US COFFEE PRICES PREDICTION AND ANALYSIS

Alejandra Gonzalez, Yihan Jia, Sisi Li, Ivana Zhao

Introduction

A coffee store is making its cost predictions for the next few years, and they want to forecast the coffee prices for a couple of years. So we need first to predict the ideal time to purchase ground coffee at the minimum price. Second, find which other variables and market trends coffee retailers could track to make commodity procurement decisions.

Many people have the idea of starting their own business after graduation, and shops with low start-up capital, such as coffee shops and milk tea shops, have become a popular choice. (Eliot Brown, 2022) However, there are many competitors in this industry. The shop owner needs to do a detailed cost analysis to win in the competition. Our analysis is exactly designed for those shop owners. They can use business analysis to make cost forecasts to maximize their profit.

Our project is based on time series analysis. The predicting methods we use here include Naive Method, Average Method, Drift Method, Simple Moving Average (SMA), Simple Exponential Smoothing (SES), Error Trend Seasonality (ETS), ARMA, ARIMA, SARIMA, Dynamic Linear Regression (DLR), Dynamic Harmonic Regression (DHR), and Neuro Network Method (NN). SMA, SES, and ETS are all smoothing methods that can help get rid of the irregularities in a time series. ARMA, ARIMA, and SARIMA forecast by extracting information from the ACF and PACF plots. Dynamic Linear Regression is similar to standard linear regression but with ARMA, ARIMA, or SARIMA error terms. Dynamic Harmonic Regression introduces the Fourier series and better predicts long seasonal periods.

Different prediction methods have different characteristics, so they are better at different predicting situations. We need to find the best one based on the unique characteristics of our data.

Data

We get data from the Federal Reserve Bank of St. Louis and Berkeley Earth dataset. The Federal Reserve Bank of St. Louis offers many U.S. and international time-series data. We get the data of monthly coffee price, coffee producer price index, tea producer price index, and exchange rate from it. The Berkeley Earth averaging process generates various Output data, including a set of gridded temperature fields, regional averages, and bias-corrected station data. We get the data of temperature from it. On Table 1 you can see all the variables and their descriptions.

Overview

1) Due to the small seasonal data component, we first aggregate the monthly coffee price into quarterly data to analyze.
2) Between all forecasting methods, we found Dynamic Harmonic Regression (DHR) method has the least MAPE in predicting coffee price. So we choose the DHR method to do the forecasting.
3) Comparing the results between predicting one year and two years, we find our predictions are more accurate for a one-year time horizon.

Table 1: Variable definition

Data name	Description	What is it used for
Monthly coffee price (Average Price: Coffee, 100%, Ground Roast, All Sizes - Cost per Pound/453.6 Grams in U.S. City Average)	Monthly data of average coffee price in U.S. city average from 1980 to 2021	To predict the trend of coffee price to minimize cost
Temperature	Monthly data of land temperature from 1980 to 2021	To analyze whether land temperature has a major effect on coffee price. We analyze this variance because land temperature will affect coffee yield
Coffee producer price index (PPI) (Producer Price Index by Commodity: Processed Foods and Feeds: Coffee, Concentrated, Including Coffee Substitutes)	Monthly data of PPI by Commodity with coffee concentrated from 1980 to 2021 (index 1982=100)	To analyze whether coffee PPI has major effect on coffee price
Tea producer price index (Producer Price Index by Commodity: Processed Foods and Feeds: Tea in Consumer Packages, Excluding Canned Ice Tea)	Monthly data of PPI by Commodity with tea concentrated from 1990 to 2021	To analyze whether tea PPI has major effect on coffee price
Brazilian Reals to U.S. Dollar Exchange Rate (Brazilian Reals to U.S. Dollar Spot Exchange Rate)	Monthly data of Brazilian Reals to U.S. Dollar Spot Exchange Rate from 1980 to 2021	To analyze whether Brazilian Reals to U.S. Dollar Exchange Rate has a major effect on coffee price. We choose Brazilian Reals because Brazil is a major place of production of coffee.

4) Check the difference between predicting with outliers and without outliers.

5) Check the residuals and find that they are white noise series.

Data Analysis and Findings

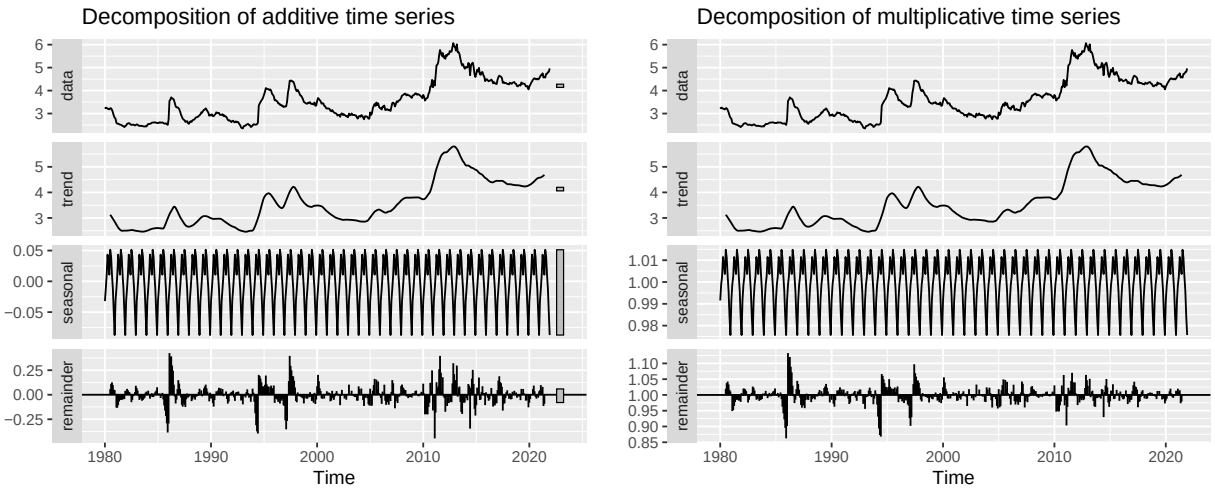


Figure 1: Decomposition of the time series

The ground coffee price is registered monthly. Figure 1 shows the preliminary analysis of this data. The trend component suggests a positive long-term path, and the seasonal component seems to be small. Testing different models, we realized the nature of the data does not capture any trend or seasonality on the data. To capture some of these components, we aggregate the data in quarters. We calculate the average of the variable during the three months of each quarter.

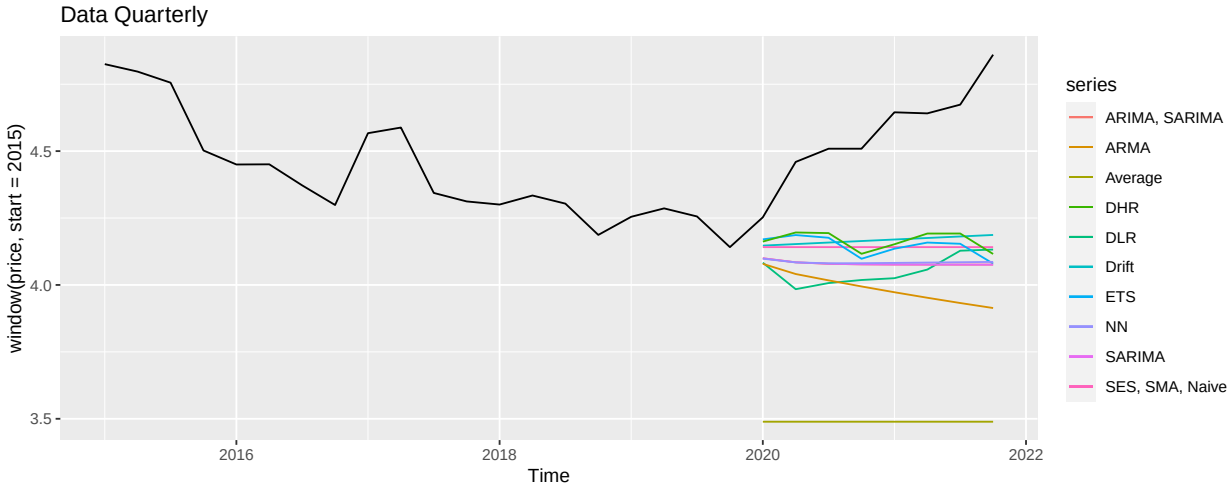


Figure 2: Two years Forecasting for Test Data Set

This initial analysis shows that our time series does not have an evident seasonality. We test different models

and confirm that the model barely captures any seasonality or trend. Figure 2 shows the forecasting for two years in the testing data for the quarterly data (these results are equivalent using monthly data). We generally see that the forecasting tends to be a flat line.

We checked different models and concluded using the quarterly data utilizing the Dynamic Harmonic Regression (DHR). First, we estimate different models using the monthly data and testing in two different horizons of time, one year and two years. Second, we repeated the first step but with quarterly data. Table 2 shows the Mean Absolute Percentage Error (MAPE) of all the tries.

Table 2: Mean Absolute Percentage Error

	Monthly Data		Quarterly Data	
Method	Two years	One year	Two years	One year
Naive	11.1541	3.8759	9.2321	4.1310
Average	23.0251	24.8461	23.5301	25.3290
Drift	10.6802	3.5098	8.6818	3.7195
Simple Moving Average	11.1541	3.8759	9.2321	4.1310
Simple Exponential Smoothing	11.1537	3.8760	9.2319	4.1310
Error Trend Seasonality	13.5427	3.4945	9.1400	2.2189
ARMA	14.5037	5.2776	12.5567	6.0251
ARIMA	11.7092	3.8376	10.5597	4.1310
SARIMA	11.9953	3.2533	10.5597	4.1310
Dynamic Linear Regression	15.1210	4.8697	11.1544	4.2146
Dynamic Harmonic Regression	10.1434	2.3946	8.7066	2.1793
Neuro Network	13.5914	4.5954	10.1970	3.5233

The Dynamic Linear Regression (DLR) model included external variables proxies for temperature, costs, and price of a substitute product. Specifically, we included the land temperature anomalies in celsius, the producer price index of coffee in the United States, the exchange rate between the Dollar and the Real (because Brazil is the most important exporter of coffee to the U.S.), and the producer price of the tea in the U.S.

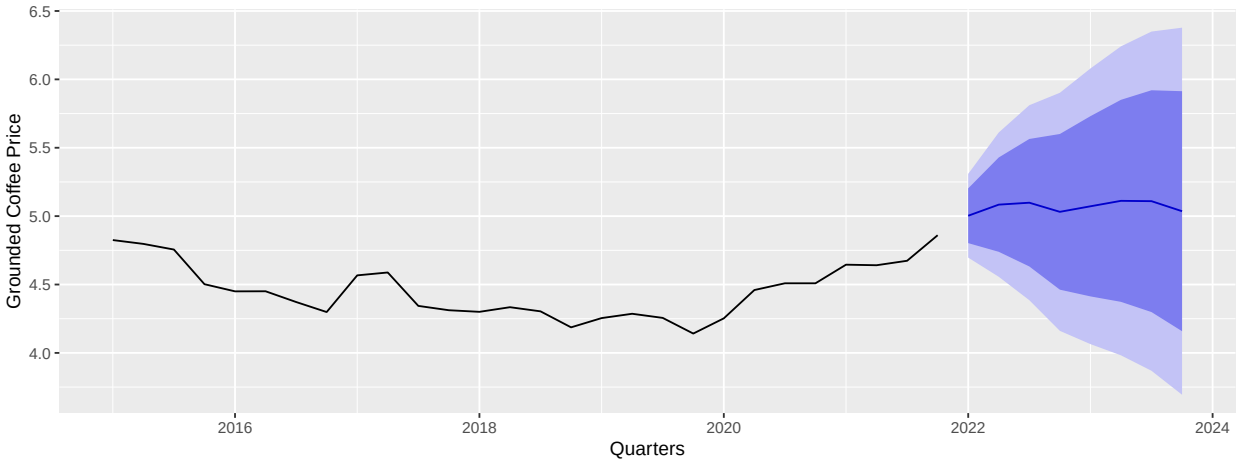


Figure 3: Forecasting Grounded Coffee Price

After looking at the results of each method, we concluded that the best model for forecasting the price of the

ground coffee is the DHR. We find the model is more precise in predicting the expected price average when we use quarterly data. These predictions are more accurate for one year horizon time. Figure 3 presents our two years forecasting of the grounded coffee price.

