R_forecast

Jerry Feng

2024-06-03

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```r
library(tseries)
```

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

```r
library(forecast)
library(vars)
```

```
## Loading required package: MASS
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

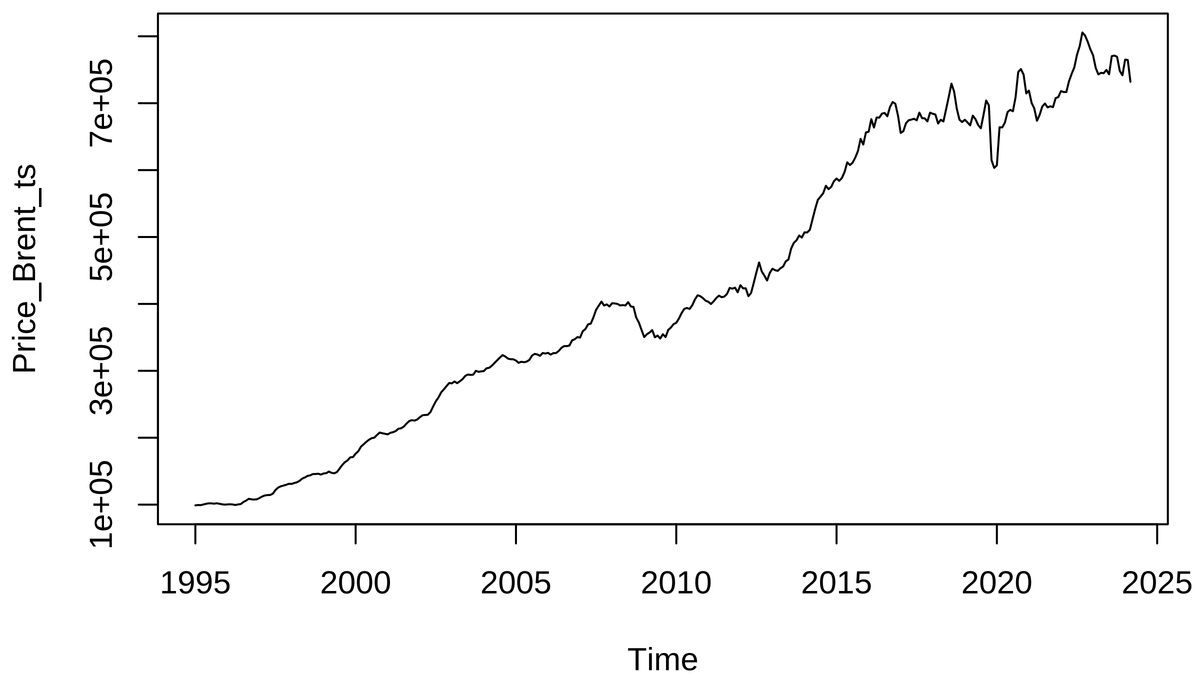
```
## Loading required package: urca
```

```
## Loading required package: lmtest
```

```r
library(readr)
library(Metrics)
```

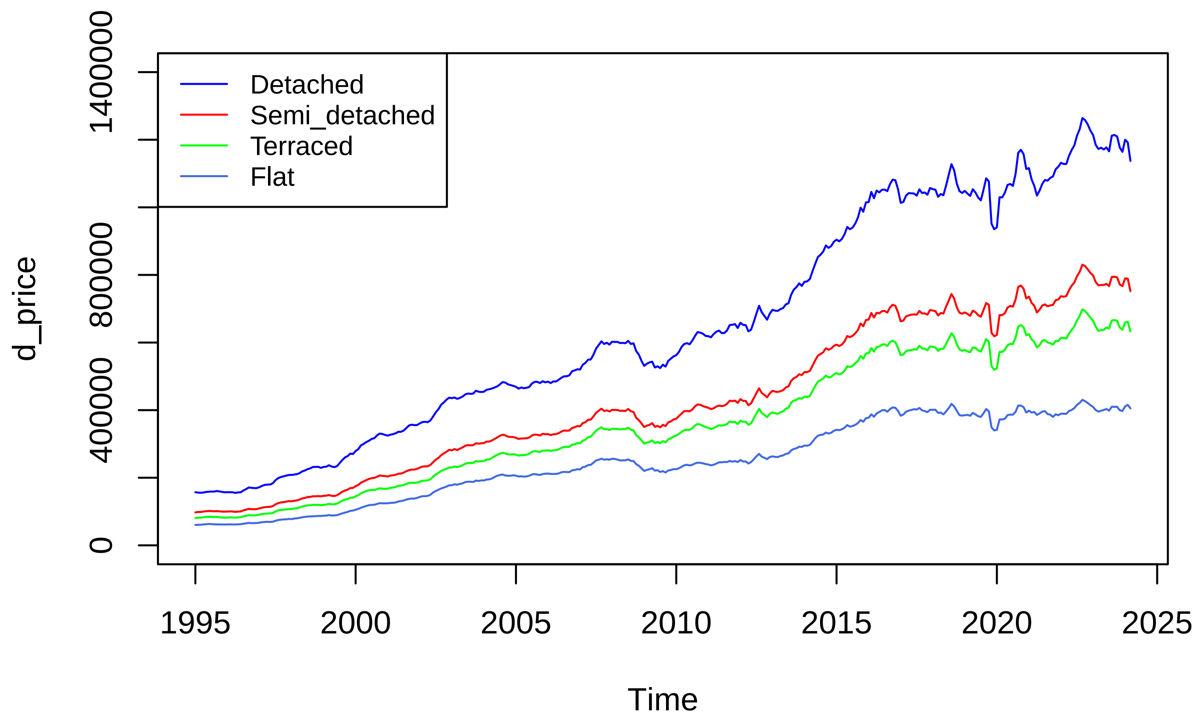
```
##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy
```

```r
Brent_df <- read.csv("Updated_Brent.csv")
Brent_df$Date <- as.Date(Brent_df$Date, format = "%Y-%m-%d")
Price_Brent_ts <- ts(Brent_df$Average_Price, start = c(1995, 1), frequency = 12)
d_price <- ts(Brent_df$Detached_Average_Price, start = c(1995 , 1), frequency = 12)
sd_price <- ts(Brent_df$Semi_Detached_Average_Price, start = c(1995, 1), frequency = 12)
t_price <- ts(Brent_df$Terraced_Average_Price, start = c(1995, 1), frequency = 12)
f_price <- ts(Brent_df$Flat_Average_Price, start = c(1995, 1), frequency = 12)
plot(Price_Brent_ts)
```

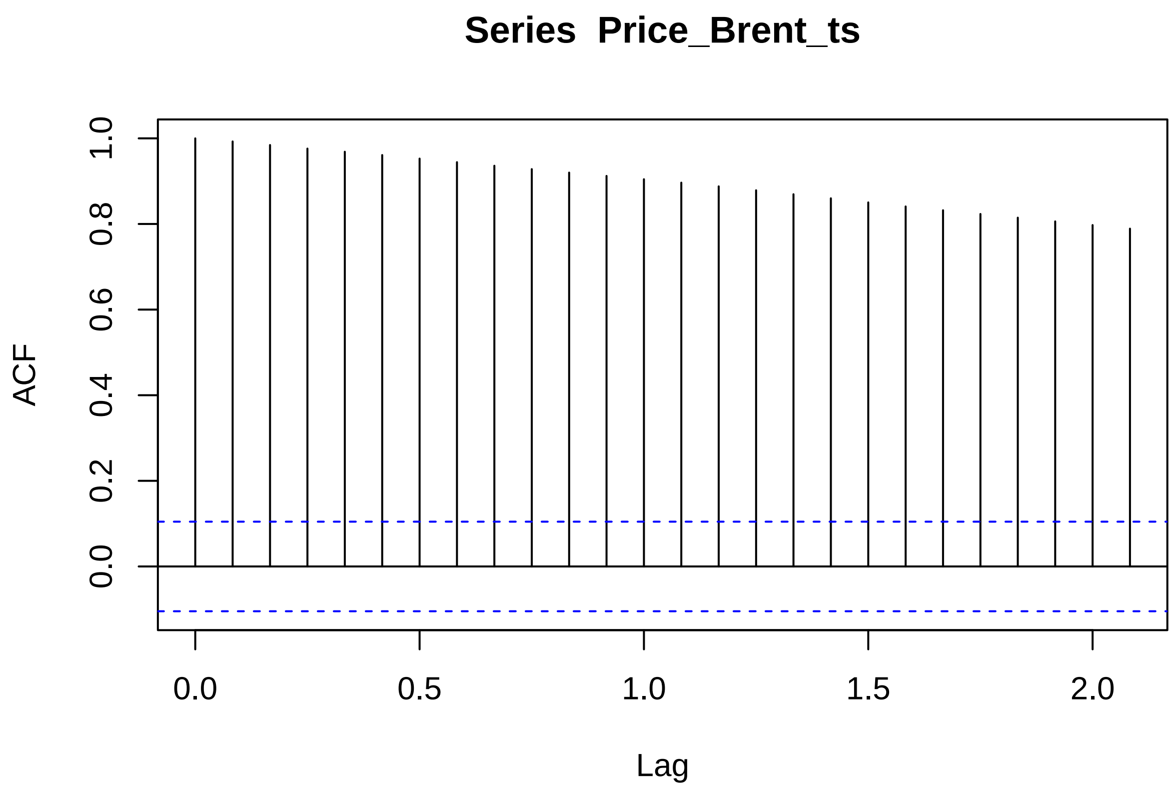



```r
plot(d_price, col = "blue", ylim = c(0, 1400000))
lines(sd_price, col = "red")
lines(t_price, col = "green")
lines(f_price, col = "royalblue")
legend("topleft",
```

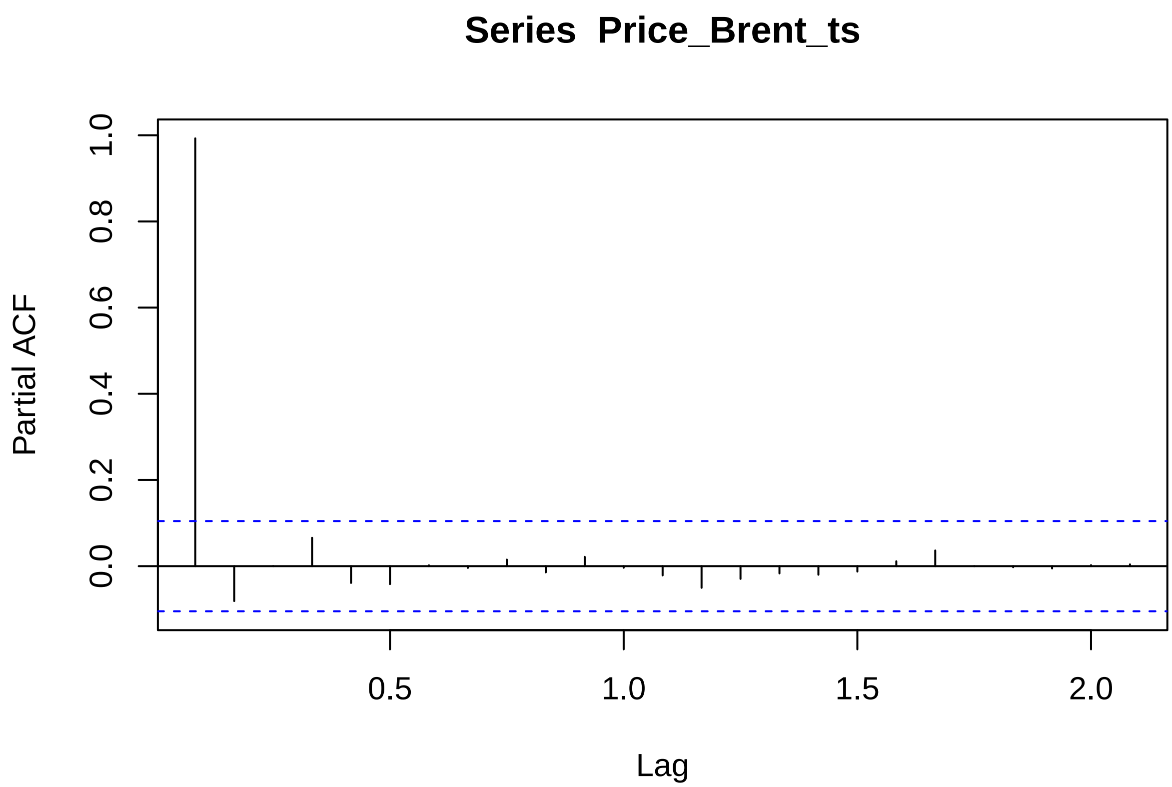
```
       legend = c("Detached", "Semi_detached", "Terraced", "Flat"),
       col = c("blue", "red", "green", "royalblue"),
       lty = 1,
       cex = 0.8)
```




```
legacy_ts <- ts(Brent_df$Average_Price, start = c(1995, 1), end = c(2023, 6), frequency = 12)
Price_ts_2023_2024 <- window(Price_Brent_ts, start = c(2023, 6), end = c(2024, 3))
diff_legacy_ts <- diff(legacy_ts, differences=1)
acf(Price_Brent_ts)
```




```
pacf(Price_Brent_ts)
```




```r
adf.test(Price_Brent_ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Price_Brent_ts
## Dickey-Fuller = -2.3824, Lag order = 7, p-value = 0.4154
## alternative hypothesis: stationary
```

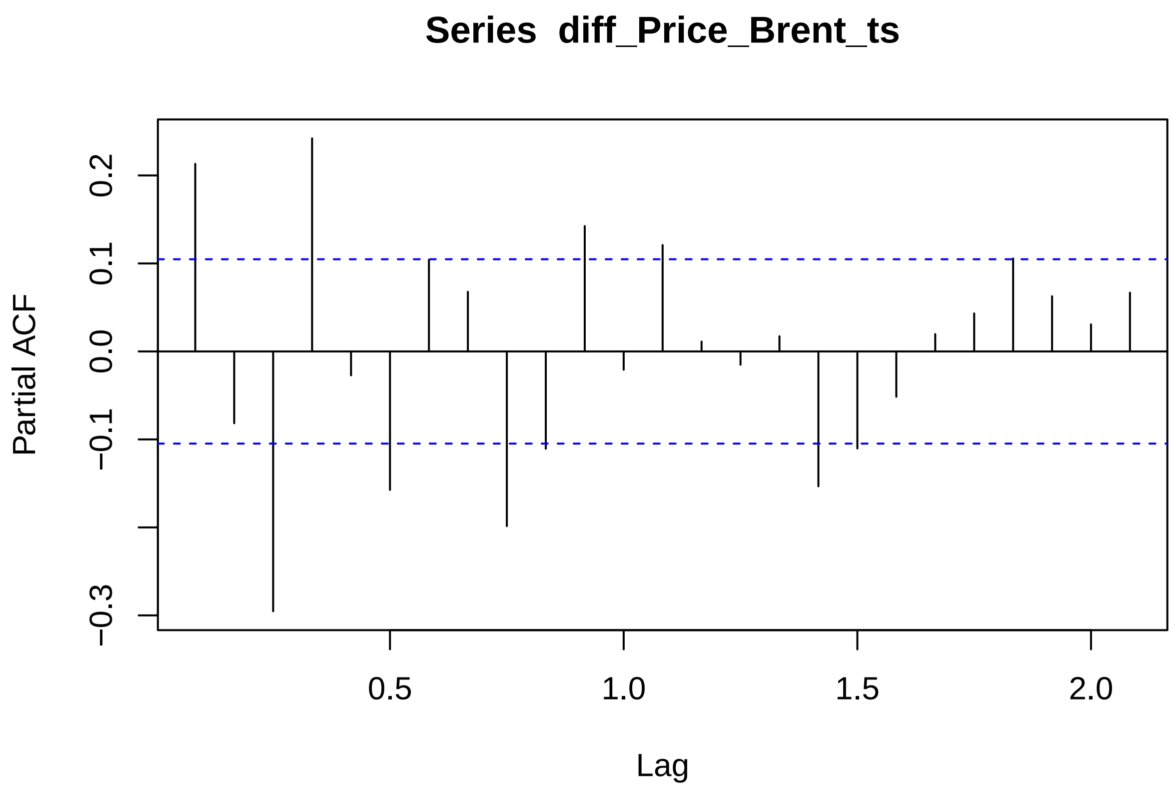
```r
diff_Price_Brent_ts <- diff(Price_Brent_ts, differences=1)
adf.test(diff_Price_Brent_ts)
```

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

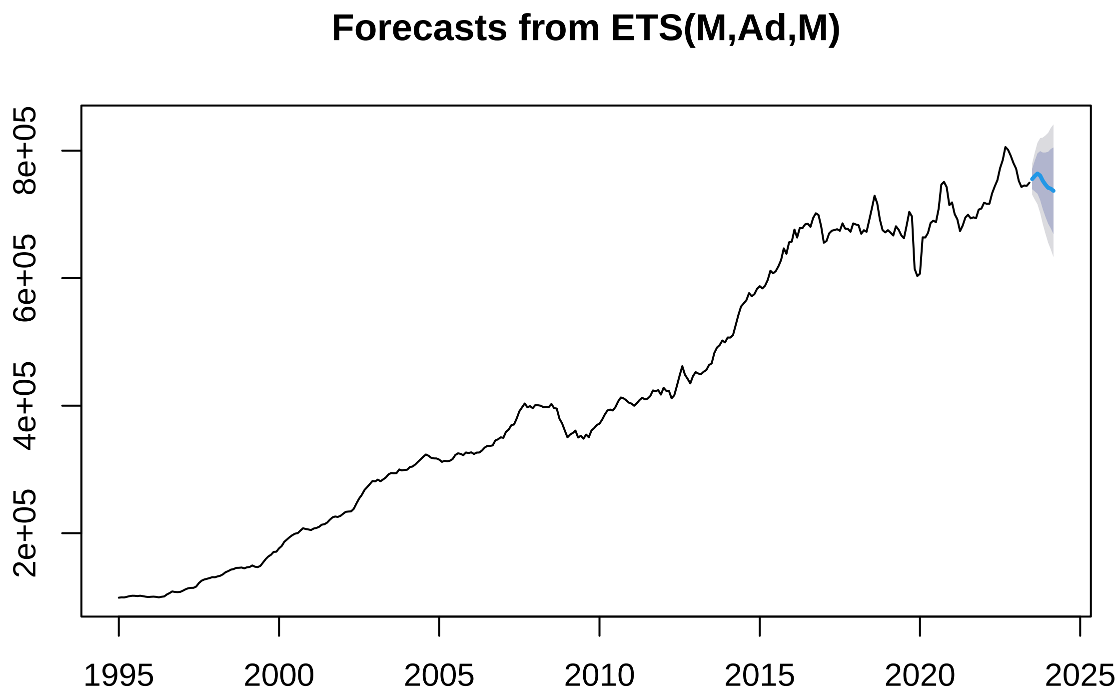
```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_Price_Brent_ts
## Dickey-Fuller = -5.7933, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
acf(diff_Price_Brent_ts)
```

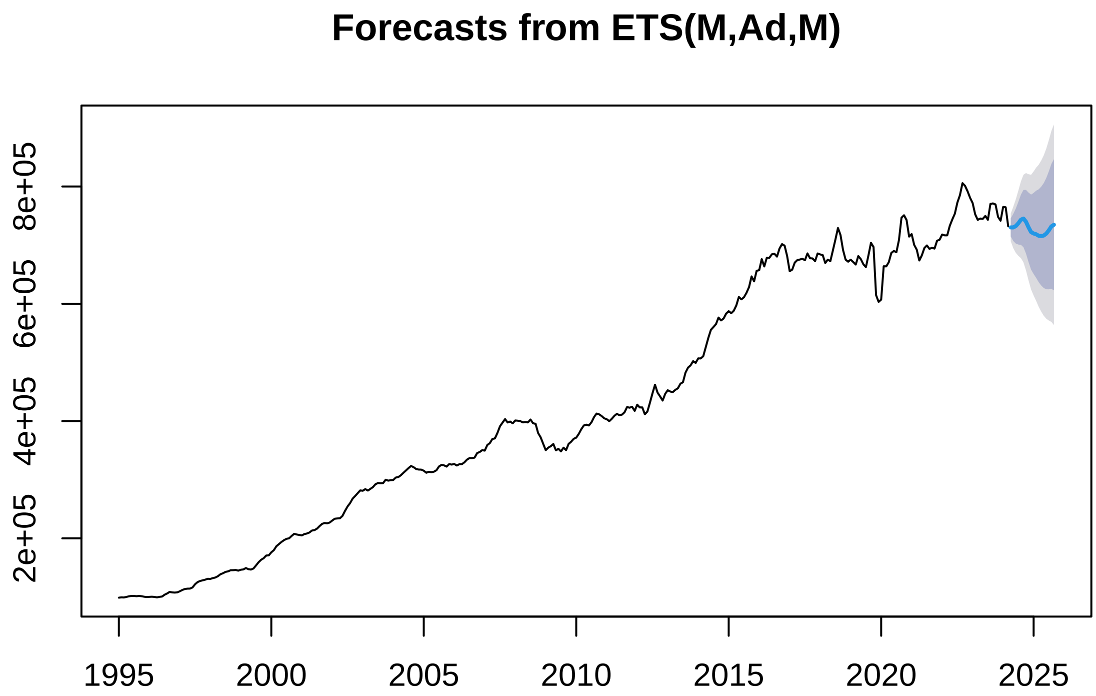

```
pacf(diff_Price_Brent_ts)
```




```r
ets_Price_Brent_model <- ets(Price_Brent_ts, model = "ZZZ")
ets_Price_Brent_model_forecast <- forecast(ets_Price_Brent_model, h = 18)
ets_legacy <- ets(legacy_ts, model = "MAM")
ets_legacy_forecast <- forecast(ets_legacy, h=9)
ets_forecasted_values_legacy <- ets_legacy_forecast$mean
ets_forecasted_ts_legacy <- ts(ets_forecasted_values_legacy, start = c(2023, 6), frequency = 12)
plot(ets_legacy_forecast)
```




```r
plot(ets_Price_Brent_model_forecast)
```

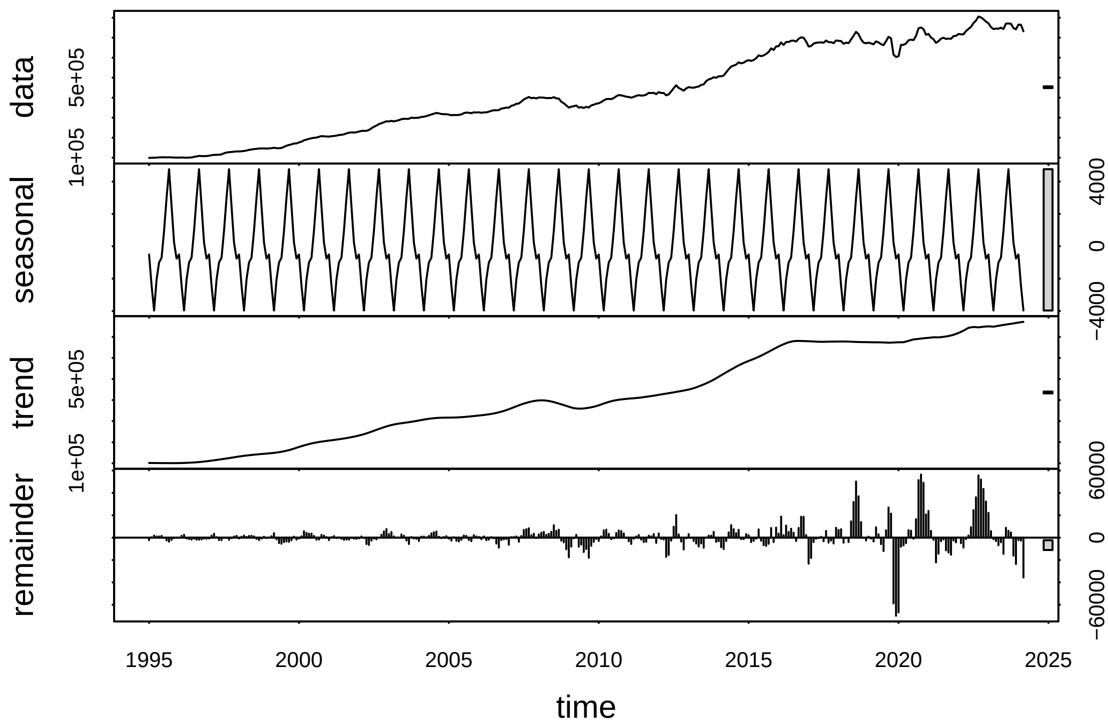



```r
summary(ets_Price_Brent_model_forecast)
```

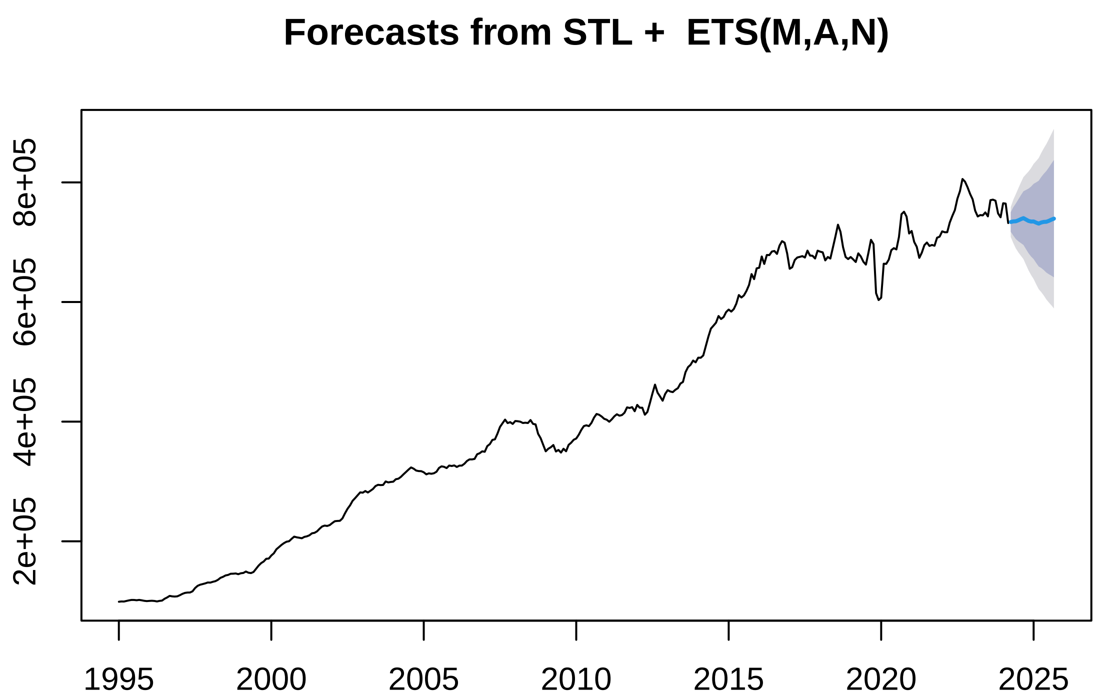
```
##
## Forecast method: ETS(M,Ad,M)
##
## Model Information:
## ETS(M,Ad,M)
##
## Call:
##  ets(y = Price_Brent_ts, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.9998
##     beta  = 0.1056
##     gamma = 2e-04
##     phi   = 0.9284
##
##   Initial states:
##     l = 98690.9278
##     b = 1236.2166
##     s = 0.9933 1.0031 1.0145 1.0202 1.0156 1.0055
##            0.9968 0.9912 0.9889 0.9886 0.9909 0.9915
##
##   sigma:  0.0166
##
```

```
##      AIC     AICc      BIC 
## 8174.991 8177.051 8244.485 
## 
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
## Training set 722.7838 9067.059 5355.794 0.2295089 1.148674 0.170222 0.09078767
## 
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Apr 2024       730253.7 714720.6 745786.8 706497.9 754009.5
## May 2024       730170.4 706986.3 753354.5 694713.4 765627.4
## Jun 2024       732599.7 702663.1 762536.2 686815.7 778383.7
## Jul 2024       737398.6 700985.6 773811.7 681709.7 793087.6
## Aug 2024       743295.8 700507.8 786083.8 677857.2 808734.4
## Sep 2024       745298.9 696457.1 794140.7 670601.8 819996.0
## Oct 2024       739867.9 685605.2 794130.7 656880.2 822855.7
## Nov 2024       730399.0 671222.7 789575.3 639896.6 820901.3
## Dec 2024       722194.1 658219.8 786168.4 624353.9 820034.4
## Jan 2025       719890.1 650750.3 789030.0 614149.8 825630.5
## Feb 2025       718506.2 644212.1 792800.3 604883.2 832129.2
## Mar 2025       715975.7 636745.0 795206.5 594802.7 837148.7
## Apr 2025       715421.7 631126.4 799717.0 586503.1 844340.2
## May 2025       716368.5 626899.1 805838.0 579536.8 853200.2
## Jun 2025       719714.3 624810.0 814618.7 574570.7 864858.0
## Jul 2025       725332.3 624700.6 825964.0 571429.4 879235.2
## Aug 2025       731981.8 625465.3 838498.3 569078.9 894884.7
## Sep 2025       734747.7 622917.8 846577.6 563718.6 905776.8
```

```r
stl_Price_Brent_model <- stl(Price_Brent_ts, s.window="periodic", robust=TRUE)
plot(stl_Price_Brent_model)
```




```r
stl_Price_Brent_model_forecast <- forecast(stl_Price_Brent_model, method="ets", h=18)
plot(stl_Price_Brent_model_forecast)
```

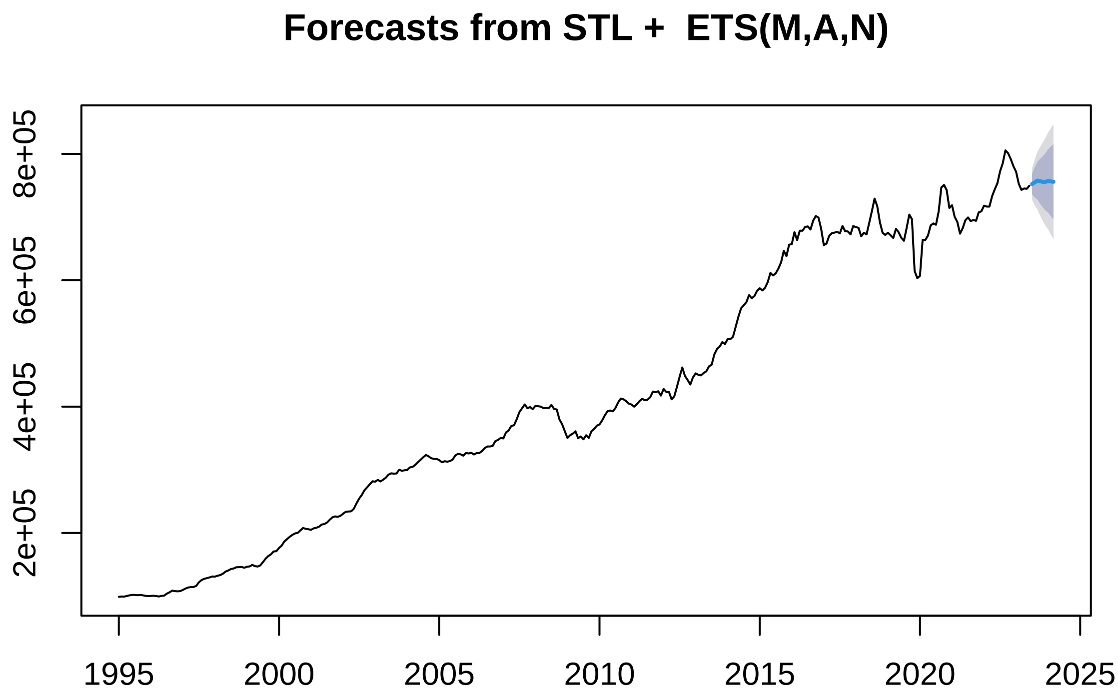



```r
summary(stl_Price_Brent_model_forecast)
```

