ECON 403B Project 2

John Macke, Pujan Thakrar, Mark Vandre

February 7, 2019

I. Introduction (describe the data, provide some background on the topic, etc.)

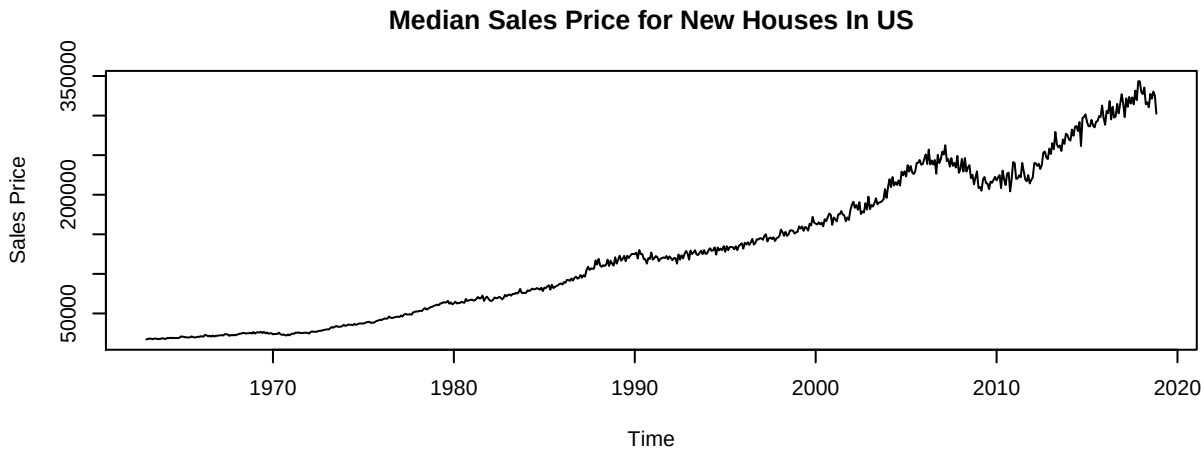
The first time series variable we choose to look at is the Median Sales Price of New Houses Sold in the United States. This data is taken from FRED and is at a monthly frequency. The data appears to have a clear increasing trend with seasonal or cyclical behavior. The second time series that we choose to look at is the US Unemployment Rate. The data is take from FRED and is at a monthly frequency. The unemployment rate represents the number of unemployed as a percentage of the labor force. Labor force data are restricted to people 16 years of age and older, who currently reside in 1 of the 50 states or the District of Columbia. We believe the unemployment rate will have a negative relationship with the median sales price of new homes because home prices tend to increase as the economy is expanding and decrease when the economy is contracting. Conversley, the unemployment rate tends to decrease as the economy is expanding and increase as the economy is contracting. Due to this, we think the unemployment rate may provide a predictive relationship with the medain sales price of new homes.

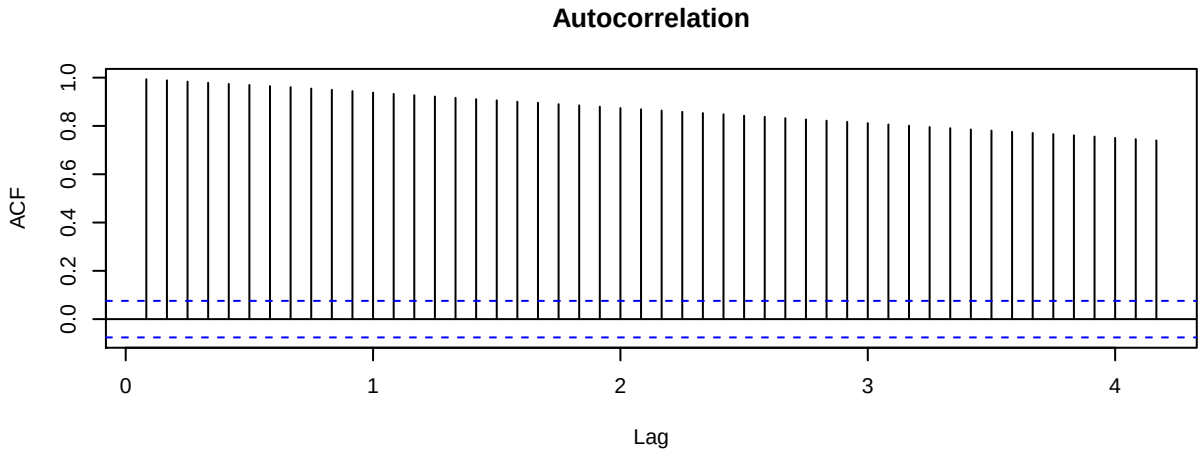
II. Results

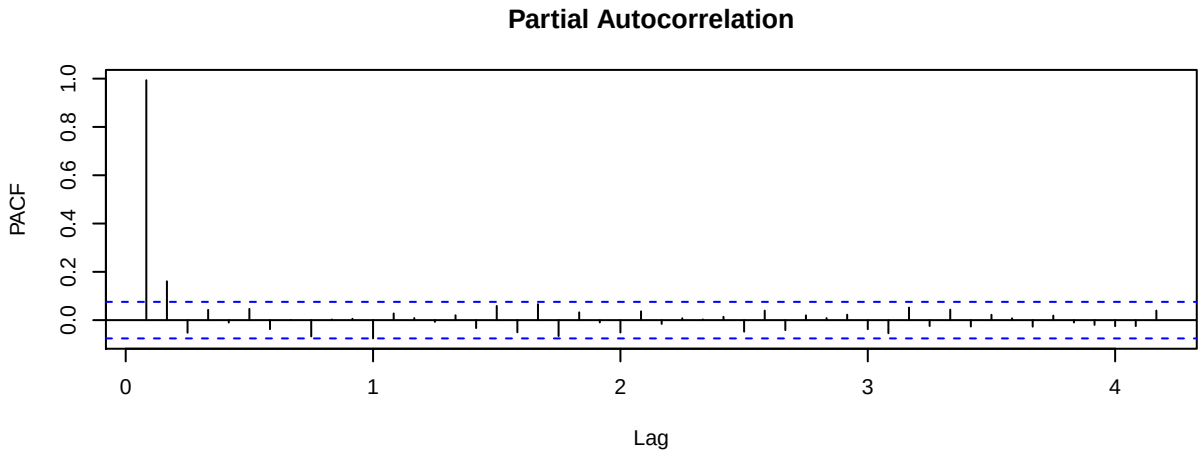
(a) Produce a time-series plot of your data including the respective ACF and PACF plots.

First we import data on median sales price for new houses sold in the United States.

```
# read in data and create time series object for price data
data = read.csv("MSP FRED.csv", header = TRUE)
data.ts = ts(data$MSPNHSUS, start = 1963, frequency = 12)
t = seq(1963, 2018.836, length = length(data.ts))

# plot data
par(mfrow=c(3,1))
plot(data.ts, xlab="Time", ylab="Sales Price", main = "Median Sales Price for New Houses In US")
acf(data.ts, type = "correlation", main="Autocorrelation",lag.max=50,ylab="ACF")
acf(data.ts, type = "partial",main="Partial Autocorrelation",lag.max=50, ylab="PACF")
```



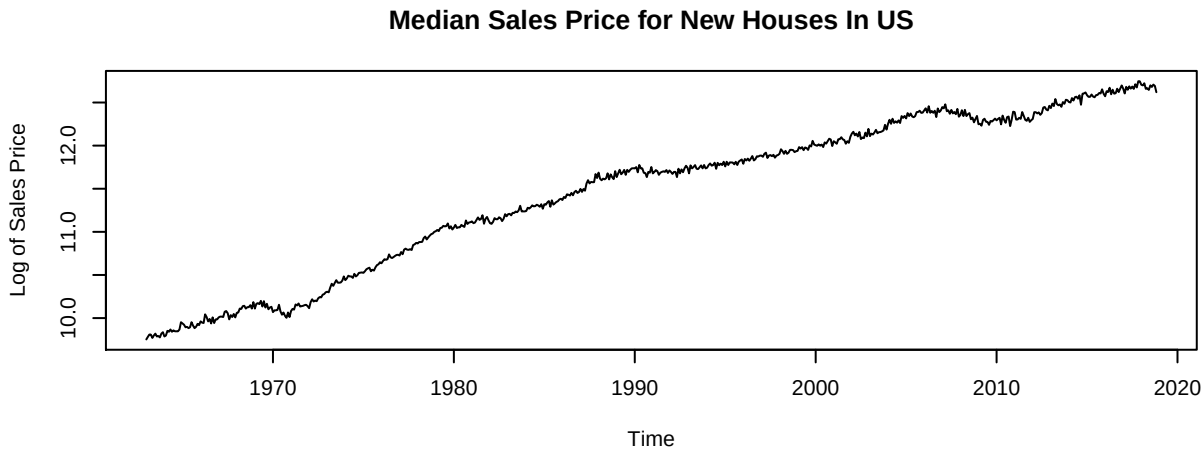


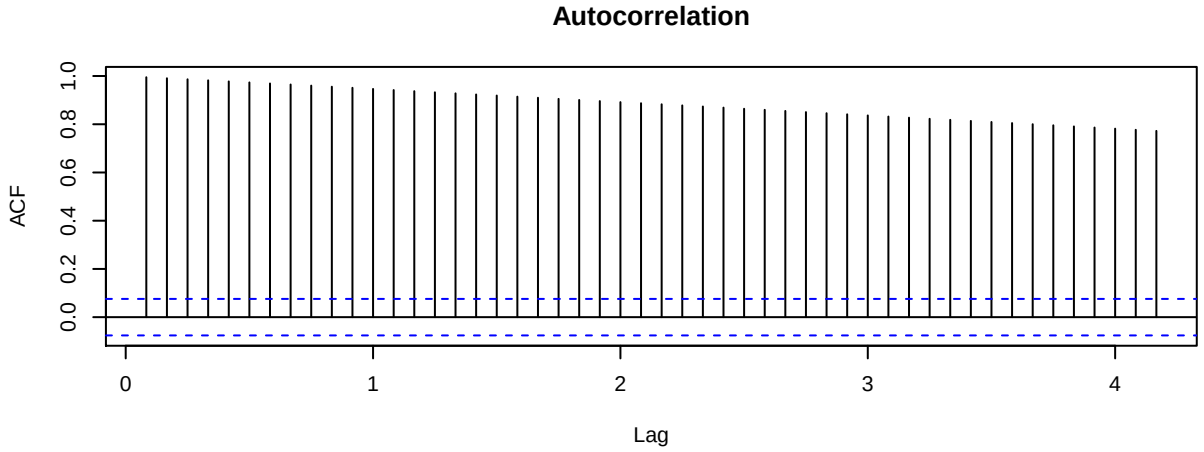


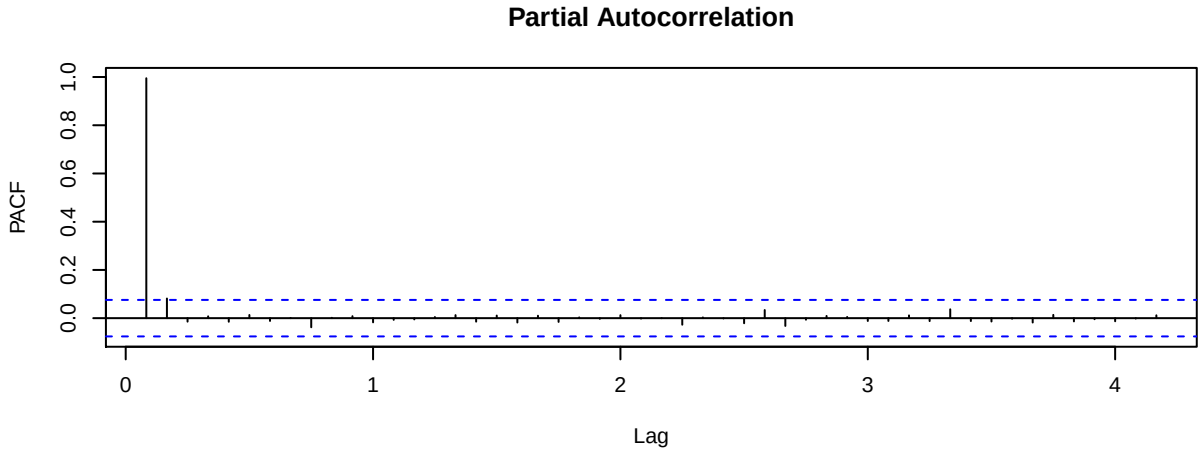
We try taking the log of the house price data to reduce the scale.

```
# log the price data
ldata.ts = log(data.ts)

# plot data
```

```r
par(mfrow=c(3,1))
plot(ldata.ts, xlab="Time", ylab="Log of Sales Price",
     main = "Median Sales Price for New Houses In US")
acf(ldata.ts, type = "correlation", main="Autocorrelation",lag.max=50,ylab="ACF")
acf(ldata.ts, type = "partial",main="Partial Autocorrelation",lag.max=50, ylab="PACF")
```