Considering that a dynamic regression with Fourier terms is often better with long seasonal periods, we tested the DHR model in the monthly and quarterly data and the two horizons of time we tested. Comparing the AIC criteria, we find the best option in the monthly data was to use four Fourier terms and two in the case of the quarterly data.

Based on these results, *our suggestion for the coffee shop is to buy the coffee the first and last quarter of each year.*

Verify Model Fit and Assumptions

Based on the graph of the original coffee price data, we can still see a little seasonality in the plot, though not very obvious. Therefore, it makes sense that the DHR is the best model to fit since it could capture seasonality from the data.

Outliers

In order to check if there are outliers in the original time series data of price, our first step is to implement na.interp() function to replace missing values with numbers, then we apply tsoutliers() function to identify outliers. It turns out that there are seven outliers, so we used the tsclean() function to replace outliers. Finally, we plot the forecasting results in the DHR model with and without outliers respectively. Based on the Figure 4, it seems that the forecasting line with outliers and without outliers are pretty similar, which shows outliers don't influence our final decision on the time to purchase coffee at the ideal lowest price.

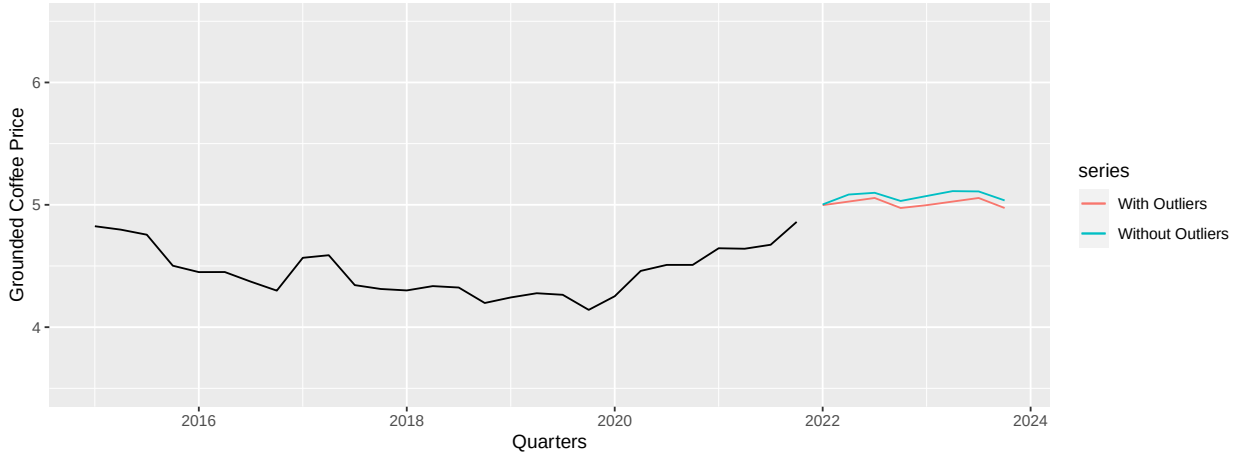


Figure 4: Forecasting with Outliers and without Outliers

Residual plot

We have plotted the residuals for the DHR model with two Fourier terms (Figure 5). Because P-value = 0.2719, which is larger than significance level 0.05, we fail to reject the null hypothesis, indicating residuals from this model are white noise series and this model succeeds in extracting all important information in the data.

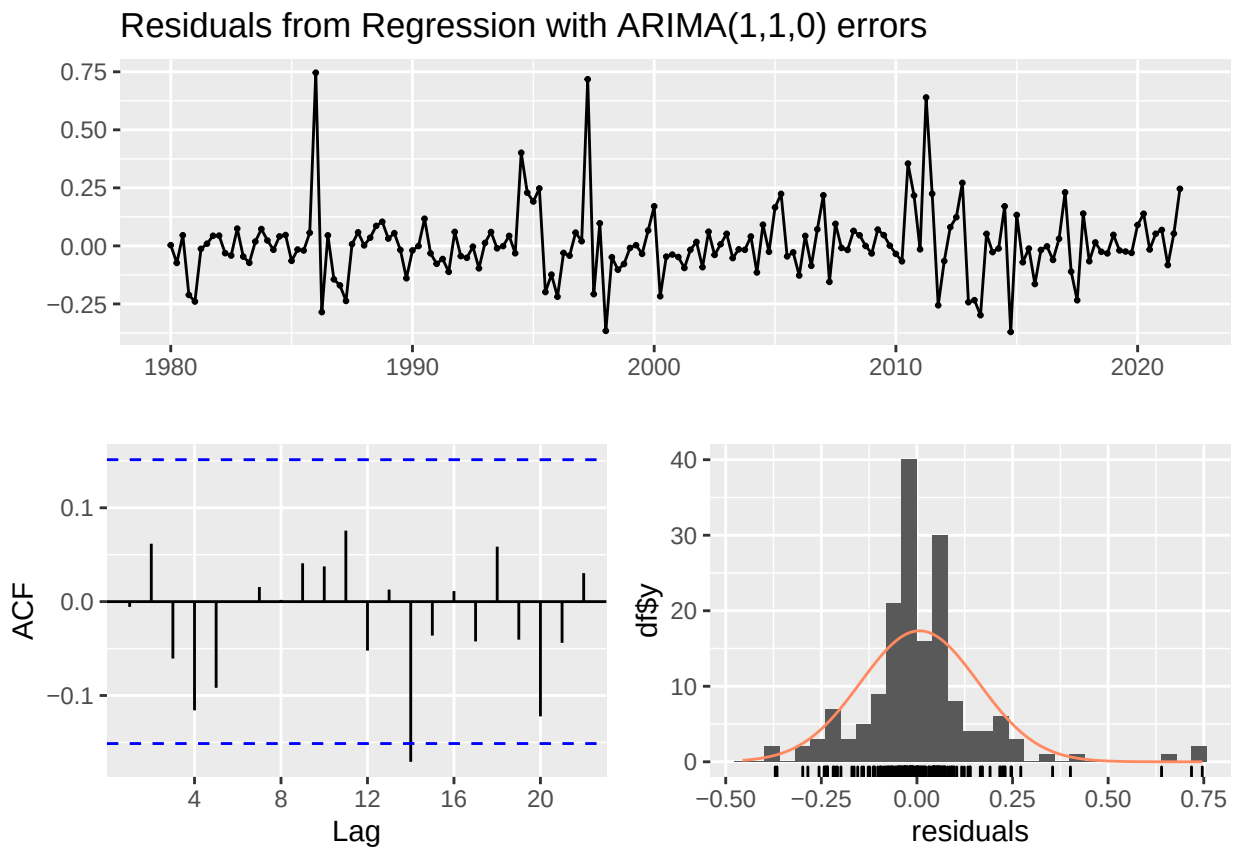


Figure 5: Residuals Plot

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,1,0) errors
## Q* = 5.1536, df = 4, p-value = 0.2719
##
## Model df: 4.   Total lags used: 8
```

Summary and Conclusions

Comparing with typical large franchising group or brand stores, SME (small and medium-sized enterprises) are known for vulnerable from many external factors such as the market fluctuation, macro economy and competitor landscape. Among many difficulties, lacking of Capital&Cash Flow are one of the most outstanding issue. On top of that, cost of goods sold is the most direct and not readily soluble (Warnes, Bryce, 2020) because that coffee grounds is linked to coffee bean, which is more of a commodity with a quite inelastic demand (Chayka, Kyle, 2014) and because that COGS has the tendency to be the 'largest number' according to Benford's Law (Craven, Jo, 2020).

Hence, to help SME in coffee business to solve the issues around COGS, we have collected the historical data from commodity, substitution, natural condition and economic indicator as possible coefficient and applied 12 forecasting model. After adjusting the data interval, we concluded that DHR is the most appropriate model. As a suggestion to SME in coffee business, the price of ground coffee in US next two years should be around 5 to 5.3 USD with a slight upward trend. In practical, coffee shop owner can use this price as a benchmark

to calculate the cost and maxmize the profit.

However, as our professor mentioned at the first class, forecasting is not a crystal ball. The risk of relying too much on forecasting includes opportunity cost and less responsivenessto market change. To mitigate these two risks, SME in coffee business can focus on the diversifying the product lines, enlarging the intangible income resource (e.g. customer service) and improving the fitting of customer segmentation.

Appendix

References

Brown, Eliot. "A Booming Startup Market Prompts an Investment Rush for Ever-Younger Companies." Wall Street Journal, 2 January 2022, https://www.wsj.com/articles/a-booming-startup-market-prompts-an-investment-rush-for-ever-younger-companies-11641119403. Accessed 24 February 2022.

Chayka, Kyle. "Coffee Shortages Won't Change the Price of Your Frappuccino." Pacific Standard, 30 July 2014, https://psmag.com/economics/coffee-shortages-wont-change-price-frappuccino-87107. Accessed 24 February 2022.

Warnes, Bryce. "How to Calculate Cost of Goods Sold." Bench Accounting, 13 February 2020, https://bench.co/blog/accounting/cost-of-goods-sold/. Accessed 24 February 2022.

Craven, Jo. "Can an Accounting Tool Detect Election Fraud?" The Wall Street Journal, 4 December 2020, https://www.wsj.com/articles/can-accounting-tool-detect-election-fraud-11607077800. Accessed 24 February 2022.

Data links

"Average Price: Coffee, 100%, Ground Roast, All Sizes (Cost per Pound/453.6 Grams) in U.S. City Average." FRED, https://fred.stlouisfed.org/series/APU0000717311. Accessed 24 February 2022. "Brazilian Reals to U.S. Dollar Spot Exchange Rate." FRED, https://fred.stlouisfed.org/series/DEXBZUS. Accessed 24 February 2022.

"Producer Price Index by Commodity: Processed Foods and Feeds: Coffee, Concentrated, Including Coffee Substitutes." FRED, https://fred.stlouisfed.org/series/WPU02630103. Accessed 24 February 2022.

"Producer Price Index by Commodity: Processed Foods and Feeds: Tea in Consumer Packages, Excluding Canned Ice Tea." FRED, https://fred.stlouisfed.org/series/WPU02630313. Accessed 24 February 2022.

"Temperatures are in Celsius and reported as anomalies relative to the Jan 1951-Dec 1980 % average." http://berkeleyearth.org/data/

Code

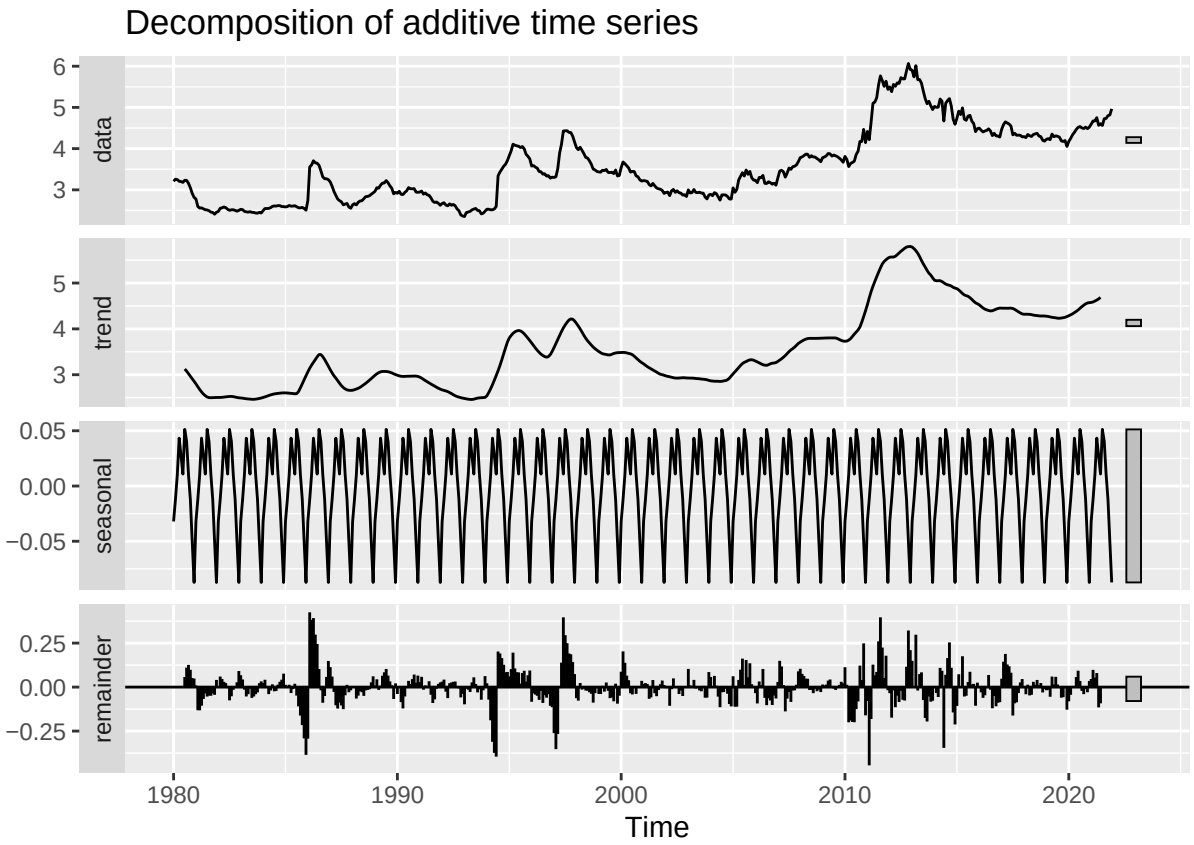
Loading data and packages

