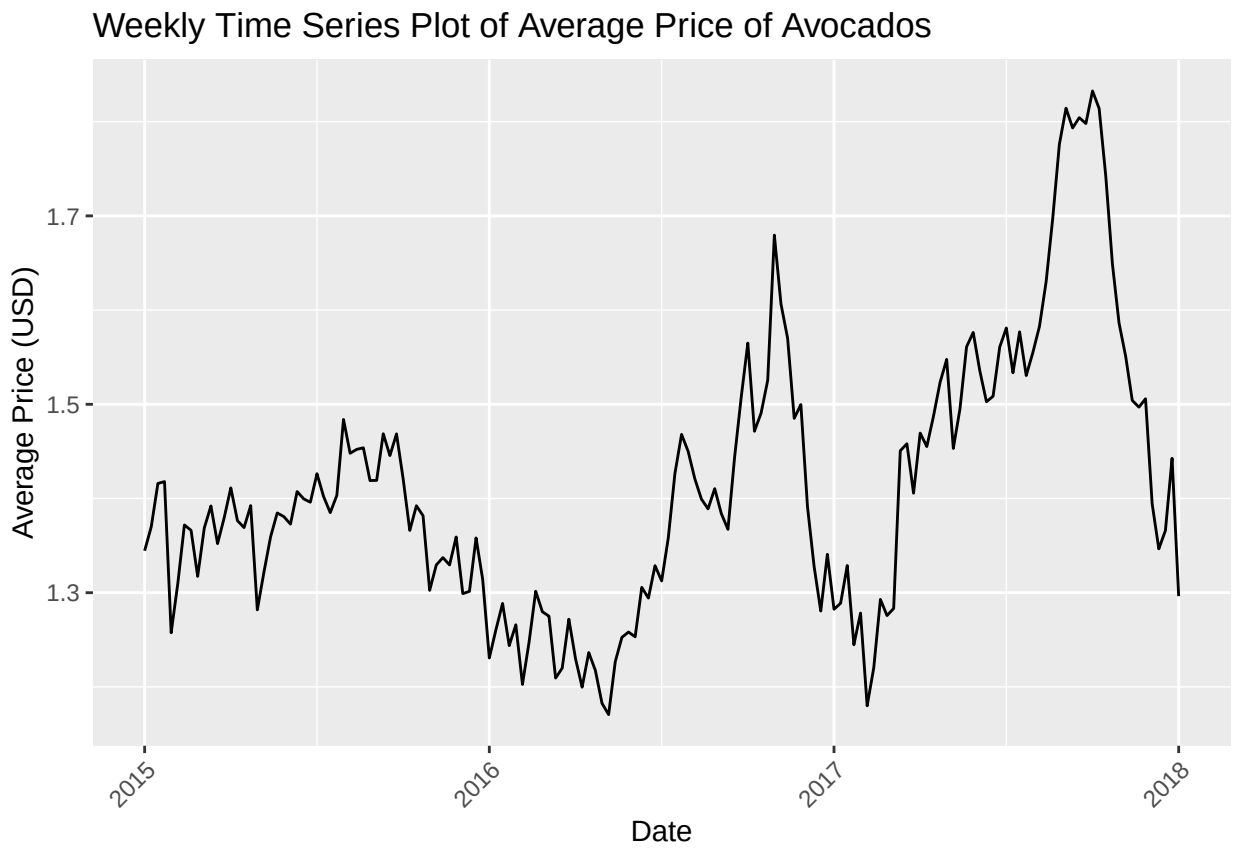

```r
# Aggregating the data to weekly frequency
weekly_data <- aggregate(training_data_imputed$AveragePrice,
                         by = list(week = format(
                        training_data_imputed$Date, "%Y-%U")),
                         FUN = mean)

# Converting the aggregated data to time series objects
ts_weekly_data <- ts(weekly_data$x, start = c(2015, 1), frequency = 52)
```