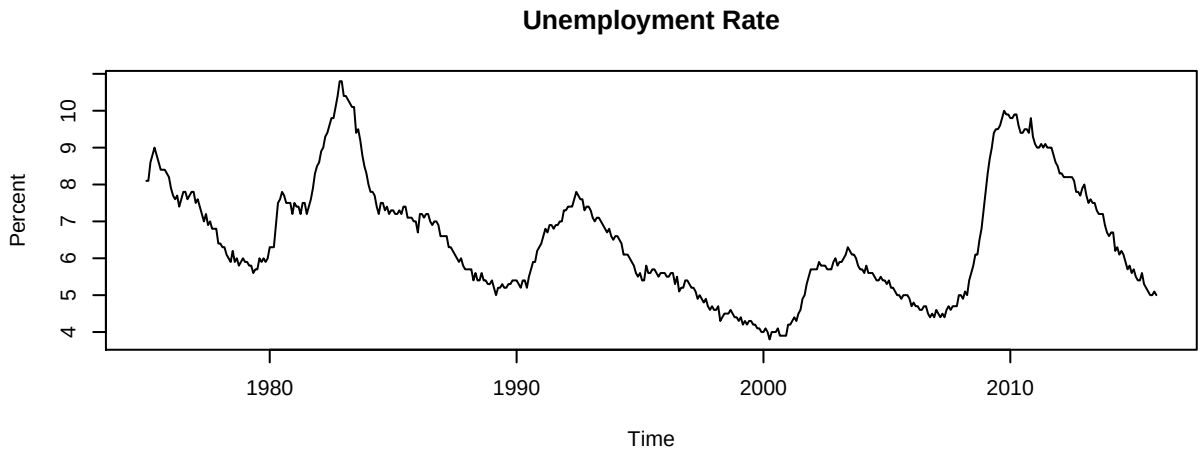


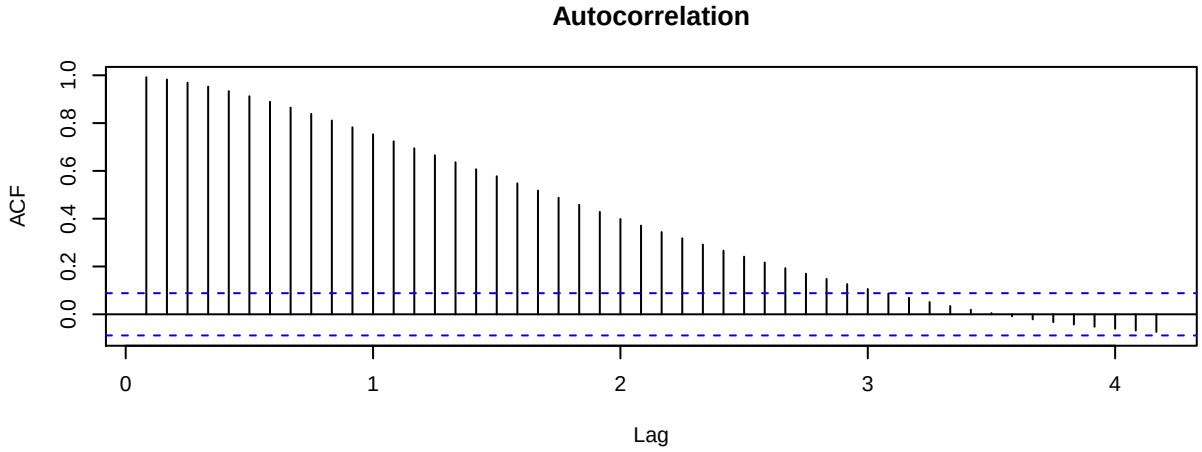


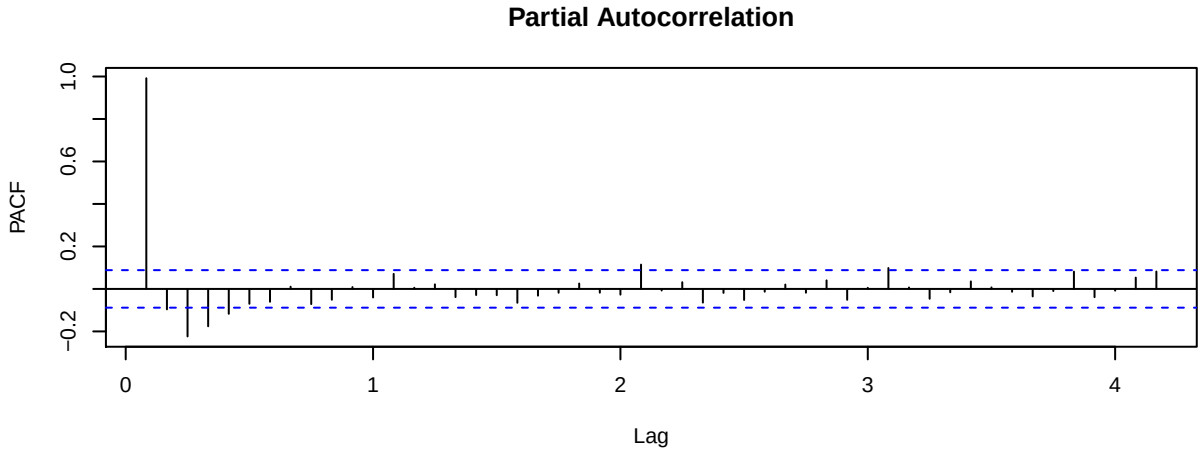
Next we import US unemployment rate data.

```
# read in and clean unemployment data
unrate = read.csv("UNRATE FRED.csv", header = TRUE)
date = as.Date(unrate$DATE, format = "%Y-%m-%d")
unrate = cbind(date, unrate)
```

```r
unrate = unrate[, c(1,3)]
unrate = subset(unrate, unrate$date >= "1975-01-01" & unrate$date <= "2015-12-01")
unrate.ts = ts(unrate$UNRATE, start = 1975, frequency = 12)
time = seq(1975, 2015.918, length = length(unrate.ts))

# plot data
par(mfrow=c(3,1))
plot(unrate.ts, xlab="Time", ylab="Percent", main = "Unemployment Rate")
acf(unrate.ts, type = "correlation", main="Autocorrelation",lag.max=50,ylab="ACF")
acf(unrate.ts, type = "partial",main="Partial Autocorrelation",lag.max=50, ylab="PACF")
```



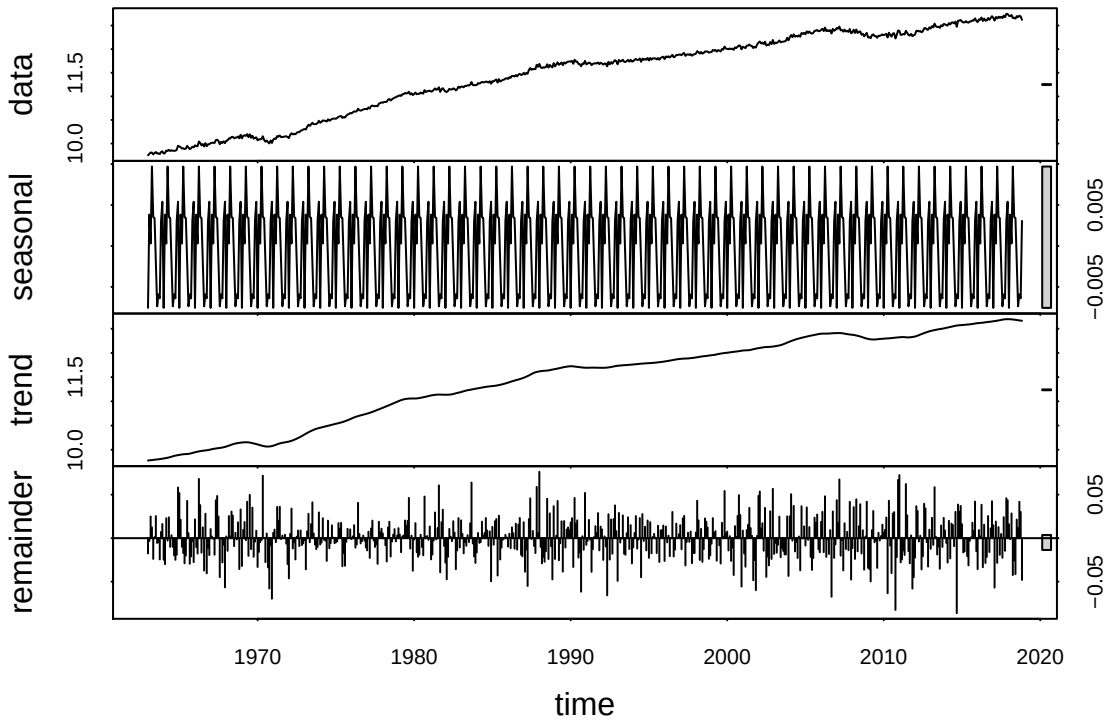




(b) Fit a model that includes, trend, seasonality and cyclical components. Make sure to discuss your model in detail.

First we fit a model to the median sales price data.

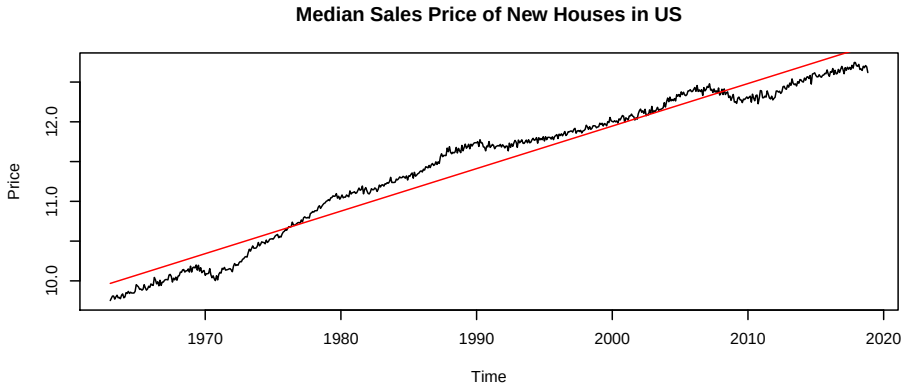
```
# look at decomposition of the time series of log of data
plot(stl(ldata.ts, s.window = "periodic"))
```

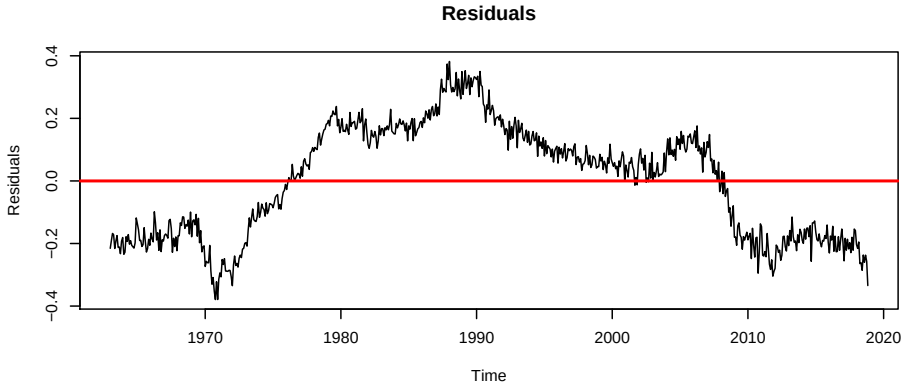


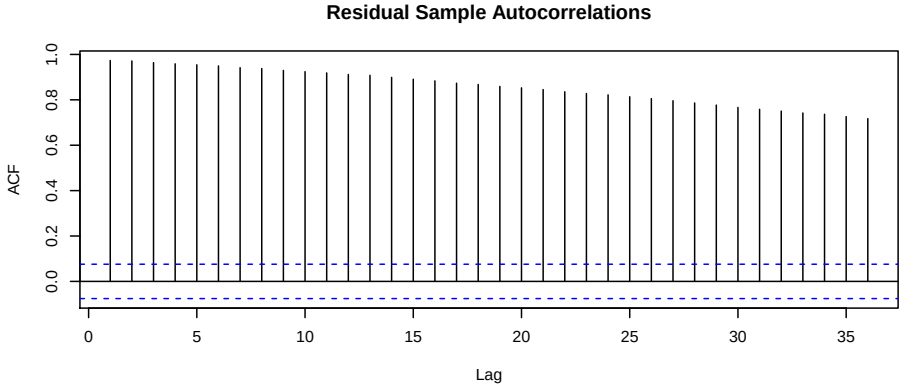
We can see from the stl decomposition of the data that there is a seasonal component but its order of magnitude is very small. Whereas the order of magnitude for the trend is significantly larger, indicating that there is a stronger trend component than seasonal component in the data.

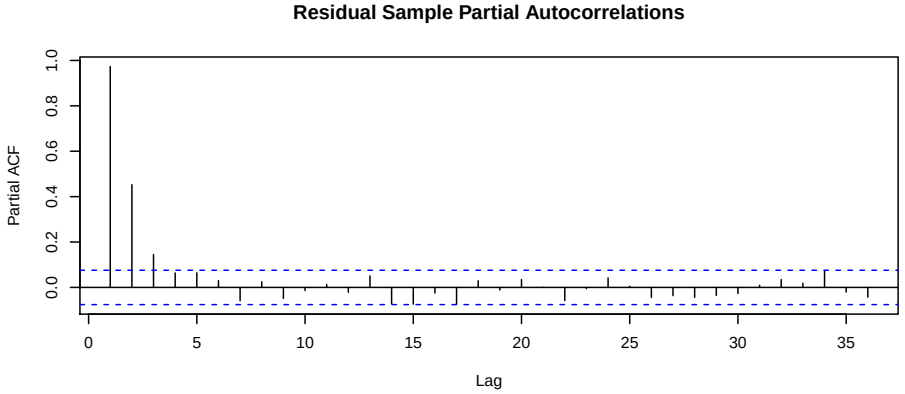
First we try fitting a linear trend to the model.

```
# Fit a linear trend model to price data
m1=lm(ldata.ts~t)

par(mfrow=c(4,1))
plot(ldata.ts,ylab="Price", xlab="Time", main = "Median Sales Price of New Houses in US")
lines(t,m1$fit,col="red")
plot(t,m1$res, ylab="Residuals",type='l',xlab="Time", main = "Residuals")
abline(h = 0, col = "red", lwd = 2)
acf(m1$res,lag=36,main="Residual Sample Autocorrelations")
pacf(m1$res,lag=36,main="Residual Sample Partial Autocorrelations")
```


Median Sales Price of New Houses in US
Price
10.0
11.0
12.0
1970
1980
1990
2000
2010
2020
Time

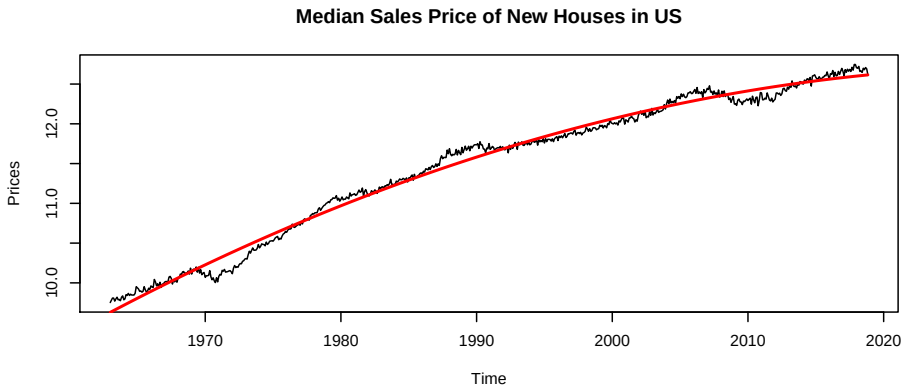

Residuals
Residuals
−0.4
−0.2
0.0
0.2
0.4
1970
1980
1990
2000
2010
2020
Time

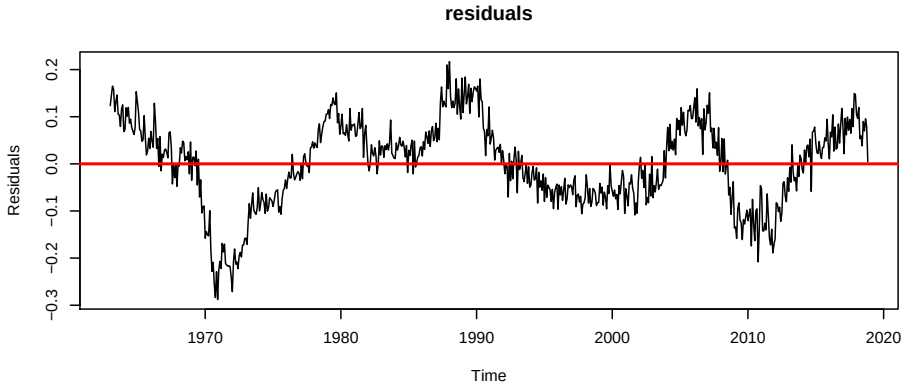

Residual Sample Autocorrelations
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag

