ECON_144_Project_2

Marc Luzuriaga, Sia Gao, Reagan Lee, Jiaxuan Huang

2024-11-10

ECON 144 Project 2

Introduction

In this paper, we aim to develop a comprehensive analysis of two interconnected financial and economic time series: the gas price index and the stock price of Chevron Corporation. Our objectives are twofold: (1) to create robust forecasting models for both the gas price index and Chevron's stock price that incorporate trend, seasonality, and cyclical components, and (2) to evaluate the dynamic relationship between these two variables using Vector Autoregressive (VAR) models, Vector Moving Average (VMA) models, and Granger causality tests.

To achieve these goals, we will first fit individual time-series models for each variable and assess their respective trends, seasonal patterns, and cyclical behaviors over a time horizon of approximately 10 years to capture long-term movements. These models will be used to produce 12-step-ahead forecasts and compared with benchmarks such as the `auto.arima()` model to determine forecasting accuracy, measured by Mean Absolute Percentage Error (MAPE).

In addition, we will assess the interactions between the gas price index and Chevron stock prices by constructing a VAR model. This will allow us to analyze how shocks to one variable impact the other over time, which will be further examined through impulse response functions. Finally, Granger causality tests will be performed to determine the directional causality between the two series, providing insights into potential leading and lagging relationships between the gas price index and Chevron stock performance.

Results

(a) Time-Series Plot of Data

Let us load the data for both the gas price index and chevron stock price as follows:

```r
# Define start date for the past decade (approximate 10 years ago)
start_date <- Sys.Date() - (10 * 52 * 7)  # Approximate 10 years in weeks
# Load the gas price index from the last ten years
getSymbols("GASREGW", src = "FRED", periodicity = "weekly", from = start_date)
```

```
## [1] "GASREGW"
```

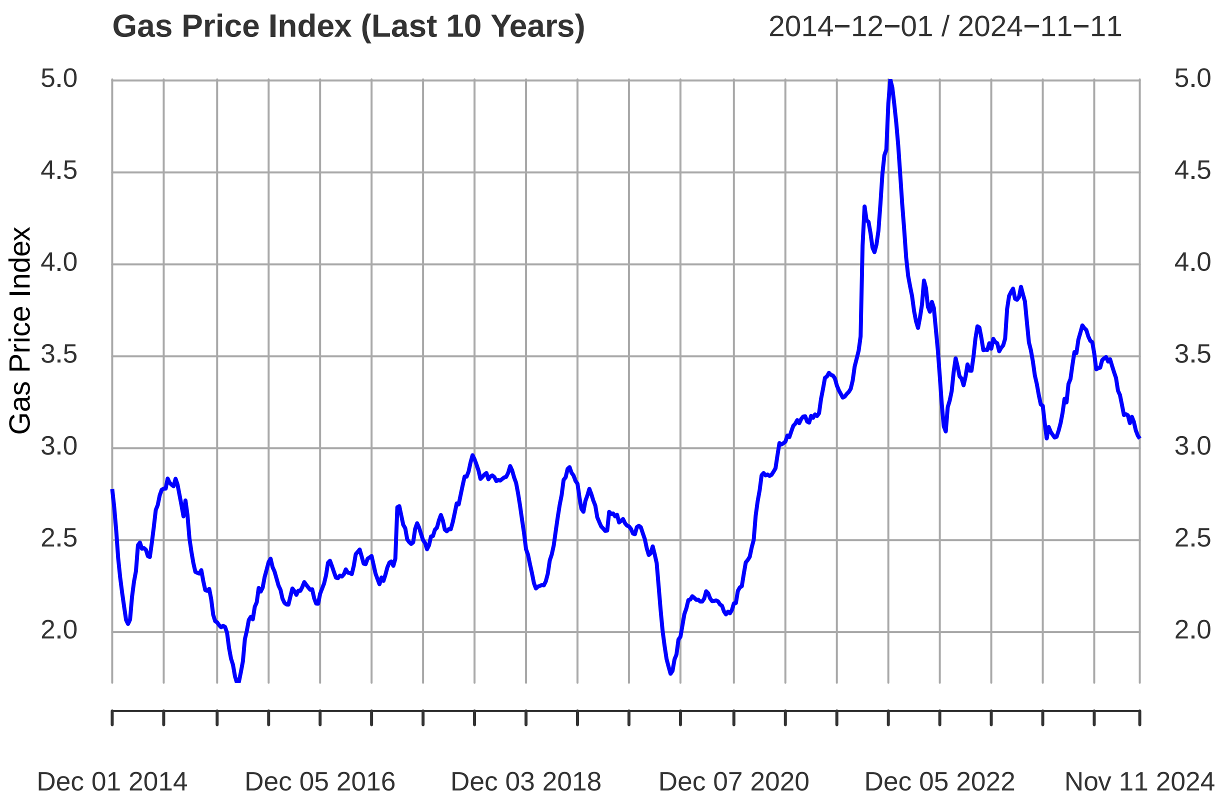
```r
gas_price_index <- na.omit(GASREGW)
# Load the Chevron stock price from the last ten years
getSymbols("CVX", src = "yahoo", periodicity = "weekly", from = start_date)
```

```
## [1] "CVX"
```

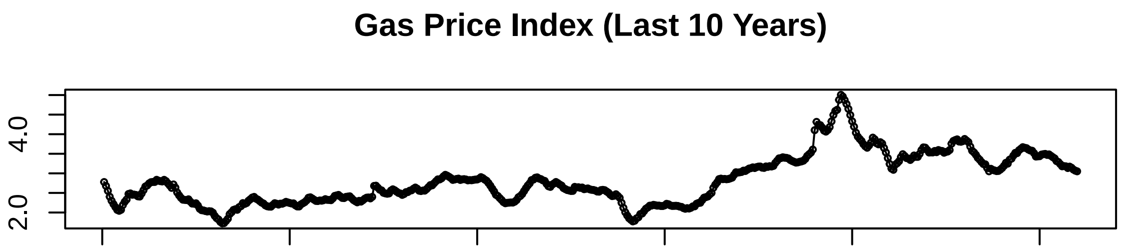
```
cvx_price  <- na.omit(Cl(CVX))
```

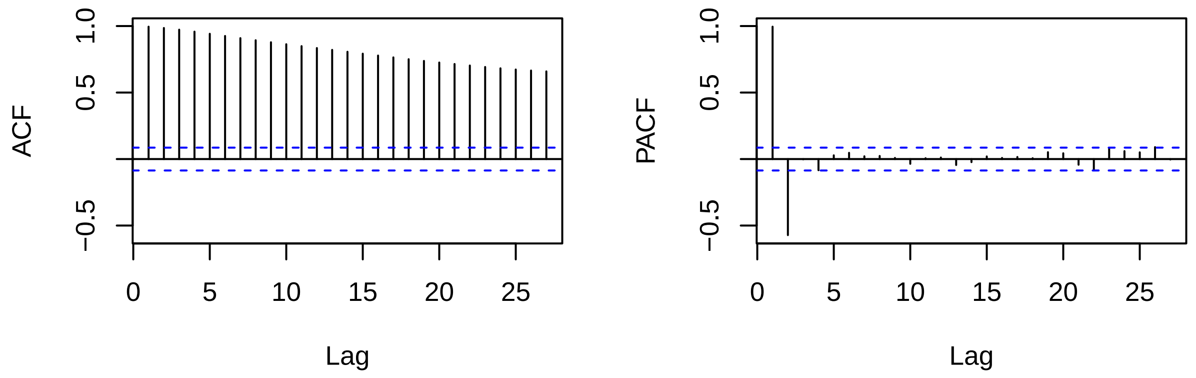
We will proceed to plot the gas price index data, along with the respective ACF and PACF plots as follows:

```
# Plot the gas price index
plot(gas_price_index,
     col = "blue",
     main = "Gas Price Index (Last 10 Years)",
     xlab = "Date",
     ylab = "Gas Price Index")
```



```
# Plot the associated ACFs and PACFs of gas price index
tsdisplay(gas_price_index, main="Gas Price Index (Last 10 Years)")
```

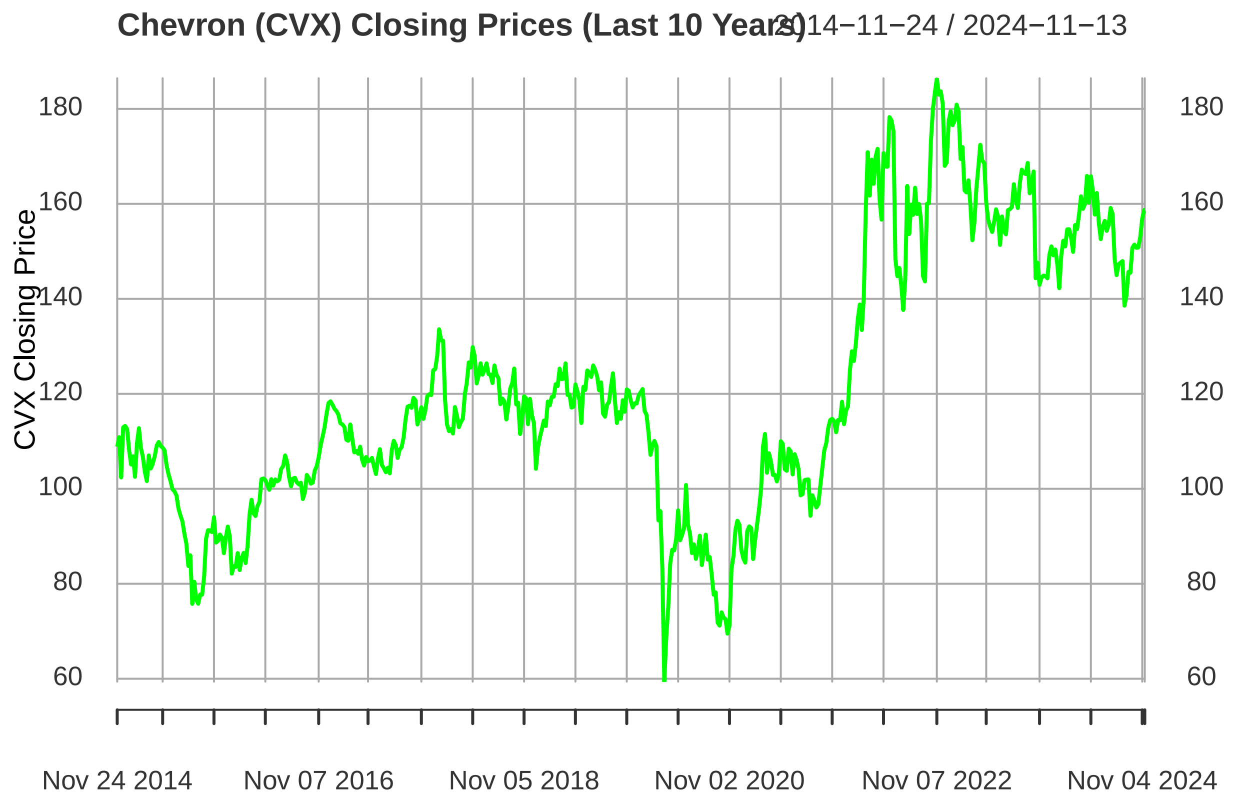




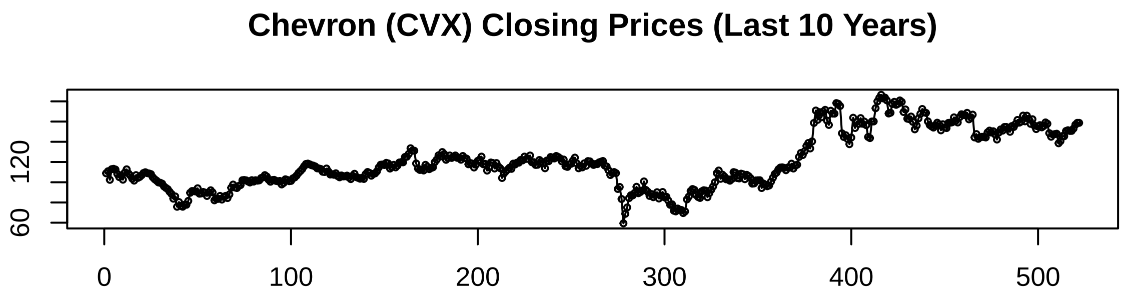
Analyzing the respective ACF and PACF plots, we see that there is a slow decay in the ACF and two statistically significant spikes in the first and second lag. Hence, this suggests to us that we need to fit an AR(2) model.

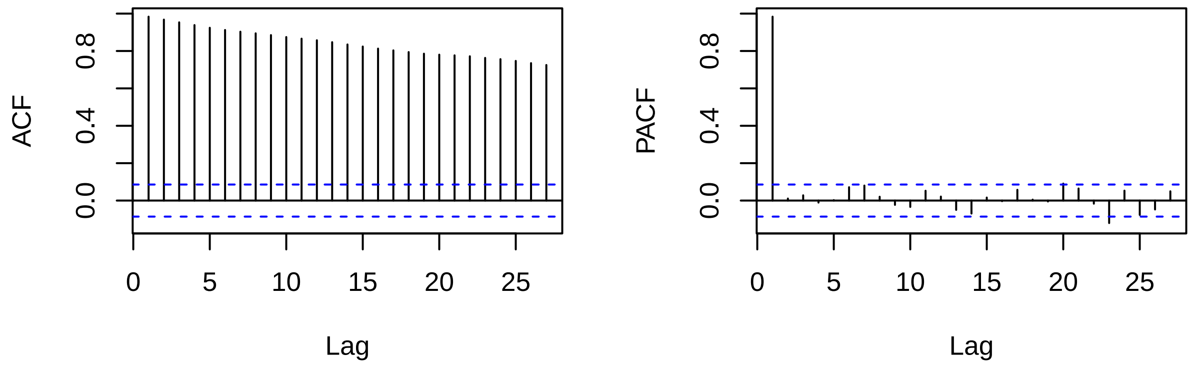
We will proceed to plot the Chevron price, along with the respective ACF and PACF plots as follows:

```r
# Plot the Chevron (CVX) price
plot(cvx_price,
     col = "green",
     main = "Chevron (CVX) Closing Prices (Last 10 Years)",
     xlab = "Date",
     ylab = "CVX Closing Price")
```



```r
# Plot the associated ACFs and PACFs of CVX
tsdisplay(cvx_price, main="Chevron (CVX) Closing Prices (Last 10 Years)")
```



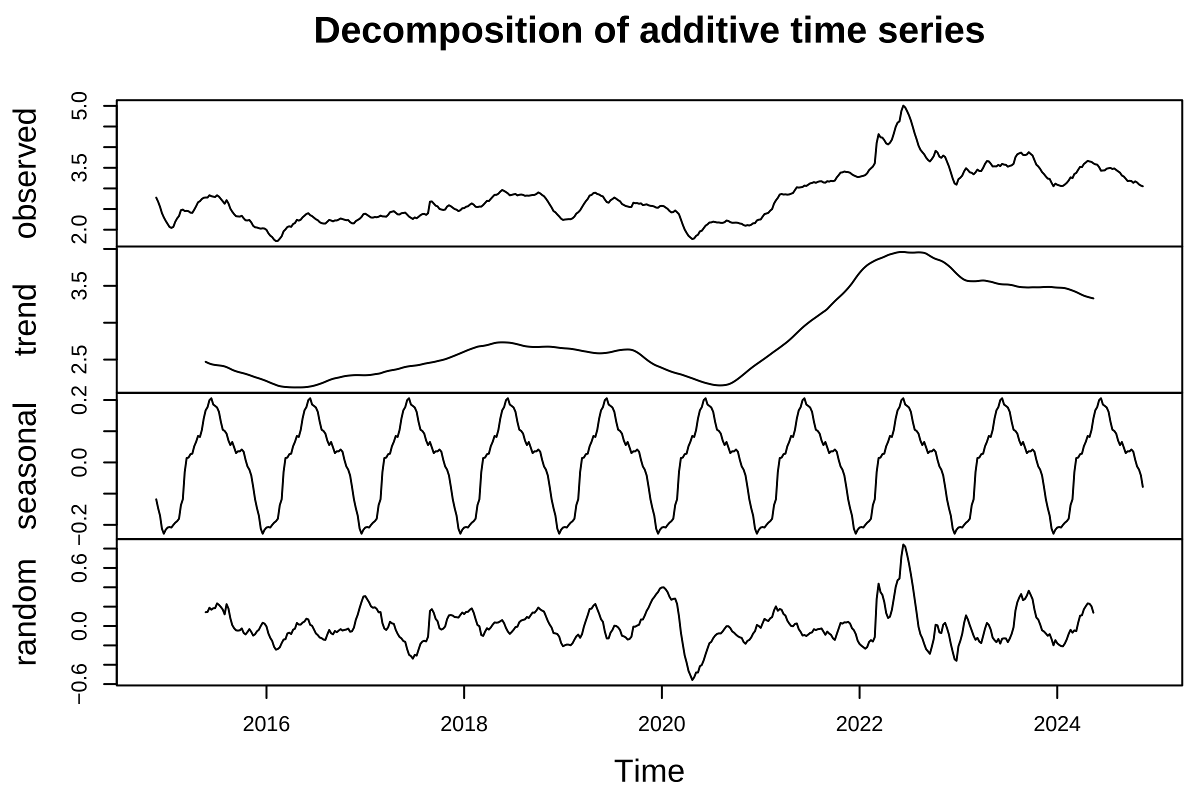


Analyzing the respective ACF and PACF plots, we see that there is a slow decay in the ACF and a single statistically significant spike in the first lag. Hence, this suggests to us that we need to fit an AR(1) model.