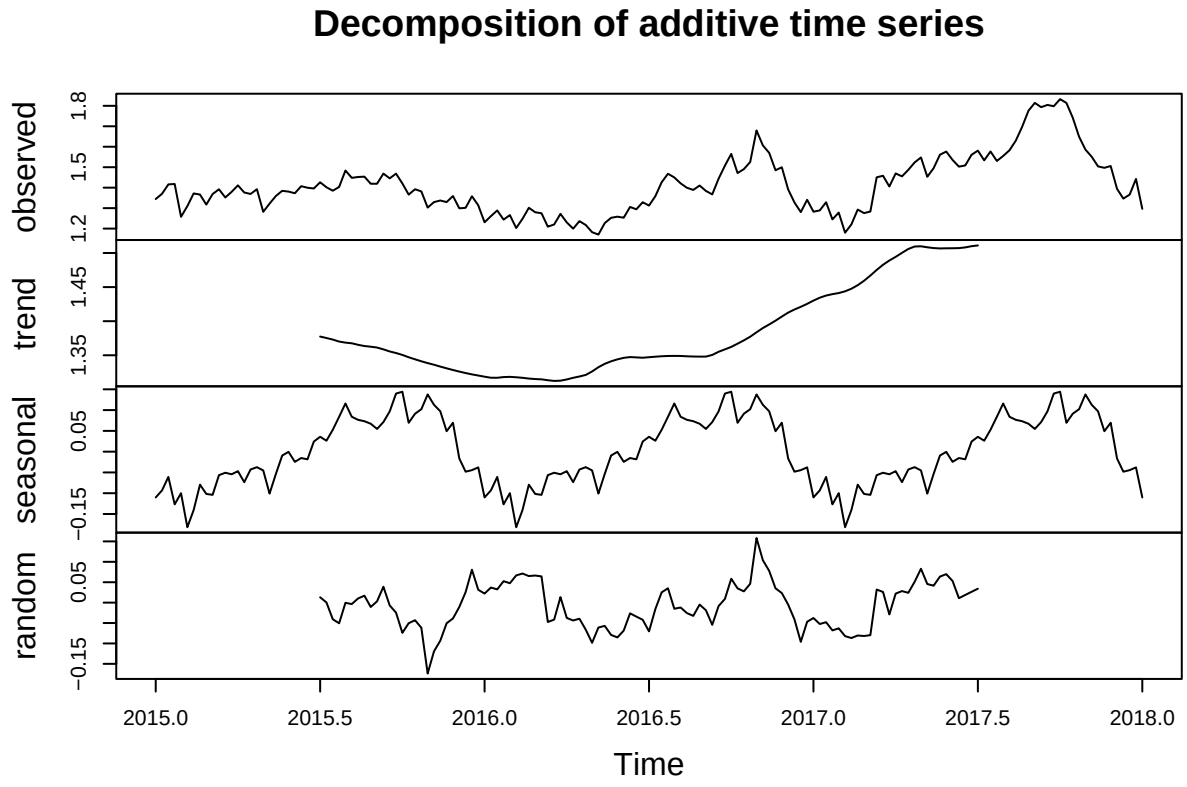
Steps for model fitting and forecasting

1) Time series plot observation :

```r
# Ploting the monthly time series data
autoplot(
  ts_weekly_data,
  main = "Weekly Time Series Plot of Average Price of Avocados",
  xlab = "Date", ylab = "Average Price (USD)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



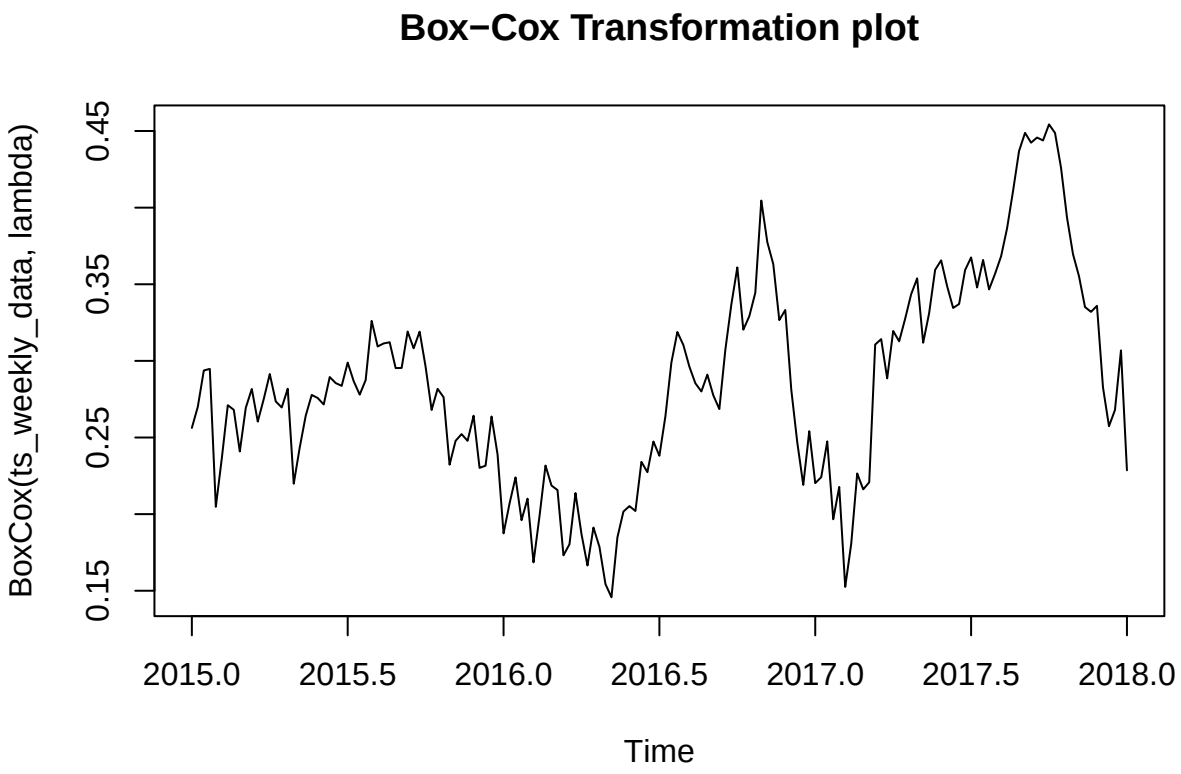

```r
# Decomposing the Weekly time series data
avocado_decomp <- decompose(ts_weekly_data)
par(mar = c(8, 5, 4, 2) + 0.1)
plot(avocado_decomp)
```



2) Variance Stablisation :

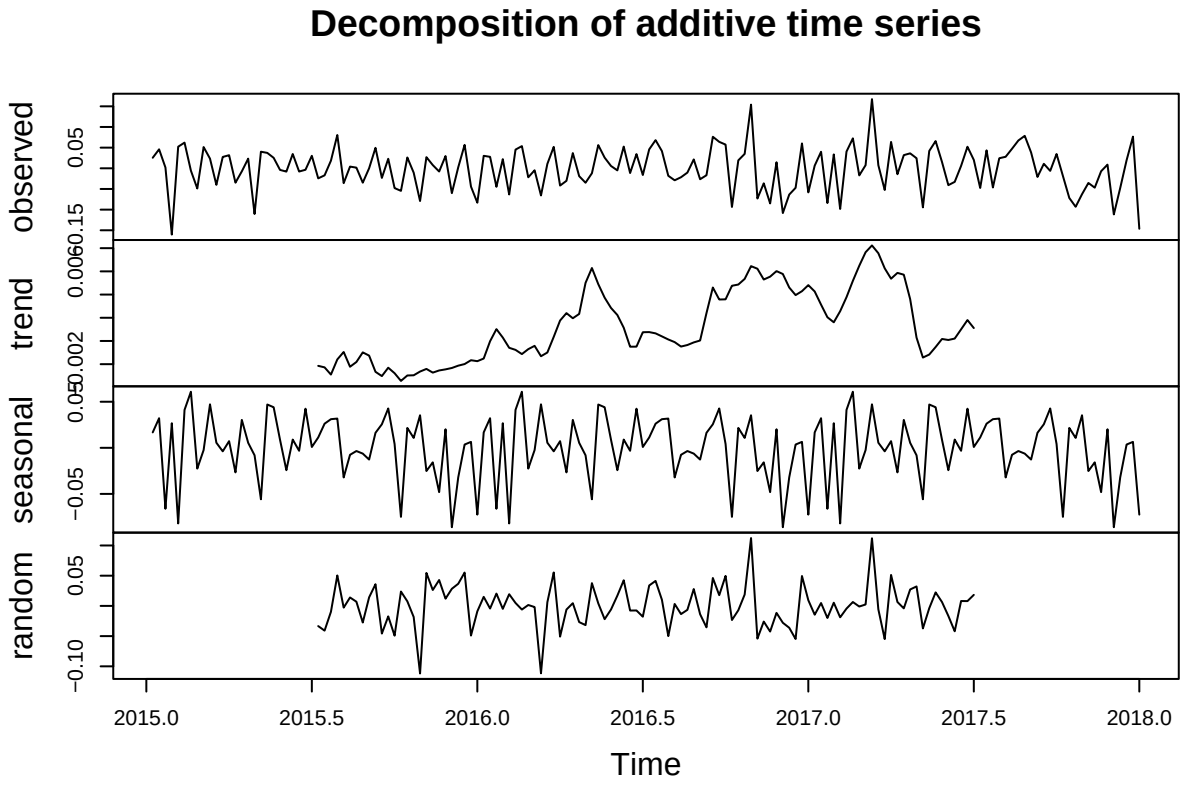
```r
# Stablising the variance using Box-Cox Transformation
lambda <- BoxCox.lambda(ts_weekly_data)