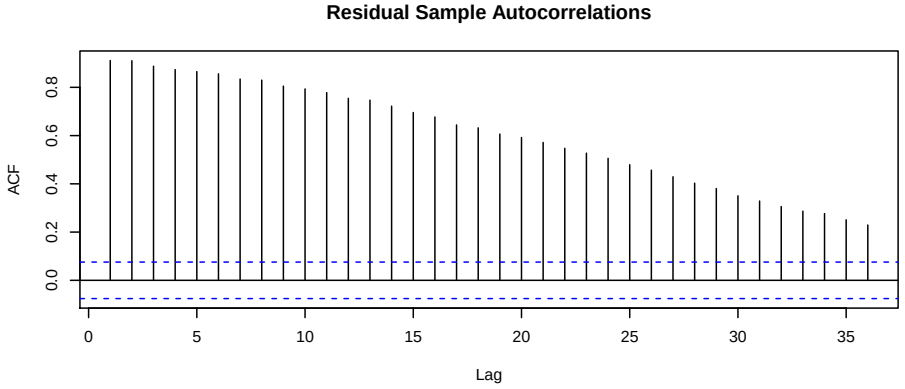

Residual Sample Partial Autocorrelations
Partial ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag

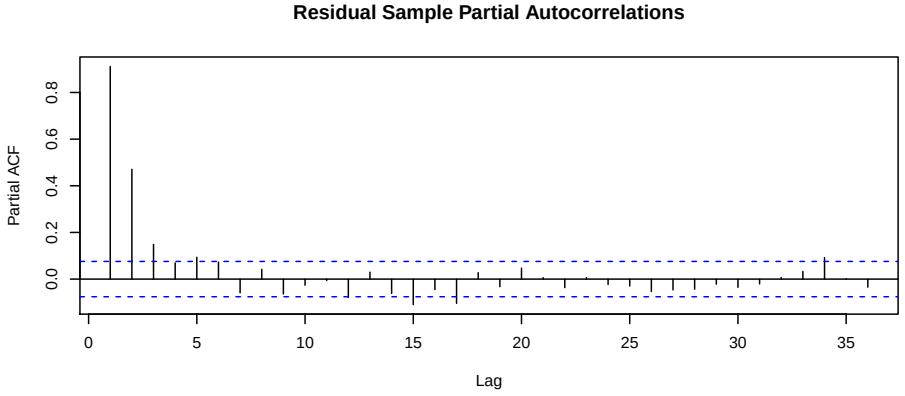
We can see clearly that the resulting residuals from the linear trend fit are not white noise. There appears to be a lot of persistence in the residuals, possibly indicating that an AR process will fit the residuals well. Looking at the ACF and the PACF, we confirm this belief because we see a slow decay in the ACF and spikes in the PACF.

```
# Fit a quadratic trend model to price data
t2 = t^2
m2=lm(ldata.ts ~ t + t2)

par(mfrow=c(4,1))
plot(ldata.ts,ylab="Prices", xlab="Time", main = "Median Sales Price of New Houses in US")
lines(t,m2$fit,col="red",lwd=2)
plot(t,m2$res, ylab="Residuals",type='l',xlab="Time", main = "residuals")
abline(h = 0, col = "red", lwd = 2)
acf(m2$res,lag=36,main="Residual Sample Autocorrelations")
pacf(m2$res,lag=36,main="Residual Sample Partial Autocorrelations")
```


Median Sales Price of New Houses in US
Prices
10.0
11.0
12.0
1970
1980
1990
2000
2010
2020
Time


residuals
Residuals
−0.3
−0.2
−0.1
0.0
0.1
0.2
1970
1980
1990
2000
2010
2020
Time

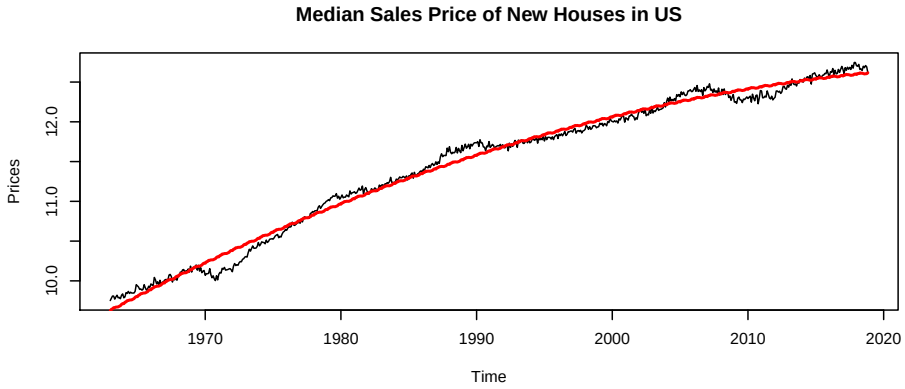

Residual Sample Autocorrelations
ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag

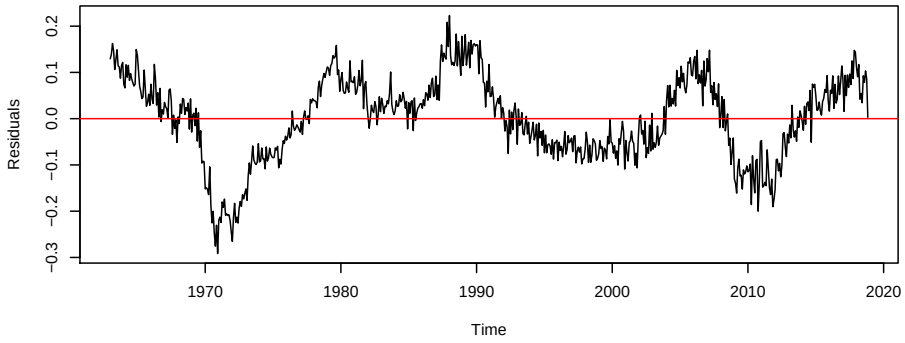

Residual Sample Partial Autocorrelations
Partial ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag

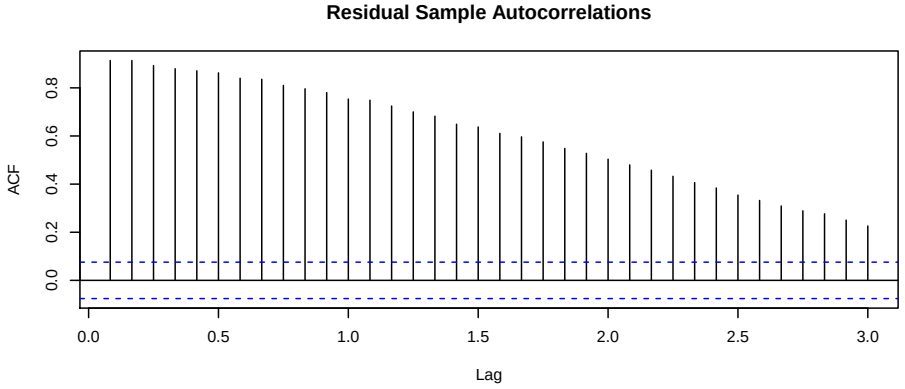
Looking only at the residuals plot, it is not clear if the quadratic trend improves the fit significantly compared with the linear trend because there still appears to be a large amount of structure and persistence in the residuals. They are not white noise. The ACF and PACF of the residuals for the quadratic fit indicate that the residuals are not white noise, confirming our belief that there is still structure left in the data that we can incorporate.

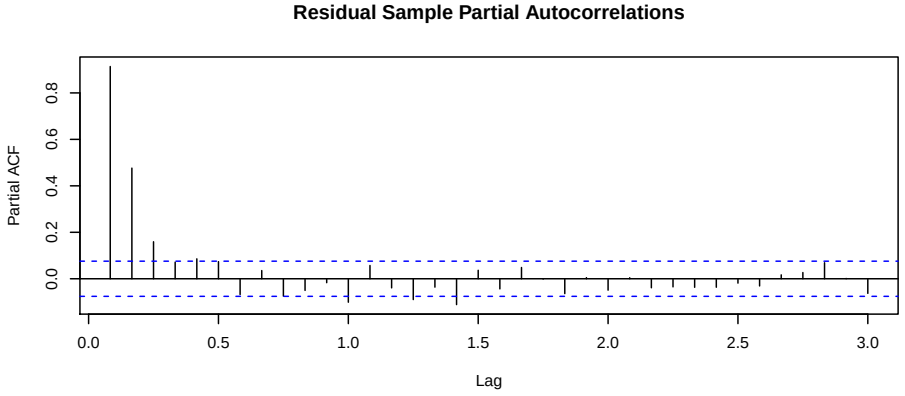
Now we try adding a seasonal component along with the quadratic trend.

```
# Fit a quadrtaic trend + seasonality model to price data
m3=tslm(ldata.ts ~ 0 + t + t2 + season)

par(mfrow=c(4,1))
plot(ldata.ts,ylab="Prices", xlab="Time", main = "Median Sales Price of New Houses in US")
lines(t,m3$fit,col="red",lwd=2)
plot(t,m3$res, ylab="Residuals",type='l',xlab="Time")
abline(h = 0, col = "red")
acf(m3$res,lag=36,main="Residual Sample Autocorrelations")
pacf(m3$res,lag=36,main="Residual Sample Partial Autocorrelations")
```


Median Sales Price of New Houses in US
Prices
10.0
11.0
12.0
1970
1980
1990
2000
2010
2020
Time


Residuals
−0.3
−0.2
−0.1
0.0
0.1
0.2
1970
1980
1990
2000
2010
2020
Time


Residual Sample Autocorrelations
ACF
0.0
0.2
0.4
0.6
0.8
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Lag


Residual Sample Partial Autocorrelations
Partial ACF
0.0
0.2
0.4
0.6
0.8
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Lag

We see that adding the seasonal component did not appear to improve the model fit very much. Also, it did not make the residuals more like white noise. We check the estimates of the fit to see if the seasonal components are significant.

```
# check estimates of m3 model (seasonal component)
summary(m3)
```

```
##
## Call:
## tslm(formula = ldata.ts ~ 0 + t + t2 + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29162 -0.05891  0.01145  0.06767  0.22308
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## t         2.654e+00  6.037e-02   43.97   <2e-16 ***
## t2       -6.531e-04  1.516e-05  -43.08   <2e-16 ***
## season1  -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season2  -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season3  -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season4  -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season5  -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season6  -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season7  -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season8  -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season9  -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season10 -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season11 -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## season12 -2.684e+03  6.009e+01  -44.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09152 on 657 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 7.56e+05 on 14 and 657 DF,  p-value: < 2.2e-16
```

After adding the seasonal component to the model, the fit and residuals do not appear to improve relative to the model with just the quadratic trend. We look at the summary of the trend plus seasonal model to check to significance of the estimates of the seasonal components. The trend and seasonal components of the model are all significant.

However, We saw from the residuals that they are not yet white noise. Additionally, we can see from the ACF and the PACF that the residuals are not white noise. This means that there is still some structure left that we can incorporate into our model to improve the fit. This indicates that there are cycles within the data that we can utilize. There is strong persistence in the residuals, indicating long-term memory in the residuals so an AR process may be appropriate to incorporate this strucure into the model.

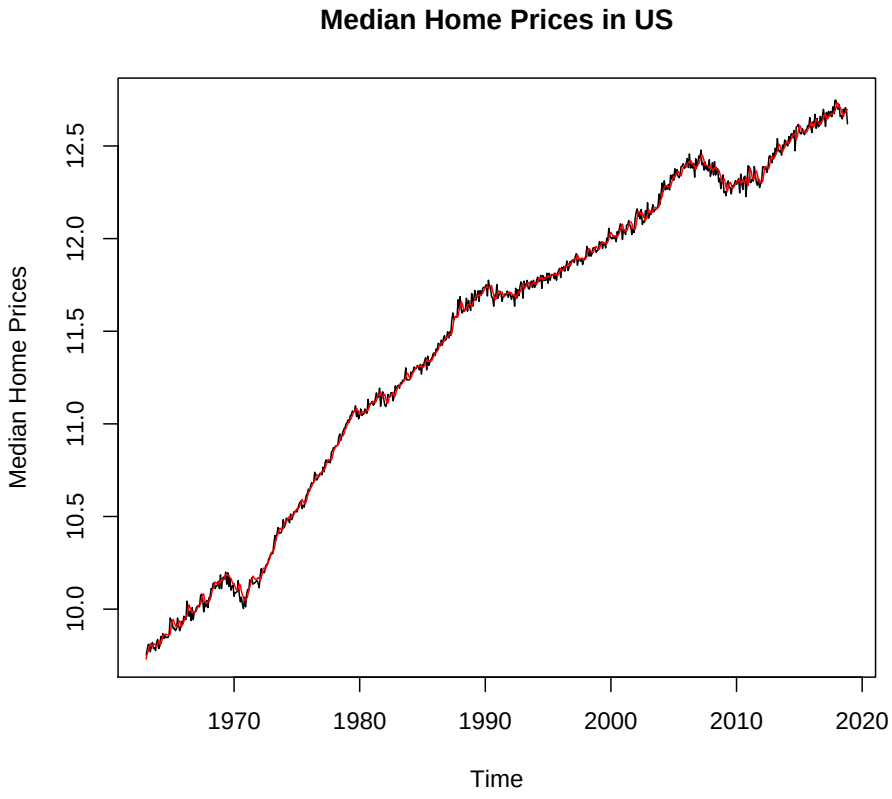
We can see that the ACF decays to zero and the PACF has spikes at lags 1 through 3. This decay in the ACF and spikes in the PACF indicates an AR(3) process.