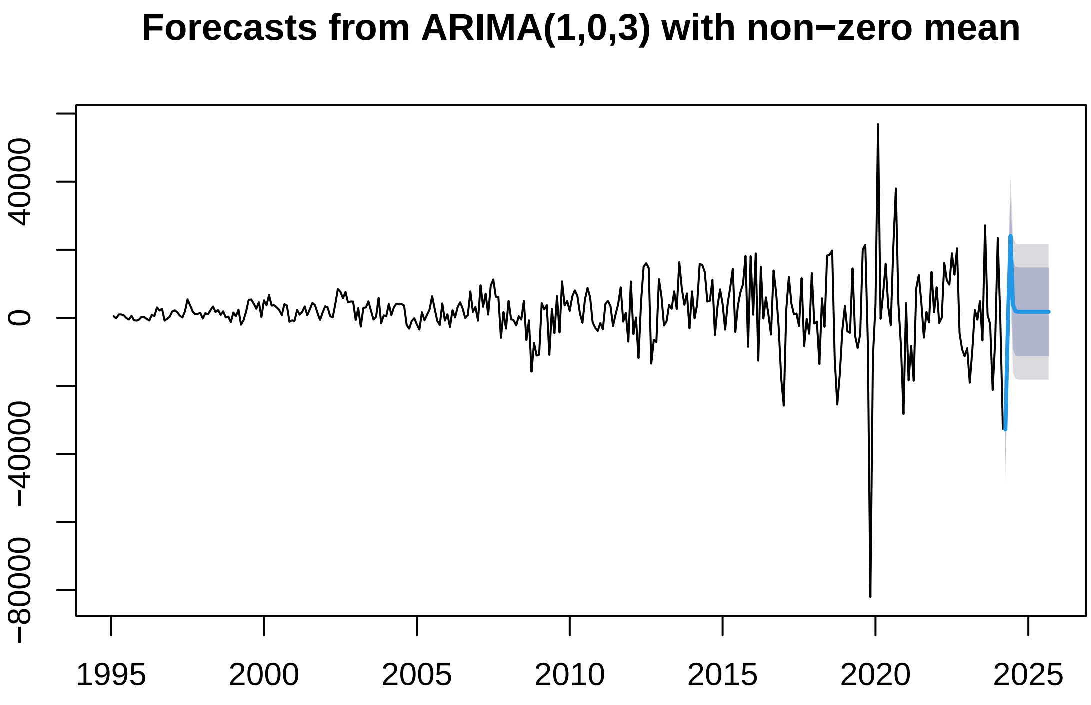
```
##
## Forecast method: STL +  ETS(M,A,N)
##
## Model Information:
## ETS(M,A,N)
##
## Call:
##  ets(y = na.interp(x), model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.046
##
##   Initial states:
##     l = 102885.2623
##     b = 98.2728
##
##   sigma:  0.0174
##
##      AIC     AICc      BIC
## 8195.481 8195.655 8214.785
##
## Error measures:
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -10.54962 9420.835 5540.837 0.06758215 1.249905 0.1761031
##                   ACF1
## Training set 0.1655694
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Apr 2024       733790.9 717412.4 750169.3 708742.2 758839.6
## May 2024       734775.8 711075.2 758476.4 698528.9 771022.8
## Jun 2024       734996.8 705305.2 764688.5 689587.3 780406.3
## Jul 2024       736585.7 701528.2 771643.2 682969.9 790201.6
## Aug 2024       738559.3 698494.7 778623.8 677285.8 799832.7
## Sep 2024       740254.6 695408.6 785100.6 671668.6 808840.7
## Oct 2024       737898.8 688419.9 787377.7 662227.3 813570.2
## Nov 2024       735572.9 681560.7 789585.1 652968.3 818177.4
## Dec 2024       734511.1 676032.2 792990.1 645075.3 823947.0
## Jan 2025       734690.4 671787.9 797592.9 638489.3 830891.4
## Feb 2025       732702.2 665402.3 800002.1 629775.9 835628.6
## Mar 2025       731085.9 659401.8 802770.1 621454.4 840717.5
## Apr 2025       732927.3 656862.2 808992.5 616595.7 849258.9
## May 2025       733912.3 653461.7 814362.9 610873.7 856950.9
## Jun 2025       734133.3 649286.6 818980.0 604371.4 863895.1
## Jul 2025       735722.2 646463.7 824980.6 599213.2 872231.2
## Aug 2025       737695.7 644005.9 831385.5 594409.5 880981.9
## Sep 2025       739391.1 641246.9 837535.2 589292.6 889489.6
```

```r
stl_legacy <- stl(legacy_ts, s.window="periodic", robust=TRUE)
stl_legacy_forecast <- forecast(stl_legacy, h=9)
stl_forecasted_values_legacy <- stl_legacy_forecast$mean
stl_forecasted_ts_legacy <- ts(stl_forecasted_values_legacy, start = c(2023, 6), frequency = 12)
plot(stl_legacy_forecast)
```




```
avg_Price_Brent_model <- auto.arima(diff_Price_Brent_ts, seasonal=FALSE)
avg_Price_Brent_model_forecast <- forecast(avg_Price_Brent_model, h=18)
plot(avg_Price_Brent_model_forecast)
```




```r
summary(avg_Price_Brent_model_forecast)
```

```
##
## Forecast method: ARIMA(1,0,3) with non-zero mean
##
## Model Information:
## Series: diff_Price_Brent_ts
## ARIMA(1,0,3) with non-zero mean
##
## Coefficients:
##          ar1     ma1     ma2      ma3       mean
##       0.0854  0.3337  0.3228  -0.6367  1777.5154
## s.e.  0.0926  0.0712  0.0609   0.0611   466.0876
##
## sigma^2 = 61668538:  log likelihood = -3635.74
## AIC=7283.48   AICc=7283.73   BIC=7306.63
##
## Error measures:
##                     ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -3.603367 7796.638 4719.731 318.7985 434.8402 0.5936389
##                     ACF1
## Training set 0.007675574
##
## Forecasts:
##          Point Forecast      Lo 80     Hi 80      Lo 95     Hi 95
```

```
## Apr 2024    -32770.5927 -42834.540 -22706.65 -48162.071 -17379.11
## May 2024      -541.9856 -11453.724  10369.75 -17230.049  16146.08
## Jun 2024     24056.9306  12564.036  35549.83   6480.065  41633.80
## Jul 2024      3679.3688  -9332.054  16690.79 -16219.885  23578.62
## Aug 2024      1939.8647 -11081.974  14961.70 -17975.319  21855.05
## Sep 2024      1791.3742 -11230.540  14813.29 -18123.925  21706.67
## Oct 2024      1778.6985 -11243.216  14800.61 -18136.602  21694.00
## Nov 2024      1777.6164 -11244.298  14799.53 -18137.684  21692.92
## Dec 2024      1777.5241 -11244.391  14799.44 -18137.776  21692.82
## Jan 2025      1777.5162 -11244.399  14799.43 -18137.784  21692.82
## Feb 2025      1777.5155 -11244.399  14799.43 -18137.785  21692.82
## Mar 2025      1777.5154 -11244.399  14799.43 -18137.785  21692.82
## Apr 2025      1777.5154 -11244.399  14799.43 -18137.785  21692.82
## May 2025      1777.5154 -11244.399  14799.43 -18137.785  21692.82
## Jun 2025      1777.5154 -11244.399  14799.43 -18137.785  21692.82
## Jul 2025      1777.5154 -11244.399  14799.43 -18137.785  21692.82
## Aug 2025      1777.5154 -11244.399  14799.43 -18137.785  21692.82
## Sep 2025      1777.5154 -11244.399  14799.43 -18137.785  21692.82
```

```r
last_value <- tail(Price_Brent_ts, n = 1)
forecasted_values <- c(last_value, avg_Price_Brent_model_forecast$mean)
forecasted_values_L80 <- c(last_value, avg_Price_Brent_model_forecast$lower[, "80%"])
print(avg_Price_Brent_model_forecast$lower[, "80%"])
```

```
##             Jan        Feb        Mar        Apr        May        Jun
## 2024                                  -42834.540 -11453.724  12564.036
## 2025 -11244.399 -11244.399 -11244.399 -11244.399 -11244.399 -11244.399
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2024  -9332.054 -11081.974 -11230.540 -11243.216 -11244.298 -11244.391
## 2025 -11244.399 -11244.399 -11244.399
```

```r
forecasted_values_L95 <- c(last_value, avg_Price_Brent_model_forecast$lower[, "95%"])
print(avg_Price_Brent_model_forecast$lower[, "95%"])
```

```
##             Jan        Feb        Mar        Apr        May        Jun
## 2024                                  -48162.071 -17230.049   6480.065
## 2025 -18137.784 -18137.785 -18137.785 -18137.785 -18137.785 -18137.785
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2024 -16219.885 -17975.319 -18123.925 -18136.602 -18137.684 -18137.776
## 2025 -18137.785 -18137.785 -18137.785
```

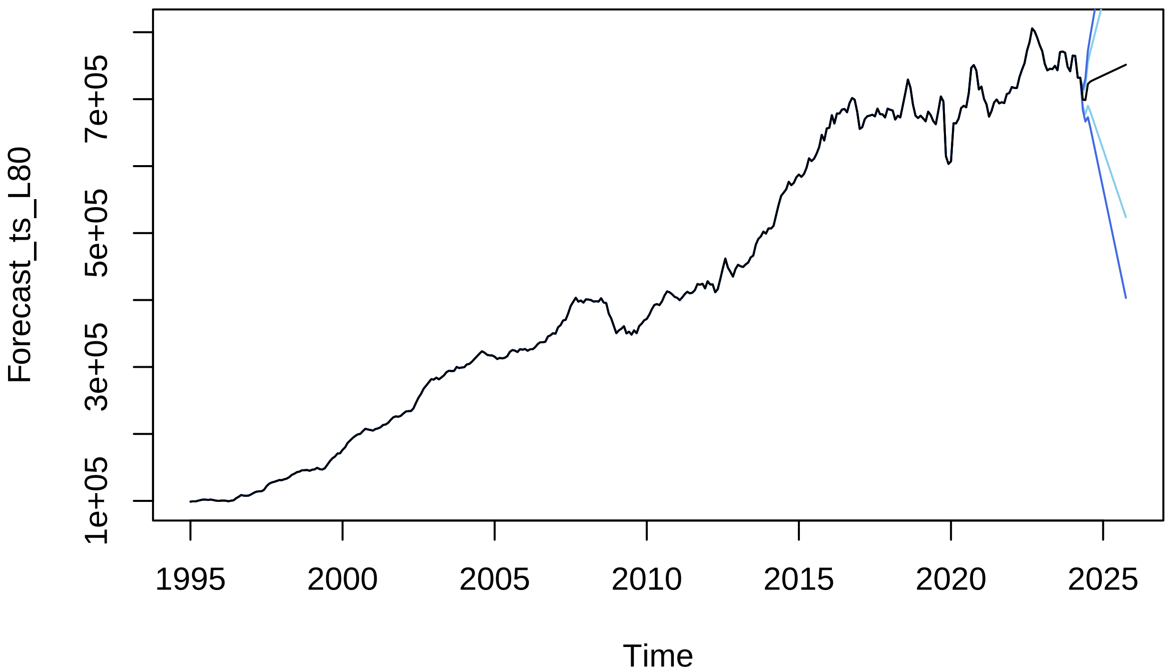
```r
forecasted_values_U80 <- c(last_value, avg_Price_Brent_model_forecast$upper[, "80%"])
print(avg_Price_Brent_model_forecast$upper[, "80%"])
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2024                               -22706.65  10369.75  35549.83  16690.79
## 2025  14799.43  14799.43  14799.43  14799.43  14799.43  14799.43  14799.43
##            Aug       Sep       Oct       Nov       Dec
## 2024  14961.70  14813.29  14800.61  14799.53  14799.44
## 2025  14799.43  14799.43
```

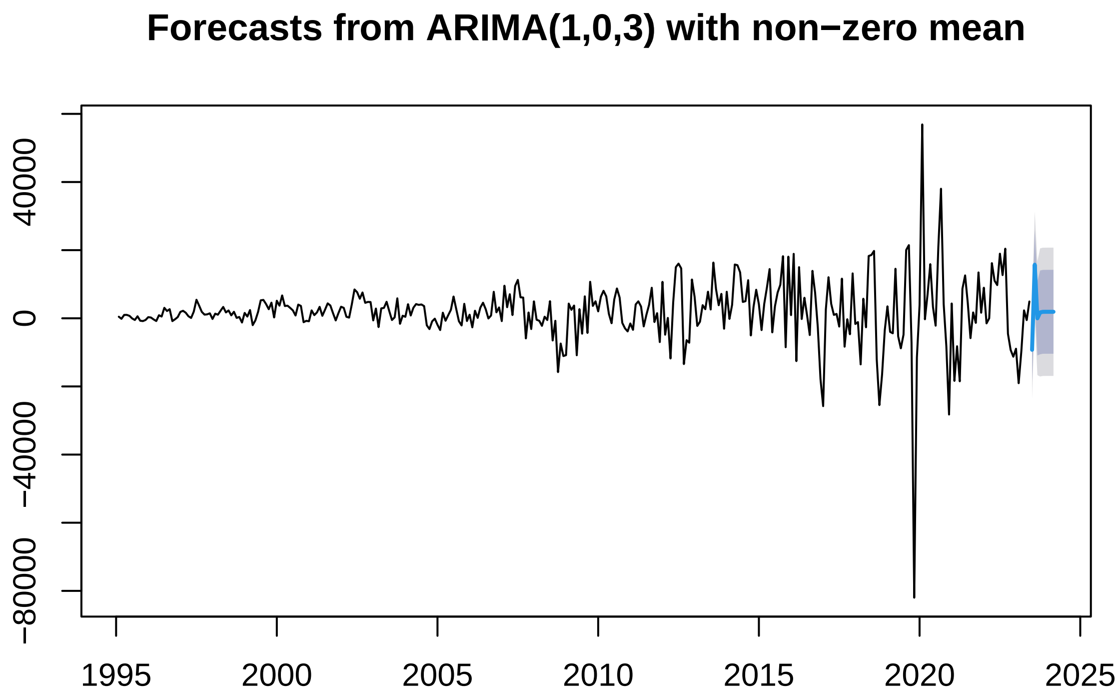
```r
forecasted_values_U95 <- c(last_value, avg_Price_Brent_model_forecast$upper[, "95%"])
print(avg_Price_Brent_model_forecast$upper[, "95%"])
```

```
##           Jan       Feb       Mar       Apr       May       Jun       Jul
## 2024                               -17379.11  16146.08  41633.80  23578.62
## 2025  21692.82  21692.82  21692.82  21692.82  21692.82  21692.82  21692.82
##           Aug       Sep       Oct       Nov       Dec
## 2024  21855.05  21706.67  21694.00  21692.92  21692.82
## 2025  21692.82  21692.82
```

```r
cumulative_forecasted_values_L80 <- cumsum(forecasted_values_L80)
cumulative_forecasted_values_L95 <- cumsum(forecasted_values_L95)
cumulative_forecasted_values_U80 <- cumsum(forecasted_values_U80)
cumulative_forecasted_values_U95 <- cumsum(forecasted_values_U95)
cumulative_forecasted_values <- cumsum(forecasted_values)
cumulative_forecasted_ts <- ts(cumulative_forecasted_values, start = c(2024, 2), frequency = 12)
cumulative_forecasted_ts_U80 <- ts(cumulative_forecasted_values_U80, start = c(2024, 2), frequency = 12
cumulative_forecasted_ts_U95 <- ts(cumulative_forecasted_values_U95, start = c(2024, 2), frequency = 12
cumulative_forecasted_ts_L80 <- ts(cumulative_forecasted_values_L80, start = c(2024, 2), frequency = 12
cumulative_forecasted_ts_L95 <- ts(cumulative_forecasted_values_L95, start = c(2024, 2), frequency = 12
combined <- c(Price_Brent_ts, cumulative_forecasted_ts)
combined_L80 <- c(Price_Brent_ts, cumulative_forecasted_ts_L80)
combined_L95 <- c(Price_Brent_ts, cumulative_forecasted_ts_L95)
combined_U80 <- c(Price_Brent_ts, cumulative_forecasted_ts_U80)
combined_U95 <- c(Price_Brent_ts, cumulative_forecasted_ts_U95)
Forecast_ts <- ts(combined, start = c(1995, 1), frequency = 12)
Forecast_ts_L80 <- ts(combined_L80, start = c(1995, 1), frequency = 12)
Forecast_ts_L95 <- ts(combined_L95, start = c(1995, 1), frequency = 12)
Forecast_ts_U80 <- ts(combined_U80, start = c(1995, 1), frequency = 12)
Forecast_ts_U95 <- ts(combined_U95, start = c(1995, 1), frequency = 12)
plot(Forecast_ts_L80, col = "skyblue")
lines(Forecast_ts_U80, col = "skyblue")
lines(Forecast_ts_L95, col = "royalblue")
lines(Forecast_ts_U95, col = "royalblue")
lines(Forecast_ts)
```



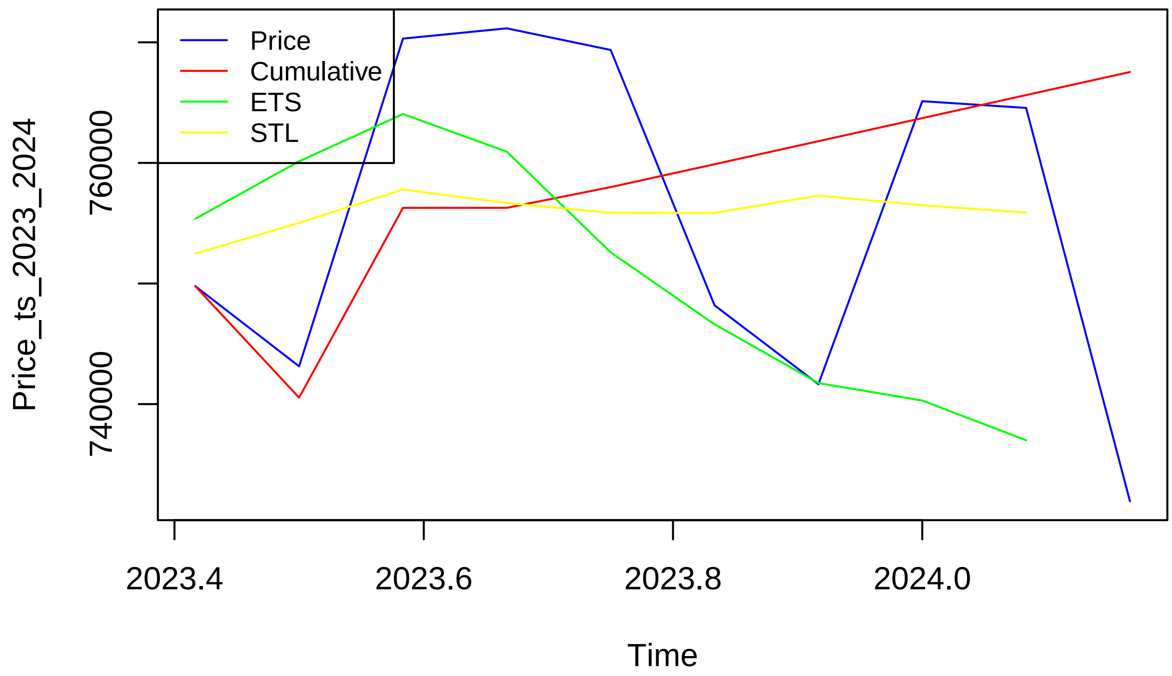

```r
avg_legacy <- arima(diff_legacy_ts, order = c(1, 0, 3))
avg_legacy_forecast <- forecast(avg_legacy, h=9)
plot(avg_legacy_forecast)
```




```
last_value_legacy <- tail(legacy_ts, n = 1)
forecasted_values_legacy <- c(last_value_legacy, avg_legacy_forecast$mean)
cumulative_forecasted_values_legacy <- cumsum(forecasted_values_legacy)
print(cumulative_forecasted_values_legacy)
```

```
##  [1] 749779.5 740536.0 756272.5 756274.9 757995.6 759888.9 761799.6 763712.1
##  [9] 765624.7 767537.4
```

```
cumulative_forecasted_ts_legacy <- ts(cumulative_forecasted_values_legacy, start = c(2023, 6), frequency

plot(Price_ts_2023_2024, type = "l", col = "blue")
lines(cumulative_forecasted_ts_legacy, col = "red")
lines(ets_forecasted_ts_legacy, col = "green")
lines(stl_forecasted_ts_legacy, col = "yellow")
legend("topleft",
       legend = c("Price", "Cumulative", "ETS", "STL"),
       col = c("blue", "red", "green", "yellow"),
       lty = 1,
       cex = 0.8)
```




```r
mse_avg <- mean((Price_ts_2023_2024 - cumulative_forecasted_ts_legacy)^2)
print(mse_avg)
```

```
## [1] 236722721
```

```r
mse_ets <- mean((Price_ts_2023_2024 - ets_forecasted_ts_legacy)^2)
print(mse_ets)
```

```
## [1] 235929800
```

```r
mse_stl <- mean((Price_ts_2023_2024 - stl_forecasted_ts_legacy)^2)
print(mse_stl)
```

```
## [1] 127873405
```

```r
mae_avg <- mae(Price_ts_2023_2024, cumulative_forecasted_ts_legacy)
print(mae_avg)
```

```
## [1] 11279.4
```

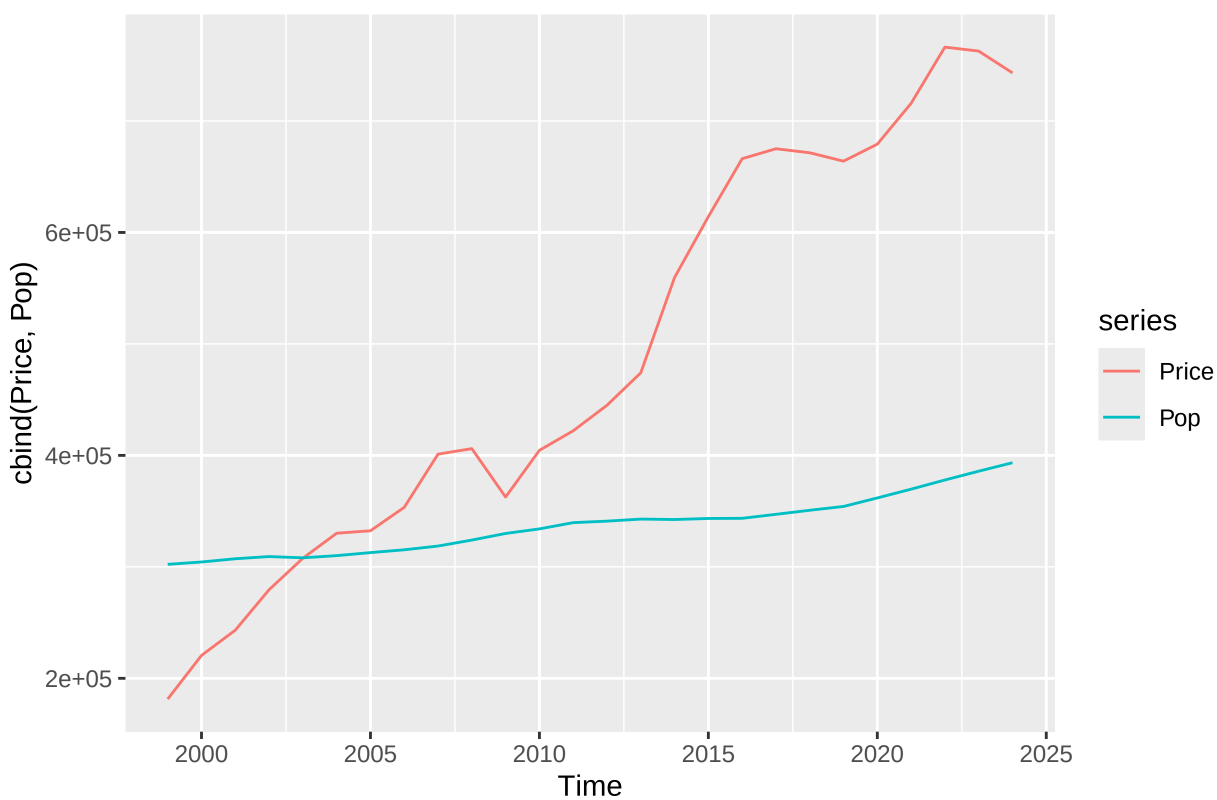
```r
mae_ets <- mae(Price_ts_2023_2024, ets_forecasted_ts_legacy)
print(mae_ets)
```

```
## [1] 12210.04
```

```r
mae_stl <- mae(Price_ts_2023_2024, stl_forecasted_ts_legacy)
print(mae_stl)
```

```
## [1] 10635.15
```

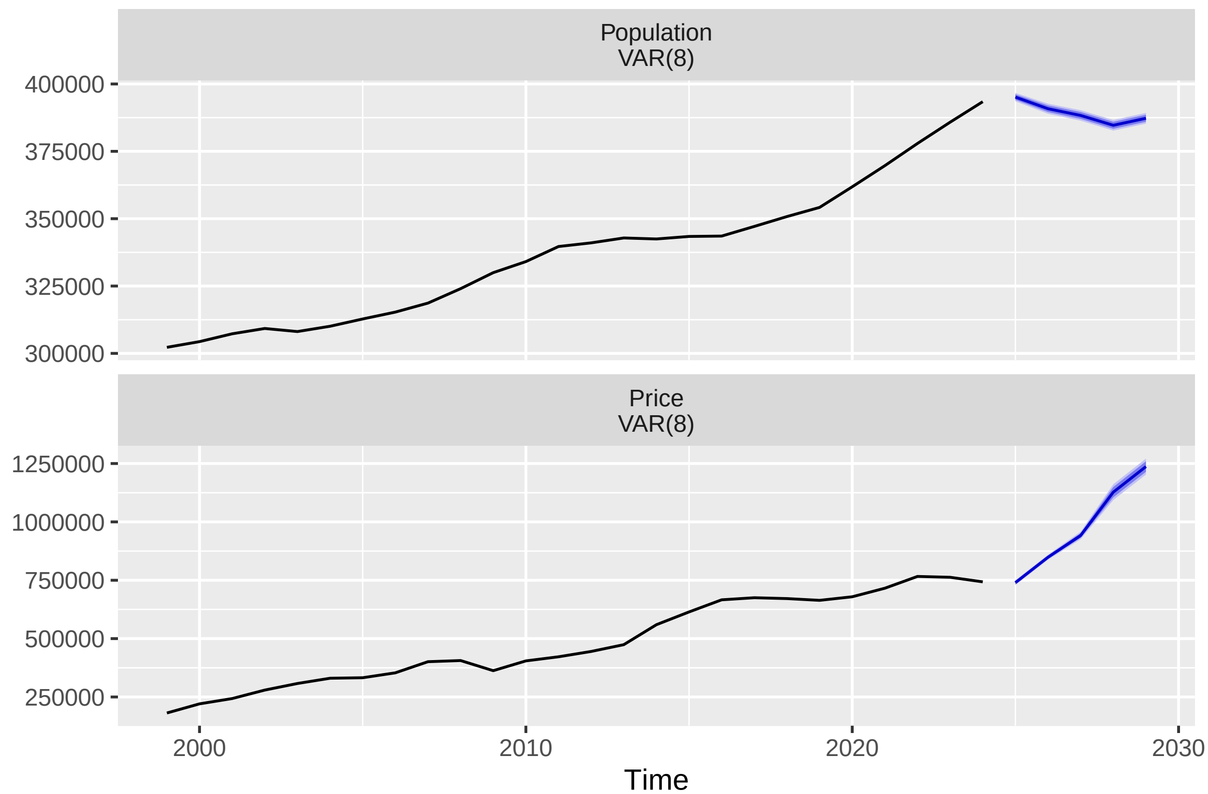
```r
Brent_df <- read.csv("Merged_Brent_Data.csv")

Brent_df$Date <- as.Date(Brent_df$Date, format = "%Y")

Price <- ts(Brent_df$Yearly_Price, start = c(1999), frequency = 1)

Pop <- ts(Brent_df$Population, start = c(1999), frequency = 1)

autoplot(cbind(Price, Pop))
```



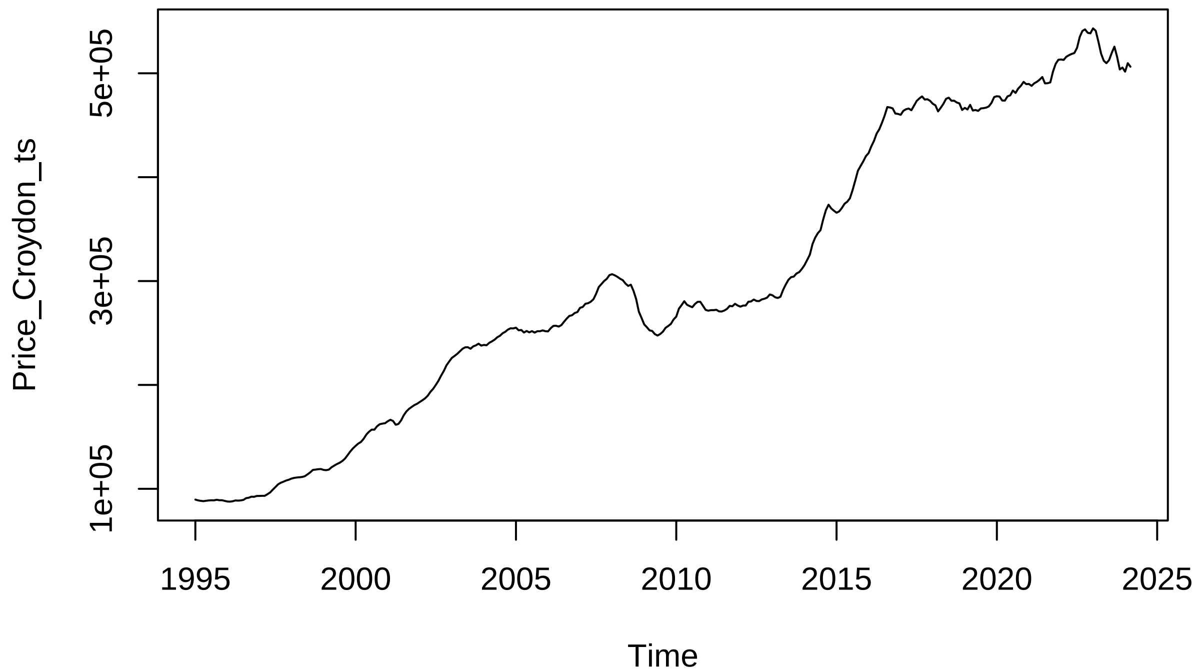

```r
Brent_df.bv <- cbind(Price, Pop)
colnames(Brent_df.bv) <- cbind("Price", "Population")

lagselect <- VARselect(Brent_df.bv, lag.max = 12, type = "const")
lagselect$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      7      7      7      7
```