# Plot the transformed data
plot(BoxCox(ts_weekly_data,lambda),
    main = "Box-Cox Transformation plot")
```



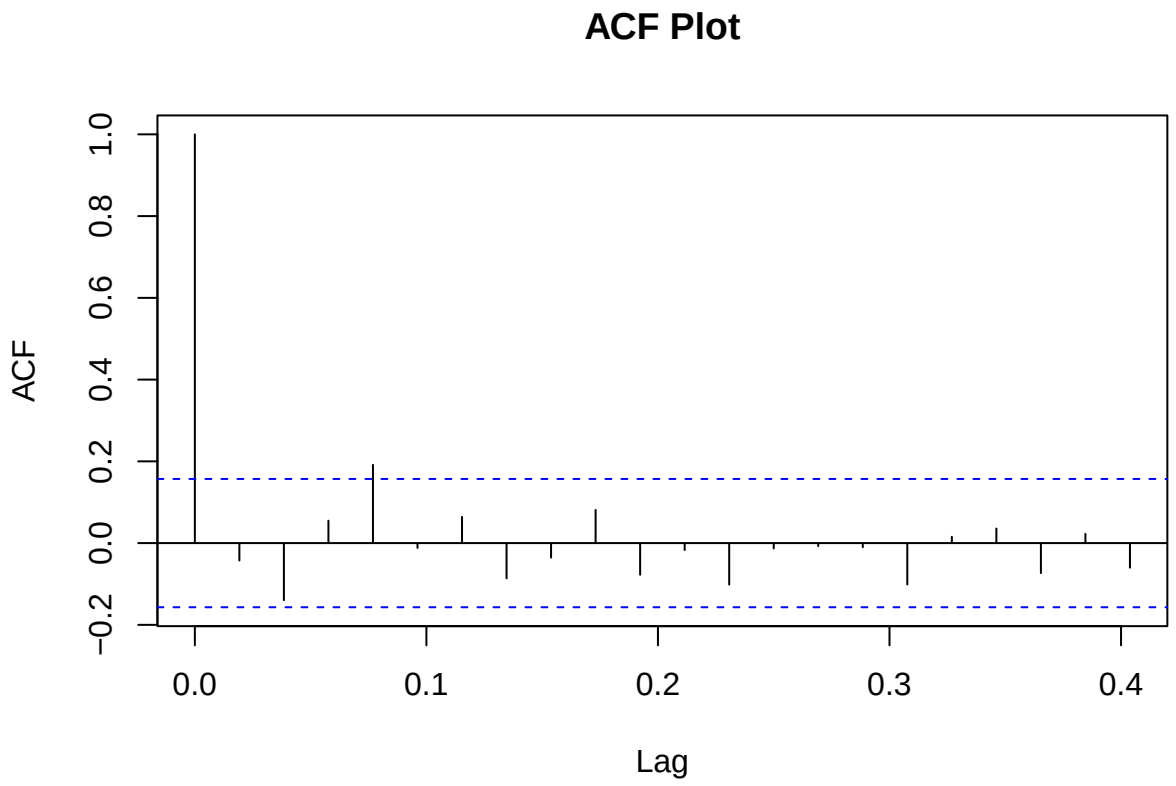
3) Removing Trend and Seasonality :