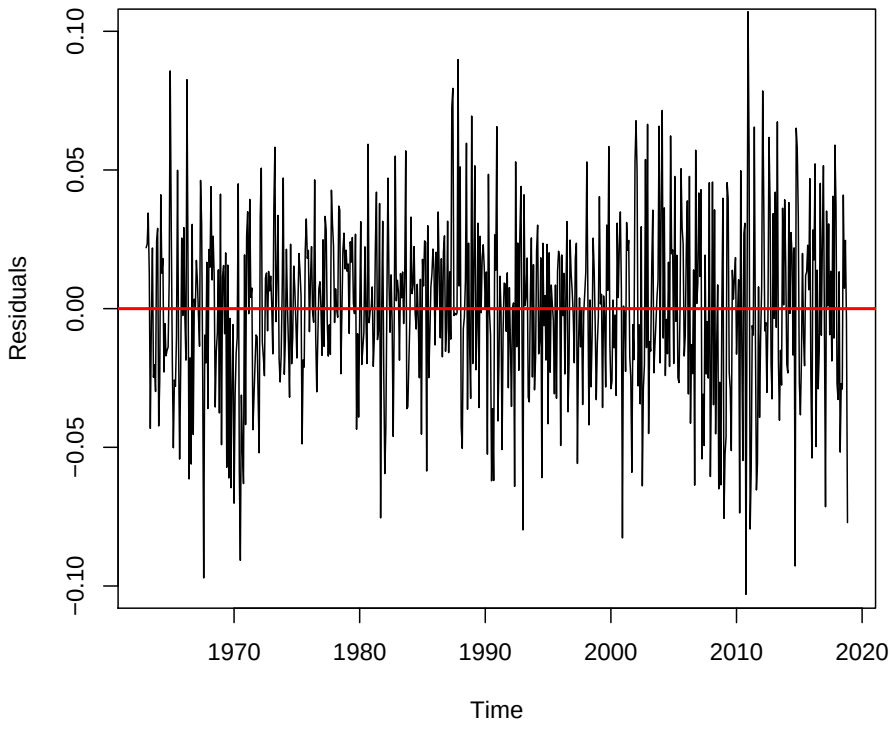
```
# Model: Quadratic Trend + Seasonal + Cycles
m4=Arima(ldata.ts,order=c(3,0,0), xreg = cbind(t, t2), seasonal=list(order=c(1,0,1)))
summary(m4)
```

```
## Series: ldata.ts
```

```
## Regression with ARIMA(3,0,0)(1,0,1)[12] errors
##
## Coefficients:
##          ar1     ar2     ar3     sar1    sma1   intercept       t      t2
##       0.3879  0.4048  0.1711  -0.4360  0.4179  -2376.1693  2.3458  -6e-04
## s.e.  0.0386  0.0379  0.0386   0.6263  0.6279    105.8772  0.1060   0e+00
##
## sigma^2 estimated as 0.001035:  log likelihood=1356.69
## AIC=-2695.37   AICc=-2695.1   BIC=-2654.79
##
## Training set error measures:
##                         ME       RMSE        MAE          MPE      MAPE
## Training set -0.0003069244 0.03198032 0.02517965 -0.004115566 0.2199765
##                    MASE        ACF1
## Training set 0.3704352 -0.01394004
```

```r
# plot of fit and residuals
par(mfrow=c(2,1))
plot(ldata.ts,ylab="Median Home Prices", xlab="Time", main="Median Home Prices in US")
lines(t,m4$fit,col="red",lwd=1)
plot(t, m4$residuals , ylab="Residuals",type="l",xlab="Time", ylim = c(-0.1,0.1))
abline(h = 0, col = "red", lwd = 2)
```

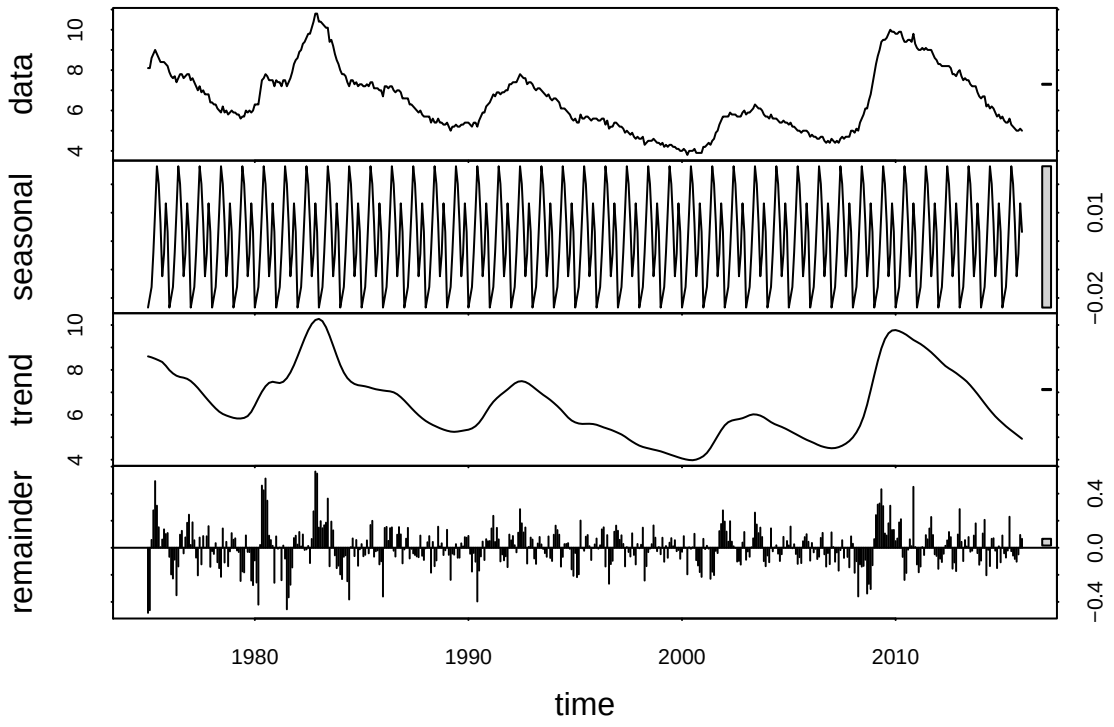

Median Home Prices in US
Median Home Prices
10.0
10.5
11.0
11.5
12.0
12.5
1970
1980
1990
2000
2010
2020
Time


Residuals
−0.10
−0.05
0.00
0.05
0.10
1970
1980
1990
2000
2010
2020
Time

We check if the residuals from this model are white noise in part (e).

Next we fit a model to the US unemployment rate data.

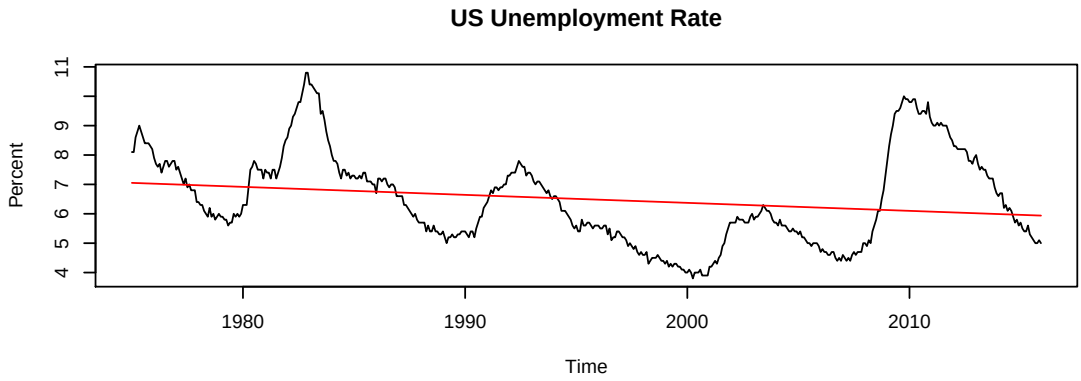
```
# look at decomposition of the time series of unrate
plot(stl(unrate.ts, s.window = "periodic"))
```

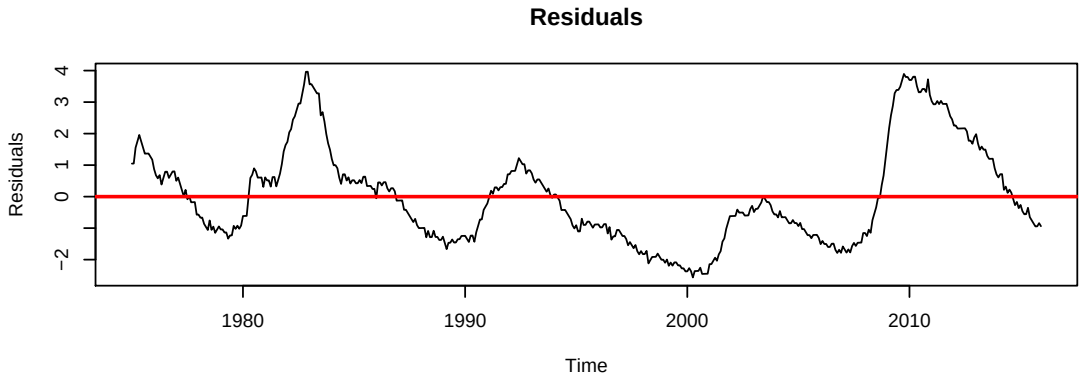


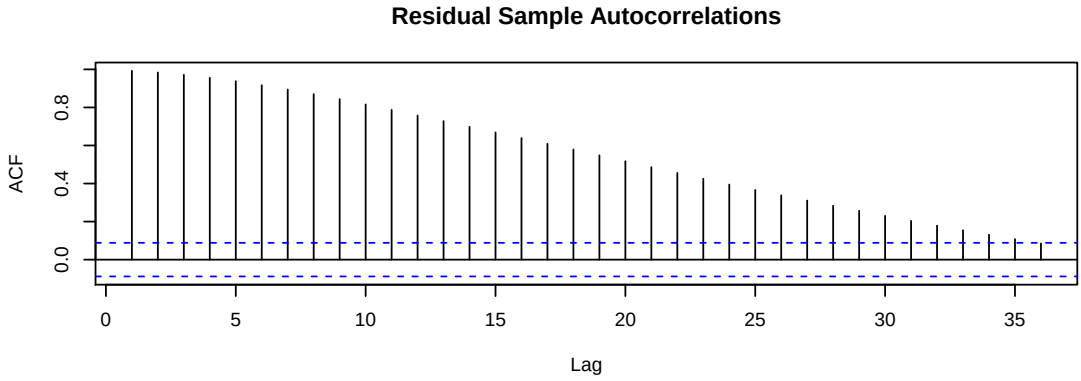
We can see that the series has a seasonal and cyclical component. However, the order of magnitude on the seasonal component is small relative to the data. From this, we suspect that the seasonal component will not play an important role.

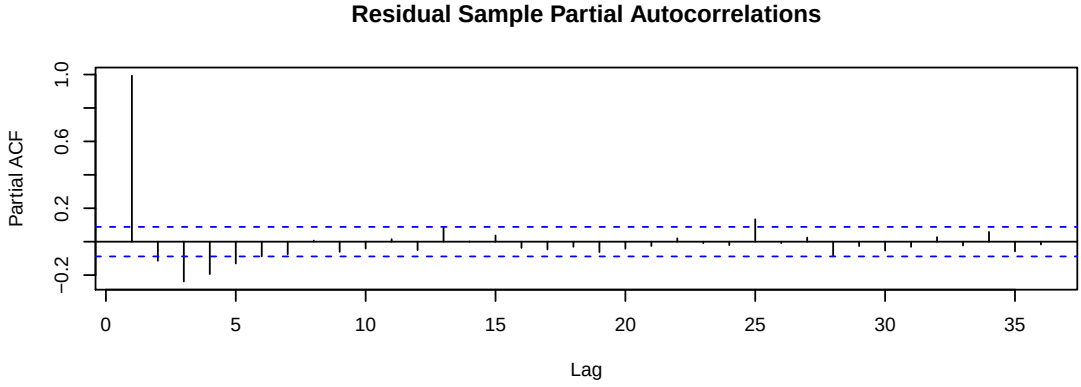
The first model we test is to fit a linear trend to the data.

```
# Fit a linear trend model to unrate
u1=lm(unrate.ts~time)

par(mfrow=c(4,1))
plot(unrate.ts,ylab="Percent", xlab="Time", main = "US Unemployment Rate")
lines(time,u1$fit,col="red")
plot(time,u1$res, ylab="Residuals",type='l',xlab="Time", main = "Residuals")
abline(h = 0, col = "red", lwd = 2)
acf(u1$res,lag=36,main="Residual Sample Autocorrelations")
pacf(u1$res,lag=36,main="Residual Sample Partial Autocorrelations")
```


US Unemployment Rate
Percent
4 5 6 7 8 9 11
1980 1990 2000 2010
Time

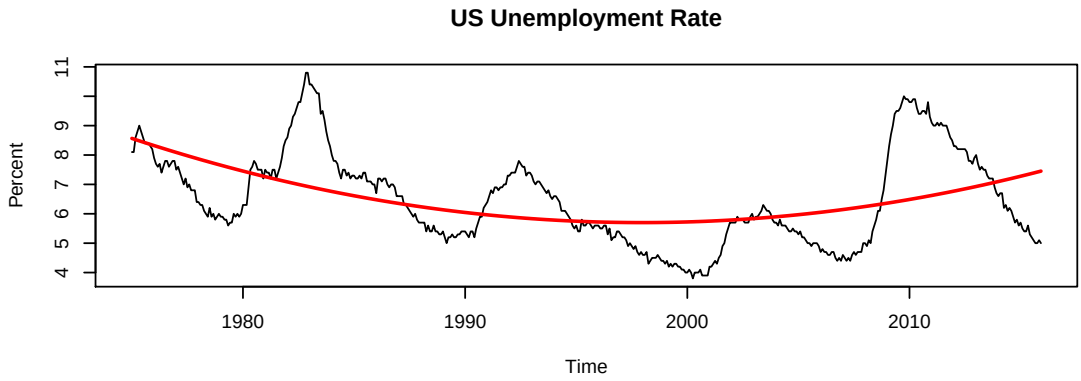

Residuals
Residuals
−2 0 1 2 3 4
1980 1990 2000 2010
Time

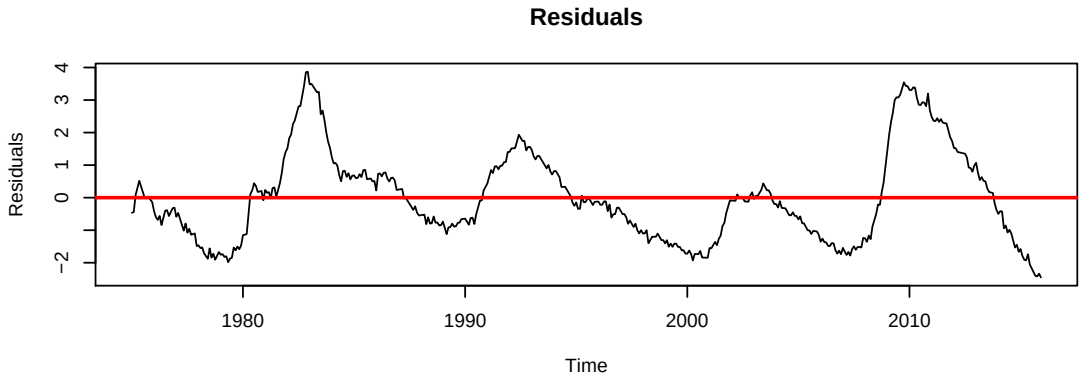

Residual Sample Autocorrelations
ACF
0.0 0.4 0.8
0 5 10 15 20 25 30 35
Lag

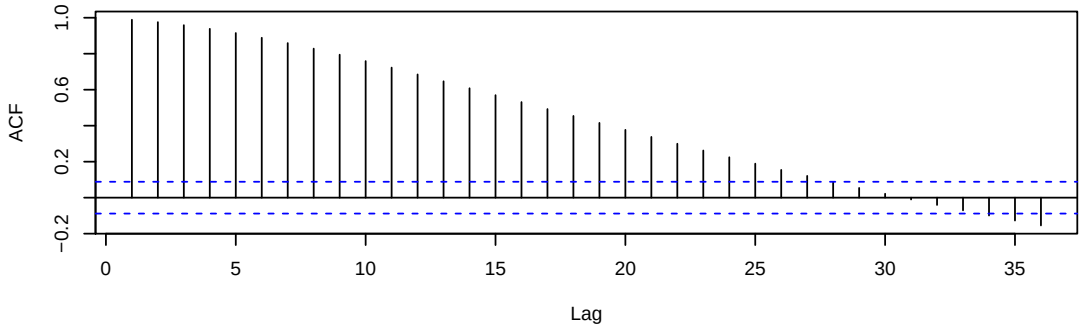

Residual Sample Partial Autocorrelations
Partial ACF
−0.2 0.2 0.6 1.0
0 5 10 15 20 25 30 35
Lag