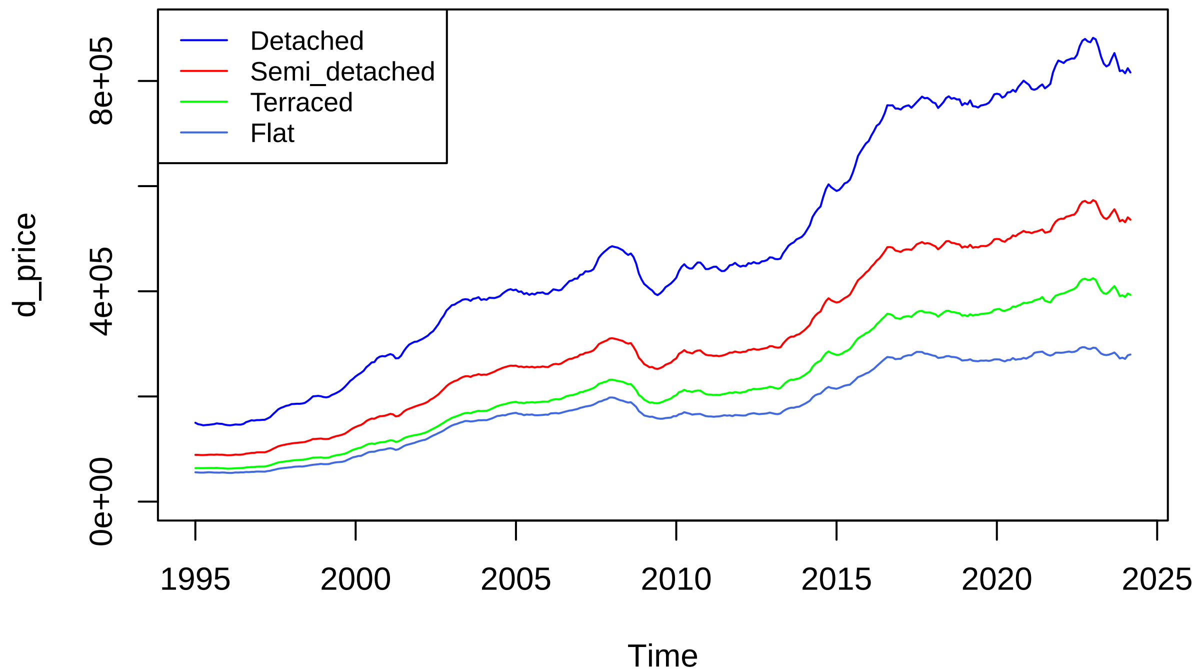
```r
var_model <- VAR(Brent_df.bv, p = 8, type = "const")

forecast_values <- forecast(var_model, h = 5)
autoplot(forecast_values)
```



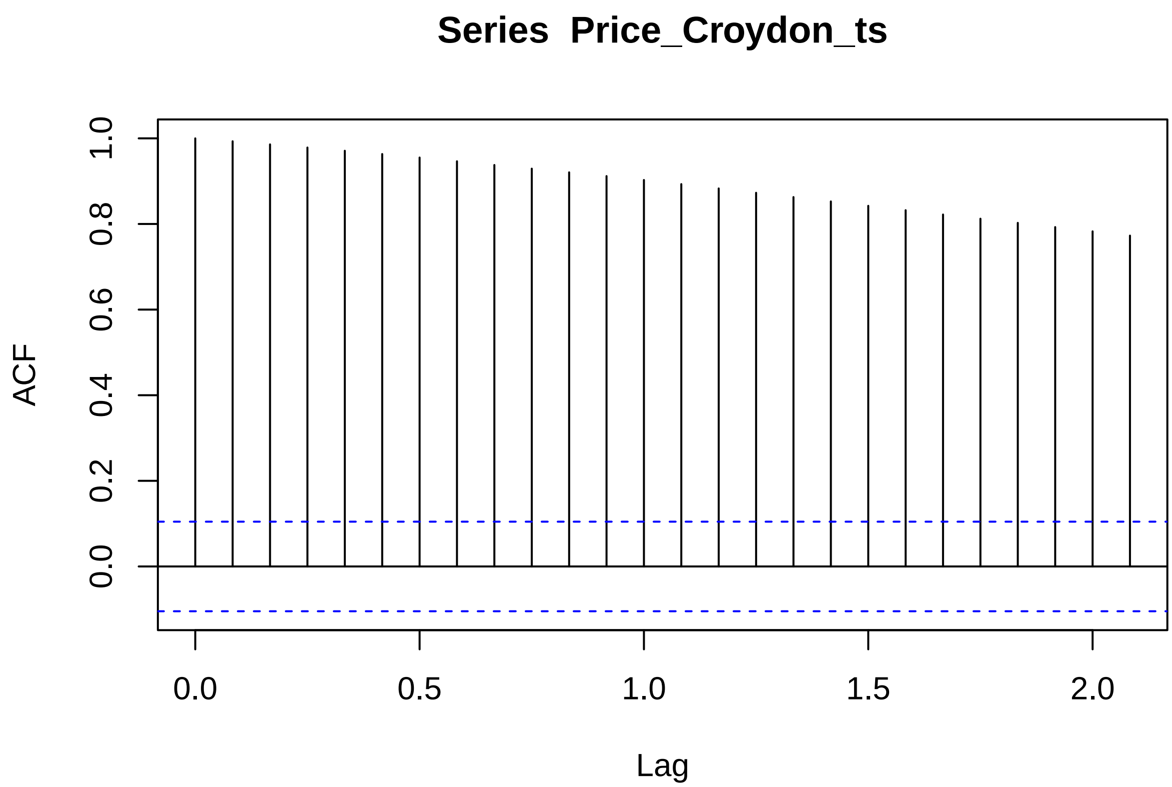

```r
library(tseries)
library(forecast)
library(vars)
library(readr)
library(Metrics)

Croydon_df <- read.csv("Updated_Croydon.csv")
Croydon_df$Date <- as.Date(Croydon_df$Date, format = "%Y-%m-%d")
Price_Croydon_ts <- ts(Croydon_df$Average_Price, start = c(1995, 1), frequency = 12)
Price_ts_2023_2024 <- window(Price_Croydon_ts, start = c(2023, 6), end = c(2024, 3))
d_price <- ts(Croydon_df$Detached_Average_Price, start = c(1995 , 1), frequency = 12)
sd_price <- ts(Croydon_df$Semi_Detached_Average_Price, start = c(1995, 1), frequency = 12)
t_price <- ts(Croydon_df$Terraced_Average_Price, start = c(1995, 1), frequency = 12)
f_price <- ts(Croydon_df$Flat_Average_Price, start = c(1995, 1), frequency = 12)
legacy_ts <- ts(Croydon_df$Average_Price, start = c(1995, 1), end = c(2023, 6), frequency = 12)
diff_legacy_ts <- diff(legacy_ts, differences=1)
plot(Price_Croydon_ts)
```



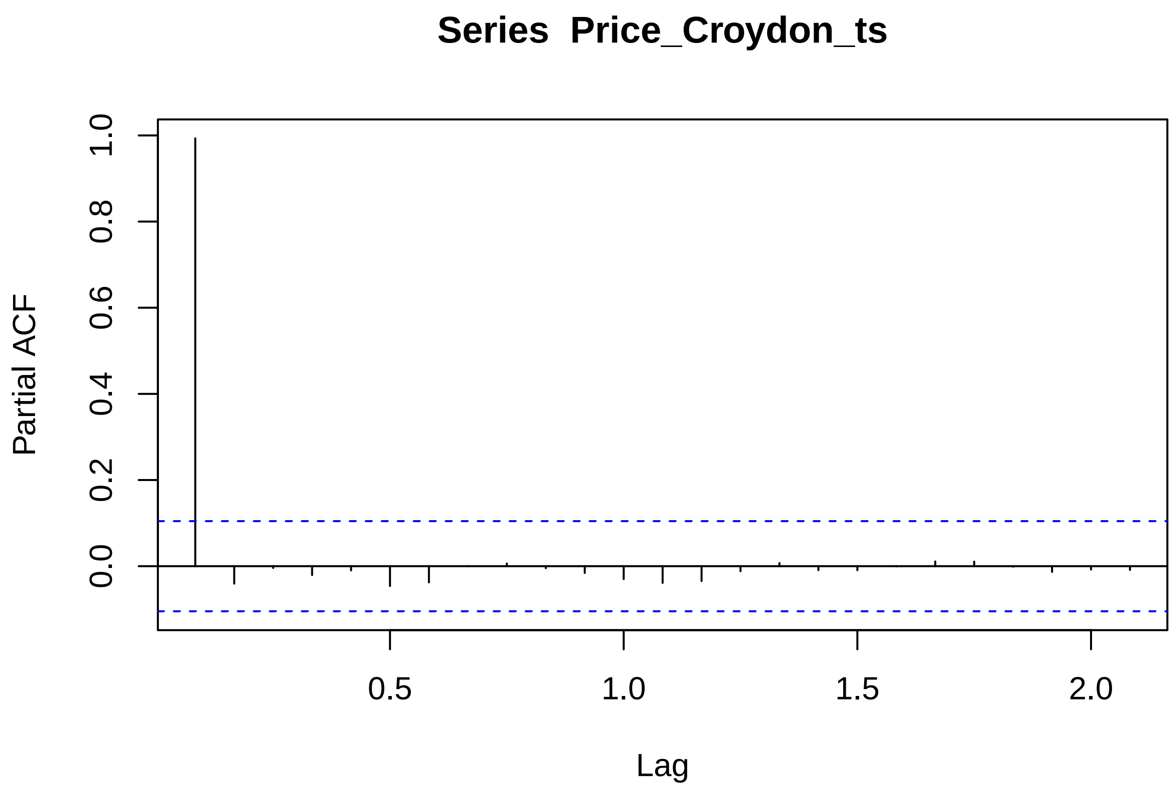

```r
plot(d_price, col = "blue", ylim = c(0, 900000))
lines(sd_price, col = "red")
lines(t_price, col = "green")
lines(f_price, col = "royalblue")
legend("topleft",
       legend = c("Detached", "Semi_detached", "Terraced", "Flat"),
       col = c("blue", "red", "green", "royalblue"),
       lty = 1,
       cex = 0.8)
```




```
acf(Price_Croydon_ts)
```




```r
pacf(Price_Croydon_ts)
```




```r
adf.test(Price_Croydon_ts)
```

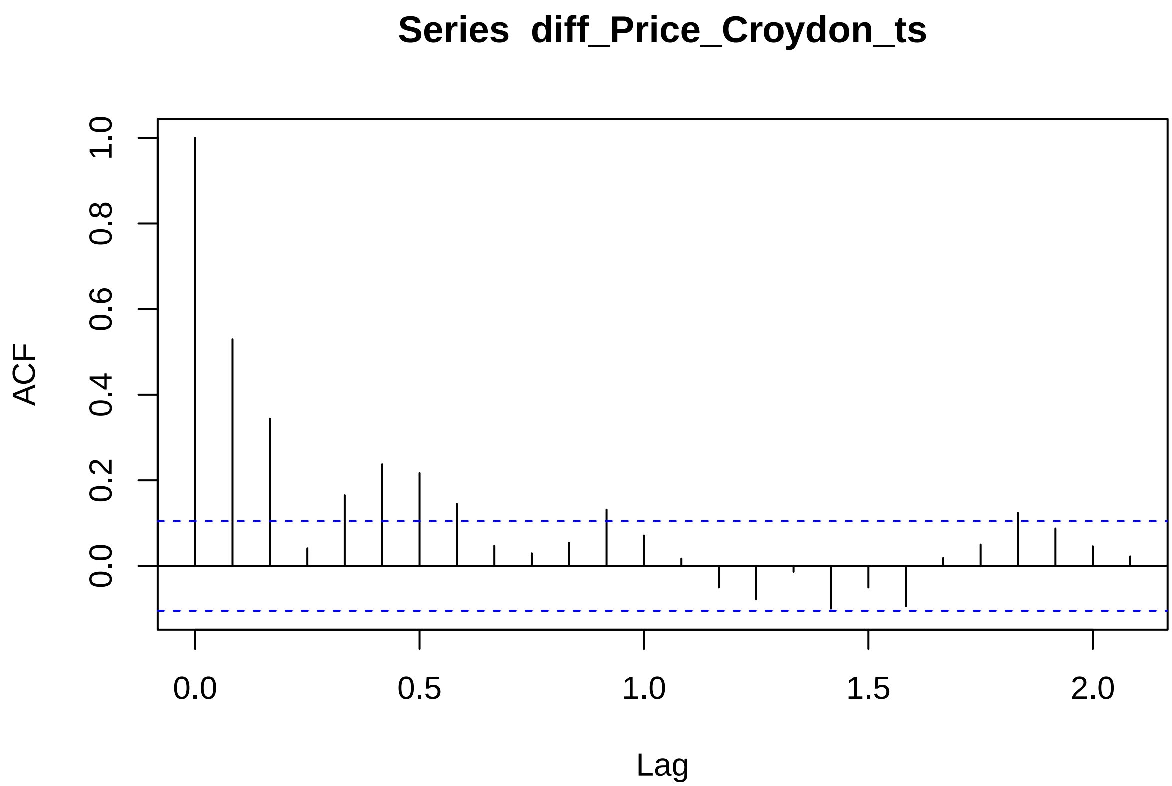
```
##
##  Augmented Dickey-Fuller Test
##
## data:  Price_Croydon_ts
## Dickey-Fuller = -2.3457, Lag order = 7, p-value = 0.4309
## alternative hypothesis: stationary
```

```r
diff_Price_Croydon_ts <- diff(Price_Croydon_ts, differences=1)
adf.test(diff_Price_Croydon_ts)
```

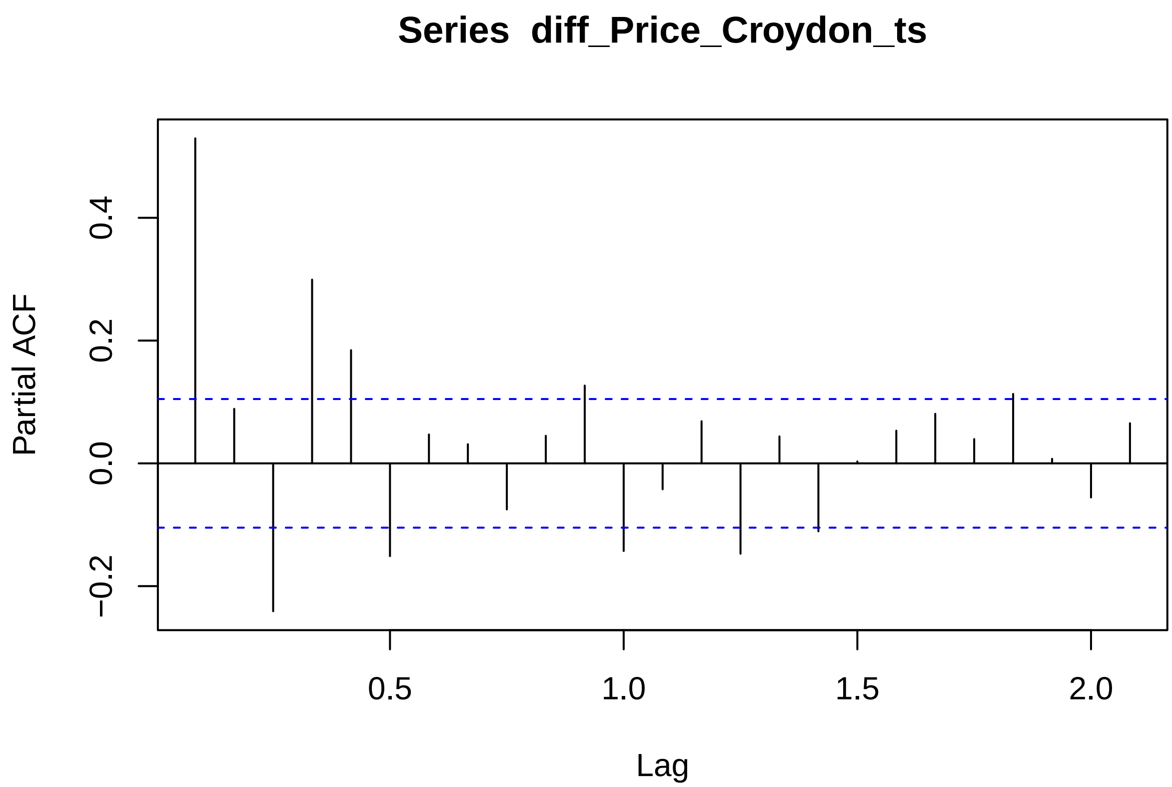
```
## Warning in adf.test(diff_Price_Croydon_ts): p-value smaller than printed
## p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_Price_Croydon_ts
## Dickey-Fuller = -4.6989, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

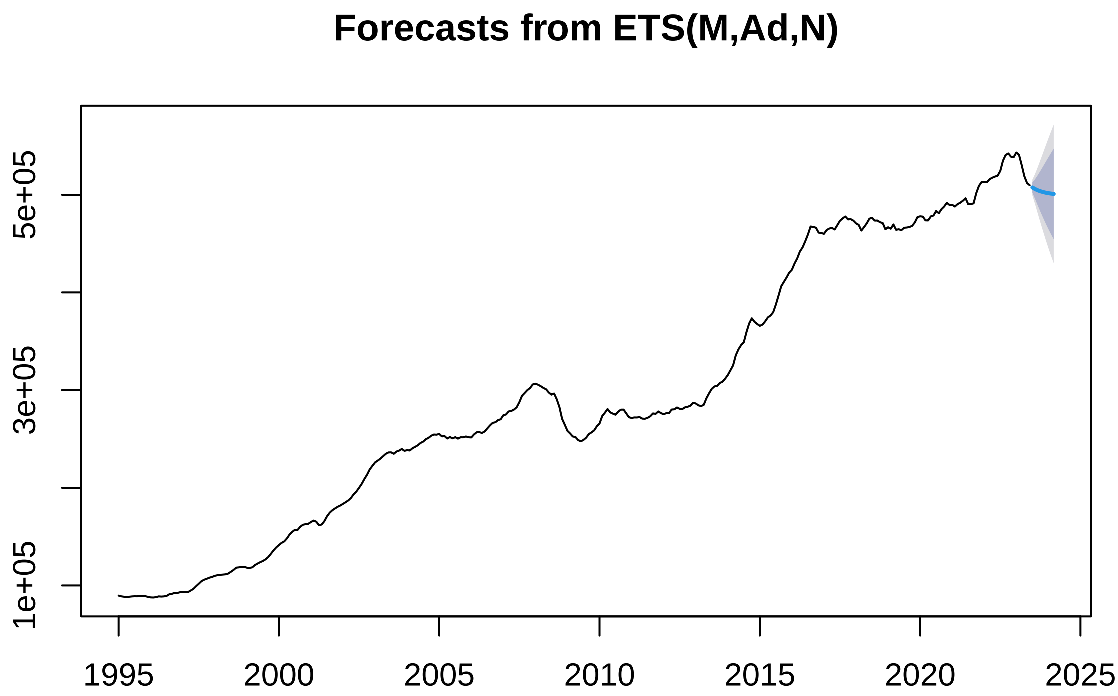
```r
acf(diff_Price_Croydon_ts)
```



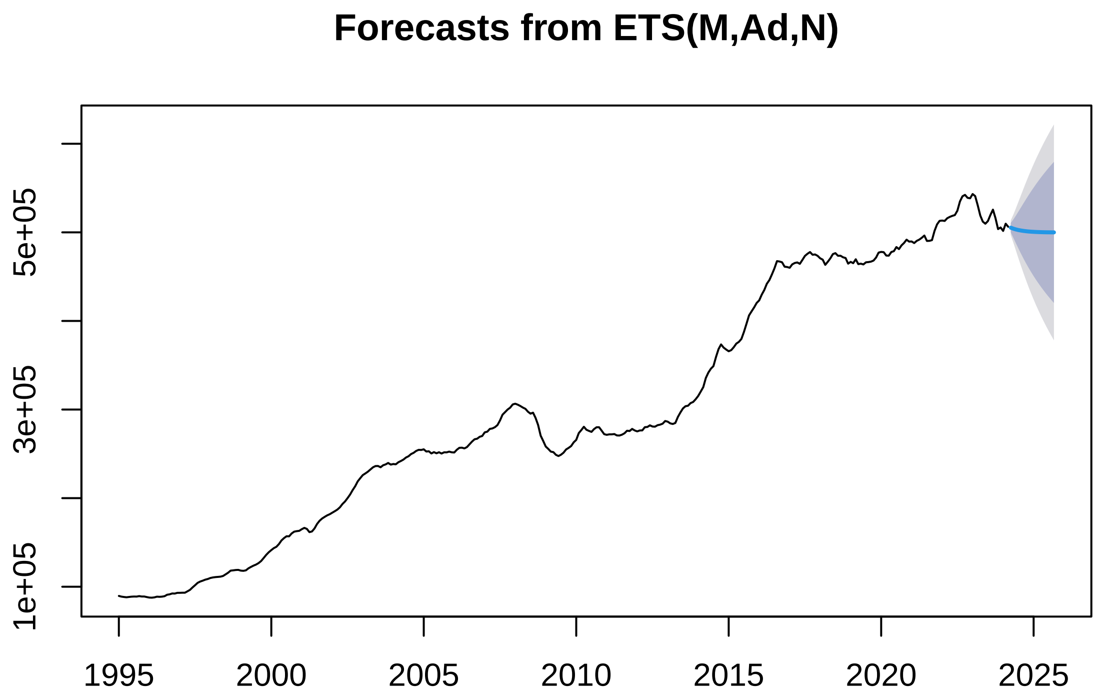

```r
pacf(diff_Price_Croydon_ts)
```




```
ets_Price_Croydon_model <- ets(Price_Croydon_ts, model = "ZZZ")
ets_Price_Croydon_model_forecast <- forecast(ets_Price_Croydon_model, h=18)
ets_legacy <- ets(legacy_ts, model = "MAN")
ets_legacy_forecast <- forecast(ets_legacy, h = 9)
ets_forecasted_values_legacy <- ets_legacy_forecast$mean
ets_forecasted_ts_legacy <- ts(ets_forecasted_values_legacy, start = c(2023, 6), frequency = 12)
plot(ets_legacy_forecast)
```




```
plot(ets_Price_Croydon_model_forecast)
```

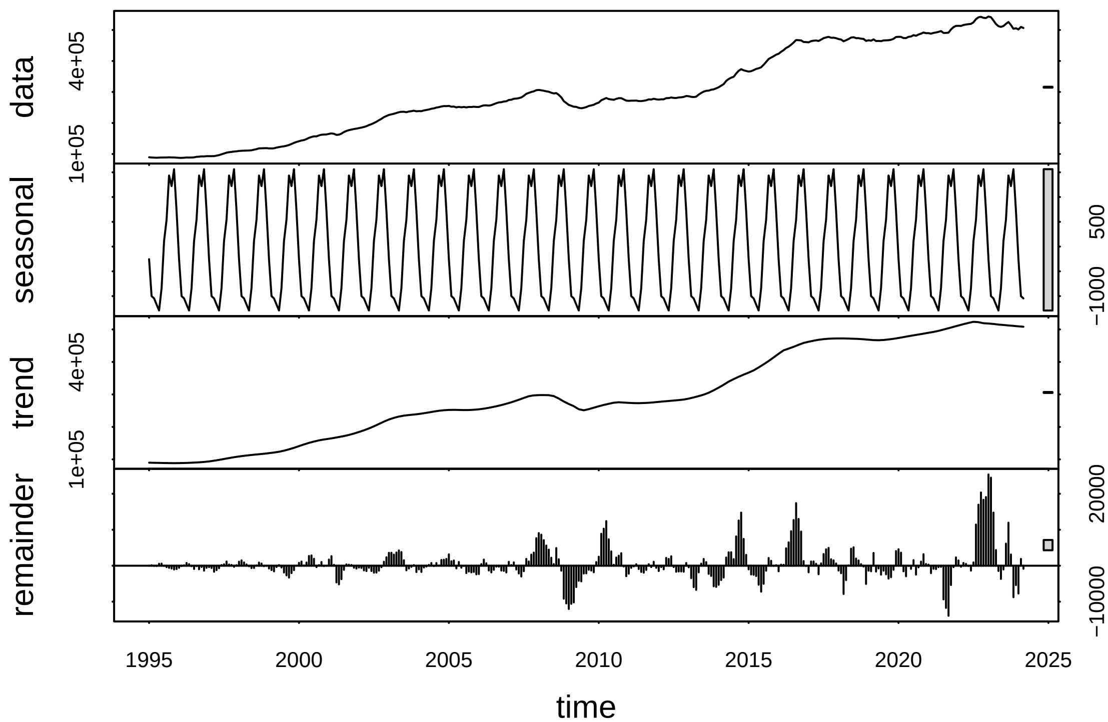



```r
summary(ets_Price_Croydon_model_forecast)
```

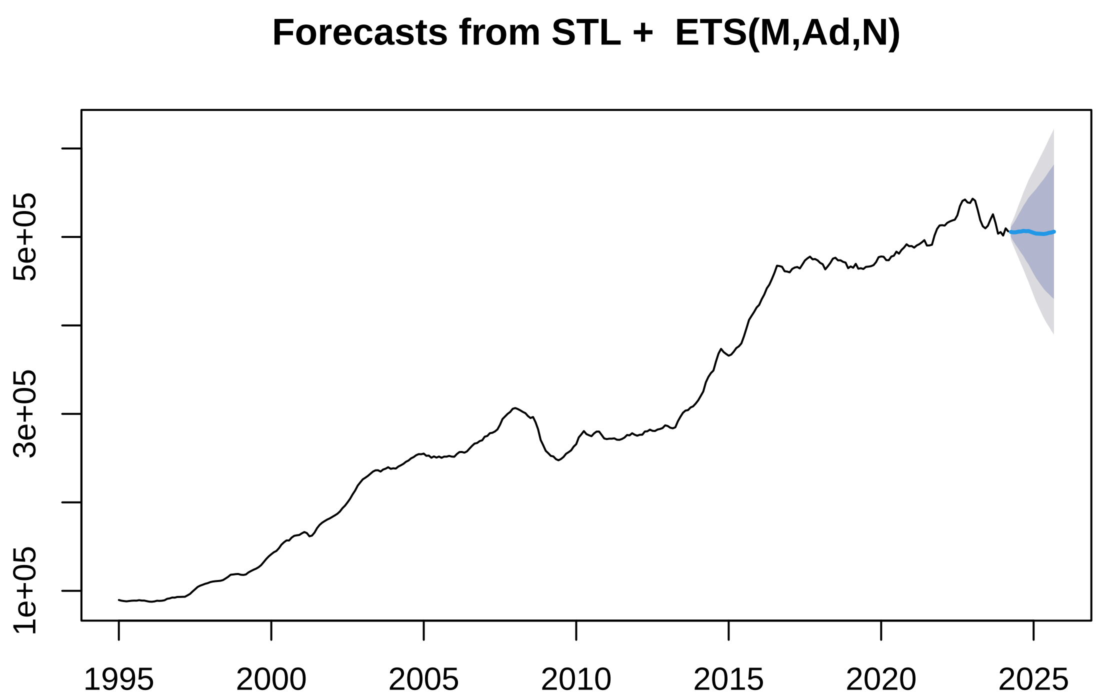
```
##
## Forecast method: ETS(M,Ad,N)
##
## Model Information:
## ETS(M,Ad,N)
##
## Call:
##  ets(y = Price_Croydon_ts, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.9554
##     beta  = 0.8375
##     phi   = 0.8
##
##   Initial states:
##     l = 88745.0343
##     b = 19.8803
##
##   sigma:  0.0083
##
##      AIC     AICc      BIC
## 7457.877 7458.121 7481.042
##
```

```
## Error measures:
##                    ME     RMSE      MAE       MPE      MAPE       MASE
## Training set 271.9737 2838.049 1947.398 0.1207829 0.6315575 0.09461202
##                     ACF1
## Training set -0.06770104
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Apr 2024       505282.3 499924.8 510639.8 497088.7 513475.9
## May 2024       504191.2 493972.7 514409.6 488563.4 519818.9
## Jun 2024       503318.3 487887.7 518748.9 479719.2 526917.4
## Jul 2024       502620.0 481887.9 523352.1 470913.0 534326.9
## Aug 2024       502061.3 476074.1 528048.6 462317.2 541805.4
## Sep 2024       501614.4 470492.4 532736.5 454017.4 549211.5
## Oct 2024       501256.9 465160.2 537353.6 446051.8 556462.0
## Nov 2024       500970.9 460078.9 541862.9 438432.0 563509.8
## Dec 2024       500742.1 455240.6 546243.5 431153.5 570330.6
## Jan 2025       500559.0 450632.8 550485.2 424203.4 576914.6
## Feb 2025       500412.6 446240.6 554584.5 417563.7 583261.4
## Mar 2025       500295.4 442048.5 558542.3 411214.5 589376.3
## Apr 2025       500201.7 438040.9 562362.4 405135.0 595268.3
## May 2025       500126.7 434202.9 566050.4 399305.0 600948.3
## Jun 2025       500066.7 430520.7 569612.8 393705.2 606428.2
## Jul 2025       500018.7 426981.1 573056.3 388317.3 611720.1
## Aug 2025       499980.3 423572.5 576388.2 383124.6 616836.1
## Sep 2025       499949.6 420283.9 579615.3 378111.5 621787.7
```

```r
stl_Price_Croydon_model <- stl(Price_Croydon_ts, s.window="periodic", robust=TRUE)
plot(stl_Price_Croydon_model)
```




```
stl_Price_Croydon_model_forecast <- forecast(stl_Price_Croydon_model, method="ets", h=18)
plot(stl_Price_Croydon_model_forecast)
```

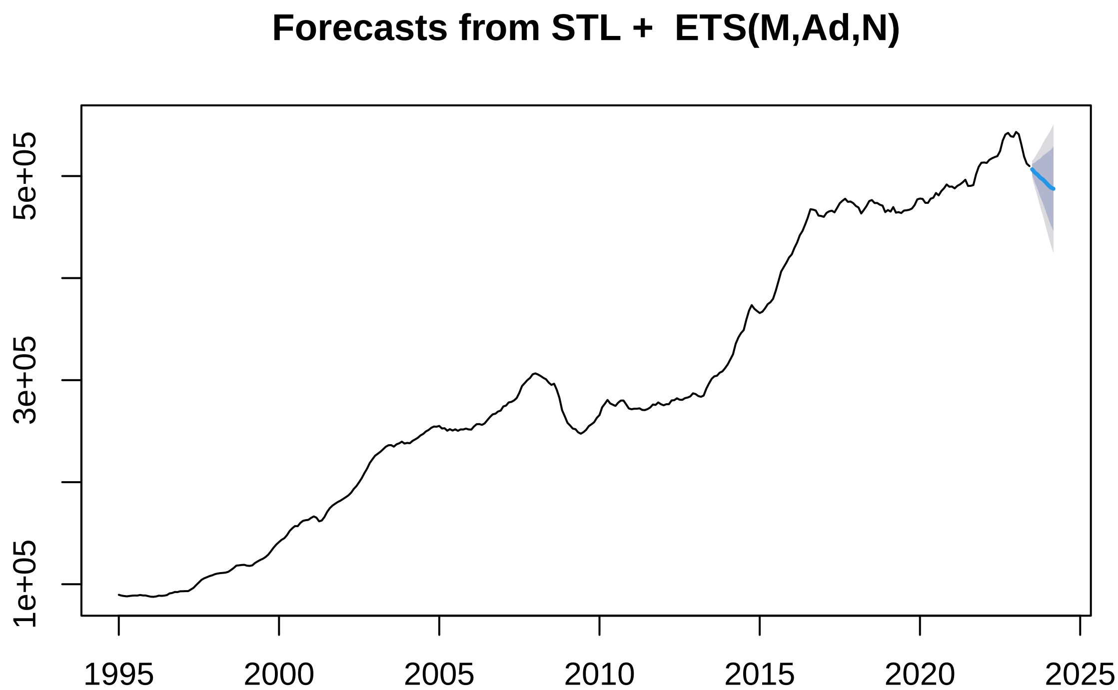



