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"
```

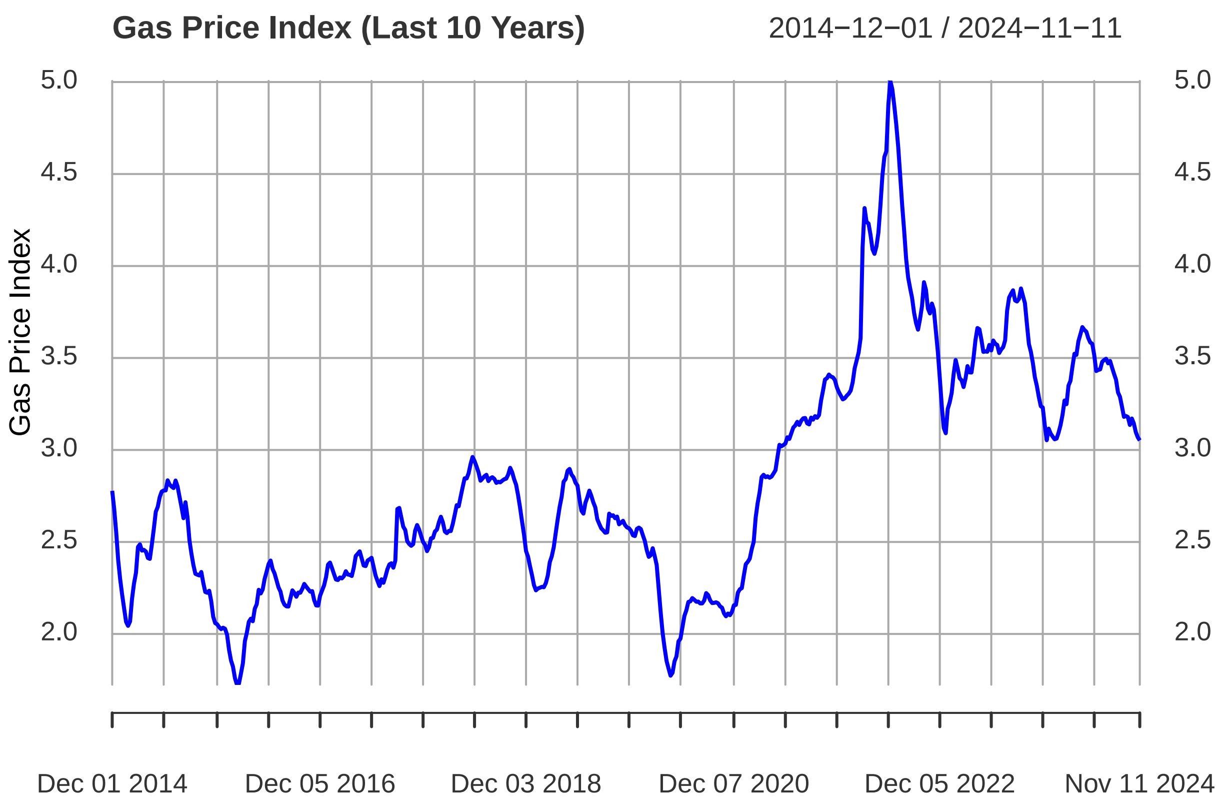
```r
gas_price_index <- na.omit(GASREGW)
# Load the Chevron stock price from the last ten years
getSymbols("CVX", src = "yahoo", periodicity = "weekly", from = start_date)
```

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

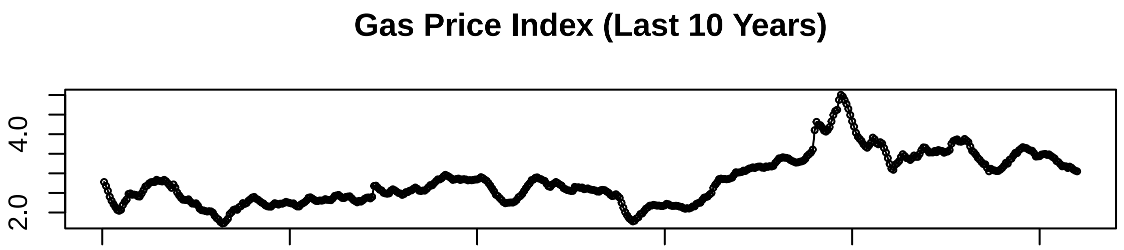
```
cvx_price  <- na.omit(Cl(CVX))
```

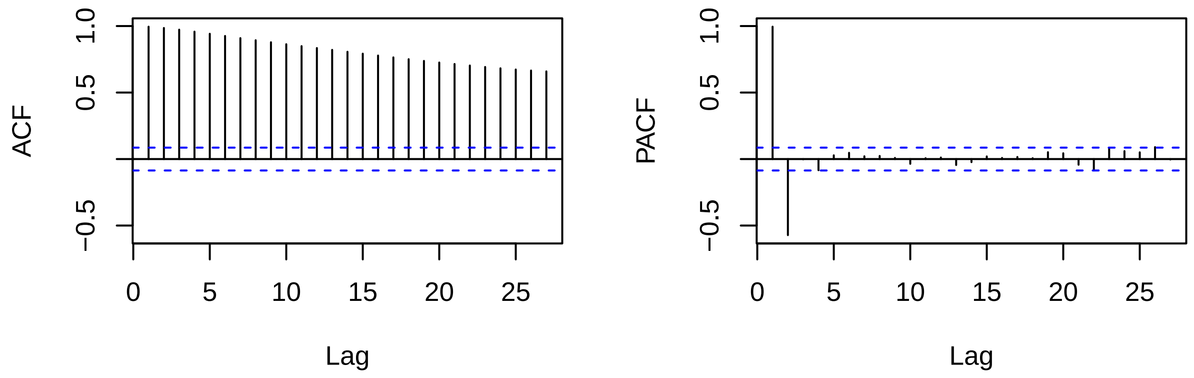
We will proceed to plot the gas price index data, along with the respective ACF and PACF plots as follows:

```
# 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)")
```

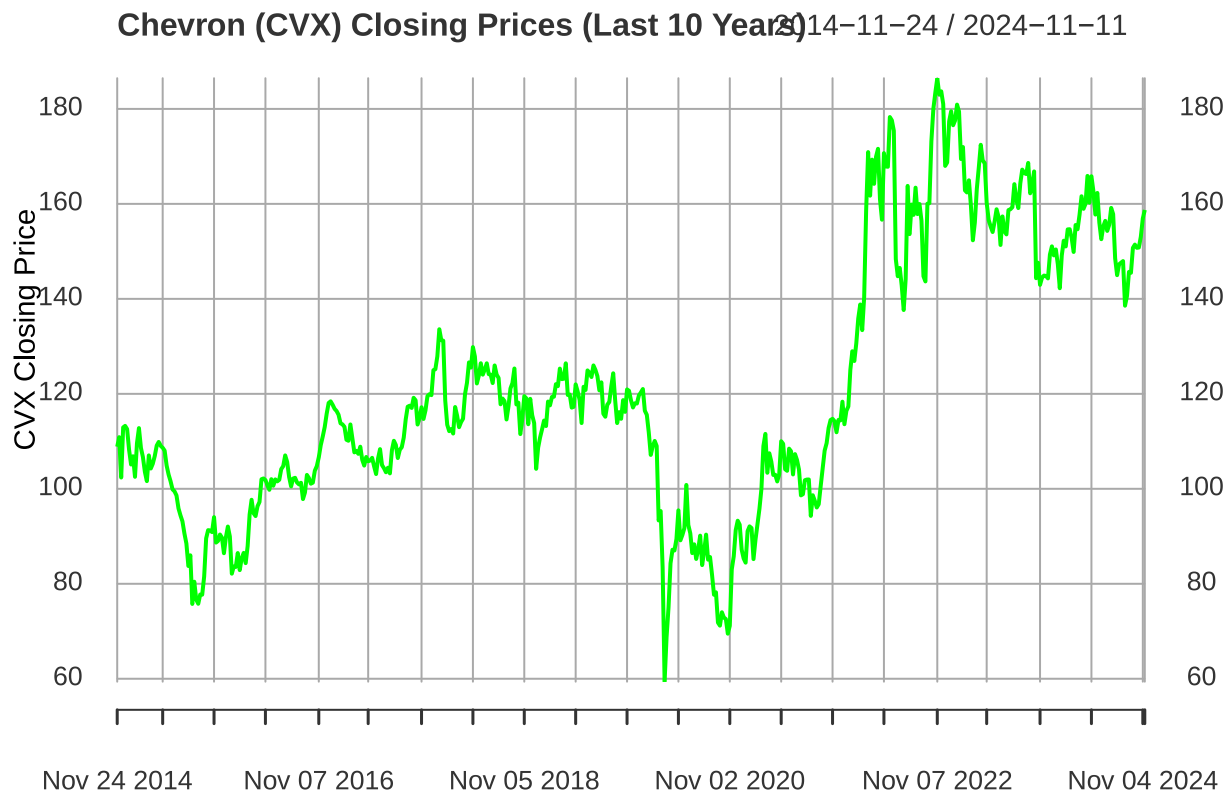




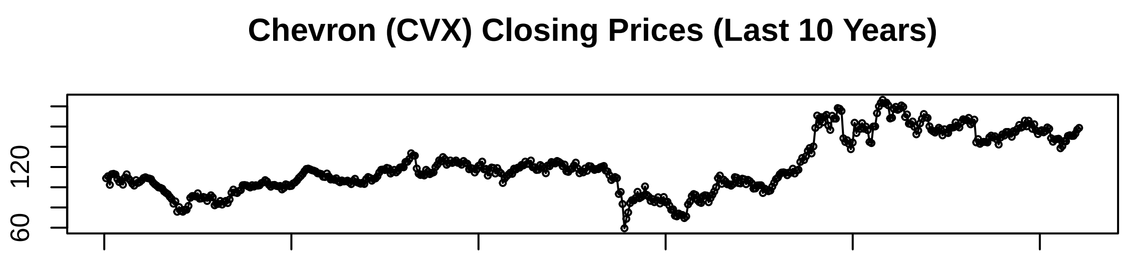
Analyzing the respective ACF and PACF plots, we see that there is a slow decay in the ACF and two statistically significant spikes in the first and second lag. Hence, this suggests to us that we need to fit an AR(2) model.

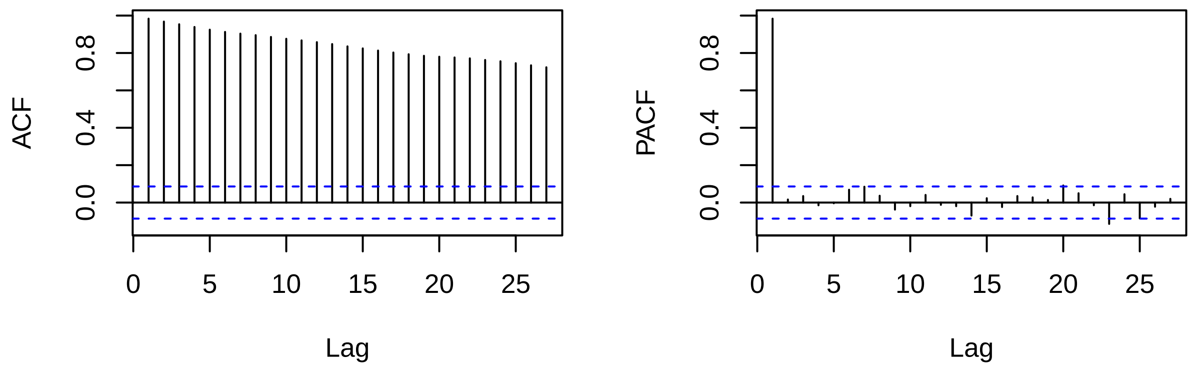
We will proceed to plot the Chevron price, along with the respective ACF and PACF plots as follows:

```
# 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)")
```



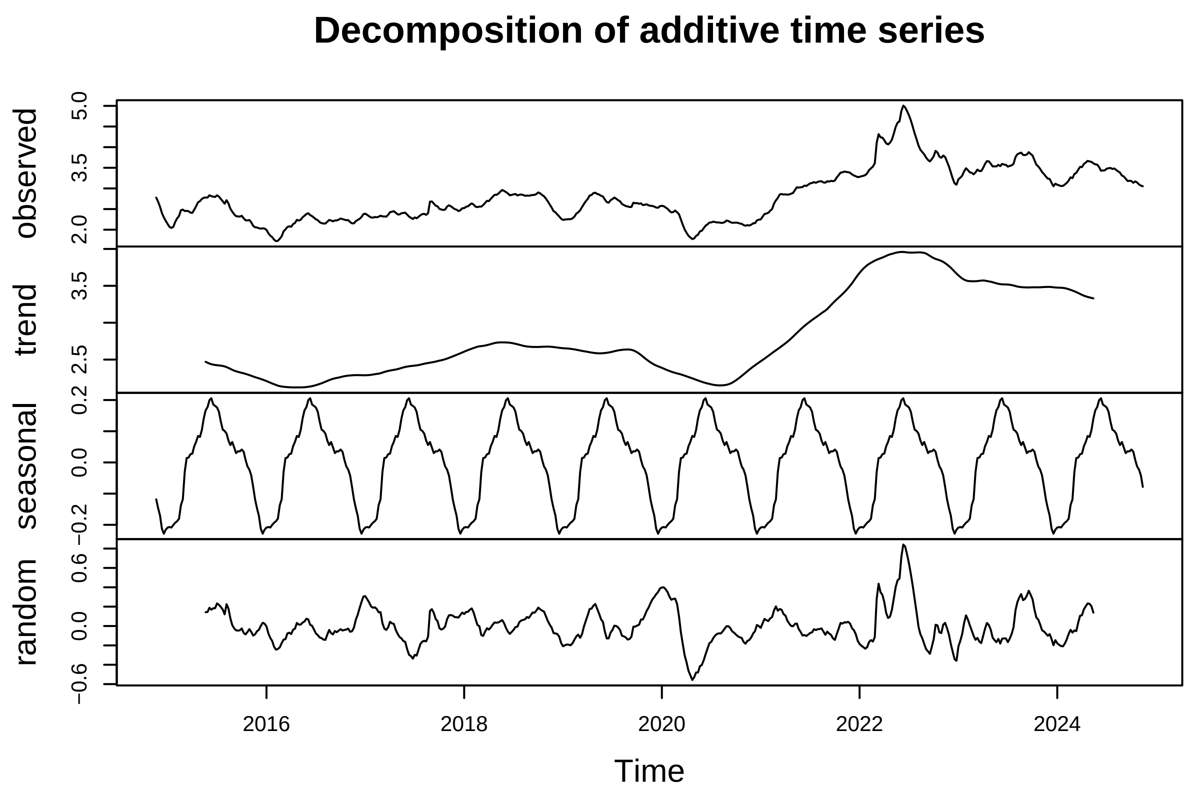


Analyzing the respective ACF and PACF plots, we see that there is a slow decay in the ACF and a single statistically significant spike in the first lag. Hence, this suggests to us that we need to fit an AR(1) model.

(b) STL Decomposition

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

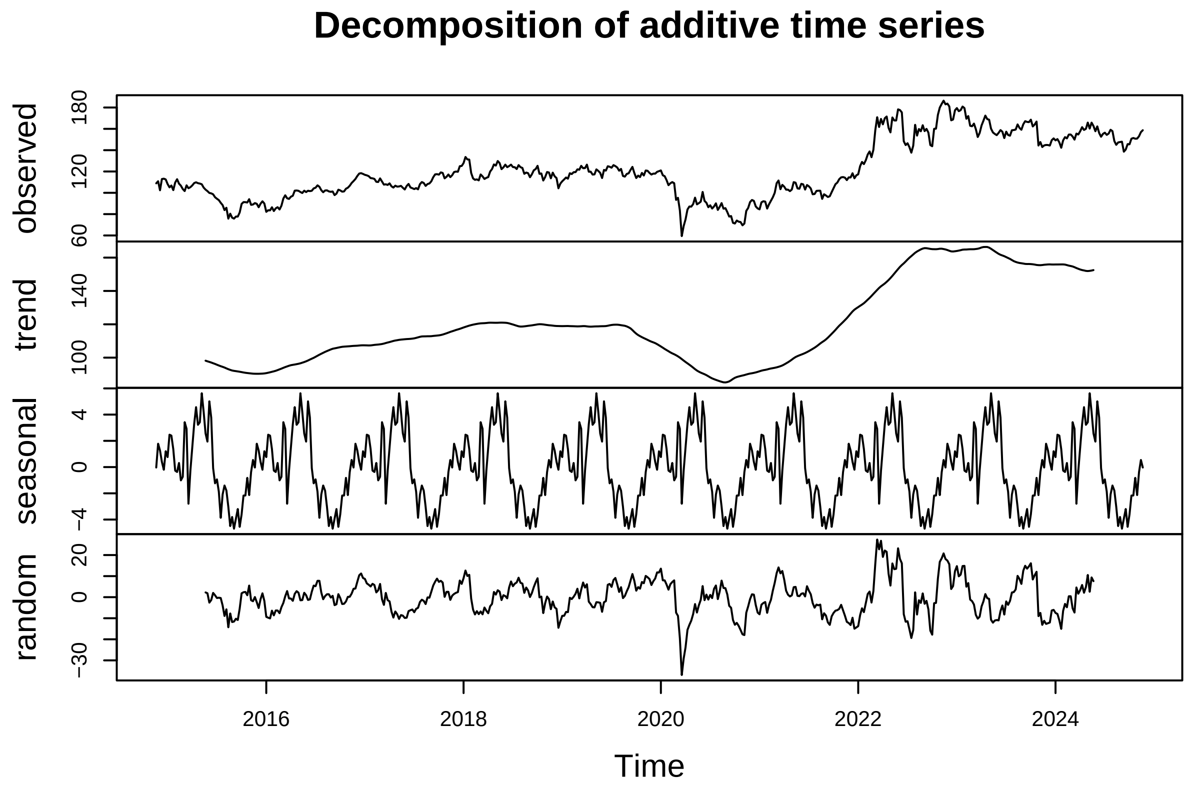
```
#Create the associated ts object
gas_price_index_ts<- ts(gas_price_index$GASREGW,
                              start = c(2014, 47), frequency = 52)
gas_price_index_decomp <- decompose(gas_price_index_ts)
plot(gas_price_index_decomp)
```



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

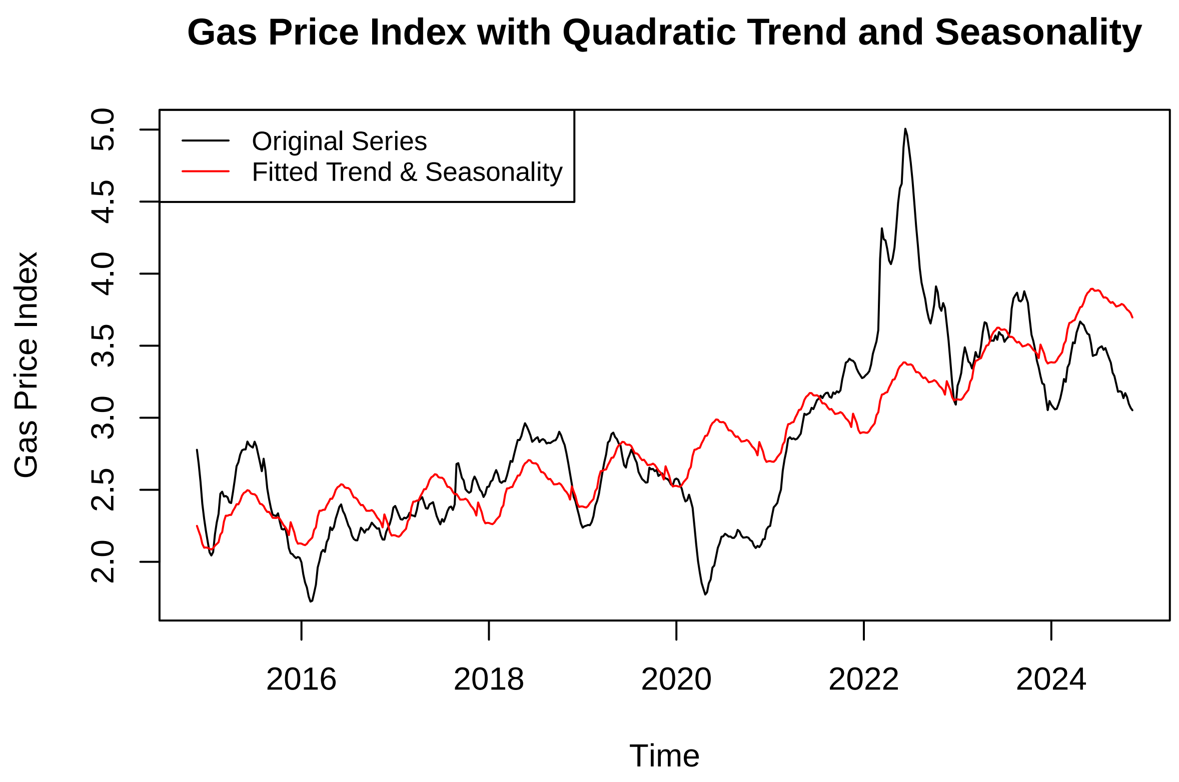
We will propose the following model for the trend and seasonality of the gas price index:

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

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

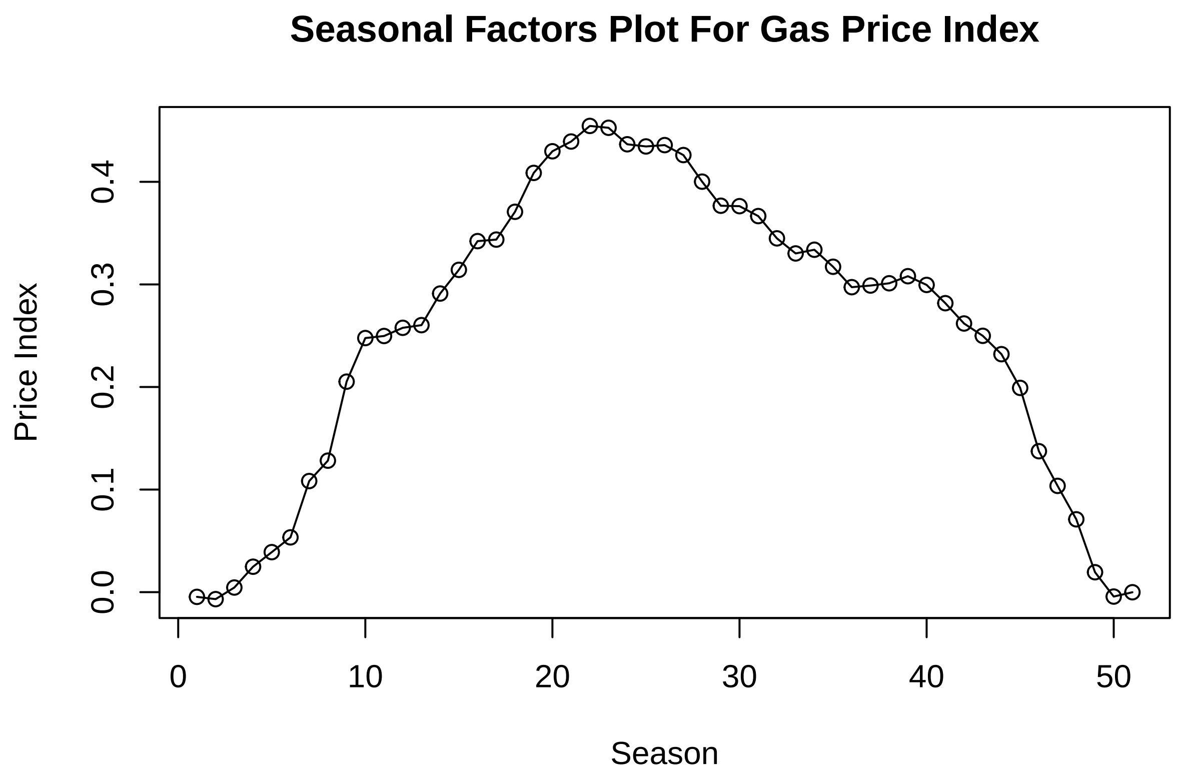
```
#Create the fit model object
fit_gas <- tslm(gas_price_index_ts ~ trend+ I(trend^2)+season)
#Plot the original data
plot(gas_price_index_ts,
     main = "Gas Price Index with Quadratic Trend and Seasonality",
     ylab = "Gas Price Index",
     xlab = "Time")
#Plot the fitted model
```

