Econ 104 Project 2

Matthew Craig, Zain Jaffer, Kento Koguchi

2023-02-16

Contents

```
rm(list=ls(all=TRUE))
library(ARDL)
library(corrplot)
library(dynlm)
library(knitr)
library(lmtest)
library(forecast)
library(ggplot2)
library(quantmod)
library(sandwich)
library(stargazer)
library(vars)
```

Dataset

```
response <- c("NATURALGASD11", "PETROLEUMD11")
invisible(getSymbols(response, src="FRED"))
predictors <- c("VMTD11", "RAILFRTCARLOADSD11", "IPB50089S")
invisible(getSymbols(predictors, src="FRED"))
```

Data Wrangling and Subsetting

```
# Subset to the same date range
start.date <- as.Date("2000-01-01")
end.date <- as.Date("2022-11-01")
NATURALGASD11 <- NATURALGASD11[paste(start.date, end.date, sep="/")]
PETROLEUMD11 <- PETROLEUMD11[paste(start.date, end.date, sep="/")]
VMTD11 <- VMTD11[paste(start.date, end.date, sep="/")]
RAILFRTCARLOADSD11 <- RAILFRTCARLOADSD11[paste(start.date, end.date, sep="/")]
IPB50089S <- IPB50089S[paste(start.date, end.date, sep="/")]

# Rescale some series
PETROLEUMD11 <- PETROLEUMD11 / 1000 # millions of barrels
VMTD11 <- VMTD11 / 1000 # billions of miles traveled
RAILFRTCARLOADSD11 <- RAILFRTCARLOADSD11 / 1000 # thousands of carloads

# Convert to ts format and take first-order difference
NATGAS <- diff(ts(NATURALGASD11, frequency=12, start=c(2000, 1)))
PETRO <- diff(ts(PETROLEUMD11, frequency=12, start=c(2000, 1)))
VMT <- diff(ts(VMTD11, frequency=12, start=c(2000, 1)))
RAIL <- diff(ts(RAILFRTCARLOADSD11, frequency=12, start=c(2000, 1)))
ENERGY <- diff(ts(IPB50089S, frequency=12, start=c(2000, 1)))

dataset <- cbind(NATGAS, PETRO, VMT, RAIL, ENERGY)
```

For this project, we have chosen a dataset from the U.S. Department of Transportation (DOT), Bureau of Transportation Statistics (BTS). We have chosen a subset of variables from the Transportation Services Index to model.

Descriptions:

- `NATGAS`: Consumption of natural gas in billions of cubic feet, seasonally adjusted
- `PETRO`: Pipeline petroleum movement in millions of barrels, seasonally adjusted
- `VMT`: Vehicle miles traveled, billions of miles, seasonally adjusted
- `RAIL`: Rail freight carloads (in thousands), seasonally adjusted
- `ENERGY`: Industrial production: energy total, index 2017=100, seasonally adjusted

The series `ENERGY` is added to this dataset from the G.17 Industrial Production and Capacity Utilization release from the Federal Reserve Board of Governors.

All series are monthly, and seasonally adjusted. We use the seasonally adjusted values in order to model the central trend of the data rather than the seasonality. We have also taken first-order differences for each series to convert them to roughly stationary data series.

1. Descriptive Analysis

Statistical Summary

```
dataset.df <- data.frame(dataset)
summary_df <- sapply(dataset.df, fivenum)
rownames(summary_df) <- c("Min", "Q1", "Median", "Q3", "Max")
summary_df <- t(summary_df)
stargazer(summary_df, type="text")
```

Five Number Summary

```
##
## ================================================
##          Min      Q1    Median   Q3     Max
## ------------------------------------------------
## NATGAS -317.100 -42.200 7.850    48   361.600
## PETRO  -33.429  -2.956  0.386  3.757  36.659
## VMT    -59.170  -1.018  0.129  1.489  37.130
## RAIL   -143.659 -15.156 -1.023 14.033 85.036
## ENERGY  -7.824  -0.567  0.220  0.874   5.467
## ------------------------------------------------
```

The five number summaries show reasonable values for all 5 series. The min and the max are roughly comparable, as are the Q1 and Q3 for each series. This indicates that the data are roughly centered. The medians are near zero (except for `NATGAS`, which also has the largest spread). Without domain knowledge, it is difficult for us to comment on the feasibility of the min and max values in the dataset.

```
stargazer(dataset.df, type="text")
```

Mean and Std. Dev

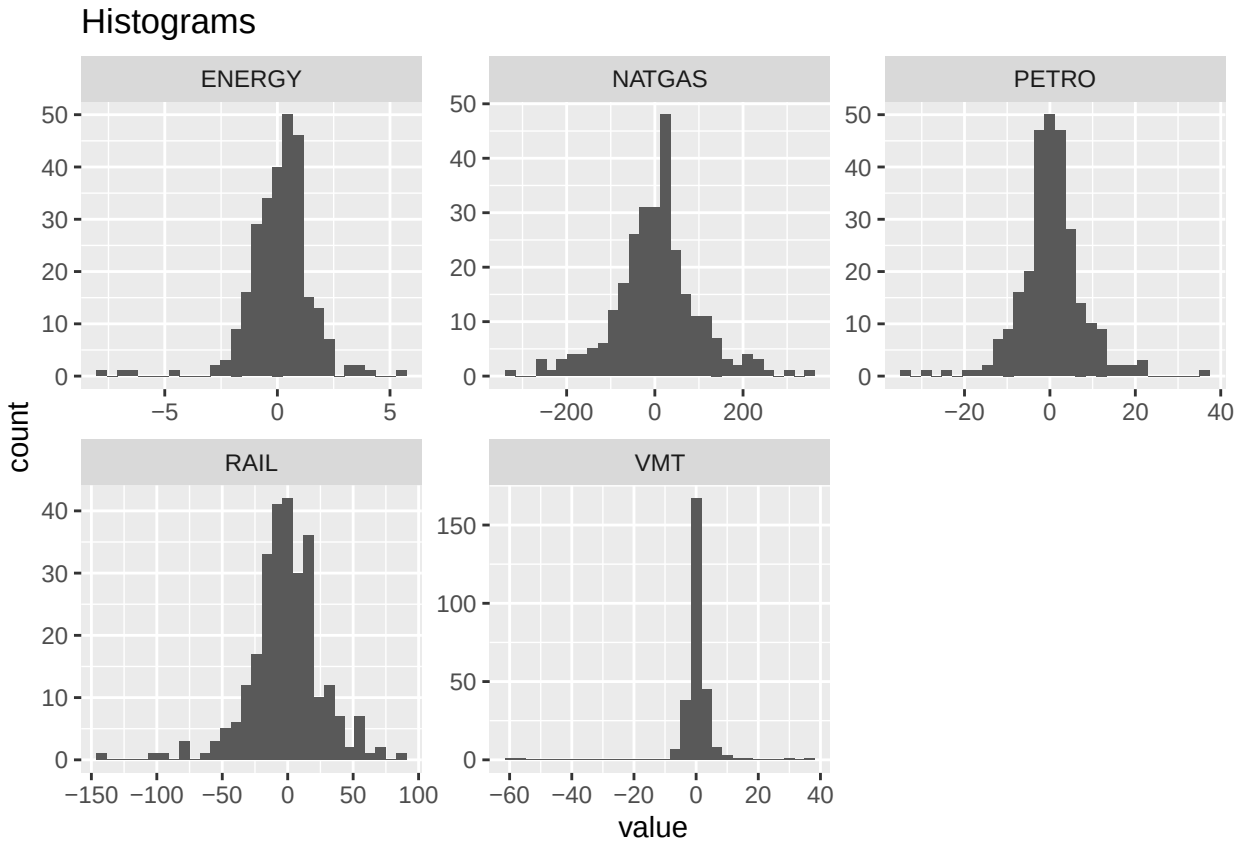
```
##
## ================================================
## Statistic  N   Mean  St. Dev.   Min      Max
## ------------------------------------------------
## NATGAS    274 3.186   95.628  -317.100 361.600
## PETRO     274 0.411   7.441   -33.429  36.659
## VMT       274 0.164   6.431   -59.170  37.130
## RAIL      274 -1.523  27.737  -143.659 85.036
## ENERGY    274 0.114   1.411    -7.824   5.467
## ------------------------------------------------
```

The means of the datasets are very close to the medians, which is a good sign that our data is not skewed. We can see that the means for `PETRO`, `VMT`, and `ENERGY` are very close to zero, while `RAIL` and `NATGAS` differ slightly but also have the widest spread (based on standard deviation). The standard deviations are reasonable compared to the minimum and maximum values - no outrageous z-scores, except possibly for `VMT`.

Distributions

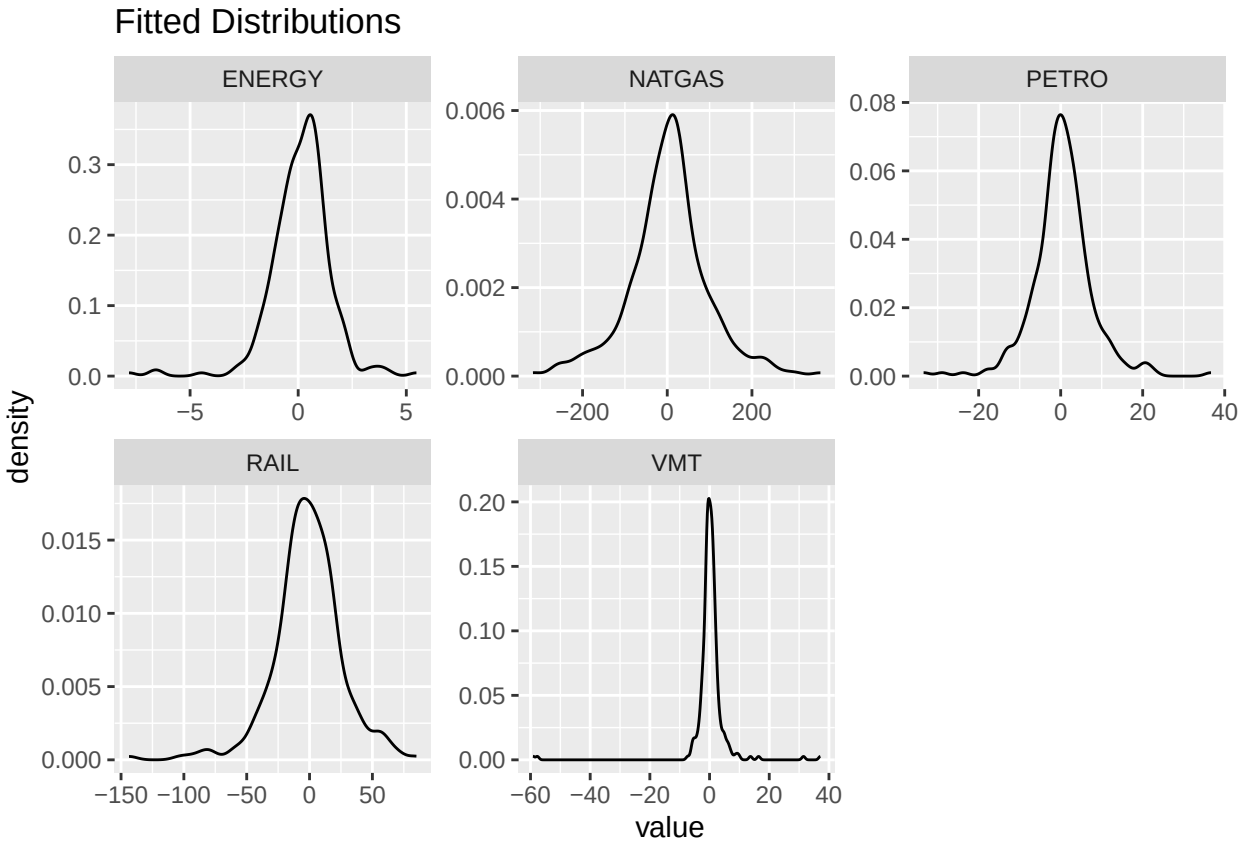
```
dataset.df %>%
  tidyr::pivot_longer(everything(), names_to="key")  %>%
  ggplot(aes(x=value)) +
  geom_histogram() +
  ggtitle("Histograms") +
  facet_wrap(~ key, scales="free")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can see from the histograms that the differenced series are all symmetric roughly about 0, which is exactly what we want to see for stationary data. It appears that differencing the data has made the non-stationary data stationary. The left tails on `ENERGY` and especially on `VMT` appear to be heavier than we might expect from a normal distribution. Overall the distributions are quite smooth, which is a great sign.

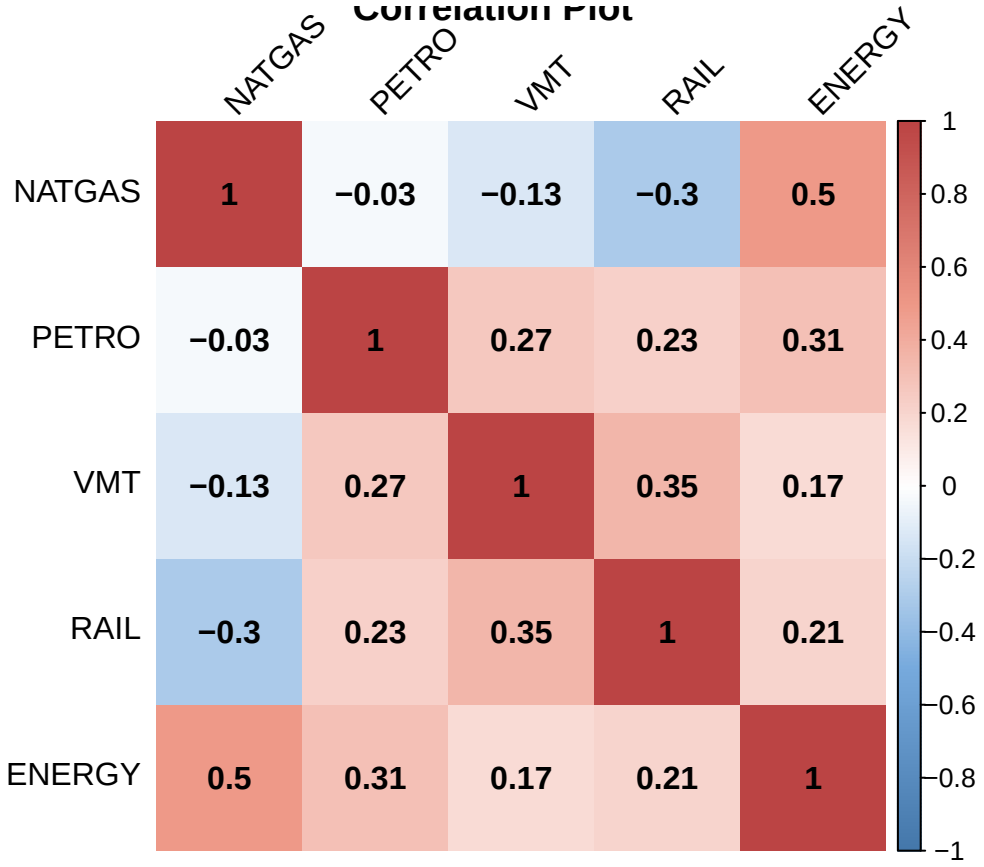
```
dataset.df %>%
  tidyr::pivot_longer(everything(), names_to="key")  %>%
  ggplot(aes(x=value)) +
  geom_density() +
  ggtitle("Fitted Distributions") +
  facet_wrap(~ key, scales="free")
```



The fitted distributions reinforce our observations above of symmetricity. `ENERGY` or `RAIL` might be slightly skewed, but this is hardly noticeable. The sharpness of the `VMT` distribution is due to extreme negative values that we saw in the histogram. Overall, these fitted distributions look like we expect for stationary data.

Correlation

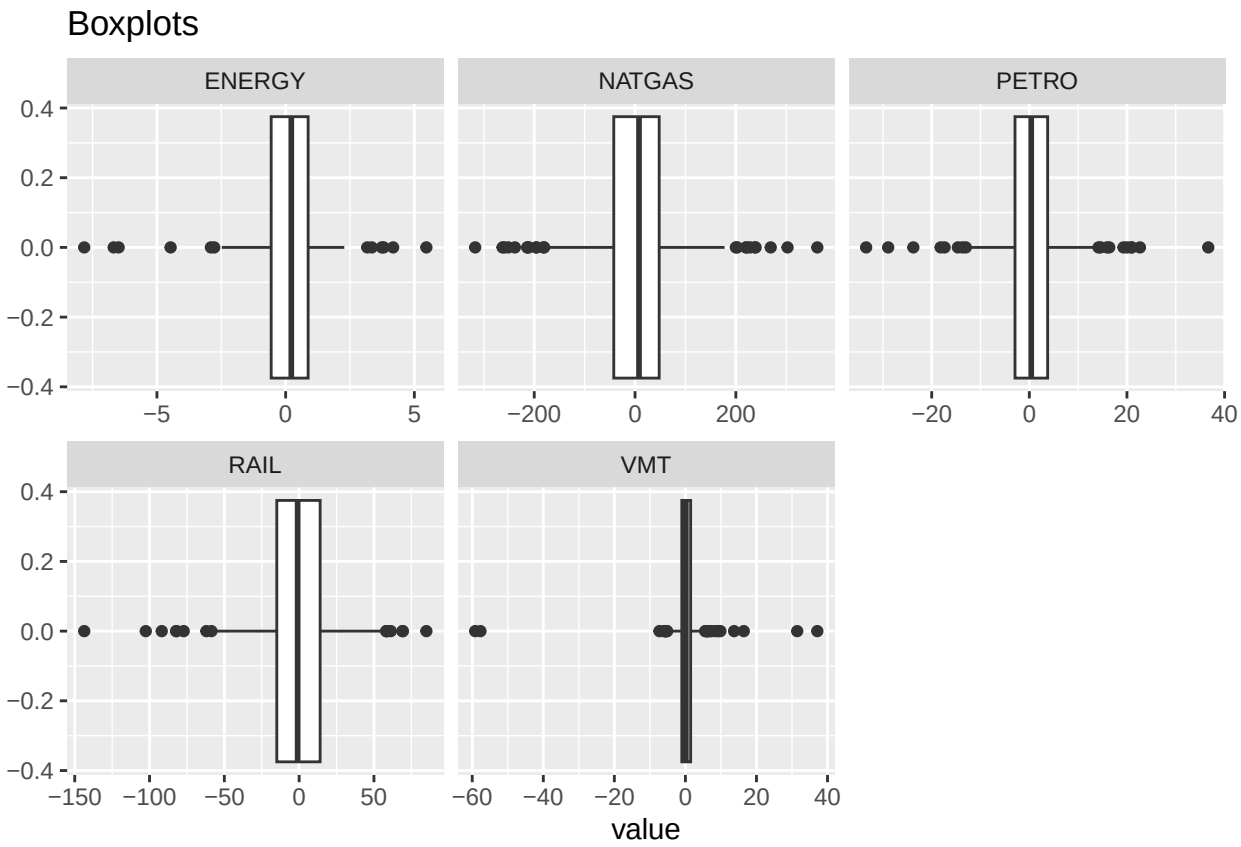
```
gradient <- colorRampPalette(c("#4477AA", "#77AADD", "#FFFFFF", "#EE9988", "#BB4444"))
corrplot(cor(dataset), method="shade", shade.col=NA, tl.col="black", tl.srt=45,
         col=gradient(200), addCoef.col="black", title="Correlation Plot")
```



The correlation plot shows fairly significant correlations for some of our predictors, in particular `ENERGY` with `NATGAS`. `VMT` and `RAIL` do have some correlation, but likely not enough to worry about collinearity. It is worth noting that this correlation plot only measures at a lag of zero, so while we can use the correlations to guide our choice of models for ARDL, we may discover different relationships with the full suite of lagged predictors.

Boxplots

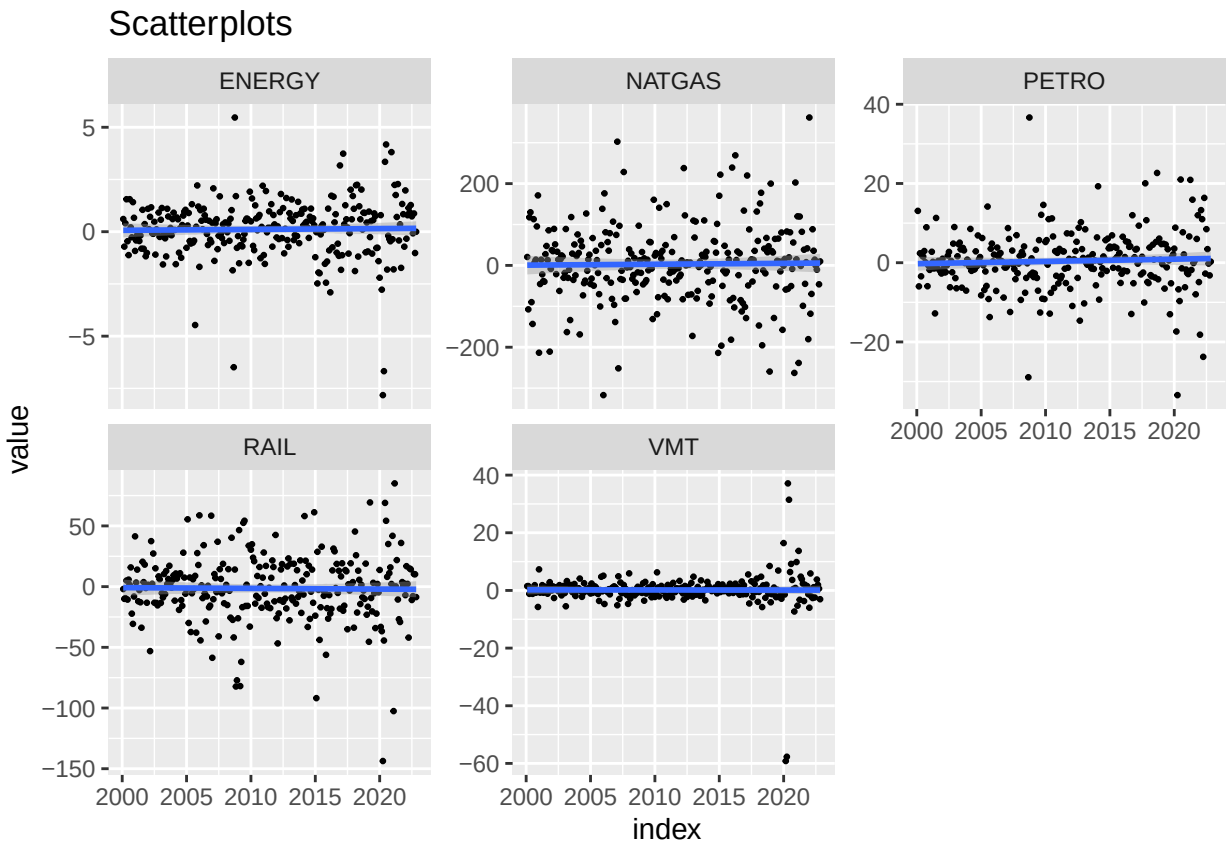
```
dataset.df %>%
  tidyr::pivot_longer(everything(), names_to="key")  %>%
  ggplot(aes(x=value)) +
  geom_boxplot() +
  ggtitle("Boxplots") +
  facet_wrap(~ key, scales="free_x")
```



The boxplots indicate a few outlying points for each of the series, but they seem to be roughly evenly distributed on the left and right tails, with the exception of the minimums of `VMT`.

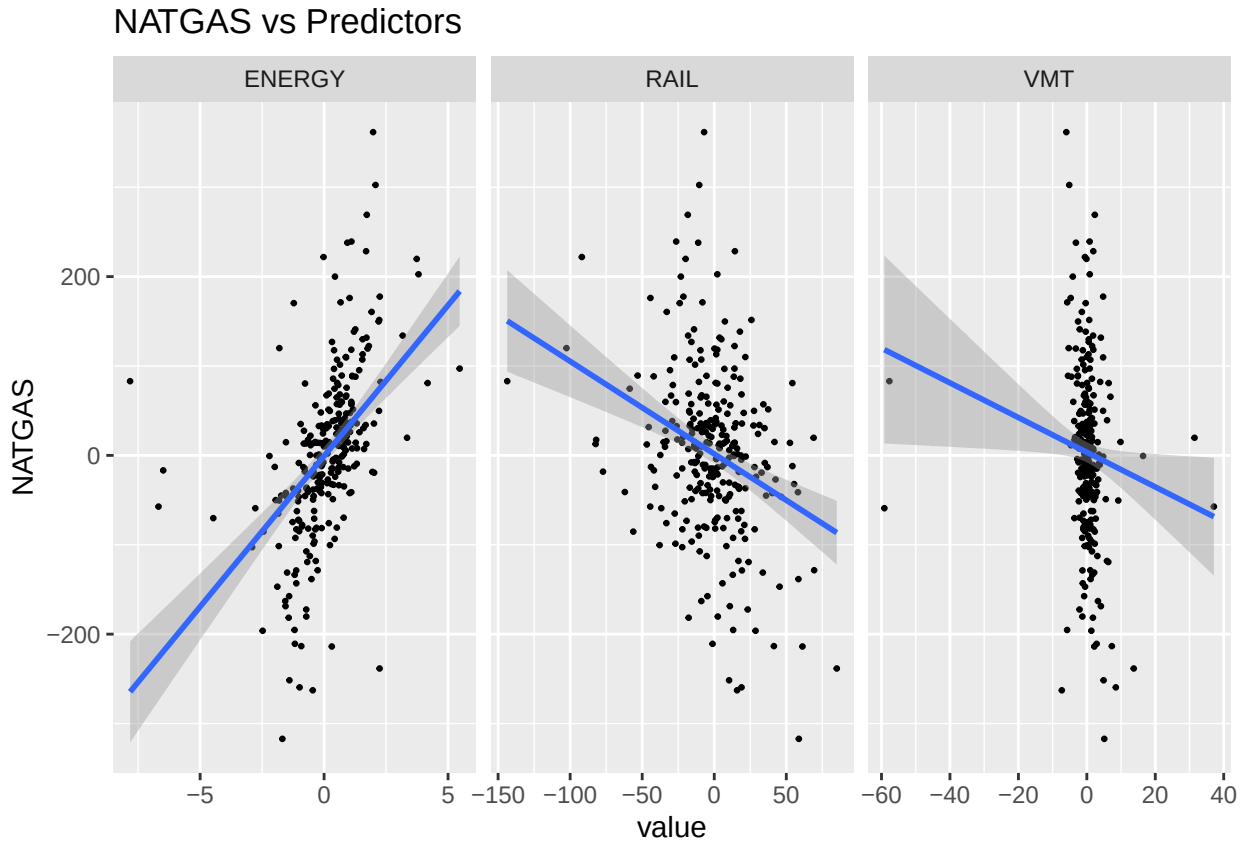
Scatterplots

