Project 2

Dustin Oakes

12/9/2020

Loading Data

```
# Reading the data from FRED into a format for R
MFG.csv <- read.csv("IPGMFN.csv")
PERMIT.csv <- read.csv("PERMITNSA.csv")
```

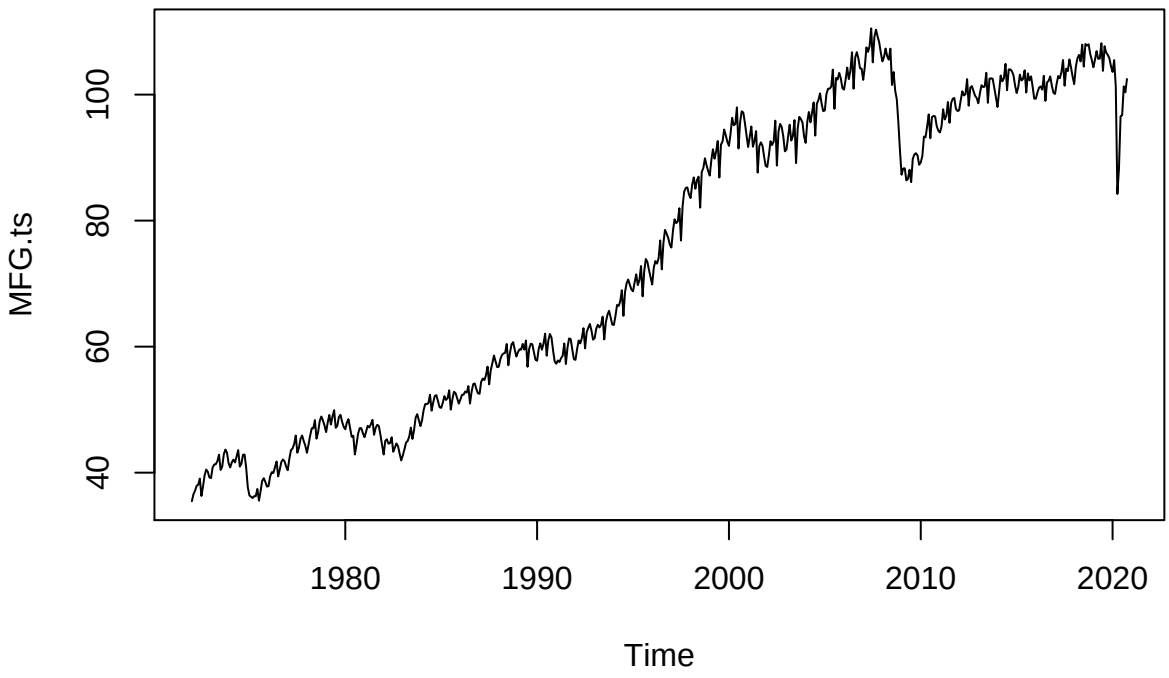
These data sets obtained from FRED include the Industrial Production: Manufacturing Index (IPGMFN) and New Private Housing Units Authorized by Building Permits (PERMITNSA) from January 1972 to October 2020. Our first thought is that an increase in manufacturing output will generally lead to more building materials purchases, hence an increase in Building Permits applied for and awarded. We will examine nearly 50 years of data to see if there is any meaningful trend we can find that relates changes in manufacturing to changes in the number of homes being built (or at the very least, being permitted).

1. Time Series & ACF/PACF Plots

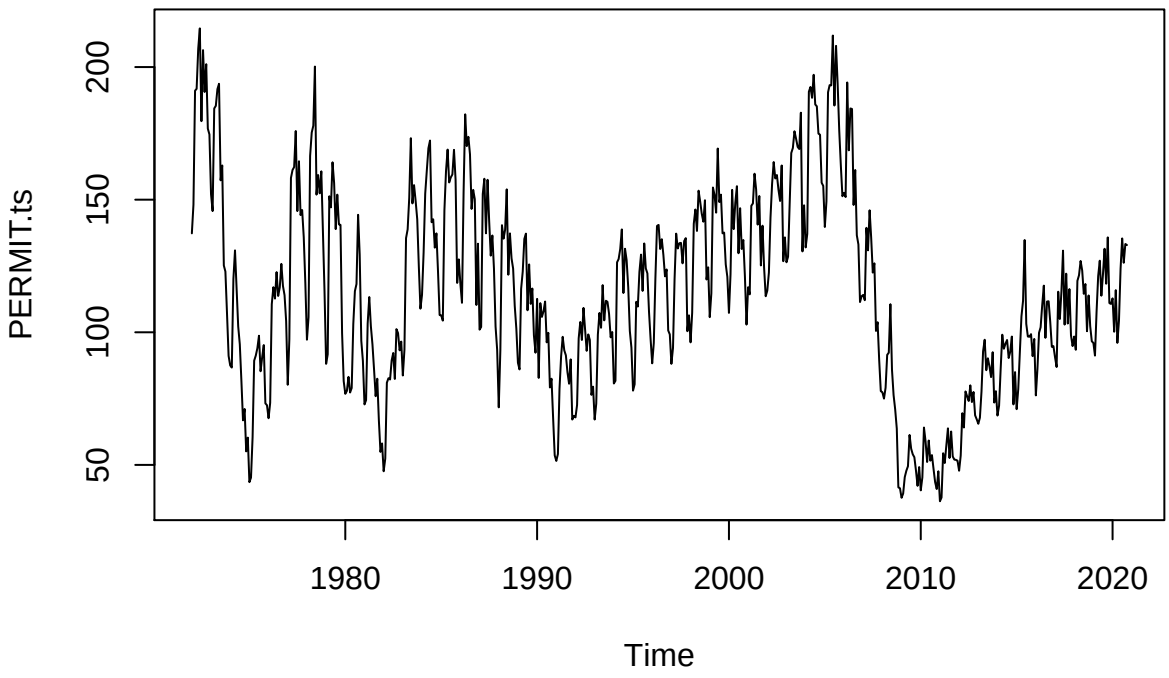
```
# Loading a library to forecast
library(forecast)
```

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

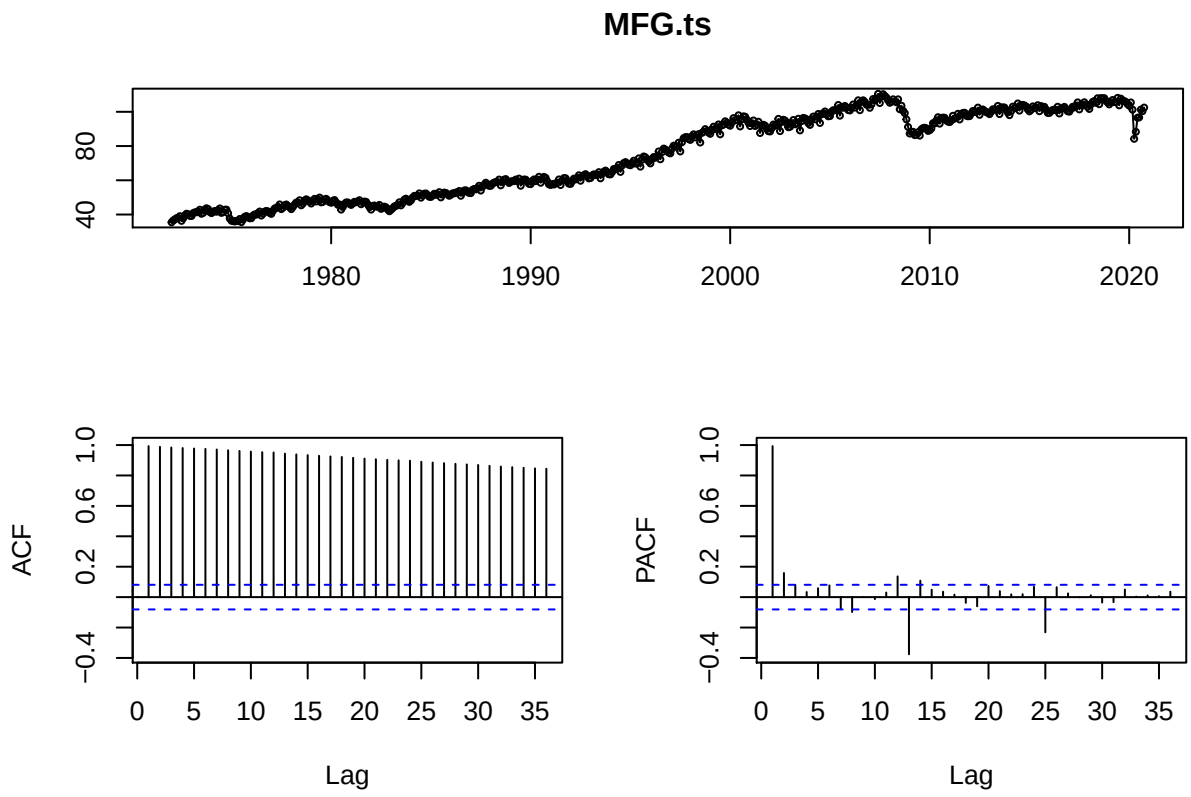
```
# Creating time series objects for use in forecasting
MFG.temp <- ts(MFG.csv, start = 1972, frequency = 12)
PERMIT.temp <- ts(PERMIT.csv, start = 1972, frequency = 12)

MFG.ts <- ts(MFG.temp[,2], start = 1972, frequency = 12)
PERMIT.ts <- ts(PERMIT.temp[,2], start = 1972, frequency = 12)
# Plotting the time series
plot(MFG.ts)
```



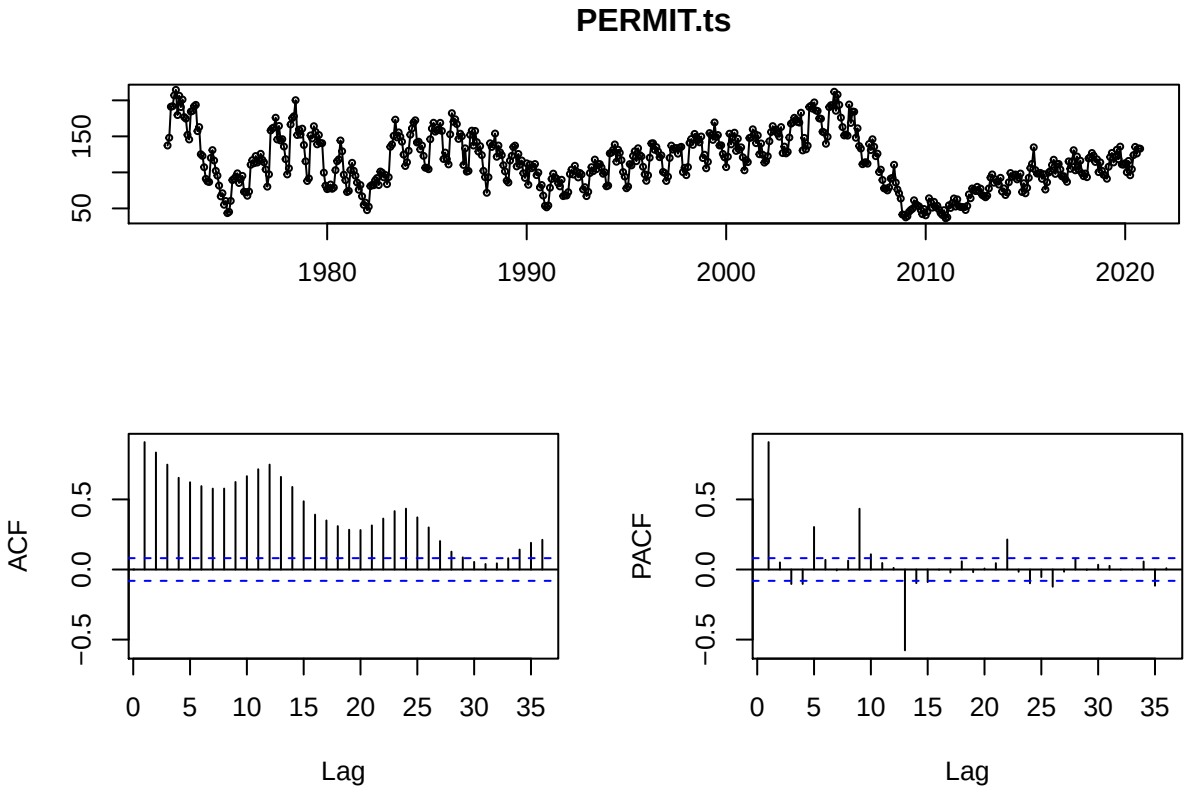

```
plot(PERMIT.ts)
```




```
# Plotting the series, ACFs, and PACFs
tsdisplay(MFG.ts)
```




```
tsdisplay(PERMIT.ts)
```