```
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:

```
# 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")
```



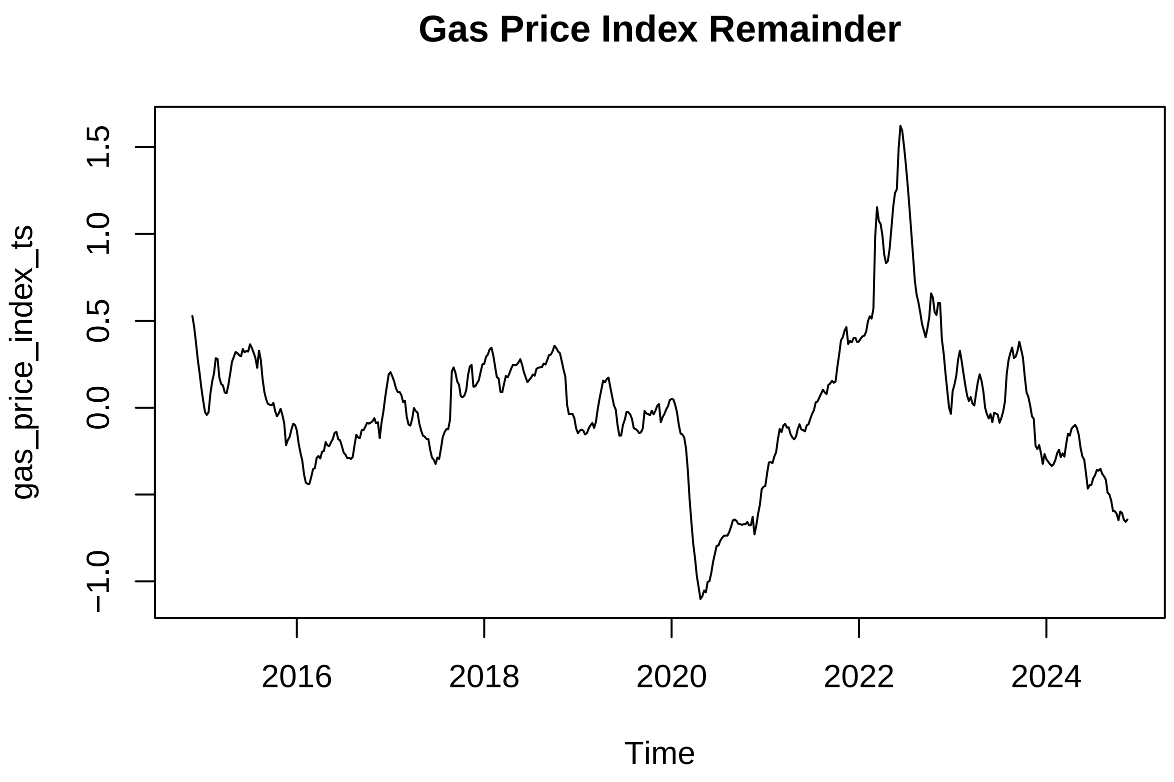
```r
#Summary of the coefficients
summary(fit_seasonal_gas)
```

```
##
## 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
```

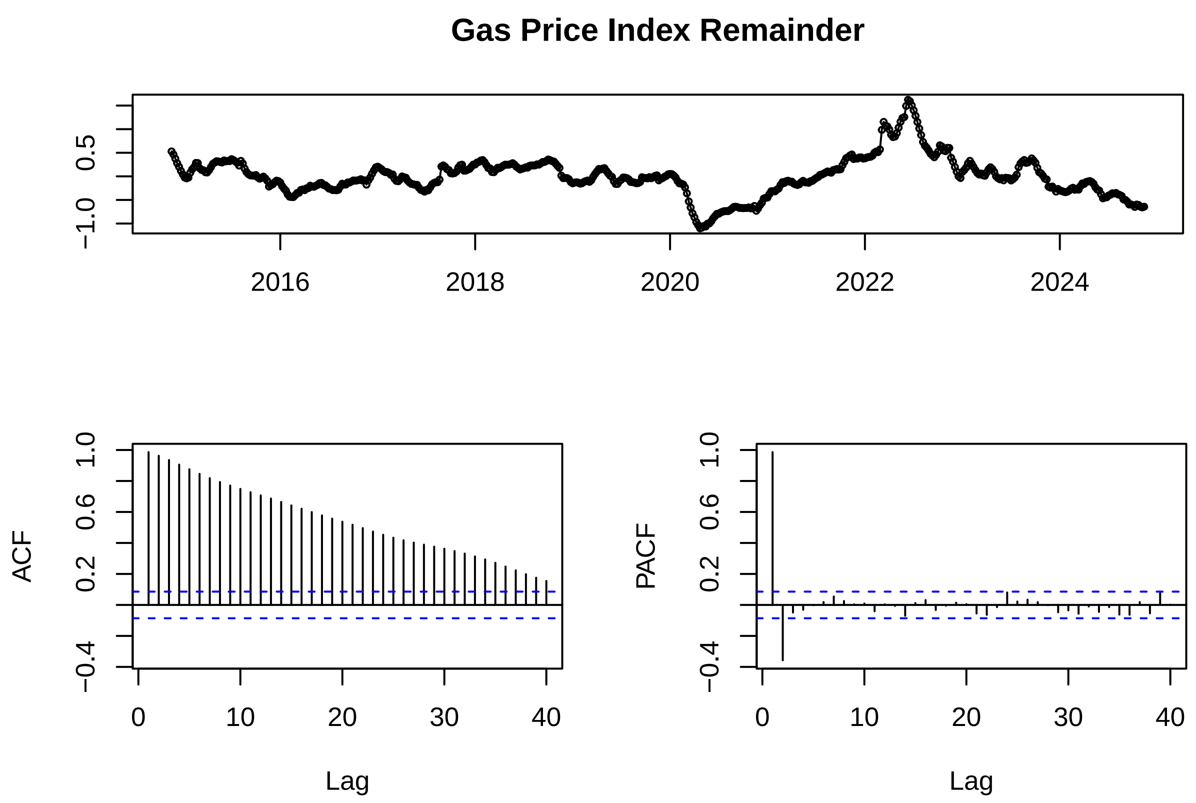
From the seasonal factors plot, we see that the price index increases from the beginning of the year to approximately around week 23. Then, it decreases from week 23 to the end of the year. If we analyze the summary of the fit, we see that seasonality is indeed present from examining the non-statistically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

```
#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")
```



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

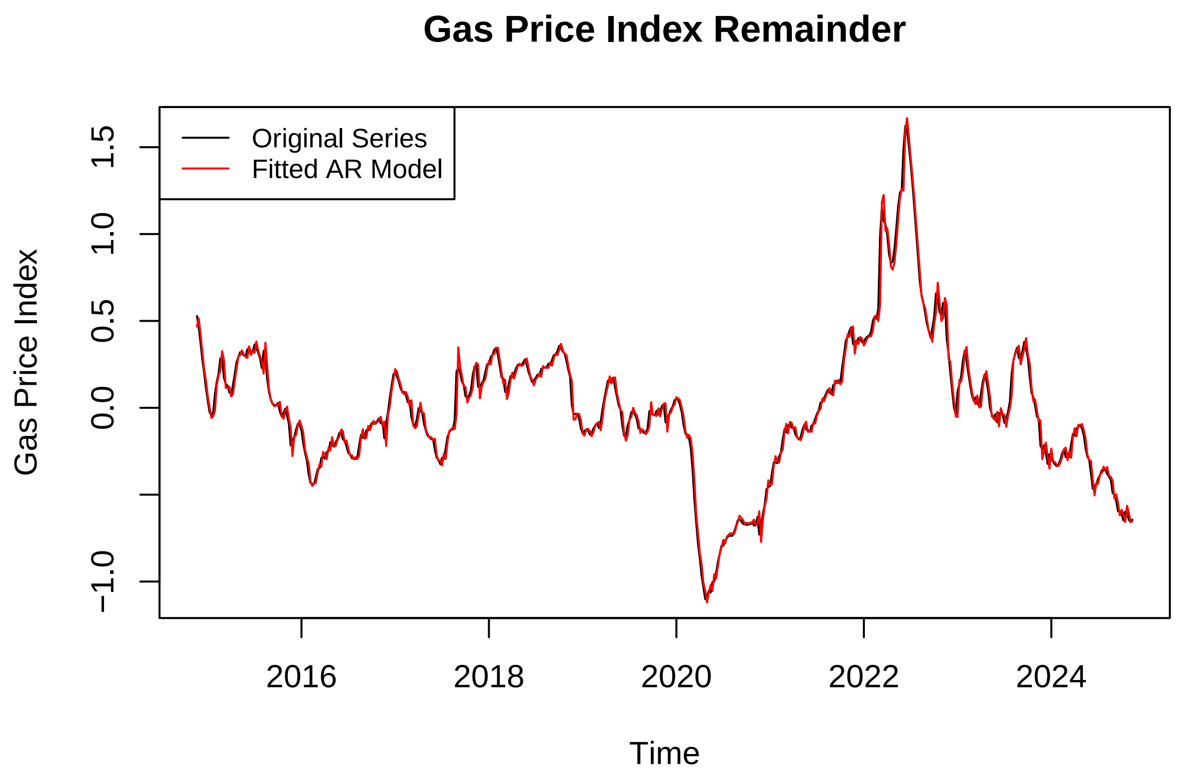


From the model above, we see a slow decay in the ACF, as well as two statistically significant spikes in the first two lags of the PACF. Hence, we propose to fit an AR(2) model to the residuals as follows:

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

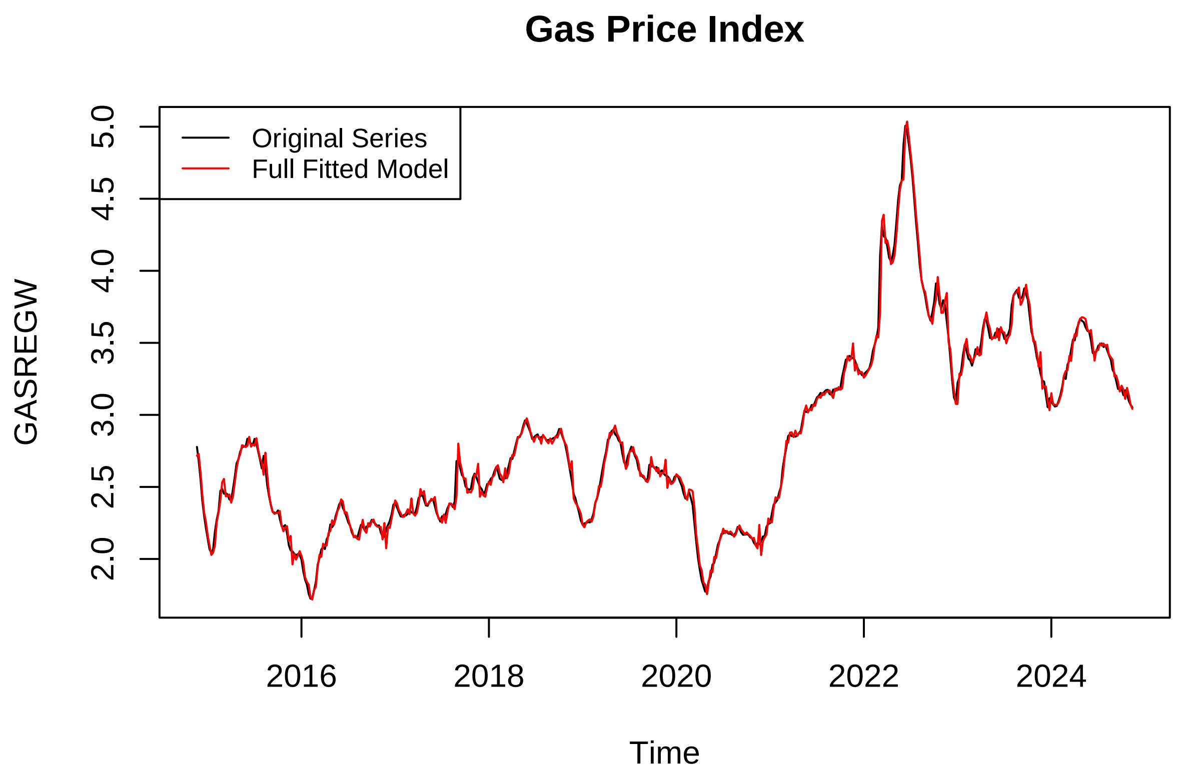
Let us make this fit using the arima() function as follows:

```r
#Create the AR(2) object on the 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:

```r
#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)
```

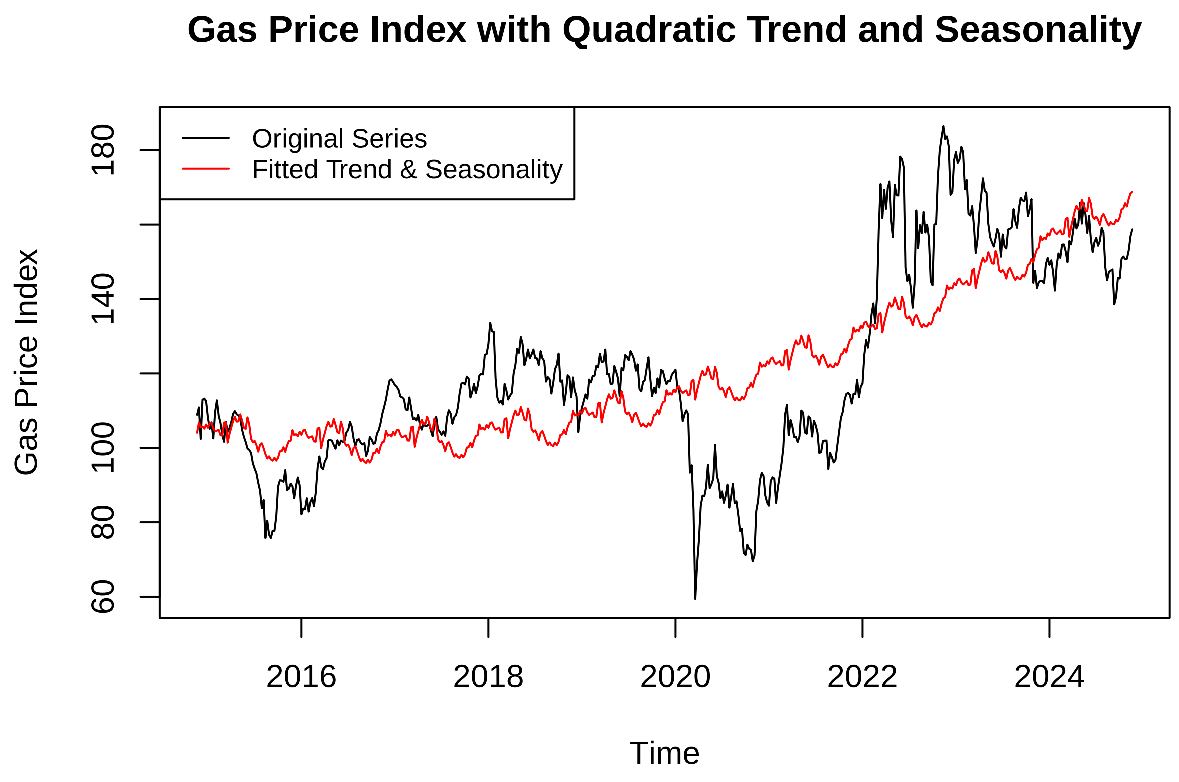


We will propose the following model for the trend and seasonality of the Chevron stock as follows:

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

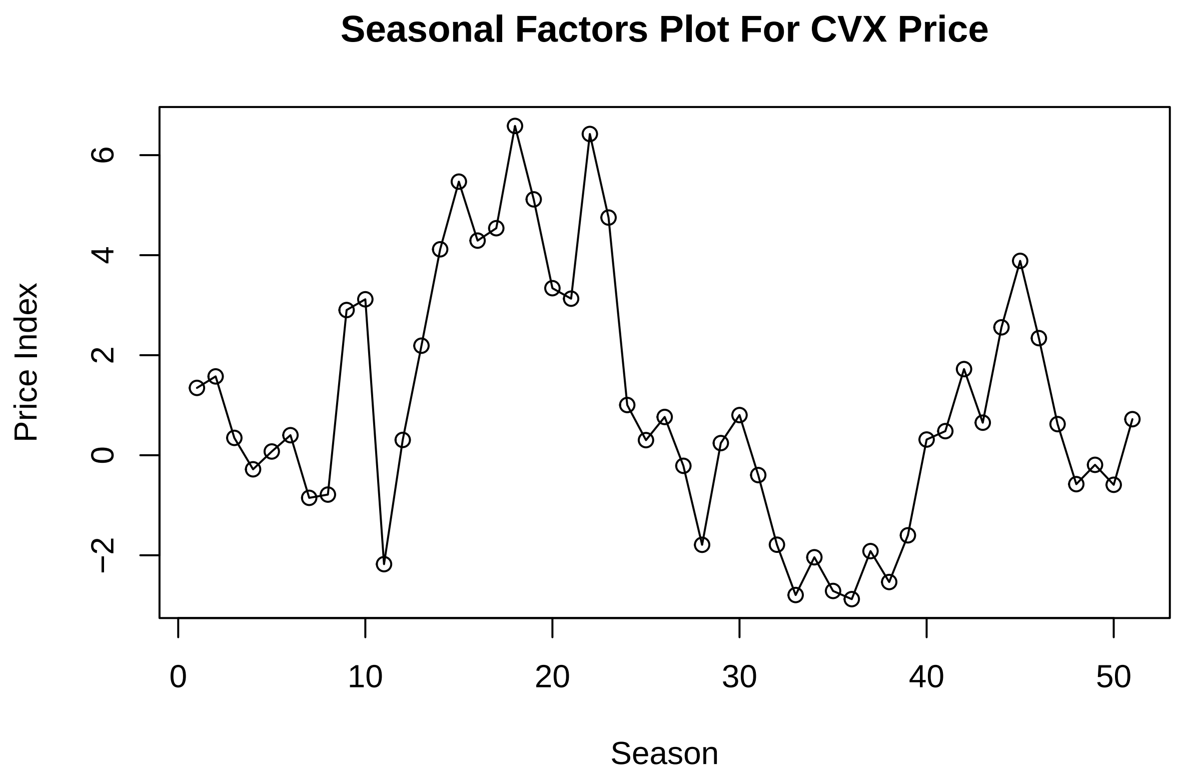
Let us first fit a model with a quadratic trend and weekly seasonal dummies using the tslm() function as follows:

```r
#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:

```
# 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")
```



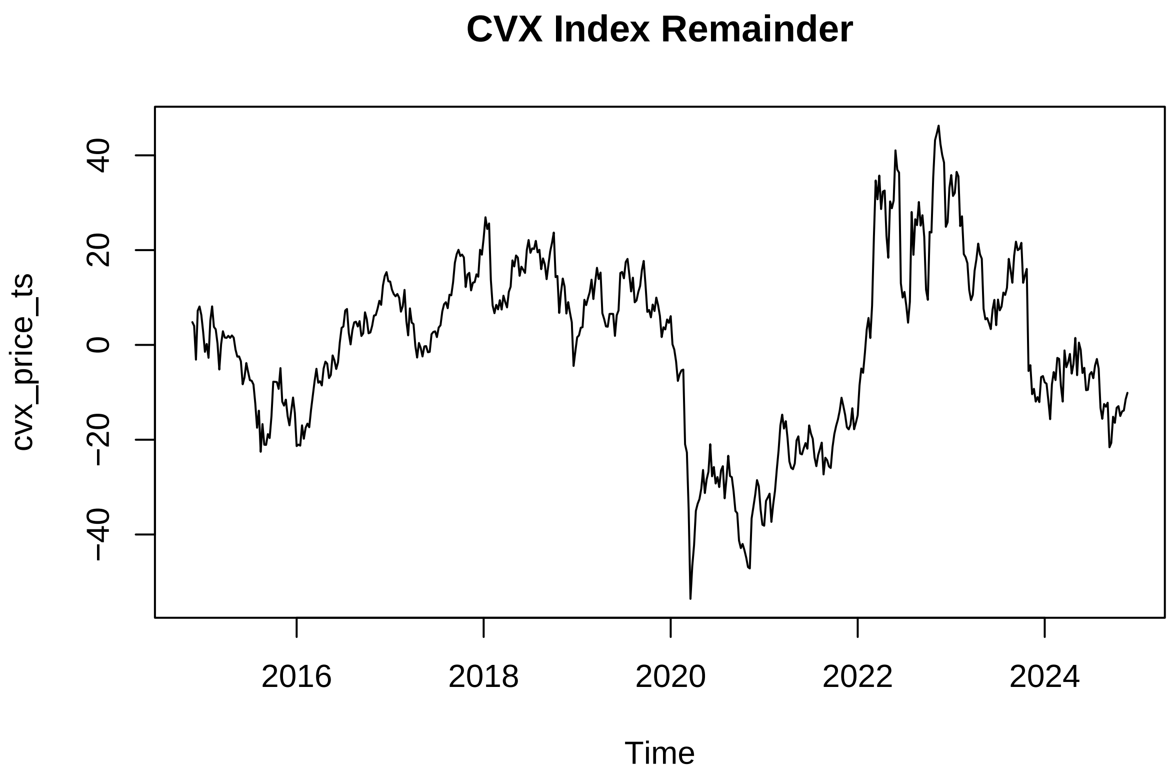
```r
#Summary of the coefficients
summary(fit_seasonal_cvx)
```

```
##
## Call:
## tslm(formula = cvx_price_ts ~ season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.168 -18.949  -5.545  24.766  63.342
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  119.736      8.996  13.309   <2e-16 ***
## season2        1.346     12.723   0.106    0.916
## season3        1.576     12.723   0.124    0.901
## season4        0.347     12.723   0.027    0.978
## season5       -0.281     12.723  -0.022    0.982
## season6        0.075     12.723   0.006    0.995
## season7        0.401     12.723   0.032    0.975
## season8       -0.852     12.723  -0.067    0.947
## season9       -0.787     12.723  -0.062    0.951
## season10       2.903     12.723   0.228    0.820
## season11       3.118     12.723   0.245    0.807
## season12      -2.178     12.723  -0.171    0.864
## season13       0.306     12.723   0.024    0.981
## season14       2.190     12.723   0.172    0.863
## season15       4.117     12.723   0.324    0.746
## season16       5.470     12.723   0.430    0.667
## season17       4.291     12.723   0.337    0.736
## season18       4.539     12.723   0.357    0.721
## season19       6.584     12.723   0.518    0.605
## season20       5.116     12.723   0.402    0.688
## season21       3.340     12.723   0.263    0.793
## season22       3.130     12.723   0.246    0.806
## season23       6.423     12.723   0.505    0.614
## season24       4.752     12.723   0.374    0.709
## season25       1.004     12.723   0.079    0.937
## season26       0.302     12.723   0.024    0.981
## season27       0.765     12.723   0.060    0.952
## season28      -0.212     12.723  -0.017    0.987
## season29      -1.791     12.723  -0.141    0.888
## season30       0.243     12.723   0.019    0.985
## season31       0.803     12.723   0.063    0.950
## season32      -0.397     12.723  -0.031    0.975
## season33      -1.789     12.723  -0.141    0.888
## season34      -2.795     12.723  -0.220    0.826
## season35      -2.040     12.723  -0.160    0.873
## season36      -2.714     12.723  -0.213    0.831
## season37      -2.877     12.723  -0.226    0.821
## season38      -1.917     12.723  -0.151    0.880
## season39      -2.535     12.723  -0.199    0.842
## season40      -1.601     12.723  -0.126    0.900
## season41       0.314     12.723   0.025    0.980
## season42       0.482     12.723   0.038    0.970
## season43       1.722     12.723   0.135    0.892
## season44       0.651     12.723   0.051    0.959
## season45       2.557     12.723   0.201    0.841
## season46       3.886     12.723   0.305    0.760
## season47       2.341     12.430   0.188    0.851
## season48       0.622     12.723   0.049    0.961
## season49      -0.578     12.723  -0.045    0.964
## season50      -0.192     12.723  -0.015    0.988
## season51      -0.591     12.723  -0.046    0.963
## season52       0.721     12.723   0.057    0.955
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.45 on 469 degrees of freedom
## Multiple R-squared:  0.008029,   Adjusted R-squared:  -0.09984
## F-statistic: 0.07444 on 51 and 469 DF,  p-value: 1
```

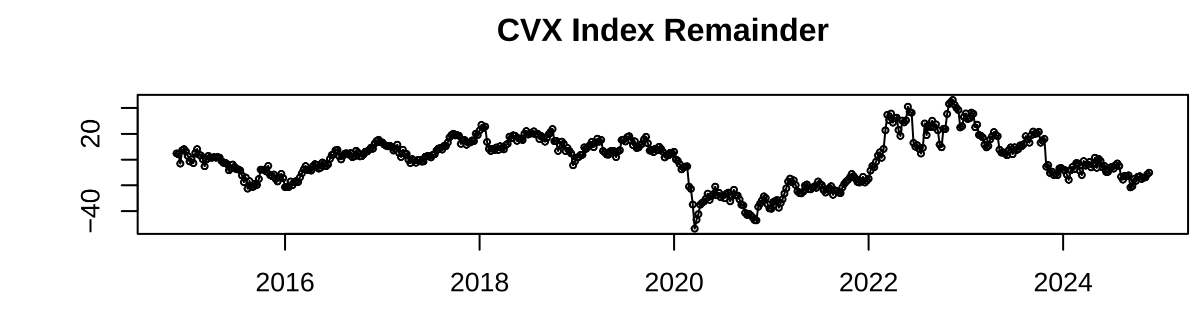
From the seasonal factors plot, seasonality varies throughout the year with the price index bottoming at weeks 10 and 40, while peaking at weeks 20 and 45. If we analyze the summary of the fit, we see that seasonality is indeed present from examining the non-statistically significant p-values.