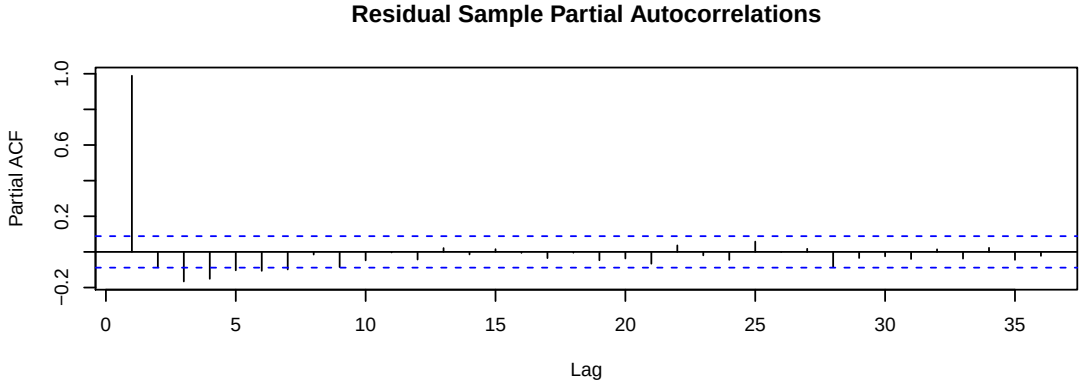
Fitting a linear trend to the model does not appear to improve the residuals. We can also see from the ACF and the PACF that the residuals have not changed very much. Next we fit a quadratic trend to the data.

```
# Fit a quadratic trend model
time2 = time^2
u2=lm(unrate.ts ~ time + time2)

par(mfrow=c(4,1))
plot(unrate.ts,ylab="Percent", xlab="Time", main = "US Unemployment Rate")
lines(time,u2$fit,col="red",lwd=2)
plot(time,u2$res, ylab="Residuals",type='l',xlab="Time", main = "Residuals")
abline(h = 0, col = "red", lwd = 2)
acf(u2$res,lag=36,main="Residual Sample Autocorrelations")
pacf(u2$res,lag=36,main="Residual Sample Partial Autocorrelations")
```


US Unemployment Rate
Percent
4
5
6
7
8
9
11
1980
1990
2000
2010
Time

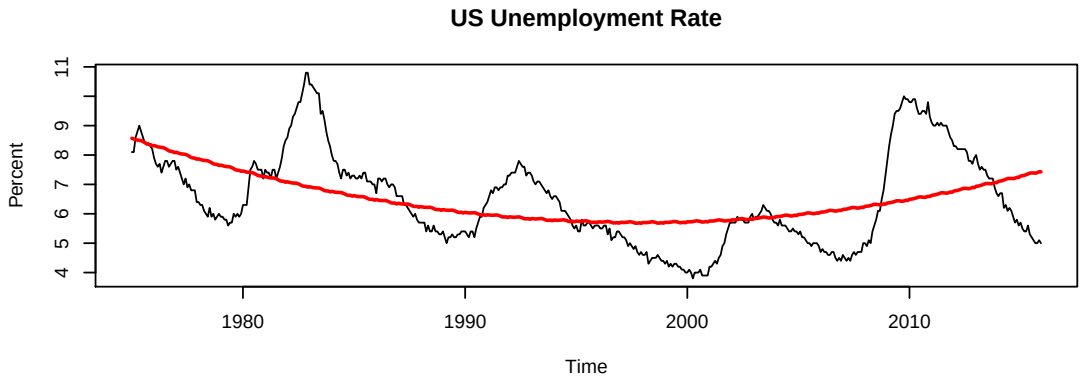

Residuals
Residuals
−2
0
1
2
3
4
1980
1990
2000
2010
Time

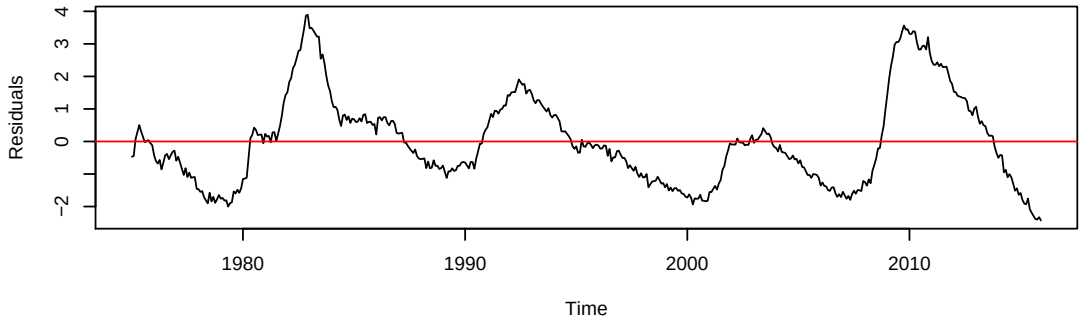

Residual Sample Autocorrelations
ACF
−0.2
0.2
0.6
1.0
0
5
10
15
20
25
30
35
Lag

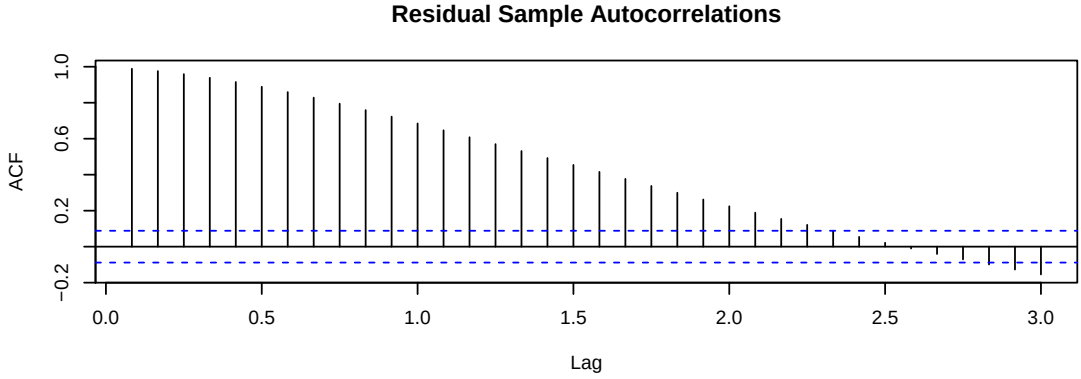

Residual Sample Partial Autocorrelations
Partial ACF
−0.2
0.2
0.6
1.0
0
5
10
15
20
25
30
35
Lag

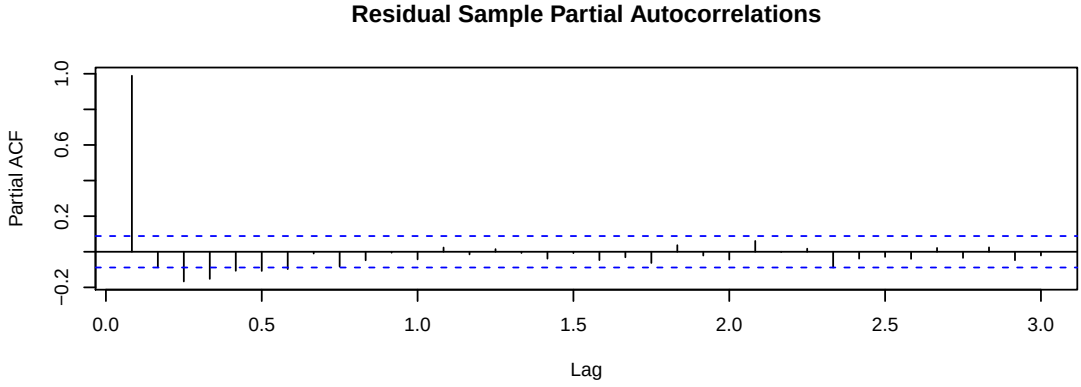
Again we can see that the residuals have not improved very much from incorportating a quadratic trend into the model. Next we try adding a seasonal component to the model.

```
# Fit a quadrtaic trend + seasonality model to unrate
u3=tslm(unrate.ts ~ 0 + time + time2 + season)

par(mfrow=c(4,1))
plot(unrate.ts,ylab="Percent", xlab="Time", main = "US Unemployment Rate")
lines(time,u3$fit,col="red",lwd=2)
plot(time,u3$res, ylab="Residuals",type='l',xlab="Time")
abline(h = 0, col = "red")
acf(u3$res,lag=36,main="Residual Sample Autocorrelations")
pacf(u3$res,lag=36,main="Residual Sample Partial Autocorrelations")
```


US Unemployment Rate
Percent
4 5 6 7 8 9 11
1980 1990 2000 2010
Time


Residuals
−2 0 1 2 3 4
1980 1990 2000 2010
Time


Residual Sample Autocorrelations
ACF
−0.2 0.2 0.6 1.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0
Lag


Residual Sample Partial Autocorrelations
Partial ACF
−0.2 0.2 0.6 1.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0
Lag

We see that adding the seasonal component did not appear to improve the model fit very much. Also, it did not make the residuals more like white noise. We check the estimates of the fit to see if the seasonal components are significant.

```
summary(u3)
```

```
##
## Call:
## tslm(formula = unrate.ts ~ 0 + time + time2 + season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4270 -1.0845 -0.1658  0.7477  3.8952
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## time     -2.171e+01  2.011e+00  -10.80   <2e-16 ***
## time2     5.433e-03  5.038e-04   10.78   <2e-16 ***
## season1   2.169e+04  2.006e+03   10.81   <2e-16 ***
## season2   2.169e+04  2.006e+03   10.81   <2e-16 ***
## season3   2.169e+04  2.006e+03   10.81   <2e-16 ***
## season4   2.169e+04  2.006e+03   10.81   <2e-16 ***
## season5   2.169e+04  2.006e+03   10.81   <2e-16 ***
## season6   2.169e+04  2.006e+03   10.81   <2e-16 ***
## season7   2.169e+04  2.006e+03   10.81   <2e-16 ***
## season8   2.169e+04  2.006e+03   10.81   <2e-16 ***
## season9   2.169e+04  2.006e+03   10.81   <2e-16 ***
## season10  2.169e+04  2.006e+03   10.81   <2e-16 ***
## season11  2.169e+04  2.006e+03   10.81   <2e-16 ***
## season12  2.169e+04  2.006e+03   10.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.4 on 478 degrees of freedom
## Multiple R-squared:  0.9573, Adjusted R-squared:  0.9561
## F-statistic: 766.3 on 14 and 478 DF,  p-value: < 2.2e-16
```

After adding the seasonal component to the model, the fit and residuals do not appear to improve relative to the model with just the quadratic trend. We look at the summary of the trend plus seasonal model to check the significance of the estimates of the seasonal components. The trend and seasonal components of the model are all significant. However, the linear and quadratic trend do not appear to capture much of the variation in the data. Because of this, we consider fitting cycles and seasonal components only to the model.

We saw from the residuals that they are not yet white noise. Additionally, we can see from the ACF and the PACF that the residuals are not white noise. This means that there is still some structure left that we can incorporate into our model to improve the fit. This indicates that there are cycles within the data that we can utilize. There is strong persistence in the residuals, indicating long-term memory in the residuals so an AR process may be appropriate to incorporate this strucure into the model.

We can see that the ACF decays to zero and the PACF has spikes at lags 1 through 4. This decay in the ACF and spikes in the PACF indicates an AR(4) process.