On the Manufacturing Index PACF, we see three significant spikes, which might lead us to use an AR(3) model. The ACF of that series slowly trends towards zero. On the Building Permit ACF, we see a cyclical trending towards zero, and two large spikes on the PACF, although there are a few more, so we might consider anything from an AR(2) to an AR(5) process for that series.

2. Fitting ARIMA

```
# Fitting an ARIMA model automatically
MFG.arima <- auto.arima(MFG.ts)
PERMIT.arima <- auto.arima(PERMIT.ts)
# Showing error results from our ARIMA models
cat("Manufacturing:","\n")
```

```
## Manufacturing:
```

```
accuracy(MFG.arima)
```

```
##                       ME     RMSE       MAE          MPE      MAPE      MASE
## Training set -0.001798144 1.074283 0.6231827 -0.006496653 0.8616581 0.2000572
##                     ACF1
## Training set 0.000450461
```

```r
cat("Building Permits:","\n")
```

```
## Building Permits:
```

```r
accuracy(PERMIT.arima)
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.2701642 9.204863 7.058819 -0.7151516 6.907948 0.3967298
##                      ACF1
## Training set -0.005572377
```

Using the auto.arima algorithm for our time series returned the above models, which both perform quite well. The Building Permits model has a quite substantial error component, a theme which we will see repeated later on.

3. Fitting Trend, Season, Cycle

```r
# Creating a time sequence to predict a model on
t<-seq(1972, 2021.83,length=length(MFG.ts))
# Fitting our own ARIMA based on ACF/PACFs
MFG.full_model <- Arima(MFG.ts,order=c(3,0,0),include.drift=TRUE, seasonal=list(order=c(1,0,1)))
# Fitting our own ARIMA based on ACF/PACFs
PERMIT.full_model <- Arima(PERMIT.ts,order=c(2,1,1),include.drift=TRUE ,seasonal=list(order=c(1,0,1)))
# Calling error reuslts from our own fitted models
cat("Manufacturing:","\n")
```

```
## Manufacturing:
```

```r
accuracy(MFG.full_model)
```

```
##                       ME     RMSE       MAE          MPE      MAPE    MASE
## Training set 0.004528983 1.083995 0.6373648 -0.001321791 0.8908599 0.20461
##                     ACF1
## Training set 0.002966073
```

```r
cat("Housing Permits:","\n")
```

```
## Housing Permits:
```

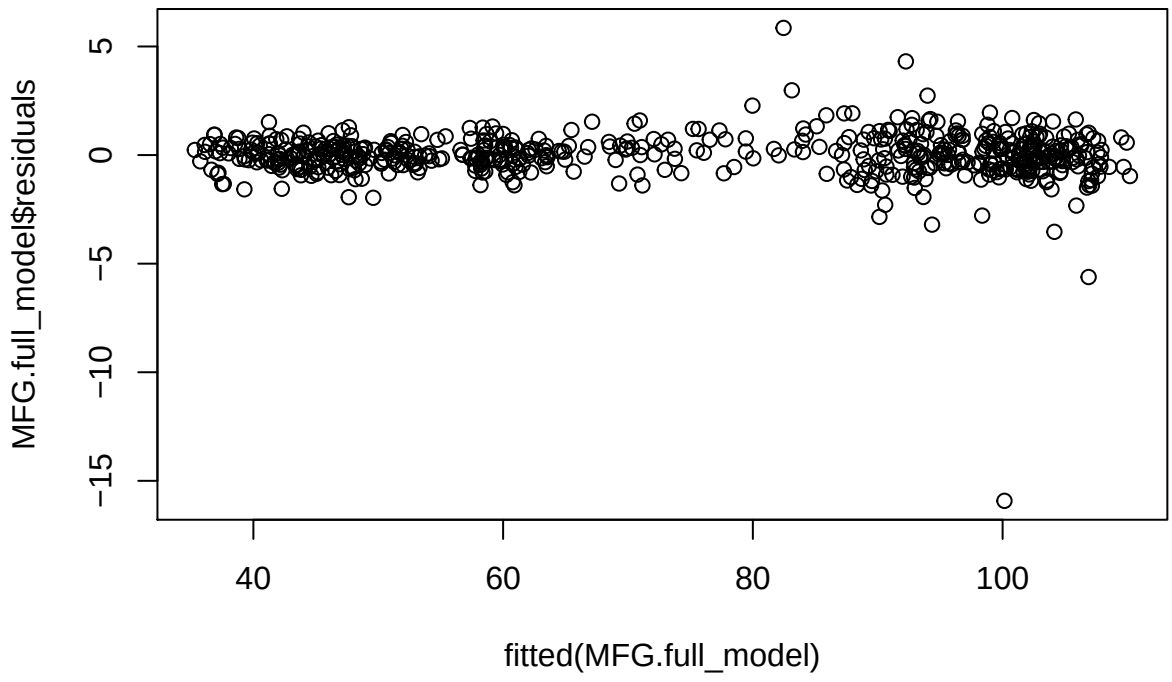
```r
accuracy(PERMIT.full_model)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.2473374 10.11925 7.851222 0.04765672 7.458855 0.4412656
##                     ACF1
## Training set 0.02531578
```

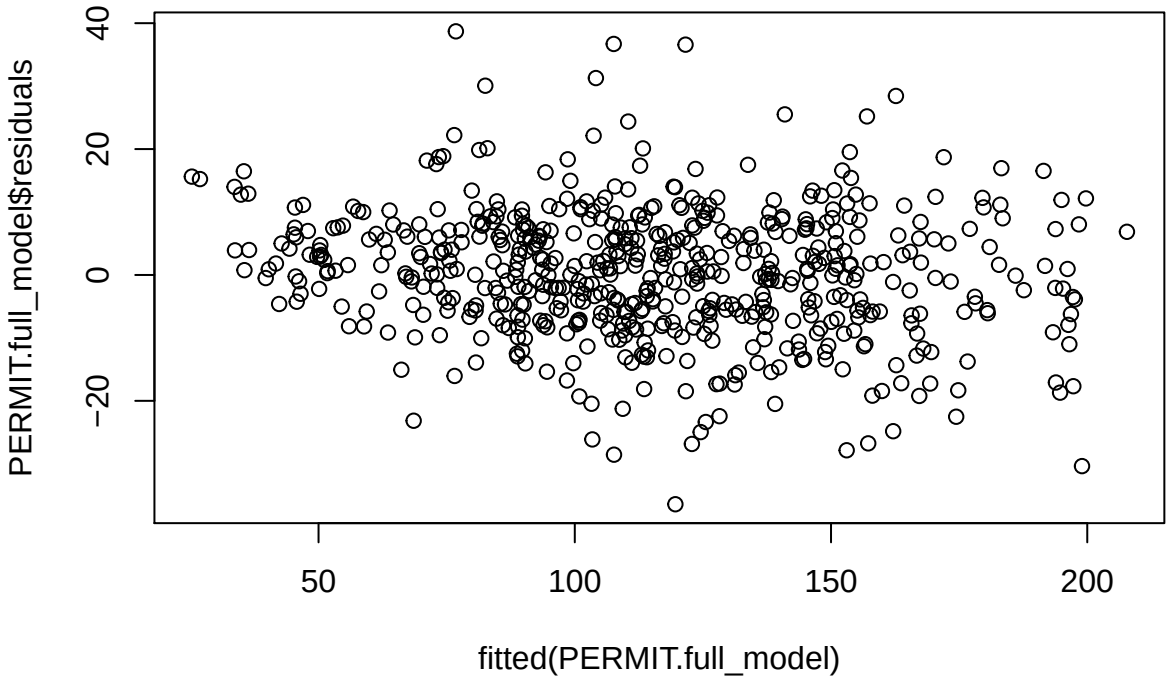
In fitting the season, cycle, and trend models, both series clearly had features of autoregressive (AR) and moving average (MA) processes, so we wanted to fit our own ARIMA model. For the Manufacturing Index model, we found that lower error were produced with an AR(3) process with a s-ARMA(1,1) modelling the seasonal component. Given the nature of the Building Permits series, it made sense to fit an ARMA(2,1) model with an s-ARMA(1,1) similar to the last series. For the Building Permit model, we found the best results were obtained with 1 integration, so we went ahead with an ARIMA(2,1,1) for the cyclical component of the permit model.