```
#Load packages
library(gridExtra)
library(smooth)
library(fpp2)
library(zoo)
library(dygraphs)
library(readxl)
#Reading the data
coffeeprice <- read_excel("coffee (1).xlsx",
                          skip = 10)
#change the columns names
```

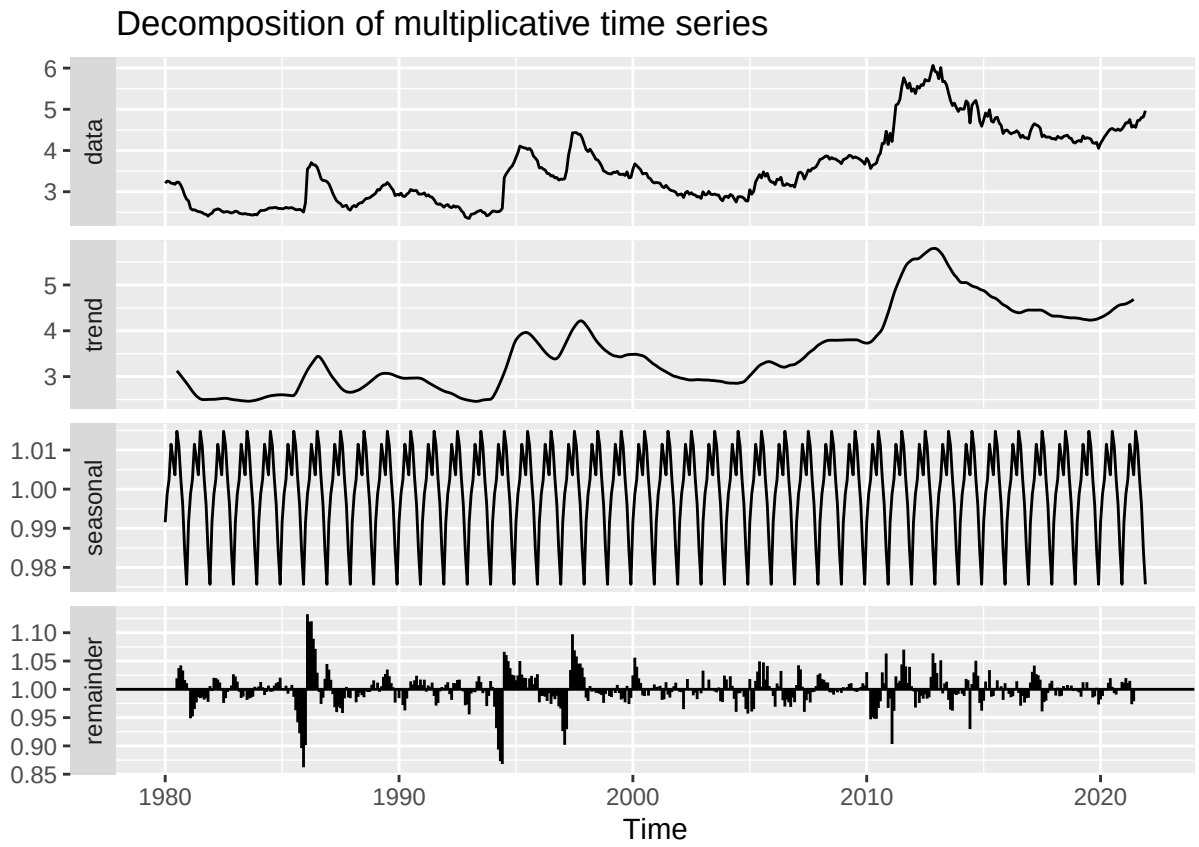
```
names(coffeeprice)<-c("date","price","ppi_coffee","ppi_tea","bean_price",
                      "temperature","er_real")

#define the data as a ts object
tsdata = ts(coffeeprice,start=1980,frequency = 12)
```

Code - Using the monthly data

```
# Monthly data
price<-tsclean(tsdata[,"price"])
ppi_coffee<-tsclean(tsdata[,"ppi_coffee"])
ppi_tea<-tsclean(tsdata[,"ppi_tea"])
bean_price<-tsclean(tsdata[,"bean_price"])
temperature<-tsclean(tsdata[,"temperature"])
er_real<-tsclean(tsdata[,"er_real"])
autoplot(decompose(price, "additive"))
```

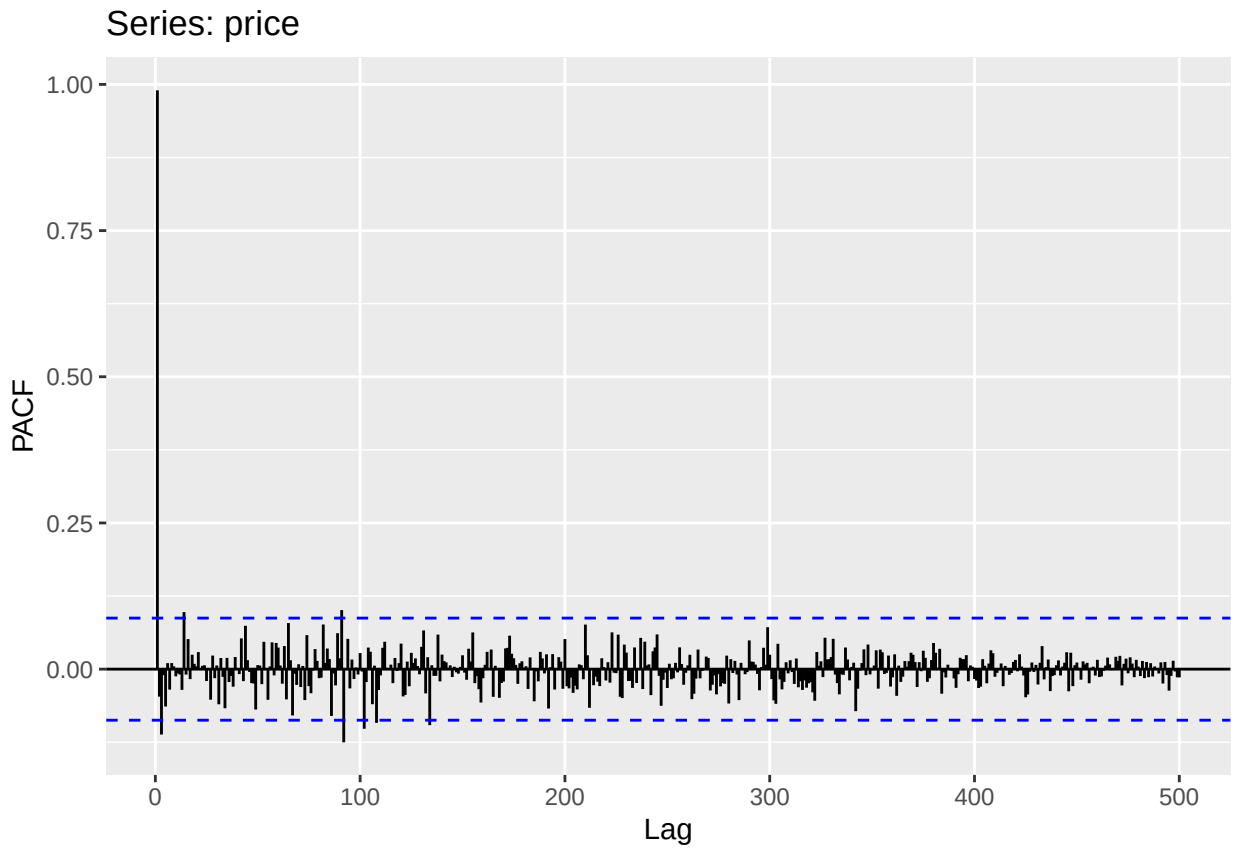