```r
# Differencing the time series to remove trend and seasonality
ts_weekly_data_diff <- diff(ts_weekly_data)
par(mar = c(8, 5, 4, 2) + 0.1)
plot(decompose(ts_weekly_data_diff))
```

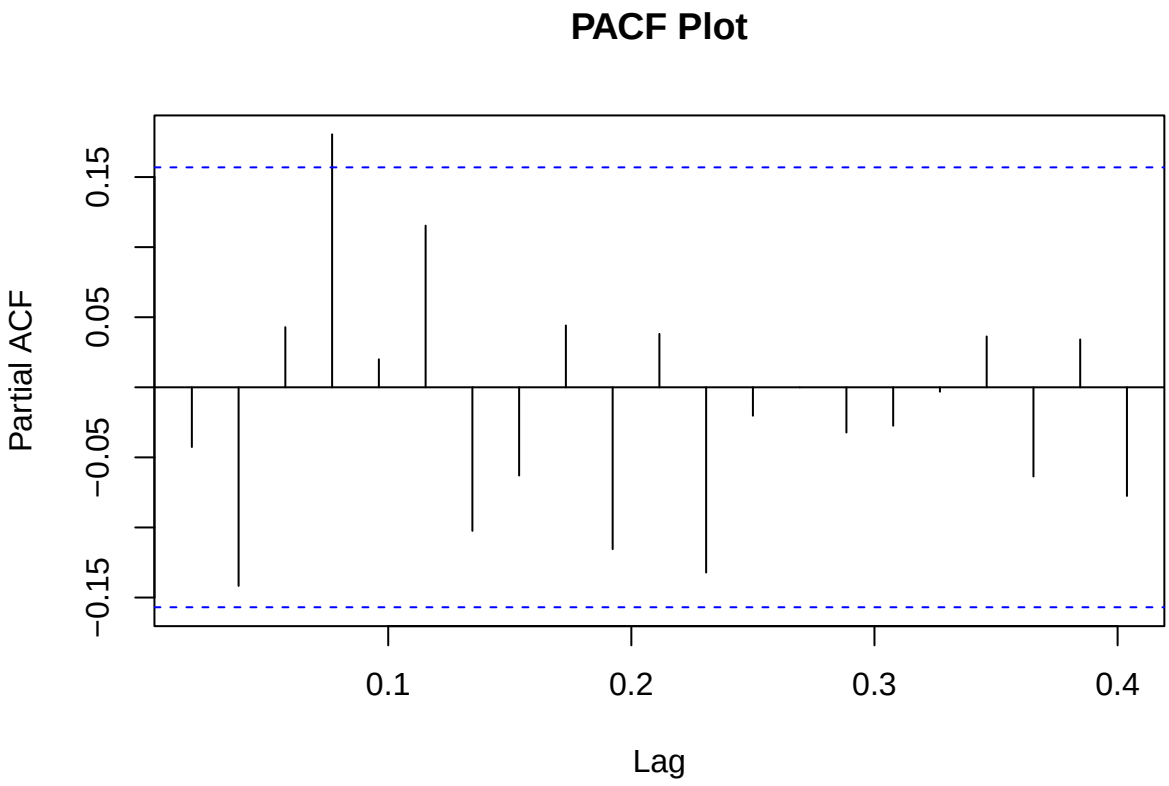


4) Model Selection :

```
# Examining ACF/PACf plots
acf_result <- acf(ts_weekly_data_diff, main = "ACF Plot")
```




```r
pacf_result <- pacf(ts_weekly_data_diff, main = "PACF Plot")
```



5) Model Fitting :

```
# Model fitting using standard ARIMA
arima_model1 <- Arima(ts_weekly_data_diff, order = c(1,1,2))
print(arima_model1)
```

```
## Series: ts_weekly_data_diff
## ARIMA(1,1,2)
##
## Coefficients:
##           ar1      ma1      ma2
##       -0.5284  -0.4578  -0.5421
## s.e.   0.8878   0.8609   0.8606
##
## sigma^2 = 0.002725:  log likelihood = 236.71
## AIC=-465.43   AICc=-465.16   BIC=-453.25
```

```r
arima_model2 <- Arima(ts_weekly_data_diff, order = c(1,1,1))
print(arima_model2)
```

```
## Series: ts_weekly_data_diff
## ARIMA(1,1,1)
##
## Coefficients:
##           ar1      ma1
##       -0.0383  -1.0000
## s.e.   0.0825   0.0215
##
## sigma^2 = 0.002703:  log likelihood = 236.8
## AIC=-467.6   AICc=-467.44   BIC=-458.47
```

```r
arima_model3 <- Arima(ts_weekly_data_diff, order = c(2,1,1))
print(arima_model3)
```

```
## Series: ts_weekly_data_diff
## ARIMA(2,1,1)
##
## Coefficients:
##           ar1      ar2      ma1
##       -0.0420  -0.1437  -1.0000
## s.e.   0.0816   0.0821   0.0245
##
## sigma^2 = 0.002663:  log likelihood = 238.32
## AIC=-468.63   AICc=-468.36   BIC=-456.46
```

```r
arima_model4 <- Arima(ts_weekly_data_diff, order = c(2,1,2))
print(arima_model4)
```

```
## Series: ts_weekly_data_diff
## ARIMA(2,1,2)
##
## Coefficients:
##           ar1      ar2      ma1      ma2
##       -0.1445  -0.1496  -0.8954  -0.1046
## s.e.   0.2953   0.0822   0.2913   0.2903
##
## sigma^2 = 0.002679:  log likelihood = 238.38
## AIC=-466.75   AICc=-466.35   BIC=-451.53
```

Model	AIC	BIC
ARIMA(1,1,2)	-465.43	-453.25
ARIMA(1,1,1)	-467.6	-458.47
ARIMA(2,1,1)	-468.63	-456.46
ARIMA(2,1,2)	-466.75	-451.53

```r
# Model fitting using autoarima function
auto_arima_model <- auto.arima(
ts_weekly_data, trace = TRUE,
approximation = TRUE, stepwise = TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2)(1,1,1)[52]                     : Inf
##  ARIMA(0,1,0)(0,1,0)[52]                     : 25.35584
##  ARIMA(1,1,0)(1,1,0)[52]                     : 4.02954
##  ARIMA(0,1,1)(0,1,1)[52]                     : 14.89156
##  ARIMA(1,1,0)(0,1,0)[52]                     : 28.42684
##  ARIMA(1,1,0)(1,1,1)[52]                     : Inf
##  ARIMA(1,1,0)(0,1,1)[52]                     : 16.29667
##  ARIMA(0,1,0)(1,1,0)[52]                     : 0.1530978
##  ARIMA(0,1,0)(1,1,1)[52]                     : Inf
##  ARIMA(0,1,0)(0,1,1)[52]                     : 14.11114
##  ARIMA(0,1,1)(1,1,0)[52]                     : 2.25349
##  ARIMA(1,1,1)(1,1,0)[52]                     : Inf
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(0,1,0)(1,1,0)[52]                     : -269.1936
##
##  Best model: ARIMA(0,1,0)(1,1,0)[52]
```