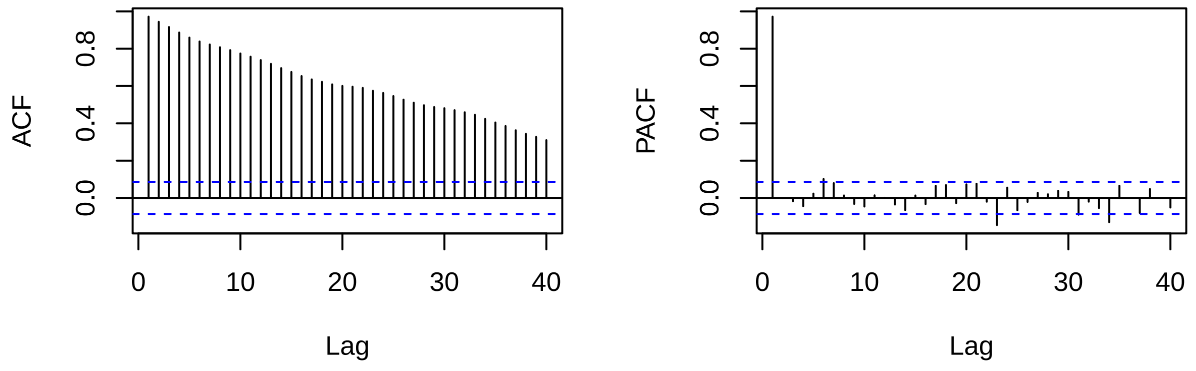
Now, let us examine the remainder component from the fitted model as follows, as well as examine the ACF and PACF of the remainder:

```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")
```



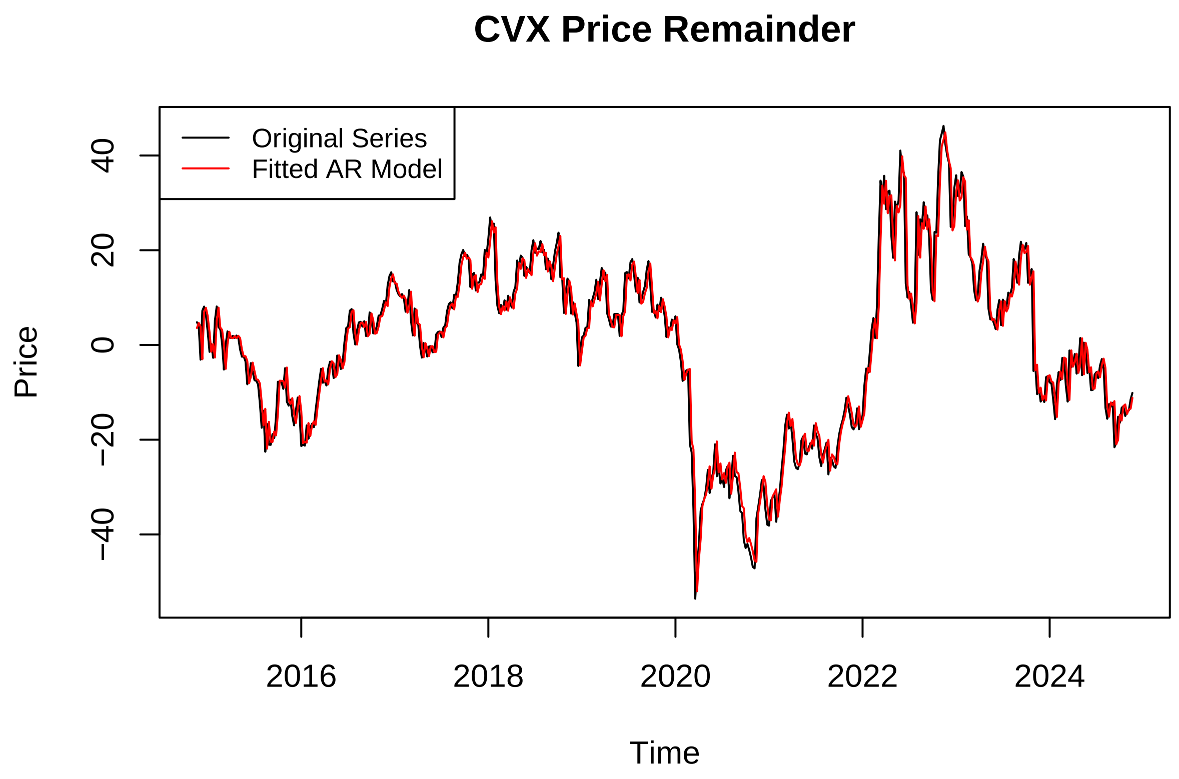


From the model above, we see a slow decay in the ACF, as well as a single statistically significant spike in the first lag of the PACF. Hence, we propose to fit an AR(1) model to the residuals as follows:

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

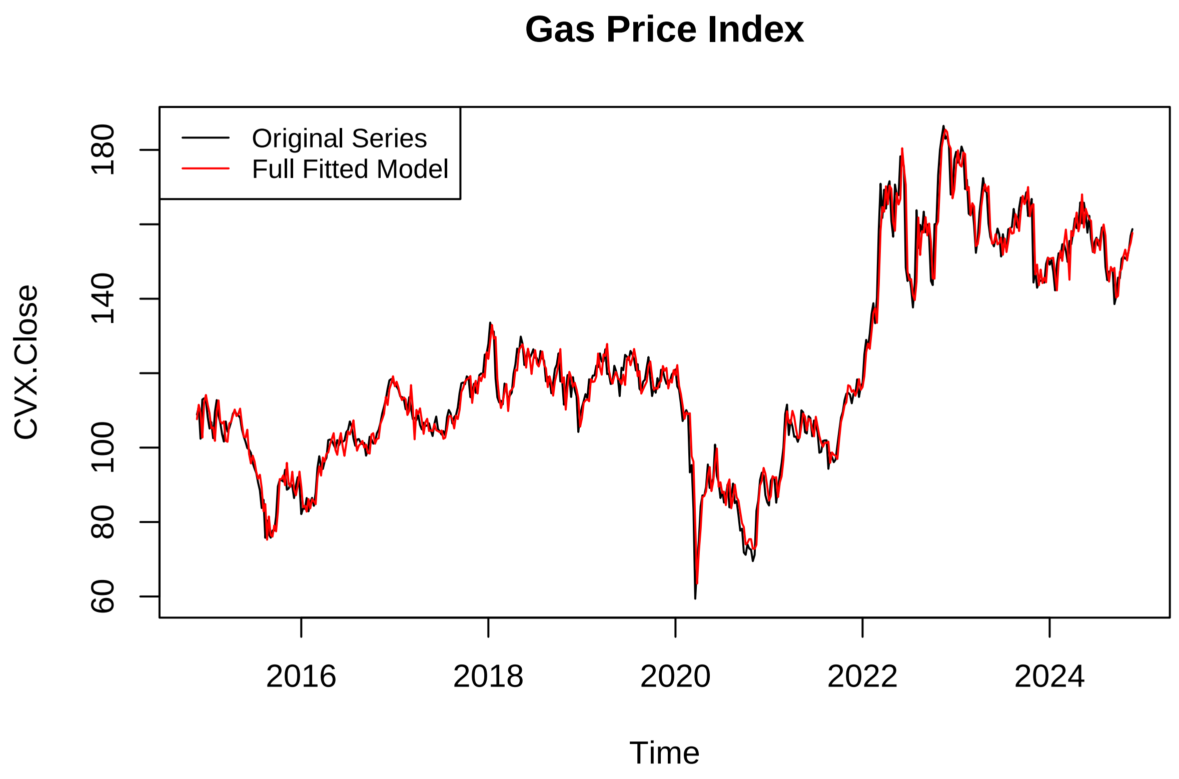
Let us make this fit using the arima() function as follows:

```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:

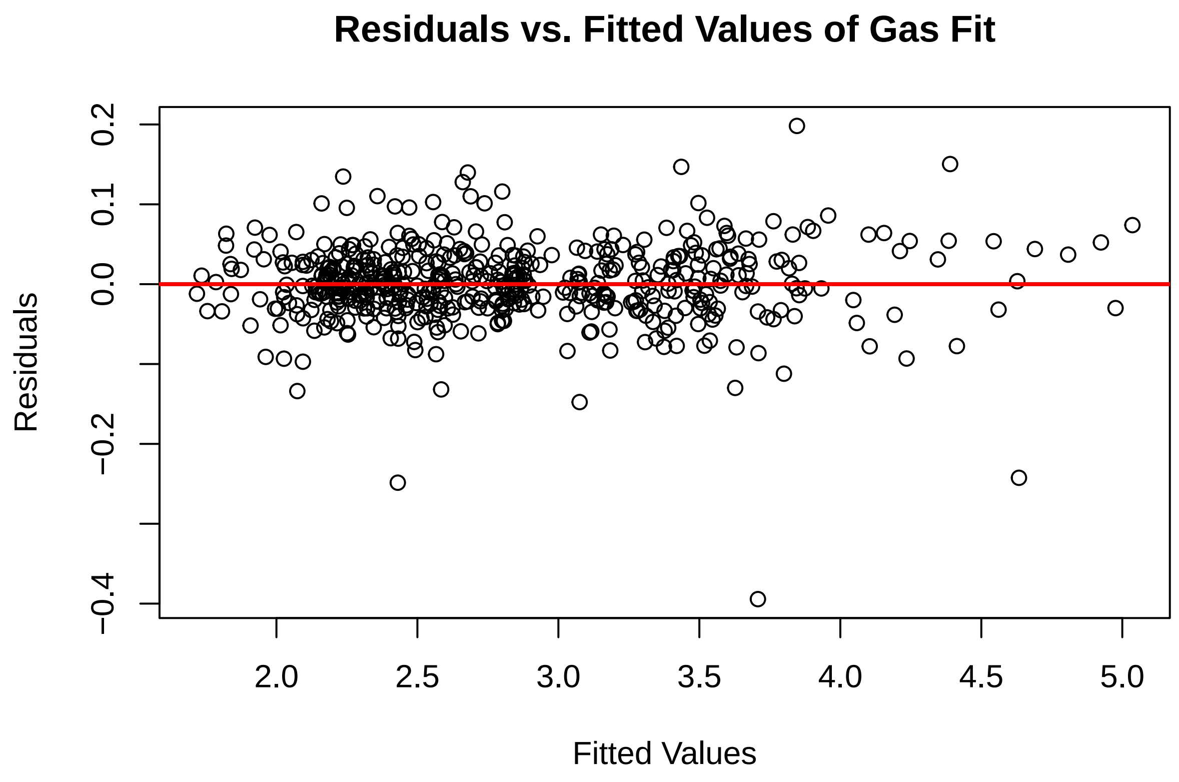
```
#Create the full model object
full_model_cvx <- fitted(fit_cvx) + fitted(resid_cvx)
#Plot the remainder object
plot(cvx_price_ts, main="Gas Price Index")
lines(full_model_cvx, col="Red")
legend("topleft",
       legend = c("Original Series", "Full Fitted Model"),
       col = c("black", "red"),
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

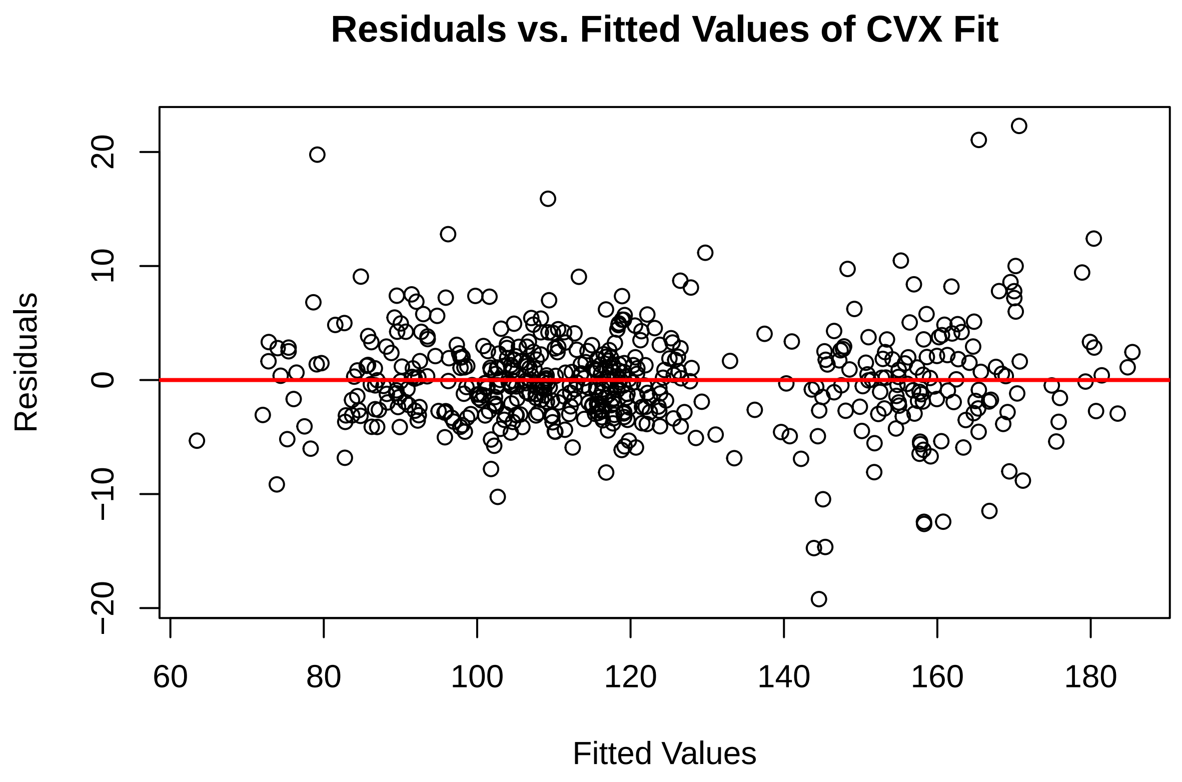
```
plot(full_model_gas, full_model_gas-gas_price_index_ts,
     main="Residuals vs. Fitted Values of Gas Fit",
     xlab="Fitted Values",
     ylab="Residuals")
# Add a horizontal line at 0 for reference
abline(h = 0, col = "red", lwd = 2)
```



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)
```

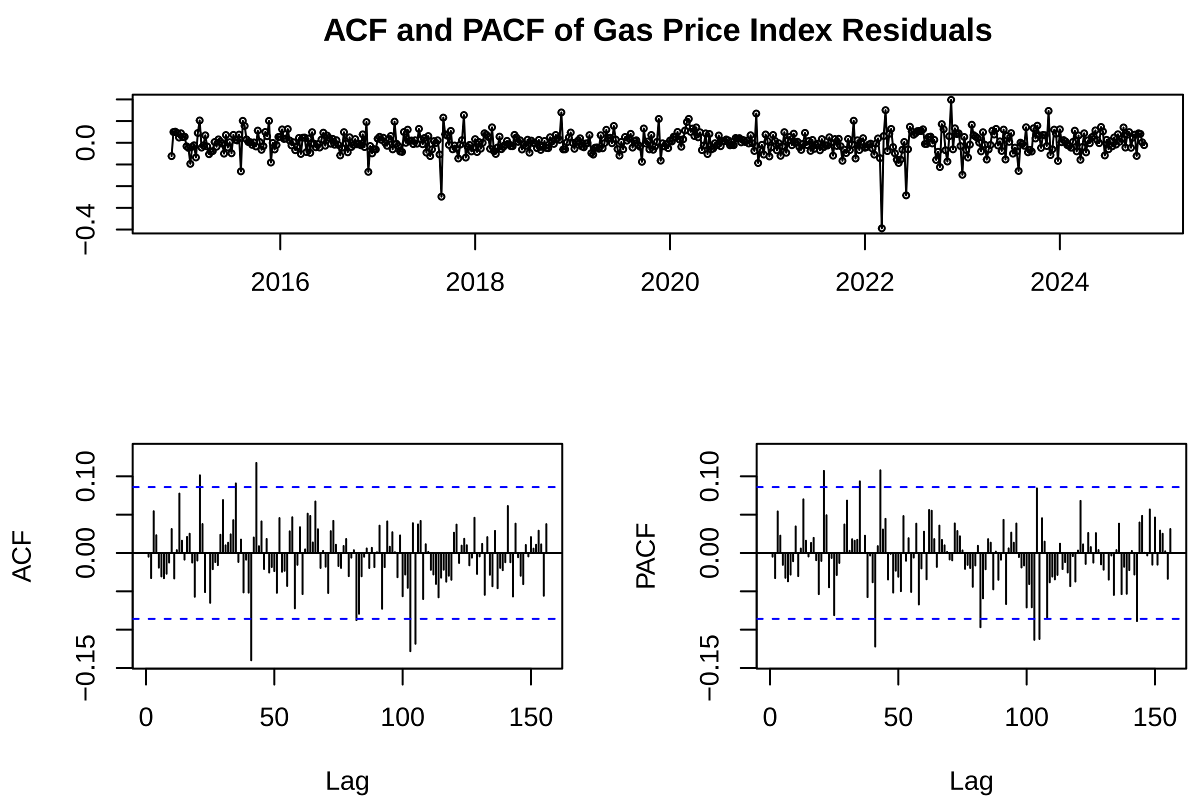


The residuals for the CVX fit are similar to that of the Gas fit. Analyzing the residuals vs fitted values of the gas plot, we see that the residuals are randomly distributed around 0 with no clear pattern. Also, the spread of residuals appears to be consistent, suggesting that variance of the residuals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

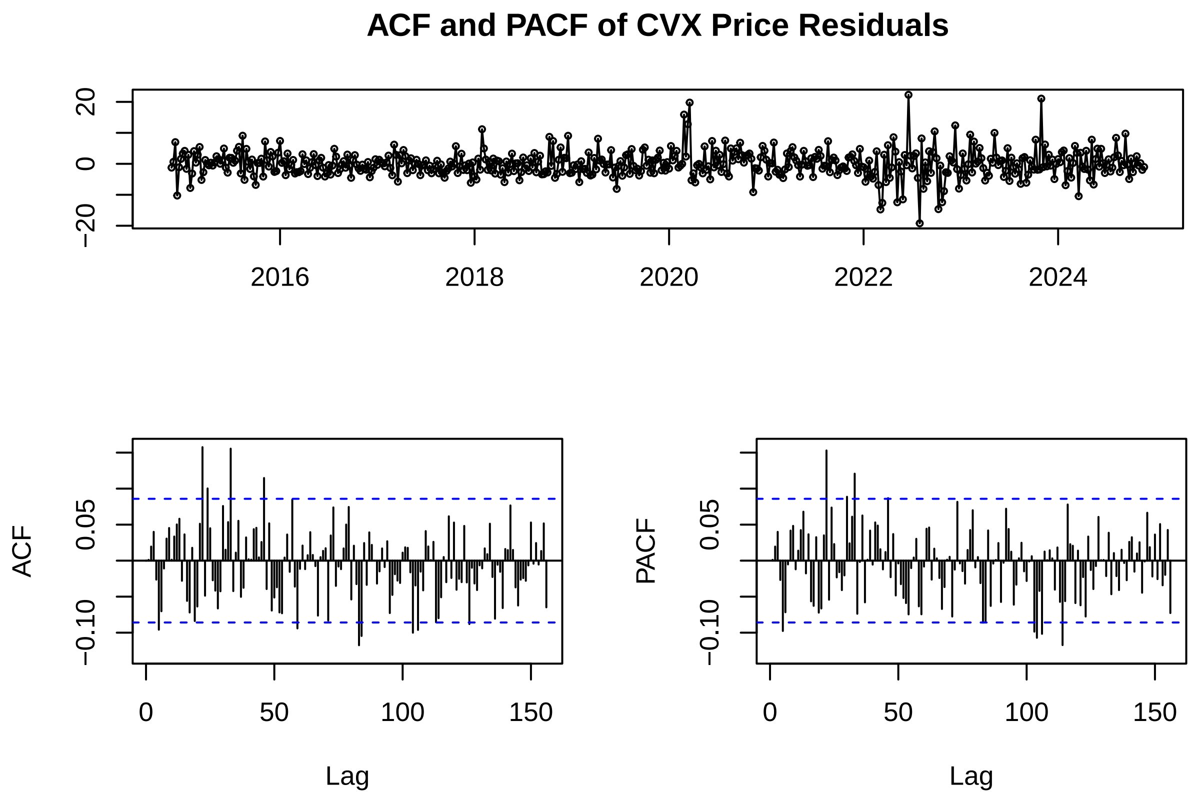
(e) ACF and PACF of respective residuals

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")
```



```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")
```