```
autoplot(decompose(price, "multiplicative"))
```



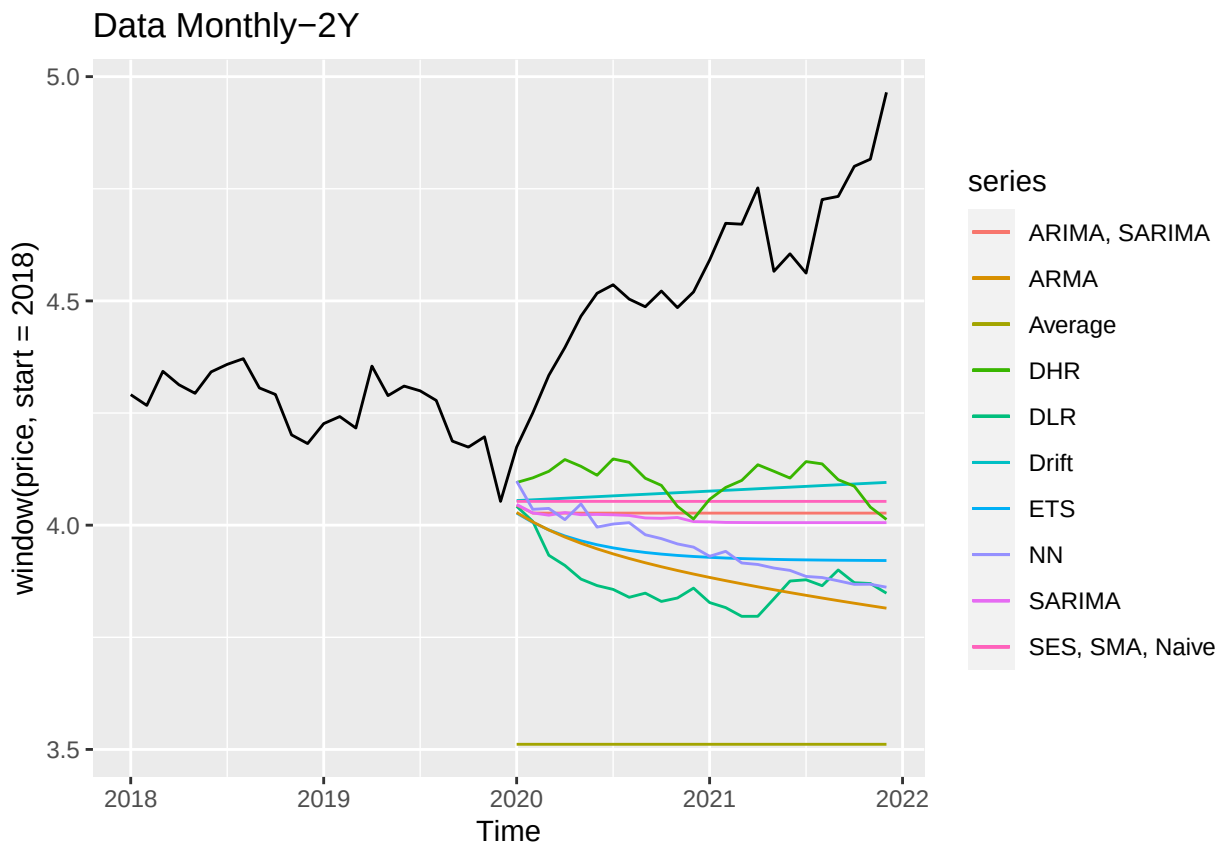

```
ggAcf(price, lag.max = 100)
```


```
ggPacf(price, lag.max = 500)
```




```
## Training the model without the last 2 years:

train_price <- window(price, end = 2020-1/12)
test_price <- window(price, start = 2020)
flength<-length(test_price)
XREG <- cbind(ppi_coffee,ppi_tea,bean_price,temperature,er_real)
train_XREG = window(XREG, end = 2020-1/12)
test_XREG  = window(XREG, start = 2020)

#training & forecasting naive
ts.naive = naive(train_price, h=flength)
#training & forecasting average
ts.ave   = meanf(train_price, h=flength)
#training & forecasting drift
ts.drift = rwf(train_price, drift=TRUE, h=flength)
#training & forecasting simple moving avg
ts.sma   = sma(train_price, h=flength)
#training & forecasting simple exponential smoothing
ts.ses   = ses(train_price, h=flength)
#training & forecasting ets: error+trend+seasonality
fit.ets = forecast::ets(train_price)
ts.ets = forecast::forecast.ets(fit.ets, h=flength)
#training & forecasting arma
fit.arma = auto.arima(train_price, d=0, seasonal=FALSE)
ts.arma  = forecast::forecast(fit.arma, h=flength)
#training & forecasting arima
```

```
fit.arima = auto.arima(train_price, seasonal=FALSE)
ts.arima  = forecast::forecast(fit.arima, h=flength)
#training & forecasting sarima
fit.sarima = auto.arima(train_price)
ts.sarima = forecast::forecast(fit.sarima, h=flength)
#training dynamic linear reg
dlr.fit = auto.arima(train_price, xreg = train_XREG)
#forecasting dynamic linear reg
dlr.fc = forecast::forecast(dlr.fit, xreg = test_XREG )
fit4 = auto.arima(train_price, xreg = fourier(train_price, K=4), seasonal=FALSE)
fc4  = forecast(fit4, xreg = fourier(train_price, K=4, h = flength))
fit_nn = nnetar(train_price)
price.nn= forecast(fit_nn, h = flength)

autoplot(window(price, start = 2018)) +
  ggtitle("Data Monthly-2Y")+
  autolayer(dlr.fc$mean, series="DLR")+
  autolayer(ts.drift$mean, series="Drift") +
  autolayer(ts.ses$mean, series="SES, SMA, Naive")+
  autolayer(ts.ets$mean, series="ETS")+
  autolayer(ts.arma$mean, series="ARMA")+
  autolayer(ts.arima$mean, series="ARIMA, SARIMA") +
  autolayer(ts.sarima$mean, series="SARIMA")+
  autolayer(ts.ave$mean, series="Average")+
  autolayer(fc4$mean, series="DHR")+
  autolayer(price.nn$mean, series = "NN")
```




```
mape.naive = accuracy(ts.naive$mean, test_price)[5]
mape.ave = accuracy(ts.ave$mean, test_price)[5]
mape.drift = accuracy(ts.drift$mean, test_price)[5]
mape.sma = accuracy(ts.sma$forecast, test_price)[5]
mape.ses = accuracy(ts.ses$mean, test_price)[5]
mape.ets = accuracy(ts.ets$mean, test_price)[5]
mape.arma = accuracy(ts.arma$mean, test_price)[5]
mape.arima = accuracy(ts.arima$mean, test_price)[5]
mape.sarima = accuracy(ts.sarima$mean, test_price)[5]
mape.dlr = accuracy(dlr.fc$mean, test_price)[5]
mape.dhr = accuracy(fc4$mean, test_price)[5]
mape.nn = accuracy(price.nn$mean, test_price)[5]



pfit1 = auto.arima(train_price, xreg = fourier(train_price, K=1), seasonal=FALSE)
pfc1  = forecast(pfit1, xreg = fourier(train_price, K=1, h = flength))
pfit2 = auto.arima(train_price, xreg = fourier(train_price, K=2), seasonal=FALSE)
pfc2  = forecast(pfit2, xreg = fourier(train_price, K=2, h = flength))
pfit3 = auto.arima(train_price, xreg = fourier(train_price, K=3), seasonal=FALSE)
pfc3  = forecast(pfit3, xreg = fourier(train_price, K=3, h = flength))
pfit4 = auto.arima(train_price, xreg = fourier(train_price, K=4), seasonal=FALSE)
pfc4  = forecast(pfit4, xreg = fourier(train_price, K=4, h = flength))
pfit5 = auto.arima(train_price, xreg = fourier(train_price, K=5), seasonal=FALSE)
pfc5  = forecast(pfit5, xreg = fourier(train_price, K=5, h = flength))
pfit6 = auto.arima(train_price, xreg = fourier(train_price, K=6), seasonal=FALSE)
```

```
pfc6  = forecast(pfit6, xreg = fourier(train_price, K=6, h = flength))

# aicc DHR - 2 years
pfit2$aicc
```

```
## [1] -781.1486
```

```
pfit1$aicc
```

```
## [1] -773.9756
```

```
pfit3$aicc
```

```
## [1] -777.1101
```

```
pfit4$aicc
```

```
## [1] -789.0683
```

```
pfit5$aicc
```

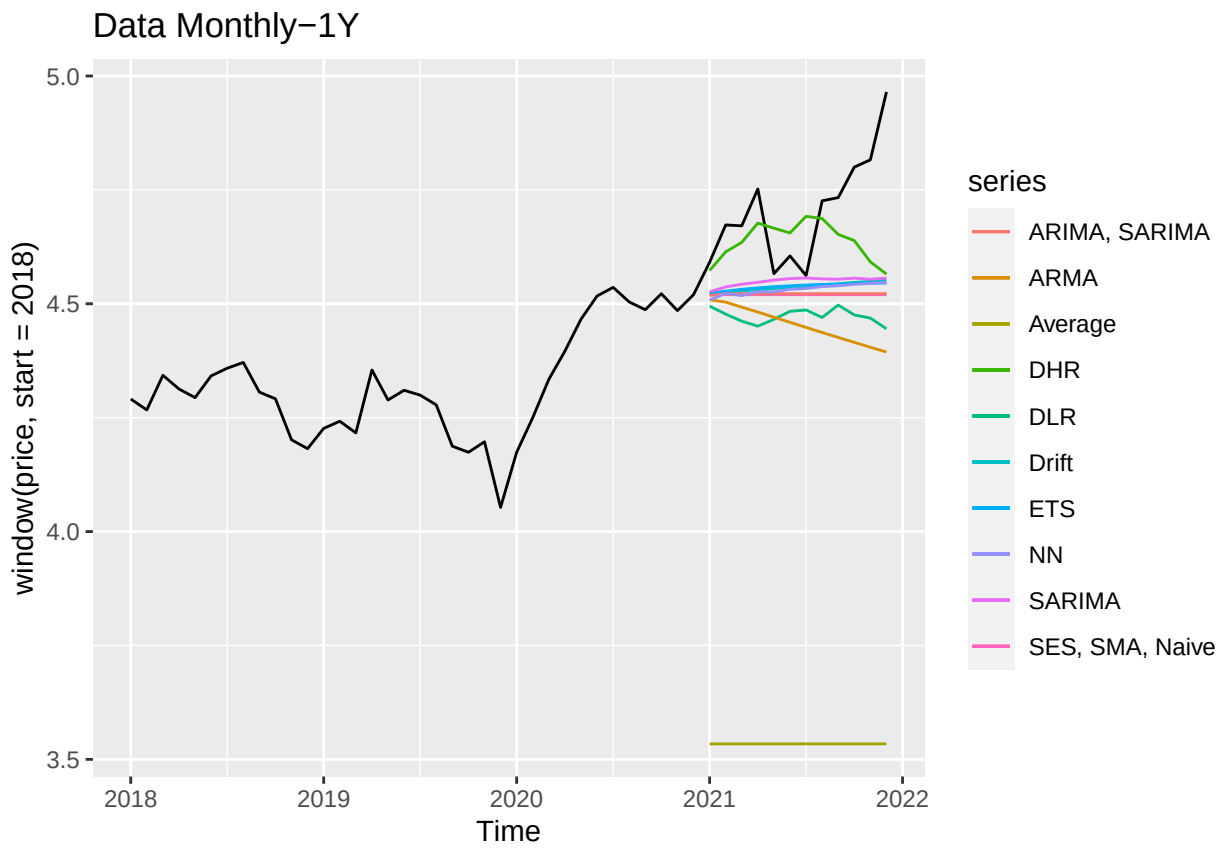
```
## [1] -785.2138
```

```
pfit6$aicc
```

```
## [1] -784.0333
```

```
## Training the model without the last year:

train_price <- window(price, end = 2021-1/12)
test_price <- window(price, start = 2021)
flength<-length(test_price)
XREG <- cbind(ppi_coffee,ppi_tea,bean_price,temperature,er_real)
train_XREG = window(XREG, end = 2021-1/12)
test_XREG  = window(XREG, start = 2021)

#training & forecasting naive
ts.naive = naive(train_price, h=flength)
#training & forecasting average
ts.ave   = meanf(train_price, h=flength)
#training & forecasting drift
ts.drift = rwf(train_price, drift=TRUE, h=flength)
#training & forecasting simple moving avg
ts.sma   = sma(train_price, h=flength)
#training & forecasting simple exponential smoothing
ts.ses   = ses(train_price, h=flength)
#training & forecasting ets: error+trend+seasonality
fit.ets = forecast::ets(train_price)
ts.ets = forecast::forecast.ets(fit.ets, h=flength)
#training & forecasting arma
fit.arma = auto.arima(train_price, d=0, seasonal=FALSE)
ts.arma  = forecast::forecast(fit.arma, h=flength)
#training & forecasting arima
fit.arima = auto.arima(train_price, seasonal=FALSE)
ts.arima  = forecast::forecast(fit.arima, h=flength)
#training & forecasting sarima
fit.sarima = auto.arima(train_price)
ts.sarima = forecast::forecast(fit.sarima, h=flength)
```