4. Residual Plots

```
# Plotting the residuals against the fitted values
plot(x=fitted(MFG.full_model), y=MFG.full_model$residuals)
```



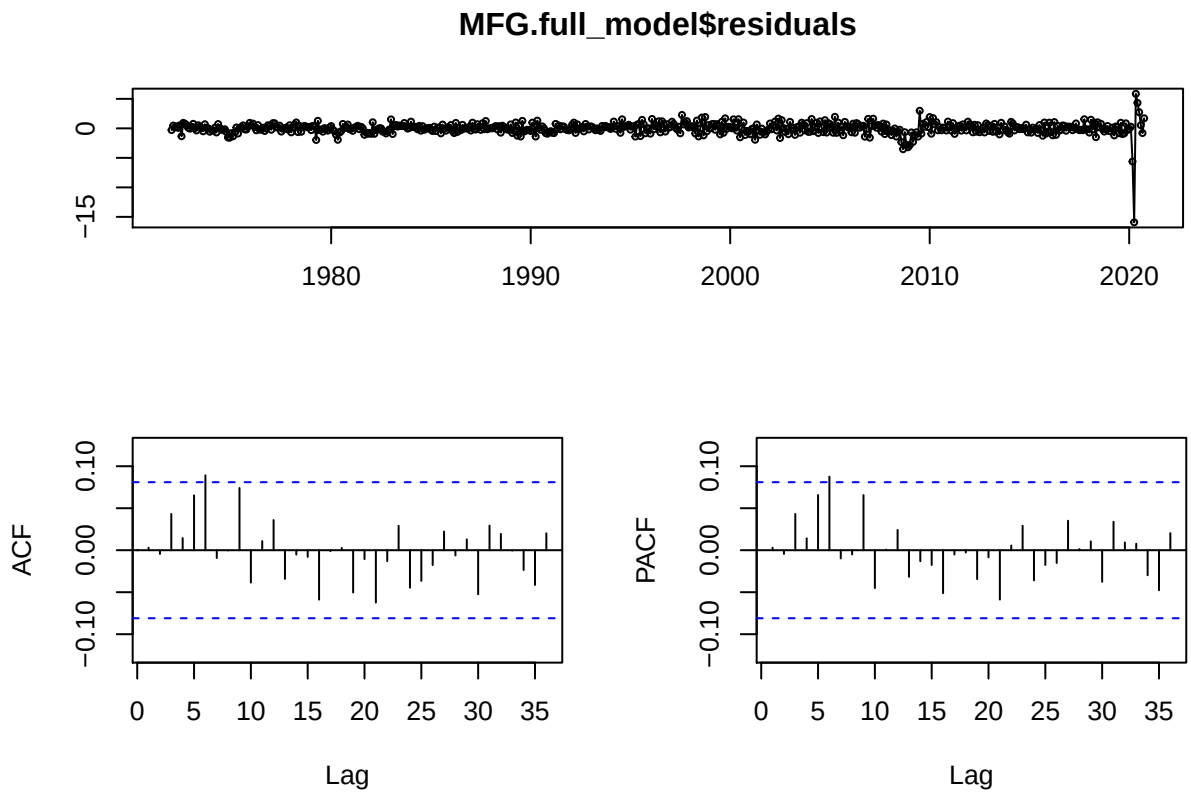

```
plot(x=fitted(PERMIT.full_model), y=PERMIT.full_model$residuals)
```



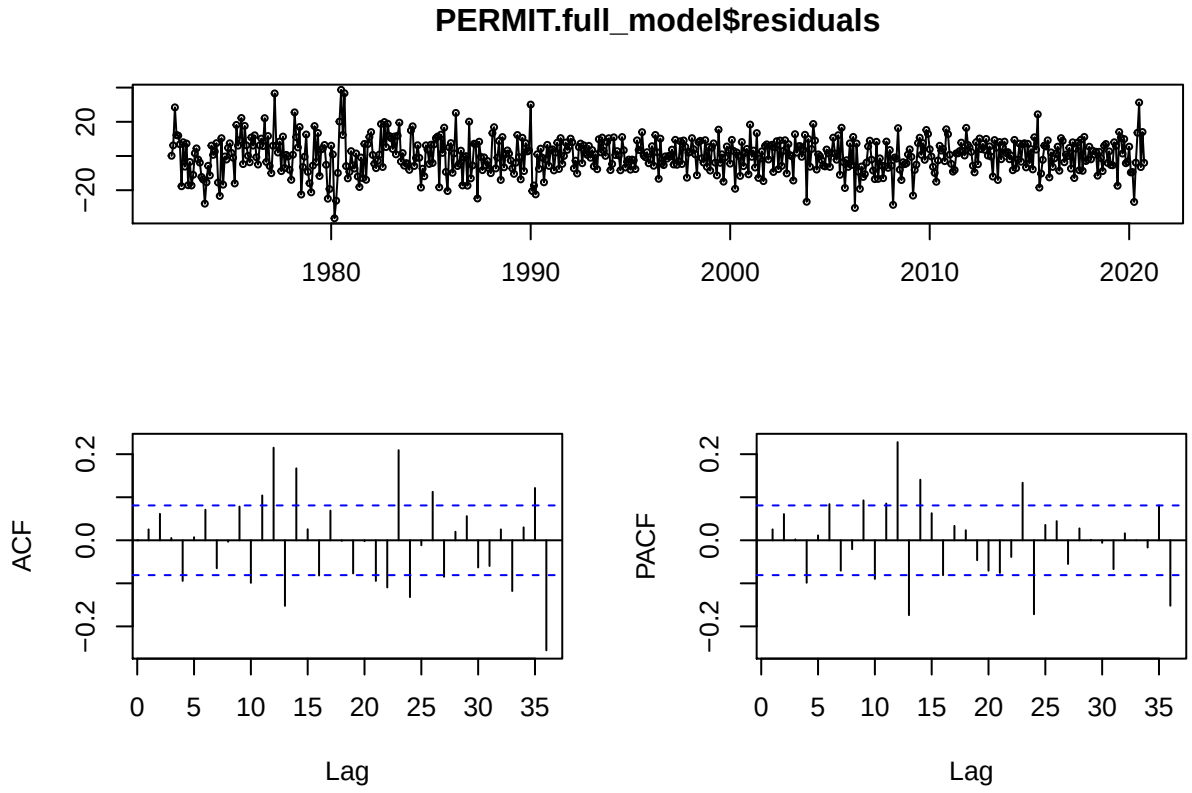
On the residual plots, we observe quite constant error variance across both models, with a couple obvious exceptions (more visible on the Manufacturing Index residual series), such as the 2008 financial crisis and the COVID-19 pandemic.

5. Residual ACF/PACF

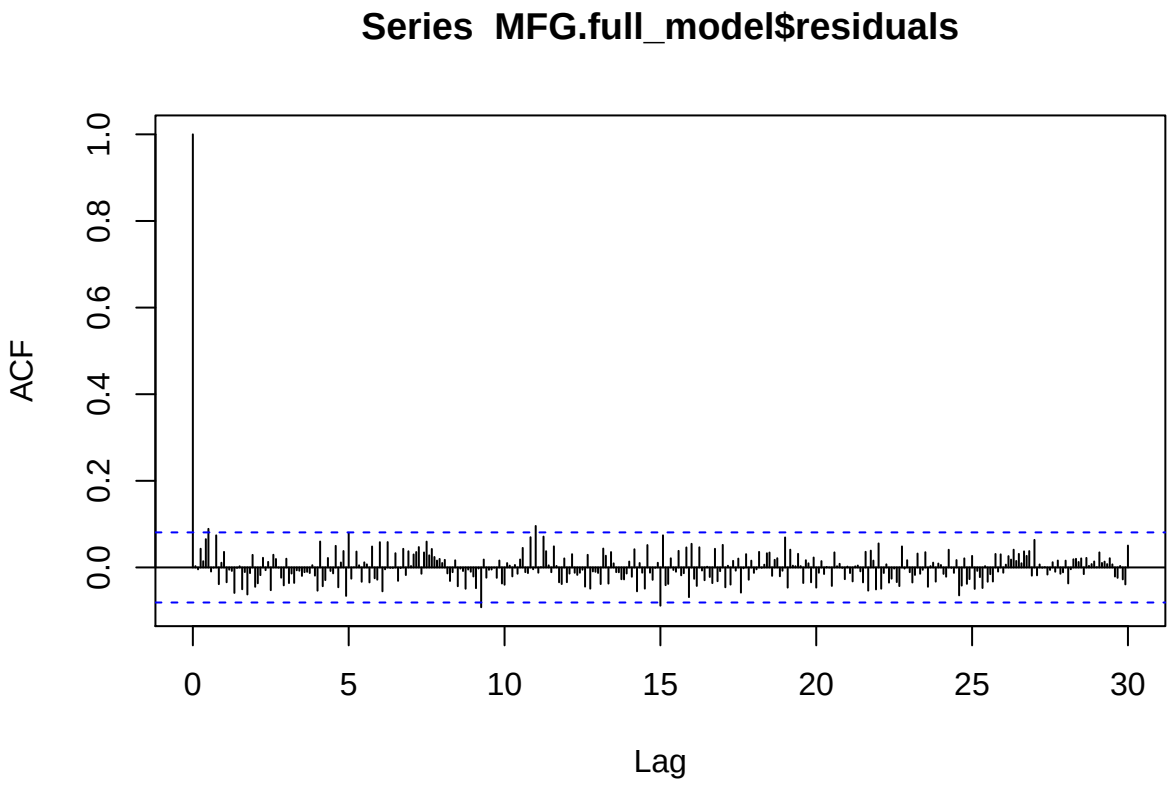
```
# Plotting the residuals as a time series, ACF/PACF
tsdisplay(MFG.full_model$residuals)
```



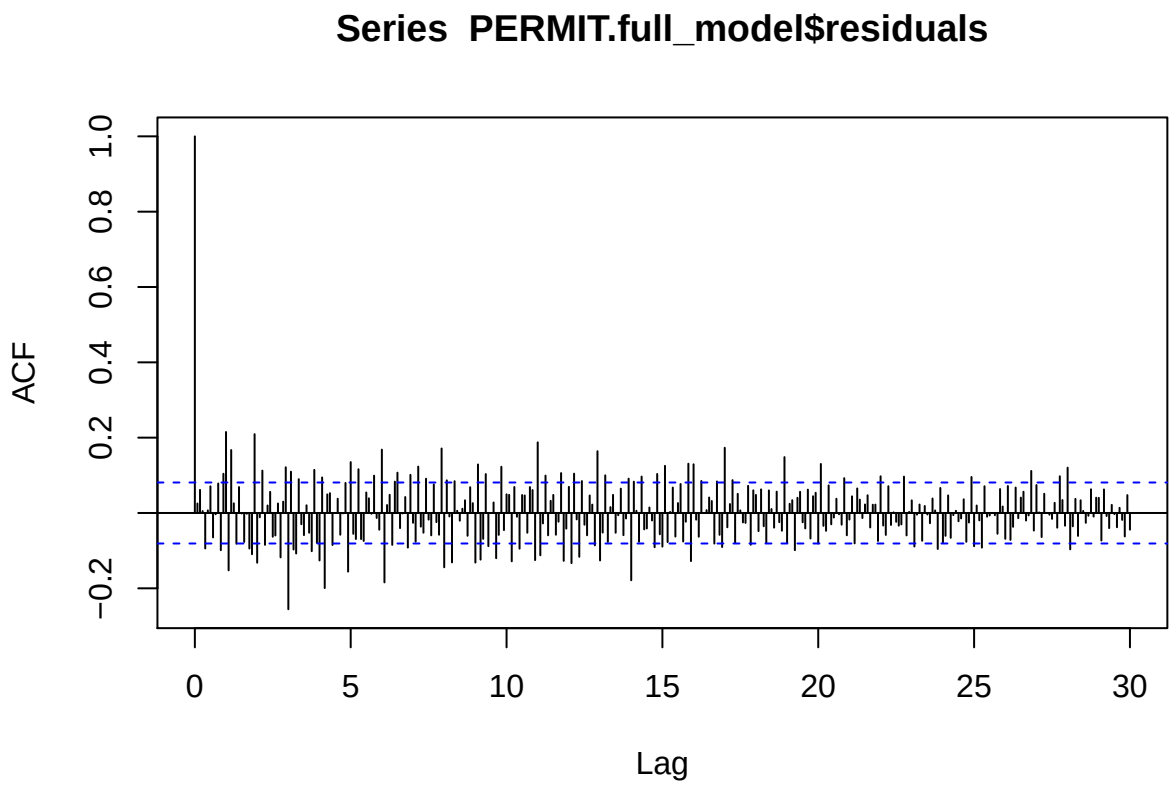

```
tsdisplay(PERMIT.full_model$residuals)
```