```
summary(stl_Price_Croydon_model_forecast)
```

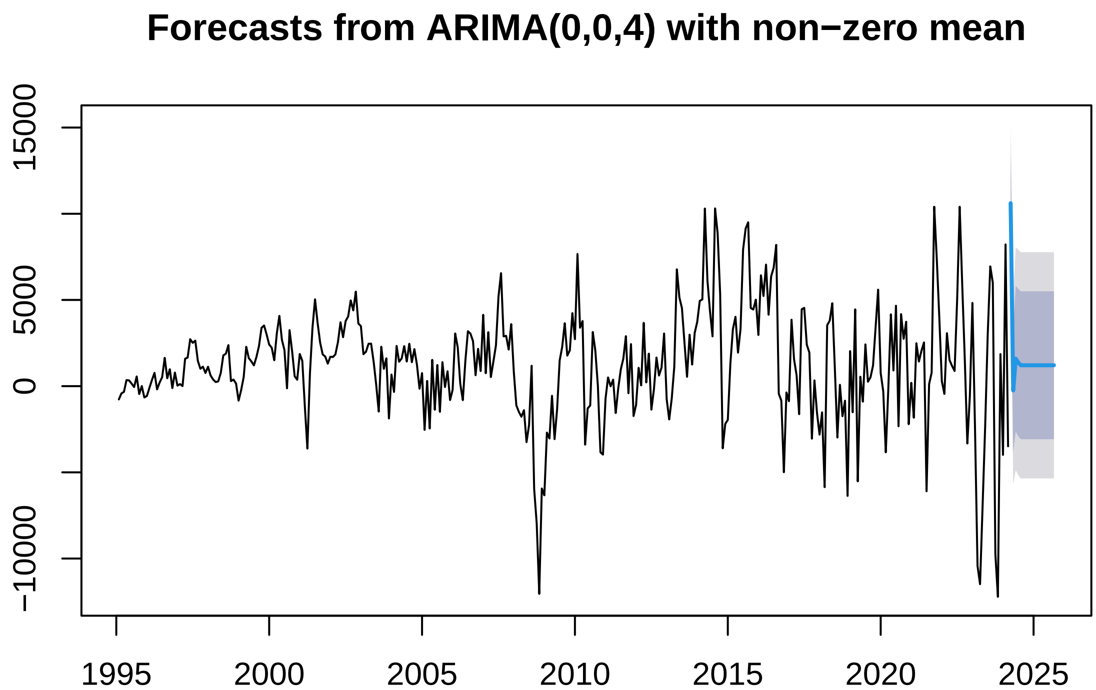
```
##
## Forecast method: STL +  ETS(M,Ad,N)
##
## Model Information:
## ETS(M,Ad,N)
##
## Call:
##  ets(y = na.interp(x), model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.6123
##     phi   = 0.8427
##
##   Initial states:
##     l = 90392.3146
##     b = -123.3315
##
##   sigma:  0.0084
##
##      AIC     AICc      BIC
## 7464.994 7465.238 7488.159
##
```

```
## Error measures:
##                    ME     RMSE      MAE       MPE      MAPE       MASE
## Training set 275.9154 2840.452 1932.484 0.1223963 0.6366755 0.09388743
##                      ACF1
## Training set -0.008402947
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Apr 2024       505691.3 500262.8 511119.9 497389.1 513993.6
## May 2024       505174.6 495318.2 515031.0 490100.6 520248.6
## Jun 2024       505292.3 490829.7 519755.0 483173.6 527411.1
## Jul 2024       505961.5 486800.1 525122.9 476656.7 535266.3
## Aug 2024       506156.5 482282.1 530030.9 469643.7 542669.2
## Sep 2024       506848.7 478301.8 535395.7 463189.9 550507.6
## Oct 2024       506465.5 473322.9 539608.2 455778.2 557152.9
## Nov 2024       506666.0 469027.8 544304.1 449103.4 564228.5
## Dec 2024       505707.9 463688.7 547727.2 441445.0 569970.8
## Jan 2025       504629.5 458351.2 550907.8 433852.9 575406.0
## Feb 2025       503797.2 453385.2 554209.3 426698.7 580895.8
## Mar 2025       503679.7 449259.3 558100.0 420450.9 586908.4
## Apr 2025       503495.9 445190.7 561801.2 414325.8 592666.1
## May 2025       503324.4 441254.4 565394.5 408396.4 598252.5
## Jun 2025       503733.1 438013.8 569452.5 403224.1 604242.1
## Jul 2025       504647.5 435389.8 573905.3 398726.9 610568.1
## Aug 2025       505049.1 432358.5 577739.7 393878.4 616219.8
## Sep 2025       505915.5 429892.4 581938.7 389648.2 622182.9
```

```r
stl_legacy <- stl(legacy_ts, s.window="periodic", robust=TRUE)
stl_legacy_forecast <- forecast(stl_legacy, h=9)
plot(stl_legacy_forecast)
```




```
stl_forecasted_values_legacy <- stl_legacy_forecast$mean
stl_forecasted_ts_legacy <- ts(stl_forecasted_values_legacy, start = c(2023, 6), frequency = 12)

avg_Price_Croydon_model <- auto.arima(diff_Price_Croydon_ts, seasonal=FALSE)
avg_Price_Croydon_model_forecast <- forecast(avg_Price_Croydon_model, h=18)
plot(avg_Price_Croydon_model_forecast)
```




```r
summary(avg_Price_Croydon_model_forecast)
```

```
##
## Forecast method: ARIMA(0,0,4) with non-zero mean
##
## Model Information:
## Series: diff_Price_Croydon_ts
## ARIMA(0,0,4) with non-zero mean
##
## Coefficients:
##          ma1     ma2      ma3     ma4       mean
##       0.6538  0.7542  -0.2192  0.1055  1210.2435
## s.e.  0.0545  0.0583   0.0578  0.0539   283.6067
##
## sigma^2 = 5451689:  log likelihood = -3211.54
## AIC=6435.09   AICc=6435.33   BIC=6458.23
##
## Error measures:
##                    ME     RMSE      MAE      MPE     MAPE      MASE        ACF1
## Training set 2.466439 2318.148 1599.495 1.608155 250.4054 0.5145618 0.008205948
##
## Forecasts:
##          Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## Apr 2024     10615.4111  7623.121 13607.701  6039.099 15191.724
## May 2024      -251.7973 -3826.785  3323.191 -5719.270  5215.675
```

```
## Jun 2024      1590.3910 -2637.293  5818.075 -4875.294  8056.076
## Jul 2024      1368.0256 -2910.251  5646.302 -5175.033  7911.084
## Aug 2024      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Sep 2024      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Oct 2024      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Nov 2024      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Dec 2024      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Jan 2025      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Feb 2025      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Mar 2025      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Apr 2025      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## May 2025      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Jun 2025      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Jul 2025      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Aug 2025      1210.2435 -3079.669  5500.156 -5350.611  7771.098
## Sep 2025      1210.2435 -3079.669  5500.156 -5350.611  7771.098
```

```r
last_value <- tail(Price_Croydon_ts, n = 1)
forecasted_values <- c(last_value, avg_Price_Croydon_model_forecast$mean)
forecasted_values_L80 <- c(last_value, avg_Price_Croydon_model_forecast$lower[, "80%"])
print(avg_Price_Croydon_model_forecast$lower[, "80%"])
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2024                                7623.121 -3826.785 -2637.293 -2910.251
## 2025 -3079.669 -3079.669 -3079.669 -3079.669 -3079.669 -3079.669 -3079.669
##            Aug       Sep       Oct       Nov       Dec
## 2024 -3079.669 -3079.669 -3079.669 -3079.669 -3079.669
## 2025 -3079.669 -3079.669
```

```r
forecasted_values_L95 <- c(last_value, avg_Price_Croydon_model_forecast$lower[, "95%"])
print(avg_Price_Croydon_model_forecast$lower[, "95%"])
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2024                                6039.099 -5719.270 -4875.294 -5175.033
## 2025 -5350.611 -5350.611 -5350.611 -5350.611 -5350.611 -5350.611 -5350.611
##            Aug       Sep       Oct       Nov       Dec
## 2024 -5350.611 -5350.611 -5350.611 -5350.611 -5350.611
## 2025 -5350.611 -5350.611
```

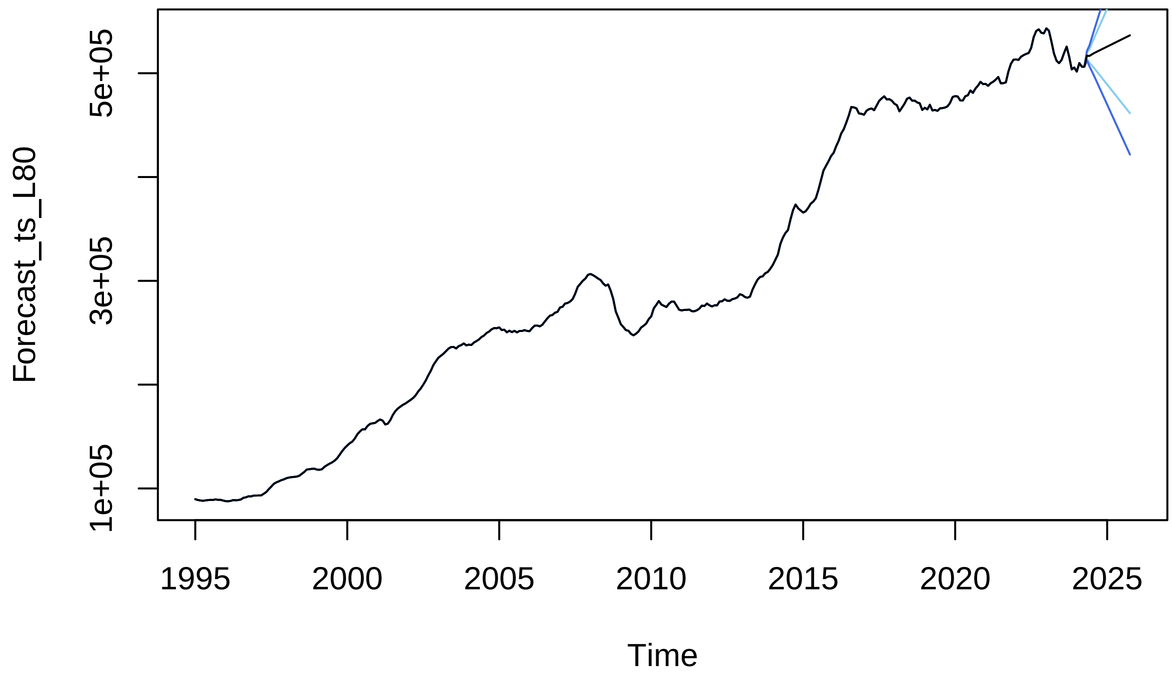
```r
forecasted_values_U80 <- c(last_value, avg_Price_Croydon_model_forecast$upper[, "80%"])
print(avg_Price_Croydon_model_forecast$upper[, "80%"])
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2024                               13607.701  3323.191  5818.075  5646.302
## 2025  5500.156  5500.156  5500.156  5500.156  5500.156  5500.156  5500.156
##            Aug       Sep       Oct       Nov       Dec
## 2024  5500.156  5500.156  5500.156  5500.156  5500.156
## 2025  5500.156  5500.156
```

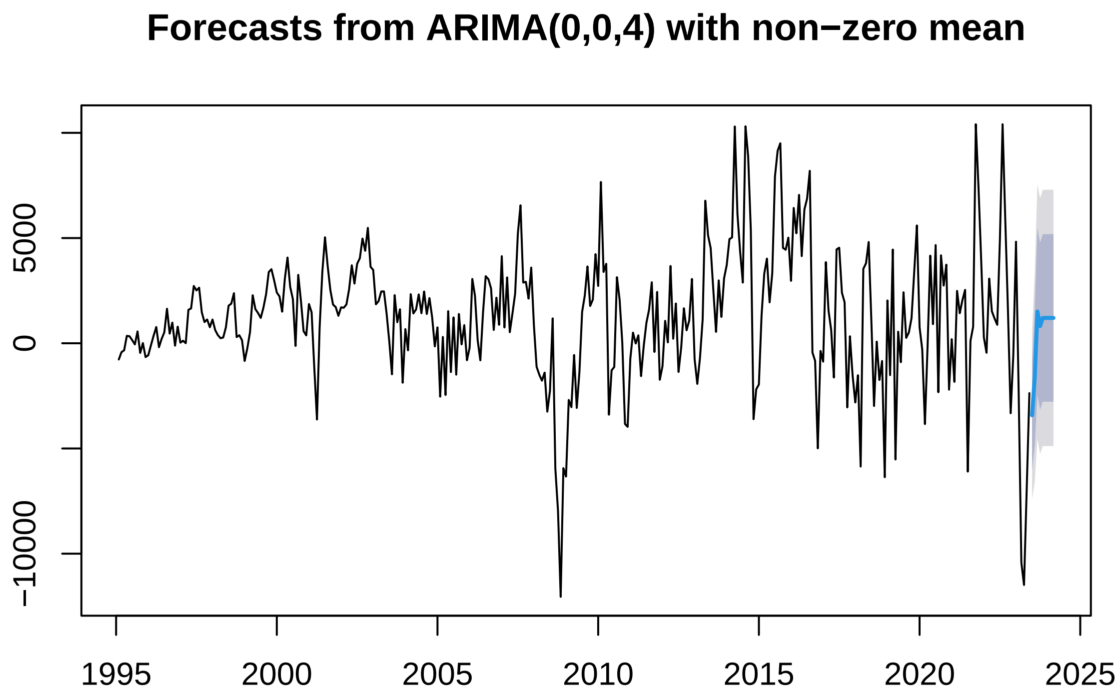
```r
forecasted_values_U95 <- c(last_value, avg_Price_Croydon_model_forecast$upper[, "95%"])
print(avg_Price_Croydon_model_forecast$upper[, "95%"])
```

```
##           Jan       Feb       Mar       Apr       May       Jun       Jul
## 2024                                15191.724  5215.675  8056.076  7911.084
## 2025  7771.098  7771.098  7771.098  7771.098  7771.098  7771.098  7771.098
##           Aug       Sep       Oct       Nov       Dec
## 2024  7771.098  7771.098  7771.098  7771.098  7771.098
## 2025  7771.098  7771.098
```

```
cumulative_forecasted_values_L80 <- cumsum(forecasted_values_L80)
cumulative_forecasted_values_L95 <- cumsum(forecasted_values_L95)
cumulative_forecasted_values_U80 <- cumsum(forecasted_values_U80)
cumulative_forecasted_values_U95 <- cumsum(forecasted_values_U95)
cumulative_forecasted_values <- cumsum(forecasted_values)
cumulative_forecasted_ts <- ts(cumulative_forecasted_values, start = c(2024, 2), frequency = 12)
cumulative_forecasted_ts_U80 <- ts(cumulative_forecasted_values_U80, start = c(2024, 2), frequency = 12
cumulative_forecasted_ts_U95 <- ts(cumulative_forecasted_values_U95, start = c(2024, 2), frequency = 12
cumulative_forecasted_ts_L80 <- ts(cumulative_forecasted_values_L80, start = c(2024, 2), frequency = 12
cumulative_forecasted_ts_L95 <- ts(cumulative_forecasted_values_L95, start = c(2024, 2), frequency = 12
combined <- c(Price_Croydon_ts, cumulative_forecasted_ts)
combined_L80 <- c(Price_Croydon_ts, cumulative_forecasted_ts_L80)
combined_L95 <- c(Price_Croydon_ts, cumulative_forecasted_ts_L95)
combined_U80 <- c(Price_Croydon_ts, cumulative_forecasted_ts_U80)
combined_U95 <- c(Price_Croydon_ts, cumulative_forecasted_ts_U95)
Forecast_ts <- ts(combined, start = c(1995, 1), frequency = 12)
Forecast_ts_L80 <- ts(combined_L80, start = c(1995, 1), frequency = 12)
Forecast_ts_L95 <- ts(combined_L95, start = c(1995, 1), frequency = 12)
Forecast_ts_U80 <- ts(combined_U80, start = c(1995, 1), frequency = 12)
Forecast_ts_U95 <- ts(combined_U95, start = c(1995, 1), frequency = 12)
plot(Forecast_ts_L80, col = "skyblue")
lines(Forecast_ts_U80, col = "skyblue")
lines(Forecast_ts_L95, col = "royalblue")
lines(Forecast_ts_U95, col = "royalblue")
lines(Forecast_ts)
```



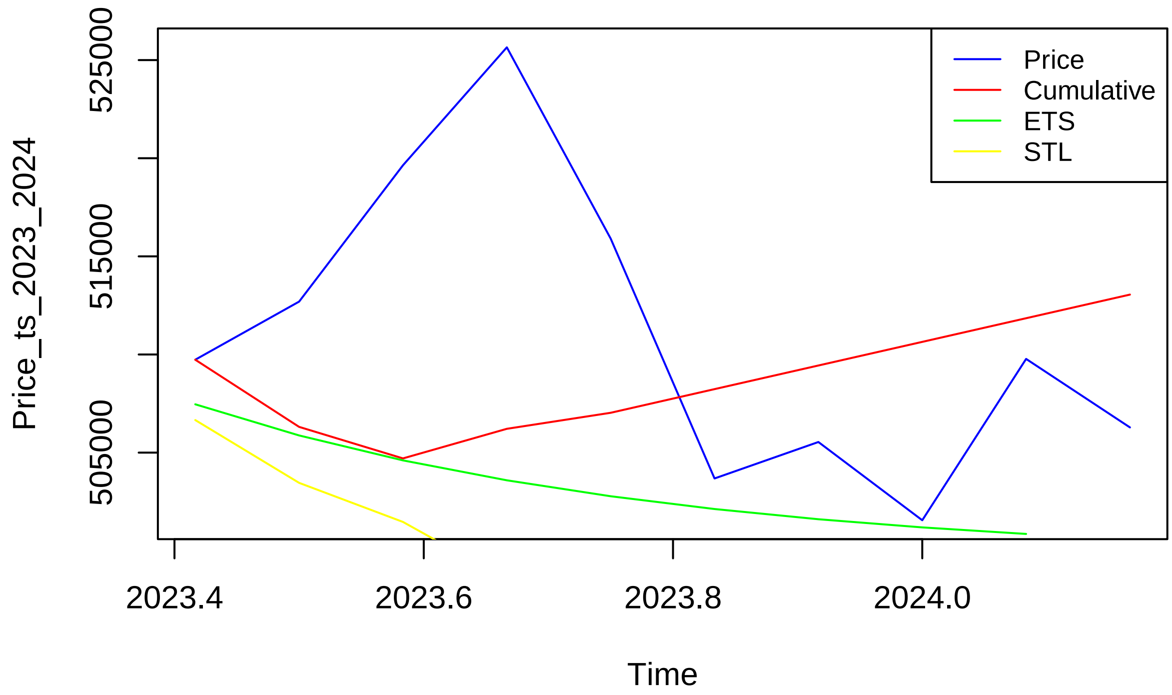

```r
avg_legacy <- arima(diff_legacy_ts, order = c(0, 0, 4))
avg_legacy_forecast <- forecast(avg_legacy, h=9)
plot(avg_legacy_forecast)
```




```
last_value_legacy <- tail(legacy_ts, n = 1)
forecasted_values_legacy <- c(last_value_legacy, avg_legacy_forecast$mean)
cumulative_forecasted_values_legacy <- cumsum(forecasted_values_legacy)
print(cumulative_forecasted_values_legacy)
```

```
##  [1] 509733.8 506311.5 504704.5 506213.6 507029.8 508234.2 509438.6 510643.1
##  [9] 511847.5 513051.9
```

```
cumulative_forecasted_ts_legacy <- ts(cumulative_forecasted_values_legacy, start = c(2023, 6), frequency

plot(Price_ts_2023_2024, type = "l", col = "blue")
lines(cumulative_forecasted_ts_legacy, col = "red")
lines(ets_forecasted_ts_legacy, col = "green")
lines(stl_forecasted_ts_legacy, col = "yellow")
legend("topright",
       legend = c("Price", "Cumulative", "ETS", "STL"),
       col = c("blue", "red", "green", "yellow"),
       lty = 1,
       cex = 0.8)
```




```
mse_avg <- mean((Price_ts_2023_2024 - cumulative_forecasted_ts_legacy)^2)
print(mse_avg)
```

```
## [1] 88885170
```

```
mse_ets <- mean((Price_ts_2023_2024 - ets_forecasted_ts_legacy)^2)
print(mse_ets)
```

```
## [1] 114910111
```

```
mse_stl <- mean((Price_ts_2023_2024 - stl_forecasted_ts_legacy)^2)
print(mse_stl)
```

```
## [1] 275808912
```

```
mae_avg <- mae(Price_ts_2023_2024, cumulative_forecasted_ts_legacy)
print(mae_avg)
```

```
## [1] 7600.35
```

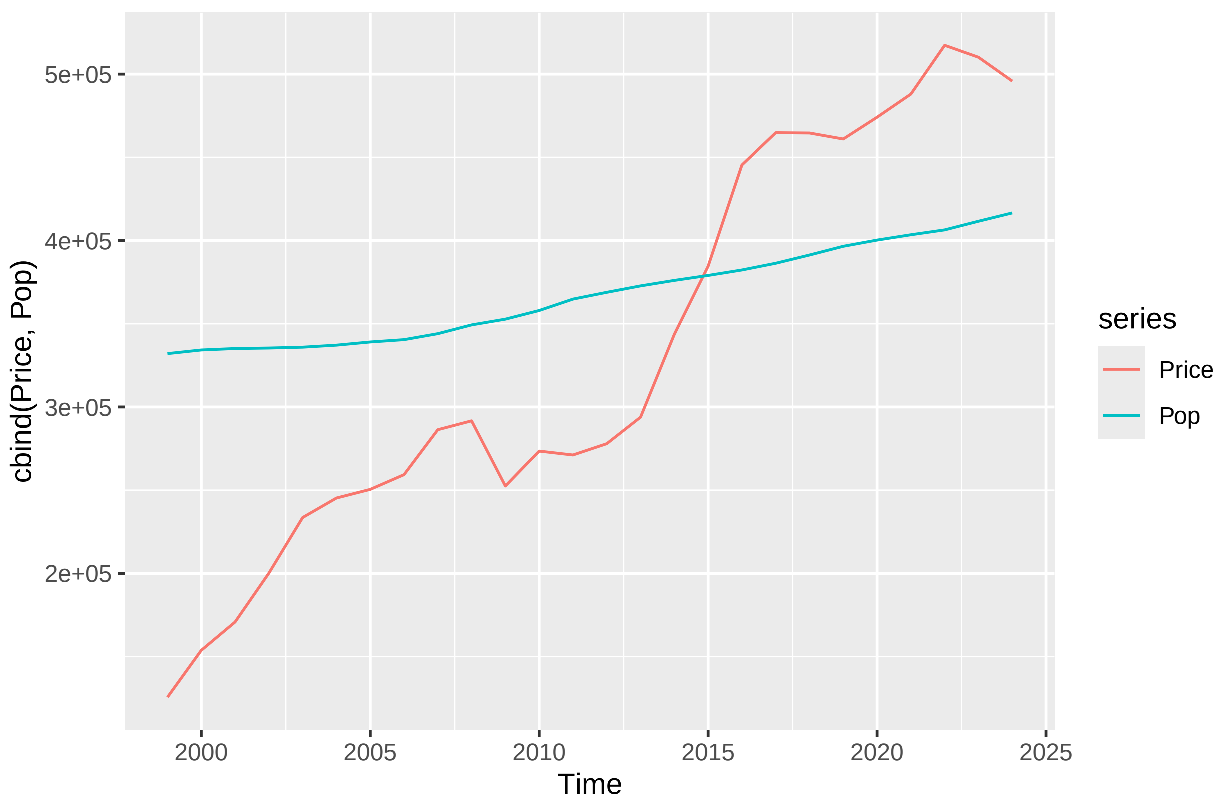
```
mae_ets <- mae(Price_ts_2023_2024, ets_forecasted_ts_legacy)
print(mae_ets)
```

```
## [1] 8231.699
```

```r
mae_stl <- mae(Price_ts_2023_2024, stl_forecasted_ts_legacy)
print(mae_stl)
```

```
## [1] 15075.61
```

```r
Croydon_df <- read.csv("Merged_Croydon_Data.csv")

Croydon_df$Date <- as.Date(Croydon_df$Date, format = "%Y")

Price <- ts(Croydon_df$Yearly_Price, start = c(1999), frequency = 1)

Pop <- ts(Croydon_df$Population, start = c(1999), frequency = 1)

autoplot(cbind(Price, Pop))
```




```r
Croydon_df.bv <- cbind(Price, Pop)
colnames(Croydon_df.bv) <- cbind("Price", "Population")

lagselect <- VARselect(Croydon_df.bv, lag.max = 12, type = "const")
```

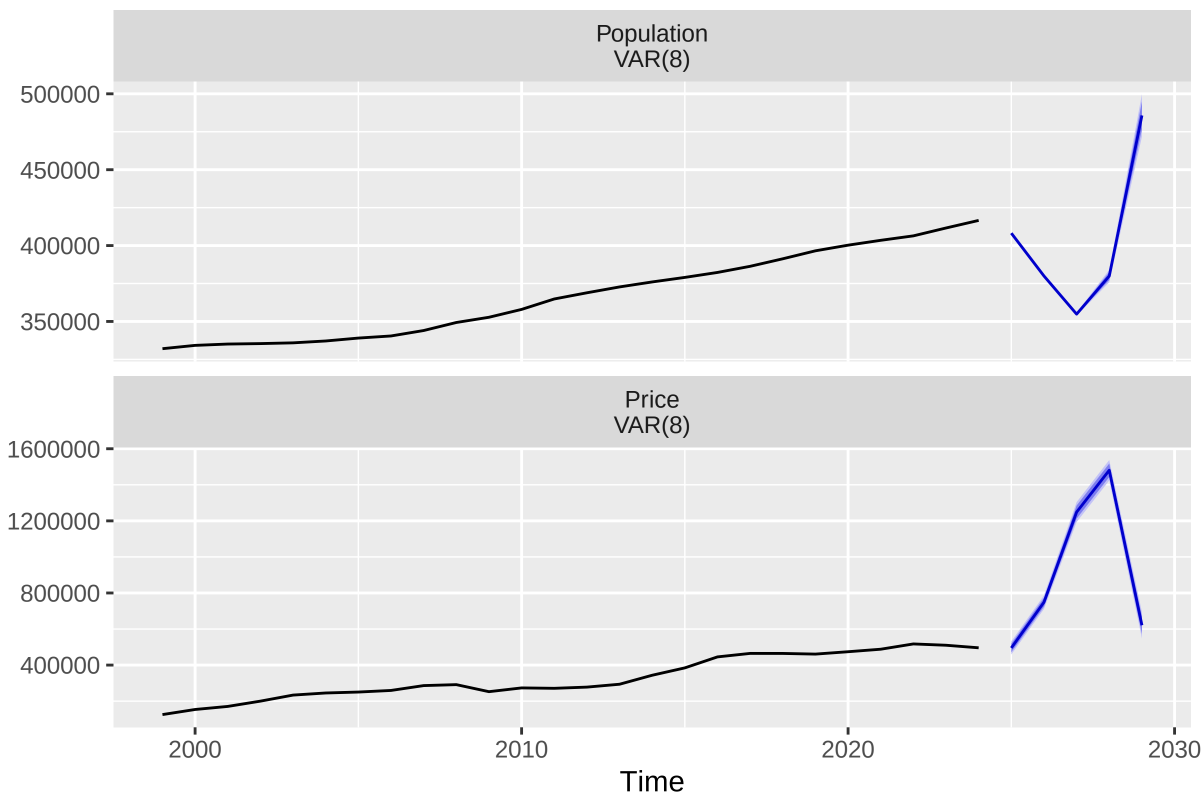
```
## Warning in log(sigma.det): NaNs produced
```

```
## Warning in log(sigma.det): NaNs produced
## Warning in log(sigma.det): NaNs produced
```

```
lagselect$selection
```

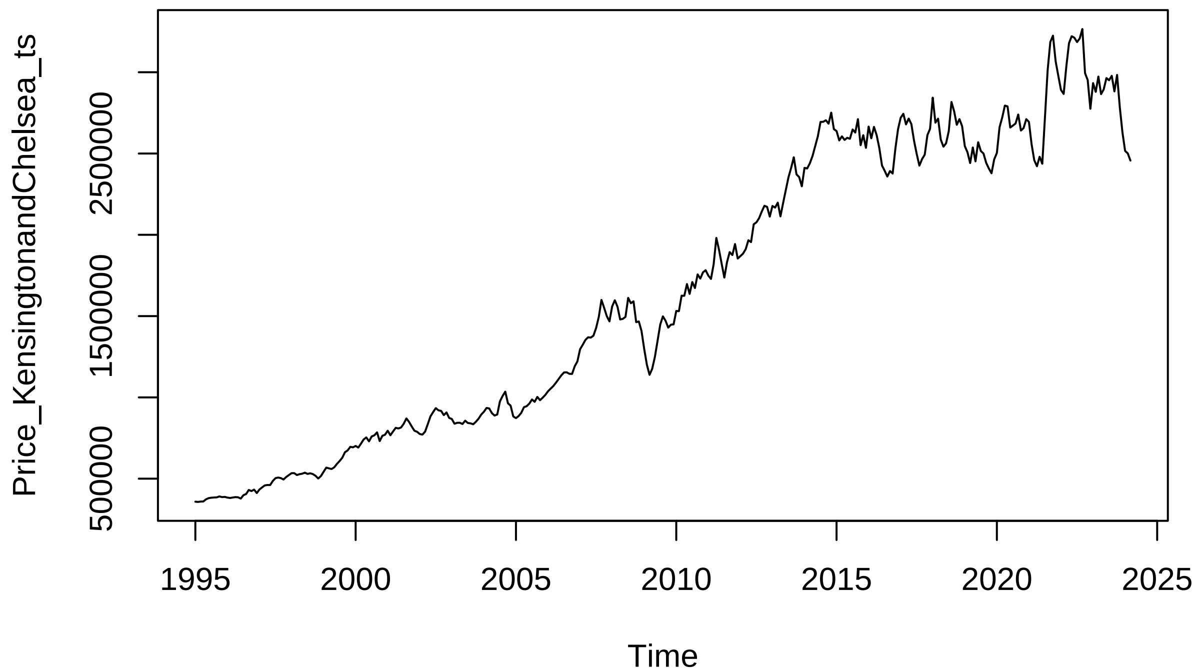
```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      7      7      7      6
```

```
var_model <- VAR(Croydon_df.bv, p = 8, type = "const")

forecast_values <- forecast(var_model, h = 5)
autoplot(forecast_values)
```

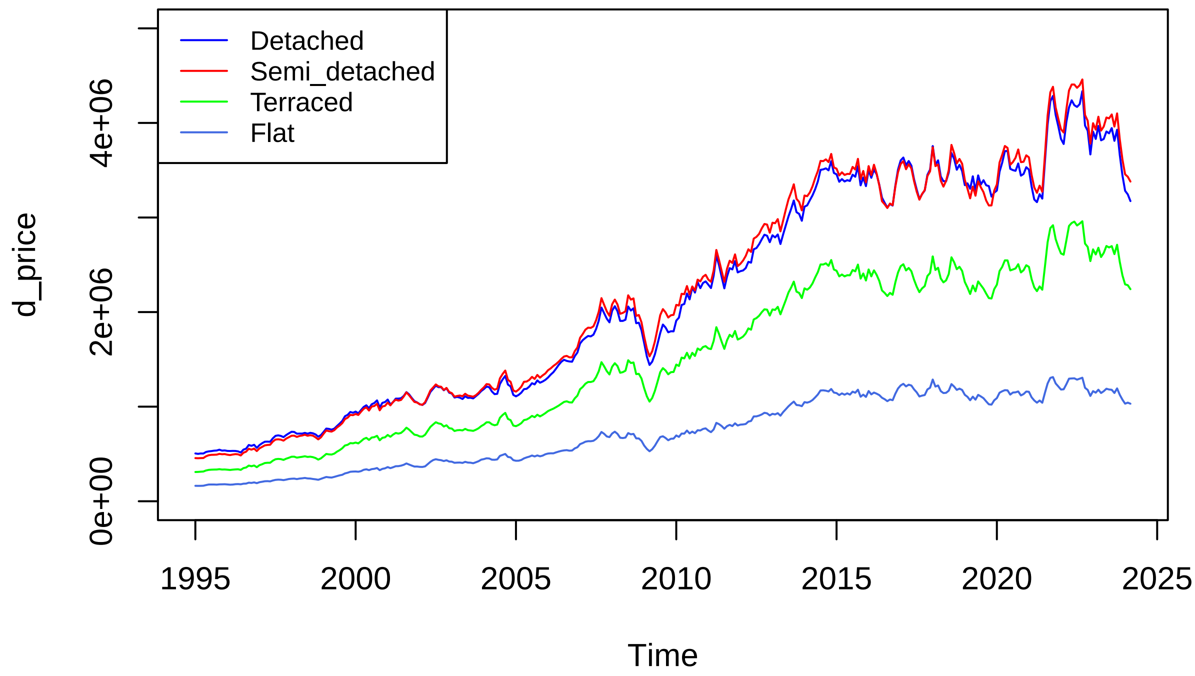