```
#training dynamic linear reg
dlr.fit = auto.arima(train_price, xreg = train_XREG)
#forecasting dynamic linear reg
dlr.fc = forecast::forecast(dlr.fit, xreg = test_XREG )
fit41 = auto.arima(train_price, xreg = fourier(train_price, K=4), seasonal=FALSE)
fc4  = forecast(fit41, xreg = fourier(train_price, K=4, h = flength))
fit_nn = nnetar(train_price)
price.nn= forecast(fit_nn, h = flength)

autoplot(window(price, start = 2018)) +
  ggtitle("Data Monthly-1Y")+
  autolayer(dlr.fc$mean, series="DLR")+
  autolayer(ts.drift$mean, series="Drift") +
  autolayer(ts.ses$mean, series="SES, SMA, Naive")+
  autolayer(ts.ets$mean, series="ETS")+
  autolayer(ts.arma$mean, series="ARMA")+
  autolayer(ts.arima$mean, series="ARIMA, SARIMA") +
  autolayer(ts.sarima$mean, series="SARIMA")+
  autolayer(ts.ave$mean, series="Average")+
  autolayer(fc4$mean, series="DHR")+
  autolayer(price.nn$mean, series = "NN")
```




```
mape.naive1 = accuracy(ts.naive$mean, test_price)[5]
mape.ave1 = accuracy(ts.ave$mean, test_price)[5]
mape.drift1 = accuracy(ts.drift$mean, test_price)[5]
mape.sma1 = accuracy(ts.sma$forecast, test_price)[5]
```

```
mape.ses1 = accuracy(ts.ses$mean, test_price)[5]
mape.ets1 = accuracy(ts.ets$mean, test_price)[5]
mape.arma1 = accuracy(ts.arma$mean, test_price)[5]
mape.arima1 = accuracy(ts.arima$mean, test_price)[5]
mape.sarima1 = accuracy(ts.sarima$mean, test_price)[5]
mape.dlr1 = accuracy(dlr.fc$mean, test_price)[5]
mape.dhr1 = accuracy(fc4$mean, test_price)[5]
mape.nn1 = accuracy(price.nn$mean, test_price)[5]


pfit11 = auto.arima(train_price, xreg = fourier(train_price, K=1), seasonal=FALSE)
pfc1  = forecast(pfit11, xreg = fourier(train_price, K=1, h = flength))
pfit21 = auto.arima(train_price, xreg = fourier(train_price, K=2), seasonal=FALSE)
pfc2  = forecast(pfit21, xreg = fourier(train_price, K=2, h = flength))
pfit31 = auto.arima(train_price, xreg = fourier(train_price, K=3), seasonal=FALSE)
pfc3  = forecast(pfit31, xreg = fourier(train_price, K=3, h = flength))
pfit41 = auto.arima(train_price, xreg = fourier(train_price, K=4), seasonal=FALSE)
pfc4  = forecast(pfit41, xreg = fourier(train_price, K=4, h = flength))
pfit51 = auto.arima(train_price, xreg = fourier(train_price, K=5), seasonal=FALSE)
pfc5  = forecast(pfit51, xreg = fourier(train_price, K=5, h = flength))
pfit61 = auto.arima(train_price, xreg = fourier(train_price, K=6), seasonal=FALSE)
pfc6  = forecast(pfit61, xreg = fourier(train_price, K=6, h = flength))

# aicc DHR - 1 year
pfit21$aicc
```

```
## [1] -810.4739
```

```
pfit11$aicc
```

```
## [1] -802.9449
```

```
pfit31$aicc
```

```
## [1] -806.3827
```

```
pfit41$aicc
```

```
## [1] -818.9163
```

```
pfit51$aicc
```

```
## [1] -815.0508
```

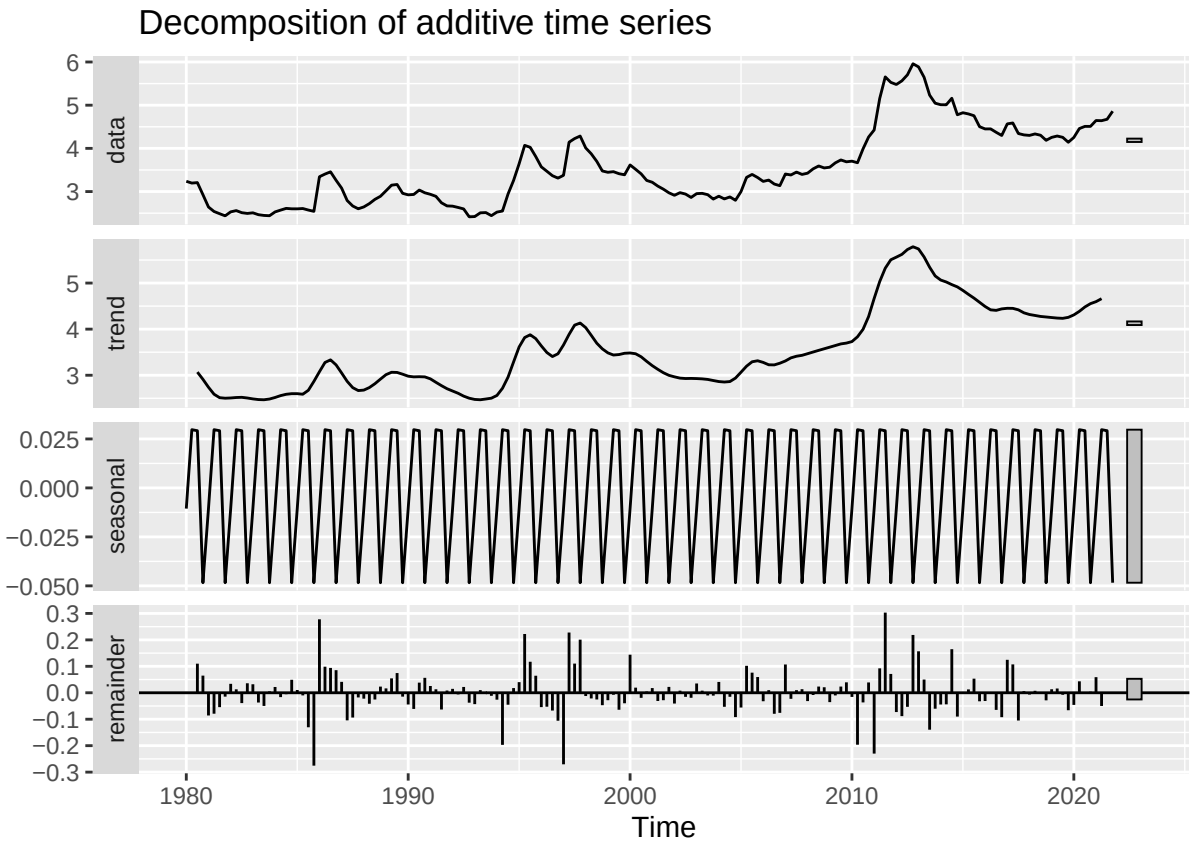
```
pfit61$aicc
```

```
## [1] -813.8015
```

Code - Using the quarterly data

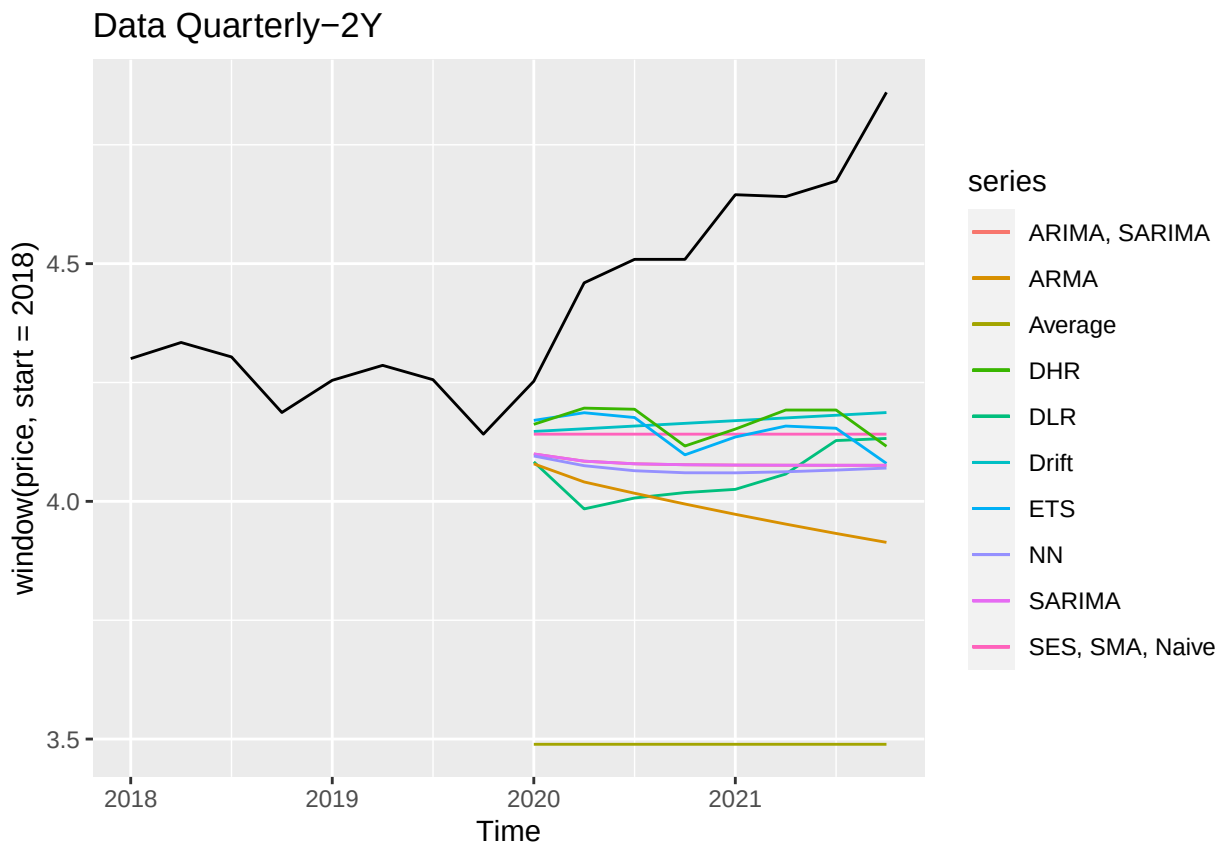
```
# Quarterly data
price<-tsclean(aggregate(tsdata[,"price"], nfrequency = 4, mean))
ppi_coffee<-tsclean(aggregate(tsdata[,"ppi_coffee"], nfrequency = 4, mean))
ppi_tea<-tsclean(aggregate(tsdata[,"ppi_tea"], nfrequency = 4, mean))
bean_price<-tsclean(aggregate(tsdata[,"bean_price"], nfrequency = 4, mean))
temperature<-tsclean(aggregate(tsdata[,"temperature"], nfrequency = 4, mean))
```

```
er_real<-tsclean(aggregate(tsdata[,"er_real"], nfrequency = 4, mean))
autoplot(decompose(price))
```




```
d1q = ggAcf(price, lag.max = 100)
d2q = ggPacf(price, lag.max = 500)

## Training the model without the last 2 years:

train_price <- window(price, end = 2020-1/4)
test_price <- window(price, start = 2020)
flength<-length(test_price)
XREG <- cbind(ppi_coffee,ppi_tea,bean_price,temperature,er_real)
train_XREG = window(XREG, end = 2020-1/4)
test_XREG  = window(XREG, start = 2020)

#training & forecasting naive
ts.naive = naive(train_price, h=flength)
#training & forecasting average
ts.ave   = meanf(train_price, h=flength)
#training & forecasting drift
ts.drift = rwf(train_price, drift=TRUE, h=flength)
#training & forecasting simple moving avg
ts.sma   = sma(train_price, h=flength)
#training & forecasting simple exponential smoothing
ts.ses   = ses(train_price, h=flength)
#training & forecasting ets: error+trend+seasonality
```