(b) STL Decomposition

We will proceed to perform an additive STL Decomposition for the gas price index as follows:

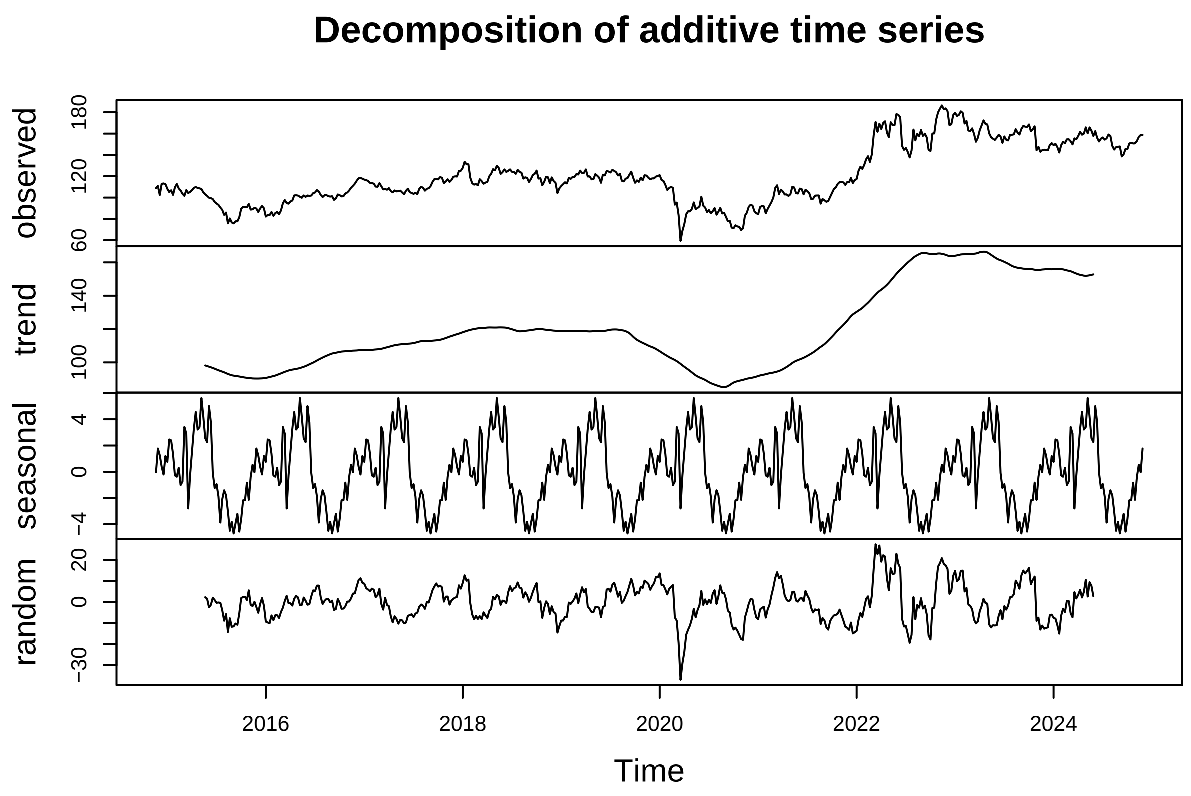
```
#Create the associated ts object
gas_price_index_ts<- ts(gas_price_index$GASREGW,
                              start = c(2014, 47), frequency = 52)
gas_price_index_decomp <- decompose(gas_price_index_ts)
plot(gas_price_index_decomp)
```



From the data above, we see a stable trend from 2014 to 2020. However, after that time period, there is an increase trend from 2020 to 2024. Also, we see an approximately yearly regular seasonal pattern from the seasonal plot above. If we observe the random component, we see that there are cycles that are still present in the residuals. We will have to take into account these cycles in our final model.

We will proceed to perform an additive STL decomposition for the Chevron stock price as follows:

```r
#Create the associated ts object
cvx_price_ts<- ts(cvx_price,
                        start = c(2014, 47), frequency = 52)
cvx_price_decomp <- decompose(cvx_price_ts)
plot(cvx_price_decomp)
```



If we analyze the trend from the plot above, we see a similar trend to that of the gas price index—to which there is stability in the price from 2014 to 2020 and a clear upward trend from 2020 to 2024. As for the seasonal patterns, we see a periodicity of approximately a year, which is similar to that of the gas price index. However, the seasonal pattern is much more volatile and unstable compared to that of the gas price index. Finally, analyzing the random component, we see there are still cycles present in the residuals of our data.

(c) Model fit of Trend, Seasonality, and Cyclical Components

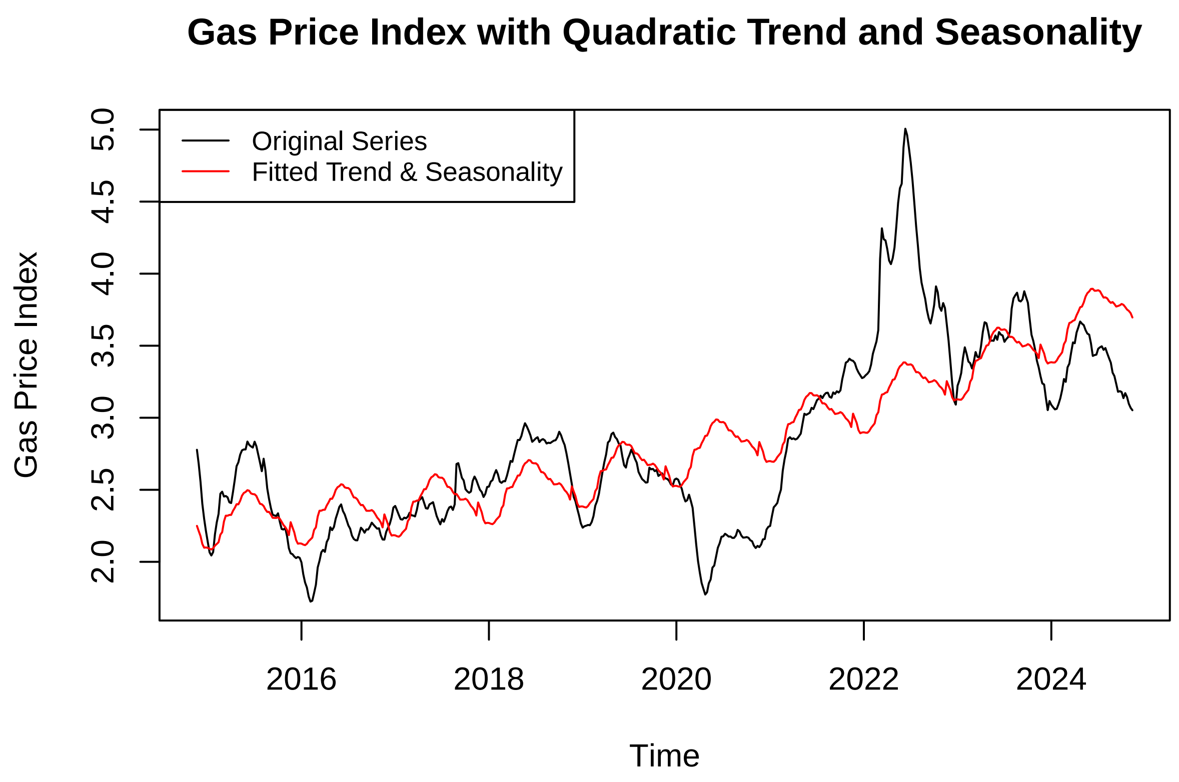
We will propose the following model for the trend and seasonality of the gas price index:

$$Gas_t = T_t + T_t^2 + \sum_{i=1}^{52} \gamma_i M_{i,t} + R_t$$

Let us first fit a model with a quadratic trend and weekly seasonal dummies using the tslm() function as follows:

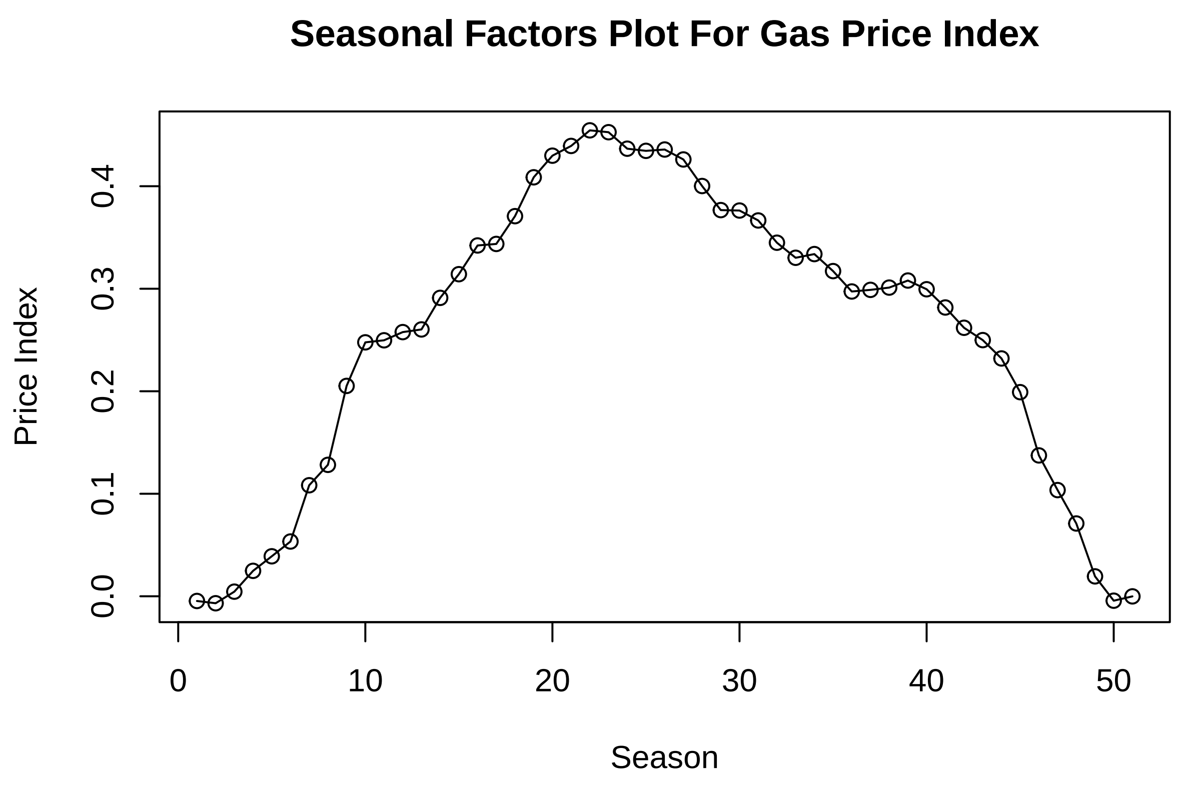
```
#Create the fit model object
fit_gas <- tslm(gas_price_index_ts ~ trend+ I(trend^2)+season)
#Plot the original data
plot(gas_price_index_ts,
     main = "Gas Price Index with Quadratic Trend and Seasonality",
     ylab = "Gas Price Index",
     xlab = "Time")
#Plot the fitted model
```

```r
lines(fitted(fit_gas), col="red")
#Create the legend
legend("topleft",
       legend = c("Original Series", "Fitted Trend & Seasonality"),
       col = c("black", "red"),
       lty = 1,
       cex = 0.8)
```



We can further analyze the seasonal dummies of our model by plotting the seasonal factors plot:

```r
# Create the seasonal plot object
fit_seasonal_gas <- tslm(gas_price_index_ts~season)
# Obtain the associated coefficients
seasonal_coefficients_gas <- coef(fit_seasonal_gas)[-1]
# Plot the associated coefficients
plot(seasonal_coefficients_gas, type = "o",
     main="Seasonal Factors Plot For Gas Price Index",
     xlab="Season",
     ylab="Price Index")
```



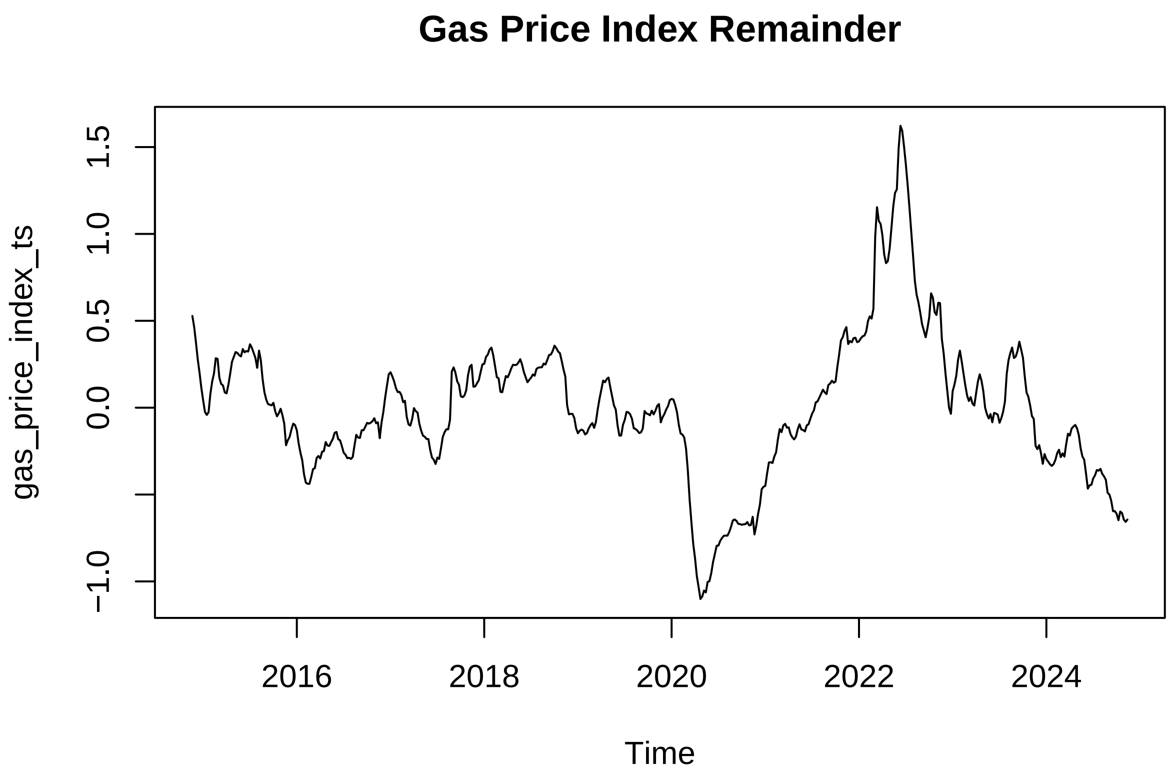
```r
#Summary of the coefficients
summary(fit_seasonal_gas)
```

```
##
## Call:
## tslm(formula = gas_price_index_ts ~ season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1394 -0.4396 -0.1461  0.5218  1.9831
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.5702     0.2035  12.629   <2e-16 ***
## season2      -0.0046     0.2878  -0.016    0.987
## season3      -0.0068     0.2878  -0.024    0.981
## season4       0.0045     0.2878   0.016    0.988
## season5       0.0248     0.2878   0.086    0.931
## season6       0.0390     0.2878   0.136    0.892
## season7       0.0534     0.2878   0.186    0.853
## season8       0.1083     0.2878   0.376    0.707
## season9       0.1282     0.2878   0.445    0.656
## season10      0.2052     0.2878   0.713    0.476
## season11      0.2477     0.2878   0.861    0.390
## season12      0.2497     0.2878   0.868    0.386
## season13      0.2577     0.2878   0.895    0.371
## season14      0.2603     0.2878   0.904    0.366
## season15      0.2911     0.2878   1.011    0.312
## season16      0.3142     0.2878   1.092    0.276
## season17      0.3422     0.2878   1.189    0.235
## season18      0.3437     0.2878   1.194    0.233
## season19      0.3708     0.2878   1.288    0.198
## season20      0.4087     0.2878   1.420    0.156
## season21      0.4298     0.2878   1.493    0.136
## season22      0.4393     0.2878   1.526    0.128
## season23      0.4545     0.2878   1.579    0.115
## season24      0.4527     0.2878   1.573    0.116
## season25      0.4366     0.2878   1.517    0.130
## season26      0.4345     0.2878   1.510    0.132
## season27      0.4358     0.2878   1.514    0.131
## season28      0.4261     0.2878   1.480    0.139
## season29      0.4002     0.2878   1.390    0.165
## season30      0.3767     0.2878   1.309    0.191
## season31      0.3763     0.2878   1.307    0.192
## season32      0.3666     0.2878   1.274    0.203
## season33      0.3449     0.2878   1.198    0.231
## season34      0.3302     0.2878   1.147    0.252
## season35      0.3338     0.2878   1.160    0.247
## season36      0.3172     0.2878   1.102    0.271
## season37      0.2973     0.2878   1.033    0.302
## season38      0.2989     0.2878   1.038    0.300
## season39      0.3011     0.2878   1.046    0.296
## season40      0.3080     0.2878   1.070    0.285
## season41      0.2995     0.2878   1.041    0.299
## season42      0.2817     0.2878   0.979    0.328
## season43      0.2619     0.2878   0.910    0.363
## season44      0.2499     0.2878   0.868    0.386
## season45      0.2320     0.2878   0.806    0.421
## season46      0.1991     0.2878   0.692    0.489
## season47      0.1374     0.2878   0.477    0.633
## season48      0.1036     0.2878   0.360    0.719
## season49      0.0710     0.2878   0.247    0.805
## season50      0.0194     0.2878   0.067    0.946
## season51     -0.0043     0.2878  -0.015    0.988
## season52     -0.0001     0.2878   0.000    1.000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6436 on 468 degrees of freedom
## Multiple R-squared:  0.05516,    Adjusted R-squared:  -0.04781
## F-statistic: 0.5357 on 51 and 468 DF,  p-value: 0.9965
```

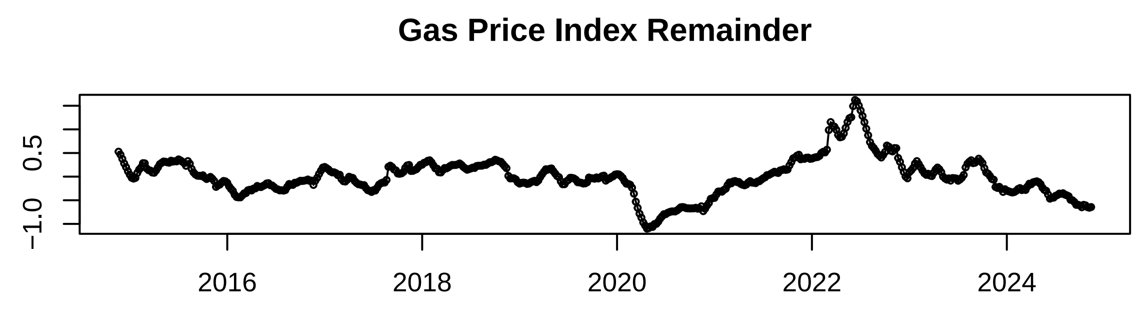
From the seasonal factors plot, we see that the price index increases from the beginning of the year to approximately around week 23. Then, it decreases from week 23 to the end of the year. If we analyze the summary of the fit, we see that seasonality is indeed present from examining the non-statistically significant p-values.