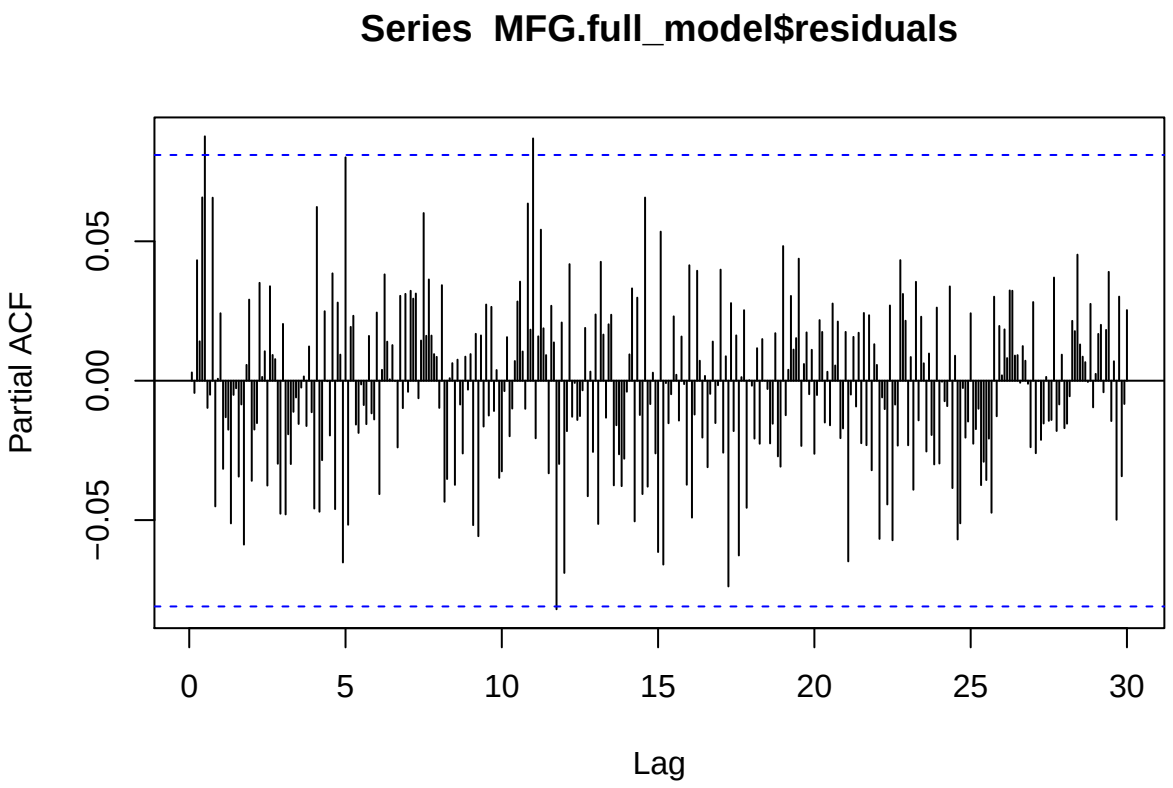

```
# Extending the lag horzion for ACF
acf(MFG.full_model$residuals, lag=360)
```



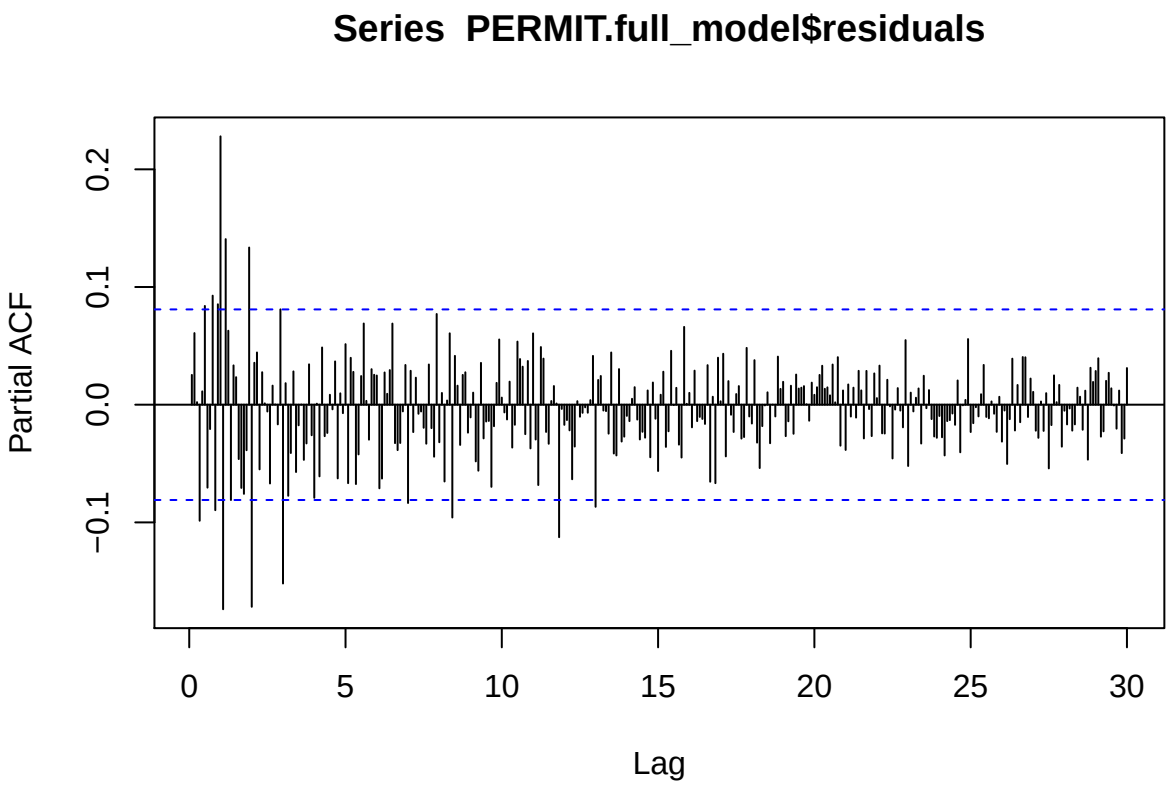

```
acf(PERMIT.full_model$residuals, lag=360)
```




```
# Extending the lag horzion for PACF
pacf(MFG.full_model$residuals, lag=360)
```




```
pacf(PERMIT.full_model$residuals, lag=360)
```



For both models, we see spikes in the ACF of the residuals at lag 1, before all the rest of the lags fit nicely into the bands. THe PACF for both models fits mostly between the bands, however the Building Permit model PACF has a few positive and negative spikes outside the bands. The is most certainly created by the residual time series of the building permit model having much more variation than the other model.

6. CUSUM

```
#Loading a library for CUSUM/resresid
library(strucchange)
```

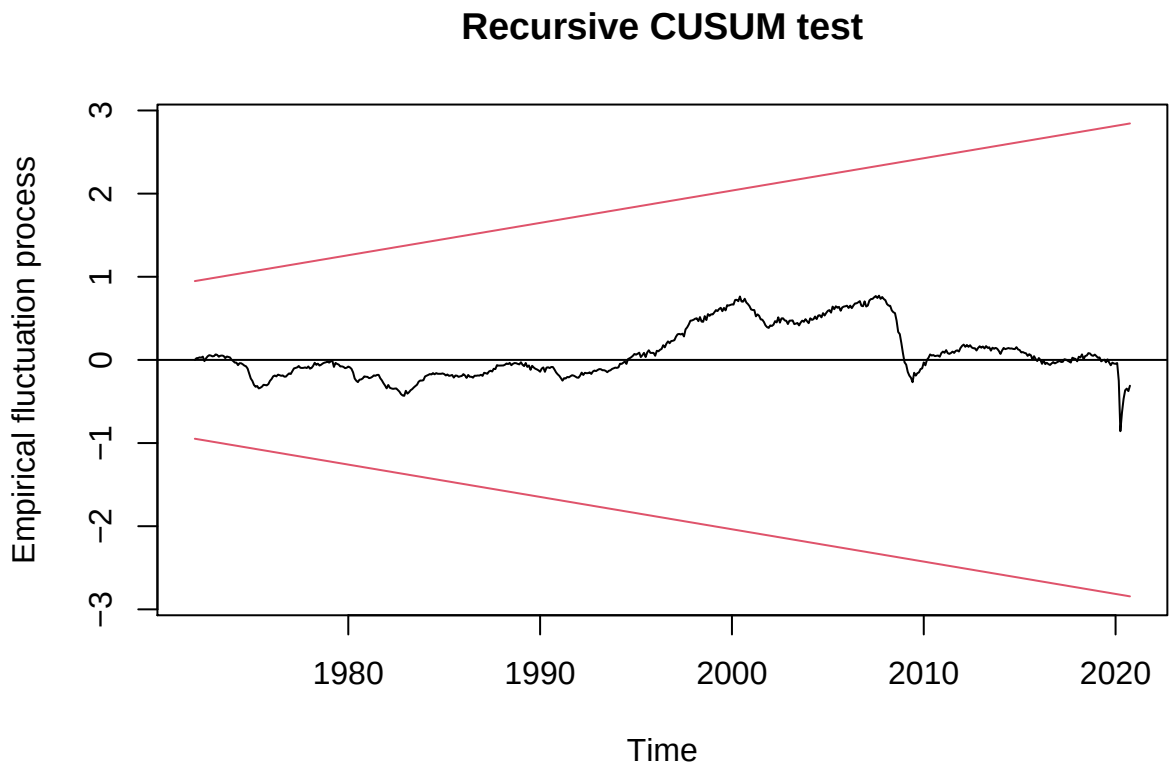
```
## Loading required package: zoo
```