```
dataset.dfi <- cbind(dataset.df, index(dataset))
colnames(dataset.dfi) <- append(colnames(dataset), "index")
dataset.dfi %>%
  tidyr::pivot_longer(c(-index), names_to="key")  %>%
  ggplot(aes(y=value, x=index)) +
  geom_point(size=0.5) +
  geom_smooth(formula=y ~ x, method="lm") +
  ggtitle("Scatterplots") +
  facet_wrap(~ key, scales="free_y")
```

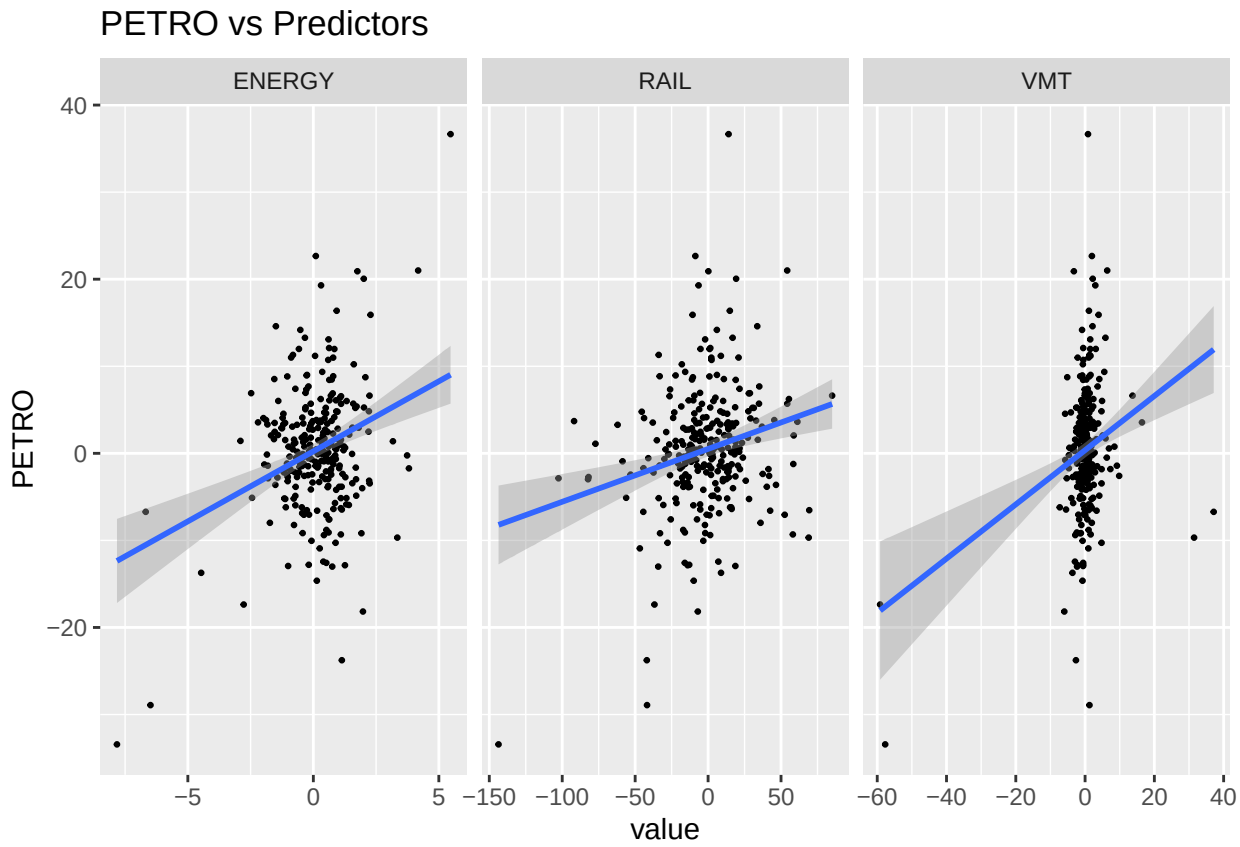


The scatterplots of variables vs year show relatively solid, symmetric bands about the regression lines, which are all very close to 0. Most of the series could have some slight increase in variance as the series progresses, the strongest effect being in `RAIL`. We could formally test for heteroskedasticity, but it seems safe to assume it is present based on the scatterplots. Of particular note is the extreme variance of `VMT` during 2020. This makes sense, since lockdowns at the beginning of the Covid-19 pandemic had a huge impact on travel, and likewise the surge after opening up is also reflected in the data. These are only a few observations, but they could pose issues with estimation later.

```
dataset.df %>%
  tidyr::pivot_longer(c(-PETRO, -NATGAS), names_to="key")  %>%
  ggplot(aes(x=value, y=NATGAS)) +
  geom_point(size=0.5) +
  geom_smooth(formula=y ~ x, method="lm") +
  ggtitle("NATGAS vs Predictors") +
  facet_wrap(~ key, scales="free_x")
```



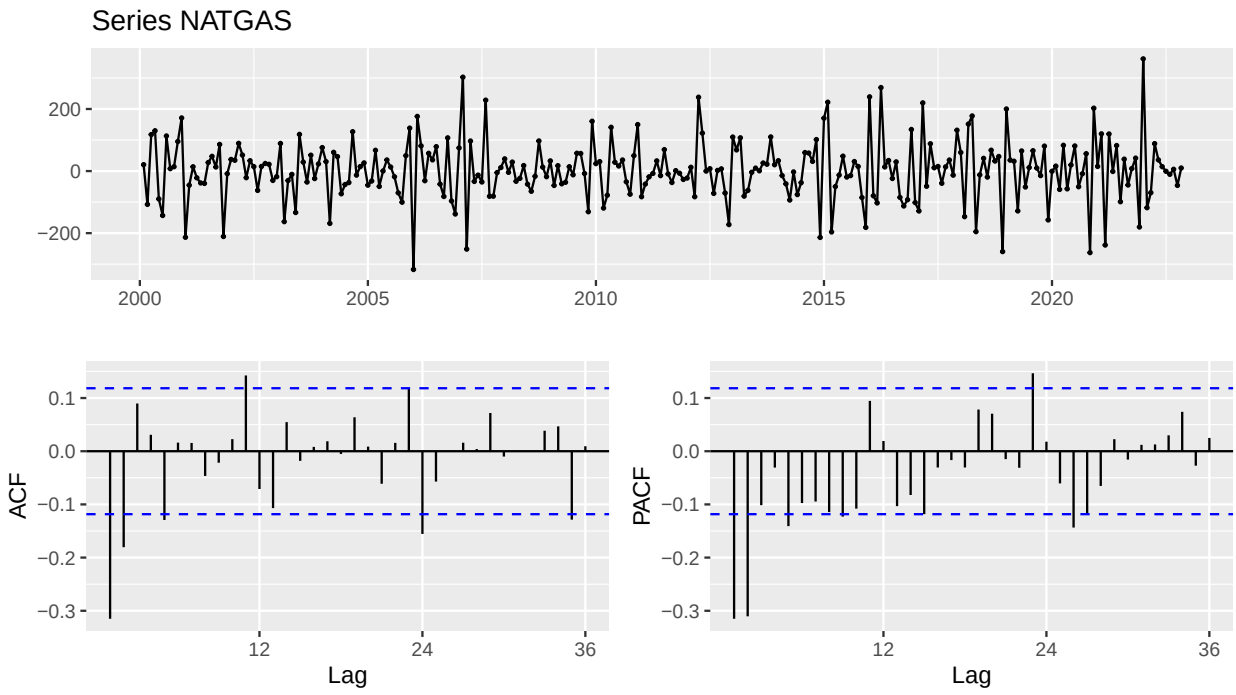

```
dataset.df %>%
  tidyr::pivot_longer(c(-PETRO, -NATGAS), names_to="key")  %>%
  ggplot(aes(x=value, y=PETRO)) +
  geom_point(size=0.5) +
  geom_smooth(formula=y ~ x, method="lm") +
  ggtitle("PETRO vs Predictors") +
  facet_wrap(~ key, scales="free_x")
```



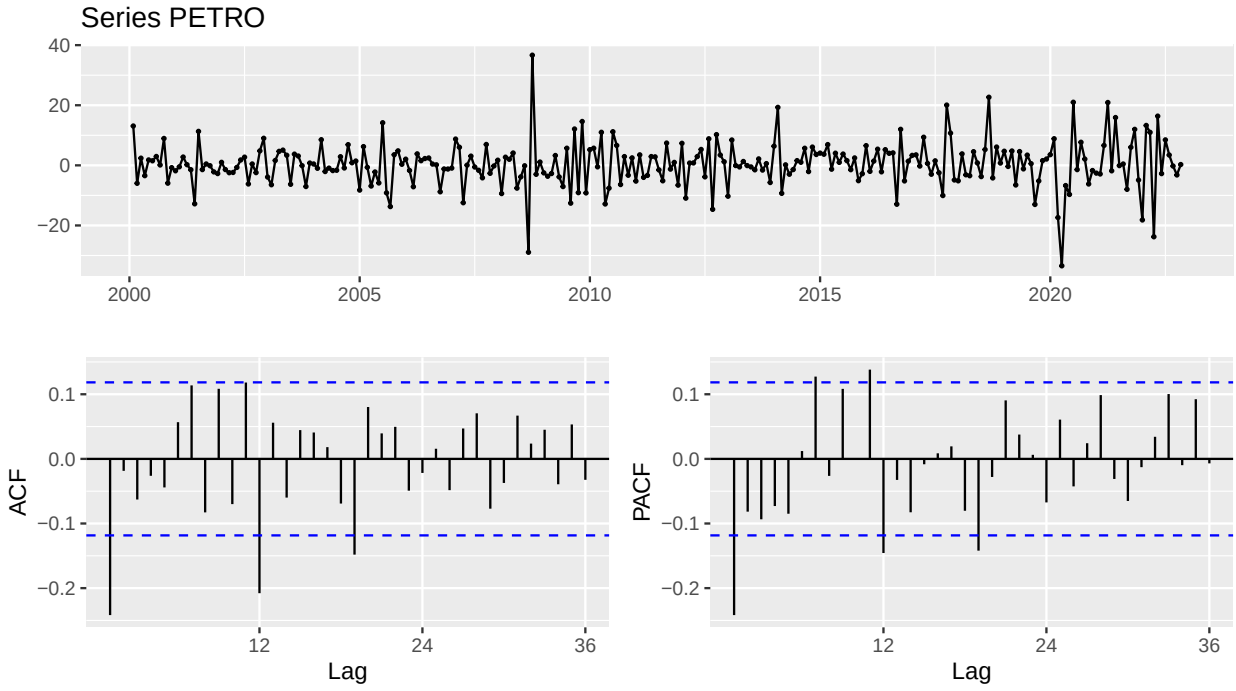
Finally, we plote scatterplots of our response variables `NATGAS` and `PETRO` against the three predictors. We can see positive relationships between `PETRO` and all three predictors, and negative relationships between `NATGAS` and `RAIL` and `VMT`. The regression line for both `NATGAS` and `PETRO` on `VMT` is dominated by the few outlier values, which leads to a much larger variance in the estimate relative to the other predictors. While we do have non-zero trend lines, none of the relationships are exceptionally tight, which is a good visualization of the results we already saw in the correlation matrix.

2. Time Series Displays

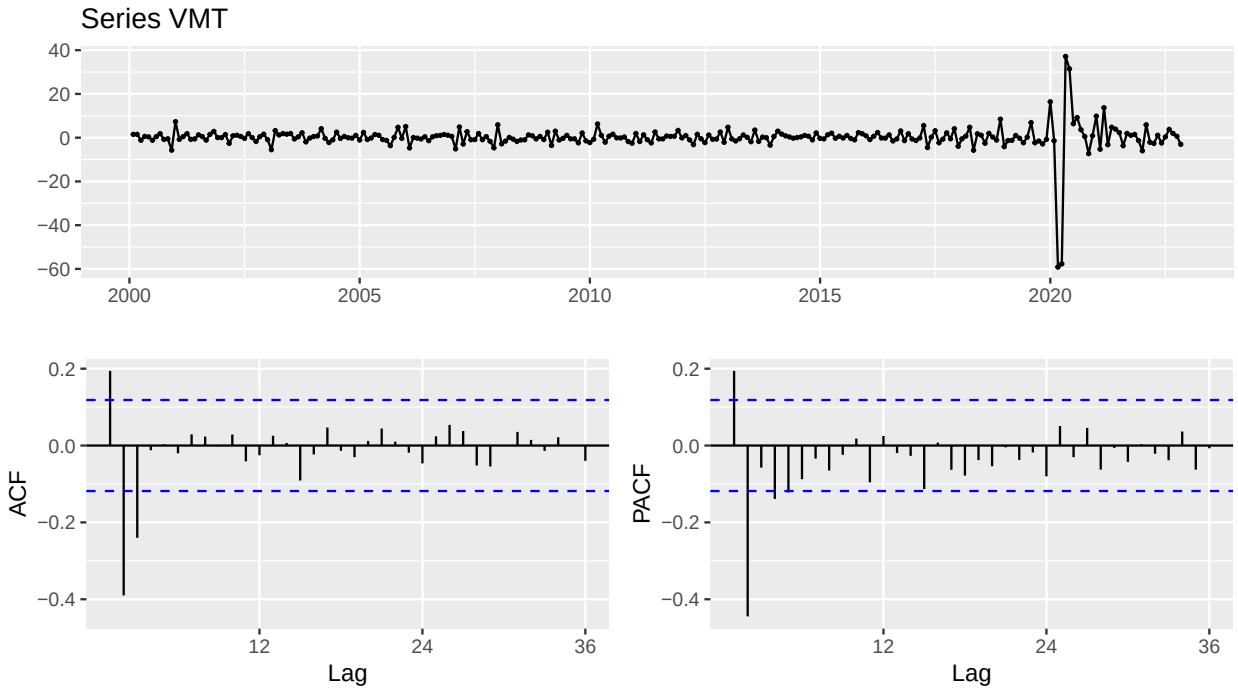
```
ggtsdisplay(NATGAS, main="Series NATGAS")
```



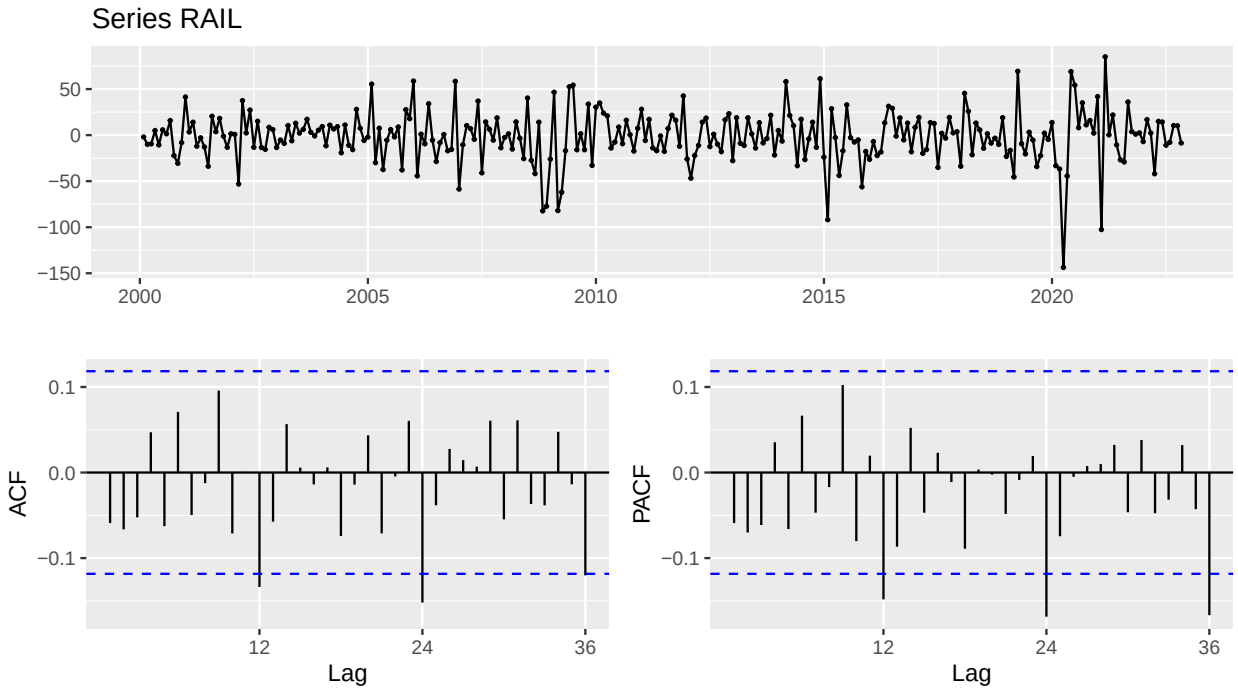