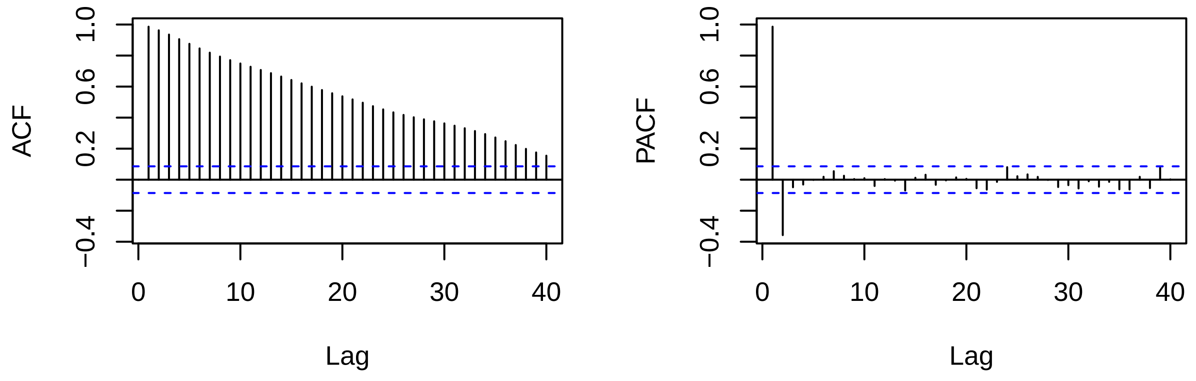
Now, let us examine the remainder component from the fitted model as follows, as well as examine the ACF and PACF of the remainder:

```r
#Create the remainder object
gas_price_index_remainder <- gas_price_index_ts-fitted(fit_gas)
#Plot the remainder object
plot(gas_price_index_remainder, main="Gas Price Index Remainder")
```



```r
#Examine the ACF and PACF of the remainder
tsdisplay(gas_price_index_remainder, lag.max=40, main="Gas Price Index Remainder")
```



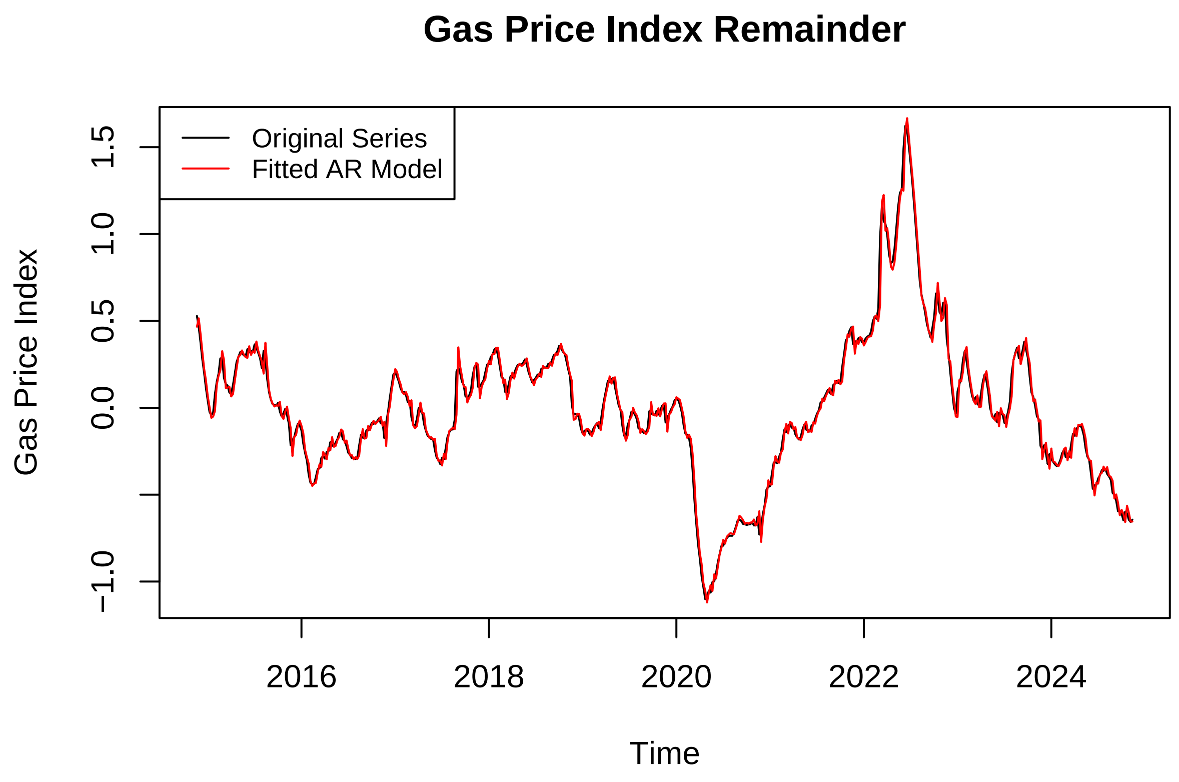


From the model above, we see a slow decay in the ACF, as well as two statistically significant spikes in the first two lags of the PACF. Hence, we propose to fit an AR(2) model to the residuals as follows:

$$R_t = \phi_1 R_{t-1} + \phi_2 R_{t-2} + \epsilon_t$$

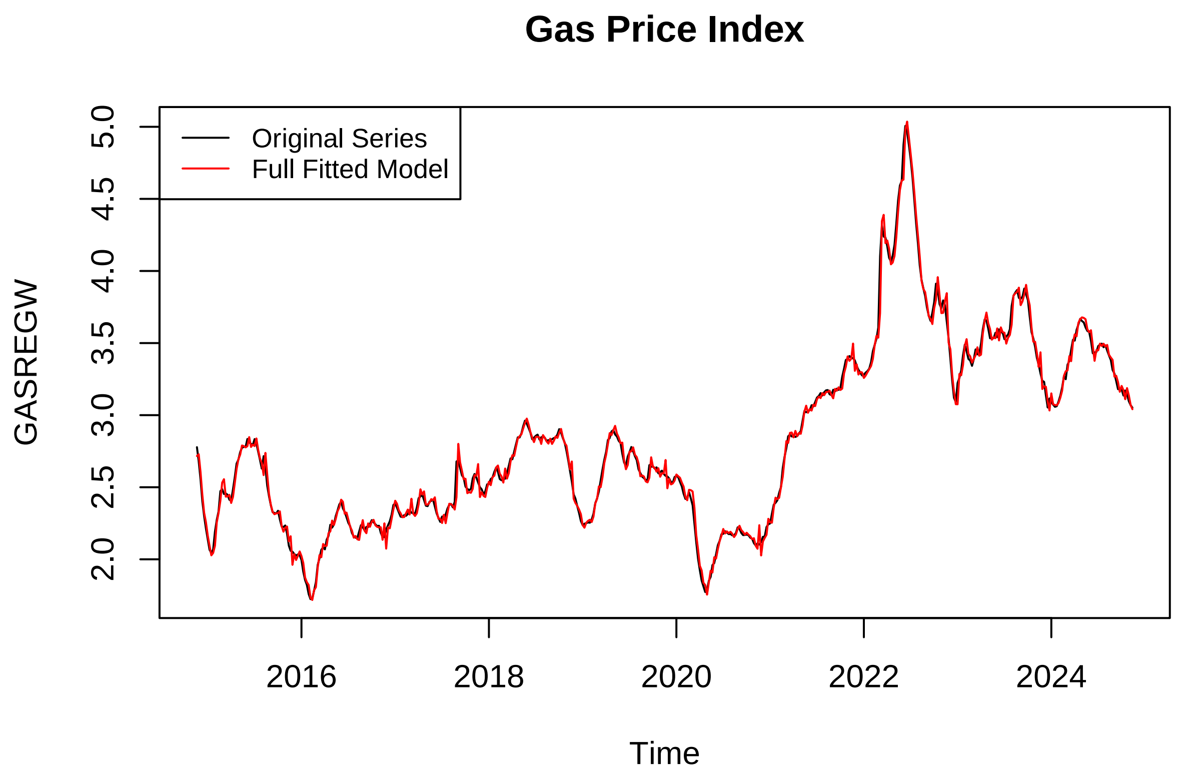
Let us make this fit using the arima() function as follows:

```r
#Create the AR(2) object on hte gas price index remainder
resid_gas <- arima(gas_price_index_remainder, order=c(2,0,0))
#Plot the remainder
plot(gas_price_index_remainder, main="Gas Price Index Remainder",
     ylab="Gas Price Index")
lines(fitted(resid_gas), col="Red")
legend("topleft",
       legend = c("Original Series", "Fitted AR Model"),
       col = c("black", "red"),
       lty = 1,
       cex = 0.8)
```



By visually inspecting the plot above, we see that the AR model does a pretty good job of fitting the data. We will proceed to plot the full model as follows:

```
#Create the full model object
full_model_gas <- fitted(fit_gas) + fitted(resid_gas)
#Plot the remainder object
plot(gas_price_index_ts, main="Gas Price Index")
lines(full_model_gas, col="Red")
legend("topleft",
       legend = c("Original Series", "Full Fitted Model"),
       col = c("black", "red"),
       lty = 1,
       cex = 0.8)
```

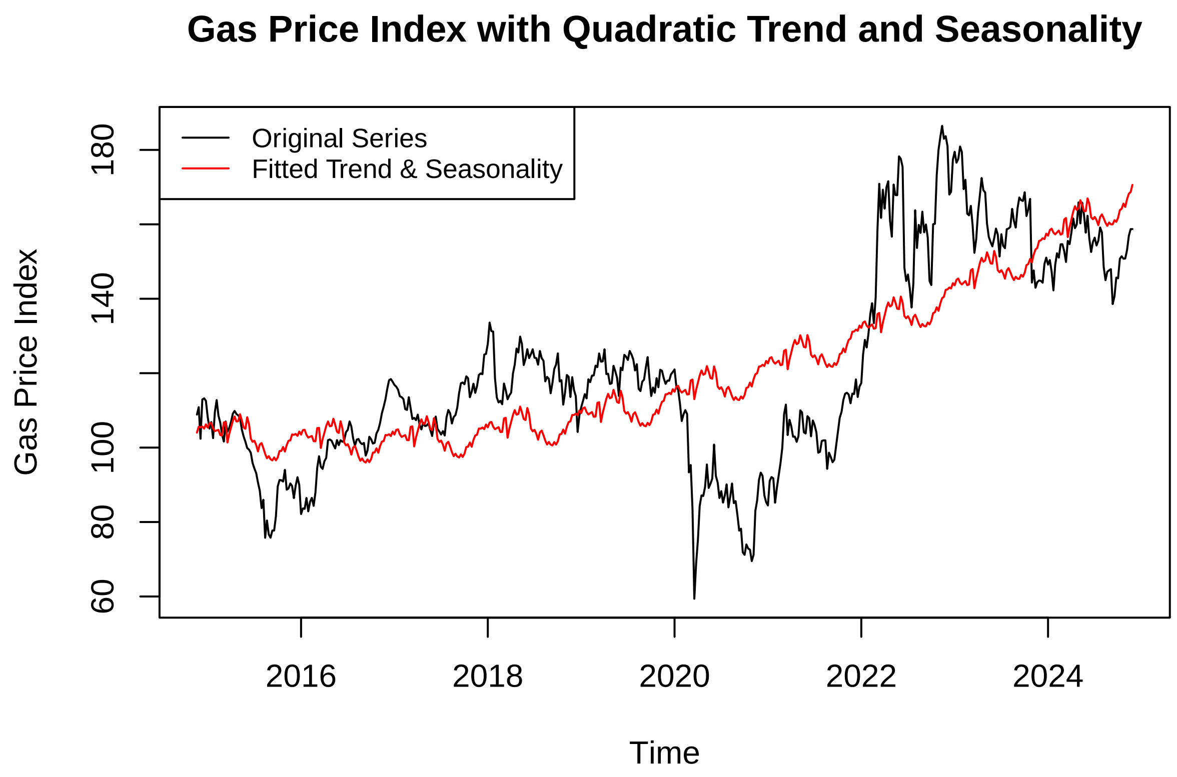


We will propose the following model for the trend and seasonality of the Chevron stock as follows:

$$CVX_t = T_t + T_t^2 + \sum_{i=1}^{52} \gamma_i M_{i,t} + R_t$$

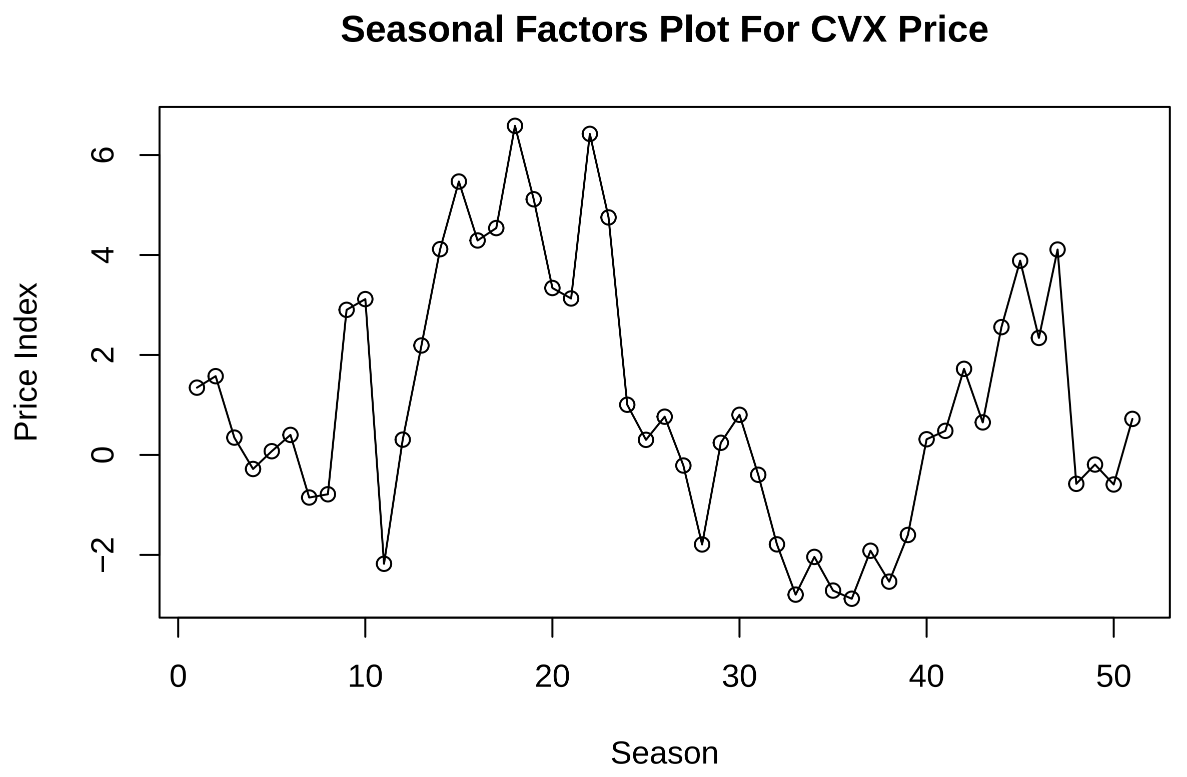
Let us first fit a model with a quadratic trend and weekly seasonal dummies using the tslm() function as follows:

```
#Create the fit model object
fit_cvx <- tslm(cvx_price_ts ~ trend+ I(trend^2)+season)
#Plot the original data
plot(cvx_price_ts,
     main = "Gas Price Index with Quadratic Trend and Seasonality",
     ylab = "Gas Price Index",
     xlab = "Time")
#Plot the fitted model
lines(fitted(fit_cvx),
      col="red")
#Create the legend
legend("topleft",
       legend = c("Original Series", "Fitted Trend & Seasonality"),
       col = c("black", "red"),
       lty = 1,
       cex = 0.8)
```