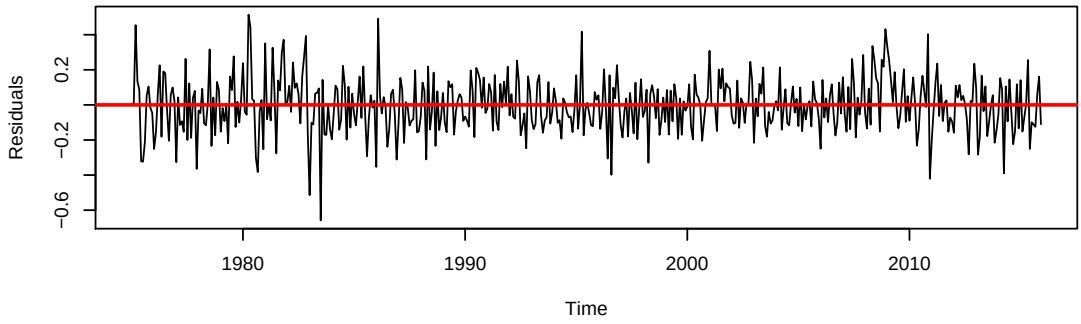
```
# Model: Quadratic Trend + Seasonal + Cycles for unrate
u4=Arima(unrate.ts,order=c(4,1,0), seasonal=list(order=c(1,0,1)))
summary(u4)
```

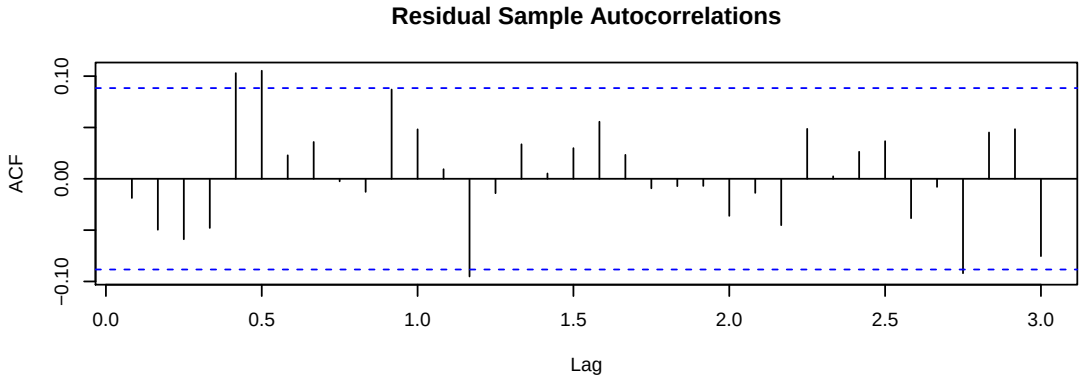
```
## Series: unrate.ts
```

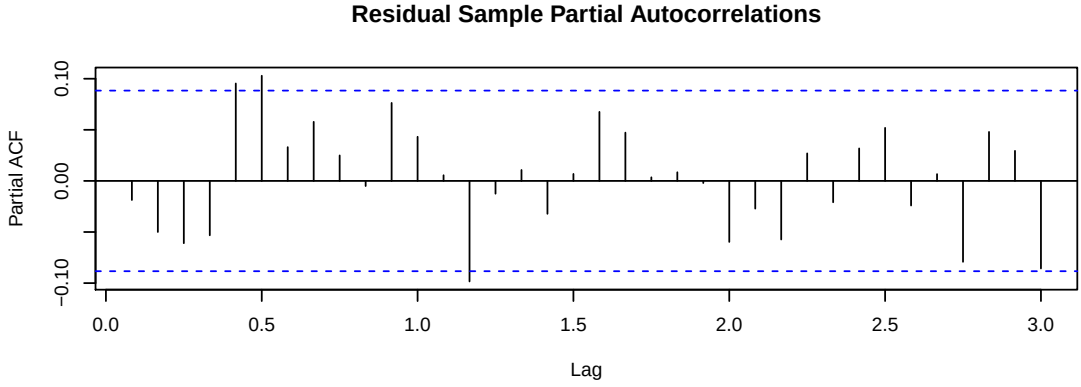
```
## ARIMA(4,1,0)(1,0,1)[12]
##
## Coefficients:
##          ar1     ar2     ar3     ar4    sar1     sma1
##       0.0426  0.1641  0.1950  0.1570  0.6259  -0.8525
## s.e.  0.0447  0.0440  0.0446  0.0455  0.0783   0.0560
##
## sigma^2 estimated as 0.02503:  log likelihood=209.91
## AIC=-405.82   AICc=-405.59   BIC=-376.45
##
## Training set error measures:
##                       ME      RMSE      MAE         MPE    MAPE      MASE
## Training set -0.003901349 0.1570682 0.123227 -0.09884671 1.95289 0.1565615
##                     ACF1
## Training set -0.01870132
```

```r
# plot of fit and residuals
par(mfrow=c(4,1))
plot(unrate.ts,ylab="Percent", xlab="Time", main=" US Unemployment Rate")
lines(time,u4$fit,col="red",lwd=1)
plot(time, u4$residuals , ylab="Residuals",type="l",xlab="Time")
abline(h = 0, col = "red", lwd = 2)
acf(u4$res,lag=36,main="Residual Sample Autocorrelations")
pacf(u4$res,lag=36,main="Residual Sample Partial Autocorrelations")
```


US Unemployment Rate
Percent
4
5
6
7
8
9
11
1980
1990
2000
2010
Time


Residuals
−0.6
−0.2
0.2
1980
1990
2000
2010
Time


Residual Sample Autocorrelations
ACF
−0.10
0.00
0.10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Lag


Residual Sample Partial Autocorrelations
Partial ACF
−0.10
0.00
0.10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Lag

```r
# null hypothesis is that residuals are white noise
Box.test(u4$residuals)
```

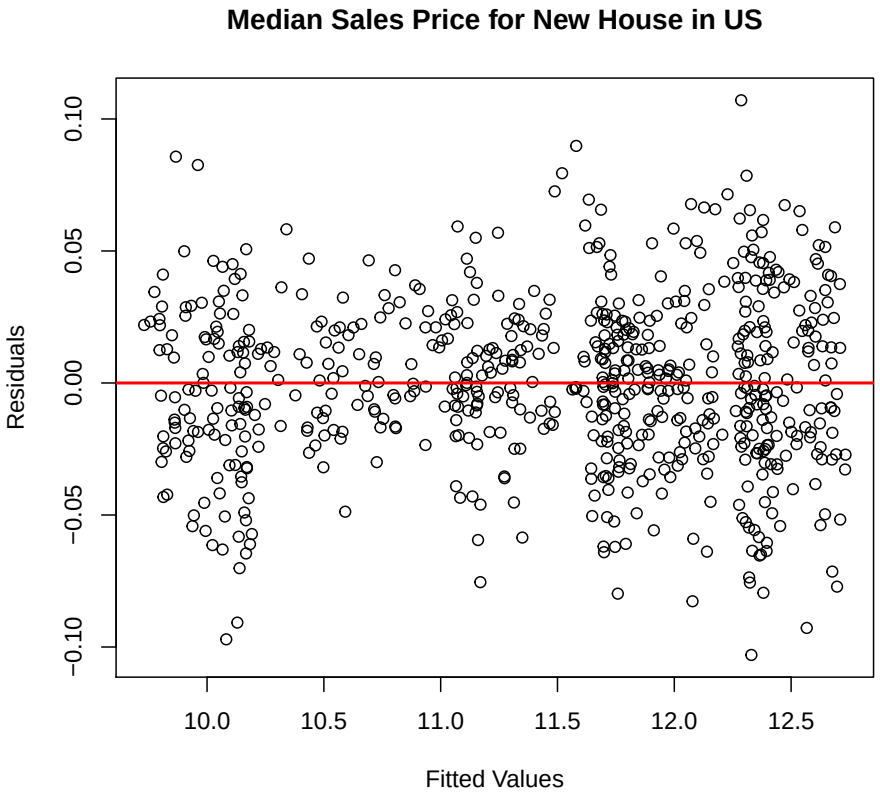
```
##
##  Box-Pierce test
##
## data:  u4$residuals
## X-squared = 0.17207, df = 1, p-value = 0.6783
```

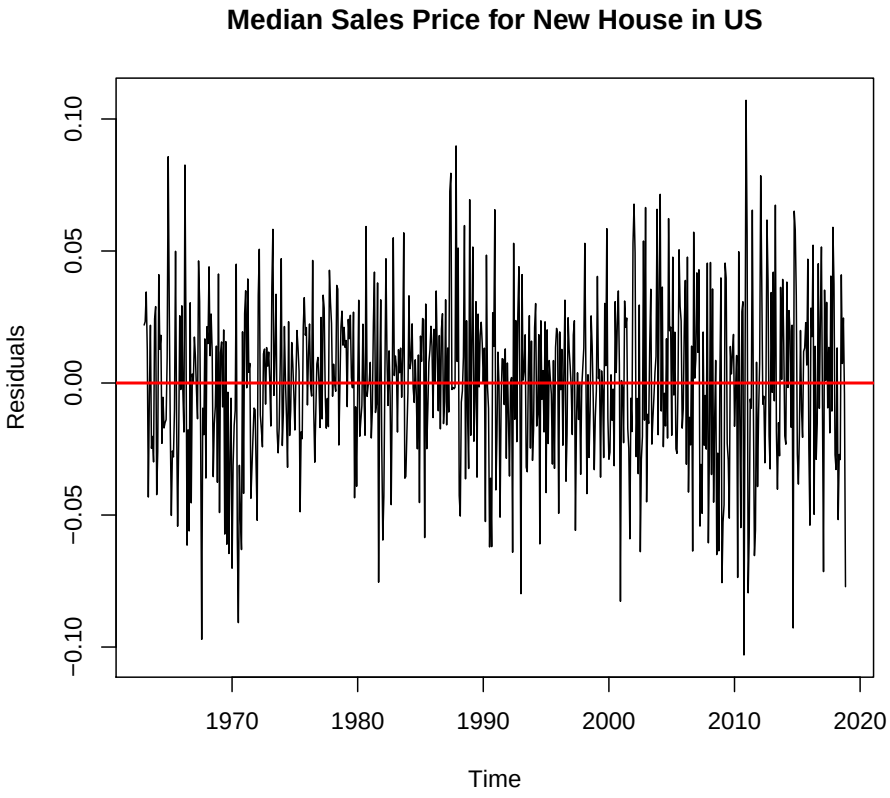
From the ACF and PACF we can see some spikes there are close to the 95% confidence bands so it is unclear if the residuals are white noise. We perform a Box-Pierce test, which has as a null hypothesis that the data is white noise. Our p-value from this test is 0.67, indicating that we fail to reject the null hypothesis and our residuals are in fact white noise.

(c) Plot the respective residuals vs. fitted values and discuss your observations.

First we plot the residuals vs fitted values and residuals vs time for the median house price data.

```r
par(mfrow=c(2,1))

# plot of residuals vs. fitted values for m4
plot(m4$fitted, m4$residuals, xlab = "Fitted Values", ylab = "Residuals",
     main = "Median Sales Price for New House in US")
abline(h = 0, col = "red", lwd = 2)

# plot of residuals vs. time for m4
plot(t, m4$residuals, type = "l", xlab = "Time", ylab = "Residuals",
     main = "Median Sales Price for New House in US")
abline(h = 0, col = "red", lwd = 2)
```

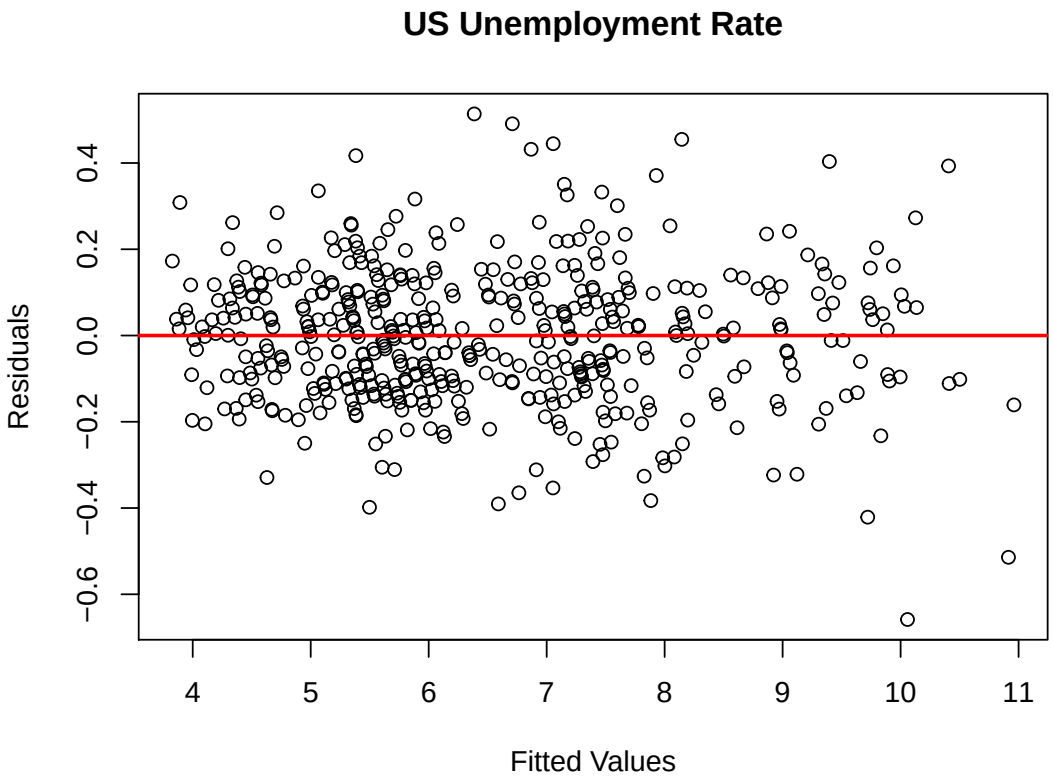

Median Sales Price for New House in US
Residuals
0.10
0.05
0.00
−0.05
−0.10
10.0
10.5
11.0
11.5
12.0
12.5
Fitted Values

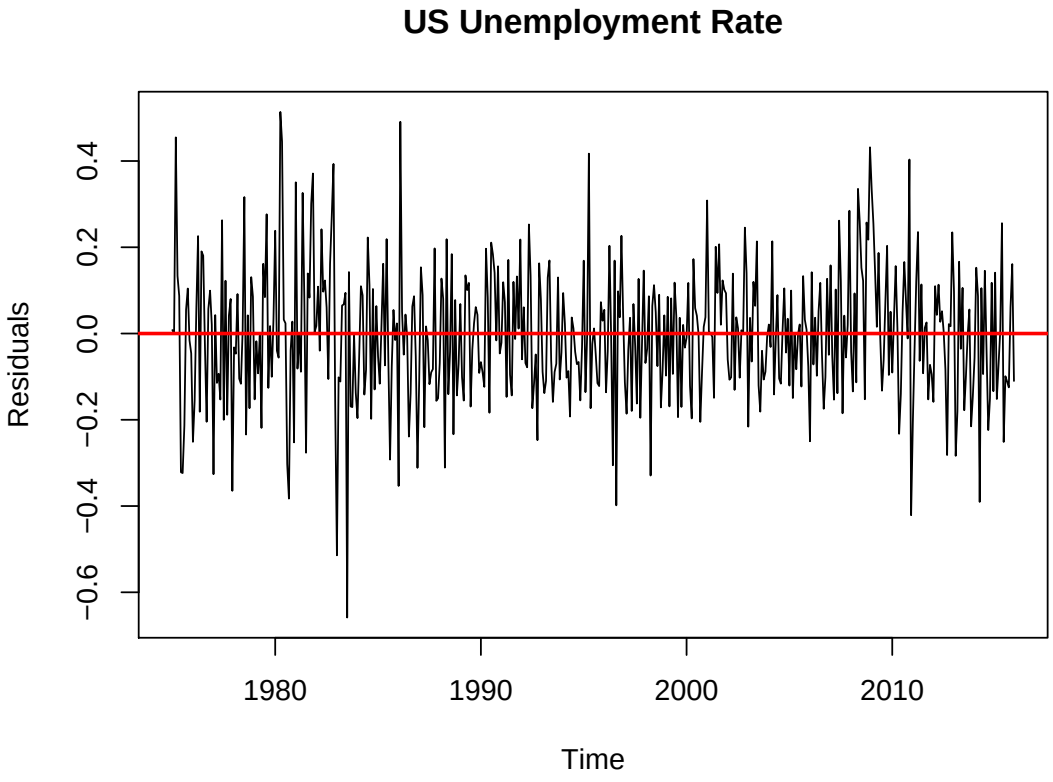

Median Sales Price for New House in US
Residuals
0.10
0.05
0.00
−0.05
−0.10
1970
1980
1990
2000
2010
2020
Time

The residuals against the fitted values appear to be randomly distributed about zero with a constant variance. The residuals against time also appear to be randomly distributed about zero with a constant variance.