```
library(tseries)
library(forecast)
library(vars)
library(readr)
library(Metrics)

KensingtonandChelsea_df <- read.csv("Updated_KensingtonandChelsea.csv")
KensingtonandChelsea_df$Date <- as.Date(KensingtonandChelsea_df$Date, format = "%Y-%m-%d")
Price_KensingtonandChelsea_ts <- ts(KensingtonandChelsea_df$Average_Price, start = c(1995, 1), frequenc
d_price <- ts(KensingtonandChelsea_df$Detached_Average_Price, start = c(1995 , 1), frequency = 12)
sd_price <- ts(KensingtonandChelsea_df$Semi_Detached_Average_Price, start = c(1995, 1), frequency = 12)
```

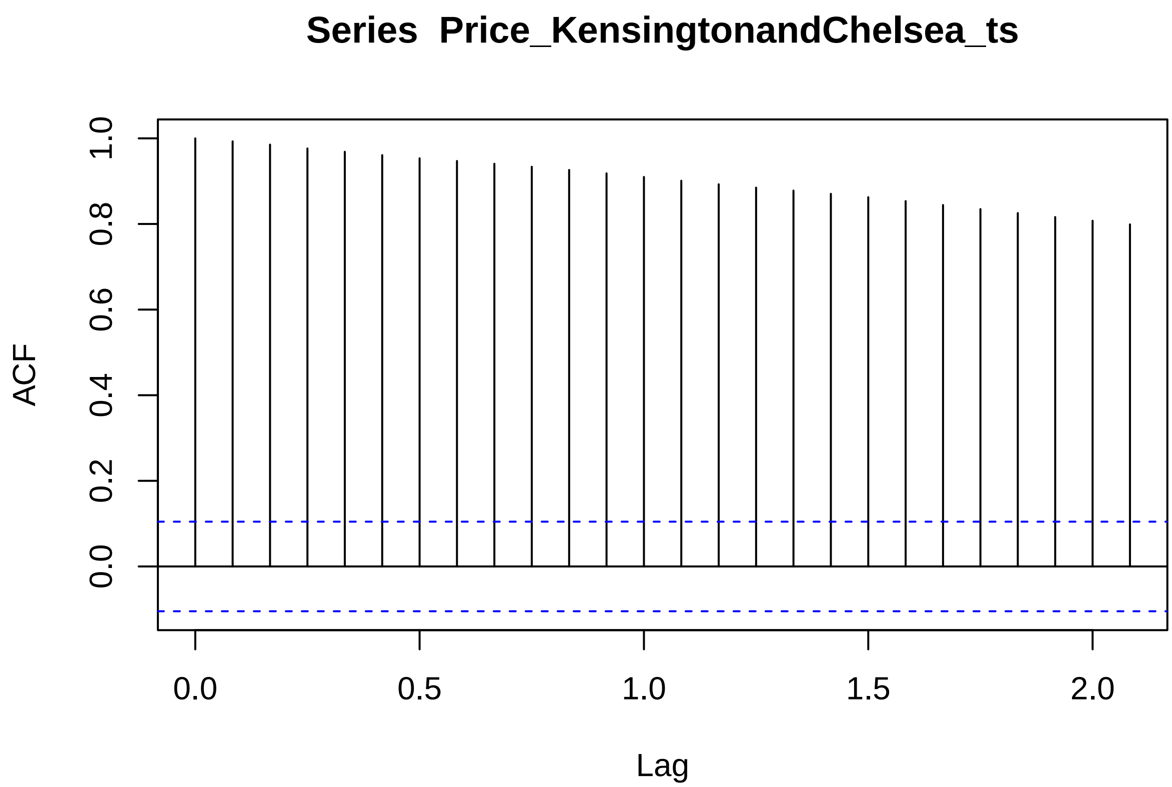
```r
t_price <- ts(KensingtonandChelsea_df$Terraced_Average_Price, start = c(1995, 1), frequency = 12)
f_price <- ts(KensingtonandChelsea_df$Flat_Average_Price, start = c(1995, 1), frequency = 12)
legacy_ts <- ts(KensingtonandChelsea_df$Average_Price, start = c(1995, 1), end = c(2023, 6), frequency =
Price_ts_2023_2024 <- window(Price_KensingtonandChelsea_ts, start = c(2023, 6), end = c(2024, 3))
diff_legacy_ts <- diff(legacy_ts, differences=1)
plot(Price_KensingtonandChelsea_ts)
```



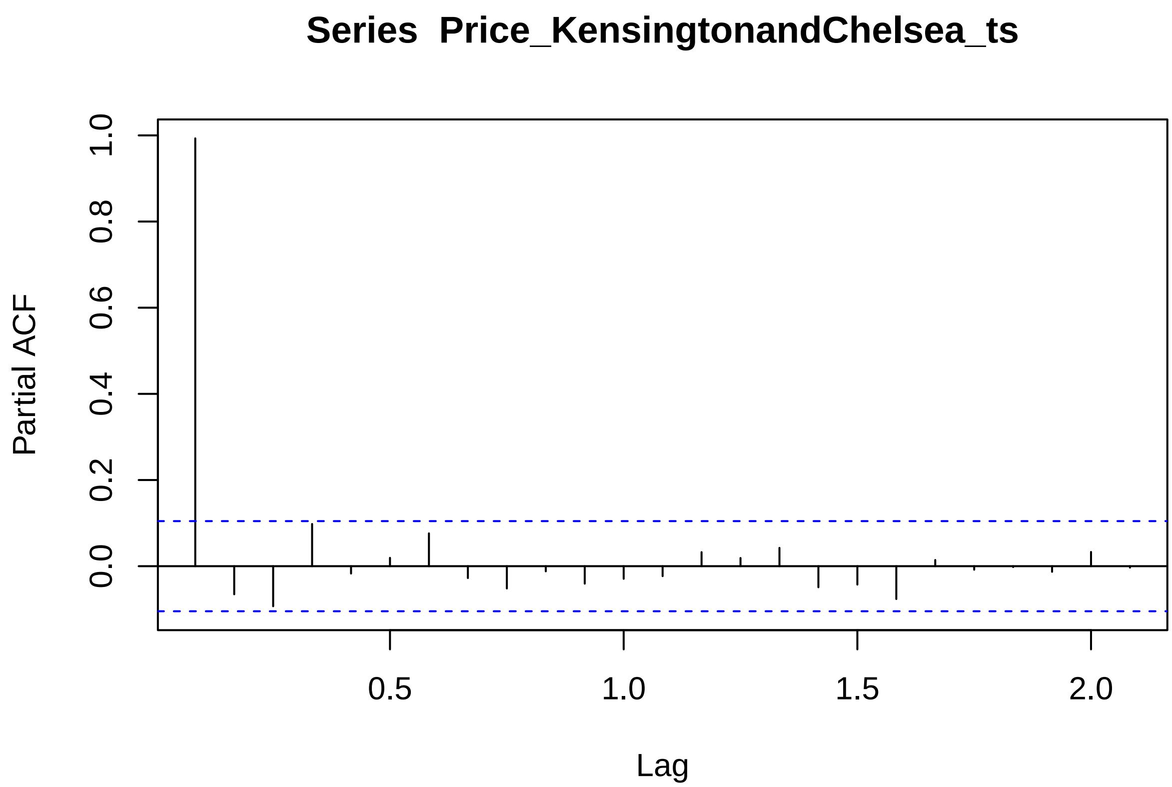

```r
plot(d_price, col = "blue", ylim = c(0, 5000000))
lines(sd_price, col = "red")
lines(t_price, col = "green")
lines(f_price, col = "royalblue")
legend("topleft",
       legend = c("Detached", "Semi_detached", "Terraced", "Flat"),
       col = c("blue", "red", "green", "royalblue"),
       lty = 1,
       cex = 0.8)
```




```r
acf(Price_KensingtonandChelsea_ts)
```




```
pacf(Price_KensingtonandChelsea_ts)
```




```r
adf.test(Price_KensingtonandChelsea_ts)
```

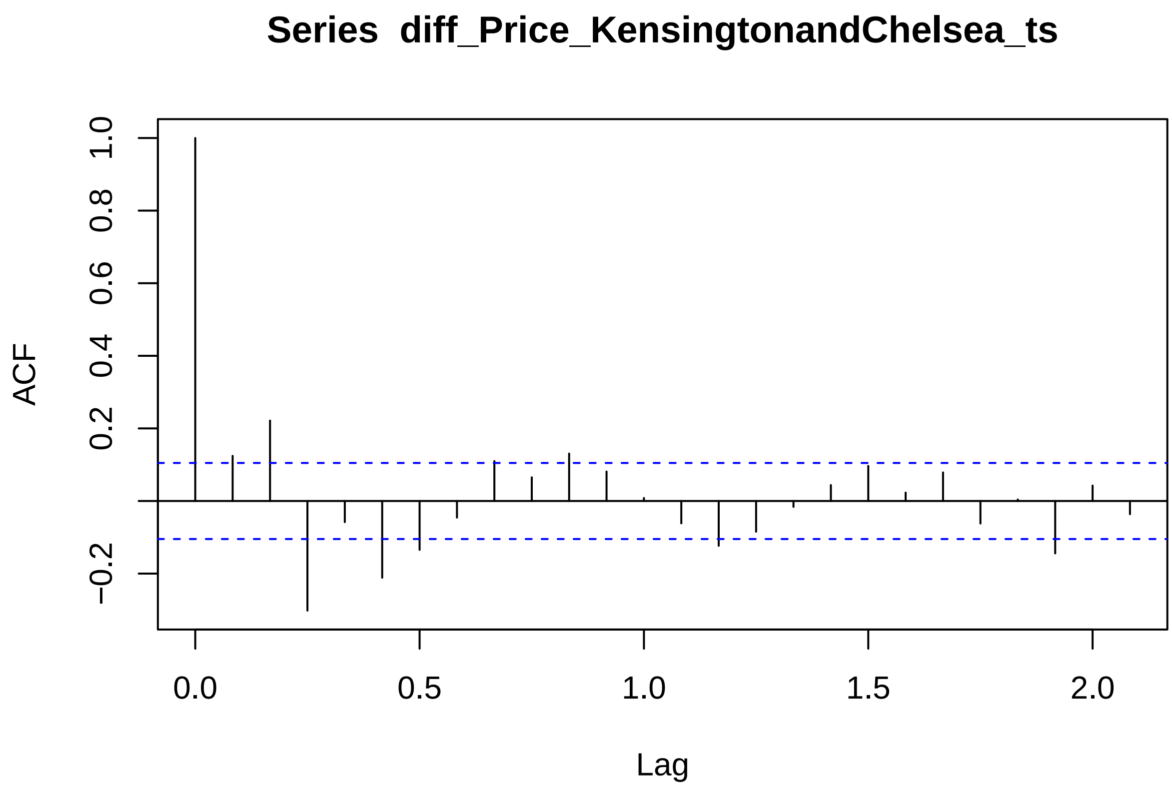
```
##
##  Augmented Dickey-Fuller Test
##
## data:  Price_KensingtonandChelsea_ts
## Dickey-Fuller = -1.4402, Lag order = 7, p-value = 0.813
## alternative hypothesis: stationary
```

```r
diff_Price_KensingtonandChelsea_ts <- diff(Price_KensingtonandChelsea_ts, differences=1)
adf.test(diff_Price_KensingtonandChelsea_ts)
```

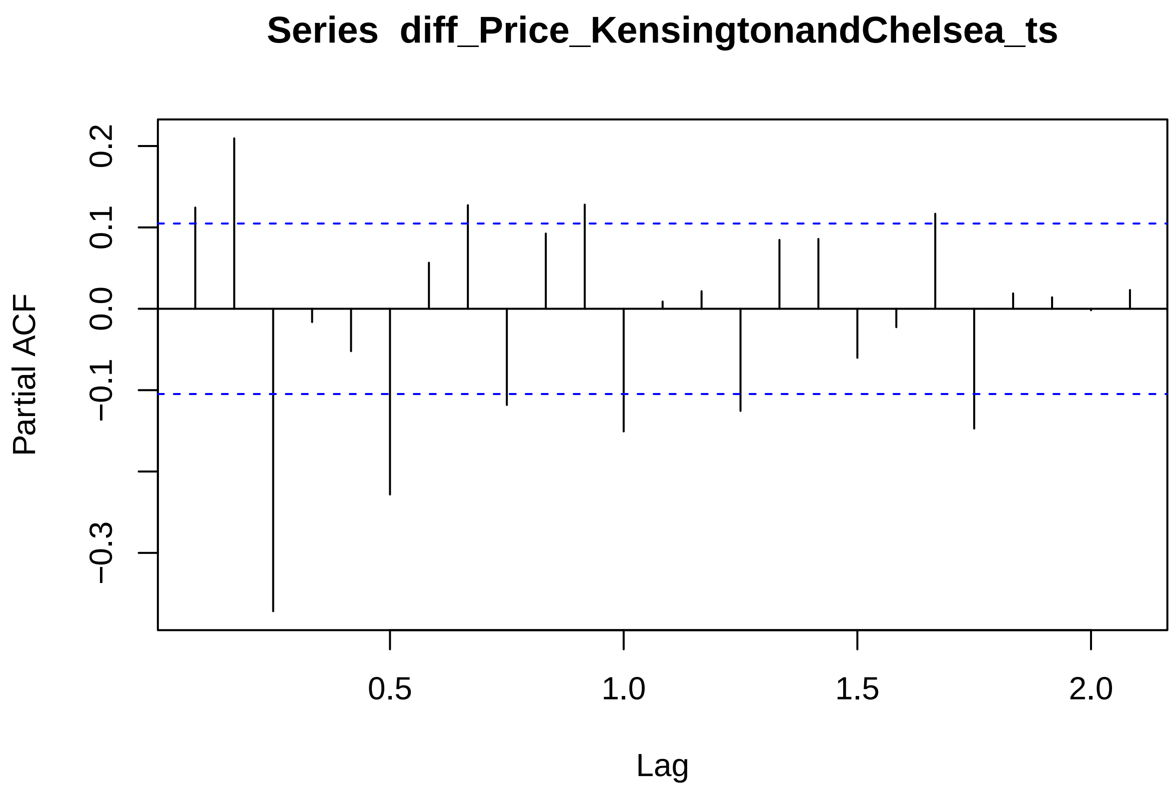
```
## Warning in adf.test(diff_Price_KensingtonandChelsea_ts): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_Price_KensingtonandChelsea_ts
## Dickey-Fuller = -6.6572, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

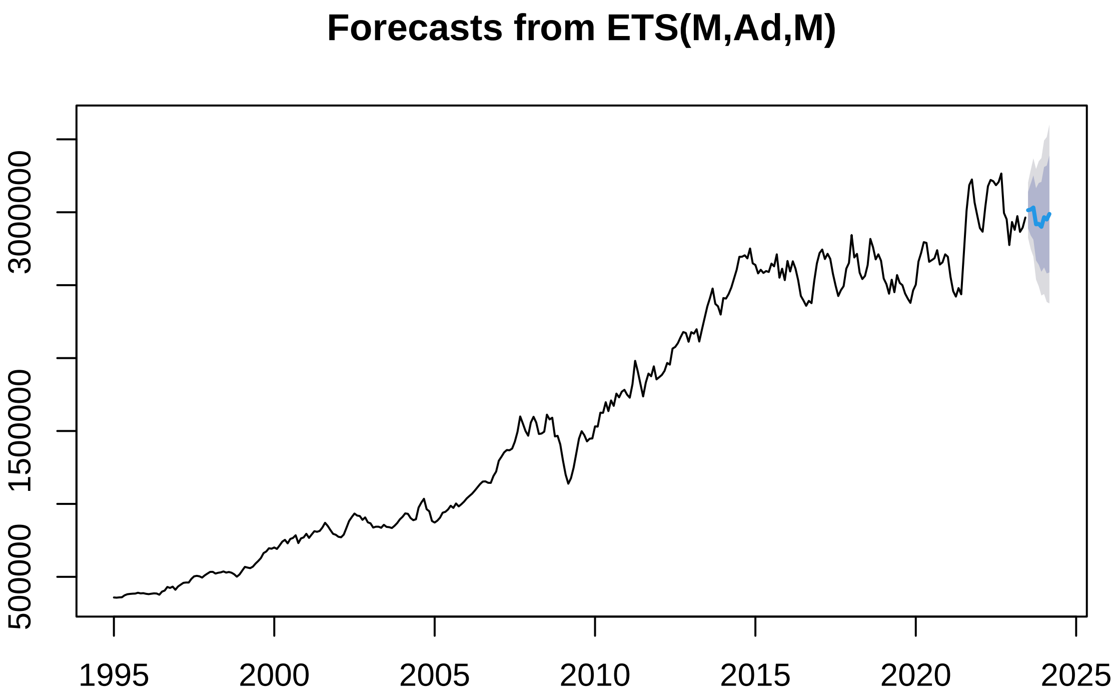
```r
acf(diff_Price_KensingtonandChelsea_ts)
```



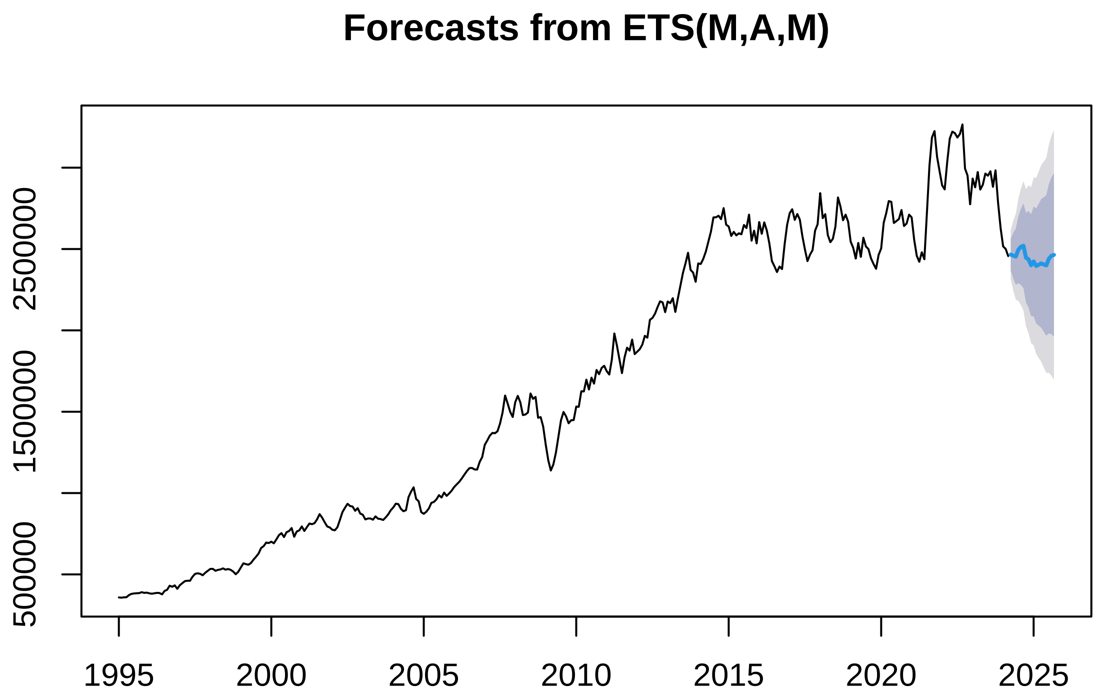

```r
pacf(diff_Price_KensingtonandChelsea_ts)
```




```r
ets_Price_KensingtonandChelsea_model <- ets(Price_KensingtonandChelsea_ts, model = "ZZZ")
ets_Price_KensingtonandChelsea_model_forecast <- forecast(ets_Price_KensingtonandChelsea_model, h = 18)
ets_legacy <- ets(legacy_ts, model = "MAM")
ets_legacy_forecast <- forecast(ets_legacy, h=9)
ets_forecasted_values_legacy <- ets_legacy_forecast$mean
ets_forecasted_ts_legacy <- ts(ets_forecasted_values_legacy, start = c(2023, 6), frequency = 12)
plot(ets_legacy_forecast)
```




```
plot(ets_Price_KensingtonandChelsea_model_forecast)
```

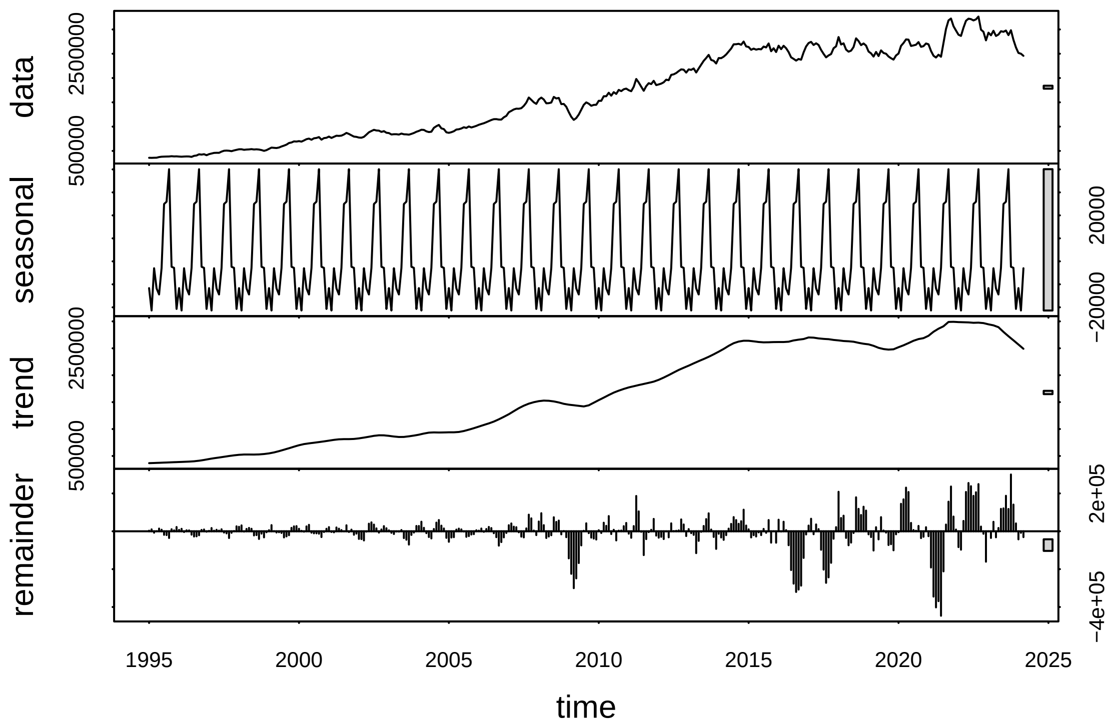



```
summary(ets_Price_KensingtonandChelsea_model_forecast)
```

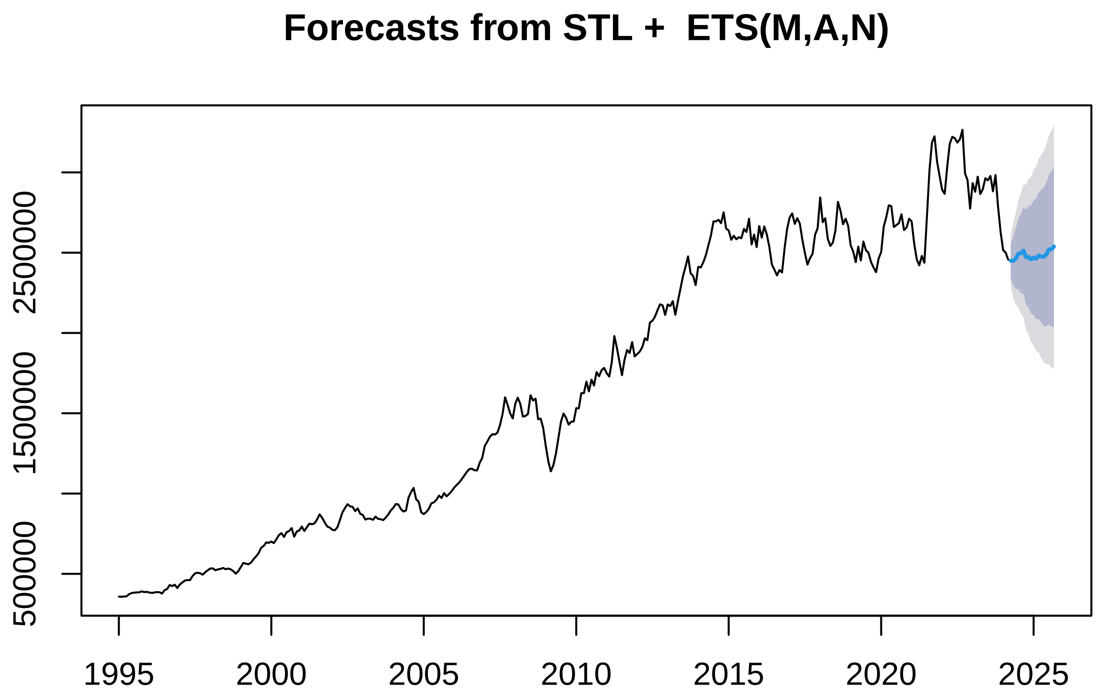
```
##
## Forecast method: ETS(M,A,M)
##
## Model Information:
## ETS(M,A,M)
##
## Call:
##  ets(y = Price_KensingtonandChelsea_ts, model = "ZZZ")
##
##   Smoothing parameters:
##     alpha = 0.9835
##     beta  = 0.0207
##     gamma = 2e-04
##
##   Initial states:
##     l = 357363.6086
##     b = 4923.0118
##     s = 0.9839 0.997 0.9991 1.027 1.0228 1.0134
##            0.9946 0.9949 0.9959 0.9906 0.9857 0.9952
##
##   sigma:  0.0315
##
##      AIC     AICc      BIC
```

```
## 9567.111 9568.948 9632.744
##
## Error measures:
##                     ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -1270.778 62269.17 42966.05 -0.05526985 2.447566 0.2654621
##                   ACF1
## Training set 0.1768669
##
## Forecasts:
##          Point Forecast   Lo 80   Hi 80   Lo 95   Hi 95
## Apr 2024        2465910 2366483 2565338 2313849 2617971
## May 2024        2458940 2318271 2599610 2243805 2674076
## Jun 2024        2453727 2280289 2627164 2188477 2718977
## Jul 2024        2495408 2289658 2701158 2180740 2810075
## Aug 2024        2513979 2279745 2748212 2155749 2872208
## Sep 2024        2519842 2259835 2779850 2122195 2917490
## Oct 2024        2446708 2171000 2722416 2025049 2868366
## Nov 2024        2437049 2140206 2733893 1983066 2891033
## Dec 2024        2400487 2086910 2714064 1920912 2880062
## Jan 2025        2423489 2086072 2760906 1907454 2939524
## Feb 2025        2396005 2042250 2749760 1854984 2937026
## Mar 2025        2403447 2028713 2778180 1830341 2976552
## Apr 2025        2411693 2015998 2807388 1806530 3016856
## May 2025        2404777 1990824 2818730 1771690 3037863
## Jun 2025        2399579 1967343 2831814 1738532 3060625
## Jul 2025        2440238 1981316 2899161 1738377 3142100
## Aug 2025        2458296 1976577 2940015 1721570 3195022
## Sep 2025        2463927 1961739 2966115 1695896 3231957
```

```r
stl_Price_KensingtonandChelsea_model <- stl(Price_KensingtonandChelsea_ts, s.window="periodic", robust=
plot(stl_Price_KensingtonandChelsea_model)
```




```
stl_Price_KensingtonandChelsea_model_forecast <- forecast(stl_Price_KensingtonandChelsea_model, method=
plot(stl_Price_KensingtonandChelsea_model_forecast)
```

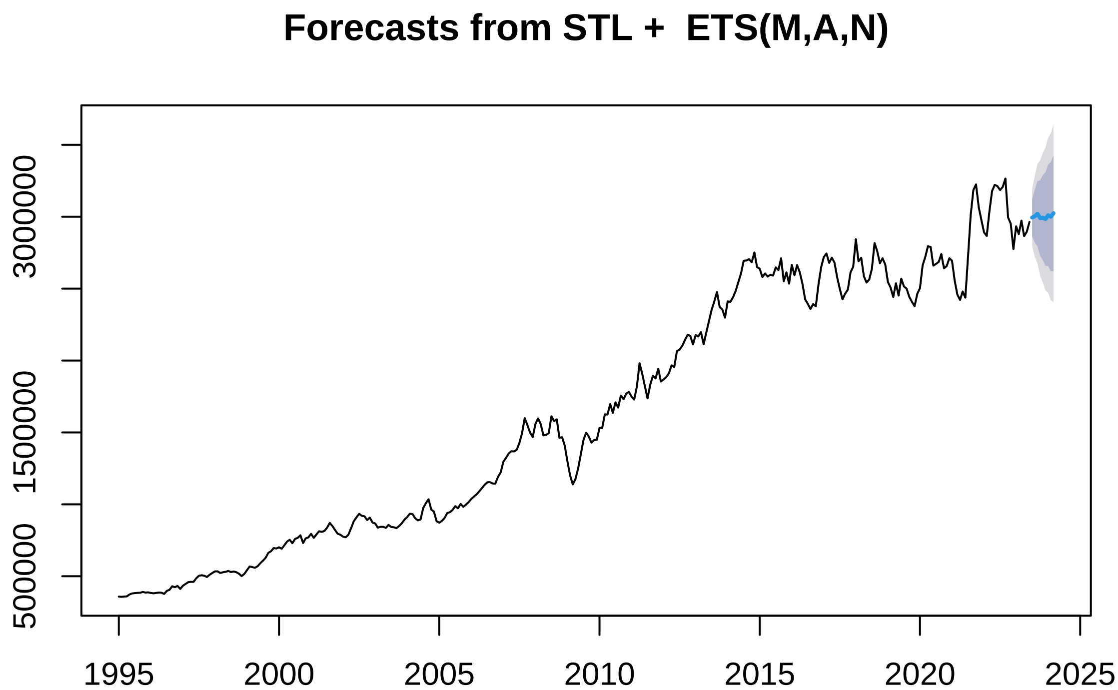



```r
summary(stl_Price_KensingtonandChelsea_model_forecast)
```