We can further analyze the seasonal dummies of our model by plotting the seasonal factors plot:

```r
# Create the seasonal plot object
fit_seasonal_cvx <- tslm(cvx_price_ts~season)
# Obtain the associated coefficients
seasonal_coefficients_cvx <- coef(fit_seasonal_cvx)[-1]
# Plot the associated coefficients
plot(seasonal_coefficients_cvx, type = "o",
     main="Seasonal Factors Plot For CVX Price",
     xlab="Season",
     ylab="Price Index")
```



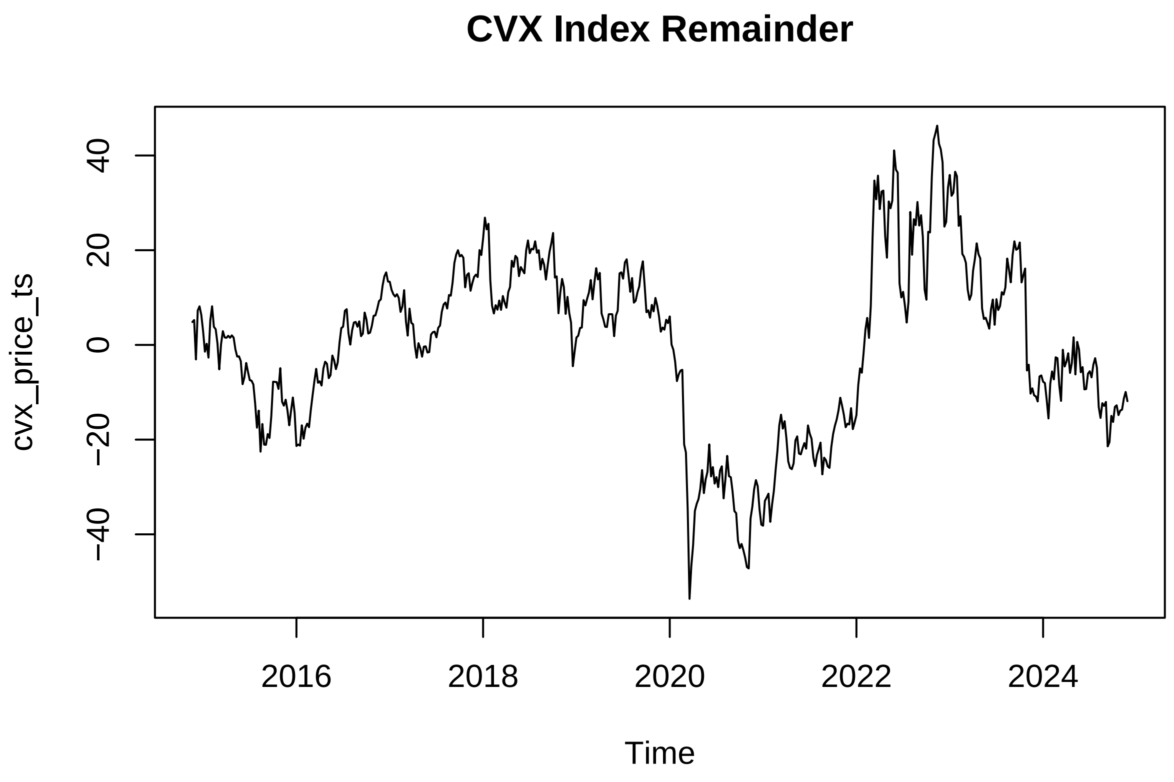
```r
#Summary of the coefficients
summary(fit_seasonal_cvx)
```

```
##
## Call:
## tslm(formula = cvx_price_ts ~ season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.168 -18.941  -5.556  25.179  62.838
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  119.736      9.003  13.300   <2e-16 ***
## season2        1.346     12.732   0.106    0.916
## season3        1.576     12.732   0.124    0.902
## season4        0.347     12.732   0.027    0.978
## season5       -0.281     12.732  -0.022    0.982
## season6        0.075     12.732   0.006    0.995
## season7        0.401     12.732   0.031    0.975
## season8       -0.852     12.732  -0.067    0.947
## season9       -0.787     12.732  -0.062    0.951
## season10       2.903     12.732   0.228    0.820
## season11       3.118     12.732   0.245    0.807
## season12      -2.178     12.732  -0.171    0.864
## season13       0.306     12.732   0.024    0.981
## season14       2.190     12.732   0.172    0.864
## season15       4.117     12.732   0.323    0.747
## season16       5.470     12.732   0.430    0.668
## season17       4.291     12.732   0.337    0.736
## season18       4.539     12.732   0.357    0.722
## season19       6.584     12.732   0.517    0.605
## season20       5.116     12.732   0.402    0.688
## season21       3.340     12.732   0.262    0.793
## season22       3.130     12.732   0.246    0.806
## season23       6.423     12.732   0.504    0.614
## season24       4.752     12.732   0.373    0.709
## season25       1.004     12.732   0.079    0.937
## season26       0.302     12.732   0.024    0.981
## season27       0.765     12.732   0.060    0.952
## season28      -0.212     12.732  -0.017    0.987
## season29      -1.791     12.732  -0.141    0.888
## season30       0.243     12.732   0.019    0.985
## season31       0.803     12.732   0.063    0.950
## season32      -0.397     12.732  -0.031    0.975
## season33      -1.789     12.732  -0.141    0.888
## season34      -2.795     12.732  -0.220    0.826
## season35      -2.040     12.732  -0.160    0.873
## season36      -2.714     12.732  -0.213    0.831
## season37      -2.877     12.732  -0.226    0.821
## season38      -1.917     12.732  -0.151    0.880
## season39      -2.535     12.732  -0.199    0.842
## season40      -1.601     12.732  -0.126    0.900
## season41       0.314     12.732   0.025    0.980
## season42       0.482     12.732   0.038    0.970
## season43       1.722     12.732   0.135    0.892
## season44       0.651     12.732   0.051    0.959
## season45       2.557     12.732   0.201    0.841
## season46       3.886     12.732   0.305    0.760
## season47       2.341     12.439   0.188    0.851
## season48       4.109     12.439   0.330    0.741
## season49      -0.578     12.732  -0.045    0.964
## season50      -0.192     12.732  -0.015    0.988
## season51      -0.591     12.732  -0.046    0.963
## season52       0.721     12.732   0.057    0.955
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.47 on 470 degrees of freedom
## Multiple R-squared:  0.008271,   Adjusted R-squared:  -0.09934
## F-statistic: 0.07686 on 51 and 470 DF,  p-value: 1
```

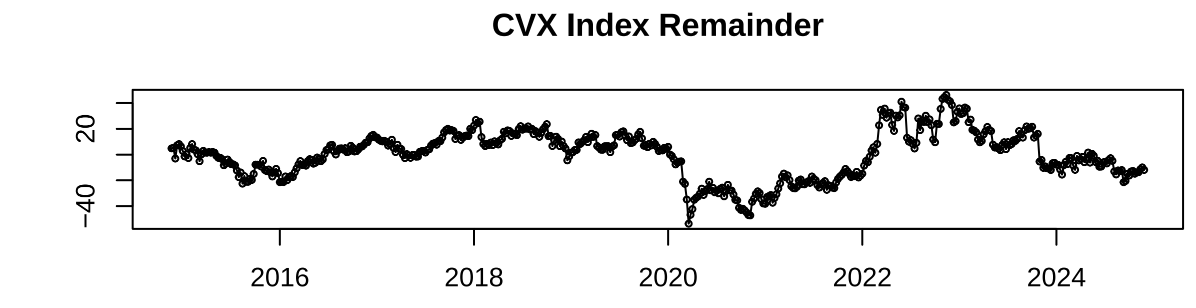
From the seasonal factors plot, seasonality varies throughout the year with the price index bottoming at weeks 10 and 40, while peaking at weeks 20 and 45. If we analyze the summary of the fit, we see that seasonality is indeed present from examining the non-statistically significant p-values.

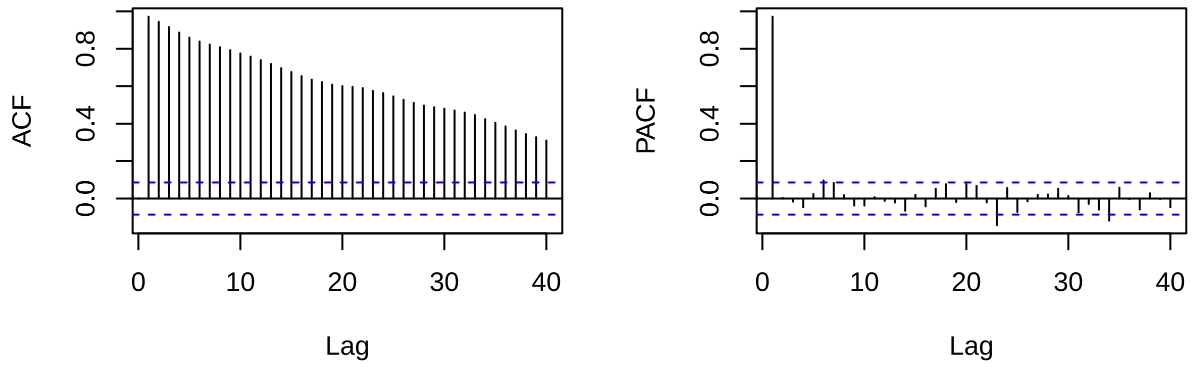
Now, let us examine the remainder component from the fitted model as follows, as well as examine the ACF and PACF of the remainder:

```r
#Create the remainder object
cvx_index_remainder <- cvx_price_ts-fitted(fit_cvx)
#Plot the remainder object
plot(cvx_index_remainder, main="CVX Index Remainder")
```



```r
#Examine the ACF and PACF of the remainder
tsdisplay(cvx_index_remainder, lag.max=40, main="CVX Index Remainder")
```



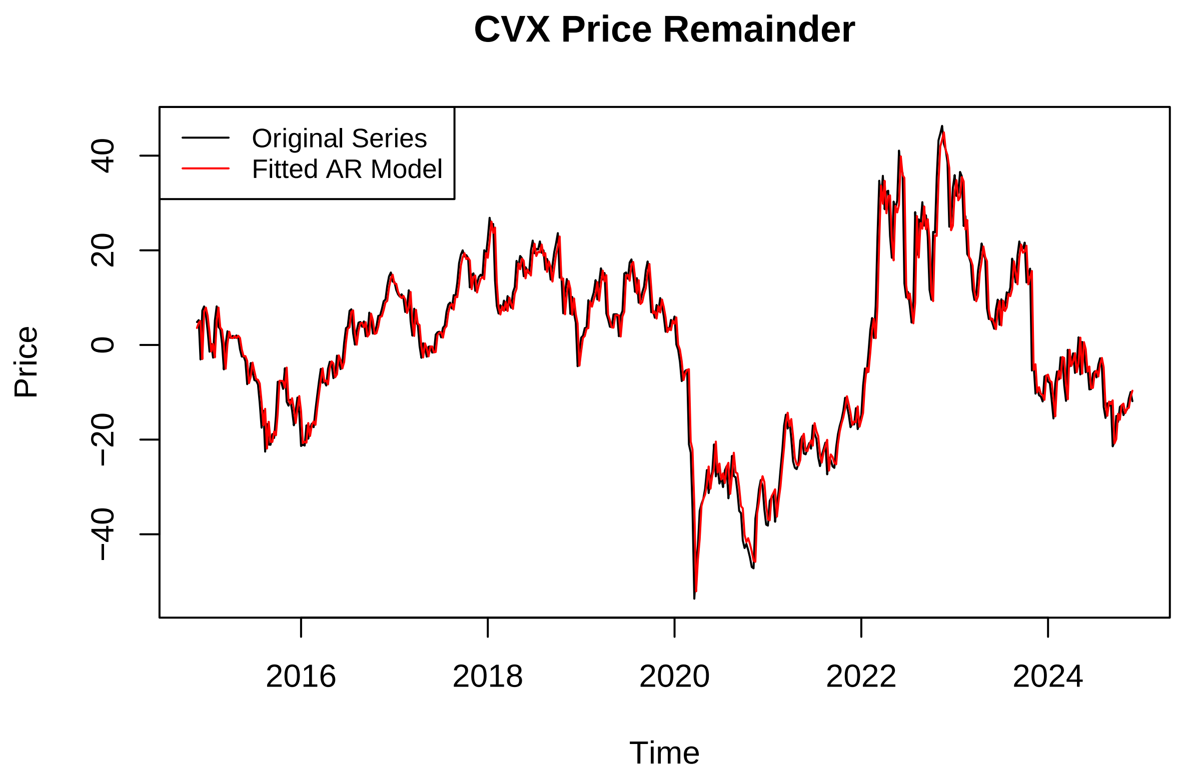


From the model above, we see a slow decay in the ACF, as well as a single statistically significant spike in the first lag of the PACF. Hence, we propose to fit an AR(1) model to the residuals as follows:

$$R_t = \phi_1 R_{t-1} + \epsilon_t$$

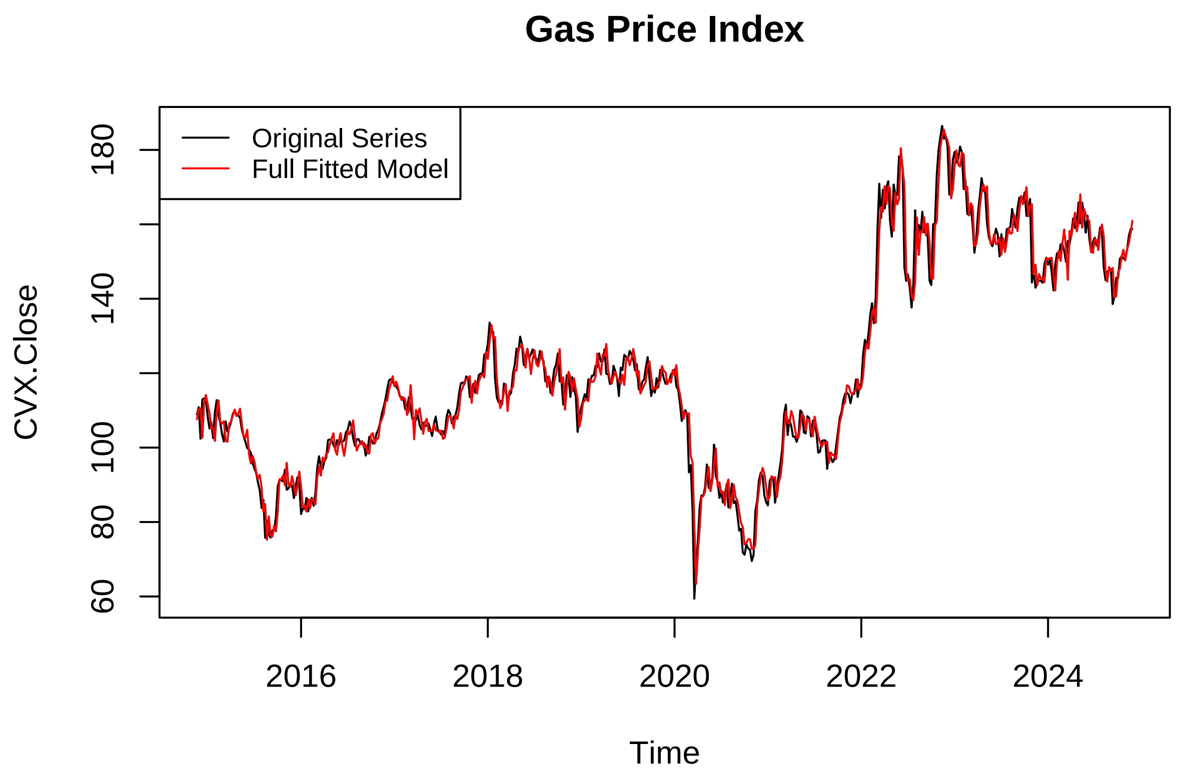
Let us make this fit using the arima() function as follows:

```r
#Create the AR(1) object on the CVX remainder
resid_cvx <- arima(cvx_index_remainder, order=c(1,0,0))
#Plot the remainder
plot(cvx_index_remainder, main="CVX Price Remainder",
     ylab="Price")
lines(fitted(resid_cvx), col="Red")
legend("topleft",
       legend = c("Original Series", "Fitted AR Model"),
       col = c("black", "red"),
       lty = 1,
       cex = 0.8)
```



By visually inspecting the plot above, we see that the AR model does a pretty good job of fitting the data. We will proceed to plot the full model as follows:

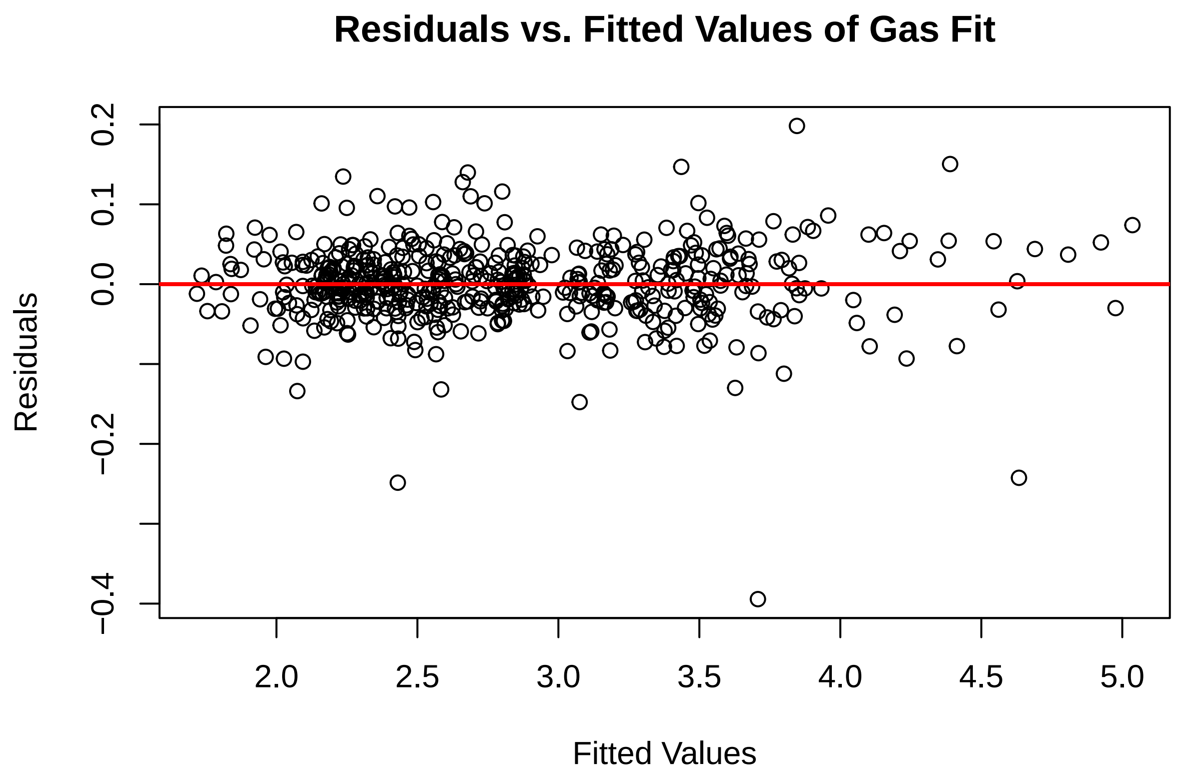
```
#Create the full model object
full_model_cvx <- fitted(fit_cvx) + fitted(resid_cvx)
#Plot the remainder object
plot(cvx_price_ts, main="Gas Price Index")
lines(full_model_cvx, col="Red")
legend("topleft",
       legend = c("Original Series", "Full Fitted Model"),
       col = c("black", "red"),
       lty = 1,
       cex = 0.8)
```



(d) Analysis of Residuals vs Fitted Values

We will proceed to plot the residuals vs fitted values for our gas price index data as follows:

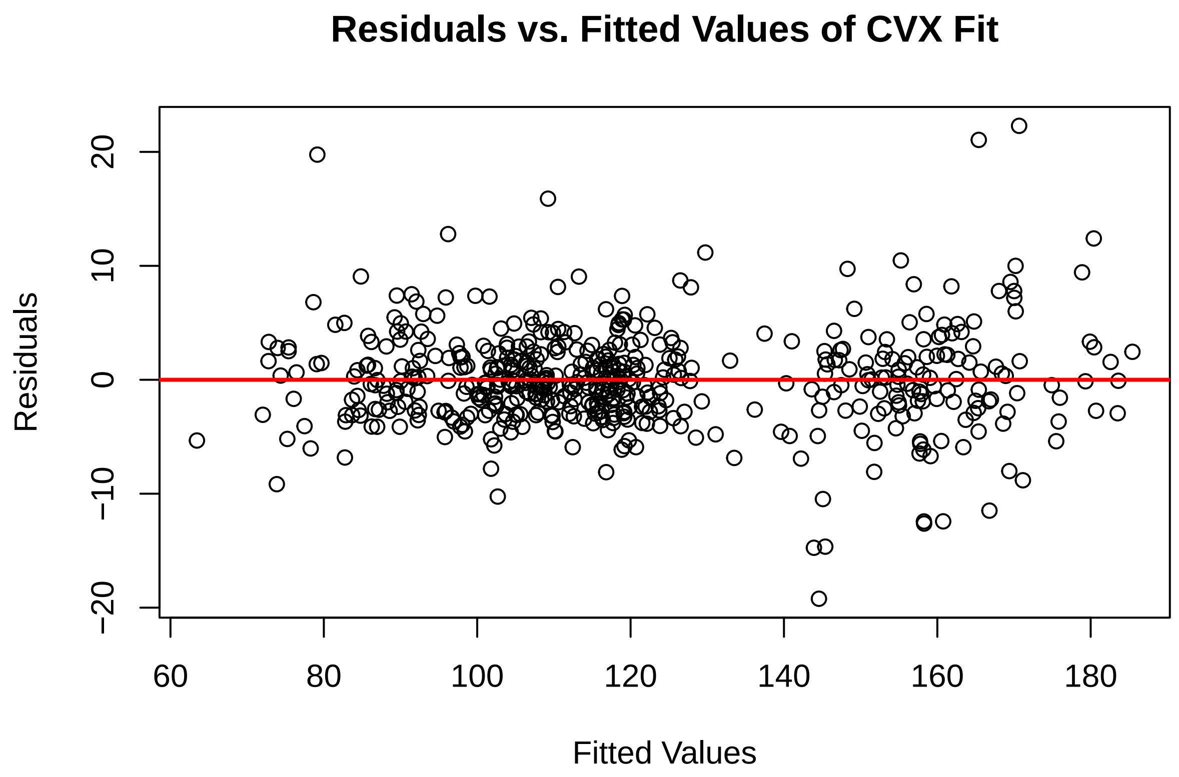
```r
plot(full_model_gas, full_model_gas-gas_price_index_ts,
     main="Residuals vs. Fitted Values of Gas Fit",
     xlab="Fitted Values",
     ylab="Residuals")
# Add a horizontal line at 0 for reference
abline(h = 0, col = "red", lwd = 2)
```



Analyzing the residuals vs fitted values of the gas plot, we see that the residuals are randomly distributed around 0 with no clear pattern. Although there are some outliers, the specific outliers don't hold much weight with respect to the rest of the points. Also, the spread of residuals appears to be consistent, suggesting that variance of the residuals exhibit homoscedasticity.

We will proceed to plot the residuals vs fitted values for our CVX price as follows:

```r
plot(full_model_cvx, full_model_cvx-cvx_price_ts,
     main="Residuals vs. Fitted Values of CVX Fit",
     xlab="Fitted Values",
     ylab="Residuals")
# Add a horizontal line at 0 for reference
abline(h = 0, col = "red", lwd = 2)
```



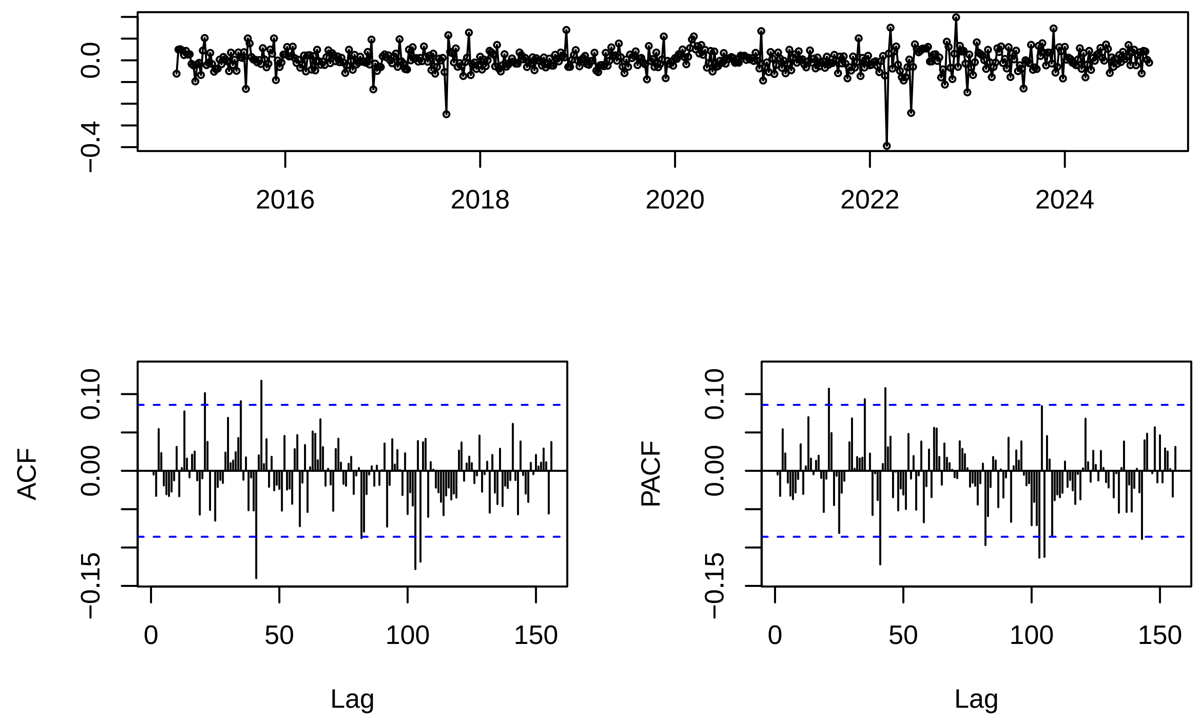
The residuals for the CVX fit are similar to that of the Gas fit. Analyzing the residuals vs fitted values of the gas plot, we see that the residuals are randomly distributed around 0 with no clear pattern. Also, the spread of residuals appears to be consistent, suggesting that variance of the residuals exhibit homoscedasticity.

(e) ACF and PACF of respective residuals

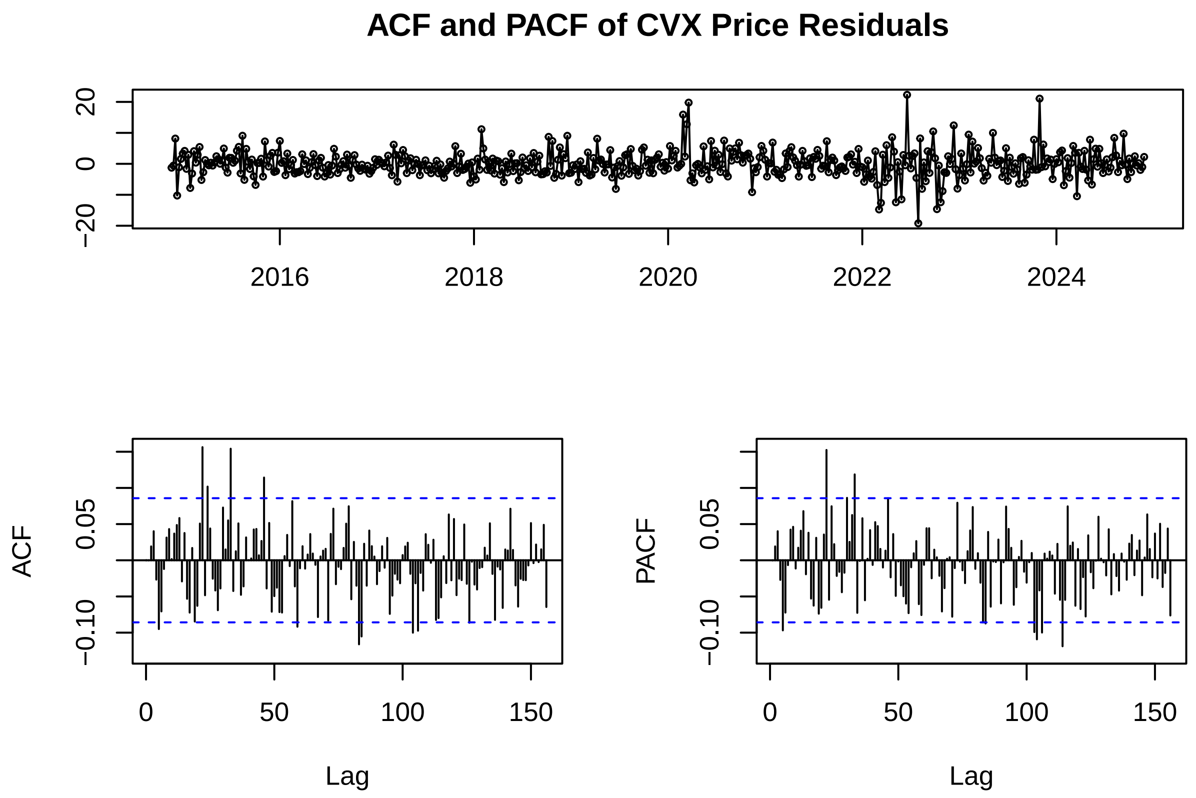
We will proceed to plot the ACF and PACF of the respective residuals as follows:

```r
# ACF and PACF for gas price index residuals
tsdisplay(full_model_gas - gas_price_index_ts,
          main="ACF and PACF of Gas Price Index Residuals")
```

ACF and PACF of Gas Price Index Residuals



```r
# ACF and PACF for Chevron stock price (CVX) residuals
tsdisplay(full_model_cvx - cvx_price_ts,
          main="ACF and PACF of CVX Price Residuals")
```

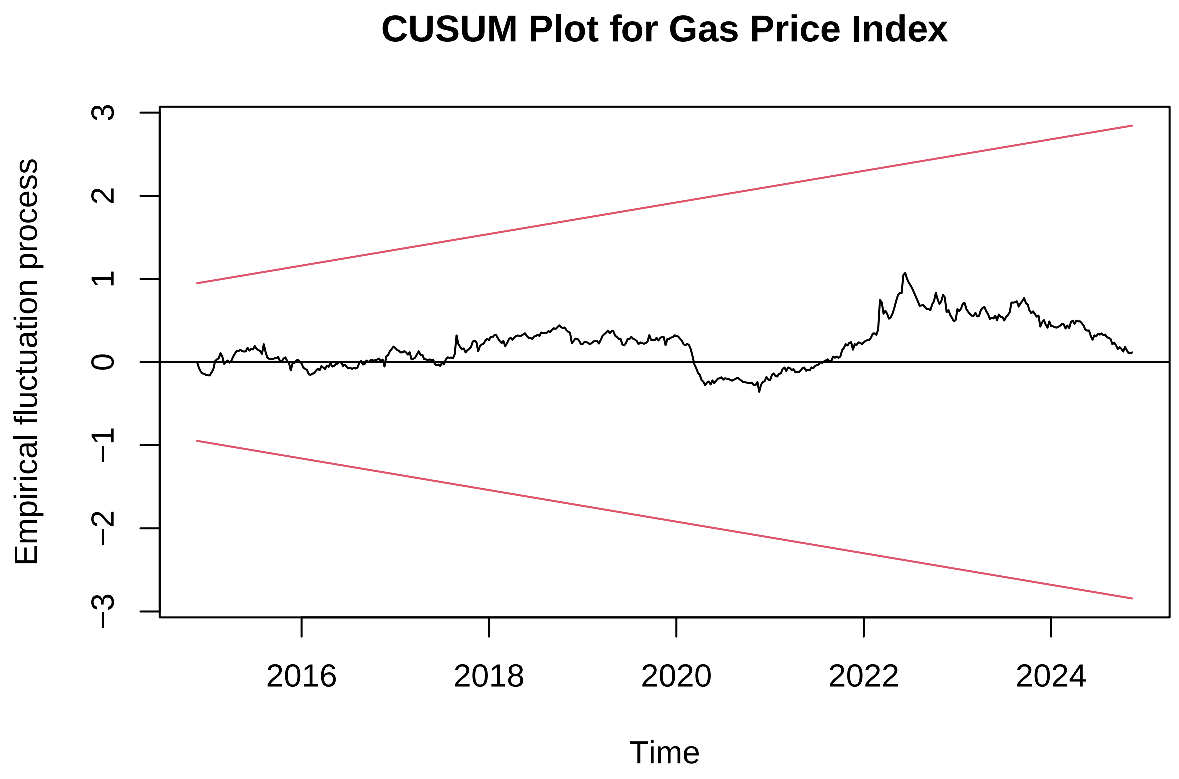


From the ACF and PACF of the residuals, we see that it appears that statistically significant lags appear to persist in some areas of the plot. However, it appears that there is no clear pattern to suggest that a seasonal AR model would fix the statistically significant lags. Other than the few statistically significant lags, the other parts appear to simulate white noise.