```
ggtsdisplay(PETRO, main="Series PETRO")
```



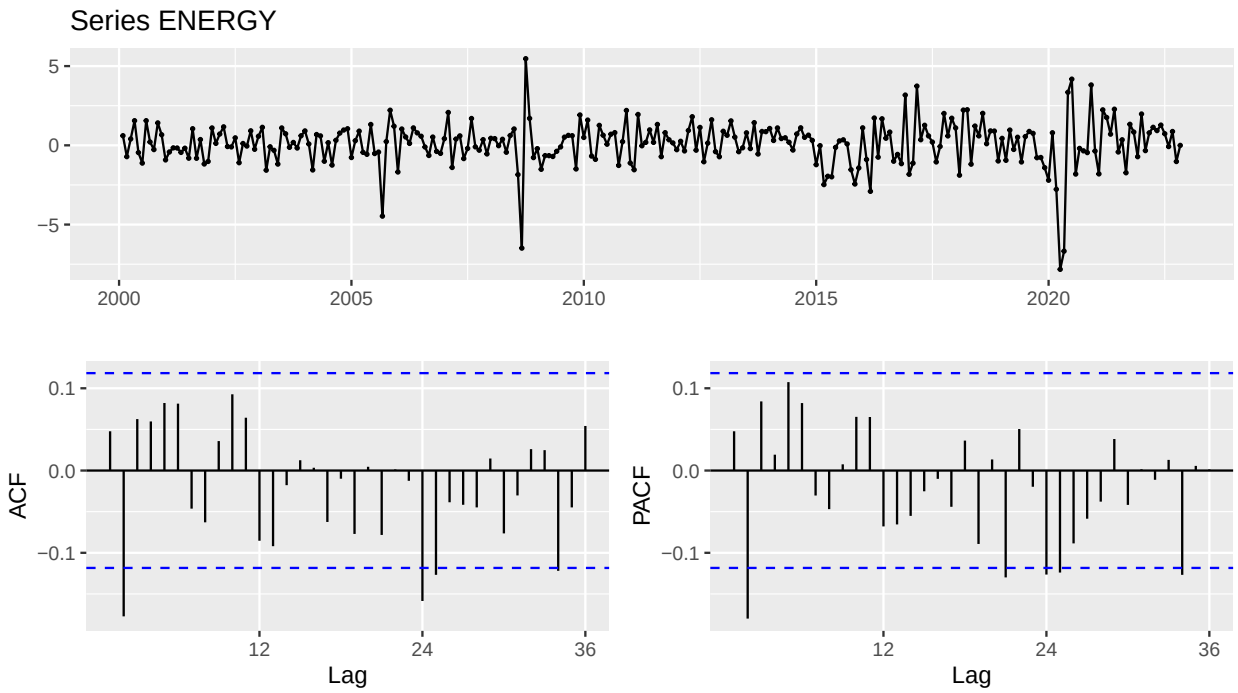
```
ggtsdisplay(VMT, main="Series VMT")
```




```
ggtsdisplay(RAIL, main="Series RAIL")
```




```
ggtsdisplay(ENERGY, main="Series ENERGY")
```



The `NATGAS` series appears stationary, with some periods of larger variance around 2006 and after 2015. We can see two very significant lags at 1 and 2 in the ACF and PACF, with some scattered barely significiant longer lags.

`PETRO` likewise shows some increasing variance after 2017 (with one major spike in 2008-9). The ACF shows significant lags at 1 and 12, while the PACF indicates that a few of the lags in between may barely reach significance.

The dominating feature of the `VMT` series is the extreme spike in 2020 previously mentioned. We can see some strong correlation at low lags (between 1 and 3) in the ACF and PACF, but beyond those lags, there is very little autocorrelation in the series.

`RAIL` shows a slowly increasing variance over the course of the series. The only significant spikes in the ACF and PACF occur at multiples of 12, indicating that the seasonal adjustment may have not fully removed the seasonality of the data. Otherwise, there is no autocorrelation in the series.

Finally, `ENERGY` shows very similar propreties to `PETRO` in the graph of the values. The only significant lag is at 2, with a few beyond 24 in the ACF/PACF.

3. Autoregressive AR(p) Models

We will begin with the series `NATGAS`. Based on the ACF and PACF, we can see potentially significant spikes at lags of 1, 2, and 5, with almost nothing else beyond these lags. This motivates our choice of models.

First, an AR(2) model:

$$NATGAS_t = \beta_0 + \beta_1 NATGAS_{t-1} + \beta_2 NATGAS_{t-2} + \epsilon_t$$

Second, an AR(5) model to capture the fifth lag:

$$NATGAS_t = \beta_0 + \sum_{i=1}^{5} \beta_i NATGAS_{t-i} + \epsilon_t$$

```r
# AR(2)
natgas.ar.mdl1 <- arima(NATGAS, order=c(2,0,0))
coeftest(natgas.ar.mdl1)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## ar1       -0.411474   0.057364 -7.1731 7.333e-13 ***
## ar2       -0.309745   0.057354 -5.4006 6.643e-08 ***
## intercept  3.254920   3.028629  1.0747    0.2825
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# AR(5)
natgas.ar.mdl2 <- arima(NATGAS, order=c(5,0,0))
coeftest(natgas.ar.mdl2)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## ar1       -0.449937   0.059769 -7.5279 5.156e-14 ***
## ar2       -0.377424   0.065358 -5.7747 7.710e-09 ***
## ar3       -0.163988   0.068643 -2.3890   0.01689 *
## ar4       -0.092843   0.065662 -1.4140   0.15738
## ar5       -0.140942   0.059816 -2.3562   0.01846 *
## intercept  3.244005   2.311673  1.4033   0.16052
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The two models have close estimates for β_1 and β_2. The magnitudes of the other coefficients in model 2 are much smaller, which reflects the relationship we observed in the ACF/PACF. The coefficient β_5 is statistically significant, but β_4 is not.

Based on the ACF and PACF of `PETRO`, we can see that a lag of 1 appears to be the most important. We can also see that a lag of 12 appears to be significant as well (possibly indicating a deficiency in the seasonal adjustment). There are potentially some significant lags at 7 and 11 as well. We will examine two models to understand this long lag.

First, an AR(1) model:

$$PETRO_t = \beta_0 + \beta_1 PETRO_{t-1} + \epsilon_t$$

Second, an AR(12) model to capture all possible significant lags:

$$PETRO_t = \beta_0 + \sum_{i=1}^{12} \beta_i PETRO_{t-i} + \epsilon_t$$

```r
# AR(1)
petro.ar.mdl1 <- arima(PETRO, order=c(1,0,0))
coeftest(petro.ar.mdl1)
```

```
##
## z test of coefficients:
##
##            Estimate Std. Error z value  Pr(>|z|)
## ar1       -0.243406   0.058809 -4.1389 3.489e-05 ***
## intercept  0.402172   0.350329  1.1480     0.251
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

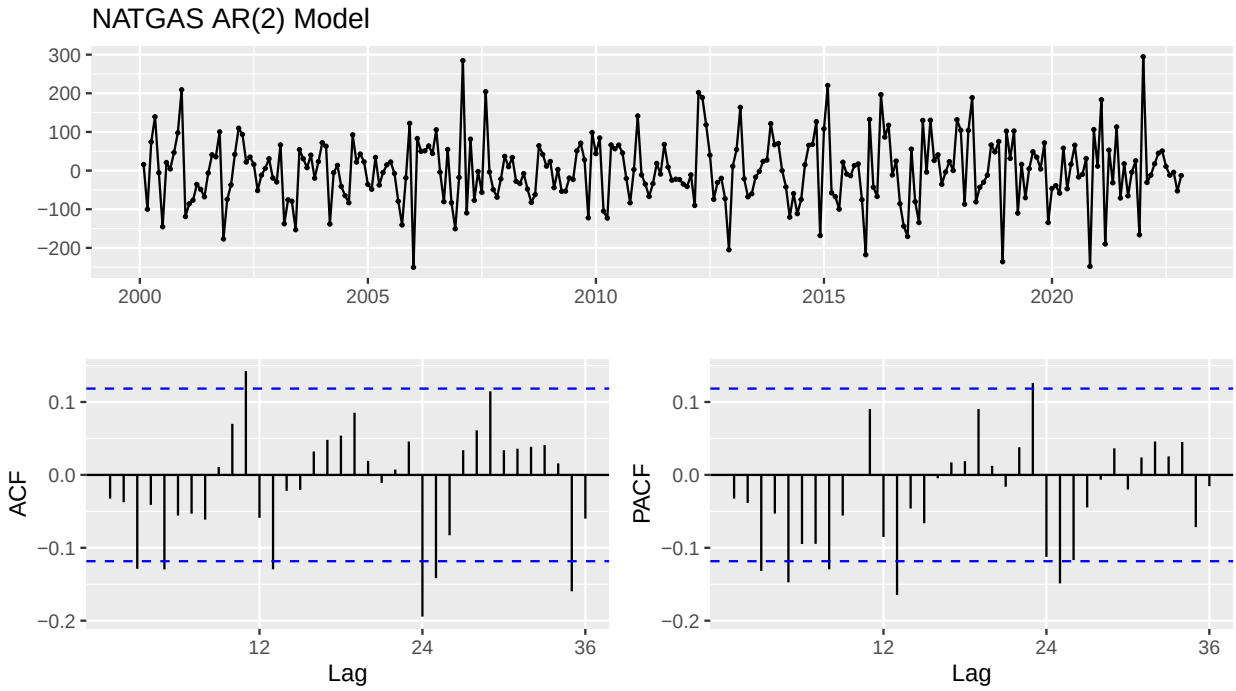
```
# AR(12)
petro.ar.mdl2 <- arima(PETRO, order=c(12,0,0))
coeftest(petro.ar.mdl2)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## ar1       -0.2518882  0.0598869 -4.2061 2.599e-05 ***
## ar2       -0.1348319  0.0614005 -2.1959  0.028096 *
## ar3       -0.0931095  0.0617475 -1.5079  0.131578
## ar4       -0.0913467  0.0616182 -1.4825  0.138217
## ar5       -0.0412480  0.0620642 -0.6646  0.506305
## ar6        0.0762259  0.0615276  1.2389  0.215386
## ar7        0.1324680  0.0615086  2.1536  0.031268 *
## ar8        0.0043629  0.0626027  0.0697  0.944439
## ar9        0.1082908  0.0625274  1.7319  0.083292 .
## ar10       0.0116282  0.0627126  0.1854  0.852900
## ar11       0.1071490  0.0630408  1.6997  0.089192 .
## ar12      -0.1611847  0.0620538 -2.5975  0.009391 **
## intercept  0.4208257  0.3096763  1.3589  0.174172
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

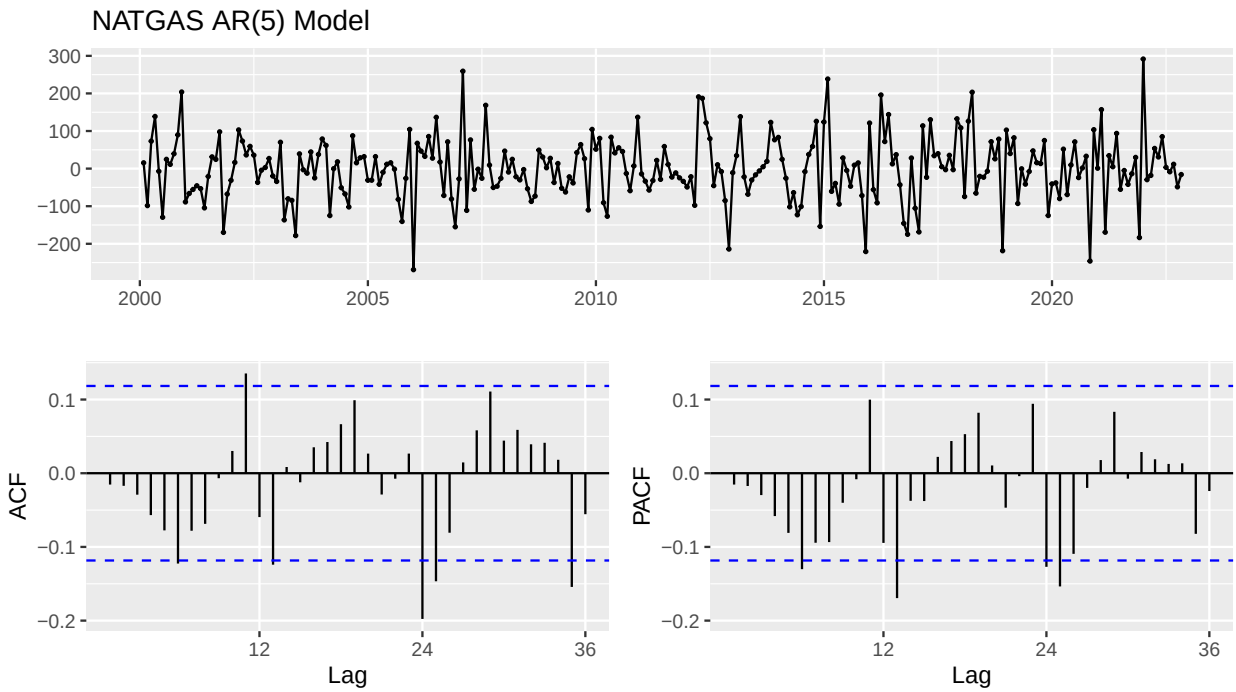
The estimate of β_1 is slightly smaller in the AR(12) model than in the AR(1), but they are very close. Most of the coefficients of the AR(12) model are not statistically significant at all.

3.1 Residual Analysis

```
ggtsdisplay(resid(natgas.ar.mdl1), main="NATGAS AR(2) Model", na.action=na.omit)
```

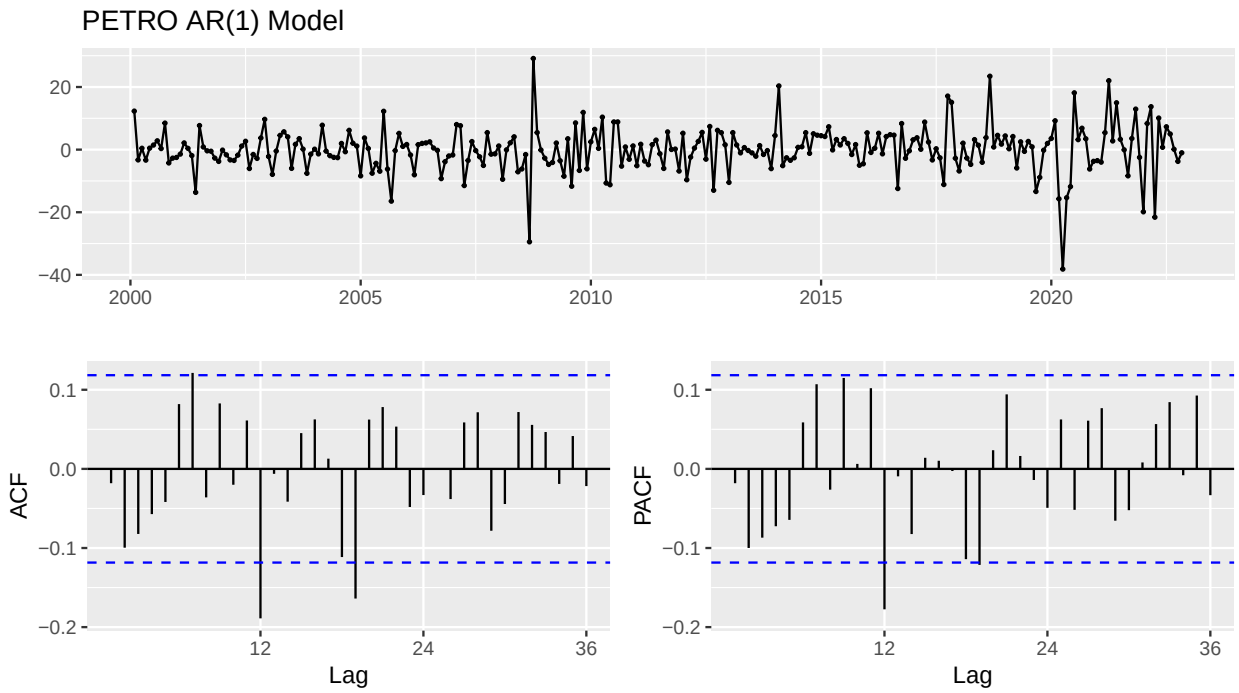



```
ggtsdisplay(resid(natgas.ar.mdl2), main="NATGAS AR(5) Model", na.action=na.omit)
```

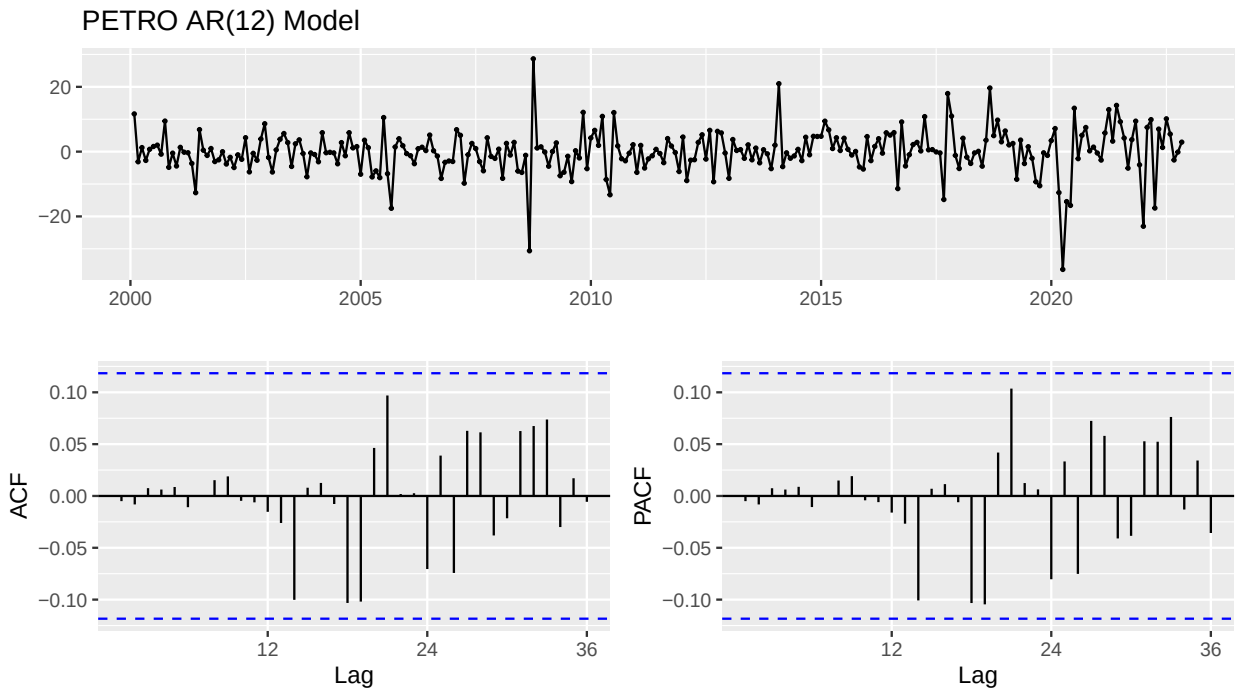


The residual plot of both models looks very good. In the ACF/PACF of the first model, we can see lags that are barely significant at 3 and 5, and then some around 12 and 24. Model 2 looks fairly similar, although the low-order lags are almost completely gone. Outside of a potentially long-term seasonal dependence, it looks like both models capture the majority of the variation in `NATGAS`.

```
ggtsdisplay(resid(petro.ar.mdl1), main="PETRO AR(1) Model")
```




```
ggtsdisplay(resid(petro.ar.mdl2), main="PETRO AR(12) Model")
```



In both plots we can see some times of large variance (especially post-2020, which may prove to be challenging for our forecasting) but the residuals look fairly well centered around 0 in both models.

We can see a significant spike in both the PACF and the ACF of model 1 at a lag of 12, which is to be expected since this model did not include that lag. Also as expected, the spike is not present in model 2. Model 2 has no significant spikes at all.