```
fit.ets = forecast::ets(train_price)
ts.ets = forecast::forecast.ets(fit.ets, h=flength)
#training & forecasting arma
fit.arma = auto.arima(train_price, d=0, seasonal=FALSE)
ts.arma  = forecast::forecast(fit.arma, h=flength)
#training & forecasting arima
fit.arima = auto.arima(train_price, seasonal=FALSE)
ts.arima  = forecast::forecast(fit.arima, h=flength)
#training & forecasting sarima
fit.sarima = auto.arima(train_price)
ts.sarima = forecast::forecast(fit.sarima, h=flength)
#training dynamic linear reg
dlr.fit = auto.arima(train_price, xreg = train_XREG)
#forecasting dynamic linear reg
dlr.fc = forecast::forecast(dlr.fit, xreg = test_XREG )
fit2 = auto.arima(train_price, xreg = fourier(train_price, K=2), seasonal=FALSE)
fc2  = forecast(fit2, xreg = fourier(train_price, K=2, h = flength))
fit_nn = nnetar(train_price)
price.nn= forecast(fit_nn, h = flength)


autoplot(window(price, start = 2018)) +
  ggtitle("Data Quarterly-2Y")+
  autolayer(dlr.fc$mean, series="DLR")+
  autolayer(ts.drift$mean, series="Drift") +
  autolayer(ts.ses$mean, series="SES, SMA, Naive")+
  autolayer(ts.ets$mean, series="ETS")+
  autolayer(ts.arma$mean, series="ARMA")+
  autolayer(ts.arima$mean, series="ARIMA, SARIMA") +
  autolayer(ts.sarima$mean, series="SARIMA")+
  autolayer(ts.ave$mean, series="Average")+
  autolayer(fc2$mean, series="DHR")+
  autolayer(price.nn$mean, series = "NN")
```



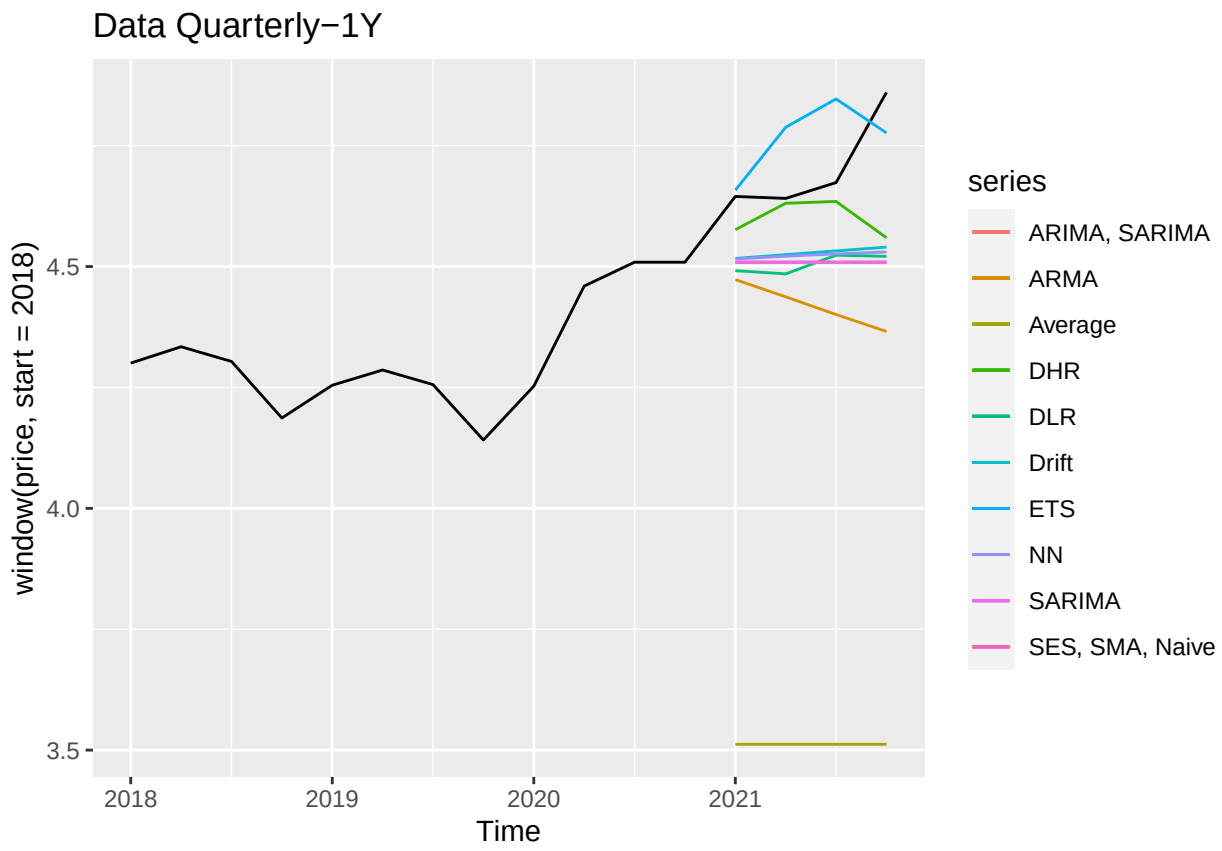

```
mape.naiveq = accuracy(ts.naive$mean, test_price)[5]
mape.aveq = accuracy(ts.ave$mean, test_price)[5]
mape.driftq = accuracy(ts.drift$mean, test_price)[5]
mape.smaq = accuracy(ts.sma$forecast, test_price)[5]
mape.sesq = accuracy(ts.ses$mean, test_price)[5]
mape.etsq = accuracy(ts.ets$mean, test_price)[5]
mape.armaq = accuracy(ts.arma$mean, test_price)[5]
mape.arimaq = accuracy(ts.arima$mean, test_price)[5]
mape.sarimaq = accuracy(ts.sarima$mean, test_price)[5]
mape.dlrq = accuracy(dlr.fc$mean, test_price)[5]
mape.dhrq = accuracy(fc2$mean, test_price)[5]
mape.nnq = accuracy(price.nn$mean, test_price)[5]

pfit1q = auto.arima(train_price, xreg = fourier(train_price, K=1), seasonal=FALSE)
pfc1q  = forecast(pfit1q, xreg = fourier(train_price, K=1, h = flength))
pfit2q = auto.arima(train_price, xreg = fourier(train_price, K=2), seasonal=FALSE)
pfc2q  = forecast(pfit2q, xreg = fourier(train_price, K=2, h = flength))

# aicc DHR - 2 years
pfit1q$aicc
```

```
## [1] -127.0523
```

```
pfit2q$aicc
```

```
## [1] -128.99
```

```
## Training the model without the last 1 years:

train_price <- window(price, end = 2021-1/4)
test_price <- window(price, start = 2021)
flength<-length(test_price)
XREG <- cbind(ppi_coffee,ppi_tea,bean_price,temperature,er_real)
train_XREG = window(XREG, end = 2021-1/4)
test_XREG  = window(XREG, start = 2021)

#training & forecasting naive
ts.naive = naive(train_price, h=flength)
#training & forecasting average
ts.ave   = meanf(train_price, h=flength)
#training & forecasting drift
ts.drift = rwf(train_price, drift=TRUE, h=flength)
#training & forecasting simple moving avg
ts.sma   = sma(train_price, h=flength)
#training & forecasting simple exponential smoothing
ts.ses   = ses(train_price, h=flength)
#training & forecasting ets: error+trend+seasonality
fit.ets = forecast::ets(train_price)
ts.ets = forecast::forecast.ets(fit.ets, h=flength)
#training & forecasting arma
fit.arma = auto.arima(train_price, d=0, seasonal=FALSE)
ts.arma  = forecast::forecast(fit.arma, h=flength)
#training & forecasting arima
fit.arima = auto.arima(train_price, seasonal=FALSE)
ts.arima  = forecast::forecast(fit.arima, h=flength)
#training & forecasting sarima
fit.sarima = auto.arima(train_price)
ts.sarima = forecast::forecast(fit.sarima, h=flength)
#training dynamic linear reg
dlr.fit = auto.arima(train_price, xreg = train_XREG)
#forecasting dynamic linear reg
dlr.fc = forecast::forecast(dlr.fit, xreg = test_XREG )
fit21 = auto.arima(train_price, xreg = fourier(train_price, K=2), seasonal=FALSE)
fc2  = forecast(fit21, xreg = fourier(train_price, K=2, h = flength))
fit_nn = nnetar(train_price)
price.nn= forecast(fit_nn, h = flength)


autoplot(window(price, start = 2018)) +
  ggtitle("Data Quarterly-1Y")+
  autolayer(dlr.fc$mean, series="DLR")+
  autolayer(ts.drift$mean, series="Drift") +
  autolayer(ts.ses$mean, series="SES, SMA, Naive")+
  autolayer(ts.ets$mean, series="ETS")+
  autolayer(ts.arma$mean, series="ARMA")+
  autolayer(ts.arima$mean, series="ARIMA, SARIMA") +
  autolayer(ts.sarima$mean, series="SARIMA")+
  autolayer(ts.ave$mean, series="Average")+
  autolayer(fc2$mean, series="DHR")+
  autolayer(price.nn$mean, series = "NN")
```




```
mape.naive1q = accuracy(ts.naive$mean, test_price)[5]
mape.ave1q = accuracy(ts.ave$mean, test_price)[5]
mape.drift1q = accuracy(ts.drift$mean, test_price)[5]
mape.sma1q = accuracy(ts.sma$forecast, test_price)[5]
mape.ses1q = accuracy(ts.ses$mean, test_price)[5]
mape.ets1q = accuracy(ts.ets$mean, test_price)[5]
mape.arma1q = accuracy(ts.arma$mean, test_price)[5]
mape.arima1q = accuracy(ts.arima$mean, test_price)[5]
mape.sarima1q = accuracy(ts.sarima$mean, test_price)[5]
mape.dlr1q = accuracy(dlr.fc$mean, test_price)[5]
mape.dhr1q = accuracy(fc2$mean, test_price)[5]
mape.nn1q = accuracy(price.nn$mean, test_price)[5]

pfit11q = auto.arima(train_price, xreg = fourier(train_price, K=1), seasonal=FALSE)
pfc11q  = forecast(pfit11q, xreg = fourier(train_price, K=1, h = flength))
pfit21q = auto.arima(train_price, xreg = fourier(train_price, K=2), seasonal=FALSE)
pfc21q  = forecast(pfit21q, xreg = fourier(train_price, K=2, h = flength))