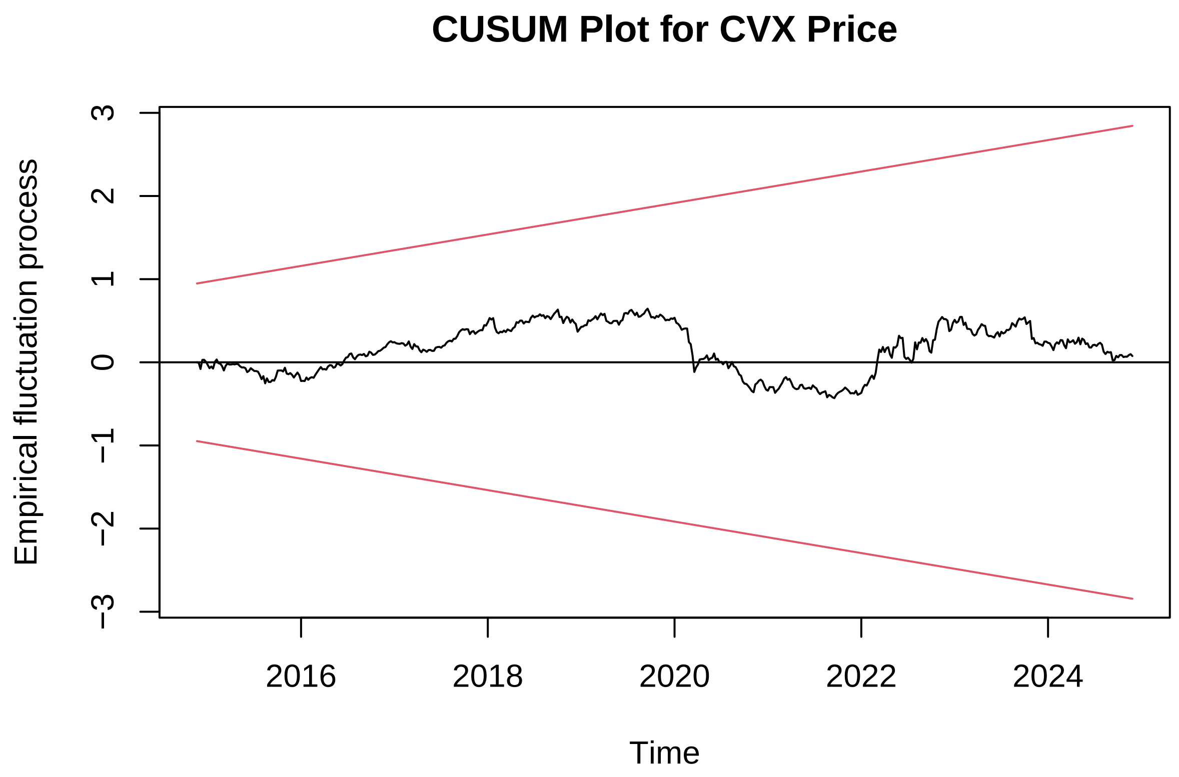
(f) CUSUM Plot

We will proceed to plot the CUSUM as follows:

```r
# Calculate residuals by subtracting the fitted model from the actual data
gas_residuals <- gas_price_index_ts - full_model_gas
gas_efp <- efp(gas_residuals ~ 1, type = "Rec-CUSUM")
plot(gas_efp, main = "CUSUM Plot for Gas Price Index")
```



```r
# (b) CUSUM Plot for CVX Price
# Calculate residuals by subtracting the fitted model from the actual data
cvx_residuals <- cvx_price_ts - full_model_cvx
cvx_efp <- efp(cvx_residuals ~ 1, type = "Rec-CUSUM")
plot(cvx_efp, main = "CUSUM Plot for CVX Price")
```



From the CUSUM Plots above, we see that the residuals stay within the confidence interval bands for both models. This suggests that there are no structural breaks in our model.

(g) Diagnostic Statistics

We will proceed to assess the accuracy and perform a Ljung-Box Test on the residuals of the our full gas model to assess the appropriateness of our model:

```
#Obtain associated error statistics
accuracy(full_model_gas, gas_price_index)
```

```
##                        ME       RMSE        MAE         MPE     MAPE
## Test set -0.0008578556 0.04878779 0.03394842 -0.05608965 1.215091
```

```
#Use Ljung-Box to test for white noise
Box.test(gas_residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  gas_residuals
## X-squared = 0.012366, df = 1, p-value = 0.9115
```

From the above statistics, we see that the model is a pretty good fit to our data, with RMSE of 0.04878779 and MAPE of 1.215091. Interpreting the MAPE, we see that, on average, the model's predictions are off by around 1.2%, which is quite accurate. Furthermore, from the Ljung-Box test performed above, we see that the p-value of 0.9115 is non-statistically significant. This indicates that the residuals simulate white noise, which informs us that most of the dynamics have been captured by the model.

We will proceed to assess the accuracy and perform a Ljung-Box Test on the residuals of the our full gas model to assess the appropriateness of our model:

```
#Obtain associated error statistics
accuracy(full_model_cvx, cvx_price)
```

```
##                      ME     RMSE      MAE        MPE     MAPE
## Test set -0.01733382 4.260634 3.020683 -0.1556359 2.589366
```

```
#Use Ljung-Box to test for white noise
Box.test(cvx_residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  cvx_residuals
## X-squared = 7.7104e-06, df = 1, p-value = 0.9978
```

From the above statistics, we see that the model is a pretty good fit to our data, with RMSE of 4.264305 and MAPE of 2.597912. Interpreting the MAPE, we see that, on average, the model's predictions are off by around 2.6%, which is quite accurate. Furthermore, from the Ljung-Box test performed above, we see that the p-value of 0.9964 is non-statistically significant. This indicates that the residuals simulate white noise, which informs us that most of the dynamics have been captured by the model.

(h) Forecasting with Model

We will proceed to perform a 12-step ahead forecast for both the gas model and cvx model as follows:

```
# Define the number of forecast periods
forecast_horizon <- 12

# Forecasting with the gas price model
gas_forecast <- forecast(full_model_gas, h = forecast_horizon)
print("Gas Price Index Forecast:")
```

```
## [1] "Gas Price Index Forecast:"
```

```
print(gas_forecast)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2024.885       3.143497 3.053875 3.233119 3.006432 3.280562
## 2024.904       2.894159 2.751681 3.036637 2.676257 3.112061
## 2024.923       2.916622 2.725055 3.108189 2.623645 3.209598
## 2024.942       2.861531 2.623135 3.099927 2.496936 3.226126
## 2024.962       2.824138 2.540879 3.107397 2.390931 3.257345
## 2024.981       2.814756 2.488512 3.141001 2.315809 3.313704
## 2025.000       2.822516 2.455103 3.189930 2.260606 3.384427
## 2025.019       2.839272 2.432434 3.246109 2.217067 3.461476
## 2025.038       2.857490 2.412887 3.302092 2.177529 3.537450
## 2025.058       2.877655 2.396852 3.358458 2.142330 3.612979
## 2025.077       2.902762 2.387223 3.418301 2.114314 3.691211
## 2025.096       2.913141 2.364231 3.462051 2.073655 3.752626
```

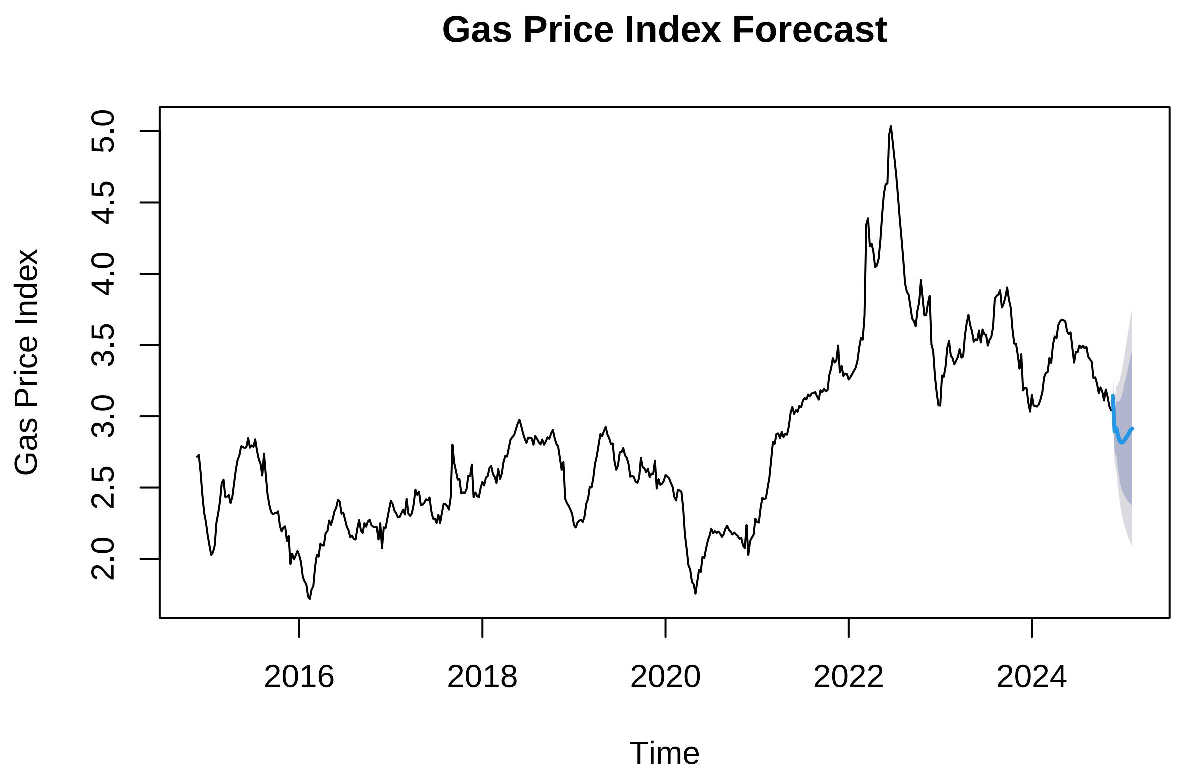
```
# Forecasting with the Chevron (CVX) stock price model
cvx_forecast <- forecast(full_model_cvx, h = forecast_horizon)
print("Chevron (CVX) Stock Price Forecast:")
```

```
## [1] "Chevron (CVX) Stock Price Forecast:"
```

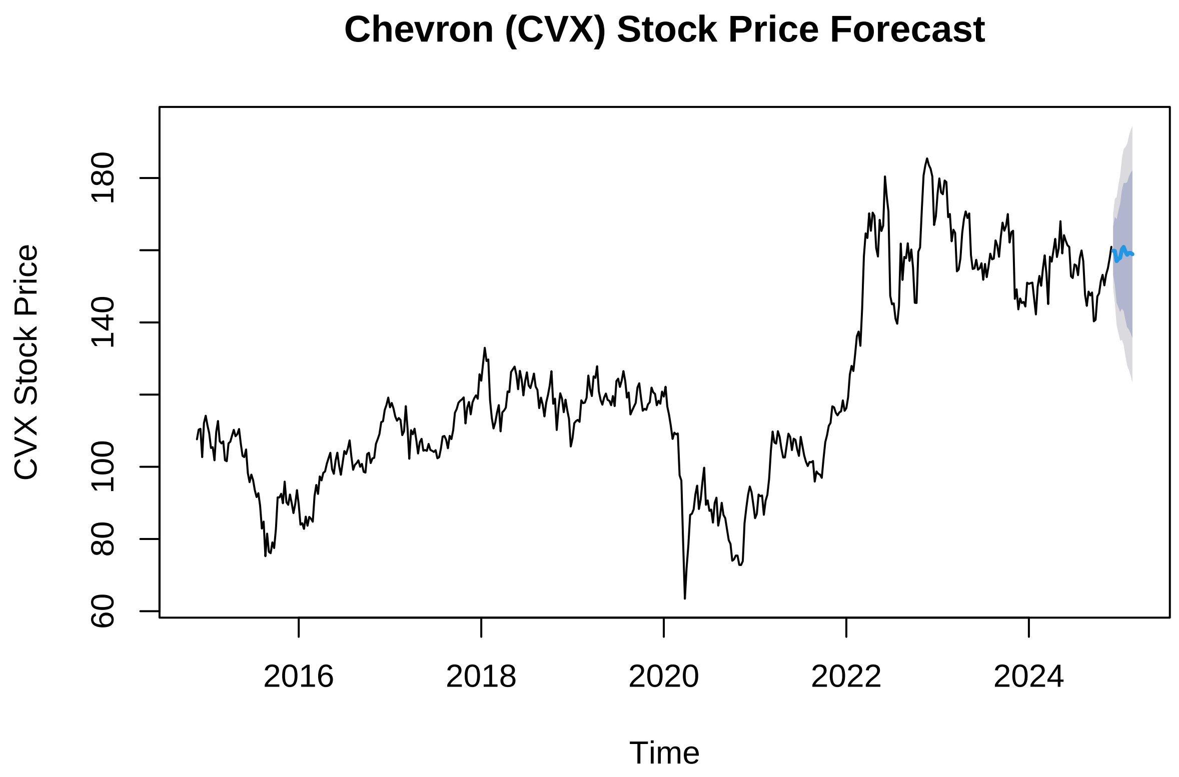
```
print(cvx_forecast)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2024.923       159.8702 153.1781 166.5624 149.6355 170.1050
## 2024.942       159.7922 150.3261 169.2584 145.3150 174.2695
## 2024.962       156.9905 145.3939 168.5870 139.2550 174.7259
## 2024.981       157.5707 144.1766 170.9647 137.0862 178.0551
## 2025.000       157.8690 142.8901 172.8480 134.9607 180.7774
## 2025.019       160.2314 143.8184 176.6444 135.1299 185.3330
## 2025.038       160.9071 143.1743 178.6399 133.7871 188.0271
## 2025.058       159.6475 140.6852 178.6098 130.6472 188.6478
## 2025.077       158.7451 138.6271 178.8630 127.9773 189.5128
## 2025.096       159.2303 138.0185 180.4422 126.7896 191.6711
## 2025.115       159.2347 136.9815 181.4879 125.2014 193.2680
## 2025.135       158.8980 135.6491 182.1470 123.3418 194.4543
```

```
# Plot gas price forecast
plot(gas_forecast, main = "Gas Price Index Forecast",
     ylab = "Gas Price Index", xlab = "Time")
```



```r
# Plot Chevron (CVX) stock price forecast
plot(cvx_forecast, main = "Chevron (CVX) Stock Price Forecast",
     ylab = "CVX Stock Price", xlab = "Time")
```



(i) Comparison to auto.arima() (Sia)

(j) Combining the Model with auto.arima() (Sia)

(k) The VAR Model (Jiaxuan)

From the model summary, we can find that for the regression towards gas, the lag 1 and 2 of CVX prices has a statistically high significance, as well as the log1 of gas price itself. The R-square is around 0.4. This indicates that gas price is explained reasonably well. However, none of the lagged terms are significant towards the prediction of CVX price and the R-square is around 0.003, so the model is not fitting well on CVX price, this is confirmed by our plot, where the fitted ice has similar pattern of the actual price, whereas the fitted CVX doesn't seem to have similar patterns

=======NOTES AND FEEDBACK START===========

Feedback 1:

Can you elaborate more on the code in this section outside of the comments? It has to be clear to the reader that we're differencing in order to make the data stationary and using the adf to check if the resulting data is indeed stationary; using the VARselection to determine the order of the model; and plotting the data. The format will look something like this:

"We will proceed to difference our data to make it stationary and perform an ADF test to statistically ensure that the data is indeed stationary:

[INSERT R MARKDOWN HERE]

From the ADF test we see that the . . . "

Essentially, what you're trying to do is explain what you're about to do, and then perform the action in an R markdown. It has to be clear to the reader what you're about to do in a sequential and logical fasion. Do the same when fitting the VAR Model and plotting the VAR Model

Feedback 2:

Can you fix the axis and legend. It appears that, for one of the plots, the legend covers the plot.

Feedback 3:

Move the interpretation of the model summary just below the call of the summary. Also, don't use R squared since it doesn't take into account the number of parameters. Interpret the fit using AIC, BIC, and Adjusted R Squared.