3.2 Model Selection

The first thing we will do is define a recursive prediction function:

```
recursive_predict <- function(object, ...) UseMethod("recursive_predict")
recursive_predict.Arima <- function(object, data, ...) {
  order <- length(coef(object))-1
  npred <- length(data)-order
  preds <- ts(NA, start=start(data)+c(0,order), end=end(data), freq=frequency(data))
  for (i in 1:npred) {
    x <- append(rev(data[i:(i+order-1)]), 1)
    preds[i] <- sum(x * coef(object))
  }
  return(preds)
}
```

Now we construct a train/test split and re-estimate our models:

```
num_train <- round(length(NATGAS) * 2 / 3)
num_test <- length(NATGAS) - num_train

# Train-test split
NATGAS.train <- window(NATGAS, end=start(NATGAS)+c(0,num_train-1))
NATGAS.test <- window(NATGAS, start=end(NATGAS)-c(0,num_test-1))
# Re-estimation
natgas.ar.mdl1.test <- arima(NATGAS.train, order=c(2,0,0))
natgas.ar.mdl2.test <- arima(NATGAS.train, order=c(5,0,0))

# Train-test split
PETRO.train <- window(PETRO, end=start(PETRO)+c(0,num_train-1))
PETRO.test <- window(PETRO, start=end(PETRO)-c(0,num_test-1))
# Re-estimation
petro.ar.mdl1.test <- arima(PETRO.train, order=c(1,0,0))
petro.ar.mdl2.test <- arima(PETRO.train, order=c(12,0,0))
```

And finally make our recursive predictions:

```
# Have to change start point of test series to include lags for the first predictions
natgas.ar.mdl1.test.pred <- recursive_predict(natgas.ar.mdl1.test,
                              window(NATGAS, start=end(NATGAS)-c(0,num_test+1)))
natgas.ar.mdl2.test.pred <- recursive_predict(natgas.ar.mdl2.test,
                              window(NATGAS, start=end(NATGAS)-c(0,num_test+4)))

natgas.ar.mse <- rbind(
  "NATGAS AR(2)"=mean((NATGAS.test - natgas.ar.mdl1.test.pred)^2),
  "NATGAS AR(5)"=mean((NATGAS.test - natgas.ar.mdl2.test.pred)^2)
)
natgas.ar.mse <- cbind(natgas.ar.mse, c(mean(resid(natgas.ar.mdl1.test)^2),
```

```
                                        mean(resid(natgas.ar.mdl2.test)^2)))
colnames(natgas.ar.mse) <- c("Test MSE", "Train MSE")
natgas.ar.mse %>% kable()
```

	Test MSE	Train MSE
NATGAS AR(2)	9640.916	6548.174
NATGAS AR(5)	9502.371	6281.554

```
petro.ar.mdl1.test.pred <- recursive_predict(petro.ar.mdl1.test,
                              window(PETRO, start=end(PETRO)-c(0,num_test)))
petro.ar.mdl2.test.pred <- recursive_predict(petro.ar.mdl2.test,
                              window(PETRO, start=end(PETRO)-c(0,num_test+11)))

petro.ar.mse <- rbind(
  "PETRO AR(1)"=mean((PETRO.test - petro.ar.mdl1.test.pred)^2),
  "PETRO AR(12)"=mean((PETRO.test - petro.ar.mdl2.test.pred)^2)
)
petro.ar.mse <- cbind(petro.ar.mse, c(mean(resid(petro.ar.mdl1.test)^2),
                                      mean(resid(petro.ar.mdl2.test)^2)))
colnames(petro.ar.mse) <- c("Test MSE", "Train MSE")
petro.ar.mse %>% kable()
```

	Test MSE	Train MSE
PETRO AR(1)	86.14989	37.04380
PETRO AR(12)	89.53942	32.39579

The AR(5) model for `NATGAS` achieves a significantly lower train and test MSE, which makes it the obvious choice for our model. It appears that the extra lags increase the model capacity and lead to better generalization.

Somewhat surprisingly, the test MSE for the AR(1) model of `PETRO` is lower than for the AR(12) model. It seems that all the statistically insignificant coefficients we saw earlier in the AR(12) model overfit to the training data and decreased performance on the test set compared to the lower order model. We can see this in the train MSE: AR(12) has a lower train MSE by a significant amount, but this appears to be overfitting. The AR(1) model is the right choice for `PETRO`.

```
AIC(natgas.ar.mdl1, natgas.ar.mdl2) %>% kable()
```

	df	AIC
natgas.ar.mdl1	4	3227.501
natgas.ar.mdl2	7	3225.009

```
BIC(natgas.ar.mdl1, natgas.ar.mdl2) %>% kable()
```

	df	BIC
natgas.ar.mdl1	4	3241.953
natgas.ar.mdl2	7	3250.301

AIC and BIC disagree on the choice of the best model, which is probably because BIC penalizes the free parameters in the AR(5) more than AIC does. Since the AR(5) model does capture a statistically significant β_3 and β_5, we would select it over AR(2) despite the disagreement. This aligns with our choice based on MSE as well.

```
AIC(petro.ar.mdl1, petro.ar.mdl2) %>% kable()
```

	df	AIC
petro.ar.mdl1	3	1865.834
petro.ar.mdl2	14	1859.930

```
BIC(petro.ar.mdl1, petro.ar.mdl2) %>% kable()
```

	df	BIC
petro.ar.mdl1	3	1876.673
petro.ar.mdl2	14	1910.513

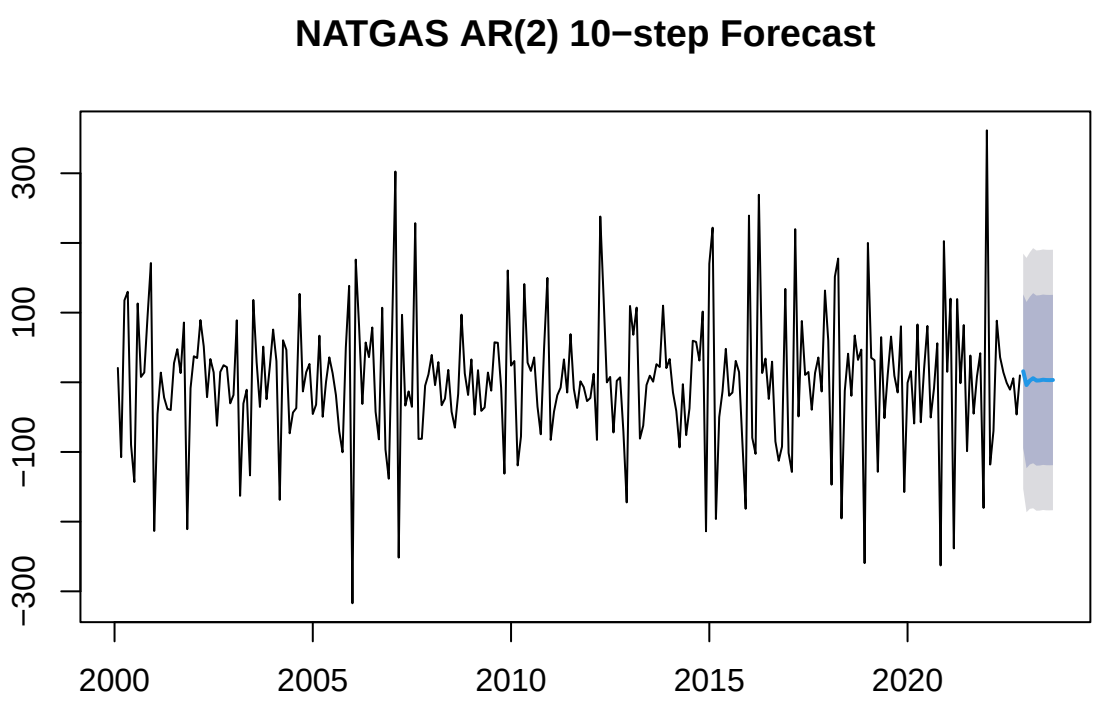
Once again, we have a split in AIC and BIC due to extra parameters. In this case, we opt for the much more parsimonious AR(1) model. The AR(12) model simply has too many parameters, and the greater difference in BIC is too much to overlook. This agrees with our choice based on MSE.

3.3 Forecast

```
forecast(natgas.ar.mdl1, 10) %>% kable()
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Dec 2022	15.839049	-94.5021	126.1802	-152.9132	184.5913
Jan 2023	-3.981404	-123.2985	115.3356	-186.4611	178.4983
Feb 2023	2.334611	-117.9844	122.6537	-181.6775	186.3467
Mar 2023	5.875018	-116.1677	127.9178	-180.7733	192.5233
Apr 2023	2.461878	-119.6343	124.5580	-184.2681	189.1918
May 2023	2.769674	-119.4226	124.9619	-184.1073	189.6466
Jun 2023	3.700227	-118.5317	125.9321	-183.2373	190.6378
Jul 2023	3.221990	-119.0101	125.4541	-183.7159	190.1598
Aug 2023	3.130538	-119.1061	125.3672	-183.8143	190.0753
Sep 2023	3.316300	-118.9209	125.5535	-183.6293	190.2619

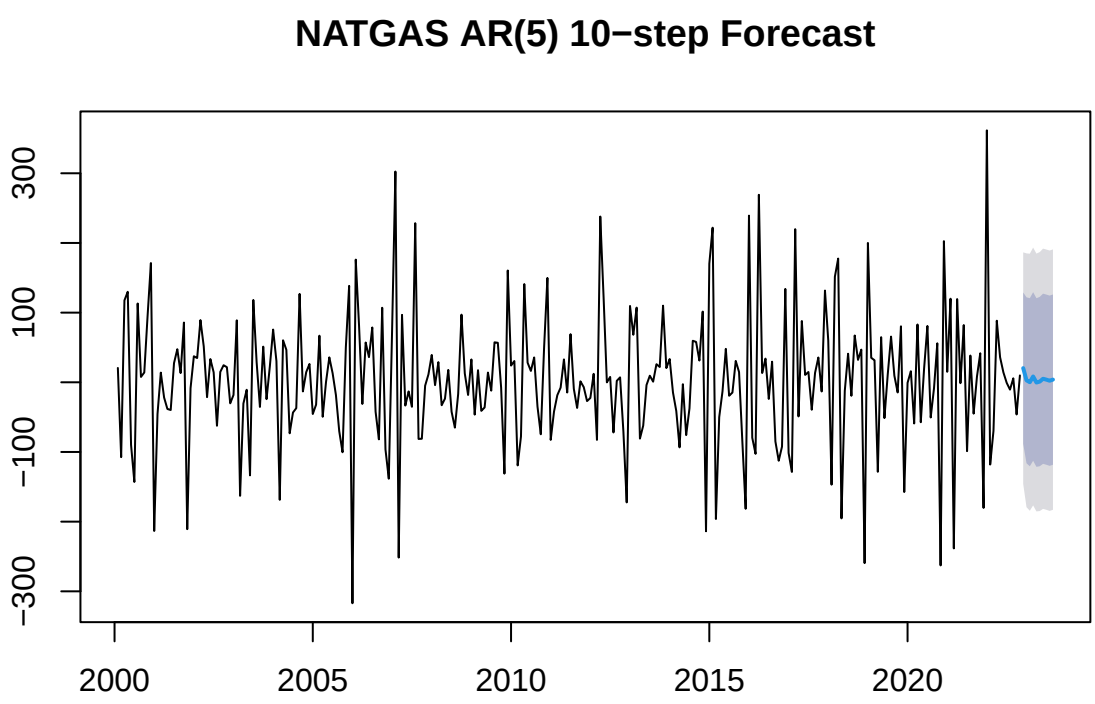
```
plot(forecast(natgas.ar.mdl1, 10), main="NATGAS AR(2) 10-step Forecast")
```




```
forecast(natgas.ar.mdl2, 10) %>% kable()
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Dec 2022	20.3538348	-88.26396	128.9716	-145.7627	186.4704
Jan 2023	2.8508094	-116.25509	121.9567	-179.3059	185.0076
Feb 2023	0.1161443	-120.49664	120.7289	-184.3452	184.5775
Mar 2023	8.3446987	-112.61729	129.3067	-176.6507	193.3401
Apr 2023	-0.3326146	-121.29849	120.6333	-185.3340	184.6687
May 2023	1.0660831	-120.45070	122.5829	-184.7778	186.9100
Jun 2023	5.0832021	-117.02273	127.1891	-181.6617	191.8281
Jul 2023	3.7922841	-118.31532	125.8999	-182.9552	190.5397
Aug 2023	2.2734736	-119.90976	124.4567	-184.5897	189.1366
Sep 2023	3.8784430	-118.30659	126.0635	-182.9874	190.7443

```
plot(forecast(natgas.ar.mdl2, 10), main="NATGAS AR(5) 10-step Forecast")
```

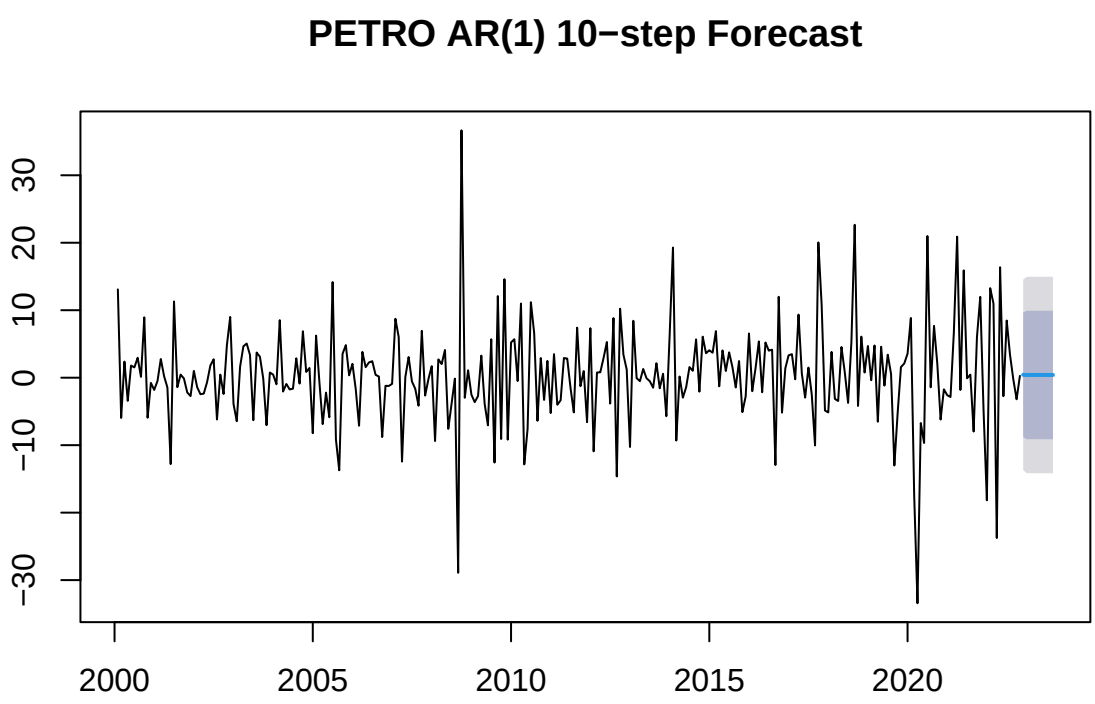


Both models have similar forecasts, predicting a large change in December 2022 and then gradually converging towards low positive numbers. The AR(5) model has slightly larger swings but converges within 10 periods as well.

```
forecast(petro.ar.mdl1, 10) %>% kable()
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Dec 2022	0.4338561	-8.800017	9.667730	-13.68813	14.55585
Jan 2023	0.3944594	-9.109015	9.897934	-14.13985	14.92877
Feb 2023	0.4040487	-9.115159	9.923256	-14.15432	14.96242
Mar 2023	0.4017146	-9.118424	9.921853	-14.15808	14.96151
Apr 2023	0.4022828	-9.117911	9.922477	-14.15760	14.96216
May 2023	0.4021445	-9.118053	9.922342	-14.15774	14.96203
Jun 2023	0.4021781	-9.118019	9.922376	-14.15771	14.96206
Jul 2023	0.4021699	-9.118028	9.922368	-14.15772	14.96206
Aug 2023	0.4021719	-9.118026	9.922370	-14.15771	14.96206
Sep 2023	0.4021715	-9.118026	9.922369	-14.15771	14.96206

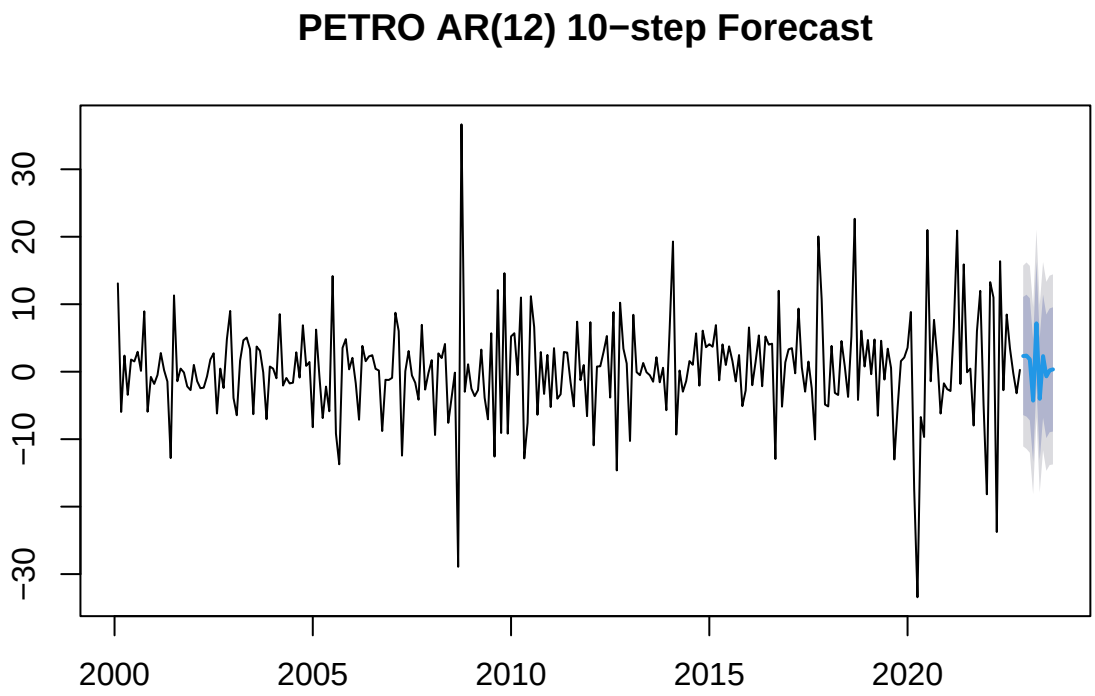
```
plot(forecast(petro.ar.mdl1, 10), main="PETRO AR(1) 10-step Forecast")
```




```
forecast(petro.ar.mdl2, 10) %>% kable()
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Dec 2022	2.3243355	-6.436741	11.085412	-11.074575	15.723246
Jan 2023	2.3752010	-6.659536	11.409939	-11.442237	16.192640
Feb 2023	1.8326130	-7.223744	10.888970	-12.017890	15.683116
Mar 2023	-4.2807020	-13.344238	4.782834	-18.142184	9.580780
Apr 2023	7.1585646	-1.914681	16.231811	-6.717767	21.034897
May 2023	-4.0154317	-13.088831	5.057968	-17.891998	9.861135
Jun 2023	2.3053292	-6.811893	11.422552	-11.638260	16.248918
Jul 2023	-0.6861460	-9.843666	8.471374	-14.691364	13.319072
Aug 2023	0.1987695	-8.977738	9.375277	-13.835488	14.233027
Sep 2023	0.3365853	-8.873955	9.547126	-13.749721	14.422892

```
plot(forecast(petro.ar.mdl2, 10), main="PETRO AR(12) 10-step Forecast")
```



The AR(1) model very quickly converges to a long-term forecast value around 0.4 with a very wide confidence interval, which is expected for such a low-order model. The AR(12) model forecasts some rise and fall in the next 12 months, but expects a gradual reversion towards the mean. It is interesting to note how wildly different the point forecasts are - the AR(1) model is very conservative, while AR(12) predicts some very large swings.

4. Autoregressive Distributed Lag ARDL(p,q,r,s) Models

With three predictor series in our dataset and a huge number of lags, we have a potentially infinite space of models to choose from. We will use some economic intuition, the correlation between variables, as well as the results of our previous section, to guide our decisions.

For `NATGAS`, energy production seems like the most important factor, since the majority of natural gas consumption is used for energy. An increase in `VMT` could also be indicative of greater economic activity and energy demand; some vehicles also run on natural gas (though not many). `RAIL` seems like the least important predictor despite its moderate correlation, since trains are almost entirely diesel-powered. Based on our results from the previous section, we will use 2 lags of `NATGAS`.

For our first model, we will try an ARDL(2,2,2,2) model, to test low-order lags of all the predictors.