Next we plot the residuals vs fitted values and residuals vs time for the unemployment rate data.

```
par(mfrow=c(2,1))

# plot of residuals vs. fitted values for u4
plot(u4$fitted, u4$residuals, xlab = "Fitted Values", ylab = "Residuals",
     main = "US Unemployment Rate")
abline(h = 0, col = "red", lwd = 2)

# plot of residuals vs. time for u4
plot(time, u4$residuals, type = "l", xlab = "Time", ylab = "Residuals",
     main = "US Unemployment Rate")
abline(h = 0, col = "red", lwd = 2)
```


US Unemployment Rate
Residuals
0.4
0.2
0.0
−0.2
−0.4
−0.6
4
5
6
7
8
9
10
11
Fitted Values


US Unemployment Rate
Residuals
0.4
0.2
0.0
−0.2
−0.4
−0.6
1980
1990
2000
2010
Time

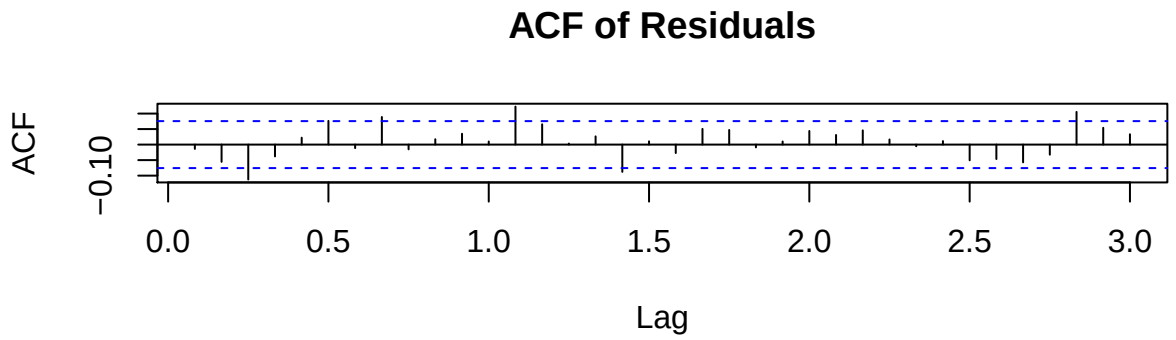
Again we can see that the residuals against the fitted values appear to be randomly distributed about zero with a constant variance. The residuals against time also appear to be randomly distributed about zero with a constant variance.

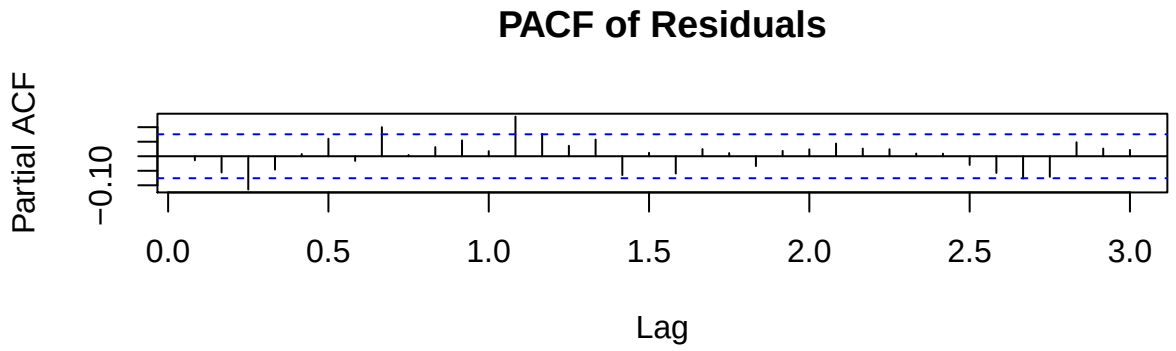
(d) There is no part (d) in the project instructions.

(e) Plot the ACF and PACF of the respective residuals and interpret the plots.

First we plot the ACF and the PACF of the residuals for the median sales price for new houses data. Then we perform the Box-Pierce test on the residuals to confirm that they are in fact white noise.

```
# ACF and PACF of residuals from m4
par(mfrow=c(2,1))
acf(m4$res,lag=36, main="ACF of Residuals")
pacf(m4$res,lag=36, main="PACF of Residuals")
```



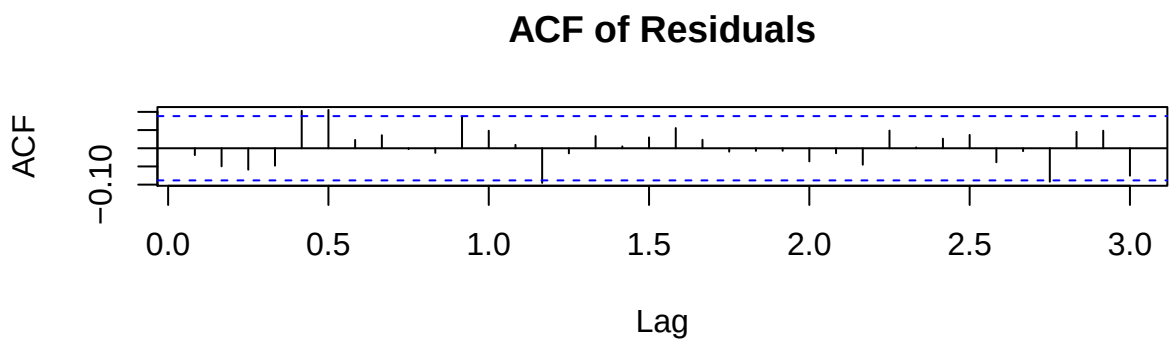


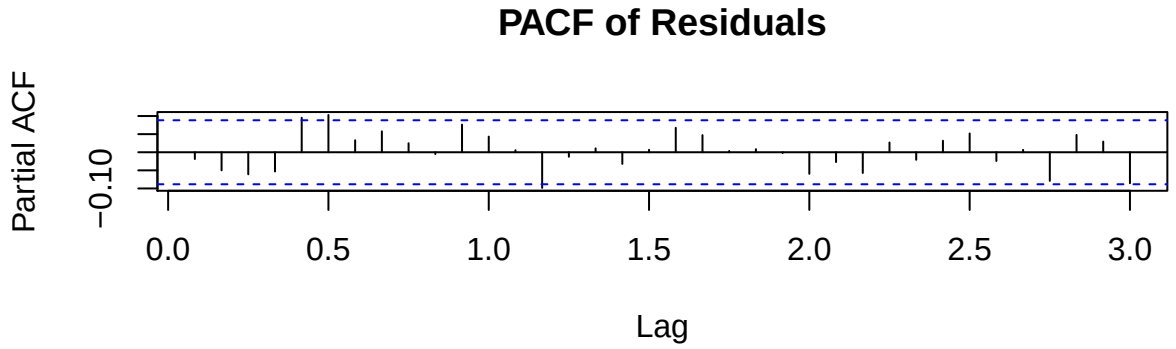

```
# null hypothesis is that residuals are white noise
Box.test(m4$residuals)
```

```
##
##  Box-Pierce test
##
## data:  m4$residuals
## X-squared = 0.13039, df = 1, p-value = 0.718
```

Next we plot the ACF and the PACF of the residuals for the US unemployment rate data. Then we perform the Box-Pierce test on the residuals to confirm that they are in fact white noise.

```r
# ACF and PACF of residuals from u4
par(mfrow=c(2,1))
acf(u4$res,lag=36, main="ACF of Residuals")
pacf(u4$res,lag=36, main="PACF of Residuals")
```





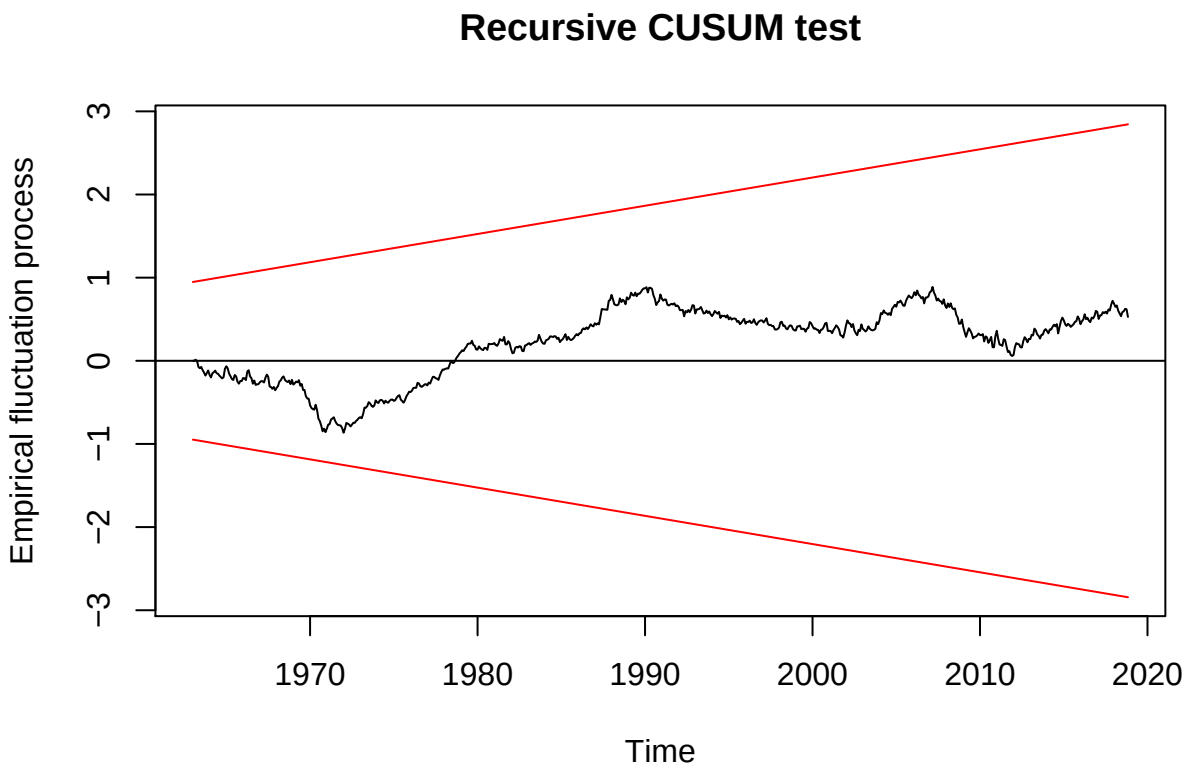

```r
# null hypothesis is that residuals are white noise
Box.test(u4$residuals)
```

```
##
##  Box-Pierce test
##
## data:  u4$residuals
## X-squared = 0.17207, df = 1, p-value = 0.6783
```

(f) Plot the respective CUSUM and interpret the plot.

First we look at the CUSUM for the median sales price data. Then we perform a structural change test.

```r
# plot CUSUM plot for m4
plot(efp(m4$res ~ 1, type = "Rec-CUSUM"))
```



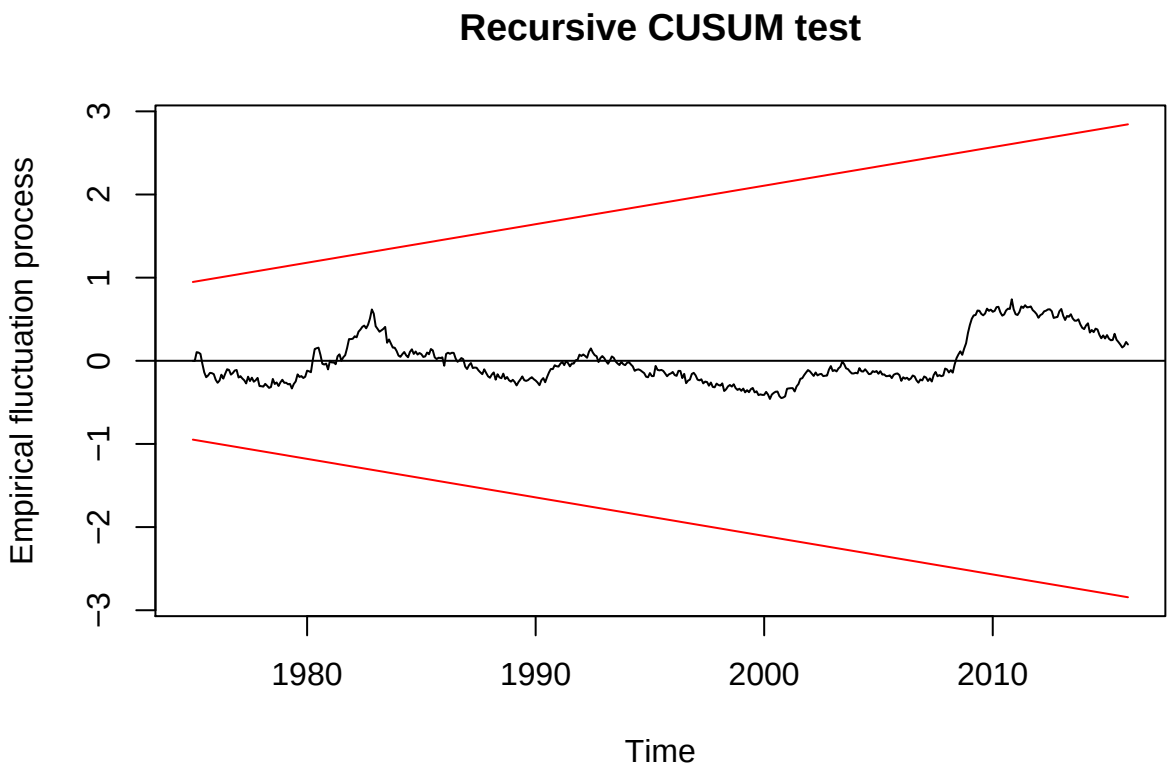

```
sctest(m4$residuals ~ 1)
```

```
##
##  Recursive CUSUM test
##
## data:  m4$residuals ~ 1
## S = 0.66839, p-value = 0.295
```

We can see from the plot of the CUSUM that there are no structural breaks in the data. Then looking at the structural change test, we confirm that there are no structural breaks in the data. The null hypothesis is that there are no structural breaks. From the test, we fail reject the null, indicating that there are no structural breaks in our fit.