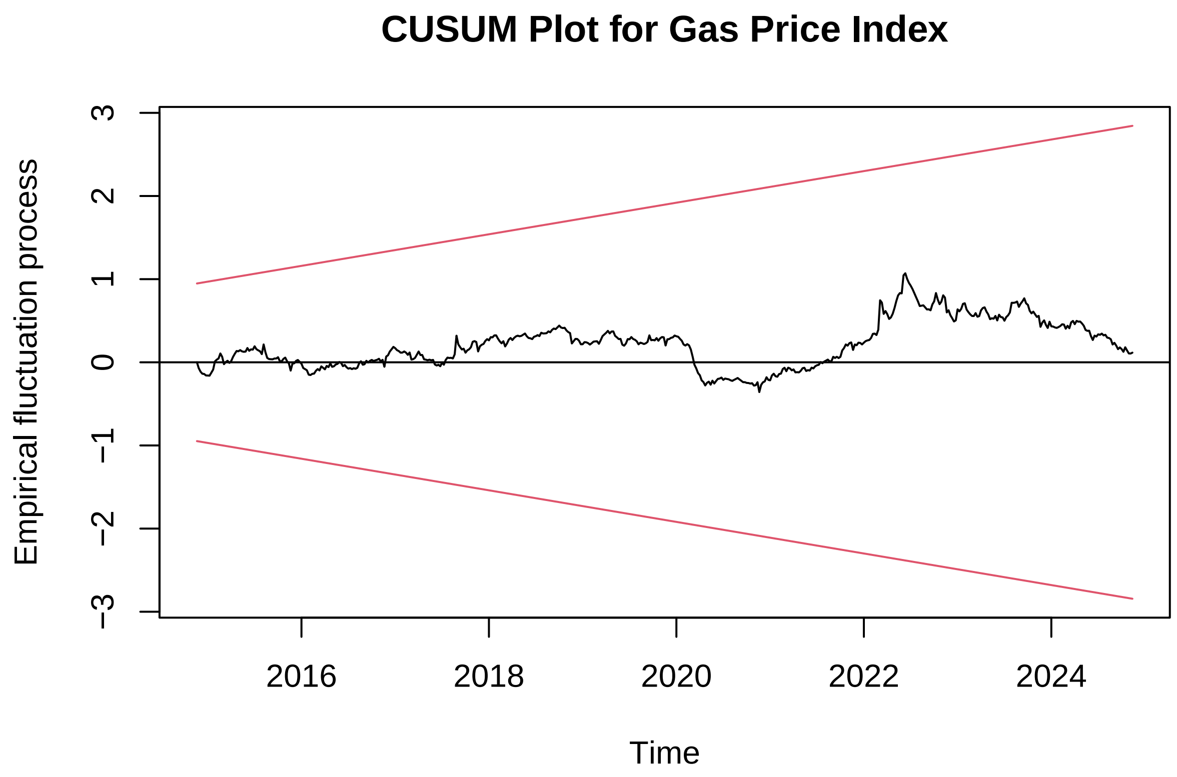


From the ACF and PACF of the residuals, we see that it appears that statistically significant lags appear to persist in some areas of the plot. However, it appears that there is no clear pattern to suggest that a seasonal AR model would fix the statistically significant lags. Other than the few statistically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

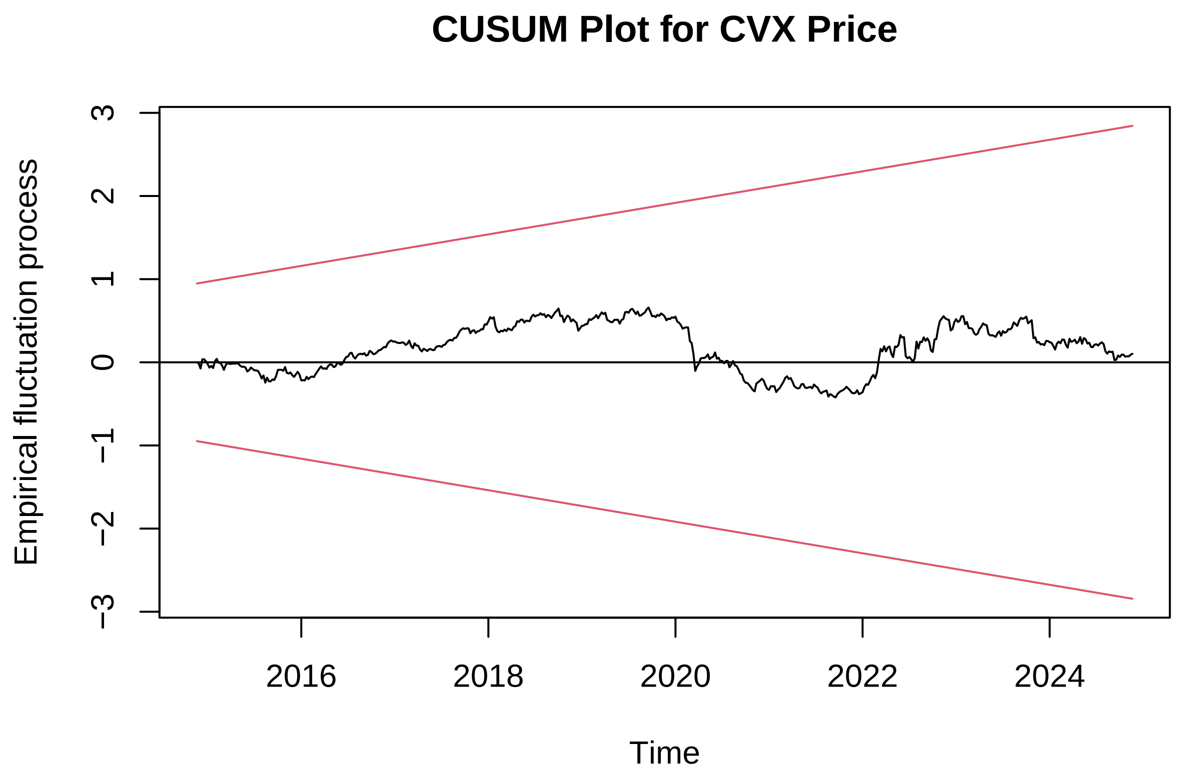
(f) CUSUM Plot

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 -0.01657363 4.262699 3.025283 -0.156142 2.593624
```

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

```
##
##  Box-Ljung test
##
## data:  cvx_residuals
## X-squared = 0.00071376, df = 1, p-value = 0.9787
```

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
```

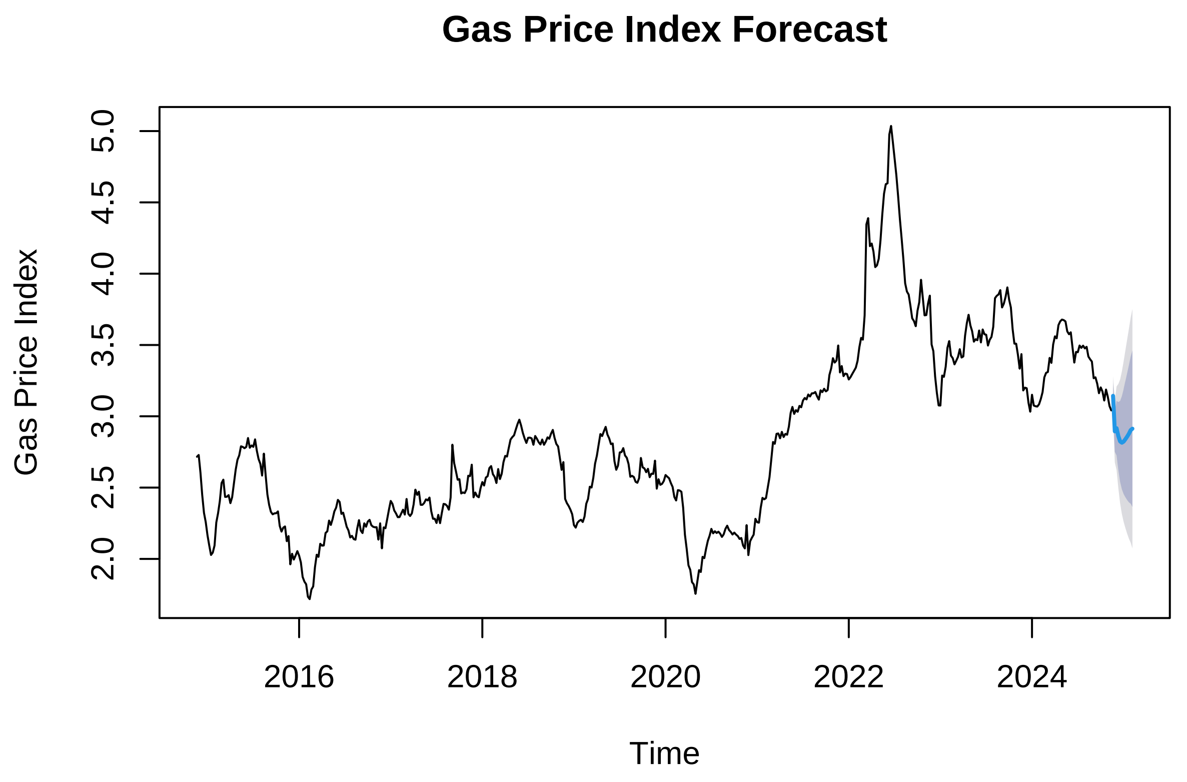
```
# Forecasting with the Chevron (CVX) stock price model
cvx_forecast <- forecast(full_model_cvx, h = forecast_horizon)
print("Chevron (CVX) Stock Price Forecast:")
```

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

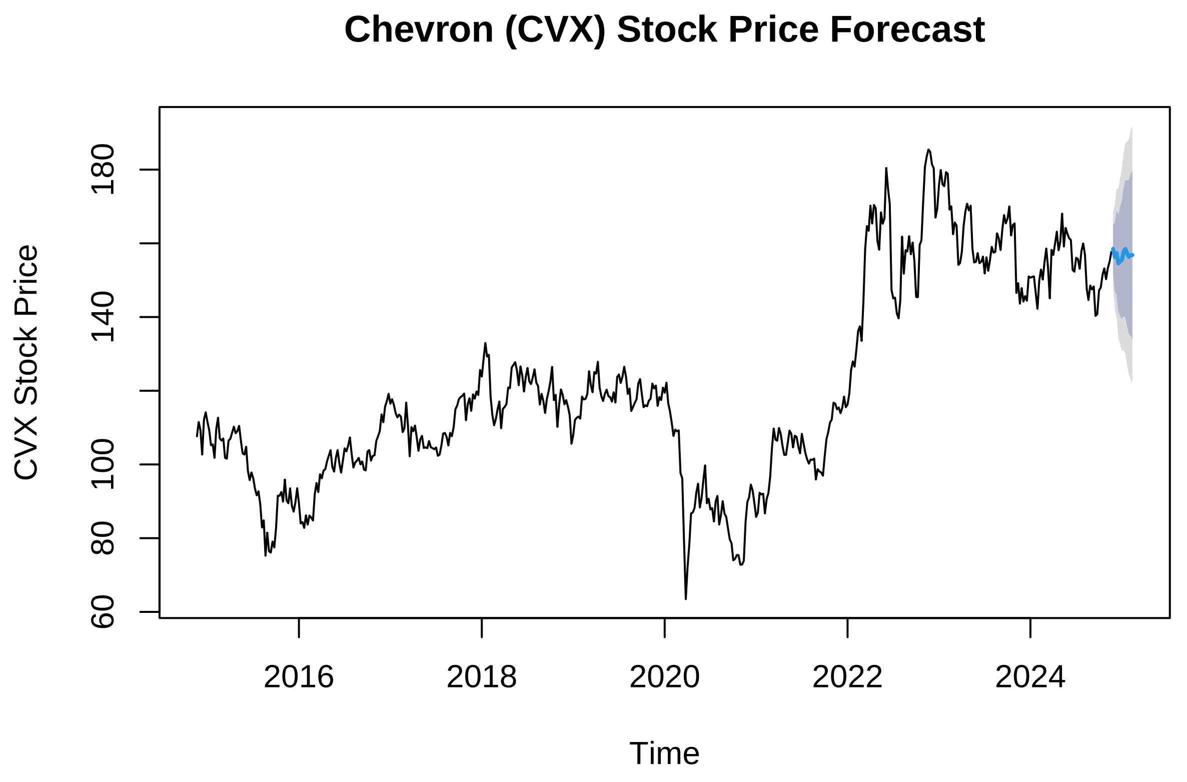
```
print(cvx_forecast)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2024.904       158.4923 151.8971 165.0875 148.4058 168.5788
## 2024.923       156.2866 146.9576 165.6157 142.0190 170.5542
## 2024.942       157.3628 145.9342 168.7915 139.8843 174.8414
## 2024.962       154.5620 141.3619 167.7621 134.3742 174.7498
## 2024.981       155.1429 140.3808 169.9050 132.5662 177.7196
## 2025.000       155.4417 139.2663 171.6171 130.7036 180.1798
## 2025.019       157.8045 140.3284 175.2806 131.0771 184.5318
## 2025.038       158.4806 139.7929 177.1684 129.9002 187.0611
## 2025.058       157.2216 137.3949 177.0482 126.8993 187.5438
## 2025.077       156.3197 135.4149 177.2245 124.3486 188.2908
## 2025.096       156.8054 134.8744 178.7364 123.2648 190.3460
## 2025.115       156.8103 133.8979 179.7227 121.7688 191.8518
```

```
# 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()

Now, we are considering using auto.arima instead of the manual capture model, as it can help us capture the trend, seasonality, cycle, and differences in the data directly if needed.

```r
# Applying auto.arima to gas price index time series
auto_arima_gas <- auto.arima(gas_price_index_ts)
summary(auto_arima_gas)
```

```
## Series: gas_price_index_ts
## ARIMA(1,1,0)(1,0,0)[52]
##
## Coefficients:
##          ar1    sar1
##       0.5826  0.0181
## s.e.  0.0357  0.0460
##
## sigma^2 = 0.002334:  log likelihood = 837.01
## AIC=-1668.02   AICc=-1667.97   BIC=-1655.26
##
## Training set error measures:
##                        ME       RMSE        MAE        MPE     MAPE       MASE
## Training set 0.0002366172 0.04816861 0.03289611 0.01399381 1.156519 0.07117341
##                     ACF1
## Training set 0.009744658
```

```r
# Applying auto.arima to Chevron stock price time series
auto_arima_cvx <- auto.arima(cvx_price_ts)
summary(auto_arima_cvx)
```

```
## Series: cvx_price_ts
## ARIMA(0,1,0)
##
## sigma^2 = 21.07:  log likelihood = -1530.34
## AIC=3062.69   AICc=3062.69   BIC=3066.94
##
## Training set error measures:
##                      ME     RMSE      MAE          MPE     MAPE      MASE
## Training set 0.09589034 4.586273 3.198769 -0.008730883 2.737685 0.1561792
##                     ACF1
## Training set -0.01997886
```

The gas price index shows us ARIMA(1,1,0)(1,0,0)[52], which means:

- The ARIMA(1,1,0)(1,0,0)[52] model is a seasonal ARIMA model that combines both non-seasonal and seasonal components to capture patterns in the data. In the non-seasonal part, it includes an autoregressive (AR) term with p = 1, meaning it has one non-seasonal autoregressive term, and it applies first-order differencing (d=1) to make the series stationary. There is no non-seasonal moving average (MA) component, as q=0. In the seasonal part, the model includes one seasonal autoregressive term with P=1, no seasonal differencing (D=0), and no seasonal moving average component (Q=0). The seasonality period is s=52, which suggests a yearly pattern with weekly data, assuming 52 weeks per year.

To represent the model mathematically, we could write as follows:

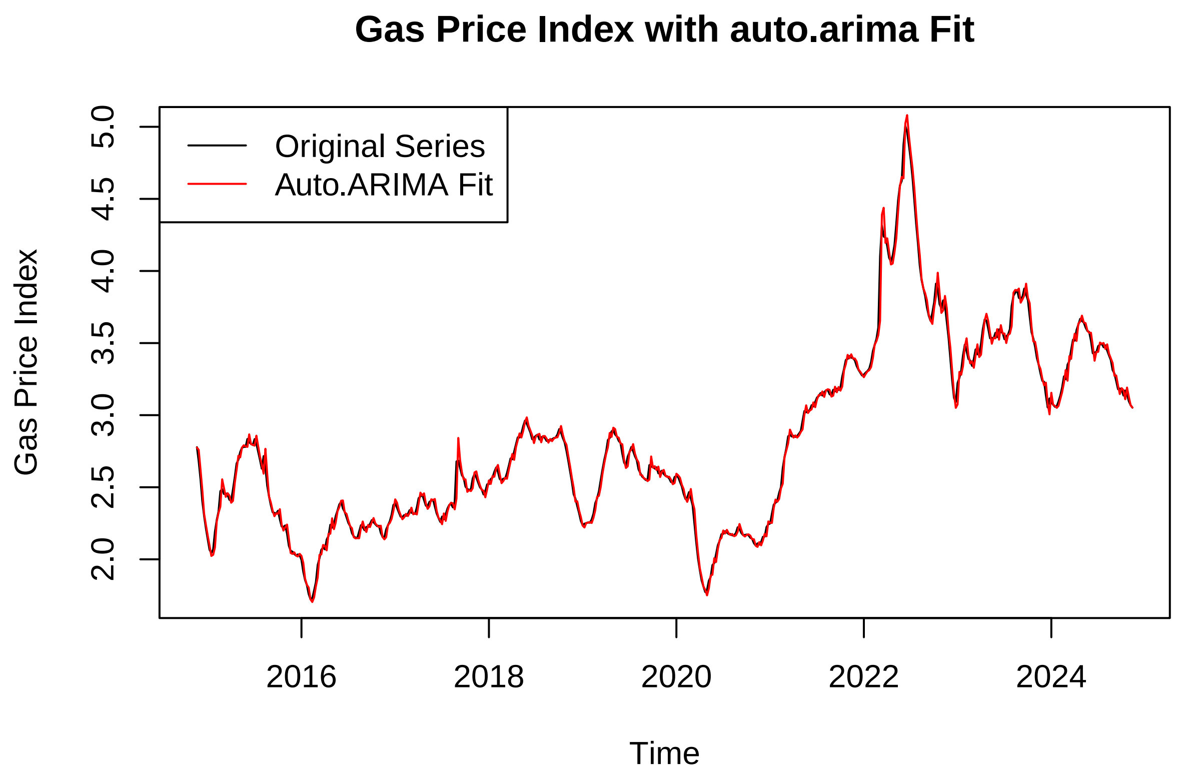
$$(1 - \phi_1 L)(1 - \Phi_1 L^{52})(1 - L)Y_t = \epsilon_t$$

The same analysis work for the Chevron's stock price. ARIMA(0,1,0) shows us that it just simple differencing model with no autoregressive or moving average components. As a result, the formula is more likely as follows:

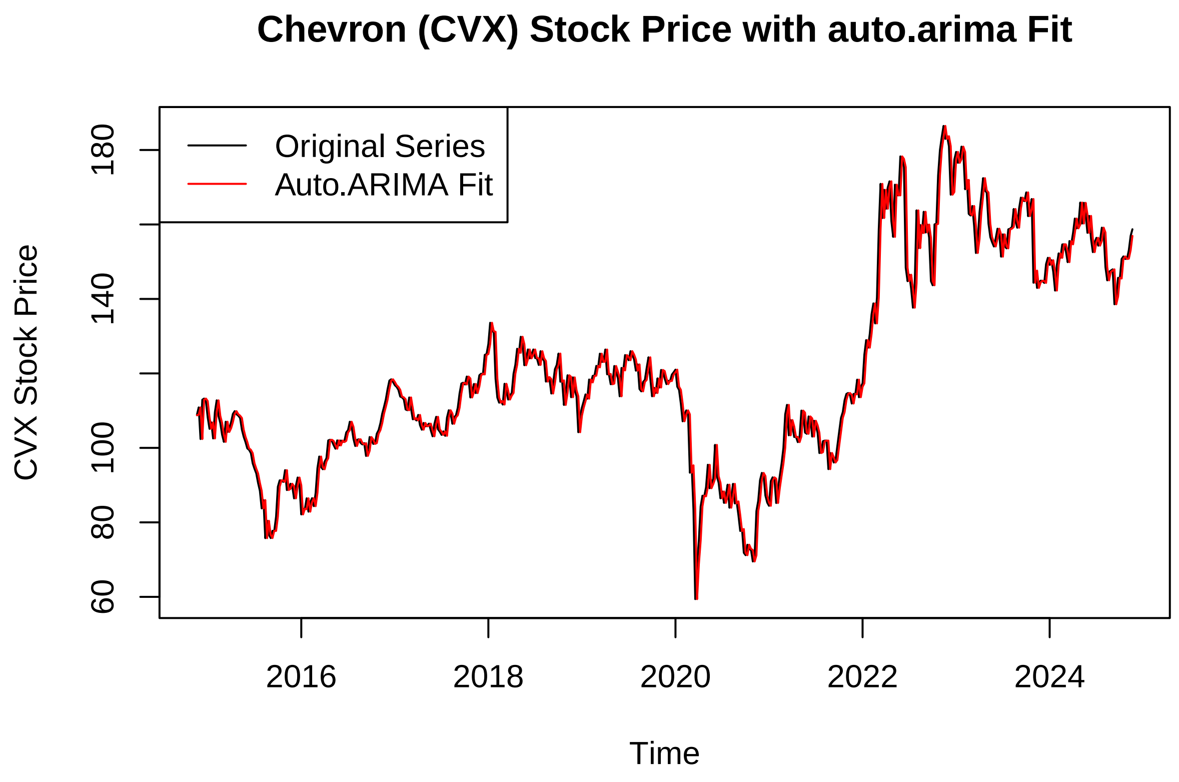
$$Y_t = Y_{t-1} + \epsilon_t$$

Next, let's plot it to visually assess the fit of auto.arima models to the original data:

```r
# Plot the fitted values for gas price index using auto.arima
plot(gas_price_index_ts, main = "Gas Price Index with auto.arima Fit",
     ylab = "Gas Price Index", xlab = "Time")
lines(fitted(auto_arima_gas), col = "red")
legend("topleft", legend = c("Original Series", "Auto.ARIMA Fit"),
       col = c("black", "red"), lty = 1)
```



```r
# Plot the fitted values for Chevron stock price using auto.arima
plot(cvx_price_ts, main = "Chevron (CVX) Stock Price with auto.arima Fit",
     ylab = "CVX Stock Price", xlab = "Time")
lines(fitted(auto_arima_cvx), col = "red")
legend("topleft", legend = c("Original Series", "Auto.ARIMA Fit"),
       col = c("black", "red"), lty = 1)
```



From above, we could find that the fitted-lines for auto.arima align perfectly with the original data since it capture the trend, seasonality, cycles, and even the turning points.

After plotting out the auto.arima model, let's compare based on the fit statistically and visually:

First, comparing statistically by using accuracy():

```
# Accuracy for Gas Price Index
accuracy_manual_gas <- accuracy(full_model_gas, gas_price_index_ts)
accuracy_auto_arima_gas <- accuracy(fitted(auto_arima_gas), gas_price_index_ts)
print(accuracy_manual_gas)
```

```
##                        ME       RMSE         MAE         MPE     MAPE         ACF1
## Test set -0.0008578556 0.04878779 0.03394842 -0.05608965 1.215091 -0.004862458
##          Theil's U
## Test set 0.8442807
```

```
print(accuracy_auto_arima_gas)
```

```
##                       ME       RMSE         MAE        MPE     MAPE        ACF1
## Test set 0.0002366172 0.04816861 0.03289611 0.01399381 1.156519 0.009744658
##          Theil's U
## Test set  0.807404
```

```r
# Accuracy for Chevron (CVX) Stock Price
accuracy_manual_cvx <- accuracy(full_model_cvx, cvx_price_ts)
accuracy_auto_arima_cvx <- accuracy(fitted(auto_arima_cvx), cvx_price_ts)
print(accuracy_manual_cvx)
```