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

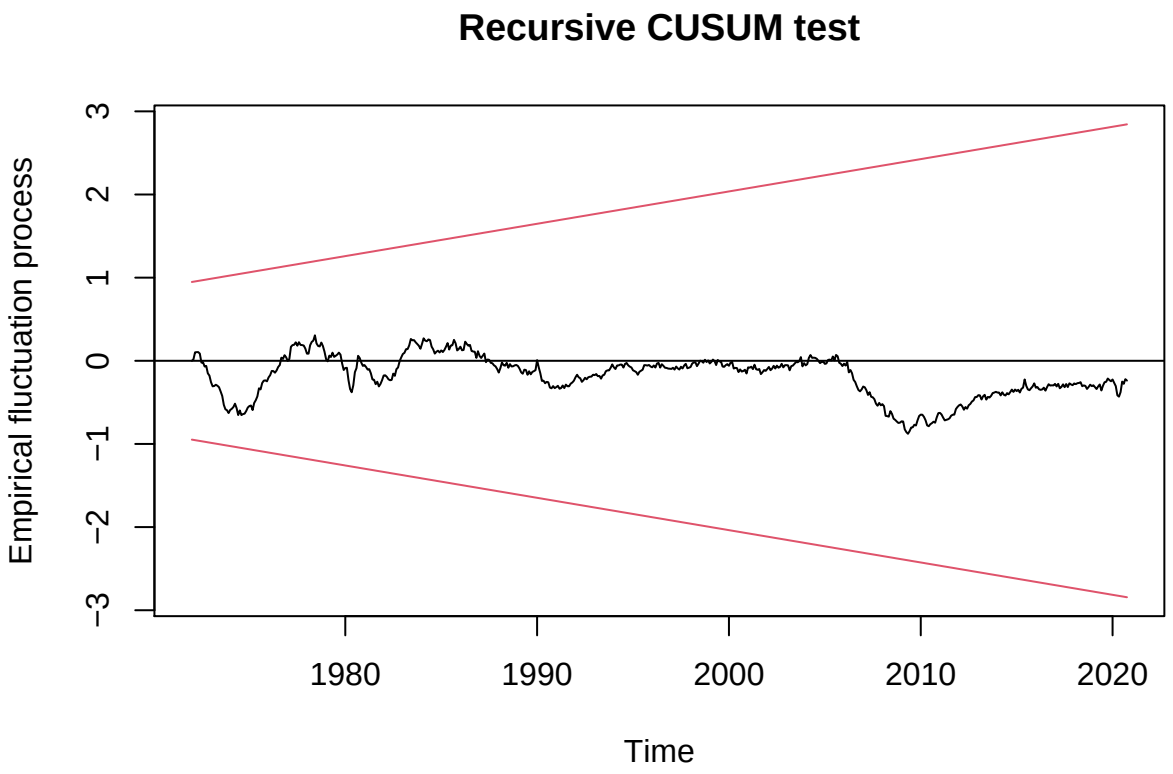
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

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

```r
# Plotting Cumulative Sums
plot(efp(MFG.full_model$res~1, type = "Rec-CUSUM"))
```



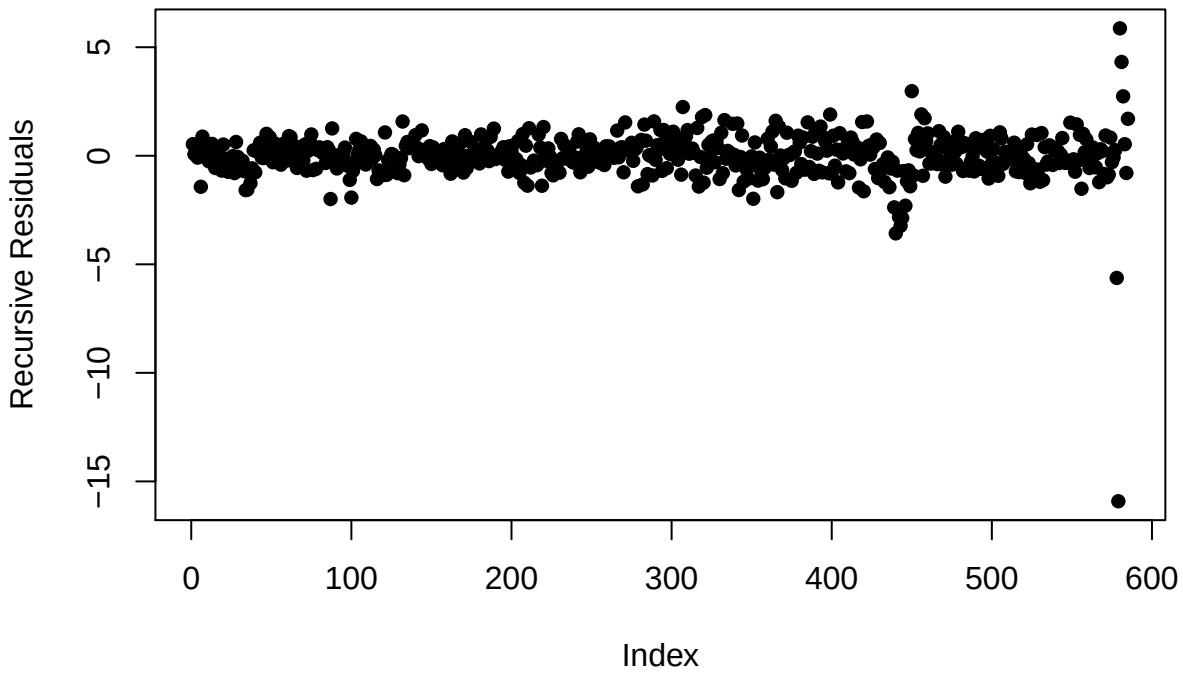

```r
plot(efp(PERMIT.full_model$res~1, type = "Rec-CUSUM"))
```



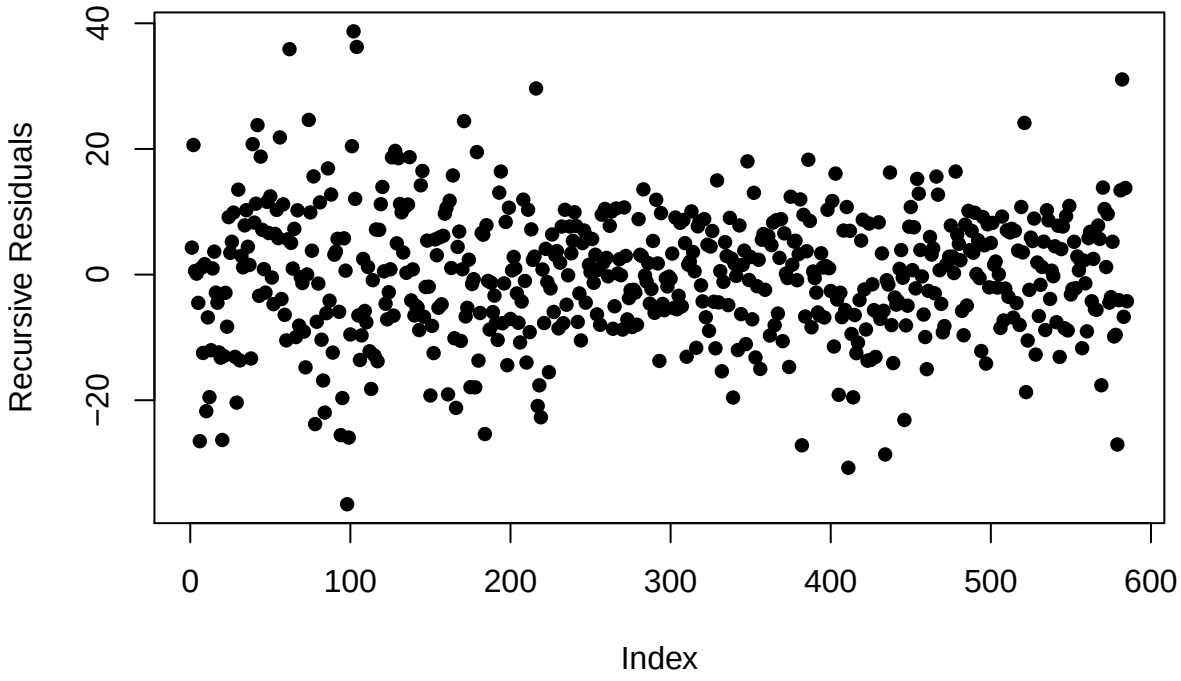
For both our models, the CUSUM plots show that the perform well across their time horizons.

7. Recursive Residuals

```
# Calculating and plotting recursive residuals
recres.MFG=recresid(MFG.full_model$res~1)
recres.PERMIT=recresid((PERMIT.full_model$res~1))
plot(recres.MFG, pch=16,ylab="Recursive Residuals")
```




```
plot(recres.PERMIT, pch=16,ylab="Recursive Residuals")
```



Our recursive residual plot shows that our Manufacturing Index model performs well, up until the COVID-19 pandemic where it loses essentially all its explanatory power. The plot for the Building Permits model shows residuals that are much more varied; however the behavior of the residuals is quite constant over time unlike the other series.

8. Diagnostic Statistics

```
# Calling diagnostics for our models, and AIC/BIC
summary(MFG.full_model)
```

```
## Series: MFG.ts
## ARIMA(3,0,0)(1,0,1)[12] with drift
##
## Coefficients:
##          ar1      ar2     ar3    sar1     sma1  intercept   drift
##       1.1242  -0.1843  0.0280  0.9810  -0.6208    40.4472  0.1082
## s.e.  0.0416   0.0620  0.0419  0.0073   0.0547    18.9444  0.0359
##
## sigma^2 estimated as 1.189:  log likelihood=-892.04
## AIC=1800.07   AICc=1800.32   BIC=1835.06
##
## Training set error measures:
##                       ME     RMSE       MAE          MPE      MAPE    MASE
```

```
## Training set 0.004528983 1.083995 0.6373648 -0.001321791 0.8908599 0.20461
##                      ACF1
## Training set 0.002966073
```

```
summary(PERMIT.full_model)
```

```
## Series: PERMIT.ts
## ARIMA(2,1,1)(1,0,1)[12] with drift
##
## Coefficients:
##           ar1      ar2     ma1    sar1     sma1    drift
##       -1.0968  -0.4056  0.7405  0.9955  -0.8775  -0.1194
## s.e.   0.0681   0.0387  0.0657  0.0025   0.0250   2.5412
##
## sigma^2 estimated as 103.6:  log likelihood=-2196.98
## AIC=4407.97   AICc=4408.16   BIC=4438.57
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.2473374 10.11925 7.851222 0.04765672 7.458855 0.4412656
##                     ACF1
## Training set 0.02531578
```

```
AIC(MFG.arima,PERMIT.arima,MFG.full_model,PERMIT.full_model)
```

```
## Warning in AIC.default(MFG.arima, PERMIT.arima, MFG.full_model,
## PERMIT.full_model): models are not all fitted to the same number of observations
```

```
##                   df      AIC
## MFG.arima          6 1742.975
## PERMIT.arima      11 4233.648
## MFG.full_model     8 1800.074
## PERMIT.full_model  7 4407.968
```

```
BIC(MFG.arima,PERMIT.arima,MFG.full_model,PERMIT.full_model)
```