# aicc DHR - 1 year
pfit11q$aicc
```

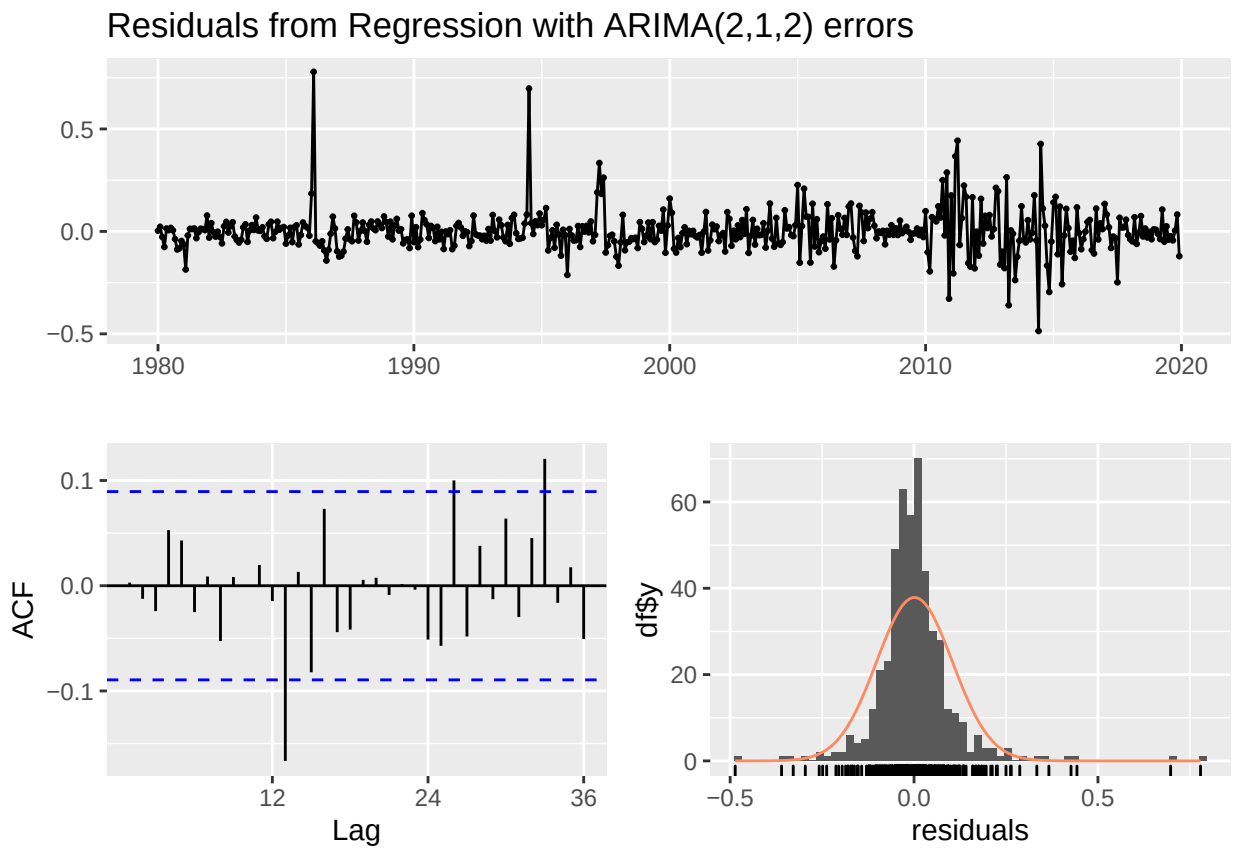
```
## [1] -133.4159
```

```
pfit21q$aicc
```

```
## [1] -135.2994
```

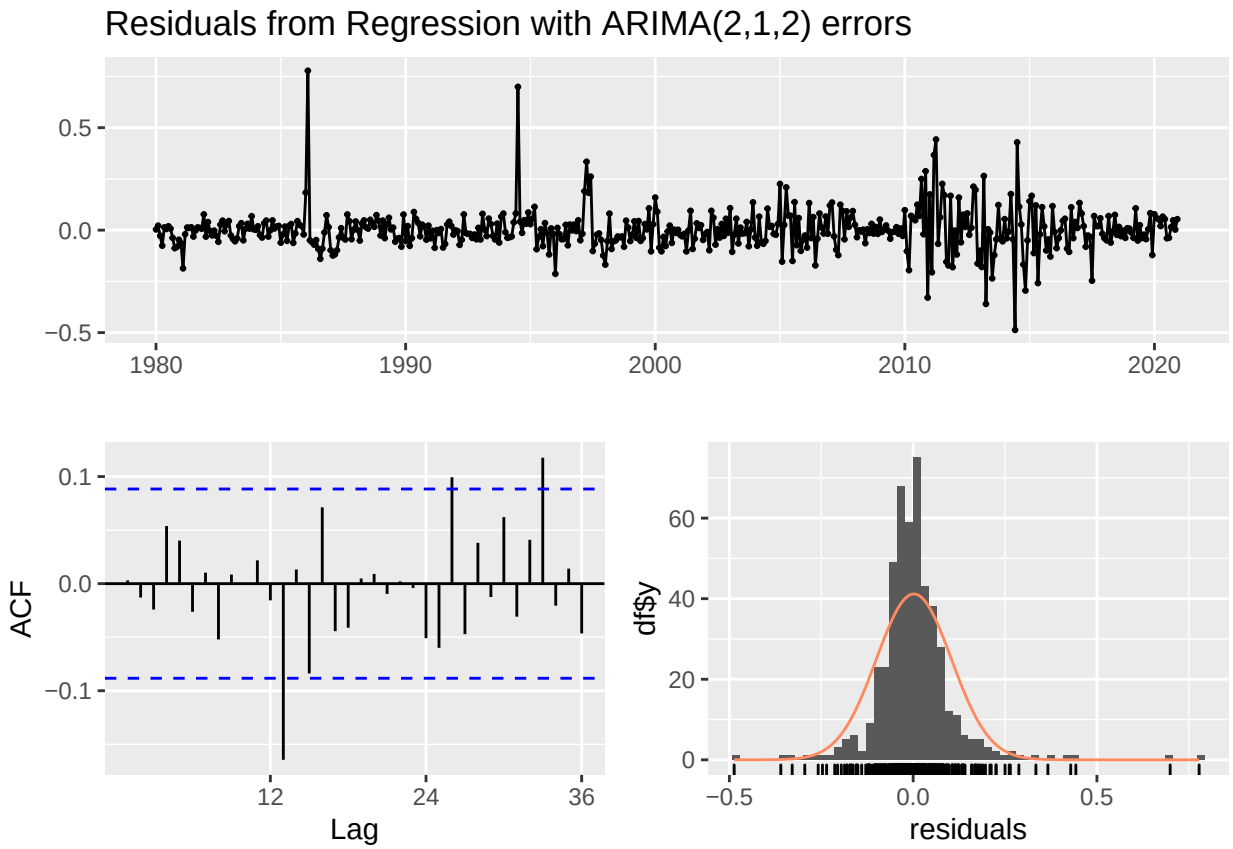
checking residuals of the best models:

```
#Monthly data
#2 years forecasting
checkresiduals(fit4)
```



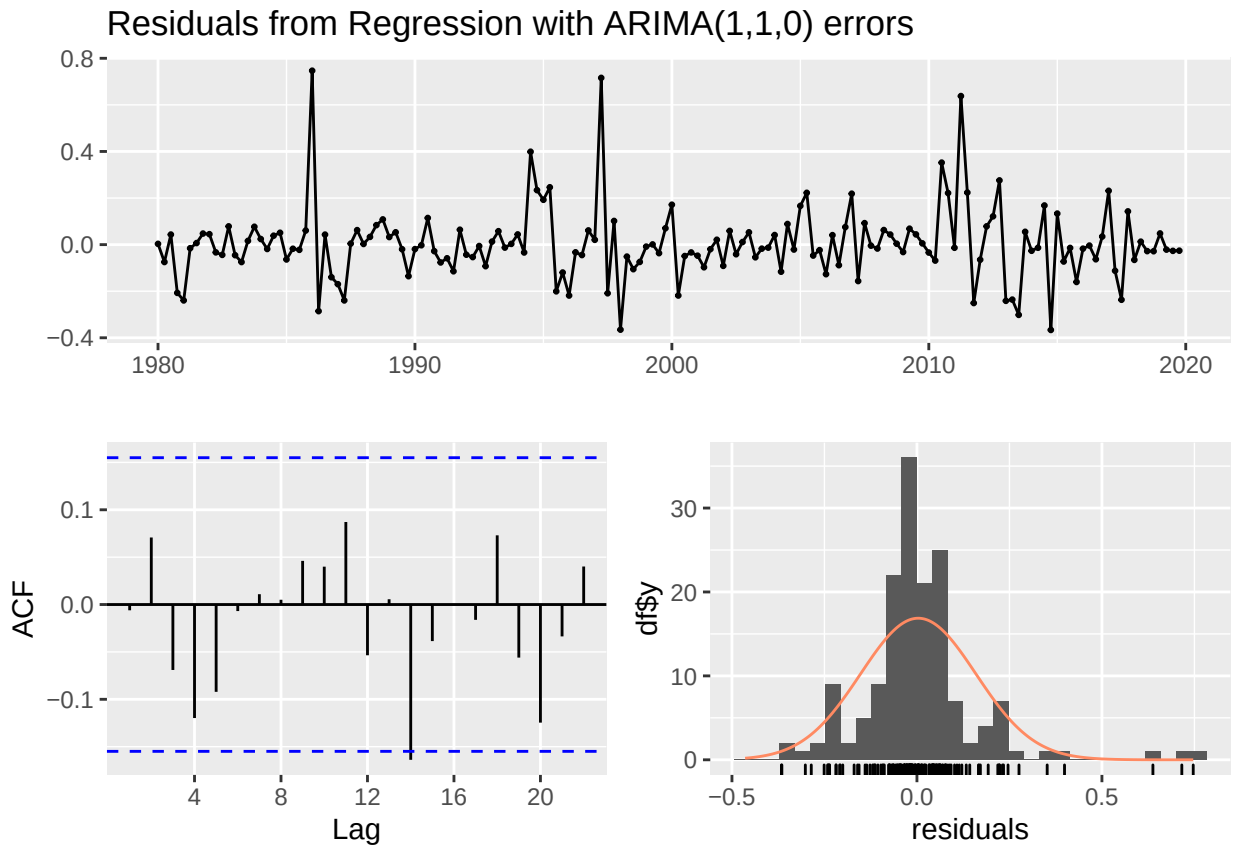

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,1,2) errors
## Q* = 27.749, df = 12, p-value = 0.006019
##
## Model df: 12.   Total lags used: 24
```

```
#1 year forecasting
checkresiduals(fit41)
```



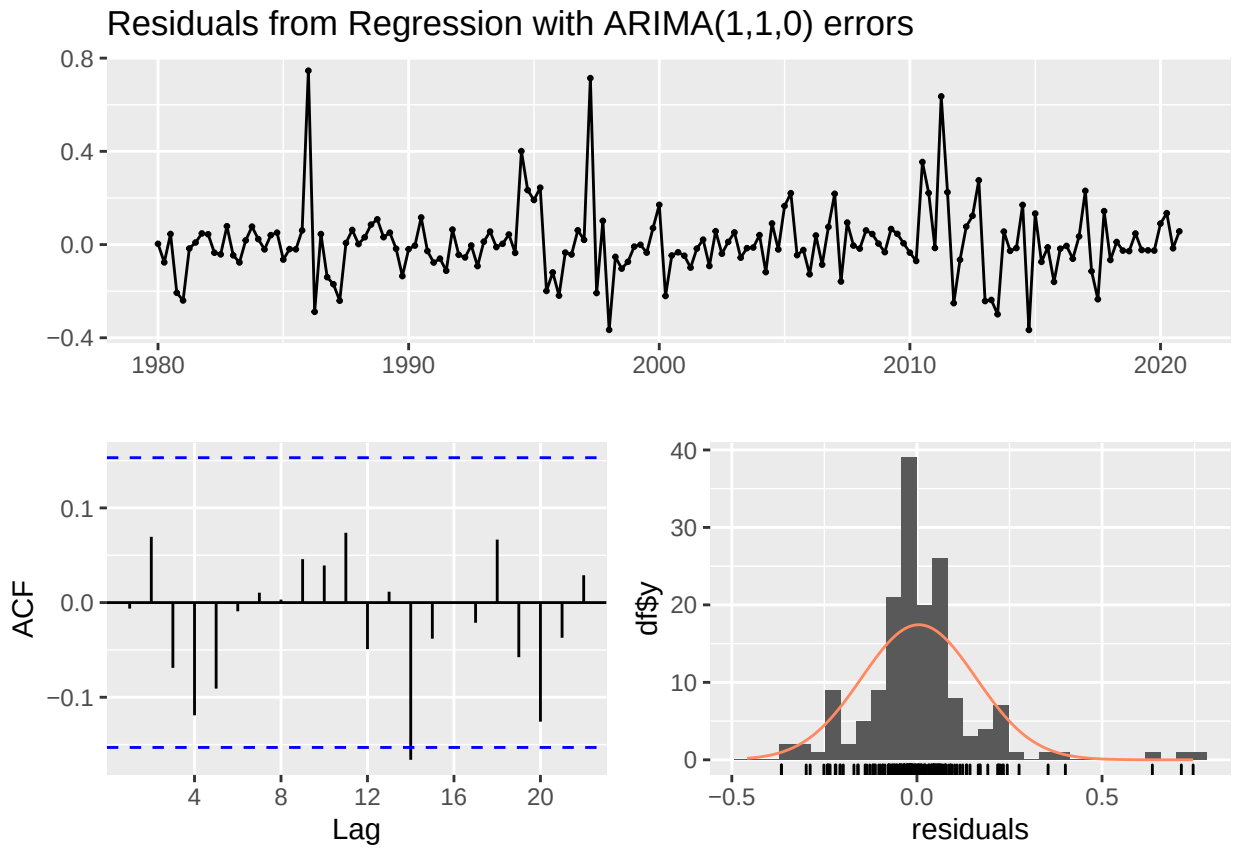

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,1,2) errors
## Q* = 28.111, df = 12, p-value = 0.00533
##
## Model df: 12.   Total lags used: 24
```