Our second model will be more parsimonious based on our experience with AR models of `NATGAS`. We will try an ARDL(2,1,0,1) model, dropping lags of `RAIL` entirely and reducing the other lags to first order.

```
# ARDL(2,2,2,2)
natgas.ardl.mdl1 <- ardl(NATGAS ~ VMT + RAIL + ENERGY, data=dataset, order=c(2,2,2,2))
coeftest(natgas.ardl.mdl1)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  -1.595492   4.215630 -0.3785 0.7053898
## L(NATGAS, 1) -0.328465   0.060359 -5.4419 1.218e-07 ***
## L(NATGAS, 2) -0.235421   0.060414 -3.8968 0.0001241 ***
## VMT          -1.792611   0.795489 -2.2535 0.0250629 *
## L(VMT, 1)    -1.633487   0.814051 -2.0066 0.0458261 *
## L(VMT, 2)    -1.691889   0.886769 -1.9079 0.0575025 .
## RAIL         -1.056257   0.168602 -6.2648 1.539e-09 ***
## L(RAIL, 1)   -0.113018   0.181087 -0.6241 0.5331033
## L(RAIL, 2)   -0.187156   0.180030 -1.0396 0.2995021
## ENERGY       40.509951   3.366824 12.0321 < 2.2e-16 ***
## L(ENERGY, 1)  6.142951   4.165132  1.4749 0.1414624
## L(ENERGY, 2)  3.204539   3.892061  0.8234 0.4110619
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# ARDL(2,1,0,1)
natgas.ardl.mdl2 <- ardl(NATGAS ~ VMT + RAIL + ENERGY, data=dataset, order=c(2,1,0,1))
coeftest(natgas.ardl.mdl2)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  -0.843398   4.167457 -0.2024 0.8397777
## L(NATGAS, 1) -0.295573   0.055668 -5.3095 2.334e-07 ***
## L(NATGAS, 2) -0.176979   0.046728 -3.7875 0.0001885 ***
## VMT          -1.120595   0.716394 -1.5642 0.1189648
## L(VMT, 1)    -2.031314   0.770746 -2.6355 0.0088979 **
## RAIL         -1.078738   0.166353 -6.4846 4.356e-10 ***
## ENERGY       39.902607   3.362163 11.8681 < 2.2e-16 ***
## L(ENERGY, 1)  1.665977   3.625138  0.4596 0.6462086
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we expect from our previous results, the two lags of NATGAS are significant in both models. Surprisingly, lags of `ENERGY` are not statistically significant at all despite current the very large coefficient at no lag. It appears that only current energy demand impacts natural gas consumption. `VMT` appears to be somewhat important over time in model 1. `RAIL` is statistically significant at no lag in both models, which matches the correlation plot, but insignificant at longer lags.

For `PETRO`, we can expect `VMT` and `RAIL` to be more important than they were for `NATGAS`. We expect `ENERGY` to still be important, since petroleum is an important source of energy, but potentially less important than transportation measures (which are almost entirely dependent on petroleum).

Our first model will be the same, ARDL(2,2,2,2).

For our second model, based on the long dependencies we observed in the AR models of `PETRO`, we will try increasing the lags of transportation statistics and decreasing energy consumption. We will try an ARDL(2,4,4,1) model.

```r
# ARDL(2,2,2,2)
petro.ardl.mdl1 <- ardl(PETRO ~ VMT + RAIL + ENERGY, data=dataset, order=c(2,2,2,2))
coeftest(petro.ardl.mdl1)
```

```
##
## t test of coefficients:
##
##                Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   0.1598119  0.3945326  0.4051 0.6857618
## L(PETRO, 1)  -0.4146196  0.0610091 -6.7960 7.317e-11 ***
## L(PETRO, 2)  -0.1293411  0.0607467 -2.1292 0.0341786 *
## VMT           0.2694624  0.0738751  3.6475 0.0003198 ***
## L(VMT, 1)     0.1434682  0.0757558  1.8938 0.0593582 .
## L(VMT, 2)     0.1861444  0.0822391  2.2635 0.0244325 *
## RAIL          0.0097531  0.0156637  0.6227 0.5340573
## L(RAIL, 1)    0.0040817  0.0157768  0.2587 0.7960587
## L(RAIL, 2)   -0.0253425  0.0156773 -1.6165 0.1071966
## ENERGY        1.2125907  0.3133699  3.8695 0.0001379 ***
## L(ENERGY, 1)  0.8935098  0.3195859  2.7958 0.0055628 **
## L(ENERGY, 2)  0.6500646  0.3055851  2.1273 0.0343384 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

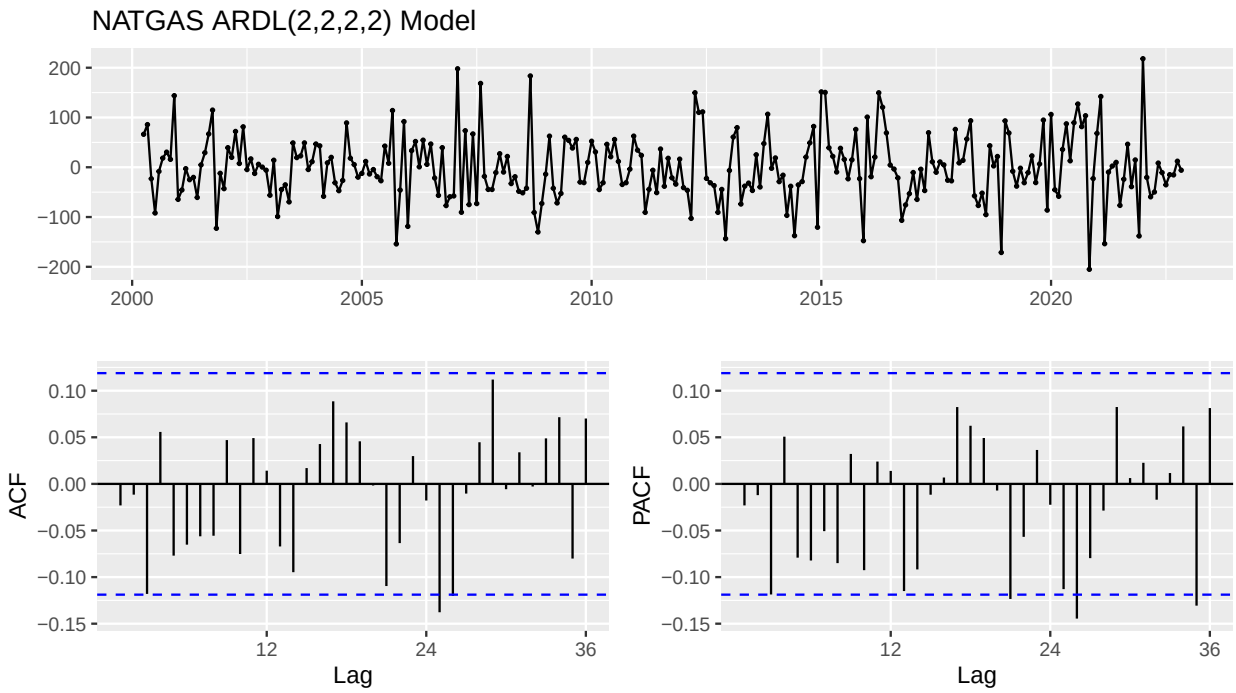
```r
# ARDL(2,4,4,1)
petro.ardl.mdl2 <- ardl(PETRO ~ VMT + RAIL + ENERGY, data=dataset, order=c(2,4,4,1))
coeftest(petro.ardl.mdl2)
```

```
##
## t test of coefficients:
##
##                Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   0.1431018  0.3910880  0.3659 0.7147380
## L(PETRO, 1)  -0.4199468  0.0610650 -6.8771 4.694e-11 ***
## L(PETRO, 2)  -0.1481839  0.0593495 -2.4968 0.0131636 *
## VMT           0.2861281  0.0742852  3.8518 0.0001484 ***
## L(VMT, 1)     0.1967929  0.0828415  2.3755 0.0182632 *
## L(VMT, 2)     0.2217688  0.0916628  2.4194 0.0162465 *
## L(VMT, 3)     0.1993275  0.0793440  2.5122 0.0126166 *
## L(VMT, 4)     0.1043139  0.0781322  1.3351 0.1830362
## RAIL          0.0126107  0.0154382  0.8169 0.4147759
## L(RAIL, 1)    0.0035064  0.0155240  0.2259 0.8214841
## L(RAIL, 2)   -0.0326146  0.0157735 -2.0677 0.0396790 *
## L(RAIL, 3)    0.0025922  0.0157076  0.1650 0.8690502
## L(RAIL, 4)   -0.0325498  0.0154907 -2.1012 0.0366009 *
## ENERGY        1.2312244  0.3088126  3.9870 8.745e-05 ***
## L(ENERGY, 1)  0.8852163  0.3181074  2.7828 0.0057930 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

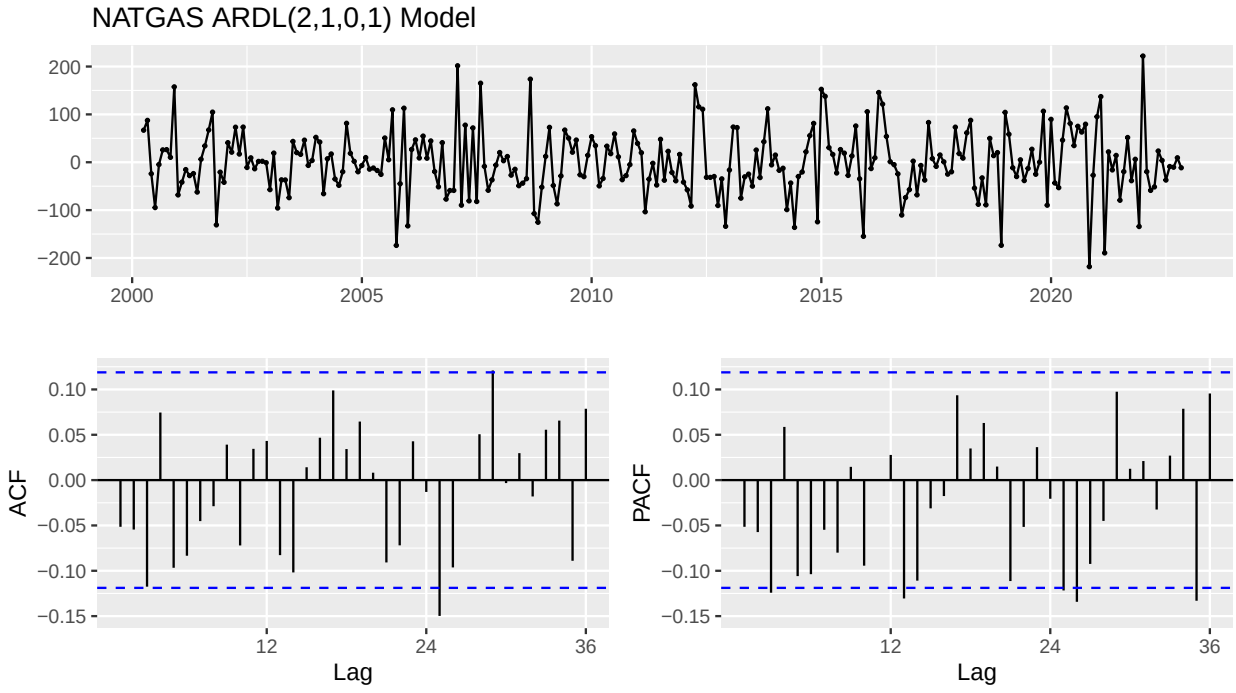
Surprisingly, our results show that `ENERGY` is the most important predictor in the dataset, more so than either `VMT` or `RAIL`. Unlike with `NATGAS`, it appears to be significant at lags as well. `VMT` is significant at least out to a lag of 3, while `RAIL` is barely significant in model 2 and not at all in model 1. These results are very surprising in light of our results for `NATGAS` and the economic intuition we started with.

4.1 Residual Analysis

```
ggtsdisplay(resid(natgas.ardl.mdl1), main="NATGAS ARDL(2,2,2,2) Model")
```

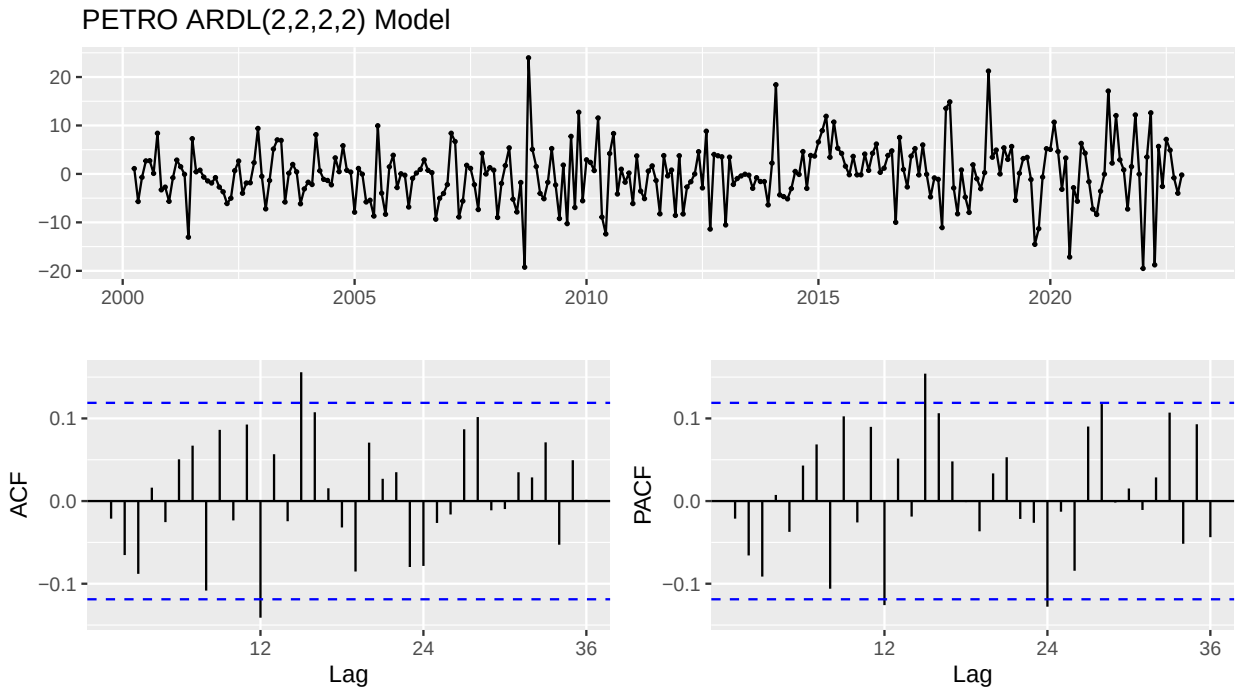



```
ggtsdisplay(resid(natgas.ardl.mdl2), main="NATGAS ARDL(2,1,0,1) Model")
```

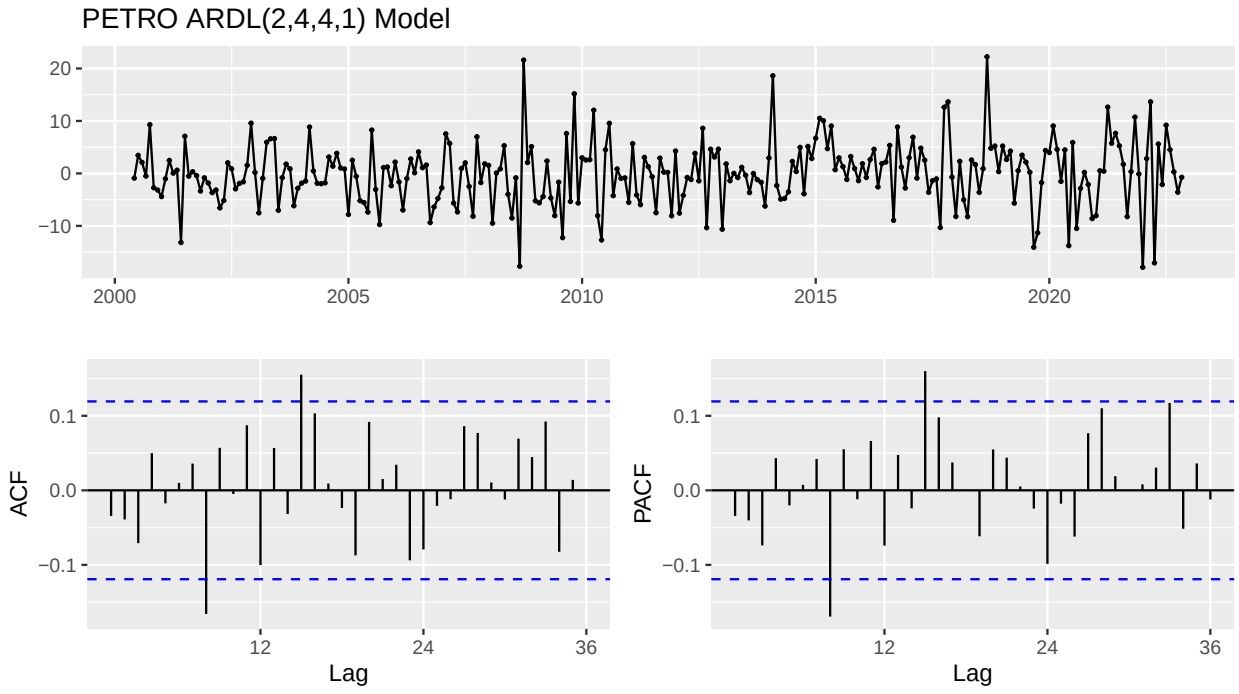


The residuals for both models look nearly identical, and are fairly good. We can see little difference in the ACF and PACF of both models. There is the possibility of a significant coefficient at a lag of 3 (which we saw in the original AR(2) model), but it looks like it barely fails to be significant. The only other significant lags are very high order.

```
ggtsdisplay(resid(petro.ardl.mdl1), main="PETRO ARDL(2,2,2,2) Model")
```




```
ggtsdisplay(resid(petro.ardl.mdl2), main="PETRO ARDL(2,4,4,1) Model")
```



The residuals of both models are clustered around zero, although we can definitely see a difference in variance after 2008 and especially around 2020. The ACF and PACF indicate significance at lag 12 in model 1 which is not present in model 2 (though a lag of 8 is). The ACF and PACF do not demonstrate any sort of pattern that would lead us to believe the model is misspecified.

4.2 Model Selection

We now define a recursive prediction function for our ARDL models:

```
recursive_predict.ardl <- function(object, data, ...) {
  # Extract the predictors in the right order
  data <- data[, names(coef(object))]
  npred <- dim(data)[1]
  preds <- ts(NA, start=start(data), end=end(data), freq=frequency(data))
  for (i in 1:npred) {
    preds[i] <- sum(data[i,] * coef(object))
  }
  return(preds)
}
```

Next we construct a train/test split and re-estimate our models:

```
# Train-test split
dataset.train <- window(dataset, end=start(dataset)+c(0,num_train-1))
dataset.test <- window(dataset, start=end(dataset)-c(0,num_test-1))

# Re-estimation
natgas.ardl.mdl1.test <- ardl(NATGAS ~ VMT + RAIL + ENERGY, data=dataset.train,
                              order=c(2,2,2,2))
natgas.ardl.mdl2.test <- ardl(NATGAS ~ VMT + RAIL + ENERGY, data=dataset.train,
                              order=c(2,1,0,1))
petro.ardl.mdl1.test <- ardl(PETRO ~ VMT + RAIL + ENERGY, data=dataset.train,
                             order=c(2,2,2,2))
petro.ardl.mdl2.test <- ardl(PETRO ~ VMT + RAIL + ENERGY, data=dataset.train,
                             order=c(2,4,4,1))
```

And finally make our recursive predictions:

```
# Construct a dataset with all the possible lags
dataset.lagged <- cbind(
  "(Intercept)"=ts(1, start=start(NATGAS), freq=frequency(NATGAS), end=end(NATGAS)),
  "NATGAS"=NATGAS, "L(NATGAS, 1)"=lag(NATGAS, -1), "L(NATGAS, 2)"=lag(NATGAS, -2),
  "PETRO"=PETRO, "L(PETRO, 1)"=lag(PETRO, -1), "L(PETRO, 2)"=lag(PETRO, -2),
  "VMT"=VMT, "L(VMT, 1)"=lag(VMT, -1), "L(VMT, 2)"=lag(VMT, -2),
  "L(VMT, 3)"=lag(VMT, -3), "L(VMT, 4)"=lag(VMT, -4),
  "RAIL"=RAIL, "L(RAIL, 1)"=lag(RAIL, -1), "L(RAIL, 2)"=lag(RAIL, -2),
  "L(RAIL, 3)"=lag(RAIL, -3), "L(RAIL, 4)"=lag(RAIL, -4),
  "ENERGY"=ENERGY, "L(ENERGY, 1)"=lag(ENERGY, -1),
  "L(ENERGY, 2)"=lag(ENERGY, -2), "L(ENERGY, 3)"=lag(ENERGY, -3))
# Clip to test period
dataset.lagged <- window(dataset.lagged, start=start(dataset.test), end=end(dataset.test))
```

```r
# Predict
natgas.ardl.mdl1.test.pred <- recursive_predict(natgas.ardl.mdl1.test, dataset.lagged)
natgas.ardl.mdl2.test.pred <- recursive_predict(natgas.ardl.mdl2.test, dataset.lagged)

natgas.ardl.mse <- rbind(
  "NATGAS ARDL(2,2,2,2)"=mean((NATGAS.test - natgas.ardl.mdl1.test.pred)^2),
  "NATGAS ARDL(2,1,0,1)"=mean((NATGAS.test - natgas.ardl.mdl2.test.pred)^2)
)
natgas.ardl.mse <- cbind(natgas.ardl.mse, c(mean(resid(natgas.ardl.mdl1.test)^2),
                                            mean(resid(natgas.ardl.mdl2.test)^2)))
colnames(natgas.ardl.mse) <- c("Test MSE", "Train MSE")
natgas.ardl.mse %>% kable()
```

	Test MSE	Train MSE
NATGAS ARDL(2,2,2,2)	18169.91	3529.921
NATGAS ARDL(2,1,0,1)	20062.00	3604.538

```r
#Predict
petro.ardl.mdl1.test.pred <- recursive_predict(petro.ardl.mdl1.test, dataset.lagged)
petro.ardl.mdl2.test.pred <- recursive_predict(petro.ardl.mdl2.test, dataset.lagged)

petro.ardl.mse <- rbind(
  "PETRO ARDL(2,2,2,2)"=mean((PETRO.test - petro.ardl.mdl1.test.pred)^2),
  "PETRO ARDL(2,4,4,1)"=mean((PETRO.test - petro.ardl.mdl2.test.pred)^2)
)
petro.ardl.mse <- cbind(petro.ardl.mse, c(mean(resid(petro.ardl.mdl1.test)^2),
                                          mean(resid(petro.ardl.mdl2.test)^2)))
colnames(petro.ardl.mse) <- c("Test MSE", "Train MSE")
petro.ardl.mse %>% kable()
```

	Test MSE	Train MSE
PETRO ARDL(2,2,2,2)	151.8054	29.10587
PETRO ARDL(2,4,4,1)	221.5068	28.87382

In both cases, the ARDL(2,2,2,2) model performs the best on the test MSE. This makes sense for `NATGAS`, which has more predictors in model 1, but `PETRO` has more statistically significant predictors in model 2 and still performs worse. The difference is large enough that we can conclusively say that model 1 is better for both series.

It is worth noting that both series perform significantly worse (nearly or more than double the test MSE) with ARDL models than with the AR models we tested in the previous section, regardless of the order. This could indicate that our predictor variables are weak, or that the large number of predictors we have is resulting in overfitting on the training data. Both models have a lower train MSE than the AR(p) models (which makes sense, as ARDL(p,...) encompases AR(p)). In either case, the AR models seem to be better fits to the data.