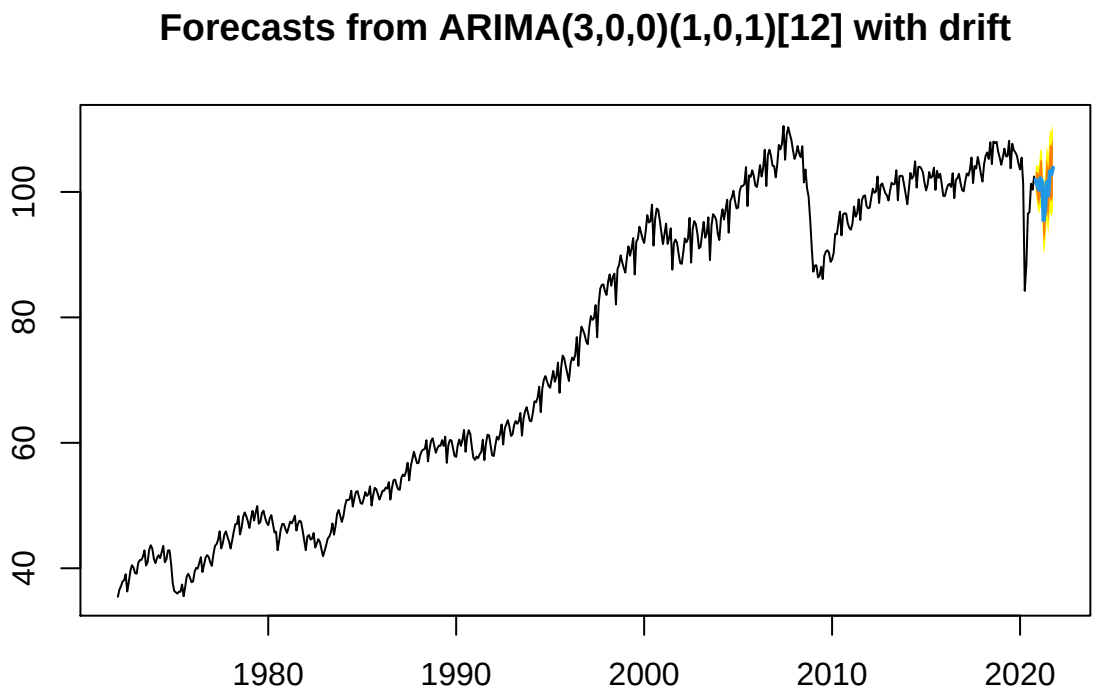
```
## Warning in BIC.default(MFG.arima, PERMIT.arima, MFG.full_model,
## PERMIT.full_model): models are not all fitted to the same number of observations
```

```
##                   df      BIC
## MFG.arima          6 1769.090
## PERMIT.arima      11 4281.527
## MFG.full_model     8 1835.060
## PERMIT.full_model  7 4438.569
```

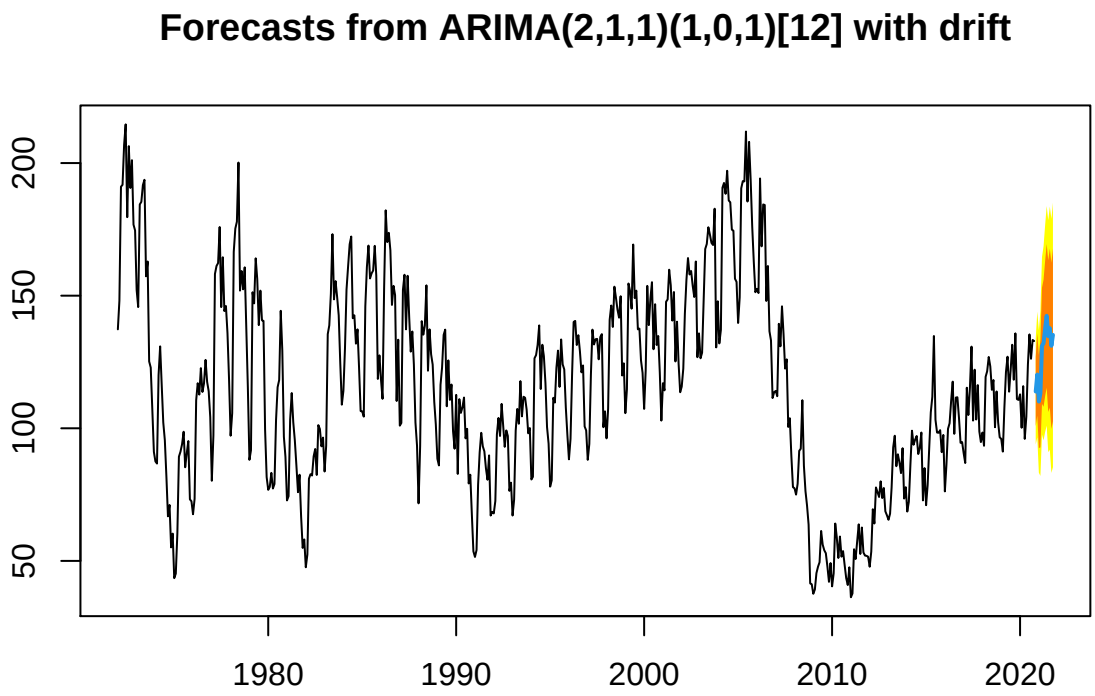
The season, cycle, trend model for Manufacturing Index fits quite well, with a MAPE of less than 1%. The Building Permits model did less well; about 7% off on average. The auto.arima models fitted earlier on are slightly prefered by AIC and BIC, for both models. The Mean Percentage Error (MPE) looks very good for the Building Permits model, at only 0.04%, but the MAPE tells us that MPE is only low because the upwards and downwards errors are quite similar. On average, the model is a lot more than 0.04% off.

9. 12-Step Forecast

```
# Forecasting 12 steps with each model, plotting it
plot(forecast(MFG.full_model,h=12),shadecols="oldstyle")
```




```
plot(forecast(PERMIT.full_model,h=12),shadecols="oldstyle")
```



10. Fitting VAR Model

```
# Combining both timme series into a data frame for VAR
y <- cbind(MFG.ts,PERMIT.ts)
y_tot <- data.frame(y)
# Laoding the library for VAR
library(vars)
```

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

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

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

```
# Fitting a VAR model
y_model=VAR(y_tot,p=4)
# Calling VAR summary statistics
summary(y_model)
```

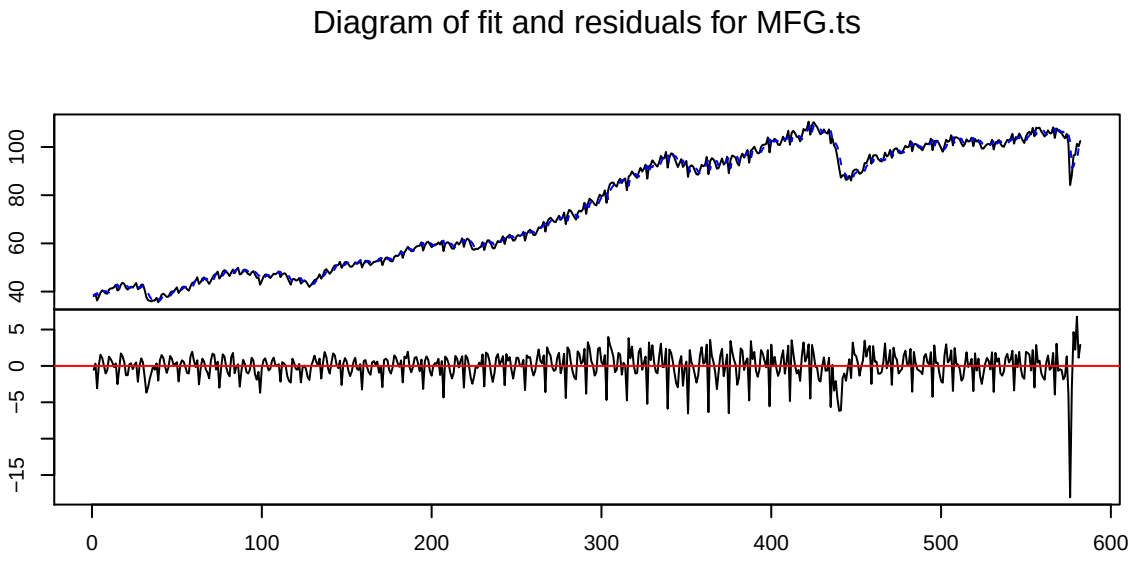
```
##
## VAR Estimation Results:
```

```
## ========================
## Endogenous variables: MFG.ts, PERMIT.ts
## Deterministic variables: const
## Sample size: 582
## Log Likelihood: -3553.463
## Roots of the characteristic polynomial:
## 0.9979 0.8294 0.6626 0.5828 0.5828 0.4439 0.4439 0.08743
## Call:
## VAR(y = y_tot, p = 4)
##
##
## Estimation results for equation MFG.ts:
## =======================================
## MFG.ts = MFG.ts.l1 + PERMIT.ts.l1 + MFG.ts.l2 + PERMIT.ts.l2 + MFG.ts.l3 + PERMIT.ts.l3 + MFG.ts.l4 +
##
##               Estimate Std. Error t value Pr(>|t|)
## MFG.ts.l1     0.643383   0.045171  14.243   <2e-16 ***
## PERMIT.ts.l1  0.003331   0.005819   0.572   0.5673
## MFG.ts.l2     0.083121   0.052418   1.586   0.1133
## PERMIT.ts.l2  0.014340   0.007435   1.929   0.0543 .
## MFG.ts.l3     0.160638   0.052459   3.062   0.0023 **
## PERMIT.ts.l3 -0.005207   0.007459  -0.698   0.4854
## MFG.ts.l4     0.109550   0.045213   2.423   0.0157 *
## PERMIT.ts.l4 -0.010021   0.005626  -1.781   0.0754 .
## const         0.158158   0.398082   0.397   0.6913
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1.977 on 573 degrees of freedom
## Multiple R-Squared: 0.9934,  Adjusted R-squared: 0.9933
## F-statistic: 1.083e+04 on 8 and 573 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation PERMIT.ts:
## ==========================================
## PERMIT.ts = MFG.ts.l1 + PERMIT.ts.l1 + MFG.ts.l2 + PERMIT.ts.l2 + MFG.ts.l3 + PERMIT.ts.l3 + MFG.ts.l
##
##               Estimate Std. Error t value Pr(>|t|)
## MFG.ts.l1    -0.008946   0.332559  -0.027 0.978549
## PERMIT.ts.l1  0.856717   0.042838  19.999  < 2e-16 ***
## MFG.ts.l2    -0.581572   0.385914  -1.507 0.132362
## PERMIT.ts.l2  0.194384   0.054742   3.551 0.000415 ***
## MFG.ts.l3    -2.048530   0.386221  -5.304 1.62e-07 ***
## PERMIT.ts.l3  0.065333   0.054919   1.190 0.234687
## MFG.ts.l4     2.619508   0.332871   7.869 1.79e-14 ***
## PERMIT.ts.l4 -0.210342   0.041423  -5.078 5.17e-07 ***
## const        12.468039   2.930797   4.254 2.45e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 14.56 on 573 degrees of freedom
## Multiple R-Squared: 0.8486,  Adjusted R-squared: 0.8464
```

```
## F-statistic: 401.3 on 8 and 573 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##           MFG.ts PERMIT.ts
## MFG.ts     3.909     10.84
## PERMIT.ts 10.837    211.90
##
## Correlation matrix of residuals:
##           MFG.ts PERMIT.ts
## MFG.ts    1.0000    0.3765
## PERMIT.ts 0.3765    1.0000
```

```
#Plotting VAR model
plot(y_model)
```