=======NOTES AND FEEDBACK END===========

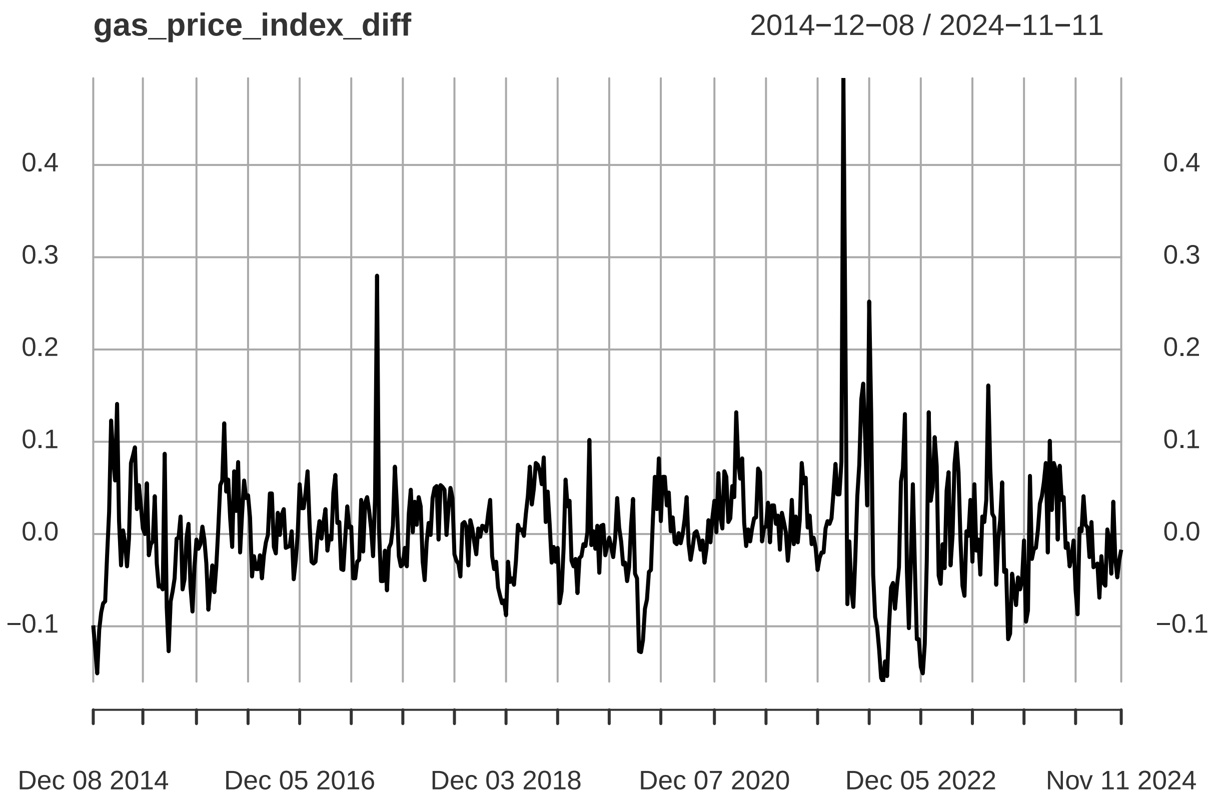
```
# Load the percentage change of the gas price index for the past decade (weekly data)
gas_price_index_diff <- na.omit(diff(gas_price_index))
adf_test_result_gas <- adf.test(gas_price_index_diff)
```

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

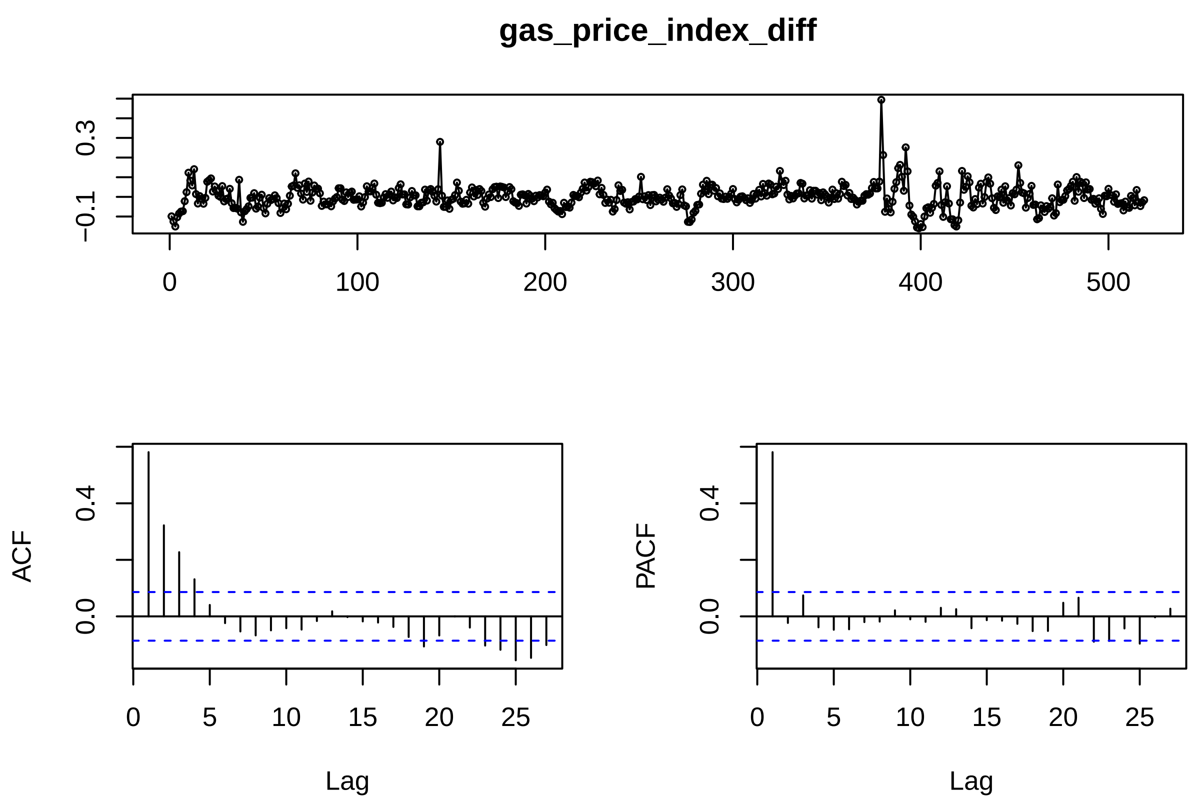
```
adf_test_result_gas
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  gas_price_index_diff
## Dickey-Fuller = -7.1008, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
plot(gas_price_index_diff)
```



```r
tsdisplay(gas_price_index_diff)
```



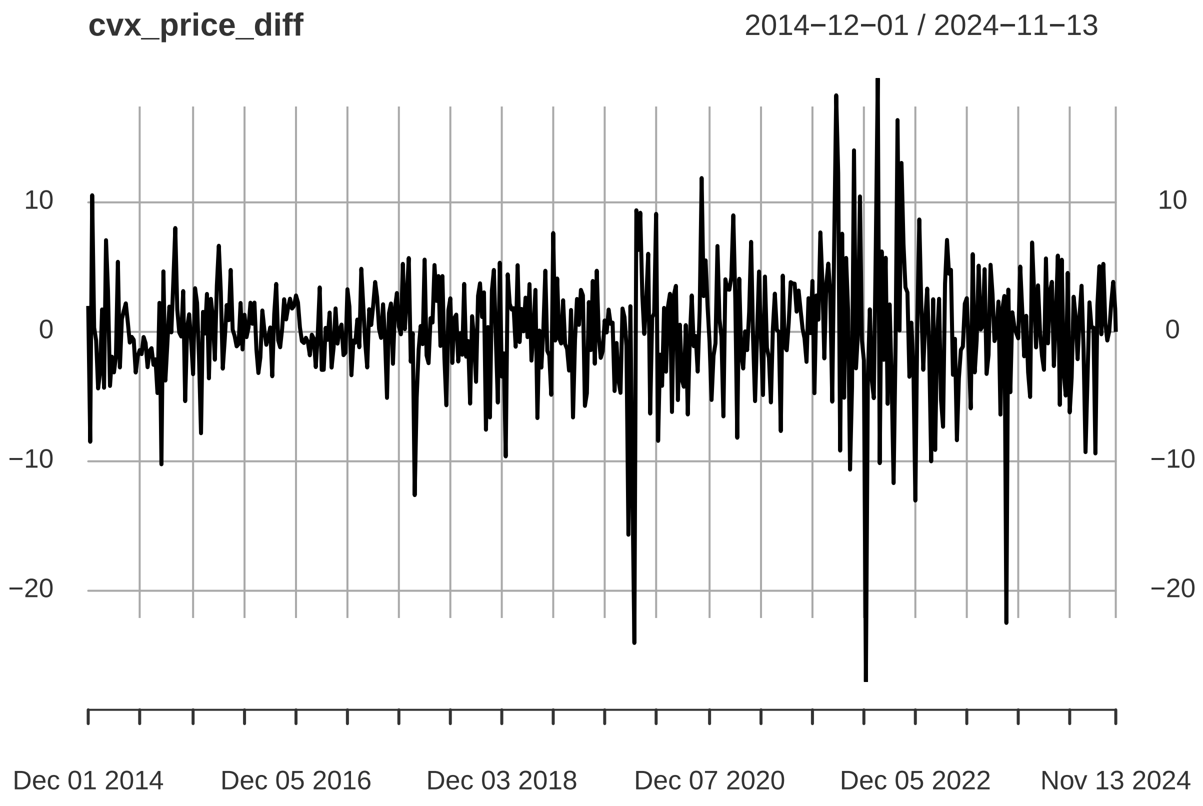
```
# Load the percentage change of the TAN (solar ETF) closing stock price for the past decade (weekly dat
cvx_price_diff  <- na.omit(diff(cvx_price))
adf_test_result_cvx  <- adf.test(cvx_price_diff)
```

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

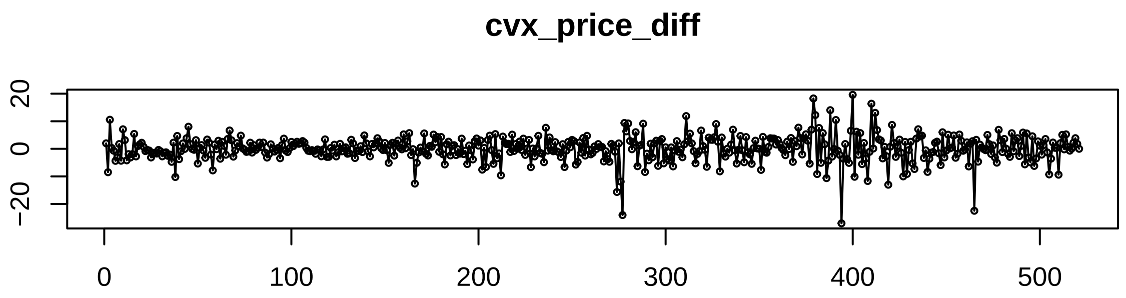
```
adf_test_result_cvx
```

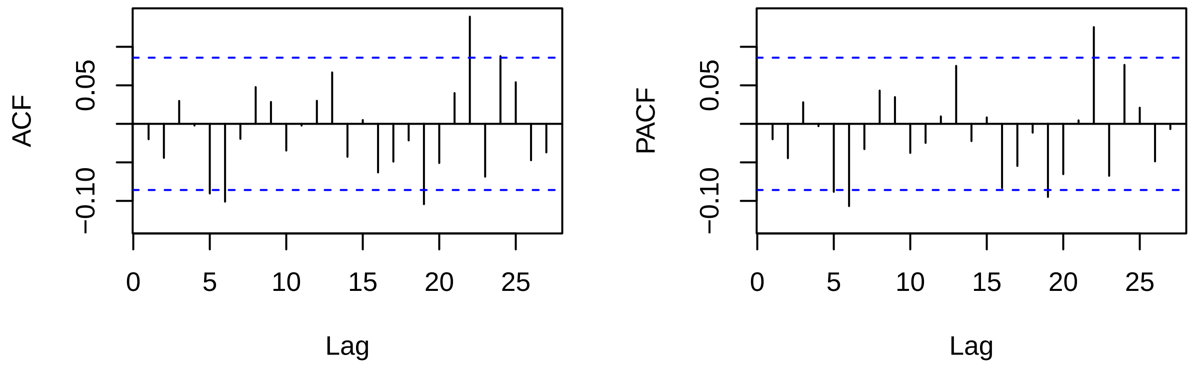
```
##
##  Augmented Dickey-Fuller Test
##
## data:  cvx_price_diff
## Dickey-Fuller = -7.9719, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
plot(cvx_price_diff)
```



```r
tsdisplay(cvx_price_diff)
```



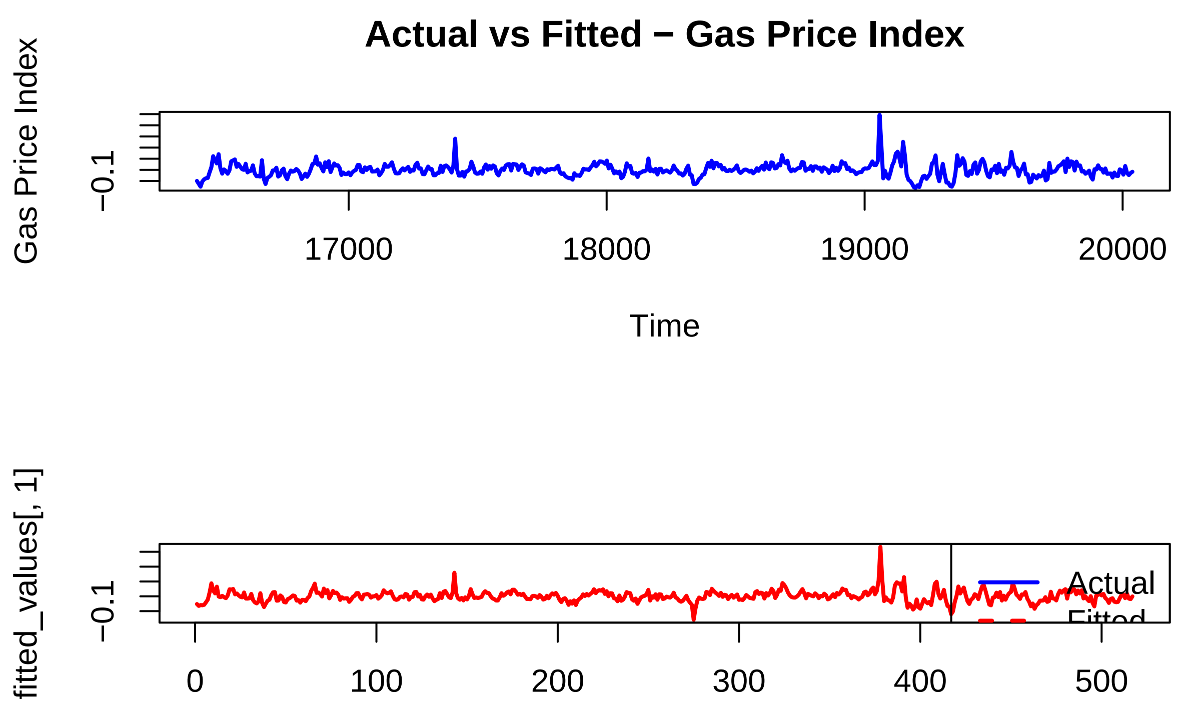


```
# Combine the two time series into one dataset
data_diff <- cbind(gas_price_index_diff, cvx_price_diff)
data_diff <- na.omit(data_diff)

lag_selection <- VARselect(data_diff, lag.max = 10, type = "const")
lag_selection
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      2      1      3
##
## $criteria
##                  1           2           3           4           5           6
## AIC(n) -3.08764342 -3.10075445 -3.10953770 -3.09398439 -3.10011808 -3.0939695
## HQ(n)  -3.06808108 -3.06815055 -3.06389224 -3.03529737 -3.02838950 -3.0091993
## SC(n)  -3.03775209 -3.01760223 -2.99312459 -2.94431039 -2.91718319 -2.8777737
## FPE(n)  0.04560932  0.04501528  0.04462173  0.04532135  0.04504449  0.0453227
##                  7           8           9          10
## AIC(n) -3.08130749 -3.07087512 -3.06476931 -3.05172343
## HQ(n)  -2.98349580 -2.96002187 -2.94087450 -2.91478706
## SC(n)  -2.83185083 -2.78815757 -2.74879087 -2.70248410
## FPE(n)  0.04590077  0.04638285  0.04666784  0.04728181
```

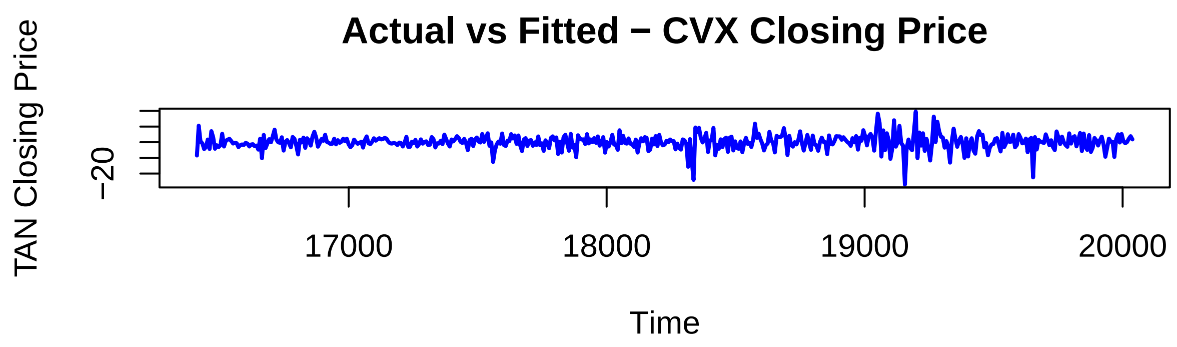
```
#Due to the result, there seems to be a  on controversy on choice of p, we choose
#p = 2 which is overall smallest
```

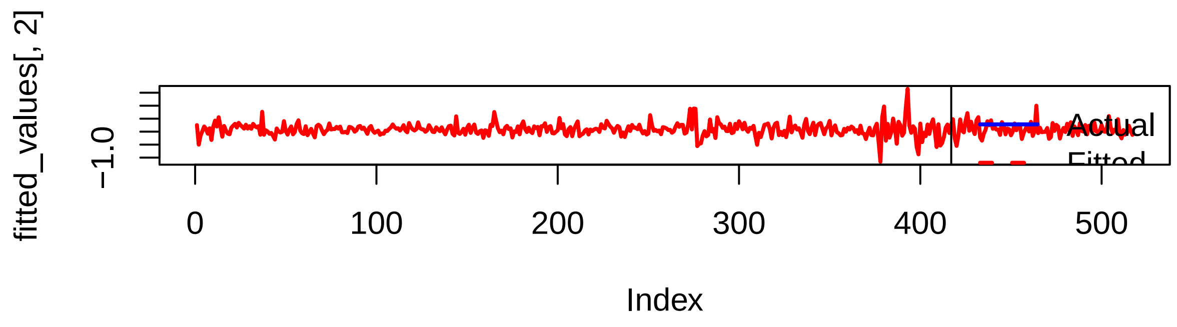
```r
# Fit our data to var
var_model <- VAR(data_diff, p = 2, type = "const")

# Extract fitted values from the VAR model
fitted_values <- fitted(var_model)

# Convert actual differenced data into a time series object for plotting consistency
actual_values <- ts(data_diff, start = start(data_diff),
                    frequency = frequency(data_diff))
# Set up plotting area
par(mfrow = c(2, 1))  # 2 rows, 1 column layout for plotting

# Plot for the first time series (e.g., gas price index)
plot(actual_values[, 1], type = "l", col = "blue", lwd = 2,
     main = "Actual vs Fitted - Gas Price Index",
     ylab = "Gas Price Index", xlab = "Time")
plot (fitted_values[, 1],type = 'l', col = "red", lwd = 2)
legend("topright", legend = c("Actual", "Fitted"), col = c("blue", "red"),
       lty = c(1, 2), lwd = 2)
```



```r
# Plot for the second time series (e.g., TAN closing price)
plot(actual_values[, 2], type = "l", col = "blue", lwd = 2,
     main = "Actual vs Fitted - CVX Closing Price",
     ylab = "TAN Closing Price", xlab = "Time")
plot(fitted_values[, 2], type = 'l', col = "red", lwd = 2)
```

```r
legend("topright", legend = c("Actual", "Fitted"), col = c("blue", "red"),
       lty = c(1, 2), lwd = 2)
```





```r
# Reset plotting area
par(mfrow = c(1, 1))
```