```r
AIC(natgas.ardl.mdl1, natgas.ardl.mdl2) %>% kable()
```

	df	AIC
natgas.ardl.mdl1	13	3077.020
natgas.ardl.mdl2	9	3075.695

```
BIC(natgas.ardl.mdl1, natgas.ardl.mdl2) %>% kable()
```

	df	BIC
natgas.ardl.mdl1	13	3123.896
natgas.ardl.mdl2	9	3108.148

Both AIC and BIC agree in their selection of model 2, which gets rid of some statistically insignificant predictors. This makes sense based on the coefficient tests we performed. However, this is counter to our empirical results on the test set, which indicate that model 1 is better.

```
AIC(petro.ardl.mdl1, petro.ardl.mdl2) %>% kable()
```

	df	AIC
petro.ardl.mdl1	13	1788.075
petro.ardl.mdl2	16	1768.947

```
BIC(petro.ardl.mdl1, petro.ardl.mdl2) %>% kable()
```

	df	BIC
petro.ardl.mdl1	13	1834.951
petro.ardl.mdl2	16	1826.522

Both AIC and BIC again agree on model 2 as the better choice. Despite dropping a lag of `ENERGY` and adding insignificant lags of `RAIL`, it appears that the extra significant lags of `VMT` and `RAIL` make model 2 better. Once again, this is counter to our empirical results on the test set.

4.3 Forecast

To forecast our models, we first need estimates of the predictor variables for the next 10 periods. We will estimate using HoltWinters.

```
VMT.FORECAST <- forecast(HoltWinters(VMT), 10)$mean
RAIL.FORECAST <- forecast(HoltWinters(RAIL), 10)$mean
ENERGY.FORECAST <- forecast(HoltWinters(ENERGY), 10)$mean
VMT.FORECAST <- ts(c(VMT, VMT.FORECAST), start=start(VMT), freq=frequency(VMT))
RAIL.FORECAST <- ts(c(RAIL, RAIL.FORECAST), start=start(RAIL), freq=frequency(RAIL))
ENERGY.FORECAST <- ts(c(ENERGY, ENERGY.FORECAST), start=start(ENERGY), freq=frequency(ENERGY))
dataset.forecast <- cbind(
  "(Intercept)"=ts(1, start=start(NATGAS), freq=frequency(NATGAS), end=end(VMT.FORECAST)),
  "NATGAS"=NATGAS, "L(NATGAS, 1)"=lag(NATGAS, -1), "L(NATGAS, 2)"=lag(NATGAS, -2),
  "PETRO"=PETRO, "L(PETRO, 1)"=lag(PETRO, -1), "L(PETRO, 2)"=lag(PETRO, -2),
```

```
  "VMT"=VMT.FORECAST, "L(VMT, 1)"=lag(VMT.FORECAST, -1),
  "L(VMT, 2)"=lag(VMT.FORECAST, -2), "L(VMT, 3)"=lag(VMT.FORECAST, -3),
  "L(VMT, 4)"=lag(VMT.FORECAST, -4),
  "RAIL"=RAIL.FORECAST, "L(RAIL, 1)"=lag(RAIL.FORECAST, -1),
  "L(RAIL, 2)"=lag(RAIL.FORECAST, -2), "L(RAIL, 3)"=lag(RAIL.FORECAST, -3),
  "L(RAIL, 4)"=lag(RAIL.FORECAST, -4),
  "ENERGY"=ENERGY.FORECAST, "L(ENERGY, 1)"=lag(ENERGY.FORECAST, -1),
  "L(ENERGY, 2)"=lag(ENERGY.FORECAST, -2), "L(ENERGY, 3)"=lag(ENERGY.FORECAST, -3))
dataset.forecast <- window(dataset.forecast, start=end(VMT)+c(0,1), end=end(VMT)+c(0,10))
```

Now that we have collected all of our "new" data into one object, let's define a forecasting function. R doesn't provide a forecast function for `ardl` objects, and neither can we recursively forecast using `dynlm` or base `lm` objects (e.g. with `forecast.lm` or `predict.lm`) with lagged series, since they expect all of the exogenous data to be available, including lags. Instead, we must write our own simplified recursive forecasting function:

```
my_forecast <- function(model, newdata, h=10, level=c(80, 95)) {
  # Extract the predictors in the right order
  preds <- newdata[, names(model$coefficients)]
  # Extract the dependent variable
  dep <- model$parsed_formula$y_part$var
  dep.order <- model$order[1]
  # Extract the standard deviation and df
  res.sd <- sd(resid(model))
  res.df <- model$df.residual
  empty_series <- ts(NA, start=start(preds), end=start(preds)+c(0,h-1), freq=12)
  forecasts.df <- ts.union("Point Forecast"=empty_series, "DUMMY"=empty_series)
  sds <- empty_series
  # Create empty series for forecast levels
  for (l in level) {
    newcols <- append(colnames(forecasts.df), c(paste("Lo", l), paste("Hi", l)))
    forecasts.df <- ts.union(forecasts.df, empty_series, empty_series)
    colnames(forecasts.df) <- newcols
  }
  for (i in 1:h) {
    # Get point forecast
    y_pred <- sum(preds[i,] * coef(model))
    forecasts.df[i, "Point Forecast"] <- y_pred
    # Update lags (if present)
    for (j in 1:h) {
      lagged_series <- paste("L(", dep, ", ", j, ")", sep="")
      if (lagged_series %in% colnames(preds) && i+j <= h) {
        preds[i+j,lagged_series] <- y_pred
      }
    }
    # Make confidence interval predictions
    sd <- res.sd
    for (lo in 0:dep.order) {
      if (lo > 0 && i - lo > 0) { # SD increases when depending on previous forecasts
        sd <- sd + sqrt(coef(model)[paste("L(", dep, ", ", lo, ")", sep="")]^2 *
                        sds[i-lo]^2)
      }
    }
    for (l in level) {
```

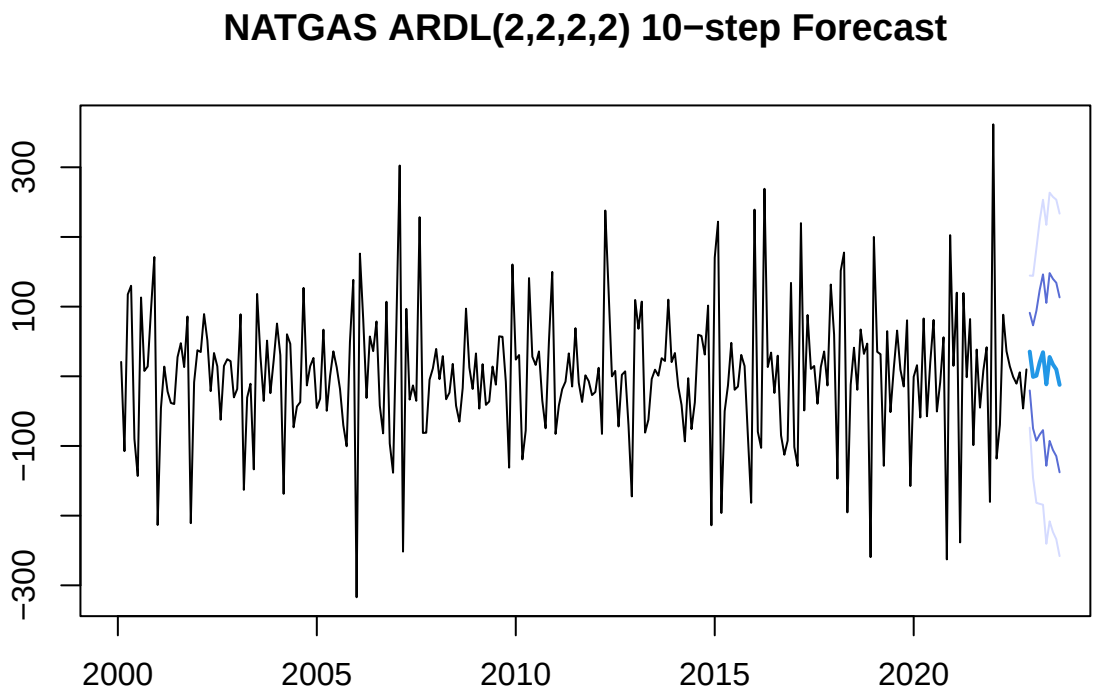
```
      bounds <- y_pred + qt(l/100, res.df) * sd * c(-1, 1)
      forecasts.df[i, c(paste("Lo", l), paste("Hi", l))] <- bounds
    }
    sds[i] <- sd
  }
  forecasts.df <- forecasts.df[, colnames(forecasts.df) != "DUMMY"]
  return(forecasts.df)
}
```

With our forecasting function painstakingly defined, we can finally make a 10-step ahead forecast for both series.

```
natgas.ardl.fore1 <- my_forecast(natgas.ardl.mdl1, dataset.forecast, 10)
natgas.ardl.fore1 %>% kable()
```

Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
35.4146048	-20.32076	91.14996	-73.72381	144.5530
-0.8470868	-74.88954	73.19536	-145.83361	144.1394
0.8139189	-92.36302	93.99086	-181.64088	183.2687
20.1010444	-83.67076	123.87285	-183.10014	223.3022
34.4845341	-77.27197	146.24104	-184.35191	253.3210
-11.5205371	-128.39398	105.35291	-240.37674	217.3357
27.6798748	-92.75406	148.11381	-208.14833	263.5081
17.0163062	-105.79176	139.82438	-223.46081	257.4934
9.8249276	-114.60118	134.25103	-233.82054	253.4704
-12.2654659	-137.78195	113.25102	-258.04608	233.5151

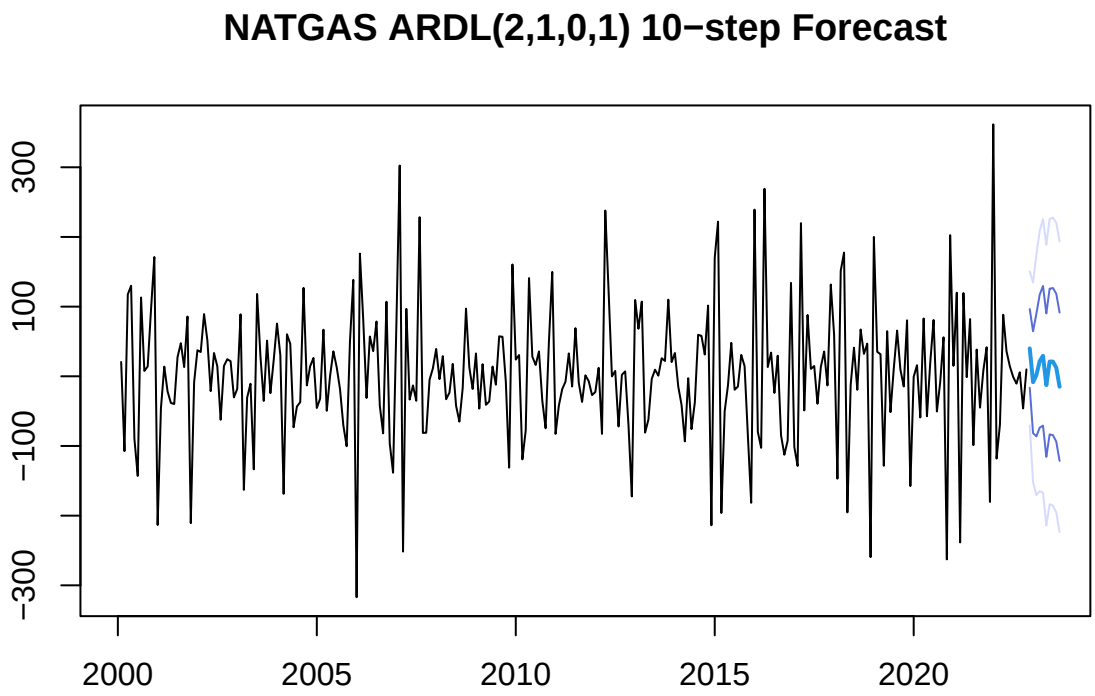
```
plot(NATGAS, xlim=c(2000, 2023.5), xlab="", ylab="",
     main="NATGAS ARDL(2,2,2,2) 10-step Forecast")
lines(natgas.ardl.fore1[,"Point Forecast"], col=4, lwd=2)
lines(natgas.ardl.fore1[,"Lo 80"], col="#596DD5", lwd=1)
lines(natgas.ardl.fore1[,"Hi 80"], col="#596DD5", lwd=1)
lines(natgas.ardl.fore1[,"Lo 95"], col="#D5DBFF", lwd=1)
lines(natgas.ardl.fore1[,"Hi 95"], col="#D5DBFF", lwd=1)
```




```
natgas.ardl.fore2 <- my_forecast(natgas.ardl.mdl2, dataset.forecast, 10)
natgas.ardl.fore2 %>% kable()
```

Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
40.049536	-16.37254	96.47162	-70.43035	150.5294
-8.427369	-81.52628	64.67155	-151.56212	134.7074
1.812484	-86.20119	89.82616	-170.52680	174.1518
21.709597	-73.66392	117.08312	-165.04098	208.4602
29.444894	-70.74360	129.63339	-166.73387	225.6237
-12.708326	-115.62254	90.20588	-214.22430	188.8077
21.177698	-83.39431	125.74971	-183.58441	225.9398
21.114726	-84.42968	126.65913	-185.55142	227.7809
12.296537	-93.82868	118.42175	-195.50689	220.1000
-14.981037	-121.45002	91.48794	-223.45759	193.4955

```
plot(NATGAS, xlim=c(2000, 2023.5), xlab="", ylab="",
     main="NATGAS ARDL(2,1,0,1) 10-step Forecast")
lines(natgas.ardl.fore2[,"Point Forecast"], col=4, lwd=2)
lines(natgas.ardl.fore2[,"Lo 80"], col="#596DD5", lwd=1)
lines(natgas.ardl.fore2[,"Hi 80"], col="#596DD5", lwd=1)
lines(natgas.ardl.fore2[,"Lo 95"], col="#D5DBFF", lwd=1)
lines(natgas.ardl.fore2[,"Hi 95"], col="#D5DBFF", lwd=1)
```



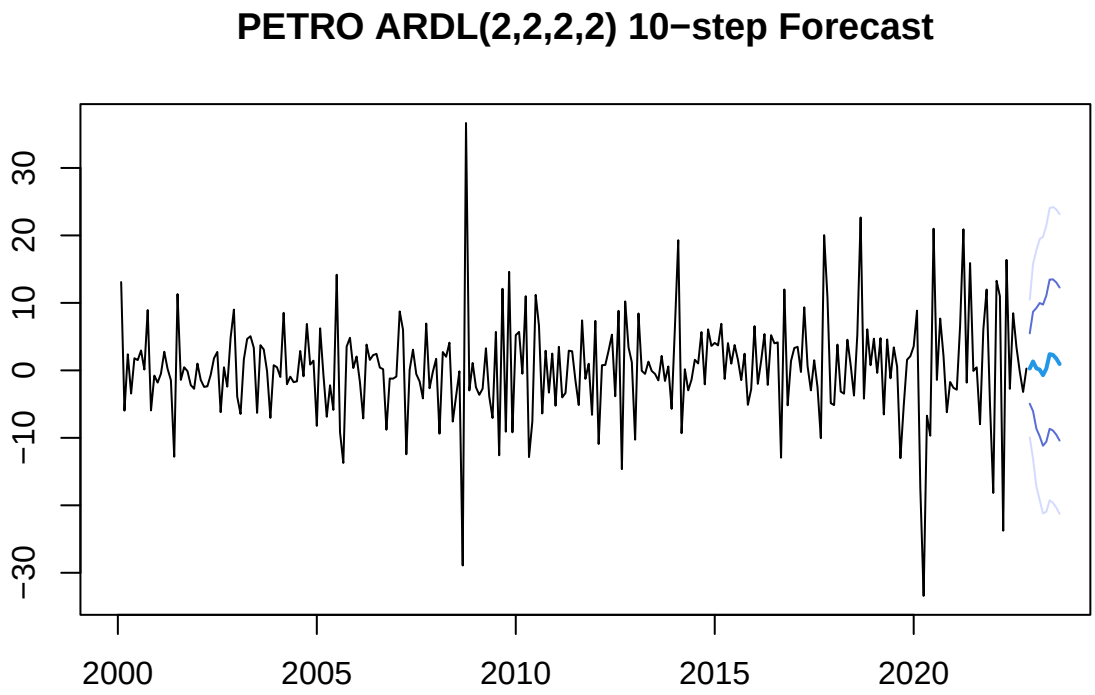
Both models have similar forecasts. The mean predictions are close to zero, but much more cyclical than the forecasts from the AR(p) models for NATGAS. This makes sense, as the low-order AR(p) models regressed quickly since they did not have exogenous variables. The confidence intervals of our ARDL model explode quickly as we take the recursive nature of our forecasts into account. The 95% confidence intervals cover a good portion of the post-2015 variance after a few steps.

```
petro.ardl.fore1 <- my_forecast(petro.ardl.mdl1, dataset.forecast, 10)
petro.ardl.fore1
```

```
##          Point Forecast      Lo 80     Hi 80      Lo 95    Hi 95
## Dec 2022      0.2737555  -4.939636  5.487147  -9.934868 10.48238
## Jan 2023      1.3219350  -6.053031  8.696901 -13.119385 15.76325
## Feb 2023      0.2993313  -8.646172  9.244835 -17.217342 17.81600
## Mar 2023      0.1118332  -9.764426  9.988092 -19.227403 19.45107
## Apr 2023     -0.7145383 -11.179842  9.750765 -21.207215 19.77814
## May 2023      0.2631095 -10.566809 11.093028 -20.943538 21.46976
## Jun 2023      2.4018253  -8.655457 13.459107 -19.250036 24.05369
## Jul 2023      2.2890338  -8.909678 13.487745 -19.639768 24.21784
## Aug 2023      1.7521429  -9.534615 13.038901 -20.349068 23.85335
## Sep 2023      0.9405245 -10.401032 12.282081 -21.267990 23.14904
```

```
plot(PETRO, xlim=c(2000, 2023.5), xlab="", ylab="",
     main="PETRO ARDL(2,2,2,2) 10-step Forecast")
lines(petro.ardl.fore1[,"Point Forecast"], col=4, lwd=2)
lines(petro.ardl.fore1[,"Lo 80"], col="#596DD5", lwd=1)
```

```
lines(petro.ardl.fore1[,"Hi 80"], col="#596DD5", lwd=1)
lines(petro.ardl.fore1[,"Lo 95"], col="#D5DBFF", lwd=1)
lines(petro.ardl.fore1[,"Hi 95"], col="#D5DBFF", lwd=1)
```

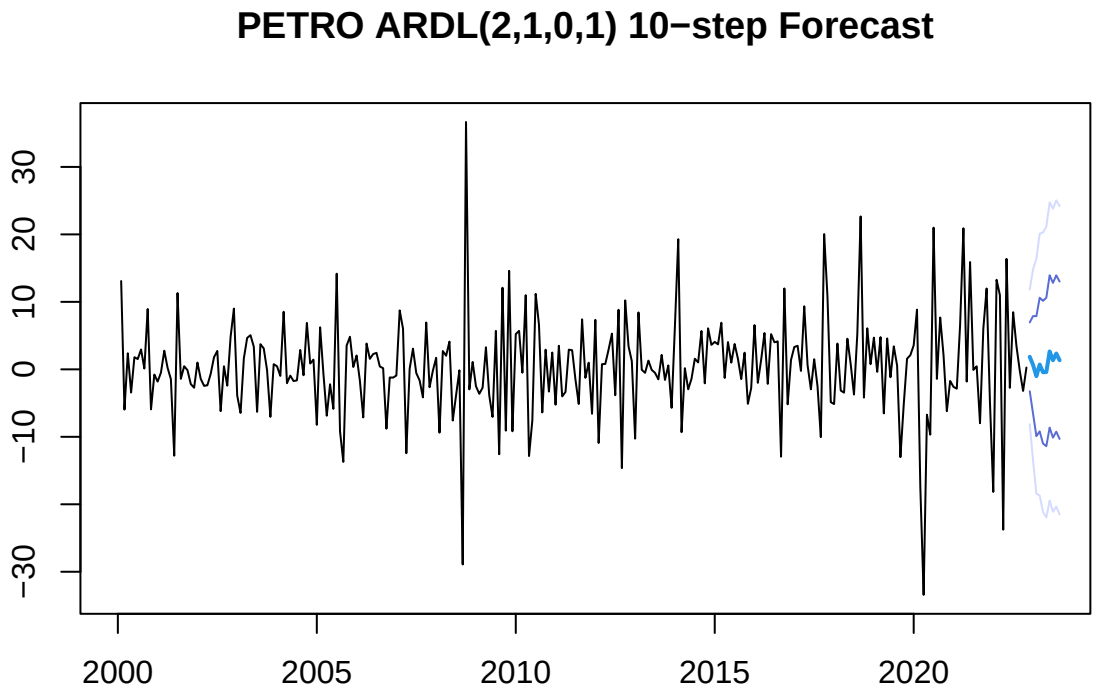