```r
print(auto_arima_model)
```

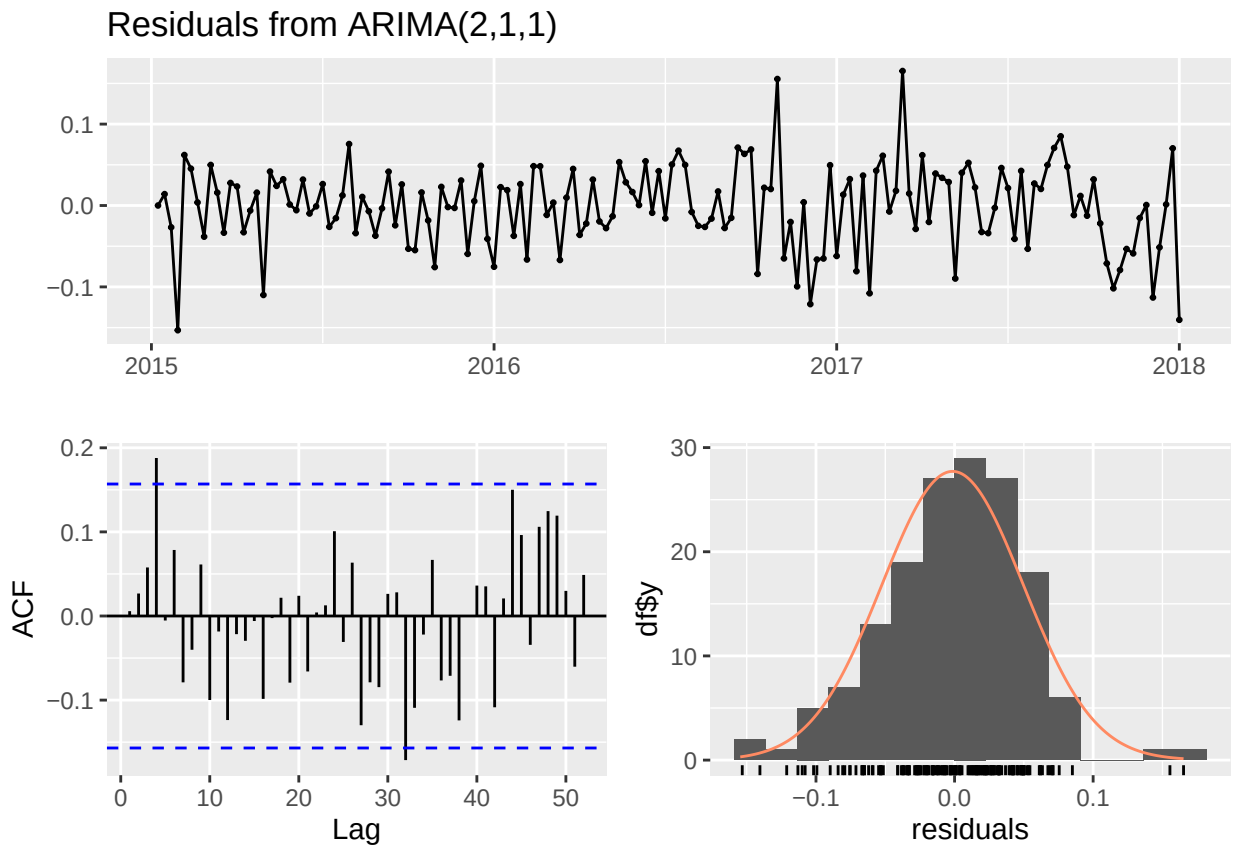
```
## Series: ts_weekly_data
## ARIMA(0,1,0)(1,1,0)[52]
##
## Coefficients:
##          sar1
##       -0.4880
## s.e.   0.1057
```

```
##
## sigma^2 = 0.003728:  log likelihood = 136.66
## AIC=-269.31   AICc=-269.19   BIC=-264.02
```

Model	AIC	BIC
ARIMA(0,1,0)(1,1,0)[52]	-269.31	-264.02

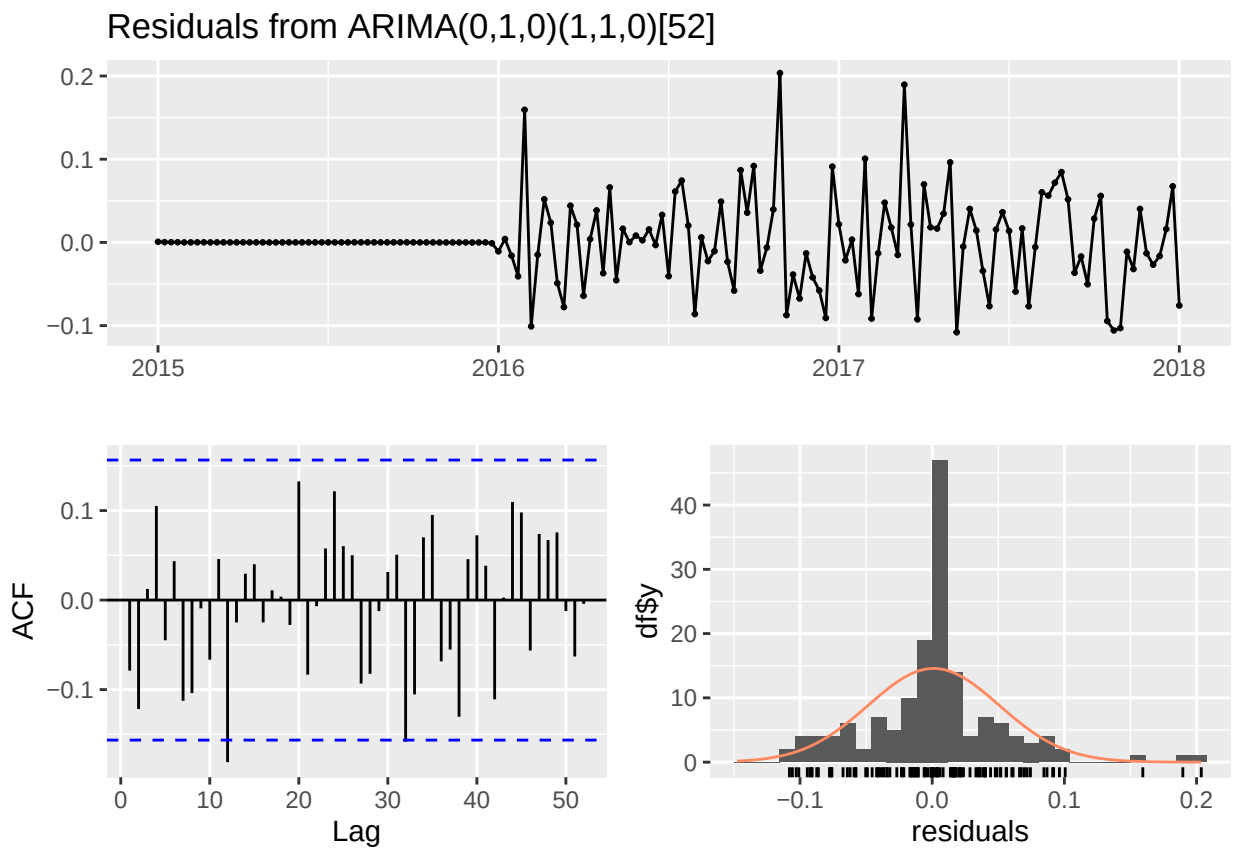
6) Diagnostic Tests :