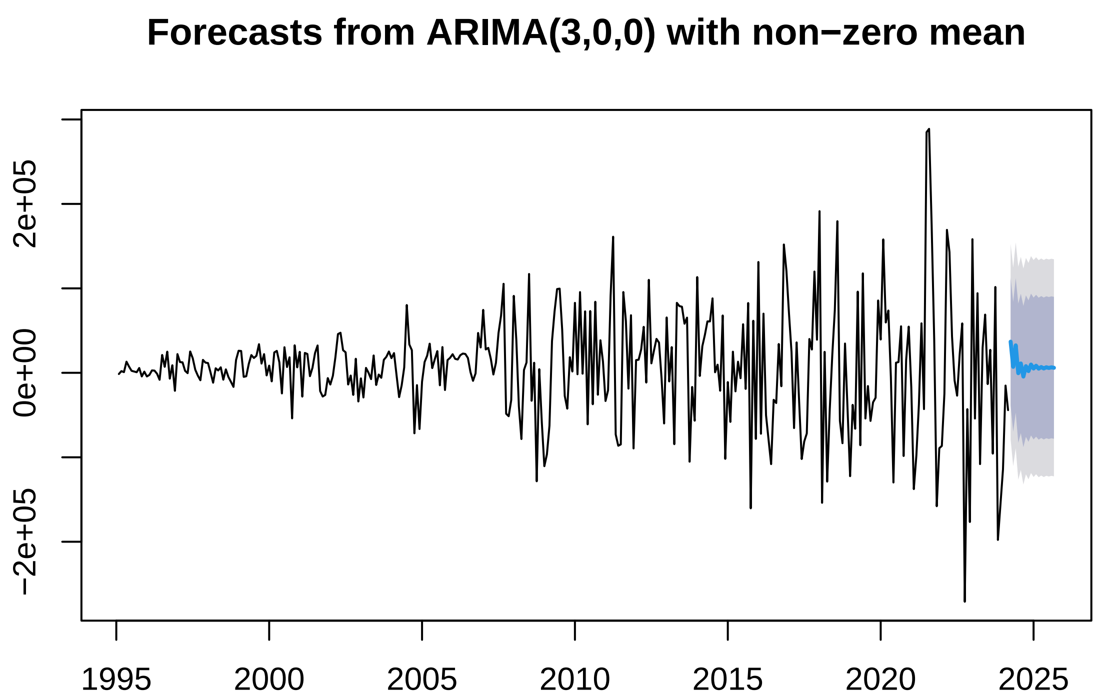
```
##
## Forecast method: STL +  ETS(M,A,N)
##
## Model Information:
## ETS(M,A,N)
##
## Call:
##  ets(y = na.interp(x), model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.0087
##
##   Initial states:
##     l = 373783.4794
##     b = 1795.9157
##
##   sigma:  0.0342
##
##      AIC     AICc      BIC
## 9612.608 9612.782 9631.912
##
## Error measures:
```

```
##                    ME     RMSE      MAE      MPE    MAPE     MASE      ACF1
## Training set 120.6386 63049.81 43931.16 0.105282 2.66542 0.271425 0.1503917
##
## Forecasts:
##          Point Forecast   Lo 80   Hi 80   Lo 95   Hi 95
## Apr 2024        2449907 2342078 2557735 2284998 2614815
## May 2024        2449415 2296155 2602675 2215024 2683806
## Jun 2024        2463101 2274447 2651754 2174580 2751622
## Jul 2024        2493103 2274164 2712042 2158264 2827941
## Aug 2024        2496276 2250260 2742291 2120028 2872523
## Sep 2024        2512585 2241733 2783436 2098353 2926816
## Oct 2024        2472288 2178268 2766309 2022623 2921954
## Nov 2024        2473987 2158093 2789881 1990869 2957105
## Dec 2024        2458271 2121543 2794999 1943289 2973252
## Jan 2025        2469516 2112806 2826225 1923976 3015056
## Feb 2025        2461925 2085947 2837902 1886917 3036932
## Mar 2025        2482481 2087842 2877121 1878932 3086030
## Apr 2025        2475873 2063091 2888654 1844577 3107168
## May 2025        2475381 2044907 2905854 1817029 3133733
## Jun 2025        2489067 2041296 2936837 1804261 3173873
## Jul 2025        2519069 2054348 2983789 1808340 3229798
## Aug 2025        2522242 2040878 3003605 1786060 3258423
## Sep 2025        2538551 2040819 3036282 1777336 3299766
```

```r
stl_legacy <- stl(legacy_ts, s.window="periodic", robust=TRUE)
stl_legacy_forecast <- forecast(stl_legacy, h=9)
plot(stl_legacy_forecast)
```




```
stl_forecasted_values_legacy <- stl_legacy_forecast$mean
stl_forecasted_ts_legacy <- ts(stl_forecasted_values_legacy, start = c(2023, 6), frequency = 12)

avg_Price_KensingtonandChelsea_model <- auto.arima(diff_Price_KensingtonandChelsea_ts, seasonal=FALSE)
avg_Price_KensingtonandChelsea_model_forecast <- forecast(avg_Price_KensingtonandChelsea_model, h=18)
plot(avg_Price_KensingtonandChelsea_model_forecast)
```




```r
summary(avg_Price_KensingtonandChelsea_model_forecast)
```

```
##
## Forecast method: ARIMA(3,0,0) with non-zero mean
##
## Model Information:
## Series: diff_Price_KensingtonandChelsea_ts
## ARIMA(3,0,0) with non-zero mean
##
## Coefficients:
##          ar1     ar2      ar3      mean
##       0.1751  0.2448  -0.3715  6145.866
## s.e.  0.0495  0.0486   0.0496  3304.052
##
## sigma^2 = 3.491e+09:  log likelihood = -4340.24
## AIC=8690.47   AICc=8690.65   BIC=8709.76
##
## Error measures:
##                     ME    RMSE     MAE       MPE    MAPE      MASE         ACF1
## Training set -14.79879 58744.3 38890.9 -31.24324 244.826 0.6247128 -0.006792641
##
## Forecasts:
##          Point Forecast     Lo 80     Hi 80      Lo 95    Hi 95
## Apr 2024     36876.1559 -38841.62 112593.93  -78924.18 152676.5
## May 2024      7091.4596 -69778.20  83961.12 -110470.54 124653.5
```

```
## Jun 2024     32539.2827 -47109.10 112187.67  -89272.41 154351.0
## Jul 2024      -417.6972 -82847.59  82012.20 -126483.35 125648.0
## Aug 2024     11105.6685 -71400.13  93611.47 -115076.06 137287.4
## Sep 2024     -4397.4277 -88010.78  79215.93 -132273.02 123478.2
## Oct 2024      7952.1907 -75790.19  91694.57 -120120.73 136025.1
## Nov 2024      2038.8895 -81711.98  85789.76 -126047.01 130124.8
## Dec 2024      9785.7329 -74177.49  93748.95 -118624.93 138196.4
## Jan 2025      5106.8637 -78861.99  89075.72 -123312.42 133526.2
## Feb 2025      8380.6166 -75606.13  92367.36 -120066.02 136827.3
## Mar 2025      4930.6263 -79083.68  88944.93 -123558.16 133419.4
## Apr 2025      6866.0760 -77149.22  90881.37 -121624.24 135356.4
## May 2025      5144.3049 -78878.18  89166.79 -123357.00 133645.6
## Jun 2025      6598.2474 -77427.21  90623.71 -121907.61 135104.1
## Jul 2025      5712.3640 -78313.62  89738.35 -122794.29 134219.0
## Aug 2025      6552.7722 -77474.88  90580.43 -121956.43 135062.0
## Sep 2025      5942.9438 -78085.06  89970.95 -122566.80 134452.7
```

```r
last_value <- tail(Price_KensingtonandChelsea_ts, n = 1)
forecasted_values <- c(last_value, avg_Price_KensingtonandChelsea_model_forecast$mean)
forecasted_values_L80 <- c(last_value, avg_Price_KensingtonandChelsea_model_forecast$lower[, "80%"])
print(avg_Price_Brent_model_forecast$lower[, "80%"])
```

```
##             Jan        Feb        Mar        Apr        May        Jun
## 2024                                  -42834.540 -11453.724  12564.036
## 2025 -11244.399 -11244.399 -11244.399 -11244.399 -11244.399 -11244.399
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2024  -9332.054 -11081.974 -11230.540 -11243.216 -11244.298 -11244.391
## 2025 -11244.399 -11244.399 -11244.399
```

```r
forecasted_values_L95 <- c(last_value, avg_Price_KensingtonandChelsea_model_forecast$lower[, "95%"])
print(avg_Price_Brent_model_forecast$lower[, "95%"])
```

```
##             Jan        Feb        Mar        Apr        May        Jun
## 2024                                  -48162.071 -17230.049   6480.065
## 2025 -18137.784 -18137.785 -18137.785 -18137.785 -18137.785 -18137.785
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2024 -16219.885 -17975.319 -18123.925 -18136.602 -18137.684 -18137.776
## 2025 -18137.785 -18137.785 -18137.785
```

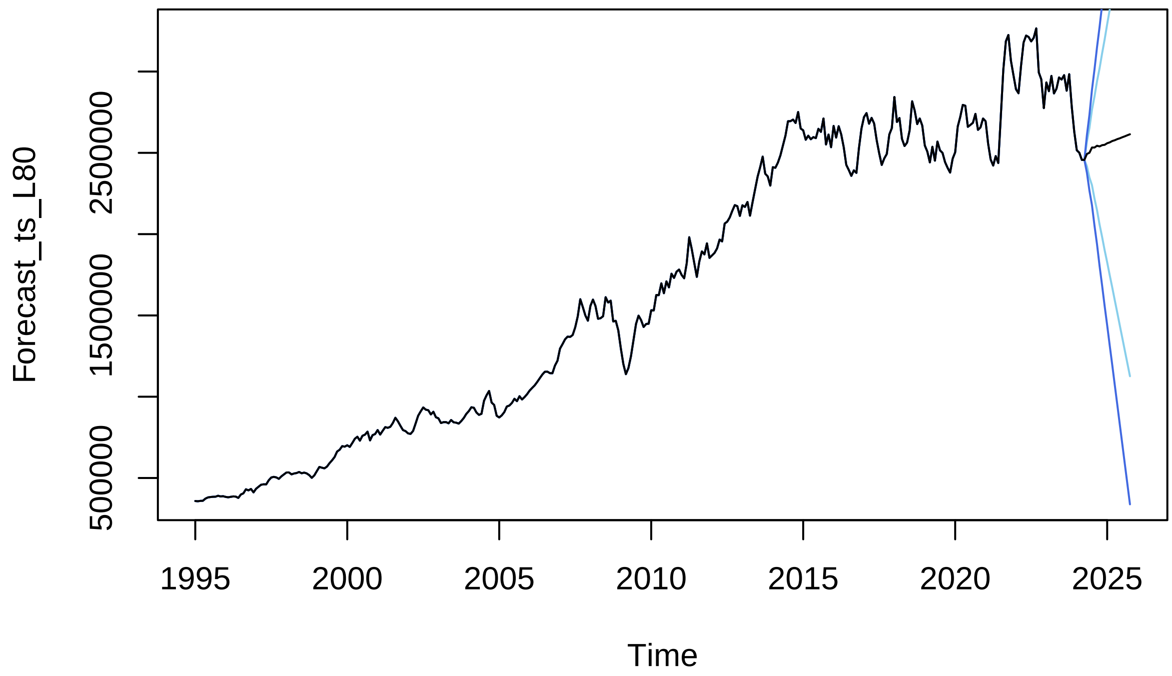
```r
forecasted_values_U80 <- c(last_value, avg_Price_KensingtonandChelsea_model_forecast$upper[, "80%"])
print(avg_Price_Brent_model_forecast$upper[, "80%"])
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2024                               -22706.65  10369.75  35549.83  16690.79
## 2025  14799.43  14799.43  14799.43  14799.43  14799.43  14799.43  14799.43
##            Aug       Sep       Oct       Nov       Dec
## 2024  14961.70  14813.29  14800.61  14799.53  14799.44
## 2025  14799.43  14799.43
```

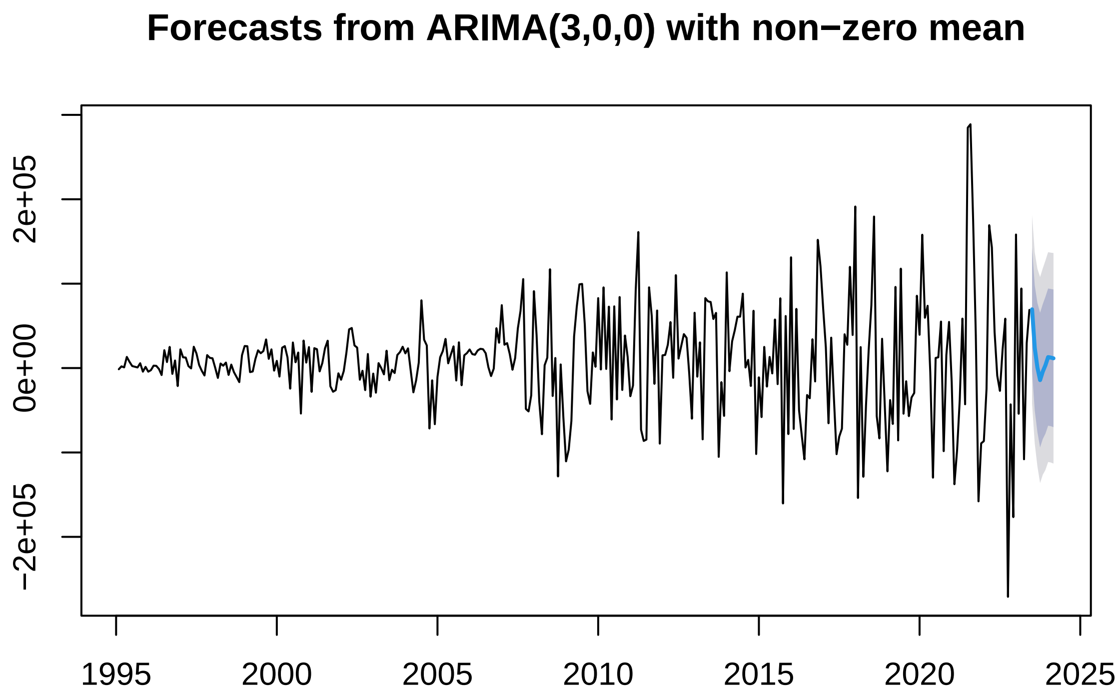
```r
forecasted_values_U95 <- c(last_value, avg_Price_KensingtonandChelsea_model_forecast$upper[, "95%"])
print(avg_Price_Brent_model_forecast$upper[, "95%"])
```

```
##           Jan       Feb       Mar       Apr       May       Jun       Jul
## 2024                               -17379.11  16146.08  41633.80  23578.62
## 2025  21692.82  21692.82  21692.82  21692.82  21692.82  21692.82  21692.82
##           Aug       Sep       Oct       Nov       Dec
## 2024  21855.05  21706.67  21694.00  21692.92  21692.82
## 2025  21692.82  21692.82
```

```r
cumulative_forecasted_values_L80 <- cumsum(forecasted_values_L80)
cumulative_forecasted_values_L95 <- cumsum(forecasted_values_L95)
cumulative_forecasted_values_U80 <- cumsum(forecasted_values_U80)
cumulative_forecasted_values_U95 <- cumsum(forecasted_values_U95)
cumulative_forecasted_values <- cumsum(forecasted_values)
cumulative_forecasted_ts <- ts(cumulative_forecasted_values, start = c(2024, 2), frequency = 12)
cumulative_forecasted_ts_U80 <- ts(cumulative_forecasted_values_U80, start = c(2024, 2), frequency = 12
cumulative_forecasted_ts_U95 <- ts(cumulative_forecasted_values_U95, start = c(2024, 2), frequency = 12
cumulative_forecasted_ts_L80 <- ts(cumulative_forecasted_values_L80, start = c(2024, 2), frequency = 12
cumulative_forecasted_ts_L95 <- ts(cumulative_forecasted_values_L95, start = c(2024, 2), frequency = 12
combined <- c(Price_KensingtonandChelsea_ts, cumulative_forecasted_ts)
combined_L80 <- c(Price_KensingtonandChelsea_ts, cumulative_forecasted_ts_L80)
combined_L95 <- c(Price_KensingtonandChelsea_ts, cumulative_forecasted_ts_L95)
combined_U80 <- c(Price_KensingtonandChelsea_ts, cumulative_forecasted_ts_U80)
combined_U95 <- c(Price_KensingtonandChelsea_ts, cumulative_forecasted_ts_U95)
Forecast_ts <- ts(combined, start = c(1995, 1), frequency = 12)
Forecast_ts_L80 <- ts(combined_L80, start = c(1995, 1), frequency = 12)
Forecast_ts_L95 <- ts(combined_L95, start = c(1995, 1), frequency = 12)
Forecast_ts_U80 <- ts(combined_U80, start = c(1995, 1), frequency = 12)
Forecast_ts_U95 <- ts(combined_U95, start = c(1995, 1), frequency = 12)
plot(Forecast_ts_L80, col = "skyblue")
lines(Forecast_ts_U80, col = "skyblue")
lines(Forecast_ts_L95, col = "royalblue")
lines(Forecast_ts_U95, col = "royalblue")
lines(Forecast_ts)
```



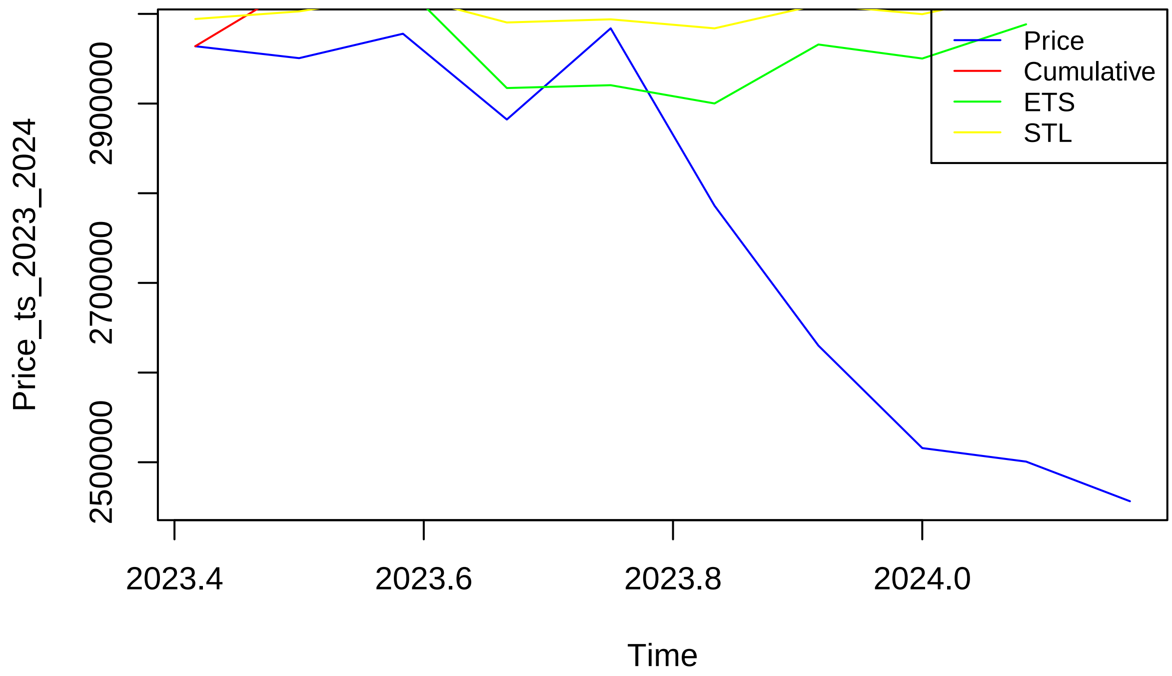

```r
avg_legacy <- arima(diff_legacy_ts, order = c(3, 0, 0))
avg_legacy_forecast <- forecast(avg_legacy, h=9)
plot(avg_legacy_forecast)
```




```r
last_value_legacy <- tail(legacy_ts, n = 1)
forecasted_values_legacy <- c(last_value_legacy, avg_legacy_forecast$mean)
cumulative_forecasted_values_legacy <- cumsum(forecasted_values_legacy)
print(cumulative_forecasted_values_legacy)
```

```
##  [1] 2963934 3033531 3057584 3058269 3044023 3039845 3043239 3056230 3068652
## [10] 3080196
```

```r
cumulative_forecasted_ts_legacy <- ts(cumulative_forecasted_values_legacy, start = c(2023, 6), frequency

plot(Price_ts_2023_2024, type = "l", col = "blue")
lines(cumulative_forecasted_ts_legacy, col = "red")
lines(ets_forecasted_ts_legacy, col = "green")
lines(stl_forecasted_ts_legacy, col = "yellow")
legend("topright",
       legend = c("Price", "Cumulative", "ETS", "STL"),
       col = c("blue", "red", "green", "yellow"),
       lty = 1,
       cex = 0.8)
```




```
mse_avg <- mean((Price_ts_2023_2024 - cumulative_forecasted_ts_legacy)^2)
print(mse_avg)
```

```
## [1] 128652451532
```

```
mse_ets <- mean((Price_ts_2023_2024 - ets_forecasted_ts_legacy)^2)
print(mse_ets)
```

```
## [1] 63074514469
```

```
mse_stl <- mean((Price_ts_2023_2024 - stl_forecasted_ts_legacy)^2)
print(mse_stl)
```

```
## [1] 78850774087
```

```
mae_avg <- mae(Price_ts_2023_2024, cumulative_forecasted_ts_legacy)
print(mae_avg)
```

```
## [1] 279779.1
```

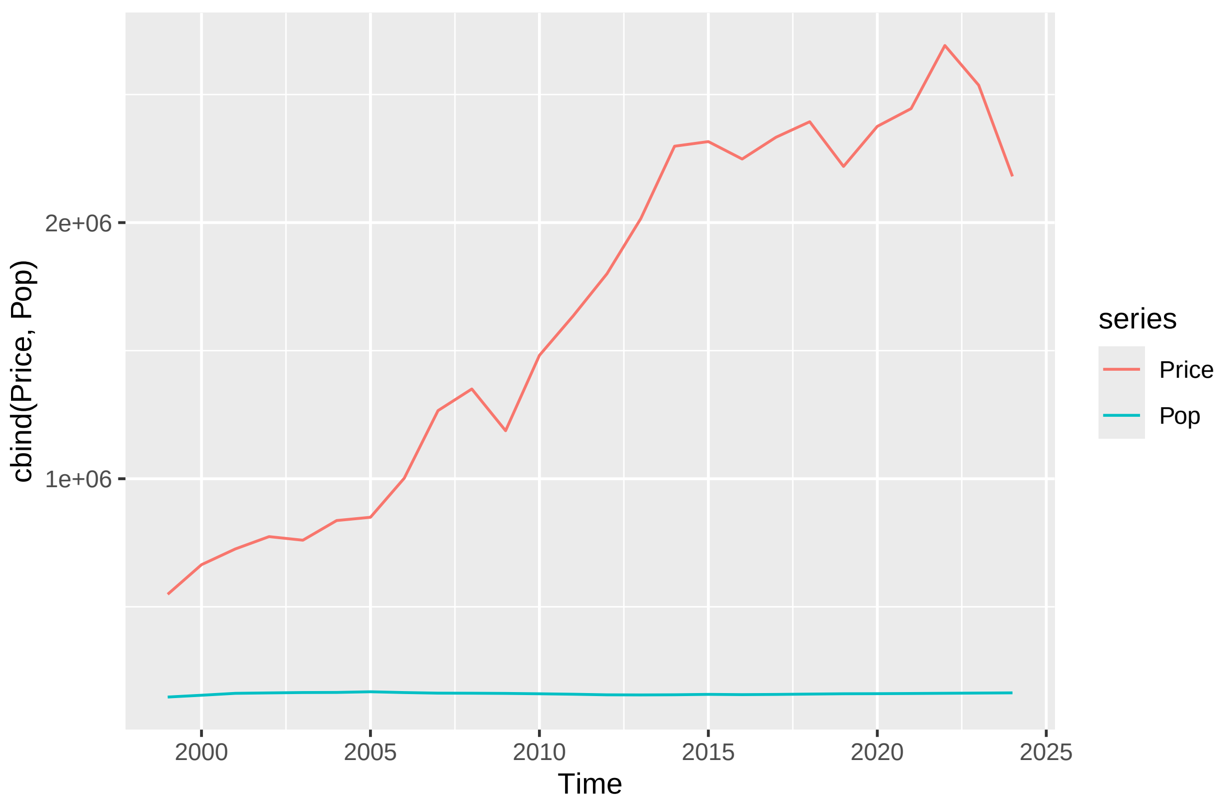
```
mae_ets <- mae(Price_ts_2023_2024, ets_forecasted_ts_legacy)
print(mae_ets)
```

```
## [1] 182562.7
```

```r
mae_stl <- mae(Price_ts_2023_2024, stl_forecasted_ts_legacy)
print(mae_stl)
```

```
## [1] 203302.3
```

```r
KensingtonandChelsea_df <- read.csv("Merged_KensingtonandChelsea_Data.csv")

KensingtonandChelsea_df$Date <- as.Date(KensingtonandChelsea_df$Date, format = "%Y")

Price <- ts(KensingtonandChelsea_df$Yearly_Price, start = c(1999), frequency = 1)

Pop <- ts(KensingtonandChelsea_df$Population, start = c(1999), frequency = 1)

autoplot(cbind(Price, Pop))
```




```r
KensingtonandChelsea_df.bv <- cbind(Price, Pop)
colnames(KensingtonandChelsea_df.bv) <- cbind("Price", "Population")

lagselect <- VARselect(KensingtonandChelsea_df.bv, lag.max = 10, type = "const")
```

```
## Warning in log(sigma.det): NaNs produced
```

```
## Warning in log(sigma.det): NaNs produced
## Warning in log(sigma.det): NaNs produced
```

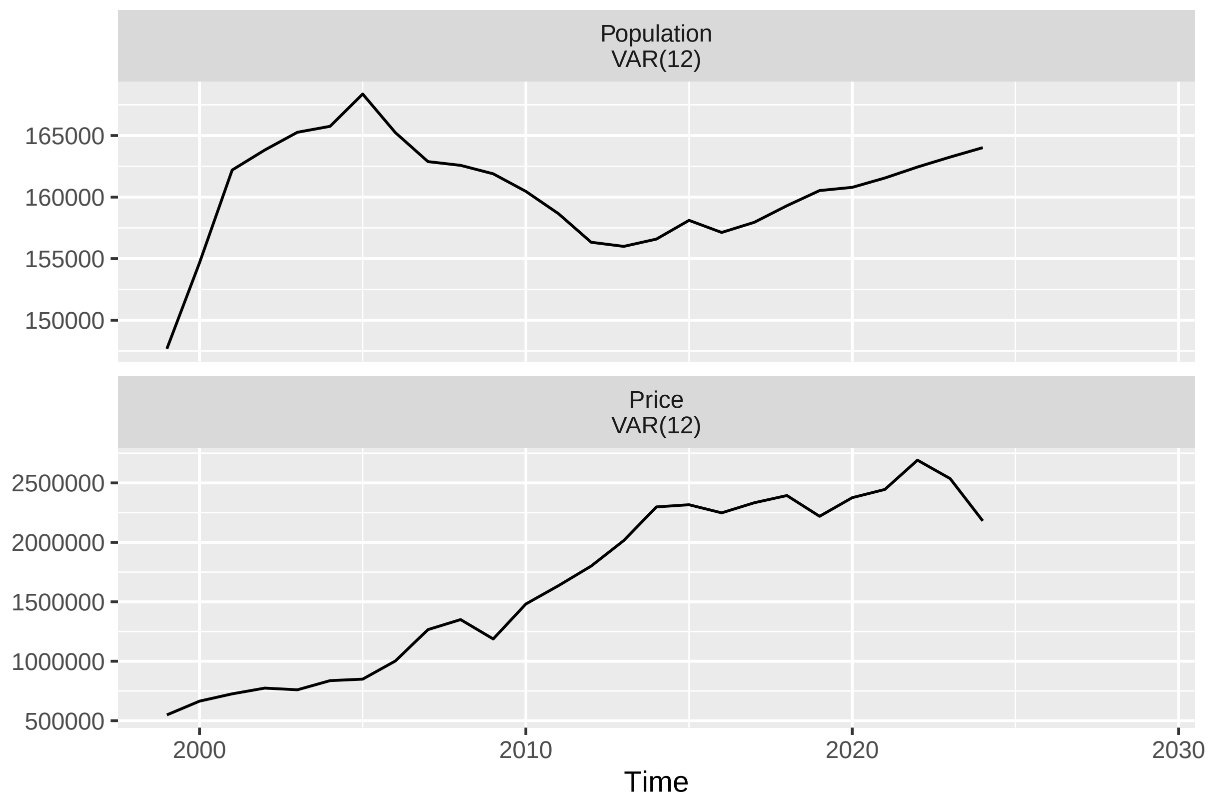
```r
lagselect$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      8      8      8      7
```

```r
var_model <- VAR(KensingtonandChelsea_df.bv, p = 12, type = "const")

forecast_values <- forecast(var_model, h = 5)
autoplot(forecast_values)
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```



```r
library(tseries)
library(forecast)
library(vars)
library(readr)
library(Metrics)
df<-read.csv("Updated_Brent.csv")
d<-df$Detached_Average_Price
sd<-df$Semi_Detached_Average_Price
t<-df$Terraced_Average_Price
f<-df$Flat_Average_Price
price_data<-data.frame(d,sd,t, f)
cor_matrix<-cor(price_data)
print(cor_matrix)
```

```
##            d        sd         t         f
## d  1.0000000 0.9997165 0.9990409 0.9917651
## sd 0.9997165 1.0000000 0.9995807 0.9936648
## t  0.9990409 0.9995807 1.0000000 0.9952224
## f  0.9917651 0.9936648 0.9952224 1.0000000
```

```r
df$Date<-as.Date(df$Date,format="%Y-%m-%d")
d_price<-diff(ts(df$Detached_Average_Price,start=c(1995,1),frequency=12), d = 1)
sd_price<-diff(ts(df$Semi_Detached_Average_Price,start=c(1995,1),frequency=12), d= 1)
t_price<-diff(ts(df$Terraced_Average_Price,start=c(1995,1),frequency=12), d = 1)
autoplot(cbind(d_price,sd_price))
```