```
##                   ME     RMSE      MAE       MPE     MAPE        ACF1 Theil's U
## Test set -0.01657363 4.262699 3.025283 -0.156142 2.593624 0.001167102 0.9134268
```

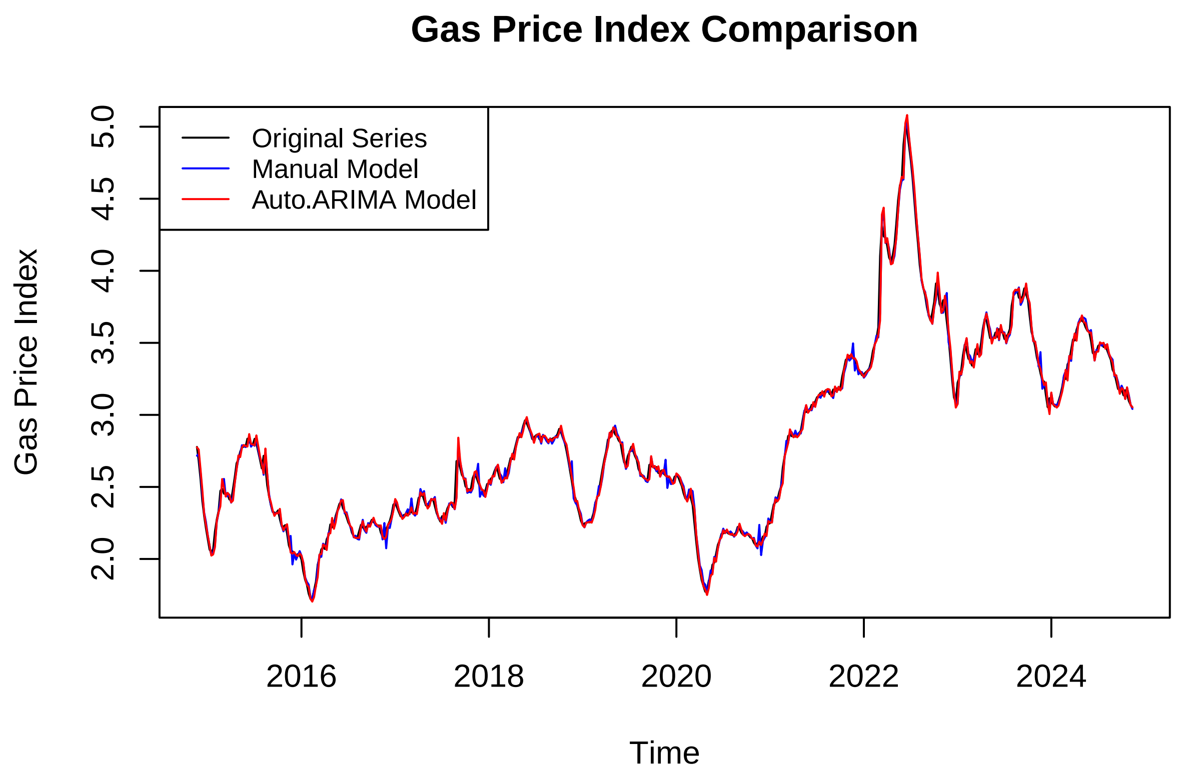
```r
print(accuracy_auto_arima_cvx)
```

```
##                  ME     RMSE      MAE          MPE     MAPE        ACF1
## Test set 0.09589034 4.586273 3.198769 -0.008730883 2.737685 -0.01997886
##          Theil's U
## Test set         1
```

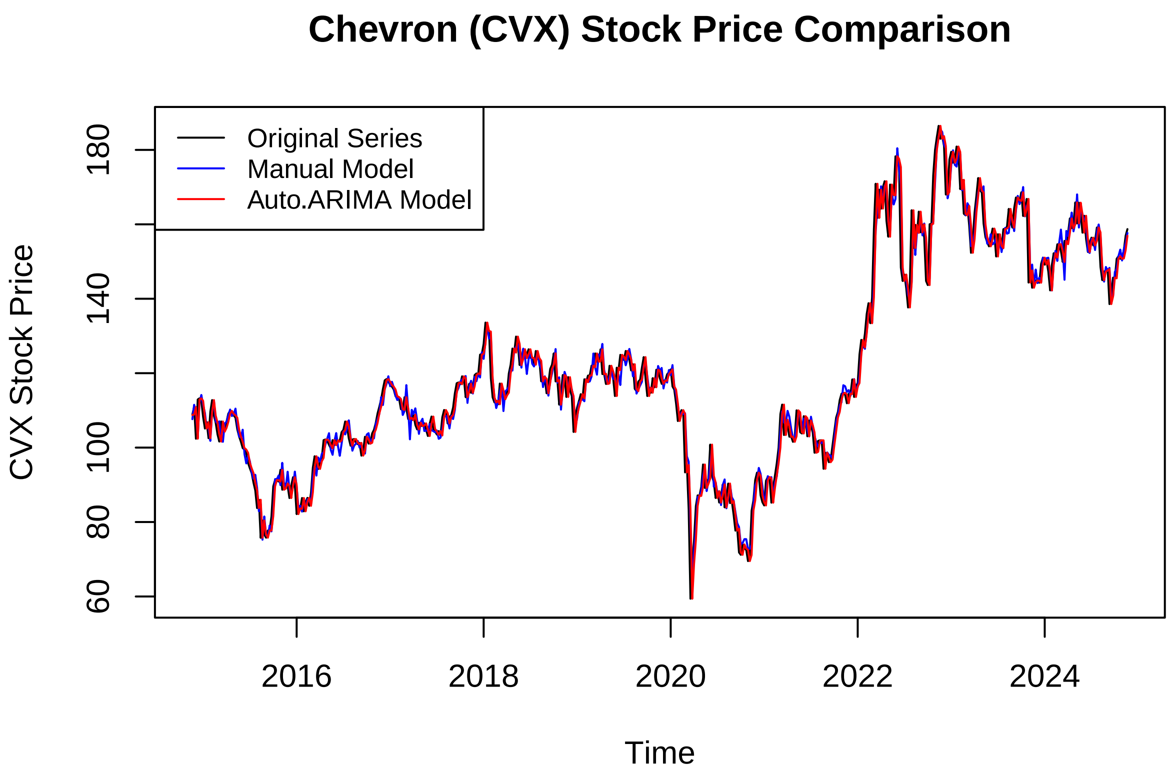
After comparison statistically, we would find that the manual caputure model have a better fit than the auto.arima model. However, it doesn't mean that auto.arima must have less accuracy when doing forecast since robustness is also a really important factor.

Second, comparing by using plots:

```r
# Plot for Gas Price Index
plot(gas_price_index_ts, main = "Gas Price Index Comparison",
     ylab = "Gas Price Index", xlab = "Time")
lines(fitted(fit_gas) + fitted(resid_gas),
      col = "blue", lty = 1)  # Manual capture model
lines(fitted(auto_arima_gas), col = "red", lty = 1)  # auto.arima model
legend("topleft",
       legend = c("Original Series", "Manual Model", "Auto.ARIMA Model"),
       col = c("black", "blue", "red"), lty = 1, cex = 0.8)
```



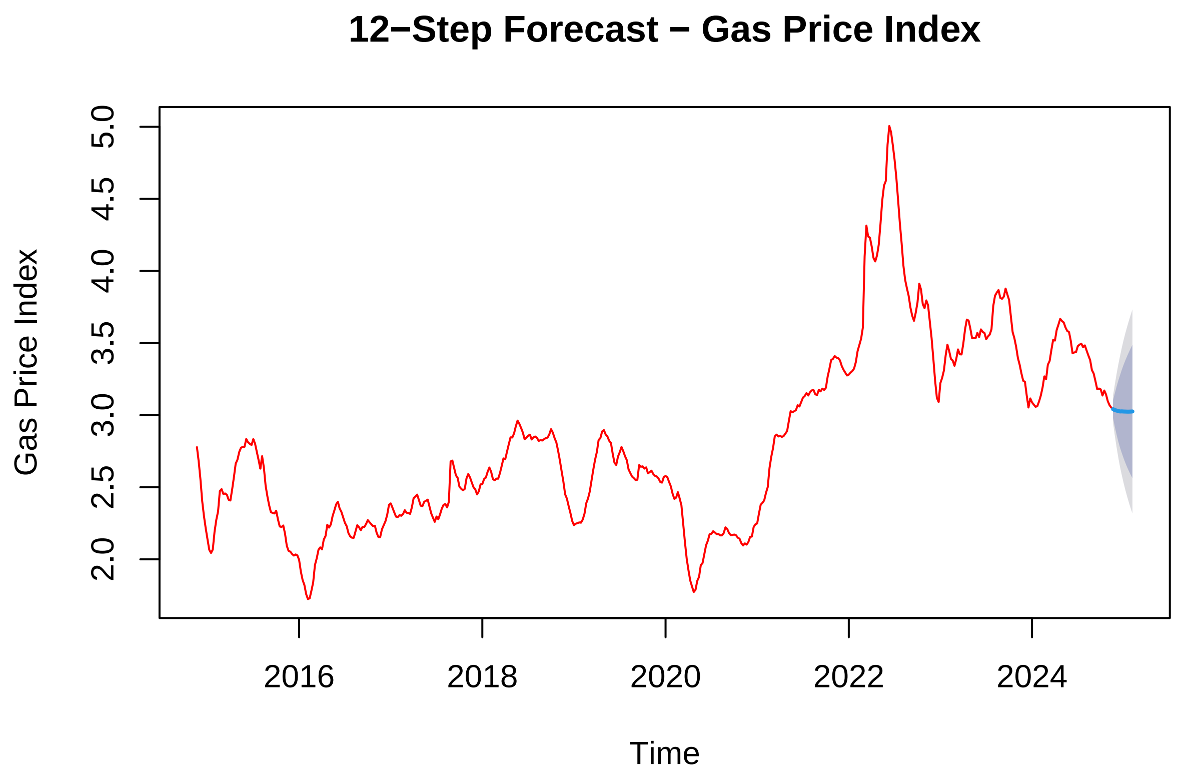
```r
# Plot for CVX Stock Price
plot(cvx_price_ts, main = "Chevron (CVX) Stock Price Comparison",
     ylab = "CVX Stock Price", xlab = "Time")
lines(fitted(fit_cvx) + fitted(resid_cvx),
      col = "blue", lty = 1)  # Manual capture model
lines(fitted(auto_arima_cvx), col = "red", lty = 1)  # auto.arima model
legend("topleft",
       legend = c("Original Series", "Manual Model", "Auto.ARIMA Model"),
       col = c("black", "blue", "red"), lty = 1, cex = 0.8)
```



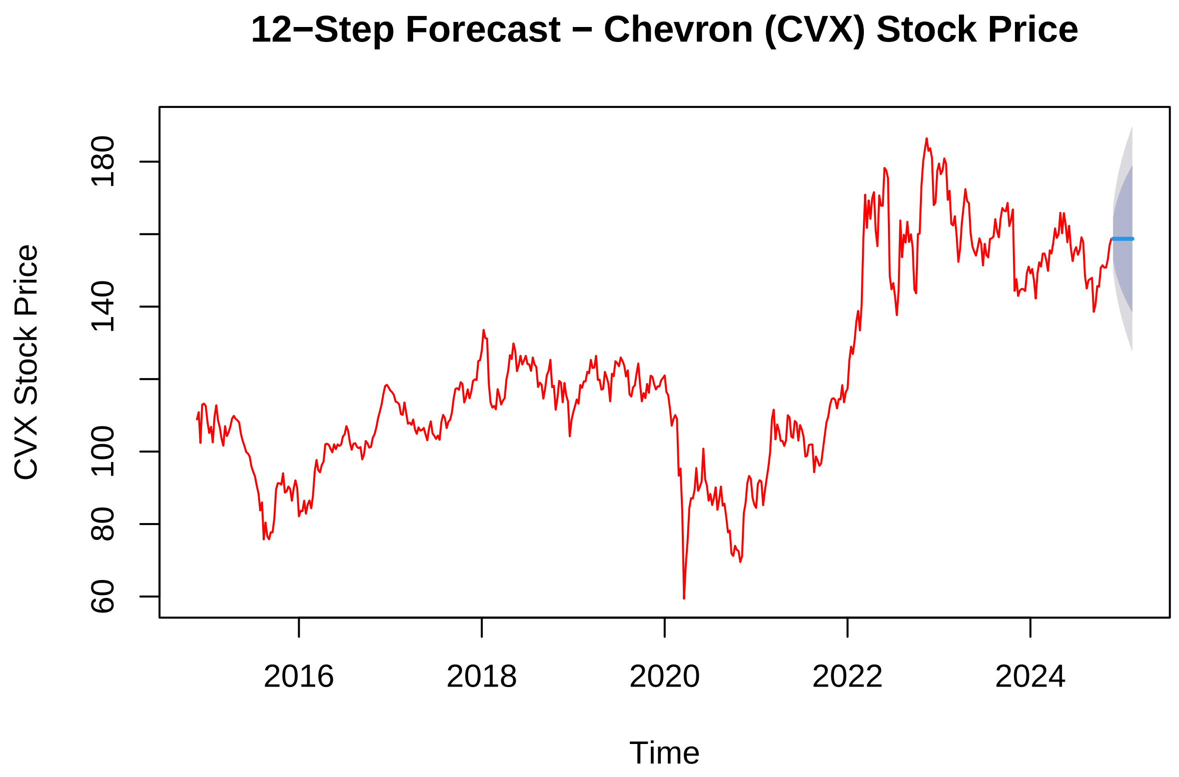
From the plots, we can observe that, compared to the manually captured models, the auto.arima models display a smoother trend. This may enhance forecasting accuracy and robustness for future out-of-sample data.

So what out the performance of auto.arima forecasting? Let's check it by using plots:

```r
# forecast 12 steps ahead
forecast_horizon <- 12

# 12-Step Forecast for Gas Price Index
auto_arima_forecast_gas <- forecast(auto_arima_gas, h = forecast_horizon)
plot(auto_arima_forecast_gas, main = "12-Step Forecast - Gas Price Index",
     ylab = "Gas Price Index", xlab = "Time", col = "red")
```



```r
# 12-Step Forecast for Chevron (CVX) Stock Price
auto_arima_forecast_cvx <- forecast(auto_arima_cvx, h = forecast_horizon)
plot(auto_arima_forecast_cvx,
     main = "12-Step Forecast - Chevron (CVX) Stock Price",
     ylab = "CVX Stock Price", xlab = "Time", col = "red")
```



Since accuracy, robustness, consistency, etc. are all factors we need to take care of when doing forecasting, let's comparing the errors, especially MAPE for the forecast instead of the fit of original model now. In order to let it work, we would split the data into training and testing sets for a fixed 12 periods window.

```
# Define the forecast horizon
forecast_horizon <- 12

# Calculate the training size by subtracting the forecast horizon
train_size_gas <- length(gas_price_index_ts) - forecast_horizon
train_size_cvx <- length(cvx_price_ts) - forecast_horizon

# Define the training and testing sets for Gas Price Index
train_data_gas <- head(gas_price_index_ts, train_size_gas)
test_data_gas <- tail(gas_price_index_ts, forecast_horizon)

# Define the training and testing sets for Chevron Stock Price (CVX)
train_data_cvx <- head(cvx_price_ts, train_size_cvx)
test_data_cvx <- tail(cvx_price_ts, forecast_horizon)

# Fit the manual models for trend and seasonality, and residuals
fit_gas_manual <- tslm(train_data_gas ~ trend + I(trend^2) + season)
resid_gas_manual <- arima(residuals(fit_gas_manual), order = c(2,0,0))

fit_cvx_manual <- tslm(train_data_cvx ~ trend + I(trend^2) + season)
resid_cvx_manual <- arima(residuals(fit_cvx_manual), order = c(1,0,0))
```

```r
# Forecast with the manual model components separately
manual_forecast_trend_season_gas <- forecast(fit_gas_manual, h = forecast_horizon)
manual_forecast_residual_gas <- forecast(resid_gas_manual, h = forecast_horizon)

manual_forecast_trend_season_cvx <- forecast(fit_cvx_manual, h = forecast_horizon)
manual_forecast_residual_cvx <- forecast(resid_cvx_manual, h = forecast_horizon)

# Function to combine forecast objects
combine_forecasts <- function(forecast1, forecast2) {
  combined_mean <- forecast1$mean + forecast2$mean
  combined_lower <- forecast1$lower + forecast2$lower
  combined_upper <- forecast1$upper + forecast2$upper

  combined_forecast <- forecast1
  combined_forecast$mean <- combined_mean
  combined_forecast$lower <- combined_lower
  combined_forecast$upper <- combined_upper

  return(combined_forecast)
}

# Combine manual forecasts for Gas Price Index and Chevron Stock Price
manual_forecast_gas <- combine_forecasts(manual_forecast_trend_season_gas,
                                         manual_forecast_residual_gas)
manual_forecast_cvx <- combine_forecasts(manual_forecast_trend_season_cvx,
                                         manual_forecast_residual_cvx)

# Fit auto.arima models for comparison
auto_arima_gas <- auto.arima(train_data_gas)
auto_arima_forecast_gas <- forecast(auto_arima_gas, h = forecast_horizon)

auto_arima_cvx <- auto.arima(train_data_cvx)
auto_arima_forecast_cvx <- forecast(auto_arima_cvx, h = forecast_horizon)

# Calculate accuracy metrics
accuracy_manual_gas <- accuracy(manual_forecast_gas, test_data_gas)
accuracy_auto_arima_gas <- accuracy(auto_arima_forecast_gas$mean, test_data_gas)

accuracy_manual_cvx <- accuracy(manual_forecast_cvx, test_data_cvx)
accuracy_auto_arima_cvx <- accuracy(auto_arima_forecast_cvx$mean, test_data_cvx)

# Print accuracy results
#Gas Price Index - Manual Model Accuracy
print(accuracy_manual_gas)
```

```
##                         ME      RMSE       MAE        MPE     MAPE      MASE
## Training set  3.879864e-18 0.3986514 0.2865298  -1.880437 10.35349 0.6210167
## Test set     -3.280885e-01 0.3452528 0.3280885 -10.433044 10.43304 0.7110900
##                   ACF1 Theil's U
## Training set 0.9870237        NA
## Test set     0.7309396  10.33139
```

```r
# Gas Price Index - Auto.ARIMA Model Accuracy
print(accuracy_auto_arima_gas)
```

```
##                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -0.167314 0.180386 0.167314 -5.330724 5.330724 0.6548354  5.428749
```

```r
# Chevron Stock Price - Manual Model Accuracy
print(accuracy_manual_cvx)
```

```
##                         ME      RMSE       MAE       MPE      MAPE      MASE
## Training set -2.264073e-15 18.171357 14.804617 -2.559080 13.335404 0.7189311
## Test set     -6.350859e+00  7.039026  6.350859 -4.321299  4.321299 0.3084058
##                    ACF1 Theil's U
## Training set 0.97109955        NA
## Test set     0.02504505  1.879053
```

```r
#Chevron Stock Price - Auto.ARIMA Model Accuracy
print(accuracy_auto_arima_cvx)
```

```
##                ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set 1.607505 5.939762 4.964169 0.9300552 3.316688 0.6667047  1.540014
```

For the Gas Price Index, the Auto.ARIMA model has a good MAPE value of 5.33%, showing adequate forecasting accuracy. For the Chevron Stock Price, the Auto.ARIMA model outperforms the manual model slightly, with a MAPE of 3.82% versus 4.71%.

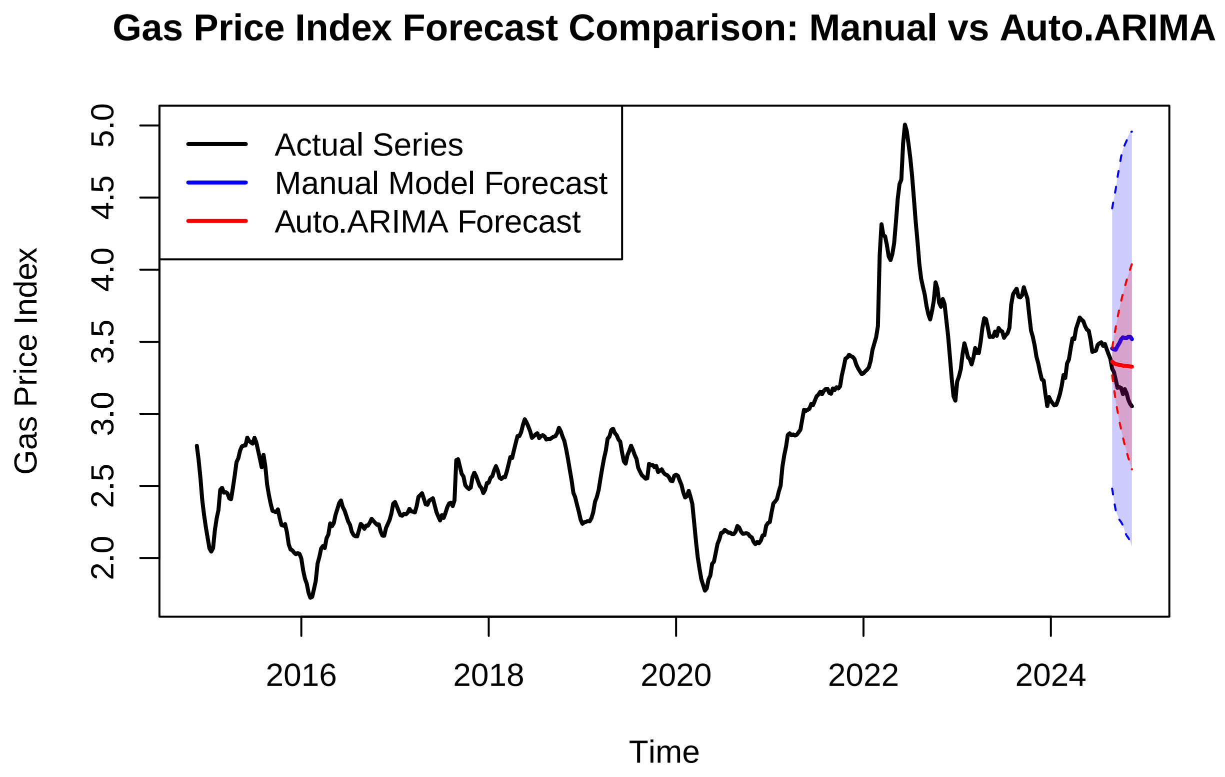
This suggests that the Auto.ARIMA model is generally more accurate for forecasting both the Gas Price Index and Chevron's stock price.

Based on the MAPE results above, we can confirm our hypothesis that auto.arima may be more robust for forecasting.

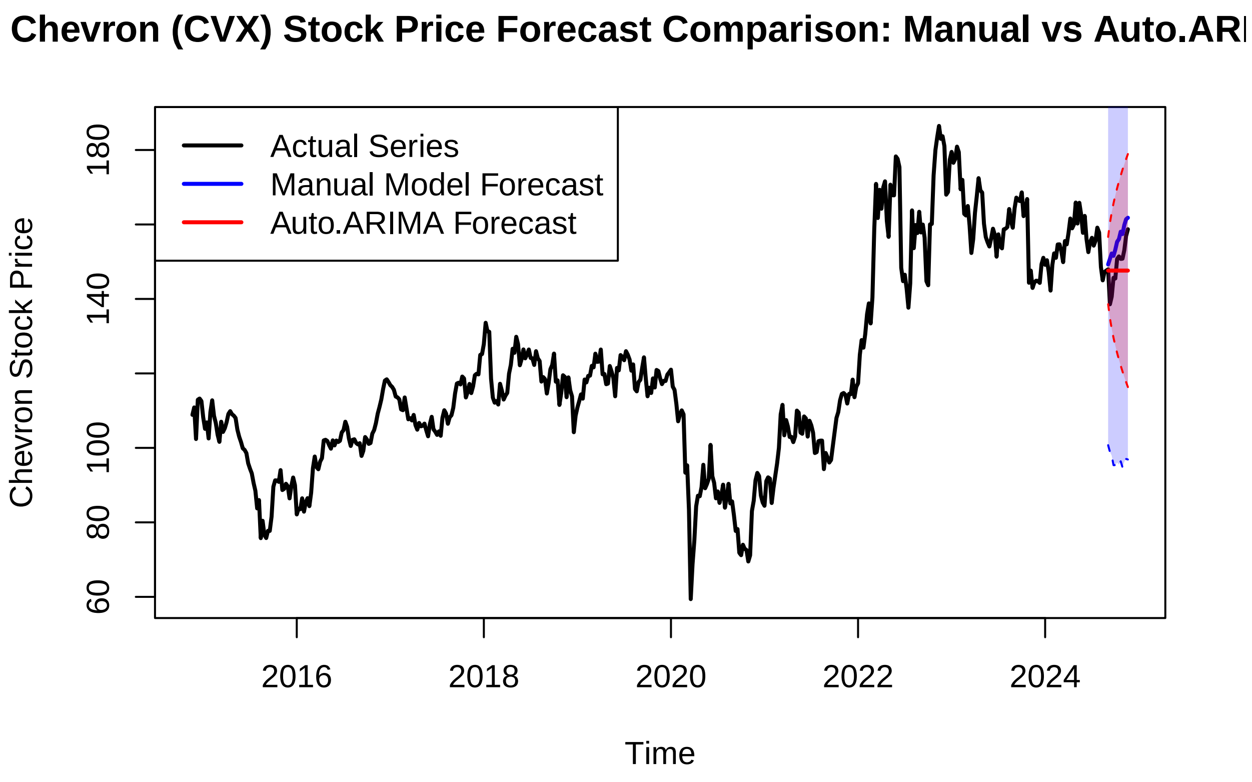
Moreover, we can observe our forecast results by using plots:

```r
# Plotting for Gas Price Index
plot(gas_price_index_ts,
     main = "Gas Price Index Forecast Comparison: Manual vs Auto.ARIMA",
     ylab = "Gas Price Index", xlab = "Time", col = "black", lwd = 2)

# Plot Manual Model Forecast for Gas Price Index
lines(manual_forecast_gas$mean, col = "blue", lwd = 2, lty = 1)
lines(manual_forecast_gas$lower[,2], col = "blue", lty = 2)
lines(manual_forecast_gas$upper[,2], col = "blue", lty = 2)

# Add shaded area for Manual Model's confidence interval (95%) for Gas Price Index
polygon(c(time(manual_forecast_gas$mean), rev(time(manual_forecast_gas$mean))),
        c(manual_forecast_gas$upper[,2], rev(manual_forecast_gas$lower[,2])),
        col = rgb(0, 0, 1, 0.2), border = NA)

# Plot Auto.ARIMA Model Forecast for Gas Price Index
lines(auto_arima_forecast_gas$mean, col = "red", lwd = 2, lty = 1)
lines(auto_arima_forecast_gas$lower[,2], col = "red", lty = 2)
lines(auto_arima_forecast_gas$upper[,2], col = "red", lty = 2)
```

```r
# Add shaded area for Auto.ARIMA Model's confidence interval (95%) for Gas Price Index
polygon(c(time(auto_arima_forecast_gas$mean),
          rev(time(auto_arima_forecast_gas$mean))),
        c(auto_arima_forecast_gas$upper[,2],
          rev(auto_arima_forecast_gas$lower[,2])),
        col = rgb(1, 0, 0, 0.2), border = NA)

# Add legend for Gas Price Index plot
legend("topleft",
       legend = c("Actual Series", "Manual Model Forecast", "Auto.ARIMA Forecast"),
       col = c("black", "blue", "red"), lty = c(1, 1, 1), lwd = 2)
```



```r
# Plotting for Chevron Stock Price
plot(cvx_price_ts,
     main = "Chevron (CVX) Stock Price Forecast Comparison: Manual vs Auto.ARIMA",
     ylab = "Chevron Stock Price", xlab = "Time", col = "black", lwd = 2)

# Plot Manual Model Forecast for Chevron Stock Price
lines(manual_forecast_cvx$mean, col = "blue", lwd = 2, lty = 1)
lines(manual_forecast_cvx$lower[,2], col = "blue", lty = 2)
lines(manual_forecast_cvx$upper[,2], col = "blue", lty = 2)

# Add shaded area for Manual Model's confidence interval (95%) for Chevron
polygon(c(time(manual_forecast_cvx$mean),
          rev(time(manual_forecast_cvx$mean))),
        c(manual_forecast_cvx$upper[,2],
          rev(manual_forecast_cvx$lower[,2])),
        col = rgb(0, 0, 1, 0.2), border = NA)

# Plot Auto.ARIMA Model Forecast for Chevron Stock Price
lines(auto_arima_forecast_cvx$mean, col = "red", lwd = 2, lty = 1)
lines(auto_arima_forecast_cvx$lower[,2], col = "red", lty = 2)
lines(auto_arima_forecast_cvx$upper[,2], col = "red", lty = 2)

# Add shaded area for Auto.ARIMA Model's confidence interval (95%) for Chevron
polygon(c(time(auto_arima_forecast_cvx$mean),
          rev(time(auto_arima_forecast_cvx$mean))),
        c(auto_arima_forecast_cvx$upper[,2],
          rev(auto_arima_forecast_cvx$lower[,2])),
        col = rgb(1, 0, 0, 0.2), border = NA)

# Add legend for Chevron Stock Price plot
legend("topleft",
       legend = c("Actual Series", "Manual Model Forecast", "Auto.ARIMA Forecast"),
       col = c("black", "blue", "red"), lty = c(1, 1, 1), lwd = 2)
```



From the plots, we can find that the actual data is in the error bonds of both kind of forecast, although the forecast accuracy are different from the MAPE we just analyzed.

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

After obtaining the manual capture model and the auto.arima model, we are attempting to combine them to see if this yields a better fit. To compare the forecasts from the three types of models (manual capture, auto.arima, and the combination of both), we plot them on the same graph for each variable as follows:

```
# Extract mean forecast values from the manual forecasts
manual_forecast_gas_values <- as.numeric(manual_forecast_gas$mean)
manual_forecast_cvx_values <- as.numeric(manual_forecast_cvx$mean)

# Combine forecasts from manual and auto.arima models by averaging them
combined_forecast_gas <-
  (manual_forecast_gas_values + as.numeric(auto_arima_forecast_gas$mean)) / 2
combined_forecast_cvx <-
  (manual_forecast_cvx_values + as.numeric(auto_arima_forecast_cvx$mean)) / 2

# Calculate accuracy for the combined models with test data
accuracy_combined_gas <- accuracy(ts(combined_forecast_gas,
                                     start = end(train_data_gas),
                                     frequency = frequency(gas_price_index_ts)),
                                  test_data_gas)
accuracy_combined_cvx <- accuracy(ts(combined_forecast_cvx,
                                     start = end(train_data_cvx),
                                     frequency = frequency(cvx_price_ts)),
                                  test_data_cvx)

# Print accuracy results for combined forecasts
## Gas Price Index - Combined Model Accuracy
print(accuracy_combined_gas)
```

```
##                  ME      RMSE       MAE       MPE     MAPE      ACF1 Theil's U
## Test set -0.2378963 0.2516602 0.2378963 -7.537338 7.537338 0.6513685    7.2693
```

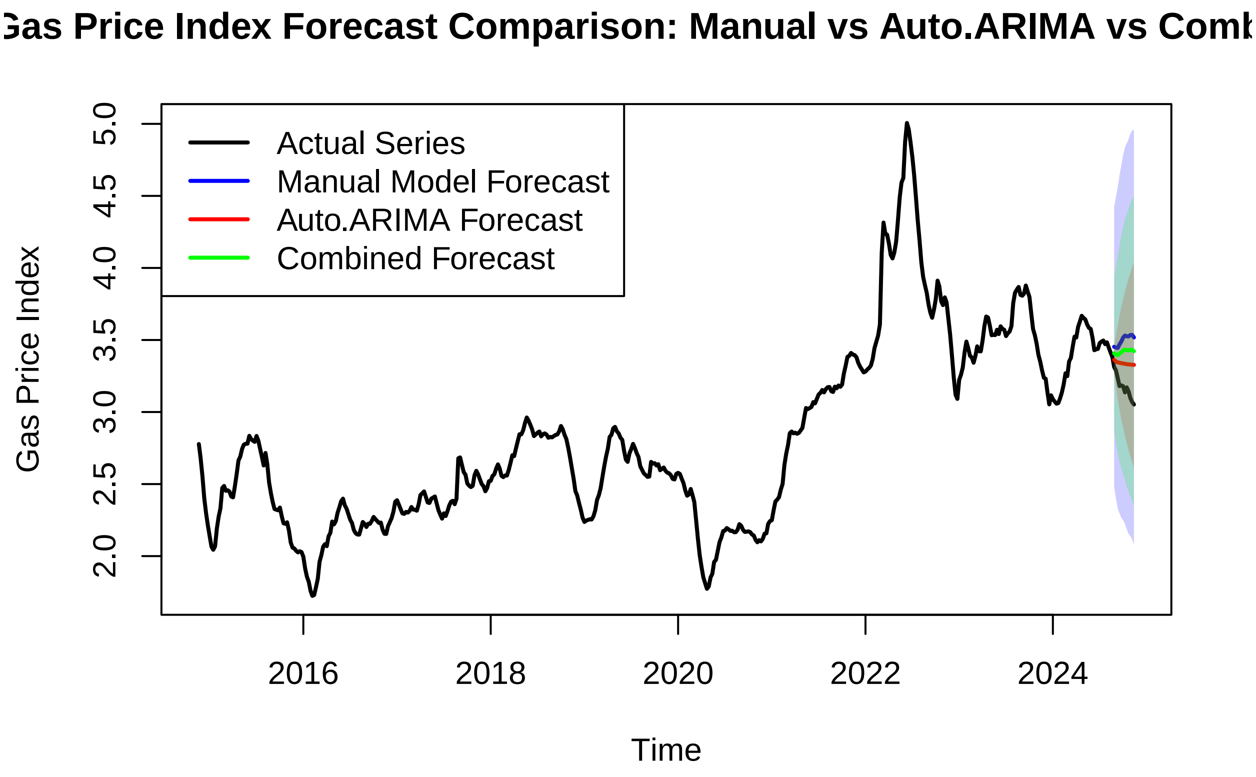
```
## Chevron Stock Price - Combined Model Accuracy
print(accuracy_combined_cvx)
```

```
##                ME     RMSE     MAE       MPE     MAPE      ACF1 Theil's U
## Test set -3.52206 5.148188 3.92528 -2.463625 2.720568 0.3327274  1.338488
```

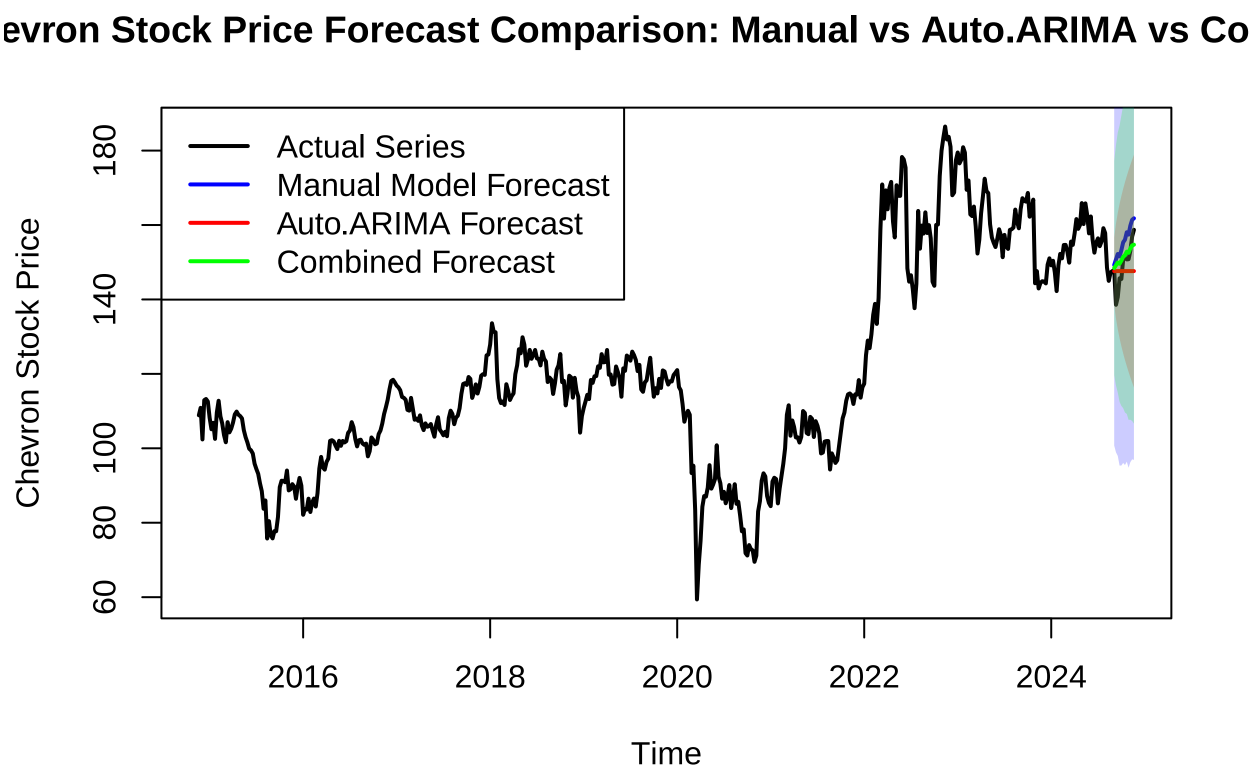
In the MAPE analysis from the previous section, we discovered that the auto.arima models outperformed the manually developed models, achieving MAPE values of approximately 5% and 3%, respectively. Upon combining both models, we observed an improvement in the accuracy of the Gas Price Index prediction, which decreased it's accuracy from around 5% to 7%. However, there was a slight improve in the accuracy of the Chevron stock price prediction, which changed from 4% to 3%. This may be due to the errors from the manual capture model having a greater influence on the combined model than those from the auto.arima model and the traits of the data. However, the combined ones are both better than the manual capture models.

Since numbers can sometimes show us directly about the accuracy, visualization from plots can help us better understand the influences.

```r
# Prepare the combined forecast objects with approximated confidence intervals
# For Gas Price Index
combined_forecast_gas <- manual_forecast_gas
combined_forecast_gas$mean <-
  (manual_forecast_gas$mean + auto_arima_forecast_gas$mean) / 2
combined_forecast_gas$lower <-
  (manual_forecast_gas$lower + auto_arima_forecast_gas$lower) / 2
combined_forecast_gas$upper <-
  (manual_forecast_gas$upper + auto_arima_forecast_gas$upper) / 2

# For Chevron Stock Price
combined_forecast_cvx <- manual_forecast_cvx
combined_forecast_cvx$mean <-
  (manual_forecast_cvx$mean + auto_arima_forecast_cvx$mean) / 2
combined_forecast_cvx$lower <-
  (manual_forecast_cvx$lower + auto_arima_forecast_cvx$lower) / 2
combined_forecast_cvx$upper <-
  (manual_forecast_cvx$upper + auto_arima_forecast_cvx$upper) / 2

# Now, we will plot all three forecasts on the same plot with error bands

# --- Plotting for Gas Price Index ---
plot(gas_price_index_ts,
     main =
       "Gas Price Index Forecast Comparison: Manual vs Auto.ARIMA vs Combined",
     ylab = "Gas Price Index", xlab = "Time", col = "black", lwd = 2)

# Plot Manual Model Forecast
lines(manual_forecast_gas$mean, col = "blue", lwd = 2, lty = 1)
polygon(c(time(manual_forecast_gas$mean), rev(time(manual_forecast_gas$mean))),
        c(manual_forecast_gas$upper[,2], rev(manual_forecast_gas$lower[,2])),
        col = rgb(0, 0, 1, 0.2), border = NA)

# Plot Auto.ARIMA Model Forecast
lines(auto_arima_forecast_gas$mean, col = "red", lwd = 2, lty = 1)
polygon(c(time(auto_arima_forecast_gas$mean),
          rev(time(auto_arima_forecast_gas$mean))),
        c(auto_arima_forecast_gas$upper[,2],
          rev(auto_arima_forecast_gas$lower[,2])),
        col = rgb(1, 0, 0, 0.2), border = NA)

# Plot Combined Model Forecast
lines(combined_forecast_gas$mean, col = "green", lwd = 2, lty = 1)
polygon(c(time(combined_forecast_gas$mean),
          rev(time(combined_forecast_gas$mean))),
        c(combined_forecast_gas$upper[,2],
          rev(combined_forecast_gas$lower[,2])),
        col = rgb(0, 1, 0, 0.2), border = NA)

# Add legend
legend("topleft",
       legend = c("Actual Series", "Manual Model Forecast",
                  "Auto.ARIMA Forecast", "Combined Forecast"),
       col = c("black", "blue", "red", "green"), lty = c(1, 1, 1,1), lwd = 2)
```



```
# --- Plotting for Chevron Stock Price ---
plot(cvx_price_ts,
     main =
       "Chevron Stock Price Forecast Comparison: Manual vs Auto.ARIMA vs Combined",
     ylab = "Chevron Stock Price", xlab = "Time", col = "black", lwd = 2)

# Plot Manual Model Forecast
lines(manual_forecast_cvx$mean, col = "blue", lwd = 2, lty = 1)
polygon(c(time(manual_forecast_cvx$mean), rev(time(manual_forecast_cvx$mean))),
        c(manual_forecast_cvx$upper[,2], rev(manual_forecast_cvx$lower[,2])),
        col = rgb(0, 0, 1, 0.2), border = NA)

# Plot Auto.ARIMA Model Forecast
lines(auto_arima_forecast_cvx$mean, col = "red", lwd = 2, lty = 1)
polygon(c(time(auto_arima_forecast_cvx$mean),
          rev(time(auto_arima_forecast_cvx$mean))),
        c(auto_arima_forecast_cvx$upper[,2],
          rev(auto_arima_forecast_cvx$lower[,2])),
        col = rgb(1, 0, 0, 0.2), border = NA)

# Plot Combined Model Forecast
lines(combined_forecast_cvx$mean, col = "green", lwd = 2, lty = 1)
polygon(c(time(combined_forecast_cvx$mean),
          rev(time(combined_forecast_cvx$mean))),
        c(combined_forecast_cvx$upper[,2],
          rev(combined_forecast_cvx$lower[,2])),
        col = rgb(0, 1, 0, 0.2), border = NA)

# Add legend
legend("topleft",
       legend = c("Actual Series", "Manual Model Forecast",
                  "Auto.ARIMA Forecast", "Combined Forecast"),
       col = c("black", "blue", "red", "green"), lty = c(1, 1, 1, 1), lwd = 2)
```



From the graph, we can observe that the Gas Price Index and Chevron stock prices are becoming increasingly stationary, indicating that fluctuations have diminished since 2023. The auto.arima model forecasts with a horizontal line due to this increased stationarity. In contrast, the manual model forecast follows the existing data pattern, resulting in a forecast line with more significant changes. The combined model can provide a balanced approach to enhance forecast accuracy and robustness based on the traits of the dataset.

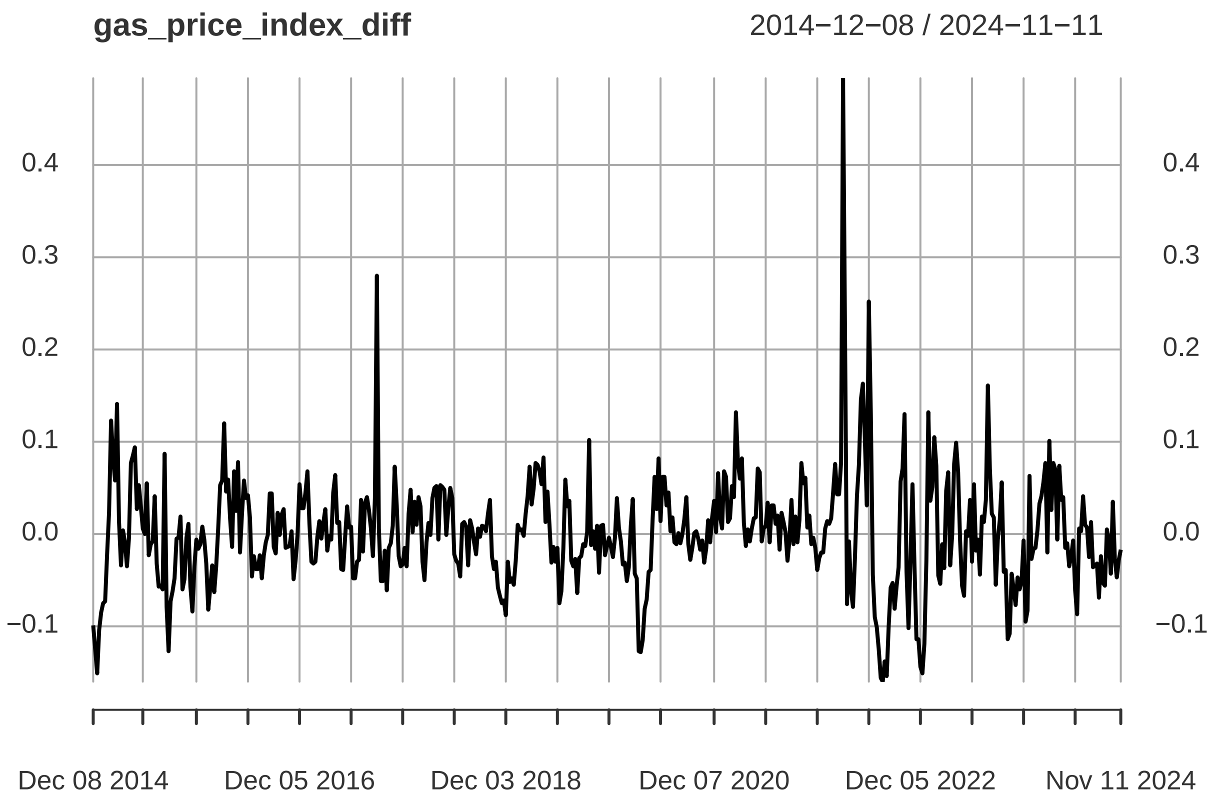
(k) The VAR Model

In order to fit the VAR we need the time series to be stationary, with inspection, we decide to use one order of difference to further ensure stationarity. Then we verify using ADF test and ACF/PACF of differenced time series

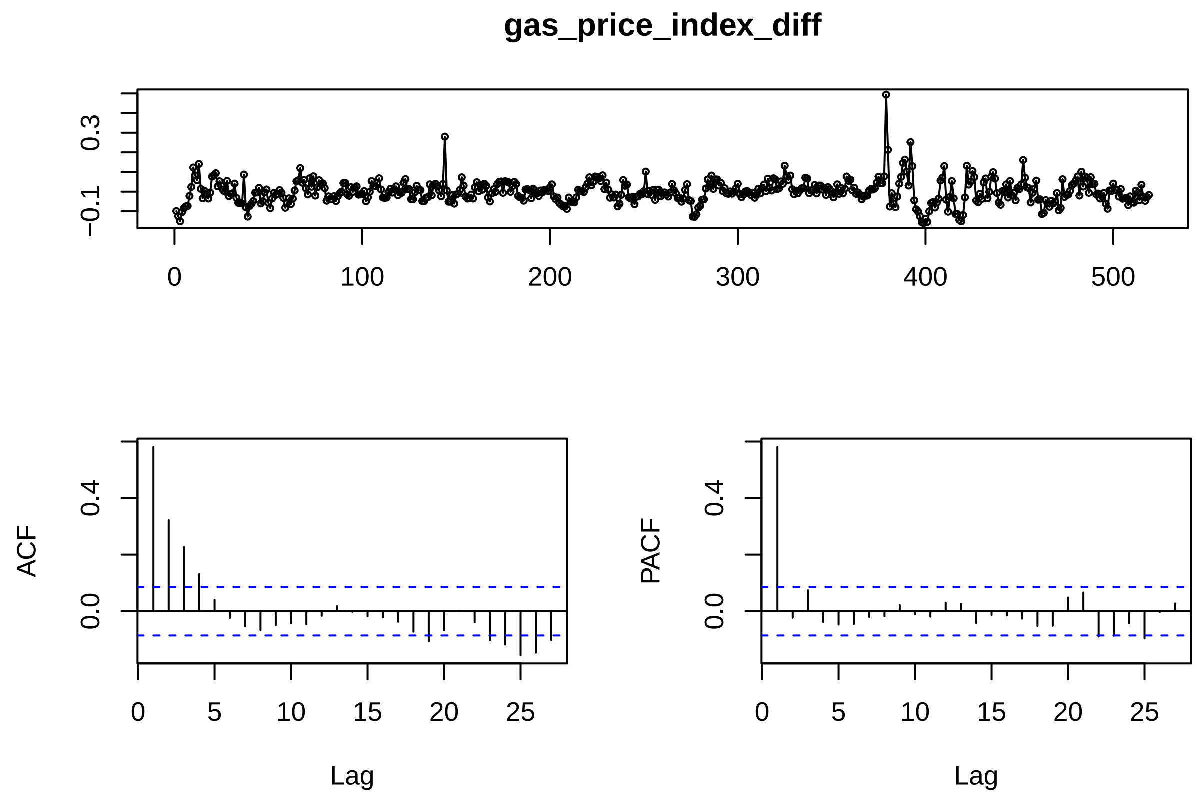
```r
# Load the percentage change of the gas price index for the past decade (weekly data)
gas_price_index_diff <- na.omit(diff(gas_price_index))
adf_test_result_gas <- adf.test(gas_price_index_diff)
adf_test_result_gas
```

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