```
# Checking residuals of the best chosen models
checkresiduals(arima_model3)
```



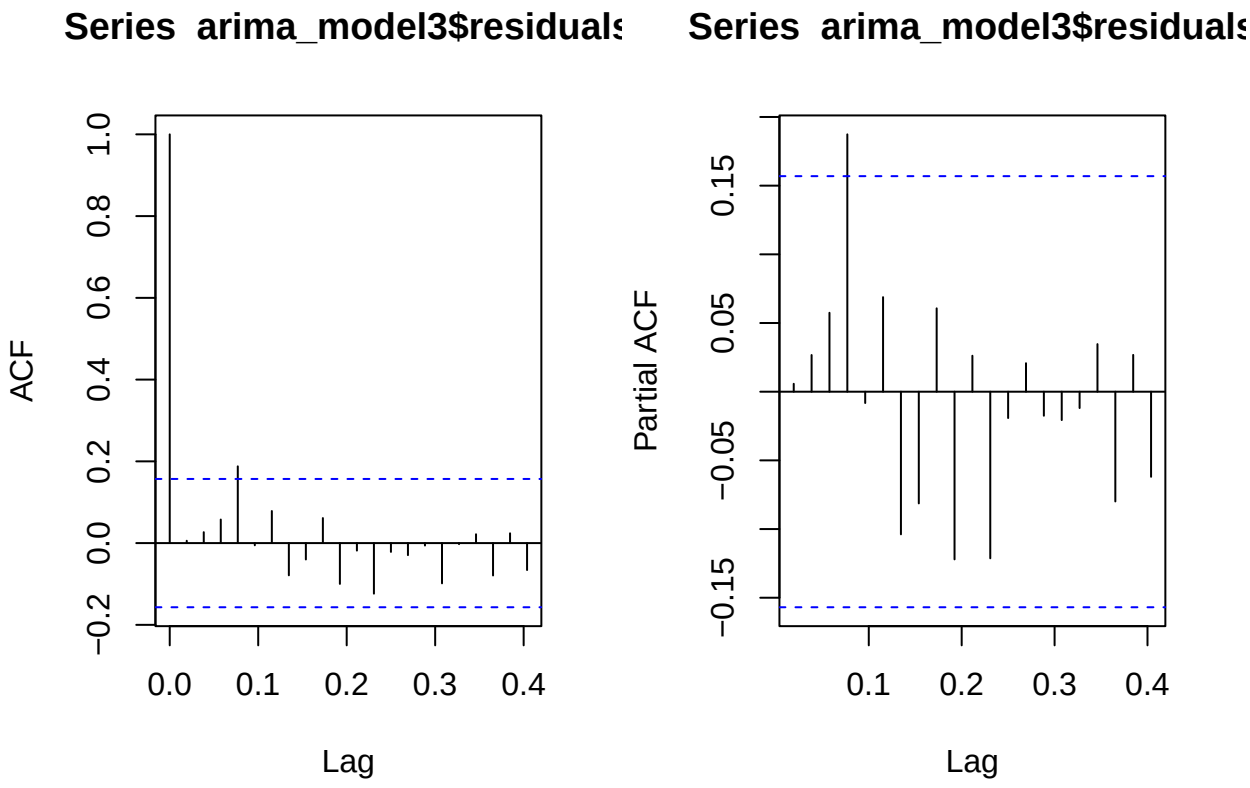

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)
## Q* = 26.714, df = 28, p-value = 0.5338
##
## Model df: 3.   Total lags used: 31
```

```
checkresiduals(auto_arima_model)
```