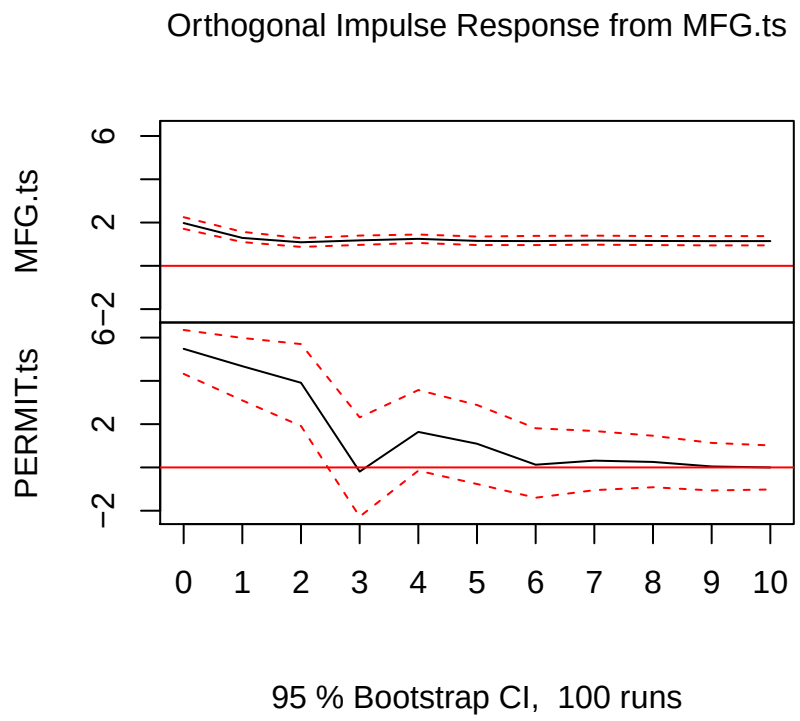
```
## Error in plot.new(): figure margins too large
```

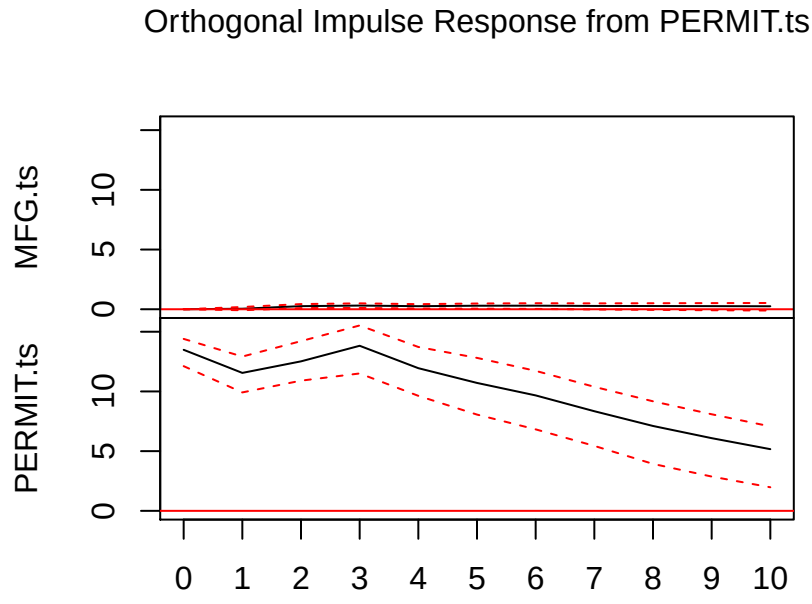


With our VAR model fitted, we now see very little trend amongst the residuals. Obviously, the 2008 financial crisis and the COVID-19 pandemic are still clearly visible, so perhaps further treatments could be made to the model to better account for those crises. However, the behavior of the residuals for the rest of the time series is precisely what we would desire.

11. Impulse Response Functions

```
# Calculating and Plotting IRF in one step
plot(irf(y_model))
```





Our impulse response function plots show that a change in the Manufacturing Index is correlated with significant change in Building Permits for a couple months after the shift in manufacturing.

12. Granger Causality Test

```
# Performing Granger test for causality
grangertest(MFG.ts ~ PERMIT.ts, order = 4)
```

```
## Granger causality test
##
## Model 1: MFG.ts ~ Lags(MFG.ts, 1:4) + Lags(PERMIT.ts, 1:4)
## Model 2: MFG.ts ~ Lags(MFG.ts, 1:4)
##   Res.Df Df      F  Pr(>F)
## 1    573
## 2    577 -4 3.0574 0.01647 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(PERMIT.ts ~ MFG.ts, order = 4)
```

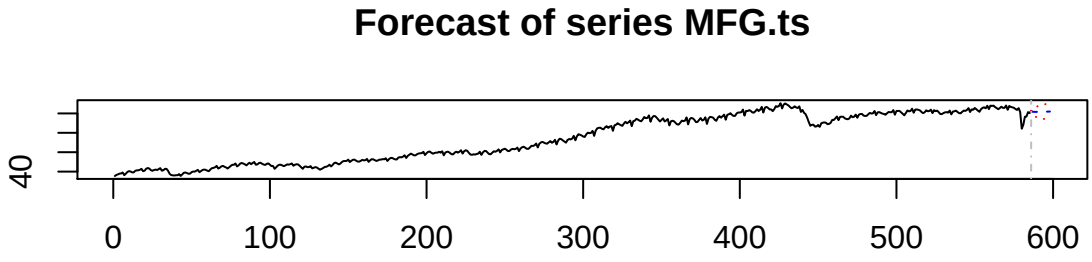
```
## Granger causality test
##
```

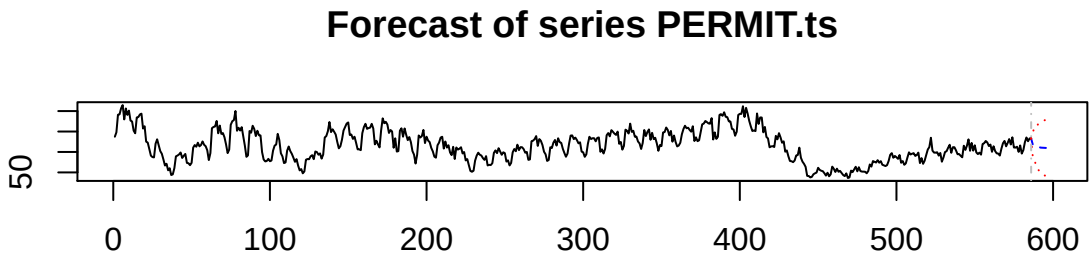
```
## Model 1: PERMIT.ts ~ Lags(PERMIT.ts, 1:4) + Lags(MFG.ts, 1:4)
## Model 2: PERMIT.ts ~ Lags(PERMIT.ts, 1:4)
##   Res.Df Df      F    Pr(>F)
## 1    573
## 2    577 -4 16.841 4.609e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Our Granger Causality testing determines that Manufacturing Index can predict Building Permits with strong significance. We do not fail to reject the reverse hypothesis, however, so the true relationship might not be as clear.

13. VAR 12-Step Forecast

```
# Creating predictions for VAR model
var.predict = predict(object=y_model, n.ahead=12)
# Plotting our created predictions
plot(var.predict)
```






```
# Forecasting using our ARIMA models to compare
forecast(MFG.full_model,h=12)
```

```
##          Point Forecast     Lo 80     Hi 80    Lo 95    Hi 95
```

```
## Nov 2020      101.97253 100.57496 103.37009 99.83513 104.1099
## Dec 2020      101.00878  98.90603 103.11153 97.79291 104.2247
## Jan 2021      100.34800  97.76010 102.93591 96.39014 104.3059
## Feb 2021      102.19090  99.22672 105.15509 97.65757 106.7242
## Mar 2021      101.70027  98.42602 104.97451 96.69275 106.7078
## Apr 2021       95.46182  91.92426  98.99938 90.05159 100.8720
## May 2021       97.06340  93.29809 100.82872 91.30485 102.8220
## Jun 2021      101.58581  97.62097 105.55064 95.52212 107.6495
## Jul 2021       99.43825  95.29700 103.57951 93.10475 105.7718
## Aug 2021      103.30436  99.00598 107.60273 96.73056 109.8782
## Sep 2021      102.80919  98.37009 107.24829 96.02017 109.5982
## Oct 2021      103.87536  99.30964 108.44107 96.89270 110.8580
```

```
forecast(PERMIT.full_model,h=12)
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Nov 2020       113.8633 100.81683 126.9098  93.91043 133.8162
## Dec 2020       120.2717 104.75652 135.7870  96.54326 144.0002
## Jan 2021       110.0872  92.53595 127.6384  83.24490 136.9295
## Feb 2021       113.1594  92.80831 133.5104  82.03512 144.2836
## Mar 2021       131.0985 109.20209 152.9948  97.61085 164.5861
## Apr 2021       132.0660 108.13092 156.0011  95.46046 168.6715
## May 2021       137.5750 112.06397 163.0860  98.55926 176.5907
## Jun 2021       142.4019 115.31504 169.4887 100.97614 183.8276
## Jul 2021       134.7345 106.14691 163.3221  91.01355 178.4555
## Aug 2021       137.7533 107.78596 167.7206  91.92222 183.5844
## Sep 2021       131.0872  99.75684 162.4176  83.17153 179.0029
## Oct 2021       135.2049 102.59770 167.8122  85.33647 185.0734
```

```
# Printing out the VAR forecast for comparison
var.predict
```

```
## $MFG.ts
##           fcst    lower    upper       CI
##  [1,] 101.6407 97.76542 105.5160 3.875286
##  [2,] 101.6321 97.00370 106.2604 4.628370
##  [3,] 101.5269 96.40474 106.6490 5.122151
##  [4,] 101.6471 95.99699 107.2971 5.650074
##  [5,] 101.6333 95.45599 107.8106 6.177322
##  [6,] 101.6275 95.02487 108.2302 6.602678
##  [7,] 101.6996 94.70302 108.6963 6.996619
##  [8,] 101.7378 94.35520 109.1205 7.382653
##  [9,] 101.7762 94.03906 109.5134 7.737170
## [10,] 101.8286 93.75847 109.8987 8.070111
## [11,] 101.8791 93.48922 110.2690 8.389871
## [12,] 101.9277 93.23286 110.6225 8.694832
##
## $PERMIT.ts
##           fcst    lower    upper       CI
##  [1,] 118.5522 90.02120 147.0832 28.53099
##  [2,] 121.3207 83.75956 158.8819 37.56115
##  [3,] 113.0504 67.53488 158.5660 45.51556
##  [4,] 112.8944 59.92394 165.8649 52.97047
```

```
##  [5,] 112.2493 54.24398 170.2545 58.00528
##  [6,] 110.6667 48.94097 172.3924 61.72569
##  [7,] 110.4011 45.83174 174.9706 64.56941
##  [8,] 110.2024 43.59107 176.8137 66.61132
##  [9,] 109.9463 41.89173 178.0009 68.05460
## [10,] 109.8185 40.72588 178.9111 69.09261
## [11,] 109.7899 39.95965 179.6201 69.83022
## [12,] 109.7561 39.41074 180.1015 70.34540
```

For the Manufacturing Index forecast, both models reported that the index would hover around 101 for the next 12 months. The season, trend, cycle model varied more in its point forecast, but was slimmer on the forecast interval, while the VAR model had a less varied point forecast but a larger interval.

For the number of housing permits issued, the models gave quite different forecasts. The VAR model posits that permits are likely to decrease over the next year, while the season, trend, cycle model has it increasing. The likely reason for this disagreement is seen on the original time series plot. The Manufacturing Index had clearly seen a drop in level from the pandemic, but the housing permits series had not yet been affected by the pandemic. Since the VAR model uses information from the Manufacturing time series, it therefore produces what is probably a more realistic forecast. On the other hand, the original model of just housing permits is more optimistic as it does not consider information about the pandemic. The couple down months from the pandemic are likely just considered by the ARIMA model to be a bad season, rather than a once-in-a-lifetime global pandemic.

Overall, the ARIMA model sees a much quicker recovery, and even growth for housing permits over the next year. However, since we concluded from the Granger Causality testing that Building Permits were affected by the Manufacturing Index with a 2-4 month lag, we should probably expect a drop in Building Permits similar to what the VAR model forecasts, instead of taking the rather optimistic forecast of the ARIMA model.

14. Backtest ARIMA Model

Since we determined above that the manufacturing index may have some predictive power on housing permits, we will forecast and backtest the permit ARIMA model from here on out, to see how it compares to the VAR forecasts.

(a) 12-Step Recursive