```r
df.bv<-cbind(d_price,sd_price)
df.bv2<-cbind(sd_price,t_price)
colnames(df.bv)<-cbind("d_price","sd_price")
colnames(df.bv2)<-cbind("sd_price","t_price")
lagselect<-VARselect(df.bv,lag.max=12,type="const")
lagselect$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     11     10      6     11
```

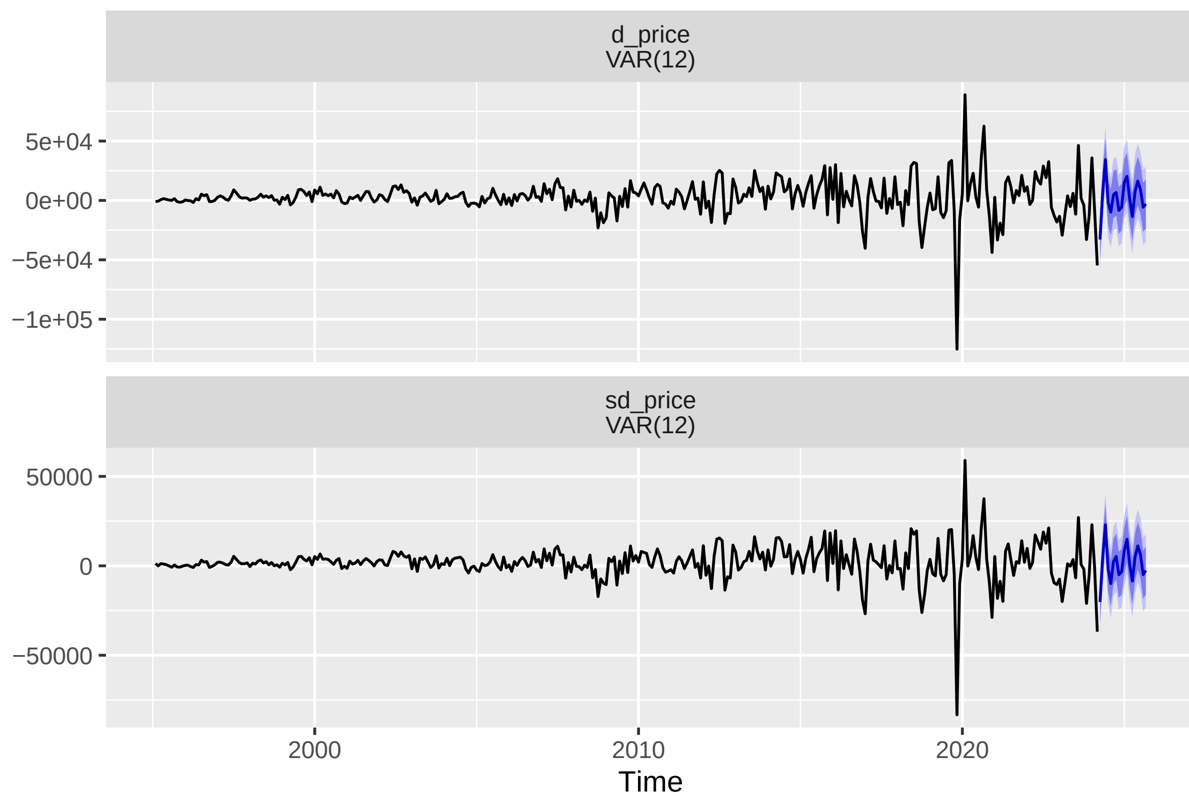
```r
var_model<-VAR(df.bv,p=12,type ="const")
summary(var_model)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: d_price, sd_price
## Deterministic variables: const
## Sample size: 338
## Log Likelihood: -6463.858
## Roots of the characteristic polynomial:
## 0.9435 0.9435 0.9268 0.9221 0.9221 0.9107 0.9107 0.9093 0.9093 0.8882 0.8882 0.8828 0.8828 0.8774 0.8
## Call:
## VAR(y = df.bv, p = 12, type = "const")
##
##
## Estimation results for equation d_price:
## ========================================
## d_price = d_price.l1 + sd_price.l1 + d_price.l2 + sd_price.l2 + d_price.l3 + sd_price.l3 + d_price.l4
##
##               Estimate Std. Error t value Pr(>|t|)
## d_price.l1      1.9408     0.4278   4.536 8.16e-06 ***
## sd_price.l1    -2.4112     0.6565  -3.673 0.000282 ***
## d_price.l2     -1.8850     0.4565  -4.129 4.68e-05 ***
## sd_price.l2     3.1330     0.6934   4.518 8.84e-06 ***
## d_price.l3     -1.6634     0.4859  -3.424 0.000700 ***
## sd_price.l3     1.5364     0.7390   2.079 0.038422 *
## d_price.l4      2.6464     0.5487   4.823 2.21e-06 ***
## sd_price.l4    -3.4693     0.8228  -4.216 3.26e-05 ***
## d_price.l5     -1.0106     0.5656  -1.787 0.074964 .
## sd_price.l5     1.9760     0.8383   2.357 0.019039 *
## d_price.l6     -1.3790     0.5741  -2.402 0.016886 *
## sd_price.l6     1.3942     0.8579   1.625 0.105139
## d_price.l7      2.1455     0.5750   3.731 0.000226 ***
## sd_price.l7    -3.1506     0.8532  -3.693 0.000262 ***
## d_price.l8     -0.5178     0.5871  -0.882 0.378512
## sd_price.l8     1.1875     0.8651   1.373 0.170818
## d_price.l9     -0.3773     0.5679  -0.664 0.506870
## sd_price.l9     0.1737     0.8427   0.206 0.836791
## d_price.l10     1.3991     0.4928   2.839 0.004821 **
## sd_price.l10   -2.3499     0.7371  -3.188 0.001576 **
## d_price.l11    -0.5627     0.4922  -1.143 0.253745
```

```
## sd_price.l11    1.0979     0.7364   1.491 0.136994
## d_price.l12    -0.2678     0.4615  -0.580 0.562220
## sd_price.l12    0.3311     0.6987   0.474 0.635896
## const       2563.3490   827.5569   3.097 0.002129 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 12190 on 313 degrees of freedom
## Multiple R-Squared: 0.4129,  Adjusted R-squared: 0.3679
## F-statistic: 9.173 on 24 and 313 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation sd_price:
## =========================================
## sd_price = d_price.l1 + sd_price.l1 + d_price.l2 + sd_price.l2 + d_price.l3 + sd_price.l3 + d_price.l
##
##               Estimate Std. Error t value Pr(>|t|)
## d_price.l1      1.0913     0.2785   3.919 0.000109 ***
## sd_price.l1    -1.3026     0.4273  -3.048 0.002498 **
## d_price.l2     -1.4201     0.2972  -4.779 2.71e-06 ***
## sd_price.l2     2.3153     0.4513   5.130 5.09e-07 ***
## d_price.l3     -0.6673     0.3162  -2.110 0.035647 *
## sd_price.l3     0.3703     0.4810   0.770 0.441975
## d_price.l4      1.5939     0.3571   4.463 1.13e-05 ***
## sd_price.l4    -2.0444     0.5356  -3.817 0.000163 ***
## d_price.l5     -0.8082     0.3682  -2.195 0.028876 *
## sd_price.l5     1.5196     0.5457   2.785 0.005679 **
## d_price.l6     -0.5837     0.3737  -1.562 0.119290
## sd_price.l6     0.4286     0.5584   0.768 0.443290
## d_price.l7      1.3483     0.3742   3.603 0.000366 ***
## sd_price.l7    -1.9941     0.5553  -3.591 0.000382 ***
## d_price.l8     -0.4317     0.3821  -1.130 0.259418
## sd_price.l8     0.9345     0.5631   1.660 0.097990 .
## d_price.l9     -0.0401     0.3696  -0.108 0.913683
## sd_price.l9    -0.1968     0.5485  -0.359 0.719988
## d_price.l10     0.8403     0.3208   2.620 0.009228 **
## sd_price.l10   -1.4281     0.4797  -2.977 0.003140 **
## d_price.l11    -0.4037     0.3203  -1.260 0.208497
## sd_price.l11    0.7818     0.4793   1.631 0.103889
## d_price.l12    -0.2035     0.3004  -0.677 0.498628
## sd_price.l12    0.2557     0.4548   0.562 0.574279
## const       1679.0340   538.6393   3.117 0.001995 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 7933 on 313 degrees of freedom
## Multiple R-Squared: 0.4134,  Adjusted R-squared: 0.3684
## F-statistic:  9.19 on 24 and 313 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
```

```
##            d_price sd_price
## d_price  148552062 95842074
## sd_price  95842074 62933089
##
## Correlation matrix of residuals:
##          d_price sd_price
## d_price   1.0000   0.9912
## sd_price  0.9912   1.0000
```

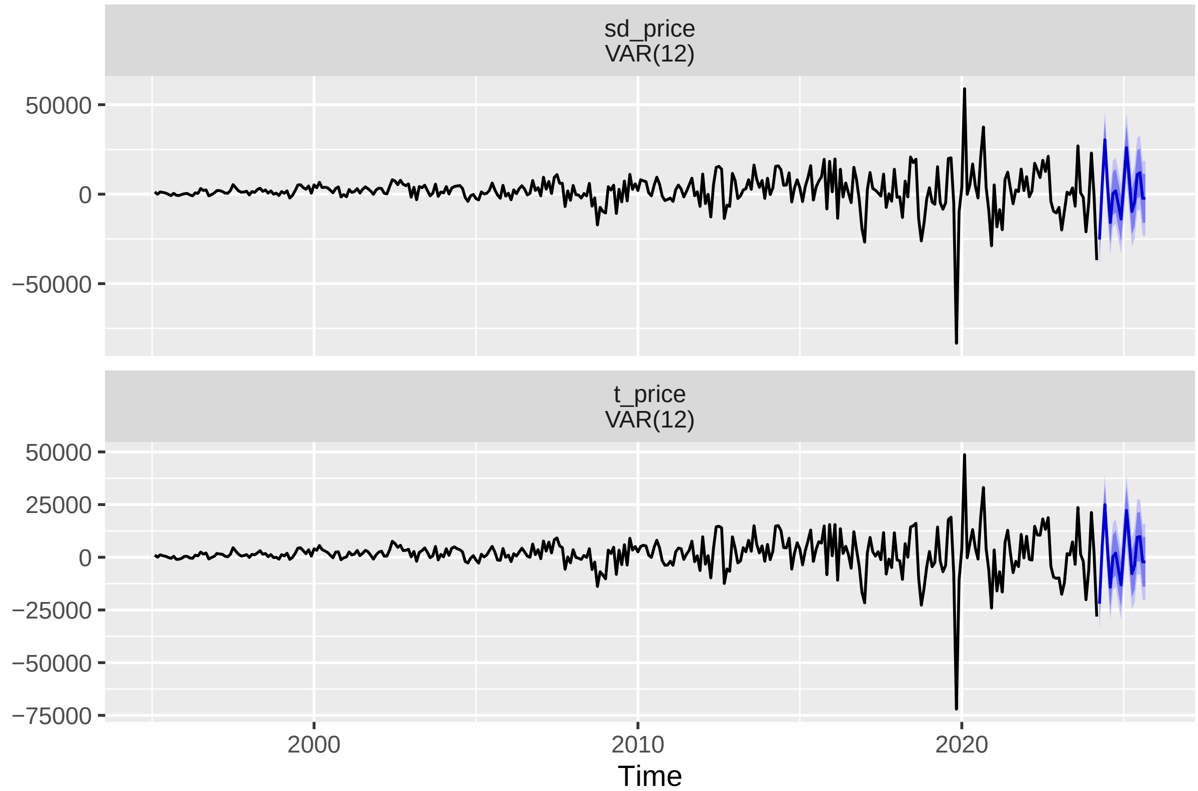
```r
forecast_values<-forecast(var_model,h=18)
autoplot(forecast_values)
```




```r
lagselect2<-VARselect(df.bv2,lag.max=12,type="const")
lagselect2$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     12      9      9     12
```

```r
var_model2<-VAR(df.bv2,p=12,type="const")
forecast_values2<-forecast(var_model2,h=18)
autoplot(forecast_values2)
```




```
summary(var_model2)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: sd_price, t_price
## Deterministic variables: const
## Sample size: 338
## Log Likelihood: -6262.744
## Roots of the characteristic polynomial:
## 0.9553 0.9553 0.9184 0.9184 0.9139 0.9139 0.9037 0.9037 0.8999 0.8999 0.8901 0.8901 0.8874 0.8874 0.8
## Call:
## VAR(y = df.bv2, p = 12, type = "const")
##
##
## Estimation results for equation sd_price:
## =========================================
## sd_price = sd_price.l1 + t_price.l1 + sd_price.l2 + t_price.l2 + sd_price.l3 + t_price.l3 + sd_price
##
##               Estimate Std. Error t value Pr(>|t|)
## sd_price.l1   -0.75070    0.41142  -1.825  0.06900 .
## t_price.l1     1.21061    0.47900   2.527  0.01198 *
## sd_price.l2    0.90377    0.43590   2.073  0.03896 *
## t_price.l2    -0.92611    0.51116  -1.812  0.07098 .
## sd_price.l3    0.29593    0.45511   0.650  0.51602
```

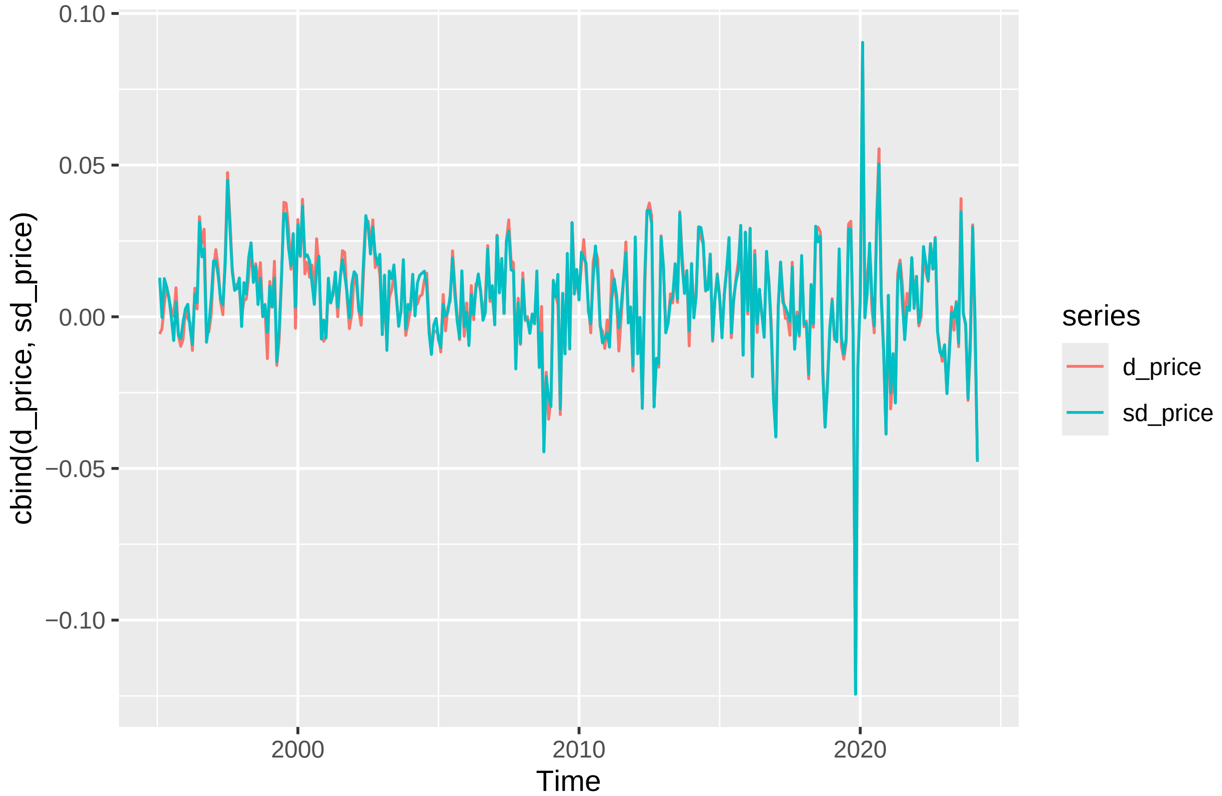
```
## t_price.l3     -1.05600    0.53388  -1.978  0.04881 *
## sd_price.l4    -0.91701    0.48753  -1.881  0.06091 .
## t_price.l4      1.45356    0.57726   2.518  0.01230 *
## sd_price.l5     1.03628    0.48724   2.127  0.03421 *
## t_price.l5     -0.84391    0.57956  -1.456  0.14636
## sd_price.l6     1.22005    0.49458   2.467  0.01417 *
## t_price.l6     -1.87937    0.58752  -3.199  0.00152 **
## sd_price.l7    -0.05396    0.50852  -0.106  0.91556
## t_price.l7      0.11495    0.60417   0.190  0.84923
## sd_price.l8     0.52609    0.50727   1.037  0.30049
## t_price.l8     -0.21339    0.60272  -0.354  0.72355
## sd_price.l9     1.17709    0.51240   2.297  0.02227 *
## t_price.l9     -1.70688    0.60919  -2.802  0.00540 **
## sd_price.l10    0.44850    0.48409   0.926  0.35491
## t_price.l10    -0.73199    0.57155  -1.281  0.20124
## sd_price.l11   -0.97805    0.46142  -2.120  0.03482 *
## t_price.l11     1.38445    0.54586   2.536  0.01169 *
## sd_price.l12   -0.49637    0.45582  -1.089  0.27701
## t_price.l12     0.45931    0.53551   0.858  0.39172
## const        1683.43780  526.45599   3.198  0.00153 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 7734 on 313 degrees of freedom
## Multiple R-Squared: 0.4425,  Adjusted R-squared: 0.3997
## F-statistic: 10.35 on 24 and 313 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation t_price:
## ========================================
## t_price = sd_price.l1 + t_price.l1 + sd_price.l2 + t_price.l2 + sd_price.l3 + t_price.l3 + sd_price.l
##
##                Estimate Std. Error t value Pr(>|t|)
## sd_price.l1    -0.93968    0.35250  -2.666 0.008079 **
## t_price.l1      1.39435    0.41040   3.398 0.000768 ***
## sd_price.l2     0.52641    0.37348   1.409 0.159684
## t_price.l2     -0.50791    0.43796  -1.160 0.247044
## sd_price.l3     0.74357    0.38994   1.907 0.057447 .
## t_price.l3     -1.48284    0.45742  -3.242 0.001316 **
## sd_price.l4    -0.91095    0.41771  -2.181 0.029942 *
## t_price.l4      1.38132    0.49459   2.793 0.005546 **
## sd_price.l5     0.76956    0.41746   1.843 0.066210 .
## t_price.l5     -0.58730    0.49656  -1.183 0.237812
## sd_price.l6     1.41713    0.42375   3.344 0.000925 ***
## t_price.l6     -2.05899    0.50338  -4.090 5.48e-05 ***
## sd_price.l7    -0.18005    0.43569  -0.413 0.679711
## t_price.l7      0.26482    0.51765   0.512 0.609303
## sd_price.l8     0.27812    0.43462   0.640 0.522692
## t_price.l8      0.01791    0.51640   0.035 0.972357
## sd_price.l9     1.35494    0.43902   3.086 0.002208 **
## t_price.l9     -1.85641    0.52195  -3.557 0.000434 ***
## sd_price.l10    0.36866    0.41476   0.889 0.374770
## t_price.l10    -0.60322    0.48969  -1.232 0.218939
```

```
## sd_price.l11   -0.96447    0.39534  -2.440 0.015259 *
## t_price.l11     1.33052    0.46769   2.845 0.004736 **
## sd_price.l12   -0.29101    0.39054  -0.745 0.456744
## t_price.l12     0.24025    0.45882   0.524 0.600912
## const        1407.75388  451.06076   3.121 0.001971 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 6626 on 313 degrees of freedom
## Multiple R-Squared: 0.4542,  Adjusted R-squared: 0.4123
## F-statistic: 10.85 on 24 and 313 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##          sd_price  t_price
## sd_price 59815031 50762251
## t_price  50762251 43909277
##
## Correlation matrix of residuals:
##          sd_price t_price
## sd_price   1.0000  0.9905
## t_price    0.9905  1.0000
```

```r
library(tseries)
library(forecast)
library(vars)
library(readr)
library(Metrics)
df<-read.csv("Updated_Brent.csv")
d<-df$Detached_Average_Price
sd<-df$Semi_Detached_Average_Price
t<-df$Terraced_Average_Price
f<-df$Flat_Average_Price
price_data<-data.frame(d,sd,t, f)
cor_matrix<-cor(price_data)
print(cor_matrix)
```

```
##            d        sd         t         f
## d  1.0000000 0.9997165 0.9990409 0.9917651
## sd 0.9997165 1.0000000 0.9995807 0.9936648
## t  0.9990409 0.9995807 1.0000000 0.9952224
## f  0.9917651 0.9936648 0.9952224 1.0000000
```

```r
df$Date<-as.Date(df$Date,format="%Y-%m-%d")
d_price<-diff(ts(log(df$Detached_Average_Price),start=c(1995,1),frequency=12), d = 1)
sd_price<-diff(ts(log(df$Semi_Detached_Average_Price),start=c(1995,1),frequency=12), d= 1)
t_price<-diff(ts(log(df$Terraced_Average_Price),start=c(1995,1),frequency=12), d = 1)
autoplot(cbind(d_price,sd_price))
```




```r
df.bv<-cbind(d_price,sd_price)
df.bv2<-cbind(sd_price,t_price)
colnames(df.bv)<-cbind("d_price","sd_price")
colnames(df.bv2)<-cbind("sd_price","t_price")
lagselect<-VARselect(df.bv,lag.max=12,type="const")
lagselect$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     11     11     10     11
```

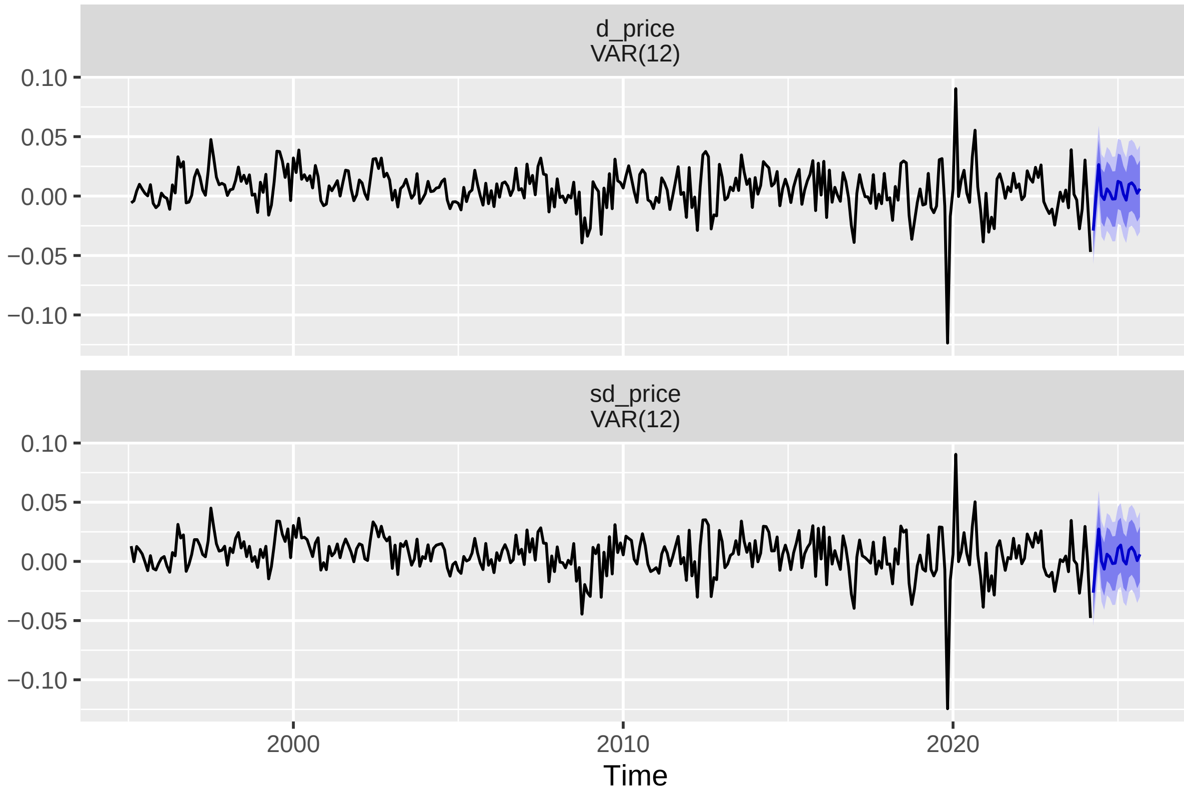
```r
var_model<-VAR(df.bv,p=12,type ="const")
summary(var_model)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: d_price, sd_price
## Deterministic variables: const
## Sample size: 338
## Log Likelihood: 2551.078
## Roots of the characteristic polynomial:
## 0.9618 0.9418 0.9418 0.9353 0.9353 0.9302 0.9302 0.9186 0.9186 0.8831 0.8759 0.8759 0.8755 0.8755 0.8
## Call:
## VAR(y = df.bv, p = 12, type = "const")
```

```
##
##
## Estimation results for equation d_price:
## ========================================
## d_price = d_price.l1 + sd_price.l1 + d_price.l2 + sd_price.l2 + d_price.l3 + sd_price.l3 + d_price.l4
##
##               Estimate Std. Error t value Pr(>|t|)
## d_price.l1    1.423826   0.343561   4.144 4.39e-05 ***
## sd_price.l1  -1.021440   0.345448  -2.957 0.003344 **
## d_price.l2   -1.448578   0.367600  -3.941 0.000100 ***
## sd_price.l2   1.663545   0.364853   4.559 7.37e-06 ***
## d_price.l3   -1.337687   0.386929  -3.457 0.000621 ***
## sd_price.l3   0.742891   0.386561   1.922 0.055539 .
## d_price.l4    1.958330   0.456963   4.286 2.43e-05 ***
## sd_price.l4  -1.566270   0.451681  -3.468 0.000599 ***
## d_price.l5   -0.797023   0.470087  -1.695 0.090979 .
## sd_price.l5   1.063175   0.459199   2.315 0.021244 *
## d_price.l6   -0.975961   0.484572  -2.014 0.044858 *
## sd_price.l6   0.586395   0.477021   1.229 0.219888
## d_price.l7    1.155106   0.475433   2.430 0.015678 *
## sd_price.l7  -1.027147   0.467833  -2.196 0.028859 *
## d_price.l8   -0.387239   0.468073  -0.827 0.408694
## sd_price.l8   0.633174   0.456408   1.387 0.166338
## d_price.l9   -0.220084   0.457972  -0.481 0.631164
## sd_price.l9  -0.015708   0.451201  -0.035 0.972251
## d_price.l10   0.940744   0.378470   2.486 0.013453 *
## sd_price.l10 -1.007394   0.376938  -2.673 0.007922 **
## d_price.l11  -0.471428   0.369954  -1.274 0.203506
## sd_price.l11  0.622357   0.366144   1.700 0.090170 .
## d_price.l12   0.175273   0.334987   0.523 0.601189
## sd_price.l12 -0.240072   0.336200  -0.714 0.475712
## const         0.003105   0.001071   2.900 0.003995 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.01434 on 313 degrees of freedom
## Multiple R-Squared: 0.4017,  Adjusted R-squared: 0.3558
## F-statistic: 8.757 on 24 and 313 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation sd_price:
## =========================================
## sd_price = d_price.l1 + sd_price.l1 + d_price.l2 + sd_price.l2 + d_price.l3 + sd_price.l3 + d_price.l
##
##               Estimate Std. Error t value Pr(>|t|)
## d_price.l1    1.094772   0.341545   3.205 0.001488 **
## sd_price.l1  -0.693292   0.343421  -2.019 0.044362 *
## d_price.l2   -1.740608   0.365442  -4.763 2.92e-06 ***
## sd_price.l2   1.936765   0.362712   5.340 1.79e-07 ***
## d_price.l3   -0.570482   0.384659  -1.483 0.139058
## sd_price.l3  -0.017579   0.384293  -0.046 0.963543
## d_price.l4    1.641806   0.454282   3.614 0.000351 ***
## sd_price.l4  -1.237055   0.449030  -2.755 0.006214 **
```

```
## d_price.l5   -1.136403   0.467328  -2.432 0.015589 *
## sd_price.l5   1.403283   0.456505   3.074 0.002298 **
## d_price.l6   -0.455736   0.481729  -0.946 0.344856
## sd_price.l6   0.058218   0.474222   0.123 0.902372
## d_price.l7    1.019076   0.472643   2.156 0.031836 *
## sd_price.l7  -0.892652   0.465088  -1.919 0.055853 .
## d_price.l8   -0.606030   0.465326  -1.302 0.193745
## sd_price.l8   0.870189   0.453730   1.918 0.056039 .
## d_price.l9    0.100622   0.455285   0.221 0.825229
## sd_price.l9  -0.342908   0.448554  -0.764 0.445160
## d_price.l10   0.802409   0.376249   2.133 0.033731 *
## sd_price.l10 -0.870653   0.374727  -2.323 0.020796 *
## d_price.l11  -0.551282   0.367783  -1.499 0.134899
## sd_price.l11  0.714111   0.363996   1.962 0.050664 .
## d_price.l12   0.047612   0.333021   0.143 0.886406
## sd_price.l12 -0.118837   0.334227  -0.356 0.722411
## const         0.003133   0.001064   2.943 0.003490 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.01426 on 313 degrees of freedom
## Multiple R-Squared: 0.3867,  Adjusted R-squared: 0.3397
## F-statistic: 8.223 on 24 and 313 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##            d_price  sd_price
## d_price  0.0002056 0.0002017
## sd_price 0.0002017 0.0002032
##
## Correlation matrix of residuals:
##          d_price sd_price
## d_price   1.0000   0.9866
## sd_price  0.9866   1.0000
```

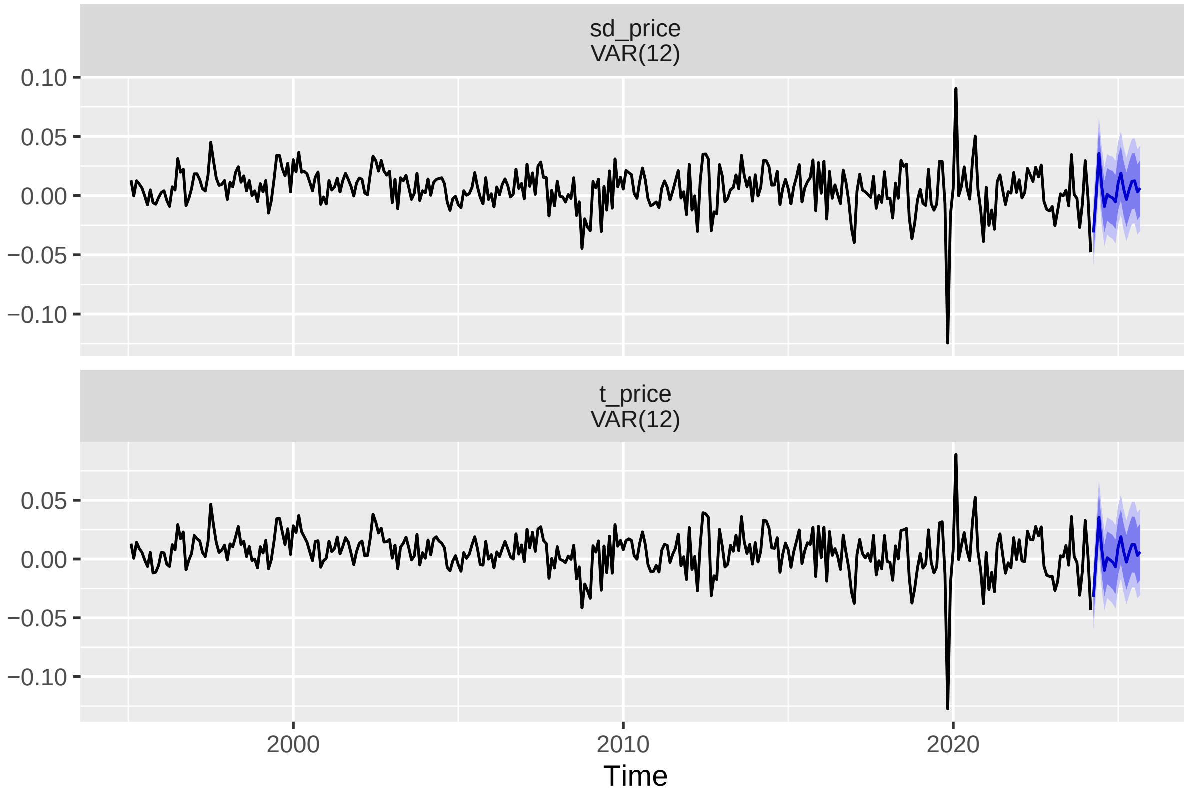
```r
forecast_values<-forecast(var_model,h=18)
autoplot(forecast_values)
```




```
lagselect2<-VARselect(df.bv2,lag.max=12,type="const")
lagselect2$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      9      6      9
```