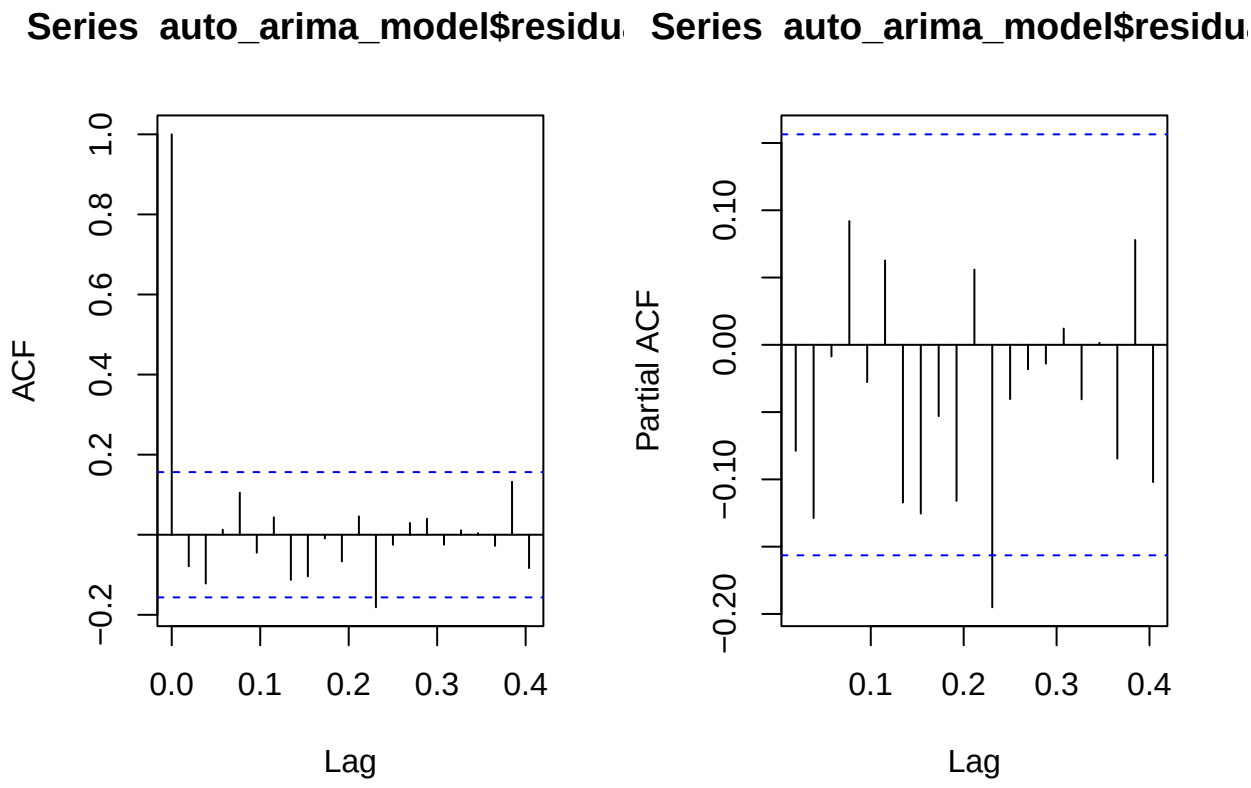



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)(1,1,0)[52]
## Q* = 30.106, df = 30, p-value = 0.4602
##
## Model df: 1.   Total lags used: 31
```

```
# Checking the residuals of the selected models using ACF/PACF
par(mfrow = c(1, 2))
acf(arima_model3$residuals)
pacf(arima_model3$residuals)
```




```
acf(auto_arima_model$residuals)
pacf(auto_arima_model$residuals)
```




```
# Stationarity test for ARIMA(2,1,1)
# Performing ADF test
adf_test1 <- adf.test(arima_model3$residuals)

# Performing Ljung-Box test
ljung_box_test1 <- Box.test(
arima_model3$residuals, lag = 10, type = "Ljung-Box")

# Results
print(adf_test1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  arima_model3$residuals
## Dickey-Fuller = -3.6759, Lag order = 5, p-value = 0.02862
## alternative hypothesis: stationary
```

```
print(ljung_box_test1)
```

```
##
##  Box-Ljung test
##
## data:  arima_model3$residuals
## X-squared = 11.008, df = 10, p-value = 0.3569
```

```
# Stationarity test for autoarima model
# Performing ADF test
adf_test_auto <- adf.test(auto_arima_model$residuals)
```

```
## Warning in adf.test(auto_arima_model$residuals): p-value smaller than printed
## p-value
```

```
# Performing Ljung-Box test
ljung_box_test_auto <- Box.test(
auto_arima_model$residuals, lag = 10, type = "Ljung-Box")

# Results
print(adf_test_auto)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  auto_arima_model$residuals
## Dickey-Fuller = -4.6527, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

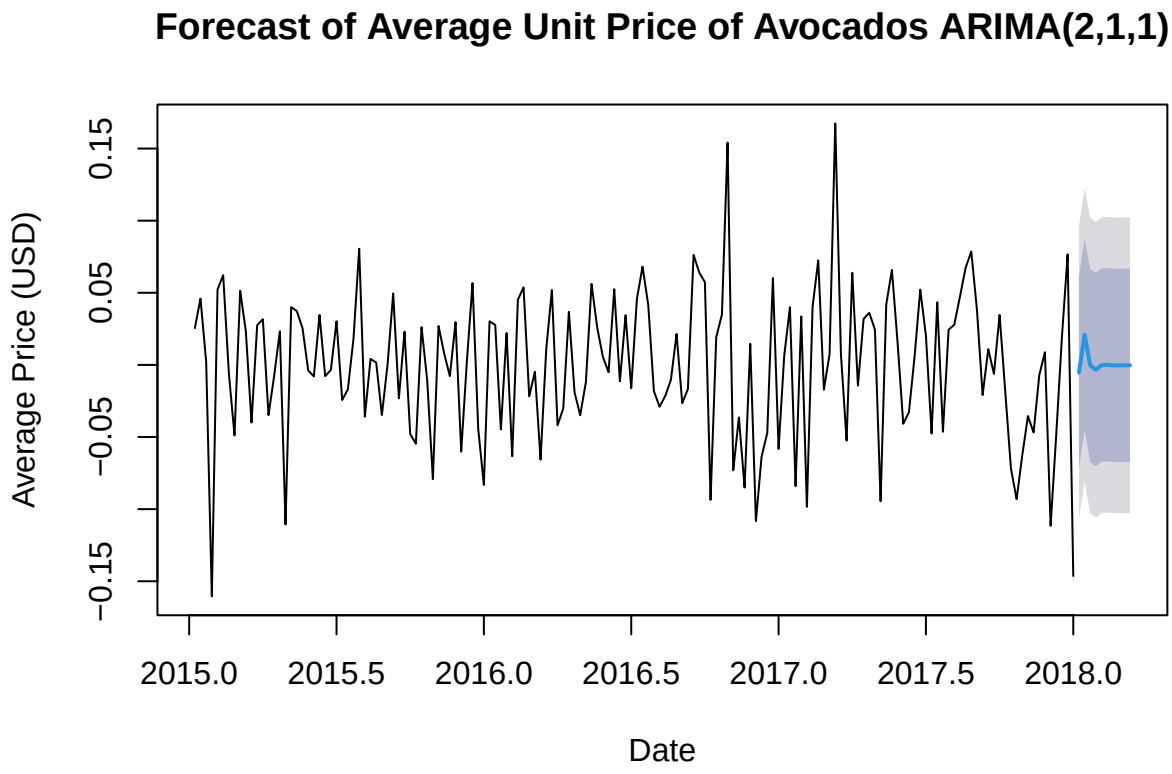
```
print(ljung_box_test_auto)
```

```
##
##  Box-Ljung test
##
## data:  auto_arima_model$residuals
## X-squared = 10.537, df = 10, p-value = 0.3947
```