```
petro.ardl.fore2 <- my_forecast(petro.ardl.mdl2, dataset.forecast, 10)
petro.ardl.fore2
```

```
##          Point Forecast      Lo 80     Hi 80      Lo 95    Hi 95
## Dec 2022      1.8565194  -3.240909  6.953948  -8.125408 11.83845
## Jan 2023      0.6629137  -6.575163  7.900991 -13.510893 14.83672
## Feb 2023     -1.0118253  -9.904218  7.880567 -18.425158 16.40151
## Mar 2023      0.7103579  -9.193968 10.614684 -18.684573 20.10529
## Apr 2023     -0.4279327 -11.002360 10.146495 -21.135074 20.27921
## May 2023     -0.3912763 -11.397063 10.614510 -21.943118 21.16057
## Jun 2023      2.6522559  -8.633977 13.938489 -19.448763 24.75327
## Jul 2023      1.3495350 -10.118391 12.817461 -21.107280 23.80635
## Aug 2023      2.3460948  -9.239690 13.931880 -20.341515 25.03370
## Sep 2023      1.3464563 -10.315747 13.008659 -21.490798 24.18371
```

```
plot(PETRO, xlim=c(2000, 2023.5), xlab="", ylab="",
     main="PETRO ARDL(2,1,0,1) 10-step Forecast")
lines(petro.ardl.fore2[,"Point Forecast"], col=4, lwd=2)
lines(petro.ardl.fore2[,"Lo 80"], col="#596DD5", lwd=1)
lines(petro.ardl.fore2[,"Hi 80"], col="#596DD5", lwd=1)
```

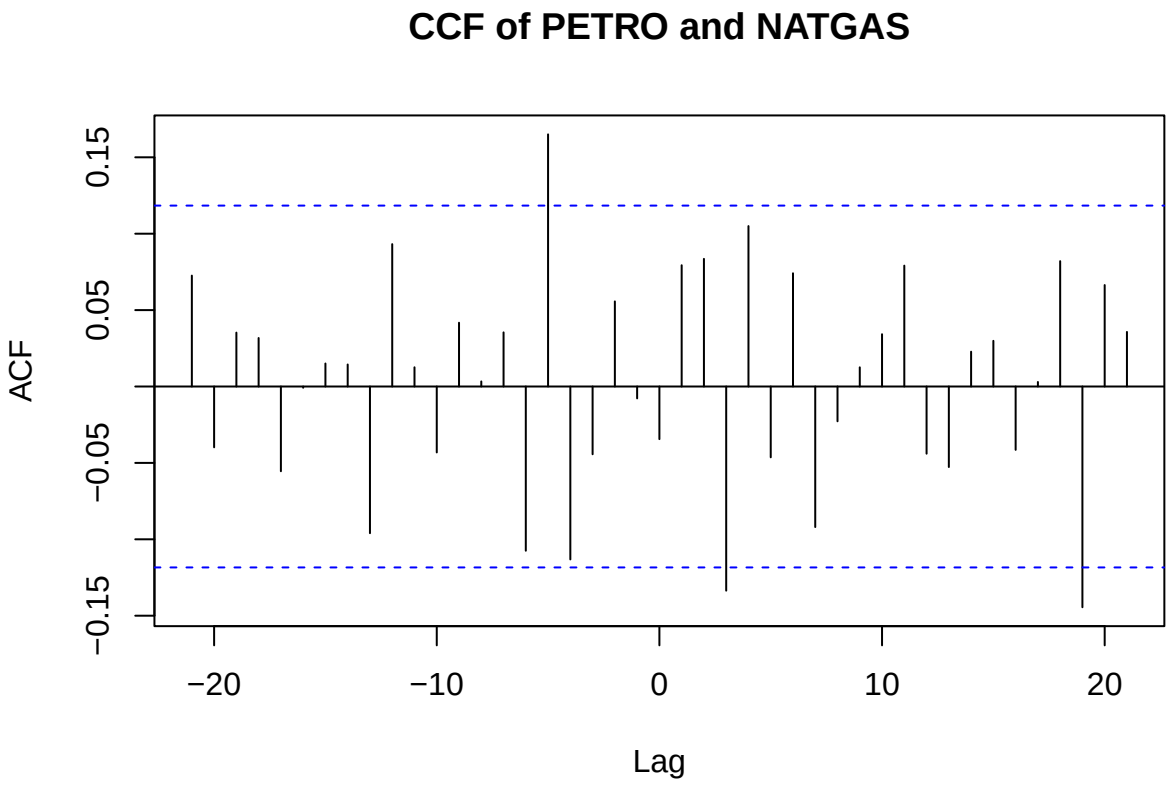
```
lines(petro.ardl.fore2[,"Lo 95"], col="#D5DBFF", lwd=1)
lines(petro.ardl.fore2[,"Hi 95"], col="#D5DBFF", lwd=1)
```



Once again, both models follow a similar trend. The forecasts from model 2 are a bit more jagged. The mean predictions are small, but the 95% confidence interval encompasses most of the variation post-2020, so these forecasts look relatively good.

5. Vector Autoregressive VAR(p) Model

```
ccf(as.numeric(PETRO), as.numeric(NATGAS), main="CCF of PETRO and NATGAS")
```



The CCF does not show many significant lags between `PETRO` and `NATGAS`. There are only two significant lags between -12 and 12. This may present a problem for our VAR model if the variables do not exhibit any shared dynamics, which we will now explore.

5.1 Granger-Causality

We have a large number of possible orders to check with a Granger causality test. Since low orders of lags worked with the AR(p) models before, we will try similar low orders - 2 and 4.

```
grangertest(NATGAS ~ PETRO, order=2)
```

```
## Granger causality test
##
## Model 1: NATGAS ~ Lags(NATGAS, 1:2) + Lags(PETRO, 1:2)
## Model 2: NATGAS ~ Lags(NATGAS, 1:2)
##   Res.Df Df      F Pr(>F)
## 1    267
## 2    269 -2 0.3537 0.7024
```

```
grangertest(PETRO ~ NATGAS, order=2)
```

```
## Granger causality test
##
## Model 1: PETRO ~ Lags(PETRO, 1:2) + Lags(NATGAS, 1:2)
```

```
## Model 2: PETRO ~ Lags(PETRO, 1:2)
##   Res.Df Df      F  Pr(>F)
## 1    267
## 2    269 -2 3.2371 0.04082 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(NATGAS ~ PETRO, order=4)
```

```
## Granger causality test
##
## Model 1: NATGAS ~ Lags(NATGAS, 1:4) + Lags(PETRO, 1:4)
## Model 2: NATGAS ~ Lags(NATGAS, 1:4)
##   Res.Df Df      F Pr(>F)
## 1    261
## 2    265 -4 1.5918 0.1768
```

```
grangertest(PETRO ~ NATGAS, order=4)
```

```
## Granger causality test
##
## Model 1: PETRO ~ Lags(PETRO, 1:4) + Lags(NATGAS, 1:4)
## Model 2: PETRO ~ Lags(PETRO, 1:4)
##   Res.Df Df      F  Pr(>F)
## 1    261
## 2    265 -4 2.5177 0.04182 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thankfully, the Granger causality test indicates that `NATGAS` Granger-causes `PETRO` at both orders of 2 and 4. The reverse is not true - `PETRO` does not Granger-cause `NATGAS`. It appears that we can proceed with our estimation of a VAR model.

5.2 Estimation

Since we observed Granger causality for at orders of both 2 and 4, we could reasonably choose either order for our VAR model. We will use AIC to determine the order of our model.

```
VARselect(ts.union(PETRO, NATGAS))
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
##
## $criteria
##                  1           2           3           4           5
## AIC(n)     13.01348    12.92151    12.92887    12.92163    12.92259
## HQ(n)      13.04614    12.97593    13.00507    13.01961    13.04233
## SC(n)      13.09476    13.05696    13.11851    13.16545    13.22059
## FPE(n) 448420.34220 409018.03439 412049.23418 409088.12894 409497.61118
##                  6           7           8           9          10
```

```
## AIC(n)      12.92221     12.92268     12.93924     12.93655     12.9576
## HQ(n)       13.06372     13.08597     13.12430     13.14338     13.1862
## SC(n)       13.27439     13.32904     13.39978     13.45127     13.5265
## FPE(n) 409367.55798 409594.83222 416481.31643 415421.27539 424334.2961
```

It appears that order 2 is the best fit. We proceed with estimating a VAR(2) model:

```
var.mdl1 <- VAR(ts.union(NATGAS, PETRO), p=2)
coeftest(var.mdl1)
```

```
##
## t test of coefficients:
##
##                       Estimate Std. Error t value  Pr(>|t|)
## NATGAS:(Intercept)  5.4836334  5.2965950  1.0353   0.30146
## NATGAS:NATGAS.l1   -0.4107671  0.0579629 -7.0867 1.224e-11 ***
## NATGAS:PETRO.l1     0.2693504  0.7345152  0.3667   0.71413
## NATGAS:NATGAS.l2   -0.3100696  0.0581358 -5.3335 2.055e-07 ***
## NATGAS:PETRO.l2     0.6001032  0.7291253  0.8230   0.41122
## PETRO:(Intercept)   0.4540894  0.4356837  1.0422   0.29824
## PETRO:NATGAS.l1     0.0088163  0.0047679  1.8491   0.06555 .
## PETRO:PETRO.l1     -0.2646441  0.0604193 -4.3801 1.705e-05 ***
## PETRO:NATGAS.l2     0.0107107  0.0047821  2.2397   0.02593 *
## PETRO:PETRO.l2     -0.0767743  0.0599759 -1.2801   0.20163
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The coefficients of our VAR(1) model mostly mirror the coefficients we found in the separate AR(p) models in the previous section. For example, we can see in the model for `NATGAS` that the first two lags are significant, with values that are nearly identical to the AR(2) model for `NATGAS`. The lags of `PETRO` are not significant at all. The model for `PETRO` is more interesting - only the first lag of `PETRO` is significant (with almost the same value as the AR(1) model), and we can see that the first lag of `NATGAS` just barely fails at the 5% significance level and the second lag is significant. This reinforces the results that we previously saw that `NATGAS` Granger-causes `PETRO`.

```
plot(var.mdl1)
```

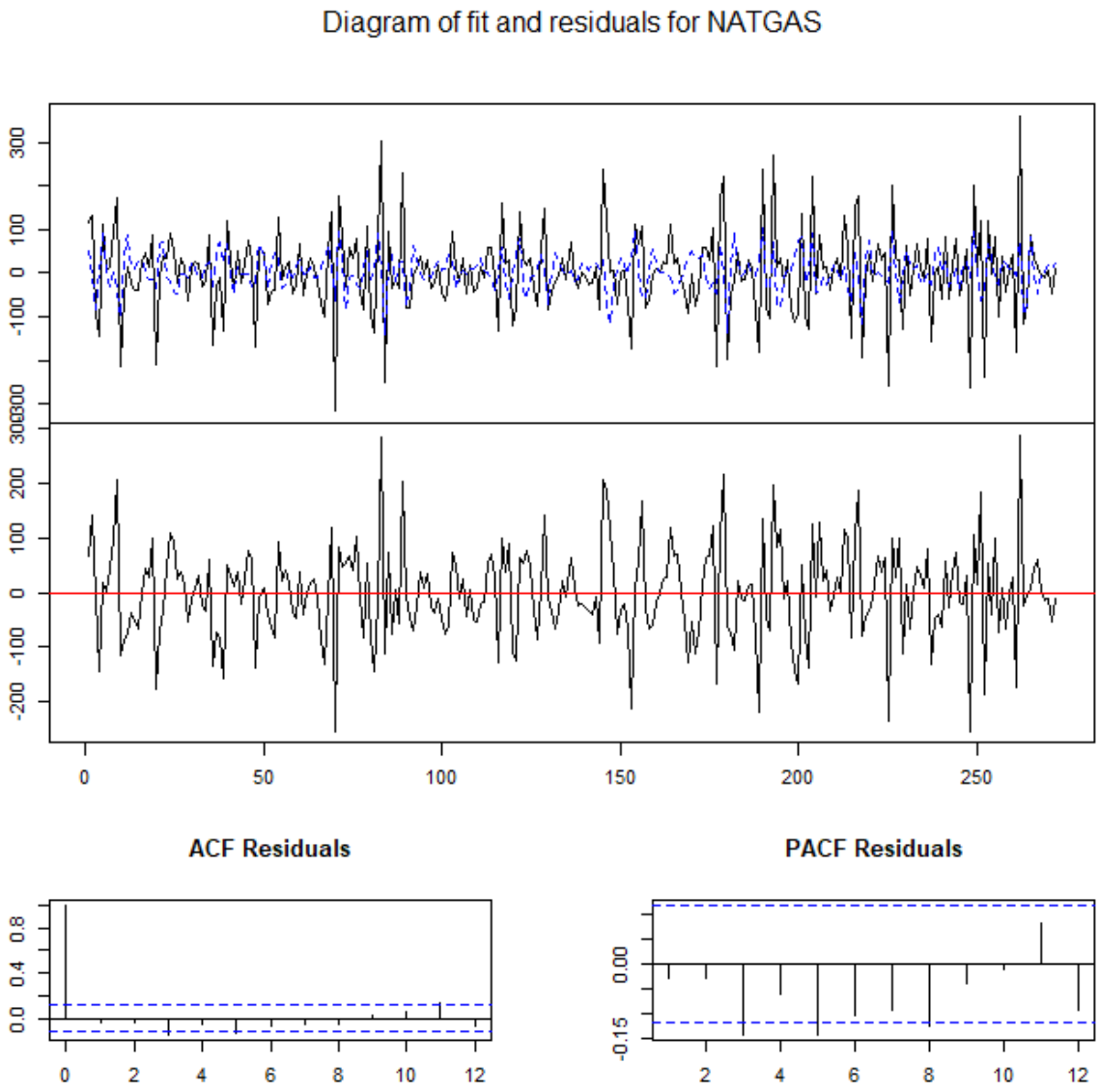
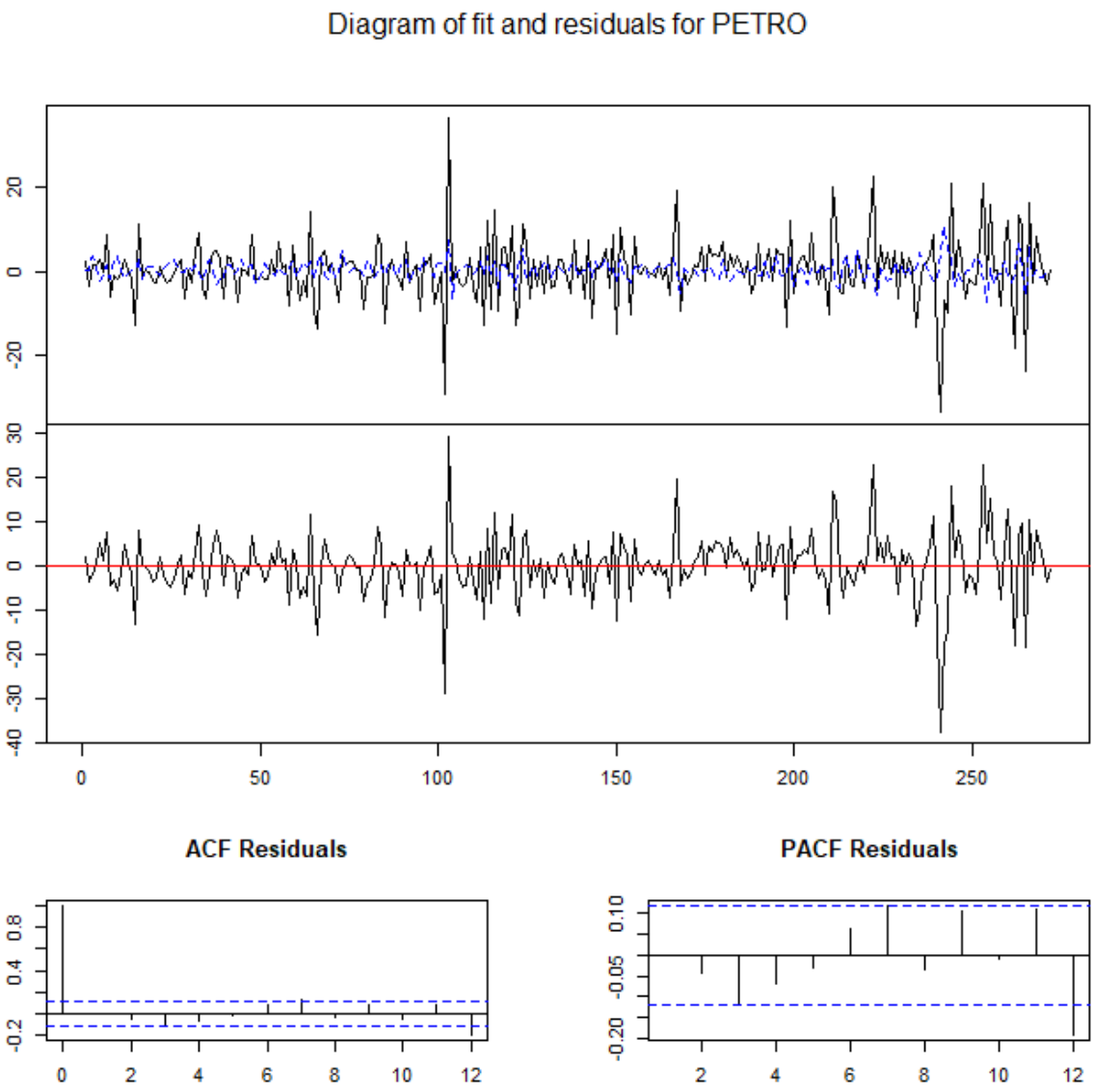


Diagram of fit and residuals for NATGAS
300
100
0
-100
300
200
100
0
-100
-200
0
50
100
150
200
250
ACF Residuals
0.8
0.4
0.0
0
2
4
6
8
10
12
PACF Residuals
0.00
-0.15
2
4
6
8
10
12

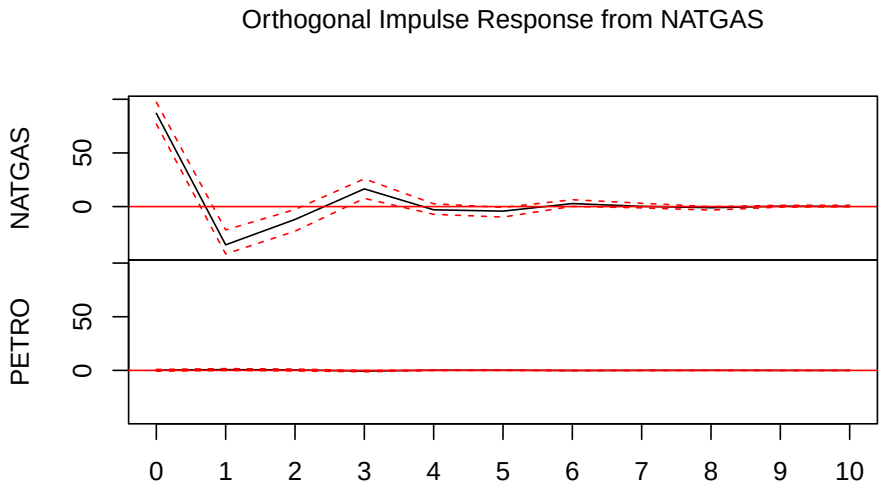


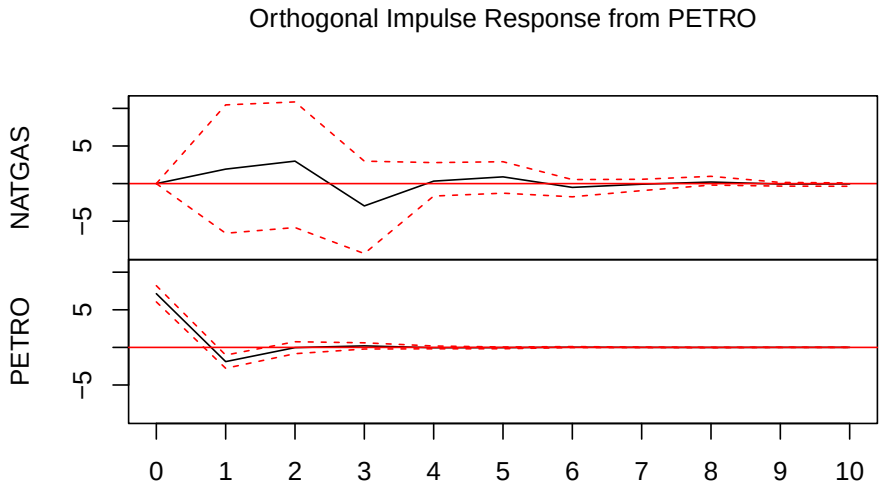
The fit for `NATGAS` fails to capture the most extreme variation in the data. From the PACF we can see that the lags at 3 and 5 are just barely significant, and the lag at 8 is questionable.

`PETRO` has similar results for the fitted values. The values in the ACF/PACF are better, with the only significant lag showing at 12.

5.3 Impulse Response