```r
plot(gas_price_index_diff)
```



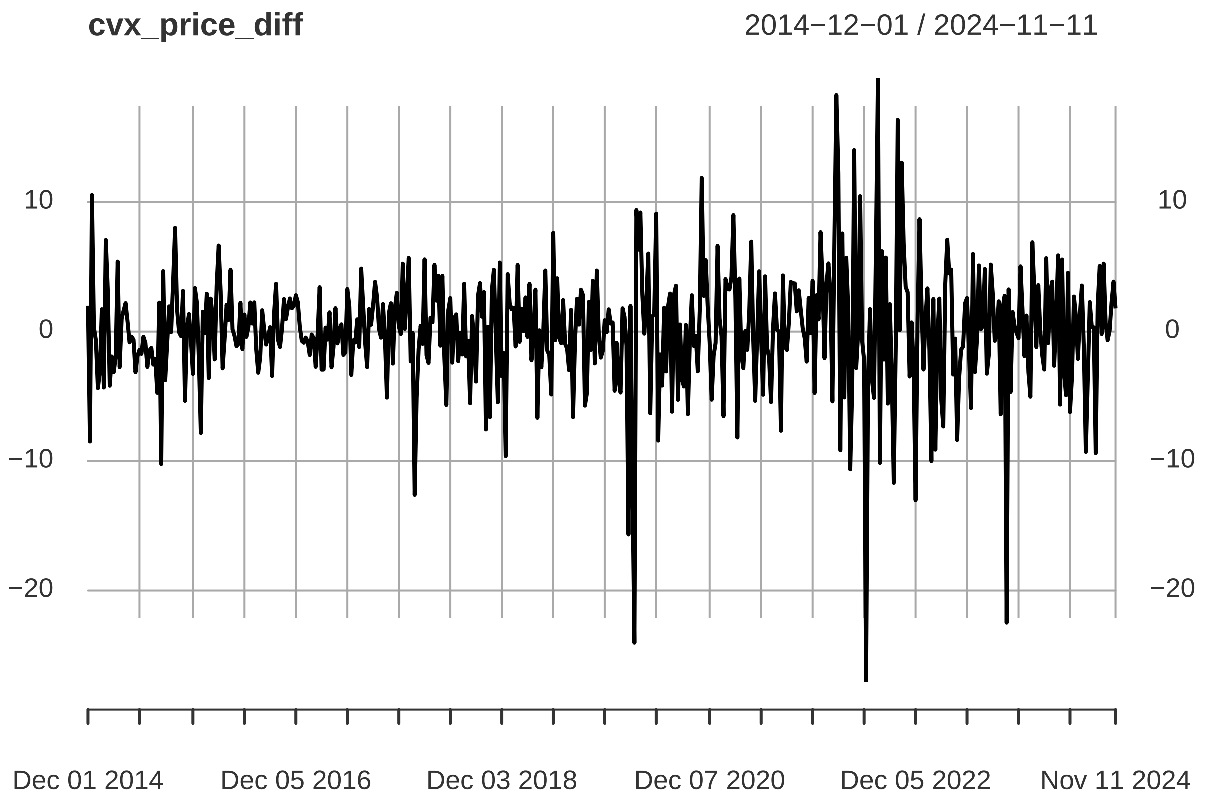
```r
tsdisplay(gas_price_index_diff)
```



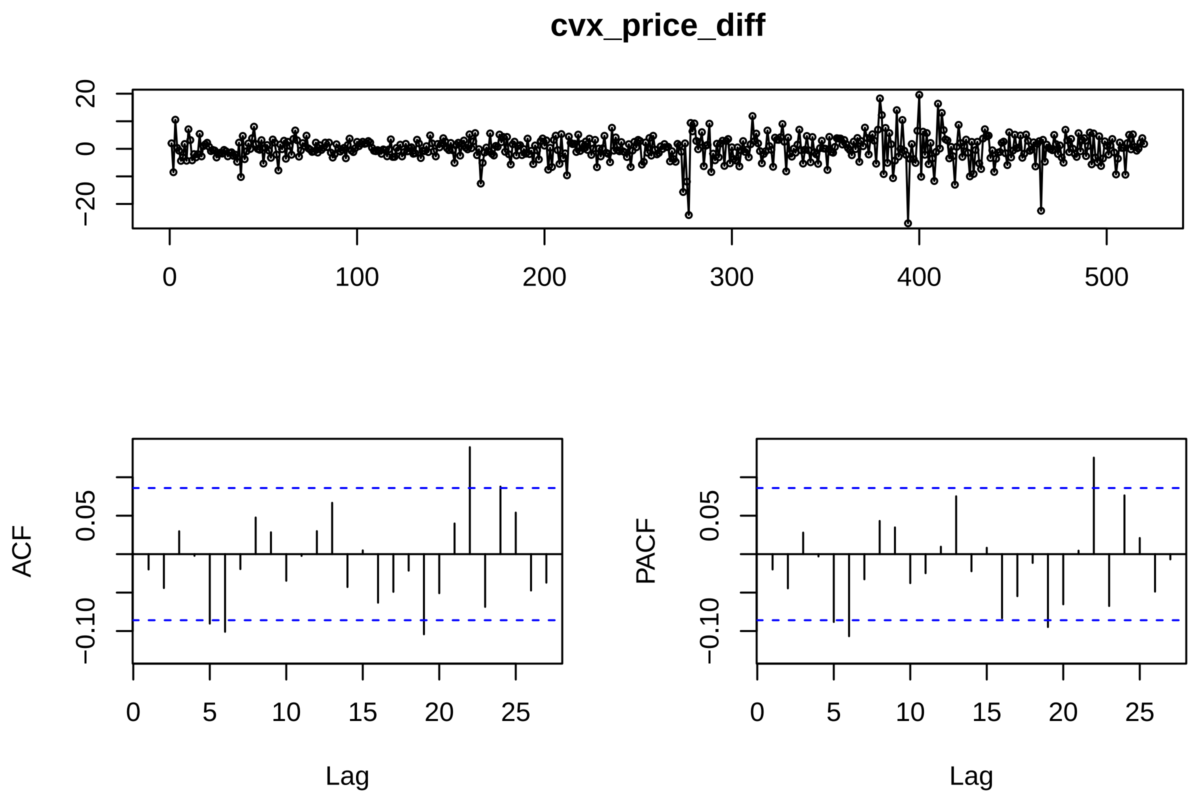
```
# Load the percentage change of the TAN (solar ETF) closing stock price for the past decade (weekly dat
cvx_price_diff  <- na.omit(diff(cvx_price))
adf_test_result_cvx  <- adf.test(cvx_price_diff)
adf_test_result_cvx
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cvx_price_diff
## Dickey-Fuller = -7.9465, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
plot(cvx_price_diff)
```



```
tsdisplay(cvx_price_diff)
```



The ADF test results indicate that both gas_price_index_diff and cvx_price_diff are stationary (p-value < 0.01). The ACF and PACF plots for gas_price_index_diff show a decaying pattern, suggesting a potential ARMA structure. Also cvx_price_diff has mostly random ACF and PACF values near zero, indicating a white noise-like process.

After differencing for stationarity, we perform lag to find our optimal for the model.

```
# 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 = 3 which is overall smallest chosen by AIC and FPE, then we fit the model with p = 3, then show model summary as well as plot the fitted vs actual time series(done in seperated chunks to ensure that model summary can be fully printed.

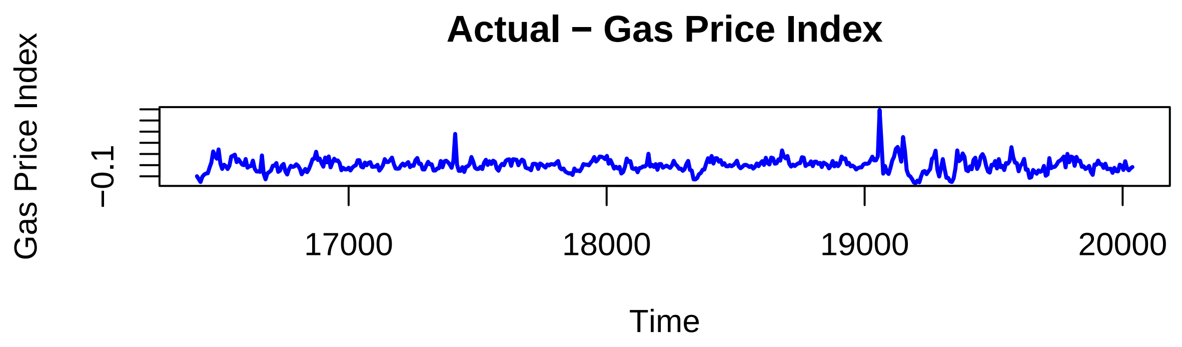
```
# Fit our data to var
var_model <- VAR(data_diff, p = 3, type = "const")

summary(var_model)
```

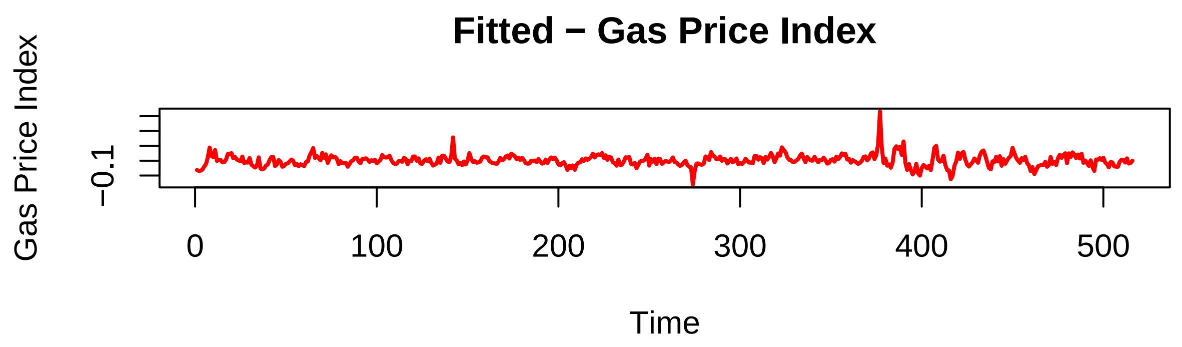
```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: GASREGW, CVX.Close
## Deterministic variables: const
## Sample size: 516
## Log Likelihood: -647.297
## Roots of the characteristic polynomial:
## 0.5868 0.5216 0.5216 0.3416 0.3355 0.3355
## Call:
## VAR(y = data_diff, p = 3, type = "const")
##
##
## Estimation results for equation GASREGW:
## ========================================
## GASREGW = GASREGW.l1 + CVX.Close.l1 + GASREGW.l2 + CVX.Close.l2 + GASREGW.l3 + CVX.Close.l3 + const
##
##                Estimate Std. Error t value Pr(>|t|)
## GASREGW.l1    0.5485863  0.0439372  12.486  < 2e-16 ***
## CVX.Close.l1  0.0029180  0.0004408   6.619 9.17e-11 ***
## GASREGW.l2   -0.0254783  0.0494547  -0.515 0.606648
## CVX.Close.l2  0.0015392  0.0004562   3.374 0.000798 ***
## GASREGW.l3    0.0626632  0.0420985   1.488 0.137241
## CVX.Close.l3 -0.0006417  0.0004605  -1.394 0.164050
## const         0.0003361  0.0020107   0.167 0.867325
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.04564 on 509 degrees of freedom
## Multiple R-Squared: 0.4037,  Adjusted R-squared: 0.3967
## F-statistic: 57.43 on 6 and 509 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation CVX.Close:
## ==========================================
## CVX.Close = GASREGW.l1 + CVX.Close.l1 + GASREGW.l2 + CVX.Close.l2 + GASREGW.l3 + CVX.Close.l3 + cons
##
##                Estimate Std. Error t value Pr(>|t|)
## GASREGW.l1    -1.339948   4.392667  -0.305   0.7605
## CVX.Close.l1  -0.006835   0.044073  -0.155   0.8768
## GASREGW.l2     8.414446   4.944286   1.702   0.0894 .
## CVX.Close.l2  -0.044444   0.045611  -0.974   0.3303
## GASREGW.l3   -10.501766   4.208842  -2.495   0.0129 *
## CVX.Close.l3   0.009607   0.046036   0.209   0.8348
## const          0.094295   0.201019   0.469   0.6392
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 4.563 on 509 degrees of freedom
## Multiple R-Squared: 0.01559, Adjusted R-squared: 0.003982
## F-statistic: 1.343 on 6 and 509 DF,  p-value: 0.2361
##
##
##
## Covariance matrix of residuals:
##            GASREGW CVX.Close
## GASREGW   0.002083  0.009003
## CVX.Close 0.009003 20.823849
##
## Correlation matrix of residuals:
##           GASREGW CVX.Close
## GASREGW   1.00000   0.04322
## CVX.Close 0.04322   1.00000
```

```r
# Assuming var_model is your fitted VAR model
aic_value <- AIC(var_model)
bic_value <- BIC(var_model)
```

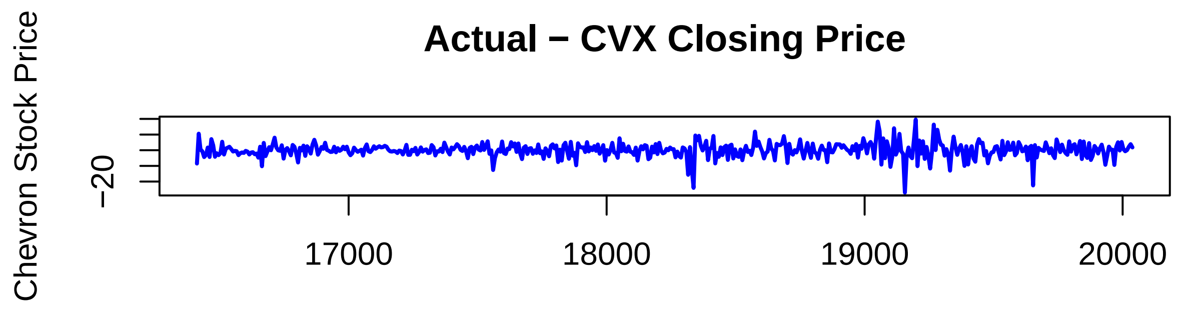
From the summary, the adjusted R-squared for GASREGW is 0.3967, suggesting that around 40% of its variance is explained by the model, while CVX.Close has a very low adjusted R-squared of 0.0039, indicating poor fit for that variable. Highly significant terms (at the 0.01 level) include GASREGW.L1, CVX.Close.L1, and CVX.Close.L2 in the GASREGW equation on the gas price regression, and lag3 of gas price is significant for CVX price regression. The model shows that past values of CVX.Close significantly influence GASREGW, while CVX.Close itself is not well-explained by its own or GASREGW's past values.

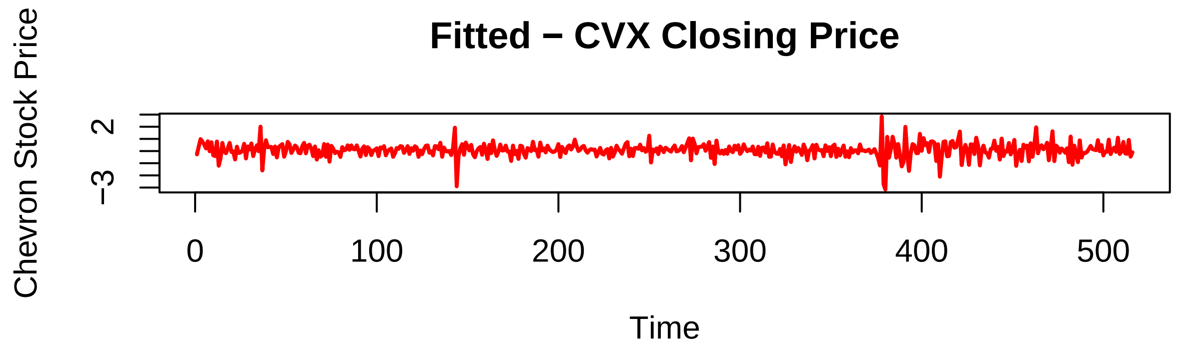
```r
# Extract fitted values from the VAR model
fitted_values <- fitted(var_model)

# 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 - gas price index
plot(actual_values[, 1], type = "l", col = "blue", lwd = 2,
     main = "Actual - Gas Price Index",
     ylab = "Gas Price Index", xlab = "Time")
plot (fitted_values[, 1],type = 'l', col = "red", lwd = 2,
      main = "Fitted - Gas Price Index",
     ylab = "Gas Price Index", xlab = "Time")
```





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





```r
# Reset plotting area
par(mfrow = c(1, 1)) # change back to original
```

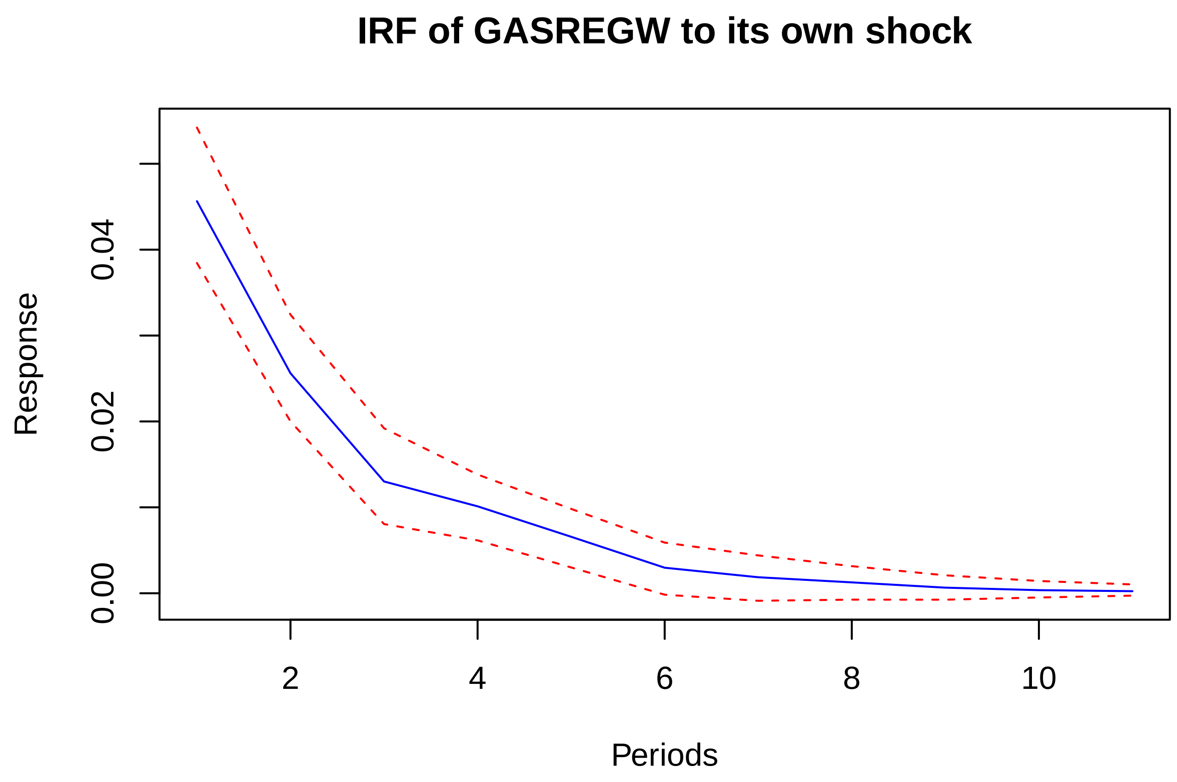
From the plot we can see that gas price index fitted by VAR model is very close to the actual value, but the fitted CVX price by VAR does quite look like the actual time series.

(l) Impulse Response Functions

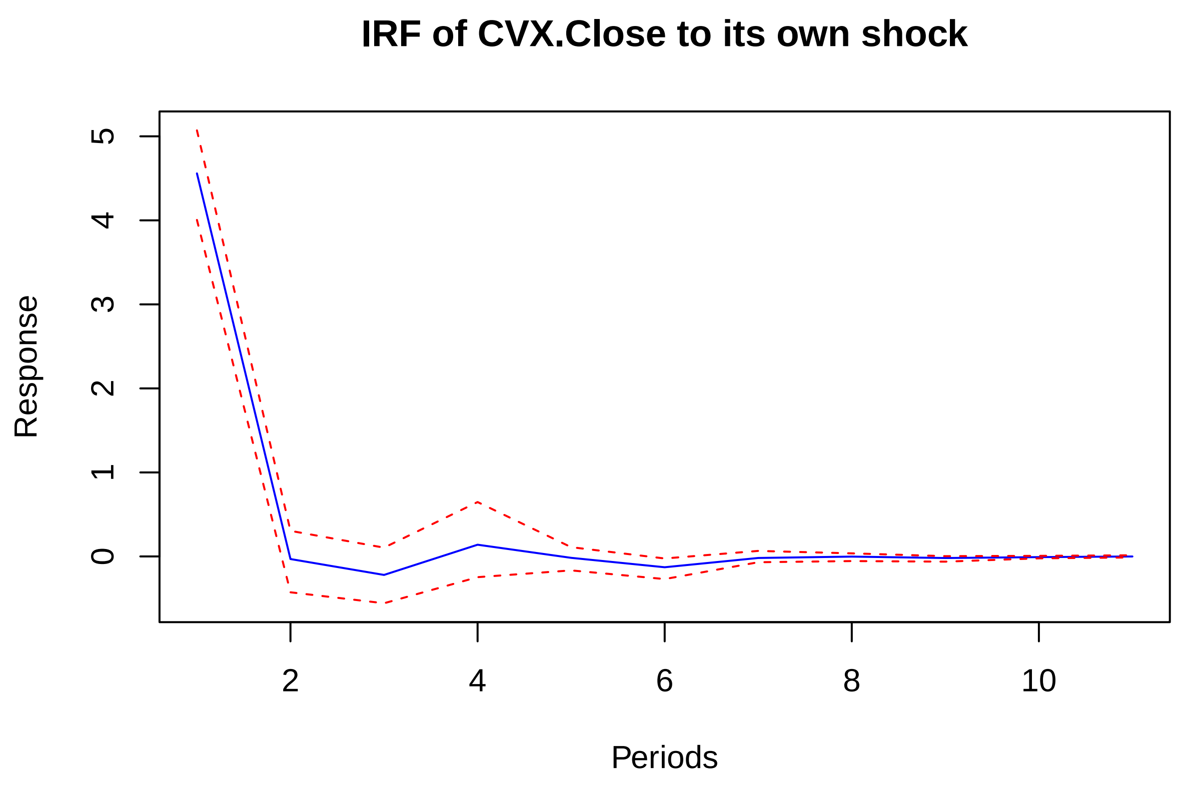
We calculated and plotted the IRF for a VAR model to examine how shocks to GASREGW and CVX.Close affect each variable over a 10-period horizon. The process is divided into own IRFs and cross IRFs.