7) Model Forecasting :

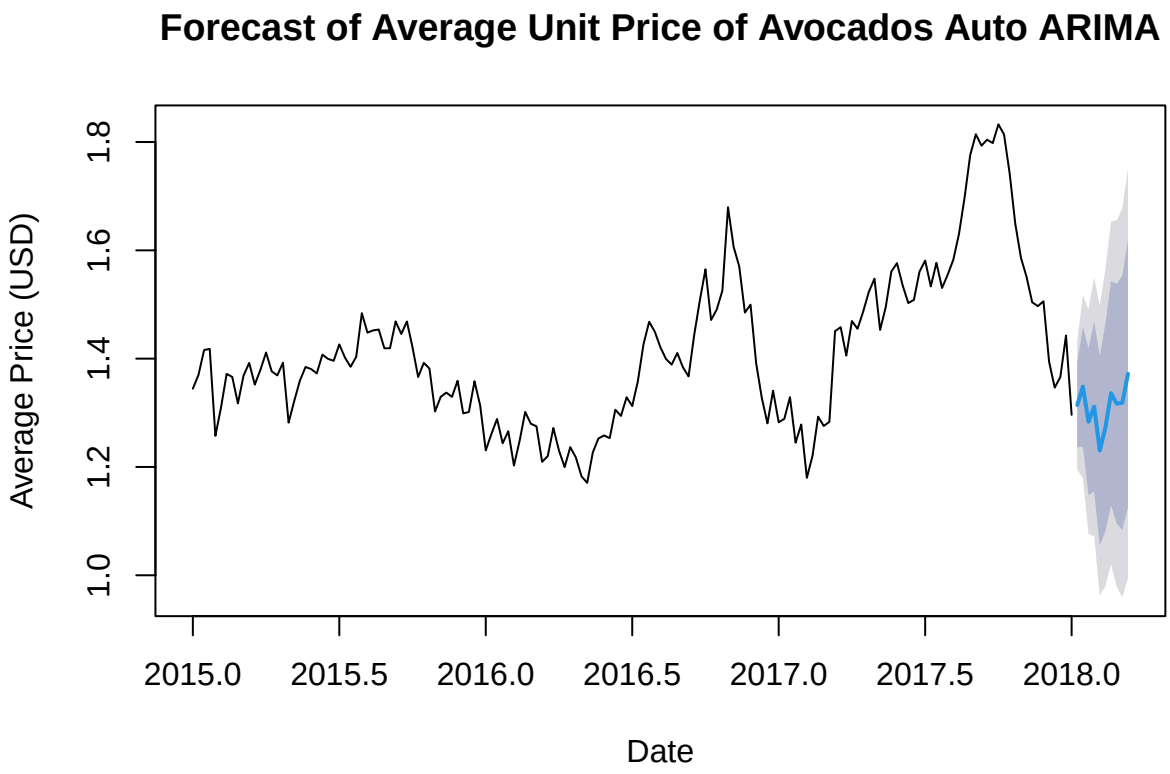
```
# Forecasting using ARIMA(2,1,1) model around mean 0
# Making forecasts for the first 10 weeks of 2018
forecast_result1 <- forecast(arima_model3, h = 10)
plot(forecast_result1,
```

```
     main = "Forecast of Average Unit Price of Avocados ARIMA(2,1,1)",
     xlab = "Date",
     ylab = "Average Price (USD)")
```




```
# Forecasting using autoarima model
# Making forecasts for the first 10 weeks of 2018
forecast_result2 <- forecast(auto_arima_model, h = 10)

plot(forecast_result2,
     main = "Forecast of Average Unit Price of Avocados Auto ARIMA",
     xlab = "Date",
     ylab = "Average Price (USD)")
```



8) Forecasting errors :

```
# Subseting the testing dataset to include only the first 10 values
testing_data_subset <- head(testing_data_imputed, 10)

# Calculate Mean Squared Error (MSE)
mse_arima <- mean(
(forecast_result1$mean - testing_data_subset$AveragePrice)^2)
mse_auto_arima <- mean(
(forecast_result2$mean - testing_data_subset$AveragePrice)^2)

# Calculate Root Mean Squared Error (RMSE)
rmse_arima <- sqrt(mse_arima)
rmse_auto_arima <- sqrt(mse_auto_arima)

print(paste("Mean Squared Error (MSE1) - ARIMA(2,1,1):", mse_arima))
```

```
## [1] "Mean Squared Error (MSE1) - ARIMA(2,1,1): 1.63276735731313"
```

```r
print(paste("Root Mean Squared Error (RMSE1) - ARIMA(2,1,1):", rmse_arima))
```

```
## [1] "Root Mean Squared Error (RMSE1) - ARIMA(2,1,1): 1.2777978546363"
```

```r
print(paste("Mean Squared Error (MSE2): - AutoArima", mse_auto_arima))
```

```
## [1] "Mean Squared Error (MSE2): - AutoArima 0.0456193287534009"
```

```r
print(paste("Root Mean Squared Error (RMSE2) - AutoArima:", rmse_auto_arima))
```

```
## [1] "Root Mean Squared Error (RMSE2) - AutoArima: 0.213586817836216"
```