```
plot(irf(var.mdl1))
```





The impluse response function for `NATGAS` shows a very strong positive immediate effect on `NATGAS`, followed by a negative effect in the second period. After that, it briefly becomes positive again before leveling off. The effect on `PETRO` is much smaller by comparison (which we can see from the small coefficients in the `PETRO` model).

The response for `PETRO` on `PETRO` also starts positive and then becomes negative before flattening out. The effect on `NATGAS` has too much variance to determine that it is different from zero with confidence.

5.4 Model Evaluation

We now define a recursive prediction function for our VAR model:

```
recursive_predict.varest <- function(object, data, ...) {
  coefs <- coef(object)
  npred <- dim(data)[1]
  pred_list <- list()
  for (series in names(coefs)) {
    # Forecast each series in turn
    series_coef <- coefs[[series]][,1]
    series_data <- data[, names(series_coef)]
    preds <- ts(NA, start=start(data), end=end(data), freq=frequency(data))
    for (i in 1:npred) {
      preds[i] <- sum(series_data[i,] * series_coef)
    }
    pred_list[[series]] <- preds
  }
  return(pred_list)
}
```

And re-estimate the model:

```
var.mdl1.test <- VAR(ts.union("NATGAS"=NATGAS.train, "PETRO"=PETRO.train), p=2)
```

And finally make our recursive predictions:

```
# Construct a dataset with necessary lags
var.dataset.lagged <- ts.union(
  "NATGAS.l1"=lag(NATGAS, -1), "NATGAS.l2"=lag(NATGAS, -2),
  "PETRO.l1"=lag(PETRO, -1), "PETRO.l2"=lag(PETRO, -2),
  "const"=ts(1, start=start(NATGAS), end=end(NATGAS), freq=frequency(NATGAS))
)
# Clip to test period
var.dataset.lagged <- window(var.dataset.lagged,
                             start=start(NATGAS.test), end=end(NATGAS.test))

# Predict
var.mdl1.test.pred <- recursive_predict(var.mdl1.test, var.dataset.lagged)

var.mse <- rbind(
  "NATGAS VAR(2)"=mean((NATGAS.test - var.mdl1.test.pred[["NATGAS"]])^2),
  "PETRO VAR(2)"=mean((PETRO.test - var.mdl1.test.pred[["PETRO"]])^2)
)
var.mse <- cbind(var.mse, c(mean(resid(var.mdl1.test)[,"NATGAS"]^2),
                            mean(resid(var.mdl1.test)[,"PETRO"]^2)))
colnames(var.mse) <- c("Test MSE", "Train MSE")
var.mse %>% kable()
```

	Test MSE	Train MSE
NATGAS VAR(2)	10011.81446	6502.33536

	Test MSE	Train MSE
PETRO VAR(2)	89.27814	34.18147

As expected, the test MSE is higher than the training MSE. The train MSE is just slightly lower for both models than the train MSEs from the AR(p) models. However, the test MSEs are slightly higher, indicating that the models did not generalize as well. While we can't compare the MSE values of the VAR models directly due to scale, the VAR(2) model for `PETRO` is about is good as the AR(p) models for `PETRO` while the model for `NATGAS` performs slightly worse, so it appears that the VAR model for `PETRO` is better than for `NATGAS`. This does make sense, as we found that `NATGAS` Granger-caused `PETRO` earlier, so the model for `PETRO` exhibits better performance.

```
AIC(var.mdl1)
```

```
## [1] 5052.085
```

```
BIC(var.mdl1)
```

```
## [1] 5088.143
```

We don't have any other VAR models to compare the AIC and BIC against, so we leave these values here.

5.5 Forecast

Thankfully, VAR models have an implementation of `predict.varest` already, so we are spared from writing another forecast function.

```
var.mdl1.fore <- predict(var.mdl1, 10)
ts(var.mdl1.fore$fcst$NATGAS, start=end(NATGAS)+c(0,1), freq=frequency(NATGAS))
```

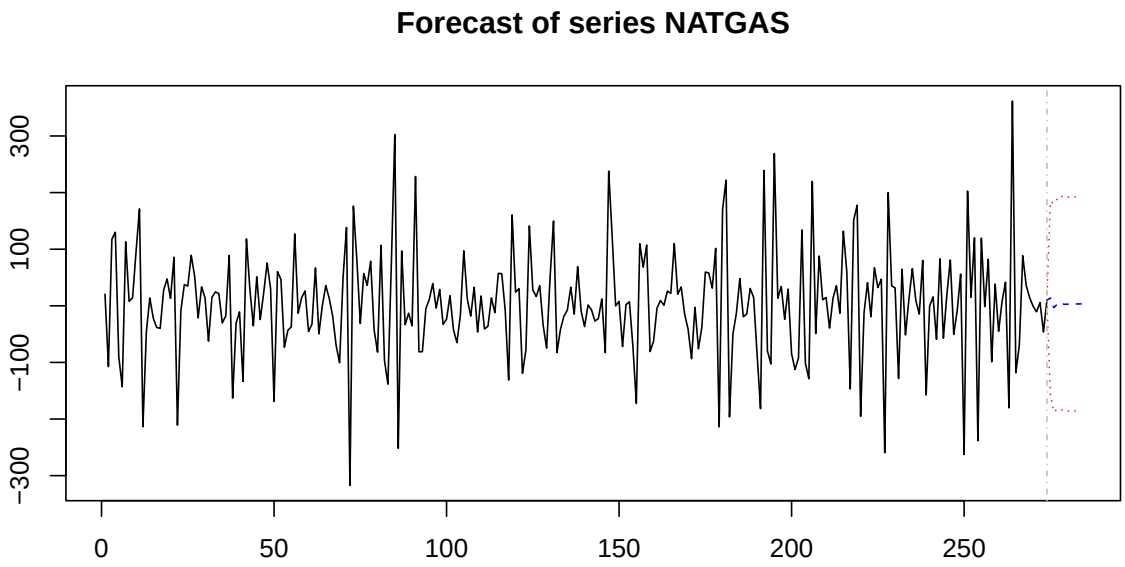
```
##               fcst     lower    upper       CI
## Dec 2022 13.896390 -156.3956 184.1884 170.2920
## Jan 2023 -3.071729 -187.1667 181.0232 184.0950
## Feb 2023  2.731178 -182.9488 188.4111 185.6800
## Mar 2023  5.783911 -182.7838 194.3516 188.5677
## Apr 2023  2.579514 -186.0805 191.2396 188.6600
## May 2023  2.921398 -185.9295 191.7723 188.8509
## Jun 2023  3.848161 -185.0854 192.7817 188.9335
## Jul 2023  3.338360 -185.5958 192.2725 188.9341
## Aug 2023  3.245272 -185.7002 192.1907 188.9455
## Sep 2023  3.455927 -185.4909 192.4028 188.9469
```

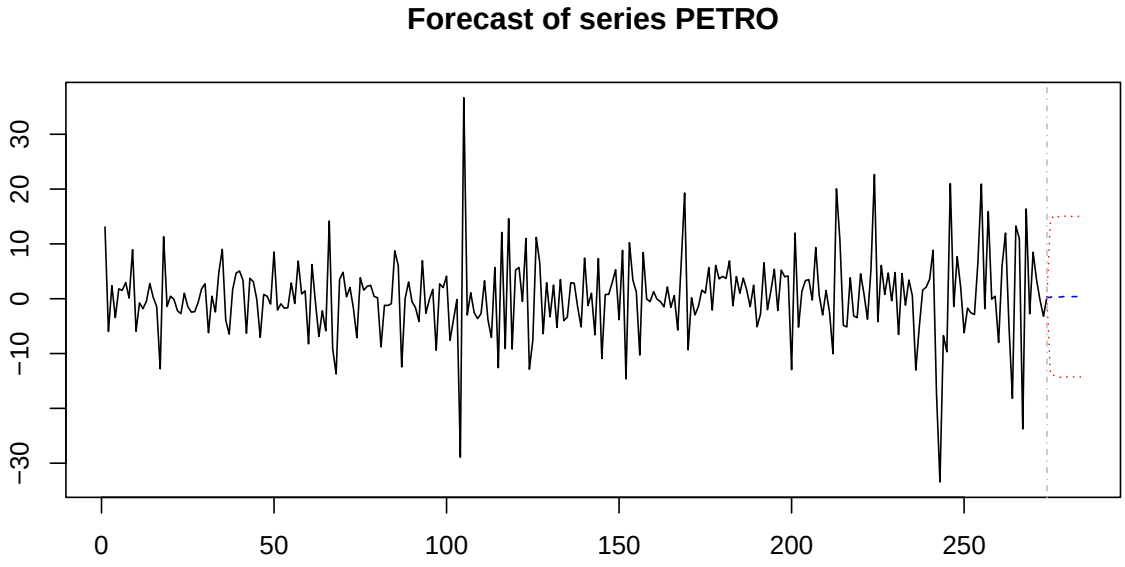
```
ts(var.mdl1.fore$fcst$PETRO, start=end(PETRO)+c(0,1), freq=frequency(PETRO))
```

```
##               fcst     lower    upper       CI
## Dec 2022 0.2200797 -13.78768 14.22784 14.00776
## Jan 2023 0.6035149 -13.95269 15.15972 14.55620
## Feb 2023 0.3992343 -14.17951 14.97798 14.57874
## Mar 2023 0.2932787 -14.34645 14.93300 14.63973
## Apr 2023 0.4260696 -14.21718 15.06932 14.64325
```

```
## May 2023 0.4035077 -14.24319 15.05020 14.64669
## Jun 2023 0.3679766 -14.28079 15.01674 14.64877
## Jul 2023 0.3909443 -14.25782 15.03971 14.64877
## Aug 2023 0.3930256 -14.25598 15.04203 14.64900
## Sep 2023 0.3844305 -14.26461 15.03347 14.64904
```

```
plot(var.mdl1.fore, plot.type="single")
```

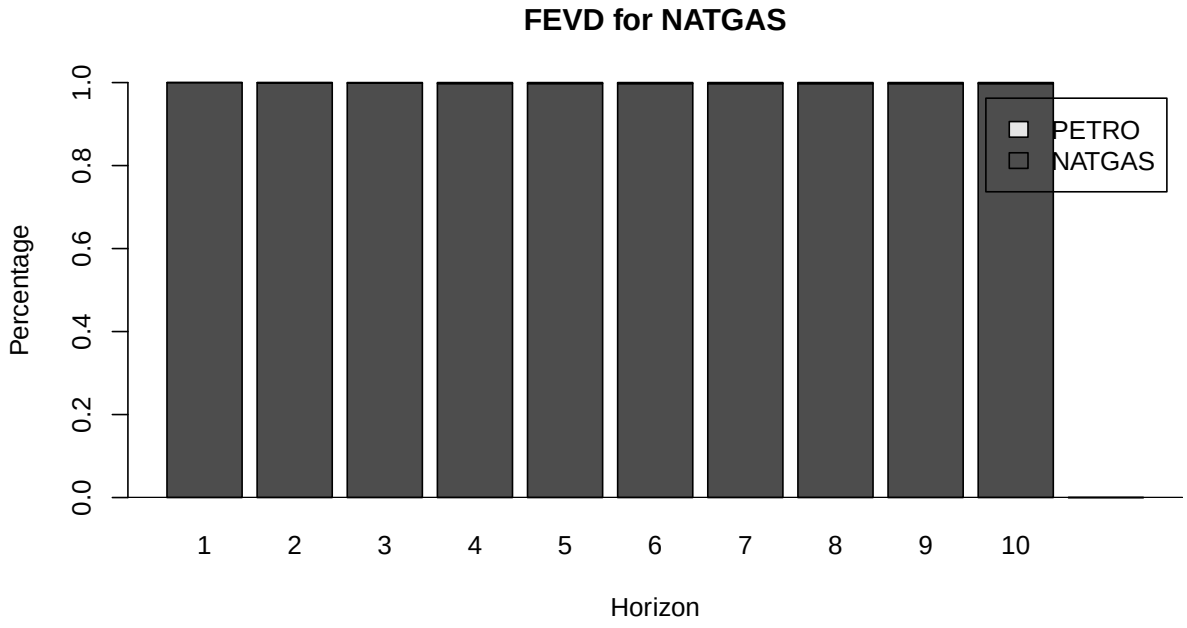


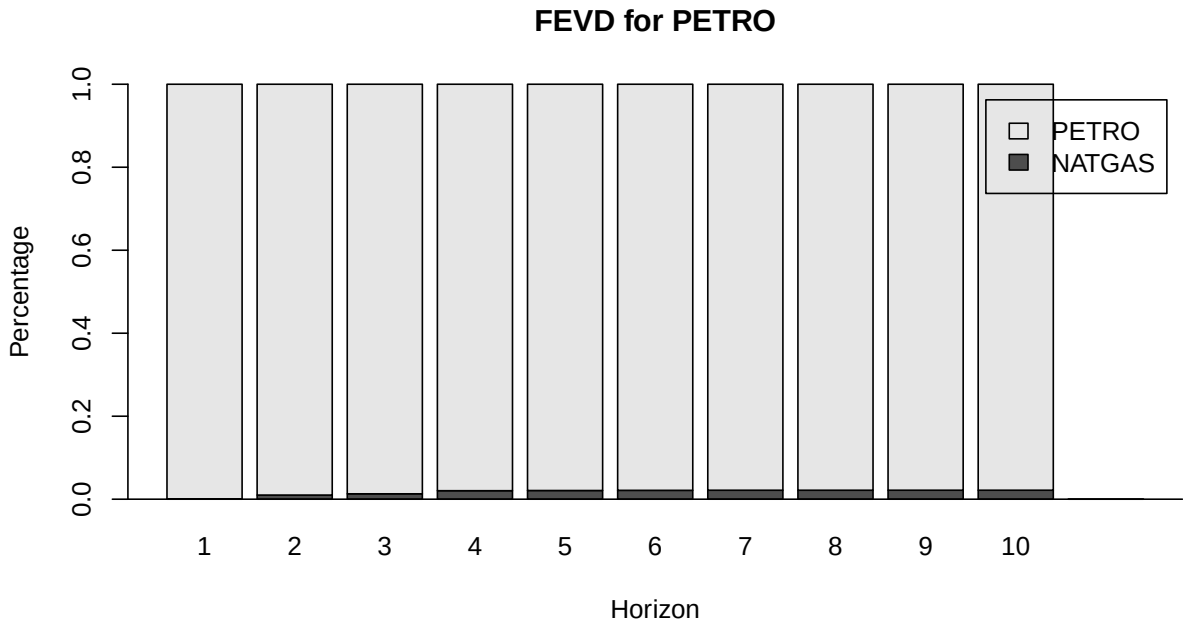


The VAR forecasts exhibit a similar trend to our forecasts from AR and ARDL models, but the magnitude

is lower. The prediction for Dec 2022 is still very positive, followed by a negative, but the spikes are not as drastic, and the forecasts converge very quickly. Similarly, the forecasts for PETRO exhibit very little variation in the short term, quickly converging on a long-term forecast.

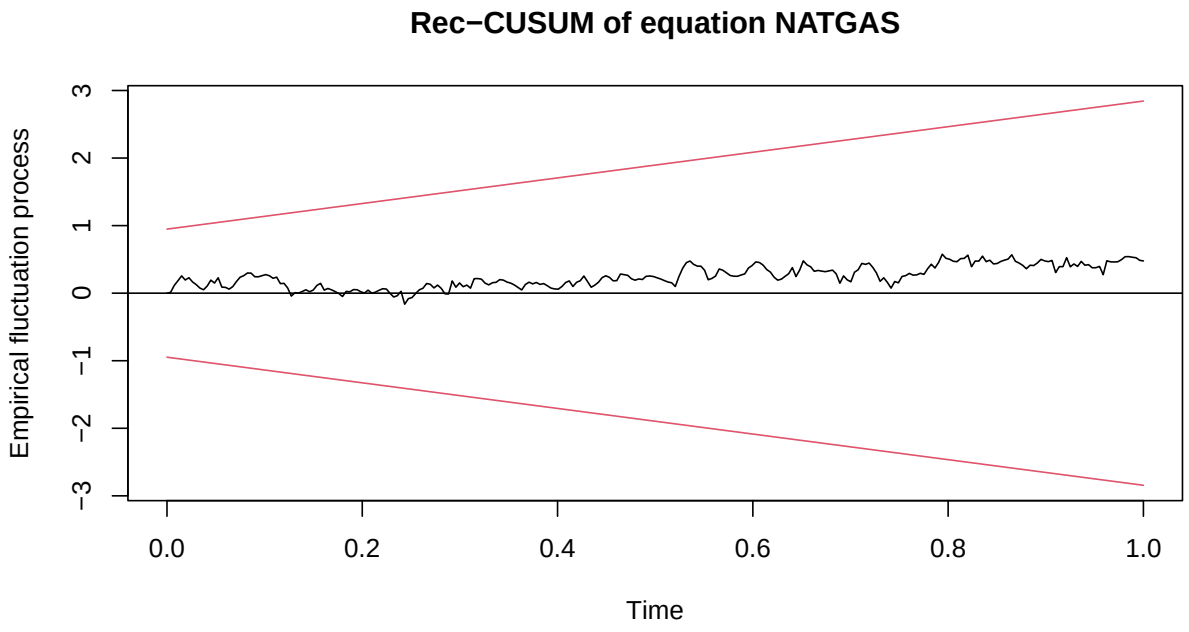
```
plot(fevd(var.mdl1, n.ahead=10), plot.type="single")
```

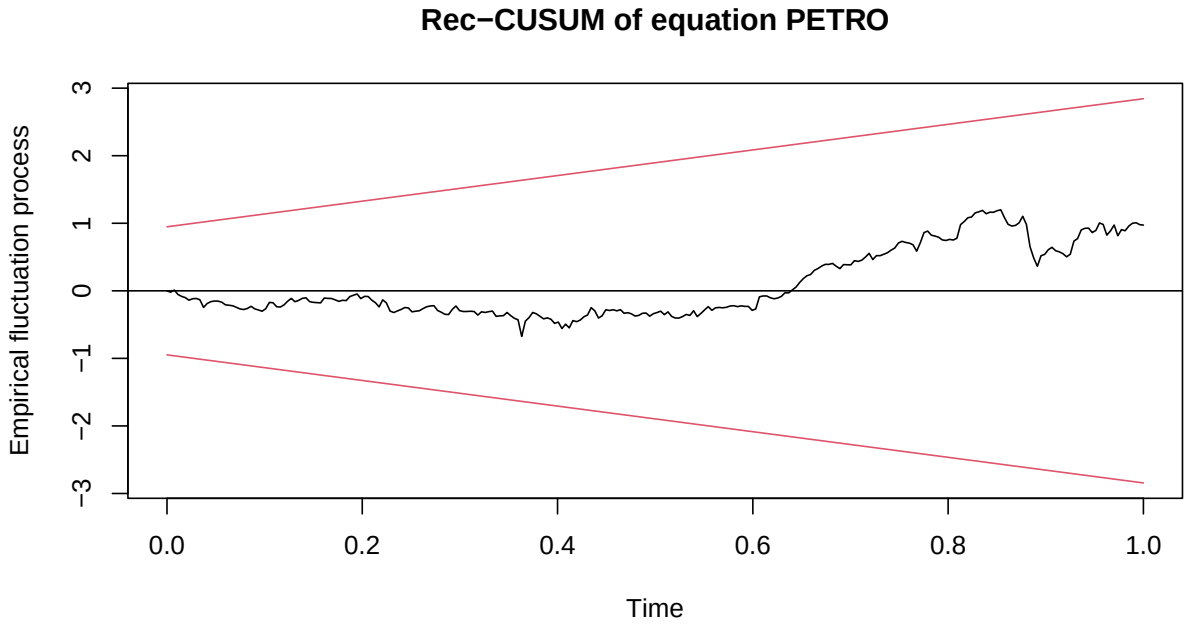




The error variance decomposition shows that the overwhelming majority of the variance for each series comes from itself. `NATGAS` barely makes an impact on `PETRO`, while `PETRO` makes no impact whatsoever on the variance of forecasts for `NATGAS`.

```
plot(stability(var.mdl1, type="Rec-CUSUM"), plot.type="single")
```





The Rec CUSUM plot for both series stays well within the significance bounds. `NATGAS` simmers in the low positives with a very slow trend, and `PETRO` starts negative before increasing somewhat towards the end of the window. Overall, our residual analysis looks good.

6. Conclusions

We tested three different model classes in our exploration of natural gas consumption and petroleum pipeline flow. Surprisingly, the best models for both series were the simple and parsimonious AR(p) models. The ARDL models we tried overfit to the test set and produced poor generalization. It is probable that this is due to inadequate choice of predictor variables on our part (possibly compounded by selection of the wrong orders for the models). Our VAR model performed about as well or slightly worse than the AR models, though we did determine that petroleum pipeline movement is Granger-caused by natural gas consumption.

6.1 Future Work

The most natural extension to our work is to search for better predictors for an ARDL model to surpass the performance of an AR model. It might be useful to examine more closely related series on the uses of natural gas and petroleum, rather than loosely-linked transportation data.

Additionally, we barely explored the model space available in all of our model classes. A more expansive search for the best order for each model could yield significant improvement in the predictive power of each model class, especially for the ARDL models, which have such a large model space.