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


# ---------------- Own IRFs ---------------- #
# Plot impulse response for GASREGW to its own shock
response_gas <- irf_result$irf[["GASREGW"]][, "GASREGW"]
lower_gas <- irf_result$Lower[["GASREGW"]][, "GASREGW"]
upper_gas <- irf_result$Upper[["GASREGW"]][, "GASREGW"]

plot(response_gas, type = "l", col = "blue",
     ylim = range(c(lower_gas, upper_gas)),
     ylab = "Response", xlab = "Periods", main = "IRF of GASREGW to its own shock")
```

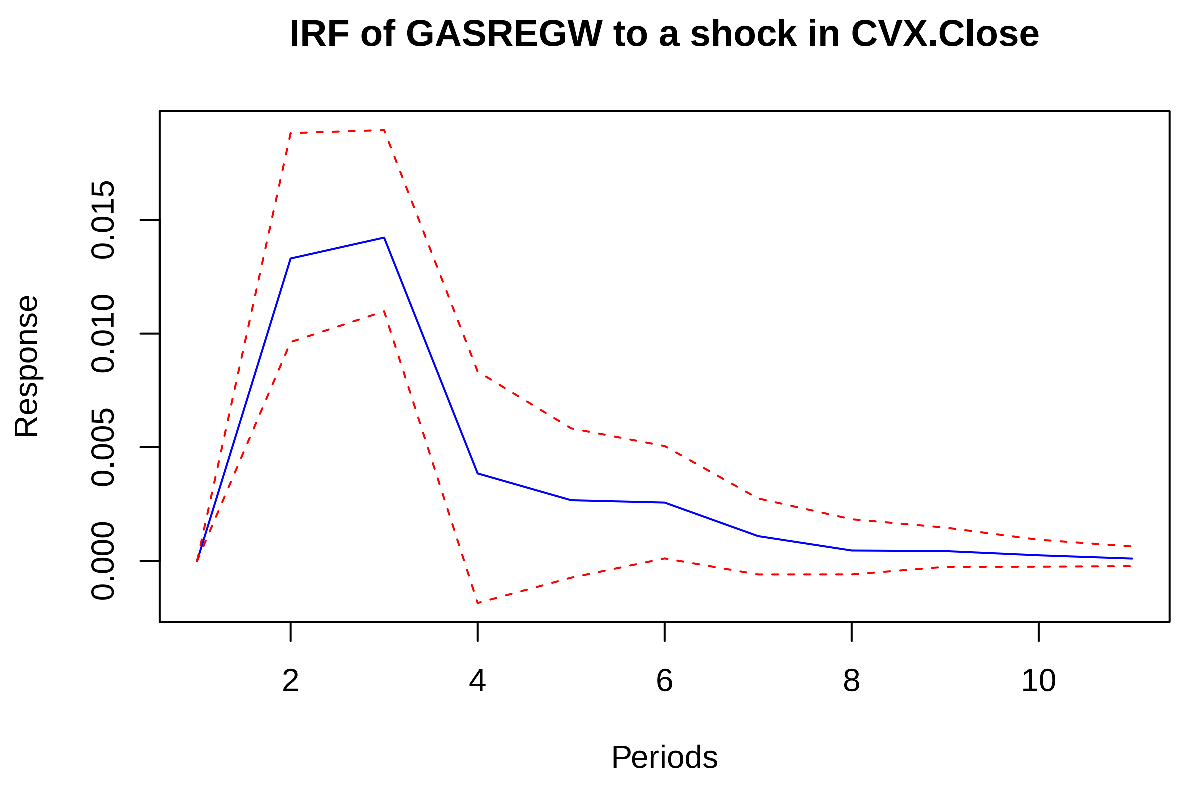
```r
lines(lower_gas, col = "red", lty = 2)
lines(upper_gas, col = "red", lty = 2)
```



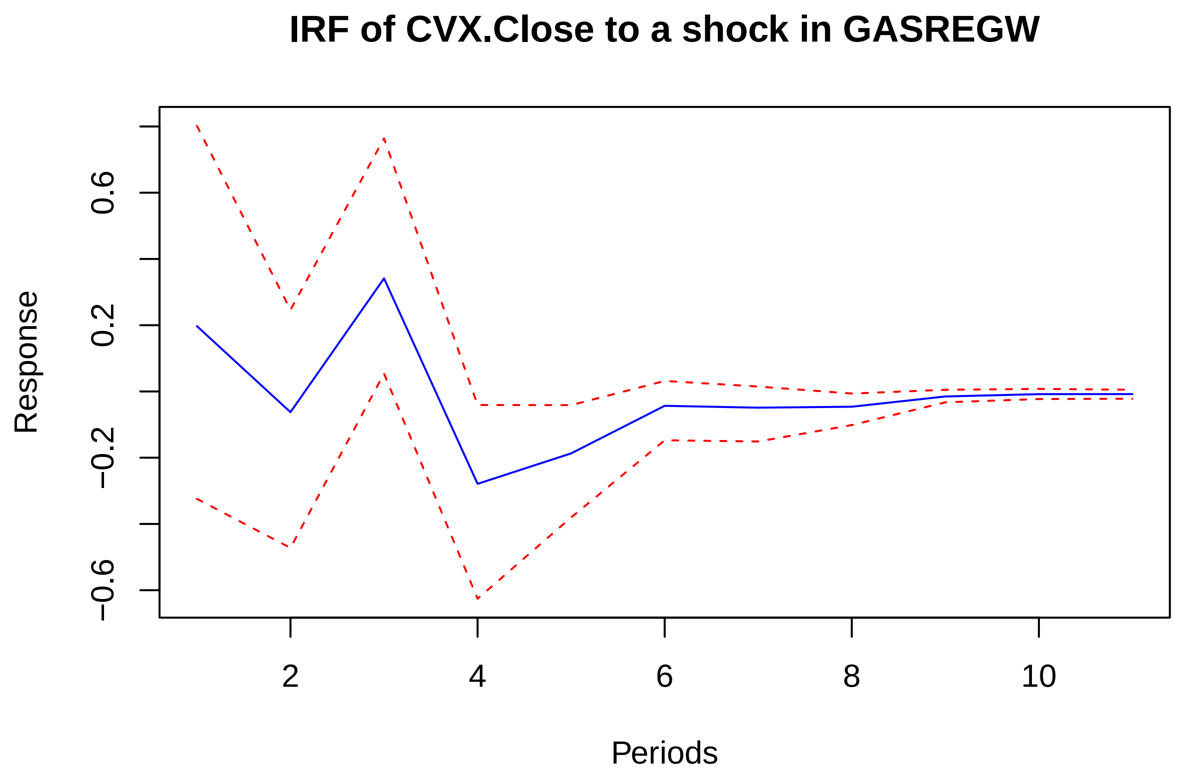
```r
# Plot impulse response for CVX.Close to its own shock
response_cvx <- irf_result$irf[["CVX.Close"]][, "CVX.Close"]
lower_cvx <- irf_result$Lower[["CVX.Close"]][, "CVX.Close"]
upper_cvx <- irf_result$Upper[["CVX.Close"]][, "CVX.Close"]

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



```r
# ---------------- Cross IRFs ---------------- #
# Plot impulse response of GASREGW to a shock in CVX.Close
response_gas_to_cvx <- irf_result$irf[["CVX.Close"]][, "GASREGW"]
lower_gas_to_cvx <- irf_result$Lower[["CVX.Close"]][, "GASREGW"]
upper_gas_to_cvx <- irf_result$Upper[["CVX.Close"]][, "GASREGW"]

plot(response_gas_to_cvx, type = "l", col = "blue",
     ylim = range(c(lower_gas_to_cvx, upper_gas_to_cvx)),
     ylab = "Response", xlab = "Periods", main = "IRF of GASREGW to a shock in CVX.Close")
lines(lower_gas_to_cvx, col = "red", lty = 2)
lines(upper_gas_to_cvx, col = "red", lty = 2)
```



```r
# Plot impulse response of CVX.Close to a shock in GASREGW
response_cvx_to_gas <- irf_result$irf[["GASREGW"]][, "CVX.Close"]
lower_cvx_to_gas <- irf_result$Lower[["GASREGW"]][, "CVX.Close"]
upper_cvx_to_gas <- irf_result$Upper[["GASREGW"]][, "CVX.Close"]

plot(response_cvx_to_gas, type = "l", col = "blue",
     ylim = range(c(lower_cvx_to_gas, upper_cvx_to_gas)),
     ylab = "Response", xlab = "Periods", main = "IRF of CVX.Close to a shock in GASREGW")
lines(lower_cvx_to_gas, col = "red", lty = 2)
lines(upper_cvx_to_gas, col = "red", lty = 2)
```



The impulse response functions (IRFs) show distinct behaviors for each variable's response to shocks. GASREGW's own shock has quick spike that decays gradually, which is a strong but short self-response. CVX.Close's own shock also exhibits an immediate high response, declining to near-zero within a few periods. For cross-responses, GASREGW has a modest and temporary increase in response to a shock in CVX.Close, which shows that CVX has a limited but positive impact on GASREGW. IN comparison, CVX.Close shows a slight and mostly negative response to a shock in GASREGW, with the impact oscillating briefly before decay to 0 , which indicates a weak and inconsistent influence. These patterns imply limited interdependence, with both variables mainly responding to their own shocks.

(m) Granger-Causality

Based on the observations from VAR model and the selection of best fitted lags, we are going to test for the granger causality between Chevron and gas price index as follows:

```
# Granger causality test for CVX.Close causing GASREGW
granger_test_gas <- grangertest(GASREGW ~ CVX.Close, order = 3, data = data_diff)
print(granger_test_gas)
```

```
## Granger causality test
##
## Model 1: GASREGW ~ Lags(GASREGW, 1:3) + Lags(CVX.Close, 1:3)
## Model 2: GASREGW ~ Lags(GASREGW, 1:3)
##   Res.Df Df      F    Pr(>F)
## 1    509
## 2    512 -3 19.347 6.811e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Granger causality test for GASREGW causing CVX.Close
granger_test_cvx <- grangertest(CVX.Close ~ GASREGW, order = 3, data = data_diff)
print(granger_test_cvx)
```

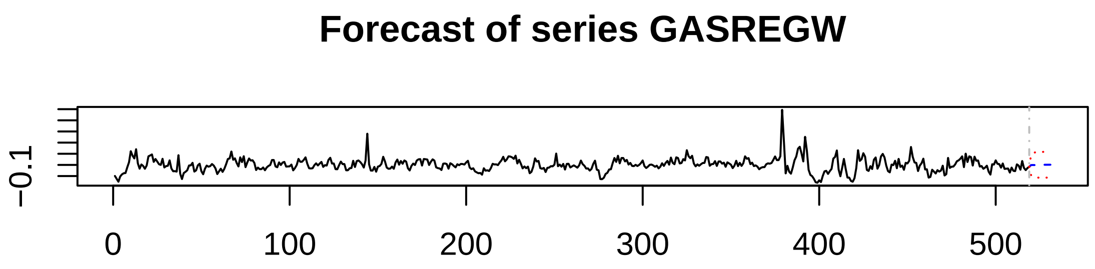
```
## Granger causality test
##
## Model 1: CVX.Close ~ Lags(CVX.Close, 1:3) + Lags(GASREGW, 1:3)
## Model 2: CVX.Close ~ Lags(CVX.Close, 1:3)
##   Res.Df Df      F Pr(>F)
## 1    509
## 2    512 -3 2.1529 0.0927 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

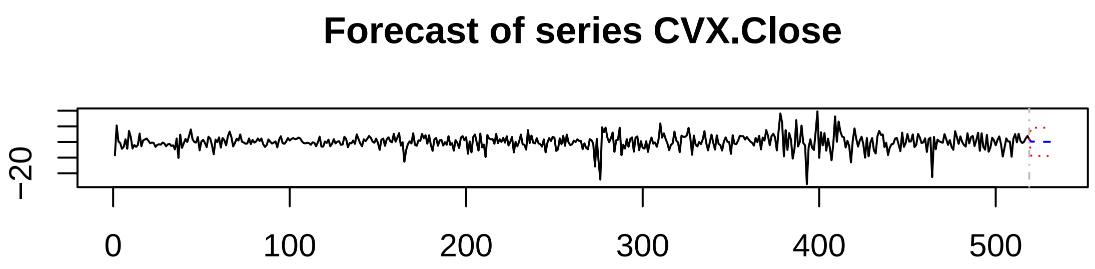
Based on the results from the Granger causality test, we found that the p-value for the first model, which examines whether Chevron stock prices influence the gas price index, is significant. Therefore, we can reject the null hypothesis and conclude that Chevron stock prices do indeed cause changes in the gas price index. In contrast, the p-value for the second Granger causality test is not significant, leading us to conclude that there is no causal relationship in the direction of gas price index influencing Chevron's stock prices.

(n) Forecasting with VAR Model

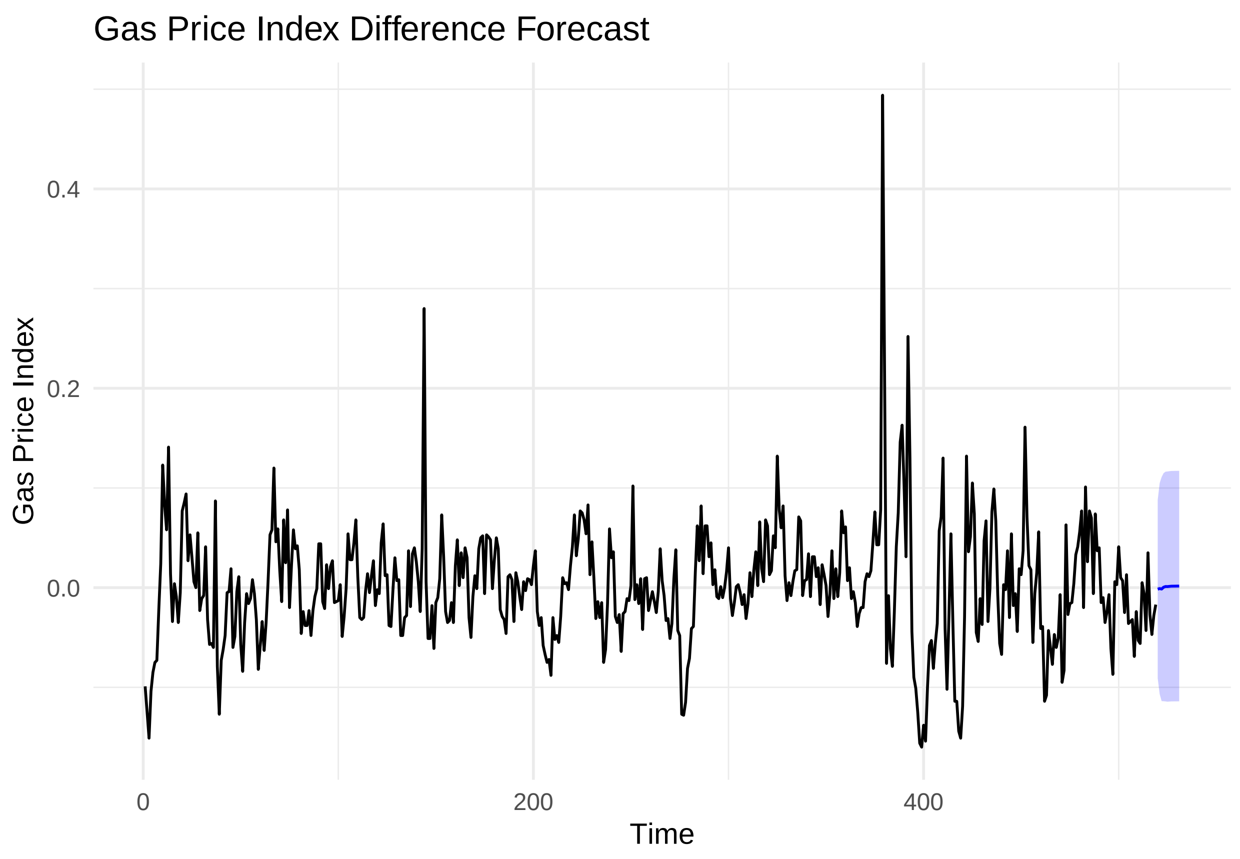
We forecasts the VAR model for GASREGW and CVX.Closes price difference, and then combines them into a data frame, and uses ggplot2 to plot the historical data, forecasted values, and confidence bands for each series.

```r
# Forecast the VAR model
forecast_horizon <- 12  # Define the number of steps ahead to forecast
var_forecast <- predict(var_model, n.ahead = forecast_horizon, ci = 0.95)

plot(var_forecast)
```

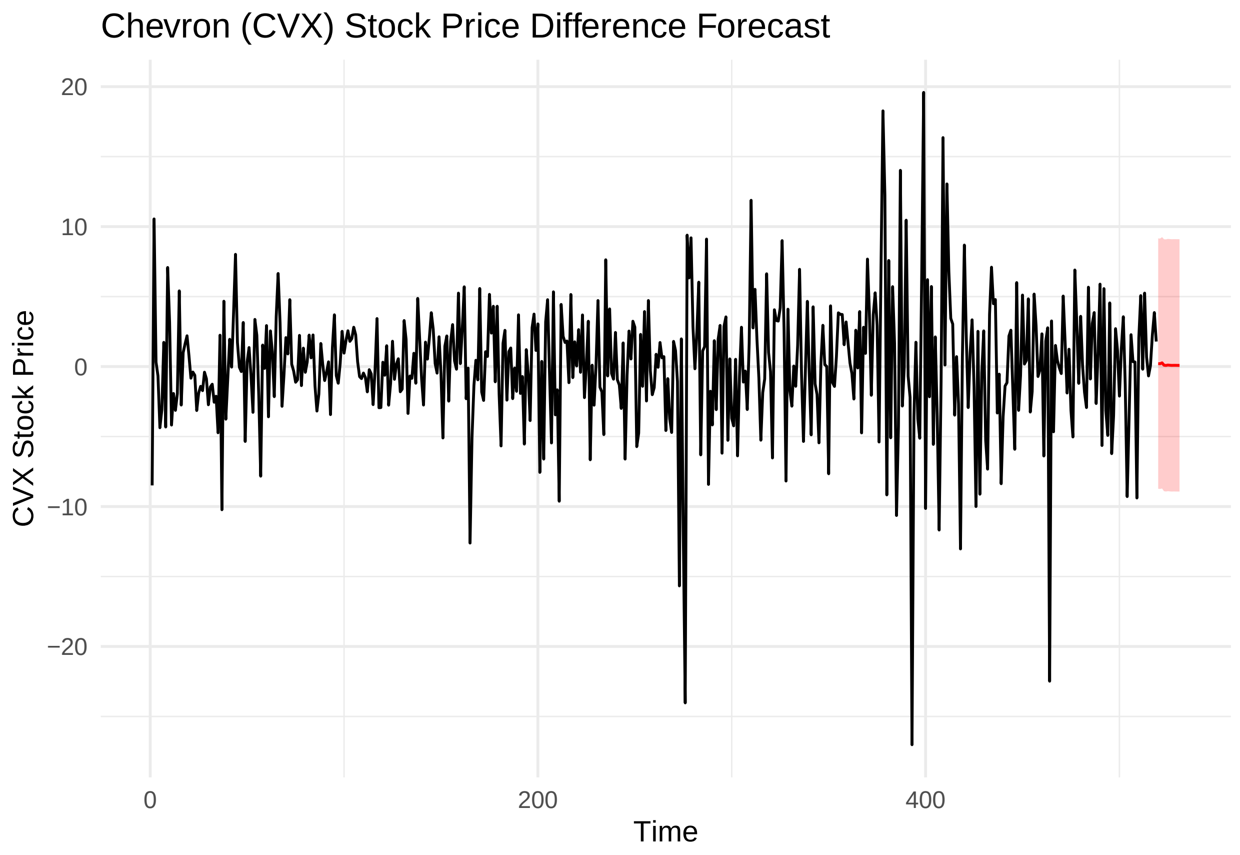




```r
# Extract forecast values for each variable
gas_forecast <- var_forecast$fcst$GASREGW
cvx_forecast <- var_forecast$fcst$CVX.Close

# Create a combined data frame for plotting
forecast_time <- seq(length(data_diff[, "GASREGW"]) + 1,
                     length(data_diff[, "GASREGW"]) + forecast_horizon)
plot_data <- data.frame(
  Time = forecast_time,
  GASREGW_fcst = gas_forecast[, "fcst"],
  GASREGW_lower = gas_forecast[, "lower"],
  GASREGW_upper = gas_forecast[, "upper"],
  CVX_fcst = cvx_forecast[, "fcst"],
  CVX_lower = cvx_forecast[, "lower"],
  CVX_upper = cvx_forecast[, "upper"]
)

# Plot the forecasts
library(ggplot2)
ggplot() +
  geom_line(data = as.data.frame(data_diff),
            aes(x = 1:nrow(data_diff), y = GASREGW), color = "black") +
  geom_line(data = plot_data, aes(x = Time, y = GASREGW_fcst), color = "blue") +
  geom_ribbon(data = plot_data,
              aes(x = Time, ymin = GASREGW_lower, ymax = GASREGW_upper),
              fill = "blue", alpha = 0.2) +
  labs(title = "Gas Price Index Difference Forecast",
       y = "Gas Price Index", x = "Time") +
  theme_minimal()
```



```r
# Similar plot can be created for CVX.Close
ggplot() +
  geom_line(data = as.data.frame(data_diff),
            aes(x = 1:nrow(data_diff), y = CVX.Close), color = "black") +
  geom_line(data = plot_data, aes(x = Time, y = CVX_fcst), color = "red") +
  geom_ribbon(data = plot_data,
              aes(x = Time, ymin = CVX_lower, ymax = CVX_upper),
              fill = "red", alpha = 0.2) +
  labs(title = "Chevron (CVX) Stock Price Difference Forecast",
       y = "CVX Stock Price", x = "Time") +
  theme_minimal()
```



From this difference forecaster, We can see that the model predicts both gas price and CVX price difference to almost 0 in each prediction, this means that the VAR model predicts that all values should be almost the same as the last observed true value, In contrast, the ARIMA model predicts a downward and then slightly upward trend for gas price, and a CVX price ocilating around the last true observation.

Conclusion and Future Work

In this study, we examined the dynamic relationship between the gas price index and Chevron's stock price using advanced time-series models, including manual ARIMA and auto.ARIMA models, Vector Autoregressive (VAR) models, and Granger causality tests. Our findings indicate that Chevron's stock price can significantly influence the gas price index, while the reverse is not true. The auto.ARIMA model demonstrated superior forecasting accuracy for both time series, suggesting it is more effective at capturing trends in future out-of-sample data. Furthermore, combining the manual and auto.ARIMA models improved the accuracy of forecasting Chevron's stock price, highlighting the potential benefits of hybrid modeling approaches for financial forecasting for some specific dataset.

Future research could enhance this analysis by incorporating additional external variables that influence gas prices and stock performance, such as global oil prices, geopolitical events, and macroeconomic indicators. Additionally, exploring alternative modeling frameworks, such as machine learning techniques for time series analysis, could improve forecasting accuracy. Testing these models on other energy-related stocks or indices might also provide broader insights into the dynamics of the energy sector and reveal patterns that are applicable across various sectors of the economy. Furthermore, this paper only utilizes a fixed window test. To strengthen the model, it will be important to conduct rolling window or recursive backtesting after adding more variables to ensure robustness.

References

Federal Reserve Bank of St. Louis. (n.d.). *Weekly U.S. All Grades All Formulations Retail Gasoline Prices (GASREGW)*. FRED, Federal Reserve Bank of St. Louis. Retrieved November 14, 2024, from https://fred.stlouisfed.org/series/GASREGW

Yahoo Finance. (n.d.). *Chevron Corporation (CVX) Stock Price, News, Quote & History*. Retrieved November 14, 2024, from https://finance.yahoo.com/quote/CVX/