```
var_model2<-VAR(df.bv2,p=12,type="const")
forecast_values2<-forecast(var_model2,h=18)
autoplot(forecast_values2)
```



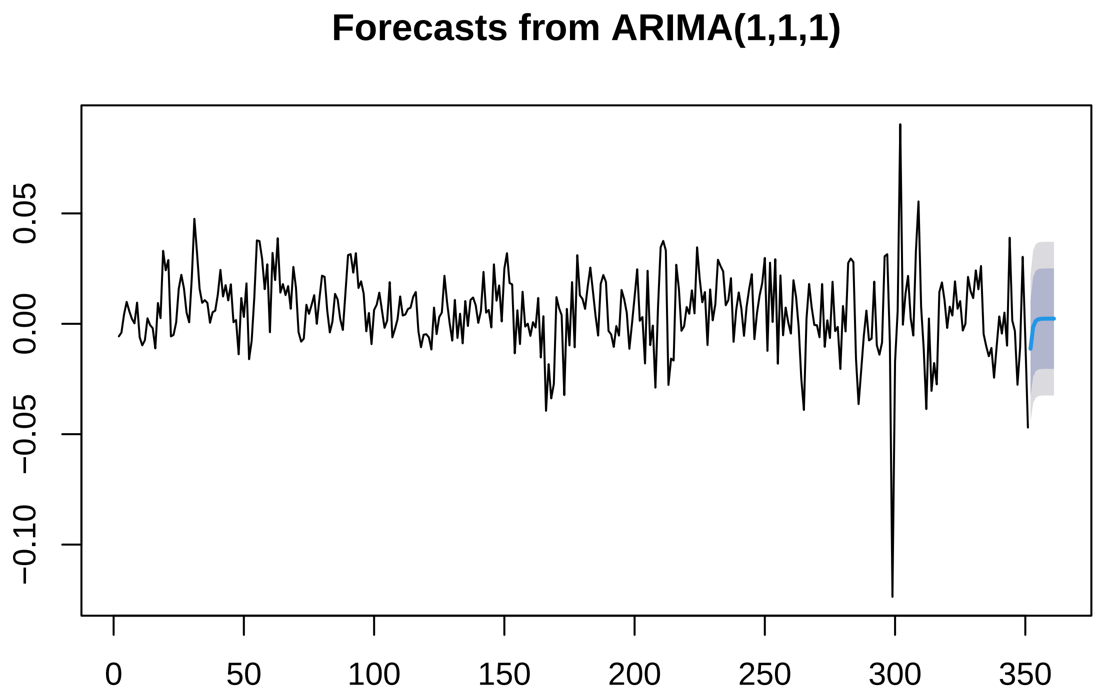

```r
summary(var_model2)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: sd_price, t_price
## Deterministic variables: const
## Sample size: 338
## Log Likelihood: 2577.798
## Roots of the characteristic polynomial:
## 0.9082 0.9082 0.8981 0.8981 0.893 0.893 0.8859 0.8859 0.8757 0.8757 0.8699 0.8699 0.8494 0.8494 0.84
## Call:
## VAR(y = df.bv2, p = 12, type = "const")
##
##
## Estimation results for equation sd_price:
## =========================================
## sd_price = sd_price.l1 + t_price.l1 + sd_price.l2 + t_price.l2 + sd_price.l3 + t_price.l3 + sd_price
##
##               Estimate Std. Error t value Pr(>|t|)
## sd_price.l1  -0.310446   0.384068  -0.808  0.41953
## t_price.l1    0.670630   0.380158   1.764  0.07869 .
## sd_price.l2   0.743585   0.404503   1.838  0.06697 .
## t_price.l2   -0.571213   0.404373  -1.413  0.15877
## sd_price.l3  -0.435950   0.421723  -1.034  0.30206
```

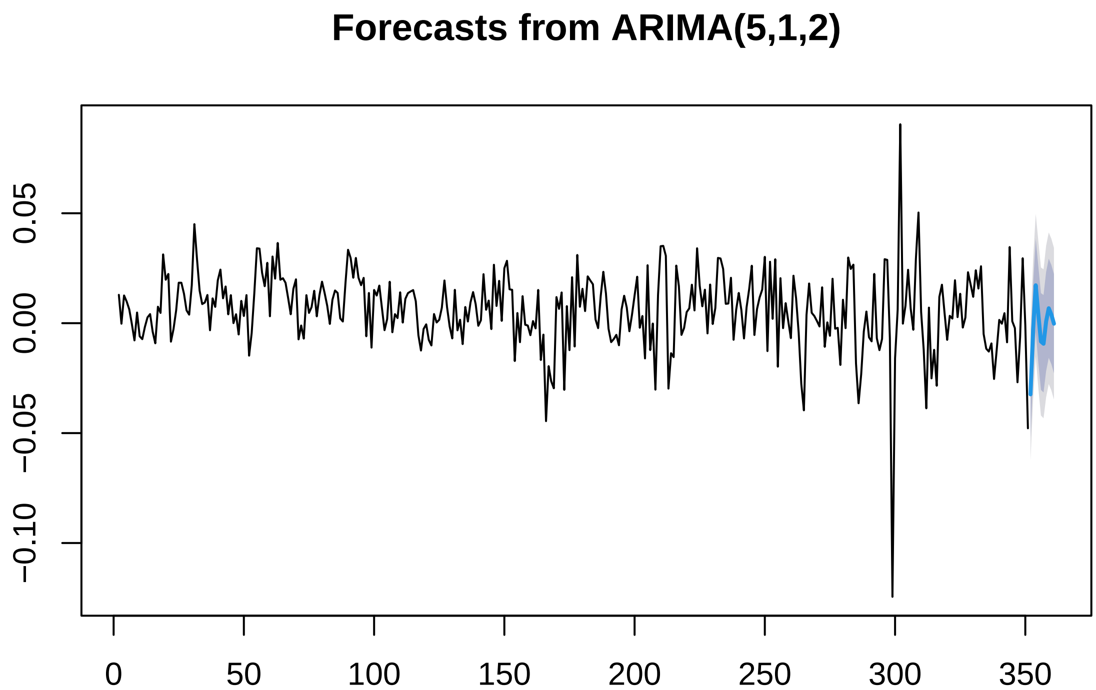
```
## t_price.l3   -0.107749   0.420163  -0.256  0.79778
## sd_price.l4  -0.759753   0.462507  -1.643  0.10145
## t_price.l4    1.139661   0.462112   2.466  0.01419 *
## sd_price.l5   0.964046   0.465415   2.071  0.03914 *
## t_price.l5   -0.680348   0.468353  -1.453  0.14733
## sd_price.l6   0.100143   0.470439   0.213  0.83157
## t_price.l6   -0.433543   0.471919  -0.919  0.35897
## sd_price.l7   0.284216   0.472514   0.601  0.54794
## t_price.l7   -0.160104   0.473709  -0.338  0.73561
## sd_price.l8   0.617291   0.468962   1.316  0.18904
## t_price.l8   -0.350620   0.471886  -0.743  0.45803
## sd_price.l9   0.079256   0.469821   0.169  0.86615
## t_price.l9   -0.346802   0.471229  -0.736  0.46231
## sd_price.l10  0.201184   0.425700   0.473  0.63683
## t_price.l10  -0.267137   0.426284  -0.627  0.53134
## sd_price.l11 -0.215515   0.407376  -0.529  0.59716
## t_price.l11   0.379612   0.409517   0.927  0.35465
## sd_price.l12 -0.612613   0.388796  -1.576  0.11611
## t_price.l12   0.509163   0.387750   1.313  0.19010
## const         0.003319   0.001078   3.078  0.00227 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.01441 on 313 degrees of freedom
## Multiple R-Squared: 0.3737,  Adjusted R-squared: 0.3256
## F-statistic:  7.78 on 24 and 313 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation t_price:
## ========================================
## t_price = sd_price.l1 + t_price.l1 + sd_price.l2 + t_price.l2 + sd_price.l3 + t_price.l3 + sd_price.l
##
##               Estimate Std. Error t value Pr(>|t|)
## sd_price.l1  -0.655224   0.387795  -1.690  0.09210 .
## t_price.l1    1.031096   0.383847   2.686  0.00761 **
## sd_price.l2   0.452458   0.408428   1.108  0.26880
## t_price.l2   -0.296814   0.408297  -0.727  0.46780
## sd_price.l3   0.159110   0.425815   0.374  0.70891
## t_price.l3   -0.706142   0.424241  -1.664  0.09702 .
## sd_price.l4  -0.961400   0.466995  -2.059  0.04035 *
## t_price.l4    1.336452   0.466596   2.864  0.00446 **
## sd_price.l5   0.820032   0.469931   1.745  0.08197 .
## t_price.l5   -0.552127   0.472897  -1.168  0.24388
## sd_price.l6   0.559493   0.475004   1.178  0.23974
## t_price.l6   -0.897823   0.476498  -1.884  0.06046 .
## sd_price.l7   0.144032   0.477099   0.302  0.76294
## t_price.l7   -0.010863   0.478306  -0.023  0.98190
## sd_price.l8   0.436981   0.473513   0.923  0.35680
## t_price.l8   -0.179106   0.476464  -0.376  0.70724
## sd_price.l9   0.389875   0.474380   0.822  0.41178
## t_price.l9   -0.647332   0.475802  -1.361  0.17465
## sd_price.l10  0.175447   0.429831   0.408  0.68342
## t_price.l10  -0.230800   0.430421  -0.536  0.59219
```

```
## sd_price.l11 -0.311868   0.411329  -0.758  0.44890
## t_price.l11   0.471218   0.413491   1.140  0.25532
## sd_price.l12 -0.540040   0.392569  -1.376  0.16991
## t_price.l12   0.437607   0.391513   1.118  0.26454
## const         0.003445   0.001089   3.164  0.00171 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.01455 on 313 degrees of freedom
## Multiple R-Squared: 0.3783,  Adjusted R-squared: 0.3306
## F-statistic: 7.936 on 24 and 313 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##            sd_price   t_price
## sd_price 0.0002076 0.0002073
## t_price  0.0002073 0.0002116
##
## Correlation matrix of residuals:
##          sd_price t_price
## sd_price   1.0000  0.9891
## t_price    0.9891  1.0000
```

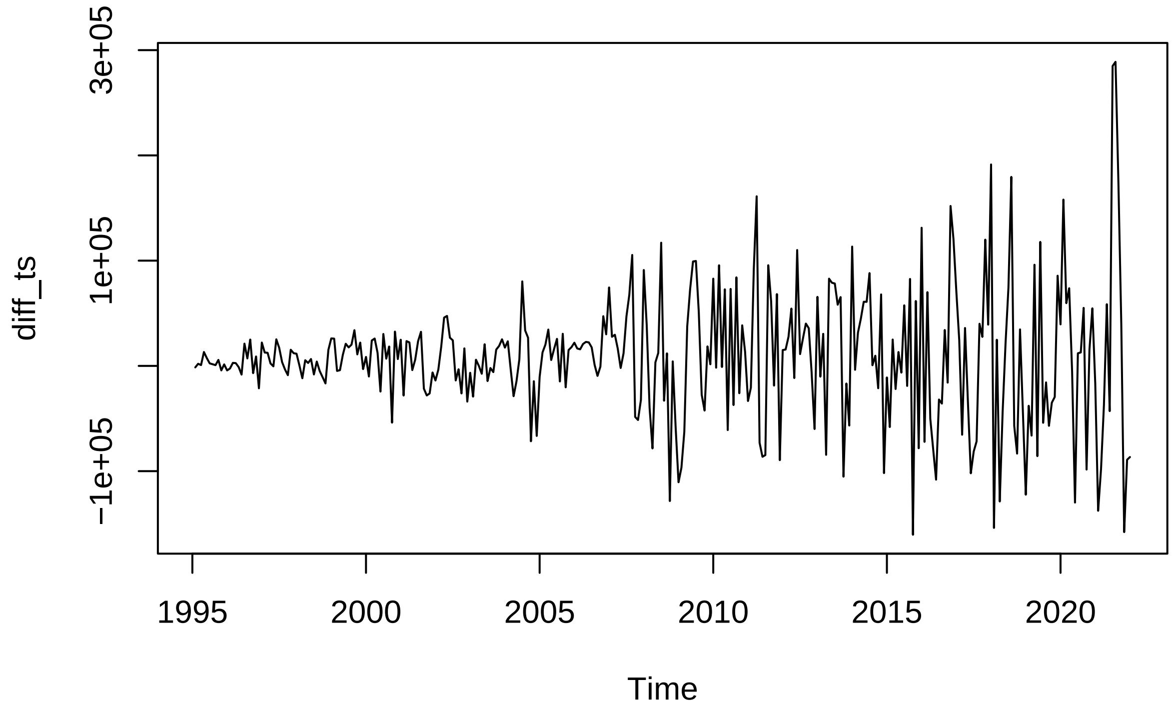
```
d_price <- diff(ts(log(df$Detached_Average_Price)), start = c(1995, 1), frequency = 12)
sd_price <- diff(ts(log(df$Semi_Detached_Average_Price)), start = c(1995, 1), frequency = 12)
d_arima <- auto.arima(d_price)
plot(forecast(d_arima))
```




```r
sd_arima <- auto.arima(sd_price)
plot(forecast(sd_arima))
```



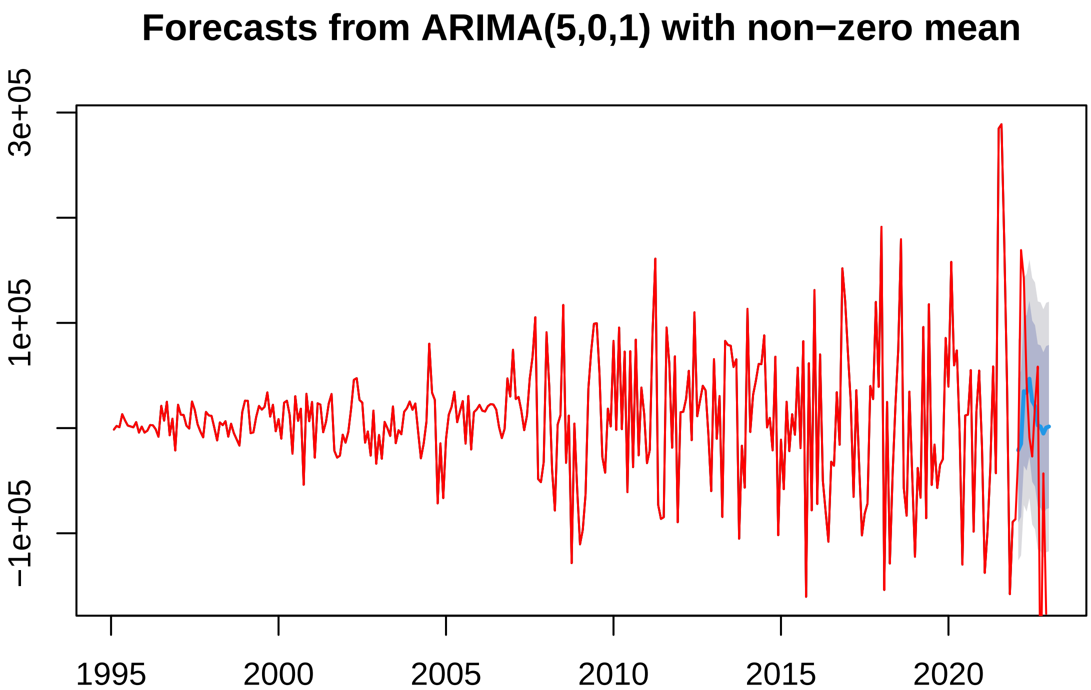

```
library(tseries)
library(forecast)
library(vars)
library(readr)
library(Metrics)
df <- read.csv("Updated_KensingtonandChelsea.csv")
df$Date <- as.Date(df$Date, format = "%Y-%m-%d")
ts <- ts(df$Average_Price, start = c(1995, 1), end = c(2022, 1), frequency = 12)
diff_ts <- diff(ts, differences=1)
ts2022 <- ts(df$Average_Price, start = c(1995, 1), end = c(2022, 12), frequency = 12)
diff_ts2022 <- diff(ts2022, differences = 1)
plot(diff_ts)
```



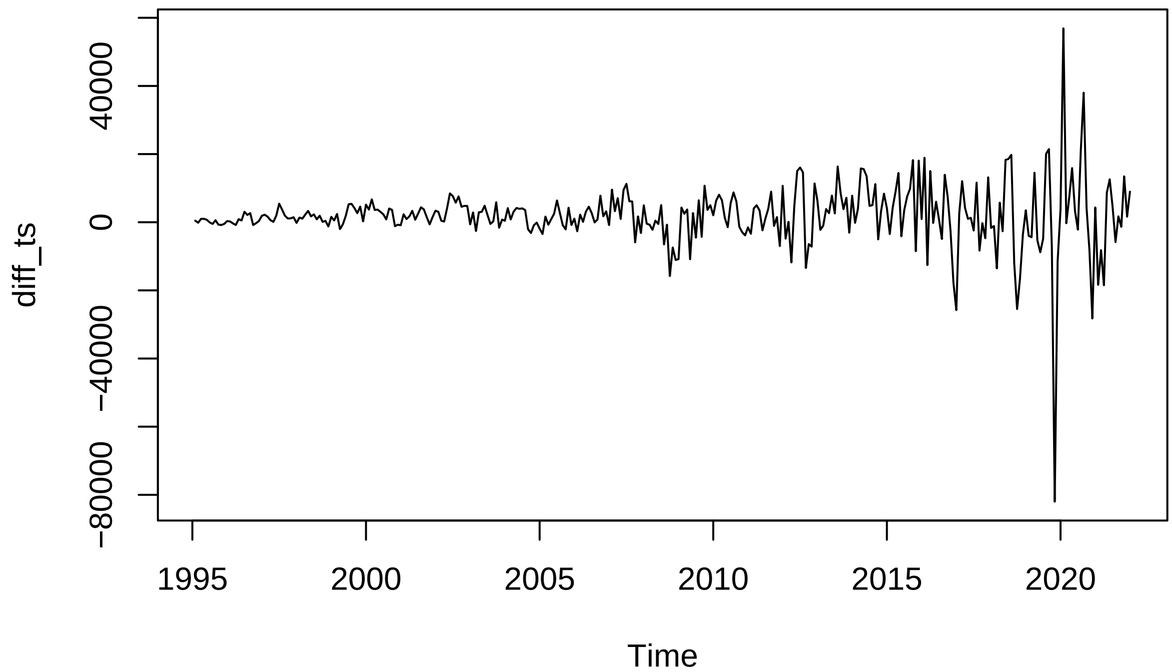

```r
library(tseries)
library(forecast)
adf.test(ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts
## Dickey-Fuller = -2.1431, Lag order = 6, p-value = 0.5162
## alternative hypothesis: stationary
```

```r
arima <- auto.arima(diff_ts)
arima_forecast <- forecast(arima, h = 12)
plot(arima_forecast)
lines(diff_ts2022, col = "red")
```



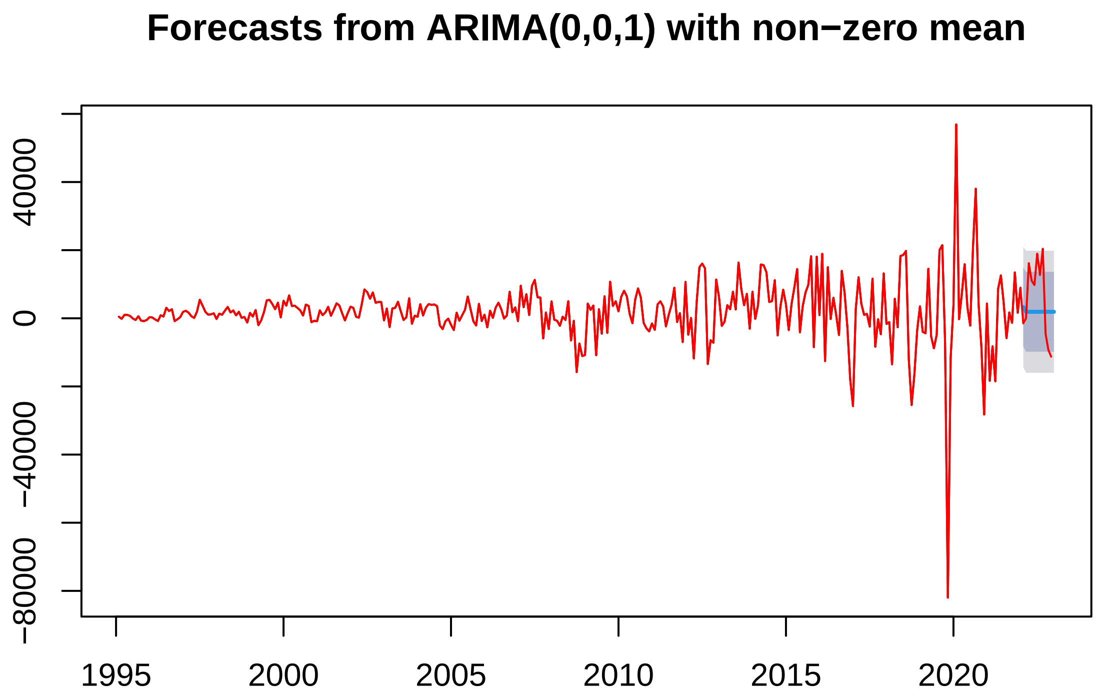

```r
library(tseries)
library(forecast)
library(vars)
library(readr)
library(Metrics)
df <- read.csv("Updated_Brent.csv")
df$Date <- as.Date(df$Date, format = "%Y-%m-%d")
ts <- ts(df$Average_Price, start = c(1995, 1), end = c(2022, 1), frequency = 12)
diff_ts <- diff(ts, differences=1)
ts2022 <- ts(df$Average_Price, start = c(1995, 1), end = c(2022, 12), frequency = 12)
diff_ts2022 <- diff(ts2022, differences = 1)
plot(diff_ts)
```



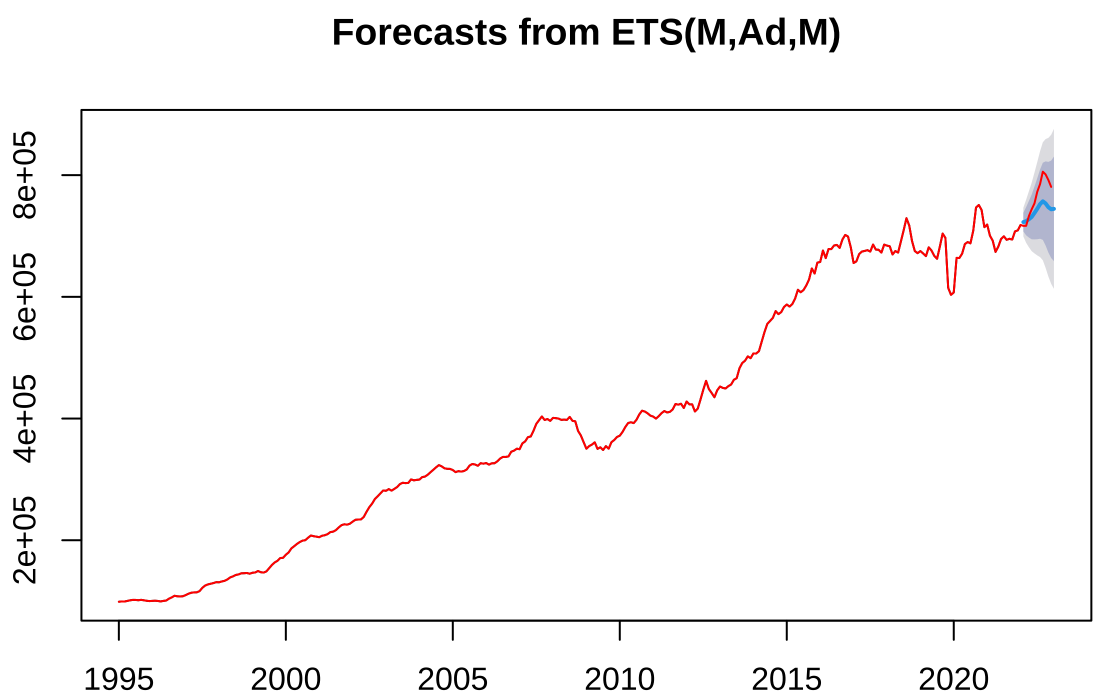

```r
library(tseries)
library(forecast)
adf.test(ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts
## Dickey-Fuller = -2.0401, Lag order = 6, p-value = 0.5596
## alternative hypothesis: stationary
```

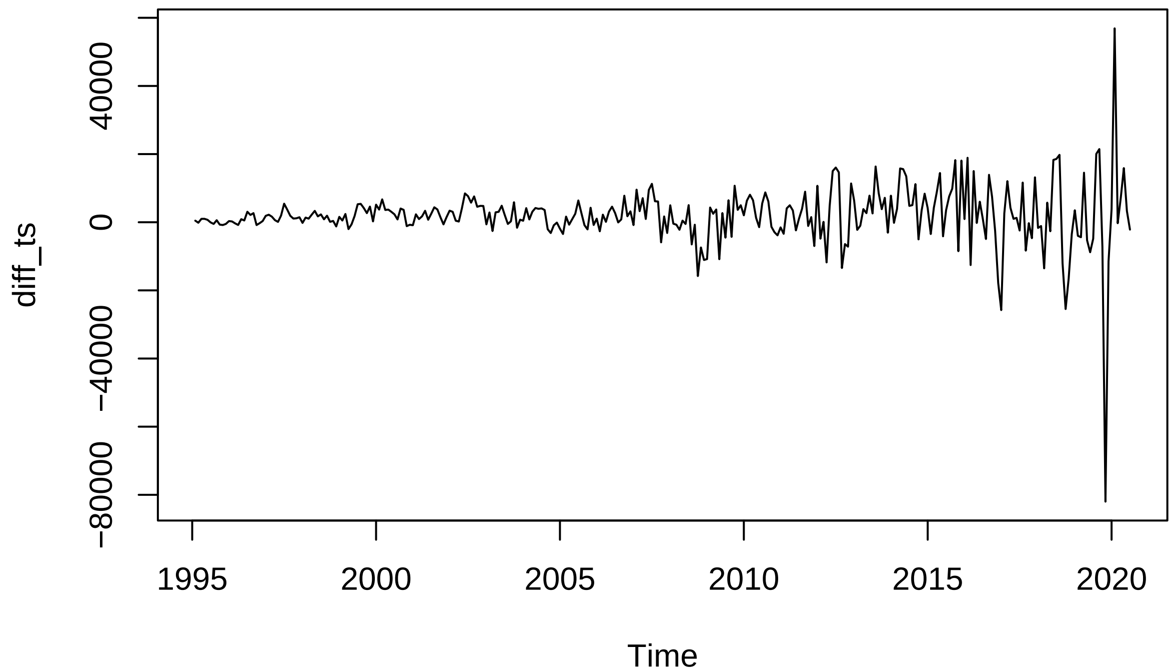
```r
arima <- auto.arima(diff_ts)
plot(forecast(arima, h = 12))
lines(diff_ts2022, col = "red")
```




```
ets <- ets(ts)
ets_forecast <- forecast(ets, h = 12)
plot(ets_forecast)
lines(ts2022, col = "red")
```



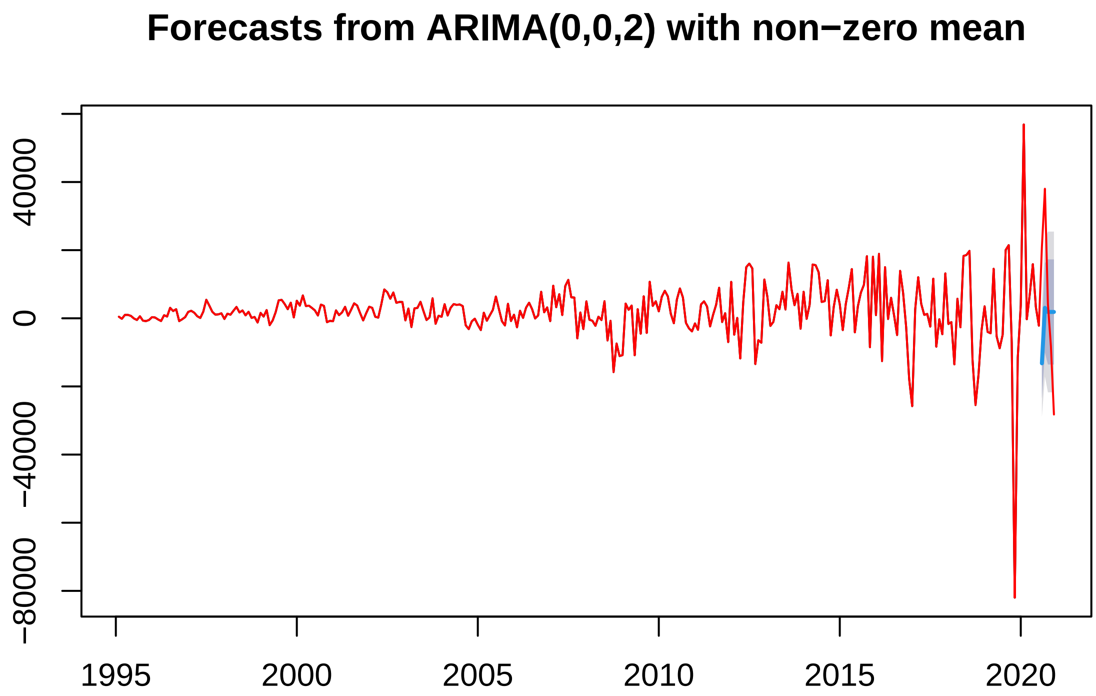

```
library(tseries)
library(forecast)
library(vars)
library(readr)
library(Metrics)
df <- read.csv("Updated_Brent.csv")
df$Date <- as.Date(df$Date, format = "%Y-%m-%d")
ts <- ts(df$Average_Price, start = c(1995, 1), end = c(2020, 7), frequency = 12)
diff_ts <- diff(ts, differences=1)
ts2020 <- ts(df$Average_Price, start = c(1995, 1), end = c(2020, 12), frequency = 12)
diff_ts2020 <- diff(ts2020, differences = 1)
plot(diff_ts)
```



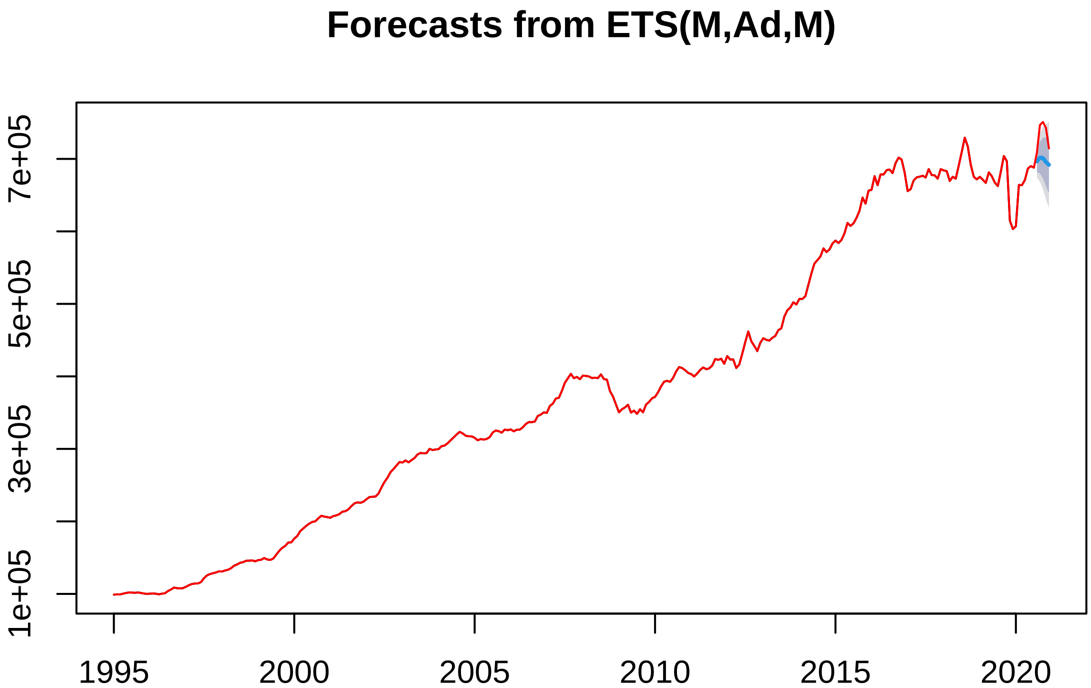

```r
library(tseries)
library(forecast)
adf.test(ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts
## Dickey-Fuller = -1.6042, Lag order = 6, p-value = 0.7432
## alternative hypothesis: stationary
```

```r
arima <- auto.arima(diff_ts)
plot(forecast(arima, h = 5))
lines(diff_ts2020, col = "red")
```




```r
ets <- ets(ts)
ets_forecast <- forecast(ets, h = 5)
plot(ets_forecast)
lines(ts2020, col = "red")
```


Forecasts from ETS(M,Ad,M)
1e+05
3e+05
5e+05
7e+05
1995
2000
2005
2010
2015
2020