```
# Loading a library for backtesting
library(MTS)
```

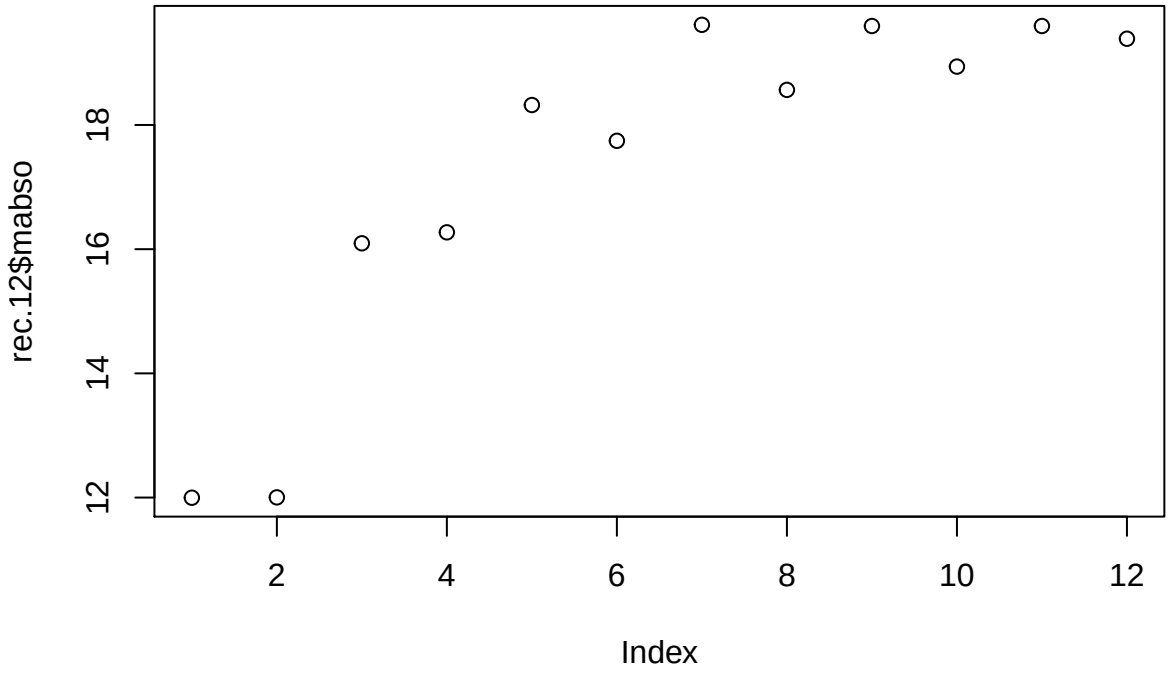
```
##
## Attaching package: 'MTS'
```

```
## The following object is masked from 'package:vars':
##
##     VAR
```

```
# Backtesting 12 steps ahead and plotting MAPE
rec.12 <- backtest(PERMIT.full_model, PERMIT.temp, 587, 12)
```

```
## [1] "RMSE of out-of-sample forecasts"
##  [1] 16.79539 17.30334 22.15651 22.42486 24.37289 23.93035 26.16855 25.17742
##  [9] 26.49045 25.33965 26.35021 26.16557
## [1] "Mean absolute error of out-of-sample forecasts"
##  [1] 11.99772 12.00300 16.09466 16.27148 18.32163 17.74509 19.61302 18.56518
##  [9] 19.59368 18.94005 19.59400 19.38897
```

```
plot(rec.12$mabso)
```

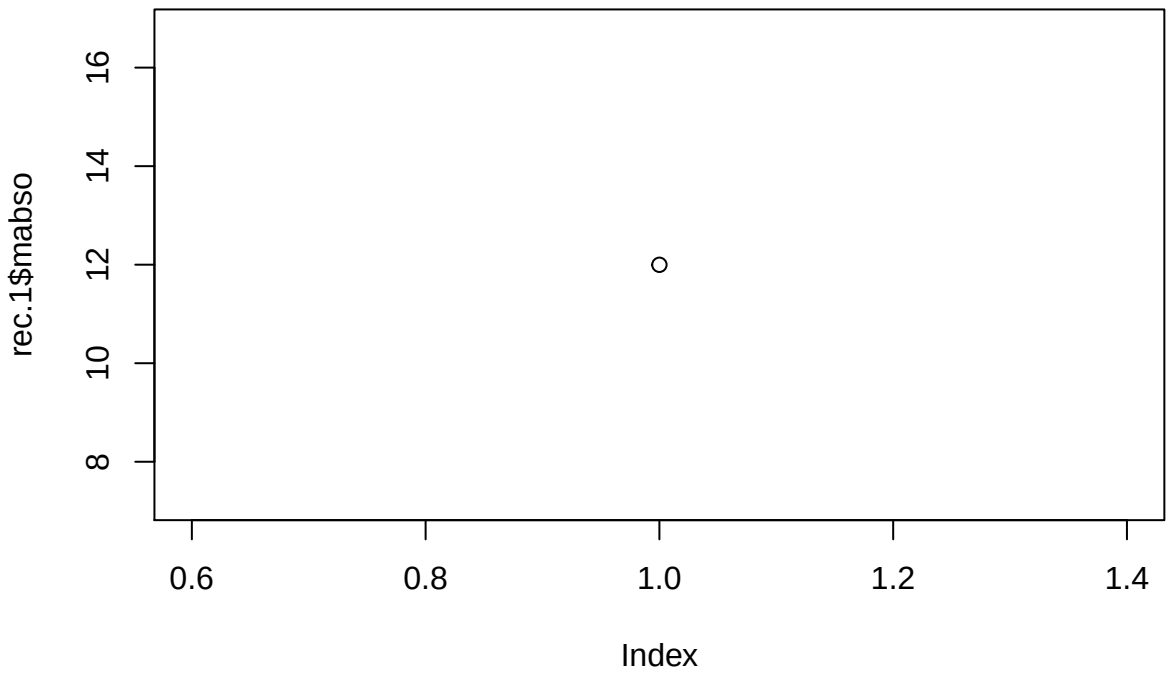


(b) 1-Step Recursive

```
# Backtesting 1 steps ahead and plotting MAPE
rec.1 <- backtest(PERMIT.full_model, PERMIT.temp, 587, 1)
```

```
## [1] "RMSE of out-of-sample forecasts"
## [1] 16.79539
## [1] "Mean absolute error of out-of-sample forecasts"
## [1] 11.99772
```

```
plot(rec.1$mabso)
```



(c) Long/Short Horizons

Based on the backtesting, the model performs best in a one or two-step forecast. As we increase the number of periods forecast, the MAPE increases significantly, as is clear on the 12-step plot of MABSO.

(d) Moving Window Backtest

```
# Creating window backtest function
# Code from TA
window_backtesting <- function(model, data, orig, h, xreg=NULL,fixed = NULL, inc.mean = TRUE,
                               reest = 1){
  if(!inherits(data,"ts"))stop("data must be a time series object")
  arma_order <- model$arma
  regor = arma_order[c(1, 6, 2)]
  seaor = list(order = arma_order[c(3, 7, 4)],  period = arma_order[5])
  T = length(data)
  if (orig > T)
    orig = T
  if (h < 1)
    h = 1
  rmse = numeric(h)
  mabso = numeric(h)
```

```
  nori = T - orig
  err = matrix(0, nori, h)
  fcst = matrix(0, nori, h)
  jlast = T - 1
  time_vec <- time(data)
  ireest <- reest
  for (n in orig:jlast) {
    jcnt = n - orig + 1
    x <- window(data, time_vec[jcnt], time_vec[n])
    if (is.null(xreg))
      pretor = NULL
    else pretor = xre[jcnt:n]
    if (ireest == reest) {
      mm = arima(x, order = regor, seasonal = seaor, xreg = pretor,
                 fixed = fixed, include.mean = inc.mean)
      ireest <- 0
    }
    else {
      ireest <- ireest + 1
    }
    if (is.null(xreg)) {
      nx = NULL
    }
    else {
      nx = xreg[(n + 1):(n + h)]
    }
    fore = predict(mm, h, newxreg = nx)
    kk = min(T, (n + h))
    nof = kk - n
    pred = fore$pred[1:nof]
    obsd = data[(n + 1):kk]
    err[jcnt, 1:nof] = obsd - pred
    fcst[jcnt, 1:nof] = pred
  }
  for (i in 1:h) {
    iend = nori - i + 1
    tmp = err[1:iend, i]
    mabso[i] = sum(abs(tmp))/iend
    rmse[i] = sqrt(sum(tmp^2)/iend)
  }
  print("RMSE of out-of-sample forecasts")
  print(rmse)
  print("Mean absolute error of out-of-sample forecasts")
  print(mabso)
  backtest <- list(origin = orig, error = err, forecasts = fcst,
                   rmse = rmse, mabso = mabso, reest = reest)
}
```

```
# Window backtesting 12 steps
window.12 <- window_backtesting(PERMIT.full_model, PERMIT.temp, 588, 12)
```

```
## Error in if (end > xtsp[2L] + ts.eps/xfreq && !extend) {: missing value where TRUE/FALSE needed
```

```r
# Window backtesting 1 step
window.1 <- window_backtesting(PERMIT.full_model, PERMIT.temp, 588, 1)
```

```
## Error in if (end > xtsp[2L] + ts.eps/xfreq && !extend) {: missing value where TRUE/FALSE needed
```

```r
# Plotting respective backtest errors
plot(window.12$mabso)
```

```
## Error in h(simpleError(msg, call)): error in evaluating the argument 'x' in selecting a method for fu
```

```r
plot(window.1$mabso)
```

```
## Error in h(simpleError(msg, call)): error in evaluating the argument 'x' in selecting a method for fu
```

(e) Recursive vs. Moving Window

Unfortunately we could not get the moving window code to not return a 'True/False needed' error, so we cannot make comparisons between the backtesting schemes now.

Conclusions / Future Work

Our models above do at least a reasonable job at providing good forecasts, however there are definitely areas for improvement. Generally, the errors on the Building Permits model were quite large, nearly to the point where it becomes hard to make any meaningful forecast. Finding a better model fit with more advanced algortihms and filters would benefit the forecasts by reducing the error variation. In terms of the Manufacturing Index model, it performed very well, although it had troubles adjusting to the the 2008 financial crisis and the COVID-19 pandemic. Further work could be done to create a model that considers these massive shocks better than our models do. If that were done, essentially all the error variance originally present in the model would be eliminated, resulting in a robust model good to use for forecasting.

References

Manufacturing Data:
https://fred.stlouisfed.org/series/IPGMFN
Building Permit Data:
https://fred.stlouisfed.org/series/PERMITNSA