```
#Quarterly data
#2 years forecasting
checkresiduals(fit2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,1,0) errors
## Q* = 5.4502, df = 4, p-value = 0.2441
##
## Model df: 4.   Total lags used: 8
```

```
#1 year forecasting
checkresiduals(fit21)
```



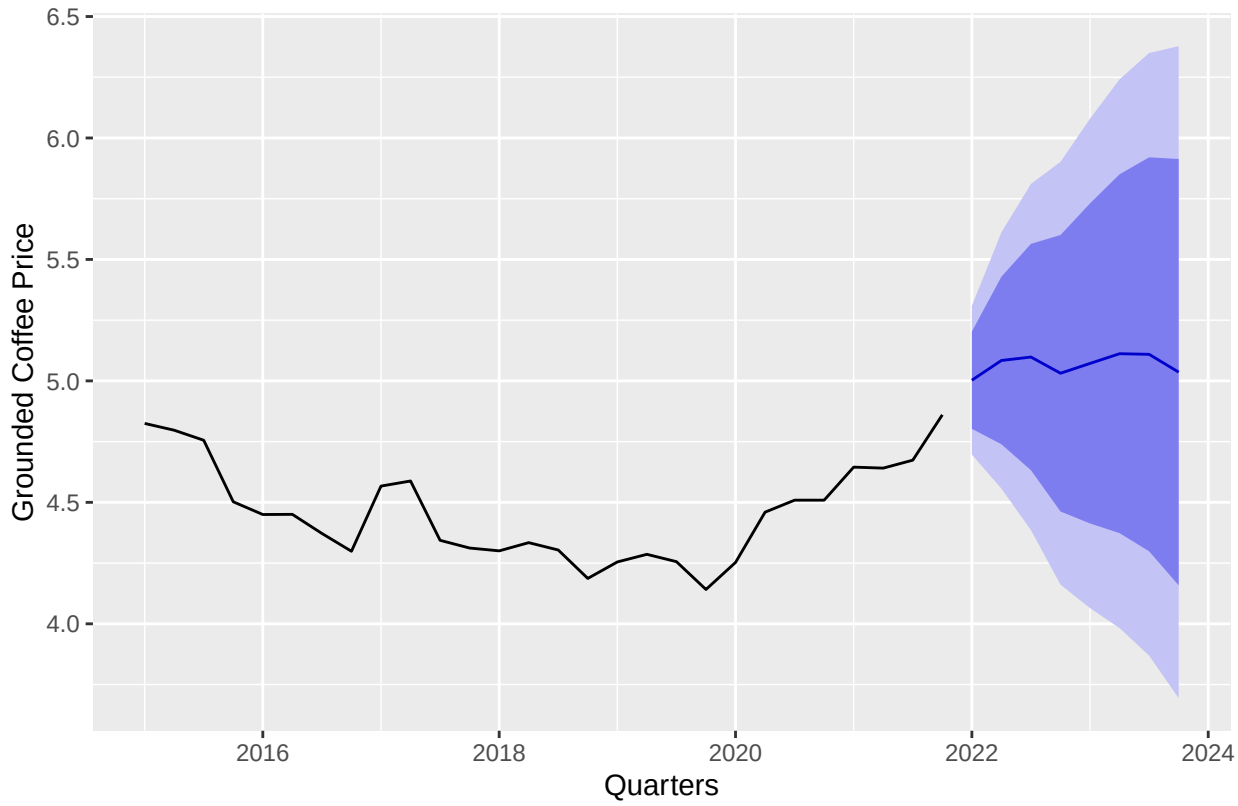

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,1,0) errors
## Q* = 5.4841, df = 4, p-value = 0.2411
##
## Model df: 4.   Total lags used: 8
```

Comparing all the MAPEs

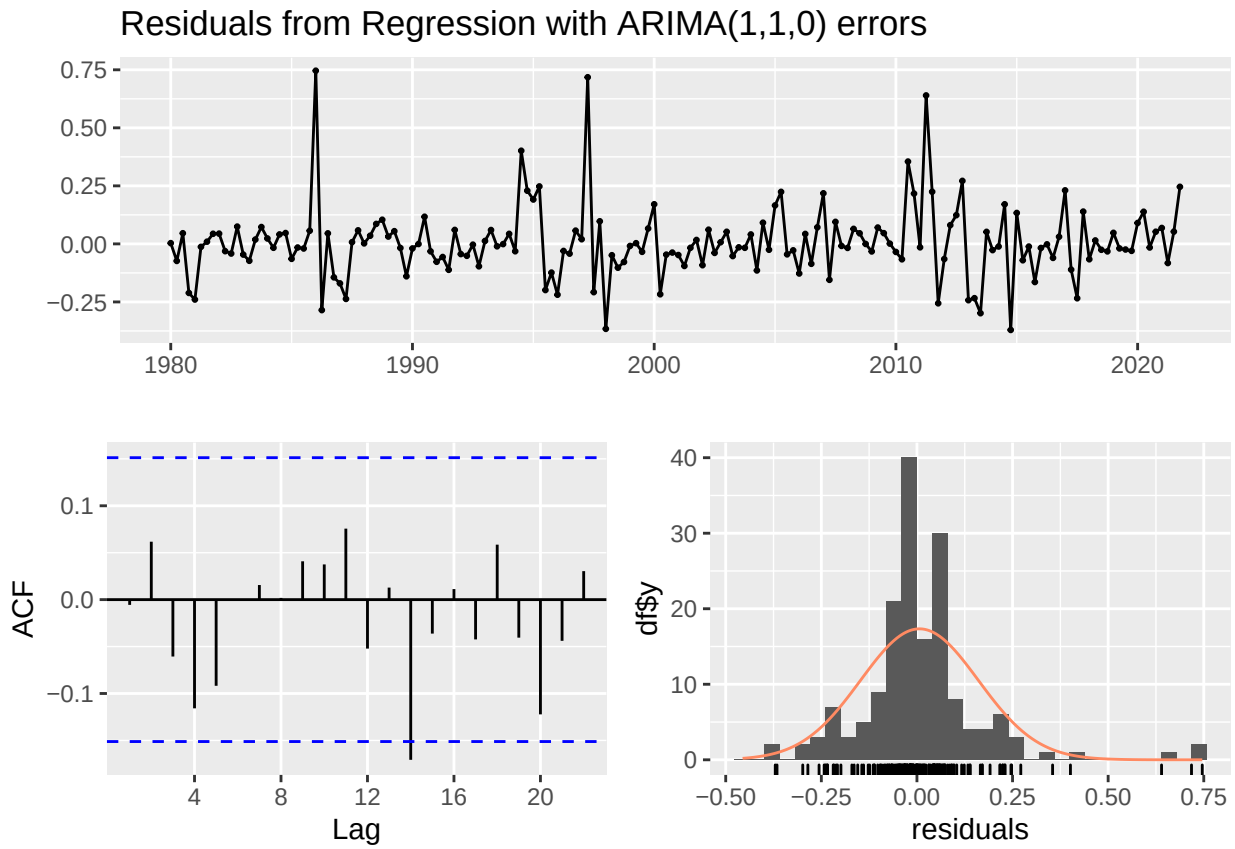
Method	MAPE Monthly-2Y	MAPE Monthly-1Y	MAPE Quarterly-2Y	MAPE Quarterly-1Y
Naive	11.1540569	3.8759041	9.2321228	4.1309914
Average	23.0251048	24.846068	23.5301041	25.3290147
Drift	10.6801872	3.5098083	8.6818107	3.7194545
SMA	11.1540569	3.8759041	9.2321228	4.1309914
SES	11.1537395	3.8759787	9.2318716	4.1309914
ETS	13.5426748	3.4945095	9.1399847	2.2189126
ARMA	14.5037279	5.2776369	12.5566802	6.0251324
ARIMA	11.7092164	3.8376177	10.5596762	4.1309914
SARIMA	11.9952936	3.2533216	10.5596762	4.1309914
DLR	15.1209795	4.8696817	11.1544422	4.214552
DHR	10.1433776	2.394642	8.7066246	2.1793244
NN	13.3258486	3.6343711	10.8092715	3.826472

Final Model

```
# Quarterly data
price<-tsclean(aggregate(tsdata[,"price"], nfrequency = 4, mean))

fit2 = auto.arima(price, xreg = fourier(price, K=2), seasonal=FALSE)
fc2  = forecast(fit2, xreg = fourier(price, K=2, h = 8))


autoplot(window(price, start = 2015)) +
  xlab("Quarters") +
  ylab("Grounded Coffee Price")+
  autolayer(fc2)
```




```
checkresiduals(fit2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,1,0) errors
## Q* = 5.1536, df = 4, p-value = 0.2719
##
## Model df: 4.   Total lags used: 8
```

```
fit2
```

```
## Series: price
## Regression with ARIMA(1,1,0) errors
##
## Coefficients:
##          ar1     S1-4     C1-4     C2-4
##       0.4032  -0.0180  -0.0382  -0.0087
## s.e.  0.0711   0.0111   0.0111   0.0043
##
## sigma^2 = 0.02439:  log likelihood = 75.04
## AIC=-140.08   AICc=-139.71   BIC=-124.49
```

```
fc2
```

```
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2022 Q1       5.002918 4.802755 5.203081 4.696795 5.309041
## 2022 Q2       5.084070 4.739172 5.428968 4.556594 5.611546
## 2022 Q3       5.098195 4.632164 5.564226 4.385462 5.810928
## 2022 Q4       5.031512 4.462467 5.600557 4.161232 5.901791
```

```
## 2023 Q1       5.071941 4.413212 5.730669 4.064502 6.079380
## 2023 Q2       5.111901 4.373352 5.850451 3.982387 6.241416
## 2023 Q3       5.109417 4.298515 5.920320 3.869249 6.349585
## 2023 Q4       5.036037 4.158596 5.913477 3.694107 6.377966
```

Outliers

```
price2<-na.interp(aggregate(tsdata[,"price"], nfrequency = 4, mean))
#outliers
fit2_out = auto.arima(price2, xreg = fourier(price2, K=2), seasonal=FALSE)
fc2_out  = forecast(fit2_out, xreg = fourier(price2, K=2, h = 8))

autoplot(window(price2, start = 2015)) +
  xlab("Quarters") +
  ylab("Grounded Coffee Price")+
  autolayer(fc2_out$mean, series="With Outliers")+
  autolayer(fc2$mean, series="Without Outliers")+
  ylim(3.5,6.5)
```