[ELABORATE WHAT YOU WILL TO DO HERE]

```r
summary(var_model)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: GASREGW, CVX.Close
## Deterministic variables: const
## Sample size: 517
## Log Likelihood: -655.958
## Roots of the characteristic polynomial:
## 0.5816 0.2587 0.2587 0.1051
## Call:
## VAR(y = data_diff, p = 2, type = "const")
##
##
## Estimation results for equation GASREGW:
```

```
## ========================================
## GASREGW = GASREGW.l1 + CVX.Close.l1 + GASREGW.l2 + CVX.Close.l2 + const
##
##                Estimate Std. Error t value Pr(>|t|)
## GASREGW.l1   0.5418711  0.0436215  12.422  < 2e-16 ***
## CVX.Close.l1 0.0028750  0.0004408   6.522 1.67e-10 ***
## GASREGW.l2   0.0088539  0.0423487   0.209 0.834476
## CVX.Close.l2 0.0016368  0.0004571   3.581 0.000375 ***
## const        0.0001121  0.0020216   0.055 0.955810
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.04594 on 512 degrees of freedom
## Multiple R-Squared: 0.4001,  Adjusted R-squared: 0.3954
## F-statistic: 85.37 on 4 and 512 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation CVX.Close:
## ==========================================
## CVX.Close = GASREGW.l1 + CVX.Close.l1 + GASREGW.l2 + CVX.Close.l2 + const
##
##              Estimate Std. Error t value Pr(>|t|)
## GASREGW.l1   -1.01597    4.34787  -0.234    0.815
## CVX.Close.l1 -0.01075    0.04394  -0.245    0.807
## GASREGW.l2    2.26516    4.22101   0.537    0.592
## CVX.Close.l2 -0.04357    0.04556  -0.956    0.339
## const         0.09263    0.20149   0.460    0.646
##
##
## Residual standard error: 4.579 on 512 degrees of freedom
## Multiple R-Squared: 0.002863,    Adjusted R-squared: -0.004927
## F-statistic: 0.3675 on 4 and 512 DF,  p-value: 0.8318
##
##
##
## Covariance matrix of residuals:
##            GASREGW CVX.Close
## GASREGW   0.002111  0.007028
## CVX.Close 0.007028 20.969504
##
## Correlation matrix of residuals:
##           GASREGW CVX.Close
## GASREGW   1.00000   0.03341
## CVX.Close 0.03341   1.00000
```

(l) Impulse Response Functions (Jiaxuan)

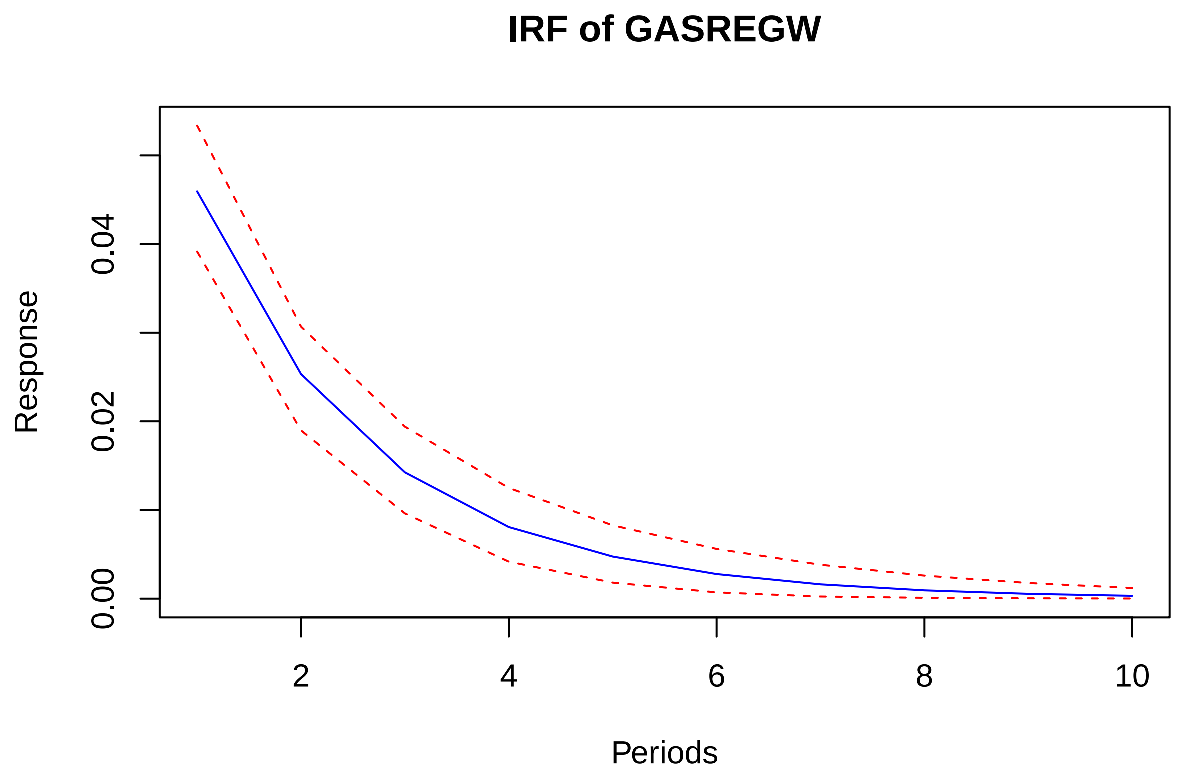
[ELABORATE WHAT YOU WILL TO DO HERE]

```
# Compute impulse response functions for 10 periods ahead
irf_result <- irf(var_model, n.ahead = 10)
# Display the IRF results
```

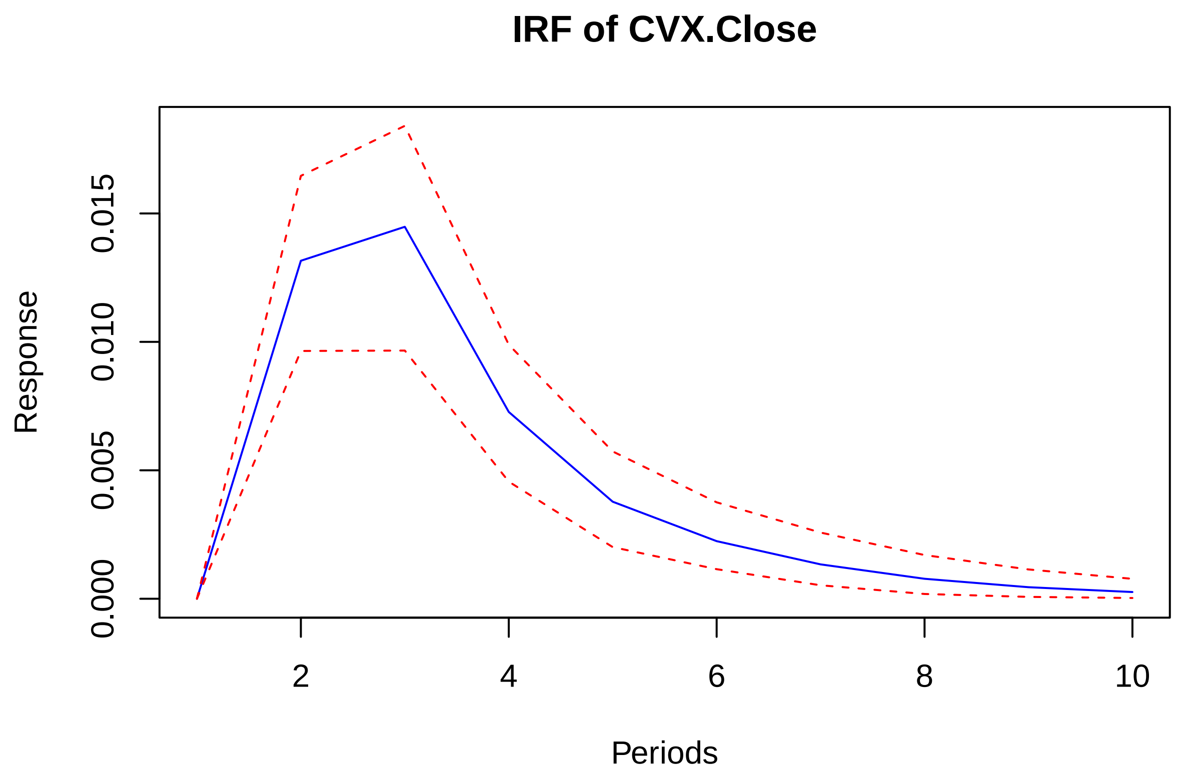
```
irf_result
```

```
##
## Impulse response coefficients
## $GASREGW
##            GASREGW     CVX.Close
##  [1,] 0.0459428497  0.1529742789
##  [2,] 0.0253349025 -0.0483210604
##  [3,] 0.0142464956  0.0721820377
##  [4,] 0.0080725053  0.0442431979
##  [5,] 0.0047457423  0.0204482850
##  [6,] 0.0027742607  0.0113163033
##  [7,] 0.0016113145  0.0069186318
##  [8,] 0.0009361016  0.0040796231
##  [9,] 0.0005445663  0.0023535044
## [10,] 0.0003168168  0.0013640904
## [11,] 0.0001842694  0.0007944373
##
## $CVX.Close
##            GASREGW     CVX.Close
##  [1,] 0.0000000000  4.5766912794
##  [2,] 0.0131578907 -0.0491968973
##  [3,] 0.0144797067 -0.2122649194
##  [4,] 0.0072718494  0.0195191917
##  [5,] 0.0037772825  0.0344503106
##  [6,] 0.0022421779  0.0114134330
##  [7,] 0.0013376178  0.0046543255
##  [8,] 0.0007767314  0.0031725464
##  [9,] 0.0004494708  0.0020038691
## [10,] 0.0002613863  0.0011429893
## [11,] 0.0001521833  0.0006529584
##
##
## Lower Band, CI= 0.95
## $GASREGW
##            GASREGW    CVX.Close
##  [1,] 3.915317e-02 -0.395138223
##  [2,] 1.897607e-02 -0.412342900
##  [3,] 9.614230e-03 -0.207299775
##  [4,] 4.170077e-03 -0.100490592
##  [5,] 1.806983e-03 -0.043029963
##  [6,] 7.063611e-04 -0.022525123
##  [7,] 2.409727e-04 -0.012378349
##  [8,] 9.577232e-05 -0.005890219
##  [9,] 3.842026e-05 -0.003230323
## [10,] 1.544630e-05 -0.002014219
## [11,] 6.221796e-06 -0.001251134
##
## $CVX.Close
##            GASREGW    CVX.Close
##  [1,] 0.000000e+00  4.040364966
##  [2,] 9.645607e-03 -0.392027039
##  [3,] 9.661976e-03 -0.581671706
##  [4,] 4.559373e-03 -0.110664901
##  [5,] 2.010036e-03 -0.054388697
##  [6,] 1.151323e-03 -0.027455476
##  [7,] 5.240468e-04 -0.012558605
##  [8,] 1.857589e-04 -0.006567351
##  [9,] 7.217027e-05 -0.003340139
## [10,] 2.874844e-05 -0.002009069
## [11,] 1.152017e-05 -0.001192553
##
##
## Upper Band, CI= 0.95
## $GASREGW
##            GASREGW  CVX.Close
##  [1,] 0.0533615137 0.84104601
##  [2,] 0.0306877600 0.28697155
##  [3,] 0.0193983404 0.37090898
##  [4,] 0.0124871261 0.21832712
##  [5,] 0.0082695088 0.10840989
##  [6,] 0.0056075714 0.06863936
##  [7,] 0.0038218966 0.05167914
##  [8,] 0.0025950041 0.03454642
##  [9,] 0.0017622240 0.02336455
## [10,] 0.0011996675 0.01642990
## [11,] 0.0008209898 0.01158482
##
## $CVX.Close
##            GASREGW   CVX.Close
##  [1,] 0.0000000000 5.024256844
##  [2,] 0.0164631384 0.380732196
##  [3,] 0.0184093722 0.110308641
##  [4,] 0.0098884470 0.140160559
##  [5,] 0.0057344248 0.152563657
##  [6,] 0.0037532970 0.062045322
##  [7,] 0.0025758502 0.034778183
##  [8,] 0.0017015412 0.024829208
##  [9,] 0.0011410055 0.018150899
## [10,] 0.0007748168 0.011874714
## [11,] 0.0005251899 0.008032836
```

```r
# Plot impulse response for "GASREGW"
response_gas <- irf_result$irf[["GASREGW"]][1:10]
lower_gas <- irf_result$Lower[["GASREGW"]][1:10]
upper_gas <- irf_result$Upper[["GASREGW"]][1:10]

plot(response_gas, type = "l", col = "blue",
     ylim = range(c(lower_gas, upper_gas)),
     ylab = "Response", xlab = "Periods", main = "IRF of GASREGW")
lines(lower_gas, col = "red", lty = 2)
lines(upper_gas, col = "red", lty = 2)
```



```r
# Plot impulse response for "CVX.Close"
response_cvx <- irf_result$irf[["CVX.Close"]][1:10]
lower_cvx <- irf_result$Lower[["CVX.Close"]][1:10]
upper_cvx <- irf_result$Upper[["CVX.Close"]][1:10]

plot(response_cvx, type = "l", col = "blue", ylim = range(c(lower_cvx,
                                                            upper_cvx)),
     ylab = "Response", xlab = "Periods", main = "IRF of CVX.Close")
lines(lower_cvx, col = "red", lty = 2)
lines(upper_cvx, col = "red", lty = 2)
```



[ELABORATE WHAT YOU DID HERE]

(m) Granger-Causality (Sia)

(n) Forecasting with VAR Model (Jiaxuan)

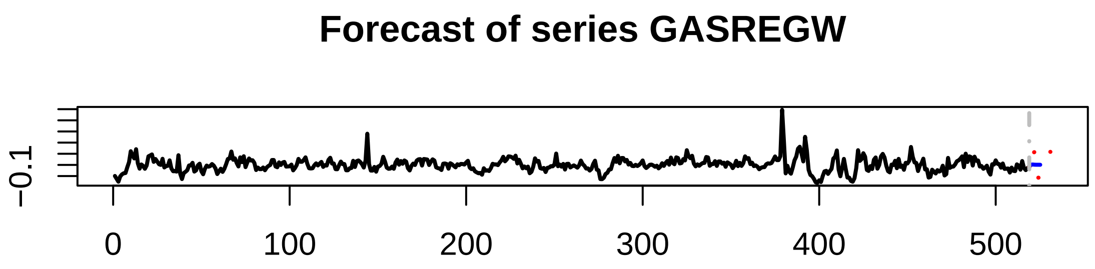
=======NOTES AND FEEDBACK START===========

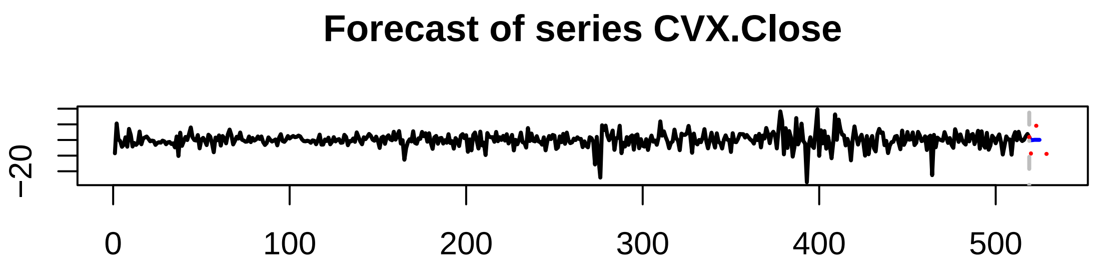
It's difficult to see the actual forecast. Can you include the original plot? Then, can you also create another plot with the window narrowed to the recent 20 data points so it is easier to see what is happening?

=======NOTES AND FEEDBACK END===========

We can see that the model predicts both gas price and CVX price as a series almost the same as the last true value observed, unlike the gas price predicted by ARIMA which has a downward trend.

```r
# Generate a 12-step-ahead forecast with error bands
forecast_var <- predict(var_model, n.ahead = 12, ci = 0.95)
# Plot the forecast with error bands
plot(forecast_var,type = 'l',lwd = 2)
```





```r
# Extract the forecast values for GASREGW and CVX.Close
gas_forecast <- forecast_var$fcst$GASREGW
cvx_forecast <- forecast_var$fcst$CVX.Close
```