Next we look at the CUSUM for the unemployment rate data and perform a structrual change test.

```
# plot CUSUM plot for m4
plot(efp(u4$res ~ 1, type = "Rec-CUSUM"))
```




```
sctest(u4$residuals ~ 1)
```

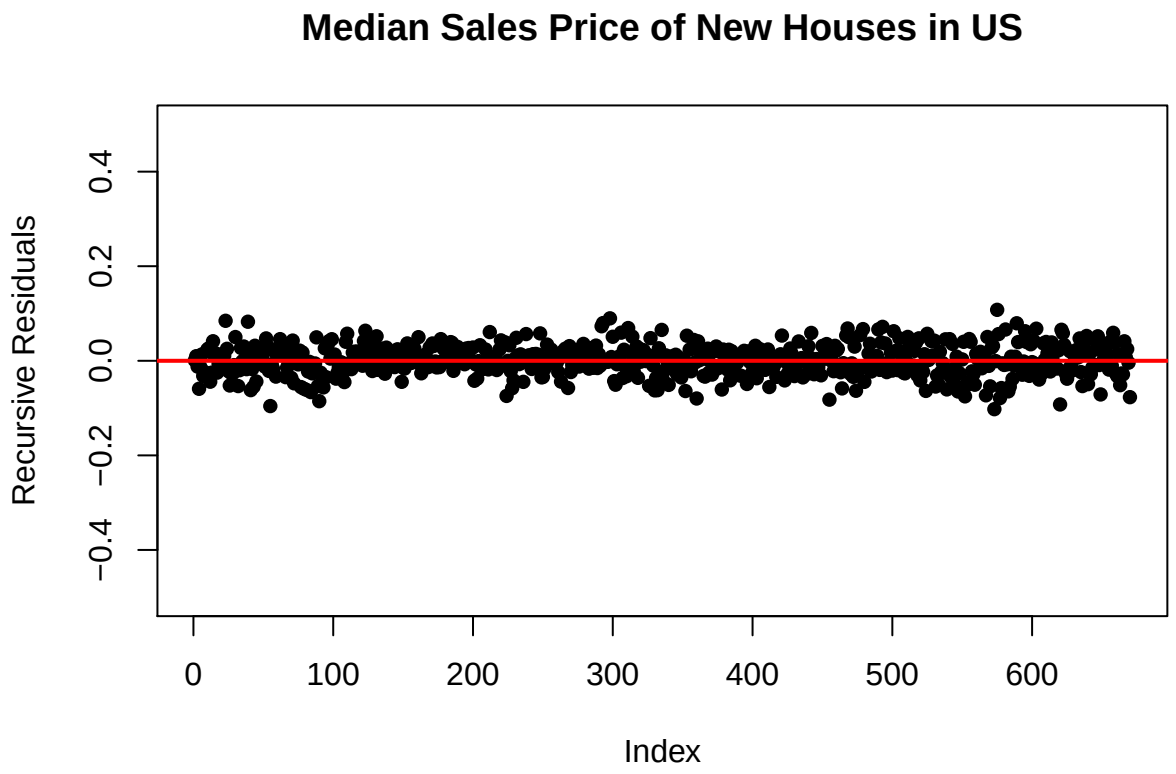
```
##
##  Recursive CUSUM test
##
## data:  u4$residuals ~ 1
## S = 0.4465, p-value = 0.7477
```

Again we can see from the figure that the CUSUM indicates that there are no structural breaks in the data. The structural change test confirms this. The null hypothesis is that there are no structural breaks. From the test, we fail reject the null, indicating that there are no structural breaks in our fit.

(g) Plot the respective Recursive Residuals and interpret the plot.

First we look at the recursive residuals for the median sales price data.

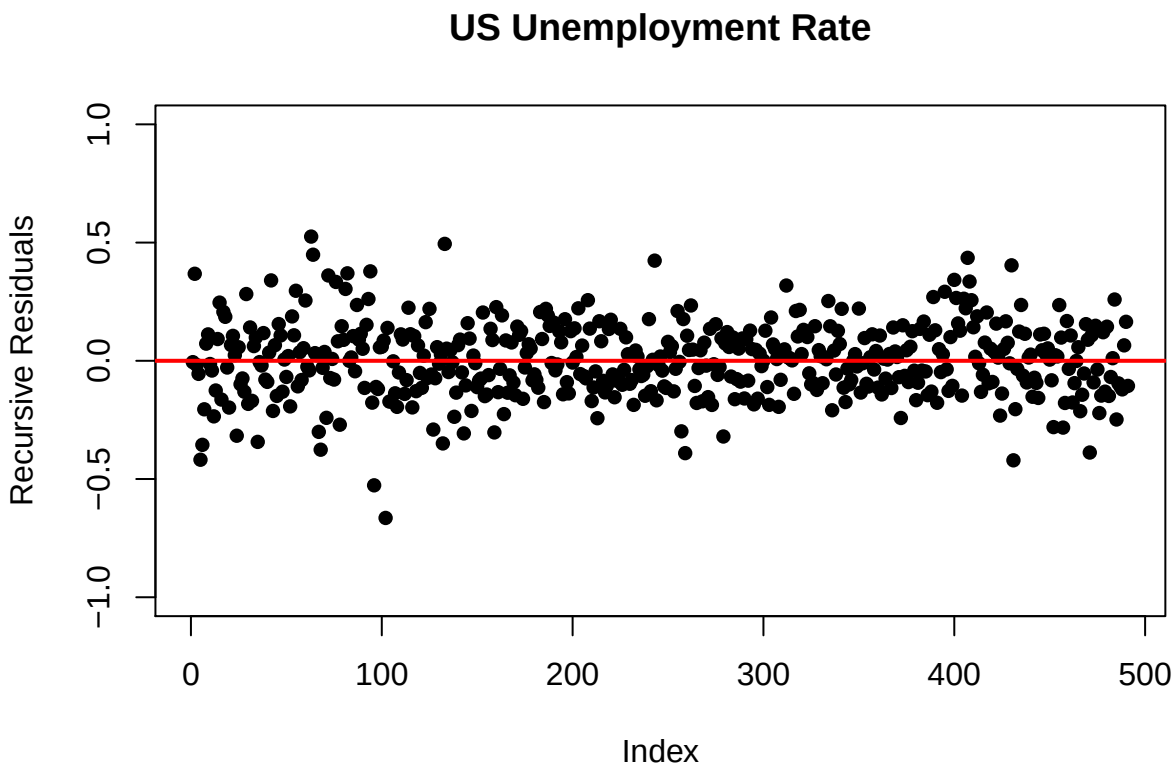
```
# mm4 = recursive residuals from m4

# calculate and plot the recursive residuals
mm4 = recresid(m4$residuals ~ 1)
plot(mm4, pch = 16, ylab = "Recursive Residuals", ylim = c(-0.5, 0.5),
     main = "Median Sales Price of New Houses in US")
abline(h = 0, col = "red", lwd = 2)
```



Recursive residuals are standardized one-step-ahead prediction errors. They are showing us if our forecasting model is stable and does not have any structural breaks. We want them to be normal with mean zero and constant variance. We can see that the residuals are randomly distributed about zero with a constant variance.

Next we look at the recursive residuals for the unemployment rate data.

```
# uu4 = recursive residuals from u4

# calculate and plot the recursive residuals
uu4 = recresid(u4$residuals ~ 1)
plot(uu4, pch = 16, ylab = "Recursive Residuals", ylim = c(-1, 1), main = "US Unemployment Rate")
abline(h = 0, col = "red", lwd = 2)
```



Again we can see that the residuals are randomly distributed about zero with a constant variance.

(h) For your model, discuss the associated diagnostic statistics.

First we look at the model we fit to the median sales price data.

```
summary(m4)
```

```
## Series: ldata.ts
## Regression with ARIMA(3,0,0)(1,0,1)[12] errors
##
## Coefficients:
##          ar1     ar2     ar3     sar1    sma1   intercept       t      t2
##       0.3879  0.4048  0.1711  -0.4360  0.4179  -2376.1693  2.3458  -6e-04
## s.e.  0.0386  0.0379  0.0386   0.6263  0.6279    105.8772  0.1060   0e+00
##
## sigma^2 estimated as 0.001035:  log likelihood=1356.69
## AIC=-2695.37   AICc=-2695.1   BIC=-2654.79
##
## Training set error measures:
##                         ME       RMSE        MAE          MPE      MAPE
## Training set -0.0003069244 0.03198032 0.02517965 -0.004115566 0.2199765
##                    MASE        ACF1
## Training set 0.3704352 -0.01394004
```

Next we look at the model we fit to the US unemployment rate data.

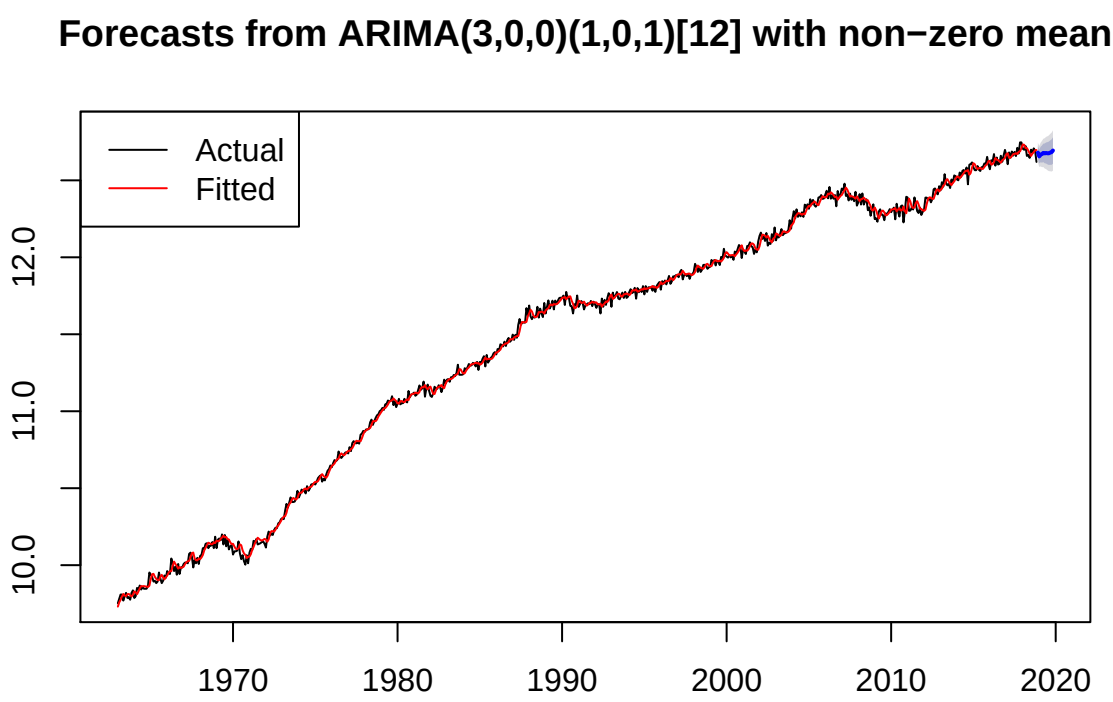
```r
summary(u4)
```

```
## Series: unrate.ts
## ARIMA(4,1,0)(1,0,1)[12]
##
## Coefficients:
##          ar1     ar2     ar3     ar4    sar1     sma1
##       0.0426  0.1641  0.1950  0.1570  0.6259  -0.8525
## s.e.  0.0447  0.0440  0.0446  0.0455  0.0783   0.0560
##
## sigma^2 estimated as 0.02503:  log likelihood=209.91
## AIC=-405.82   AICc=-405.59   BIC=-376.45
##
## Training set error measures:
##                        ME      RMSE      MAE         MPE    MAPE      MASE
## Training set -0.003901349 0.1570682 0.123227 -0.09884671 1.95289 0.1565615
##                      ACF1
## Training set -0.01870132
```

(i) Use your model to forecast 12-steps ahead. Your forecast should include the respective error bands.

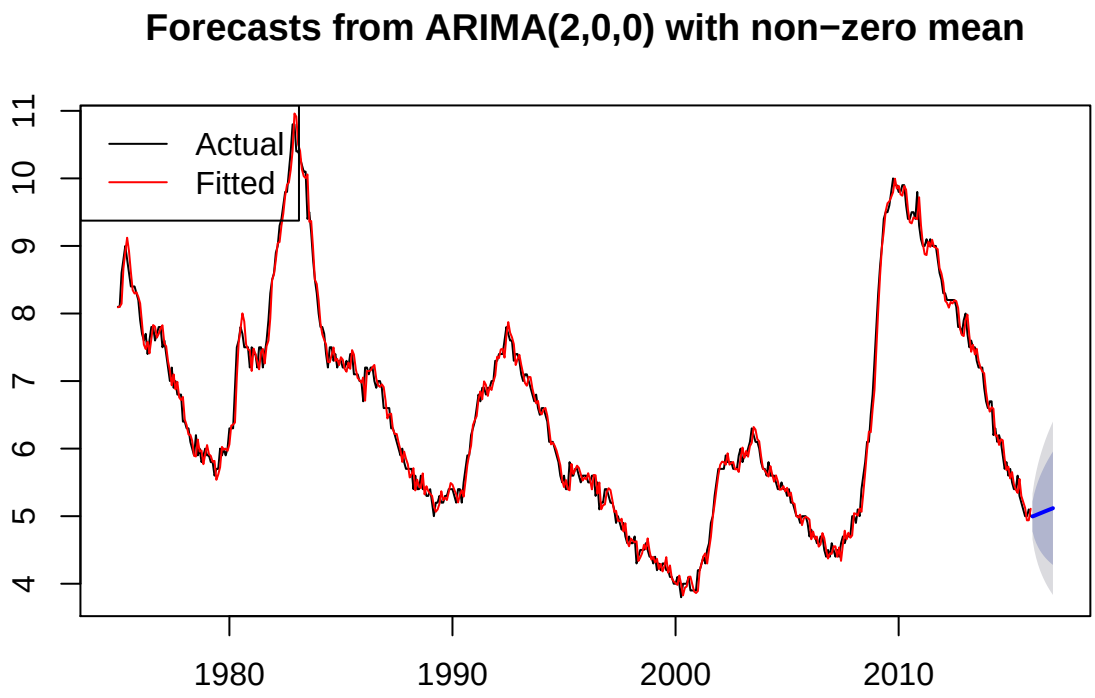
First we forecast the median sales price data.

```r
m5 = Arima(ldata.ts, order = c(3,0,0), include.drift = , seasonal = list(order = c(1,0,1)))

# forecast 12-steps ahead with m4
plot(forecast(m5, h = 12))
lines(m4$fitted, col = "red")
legend("topleft", c("Actual", "Fitted"), col = c("black", "red"), lty = c(1, 1))
```



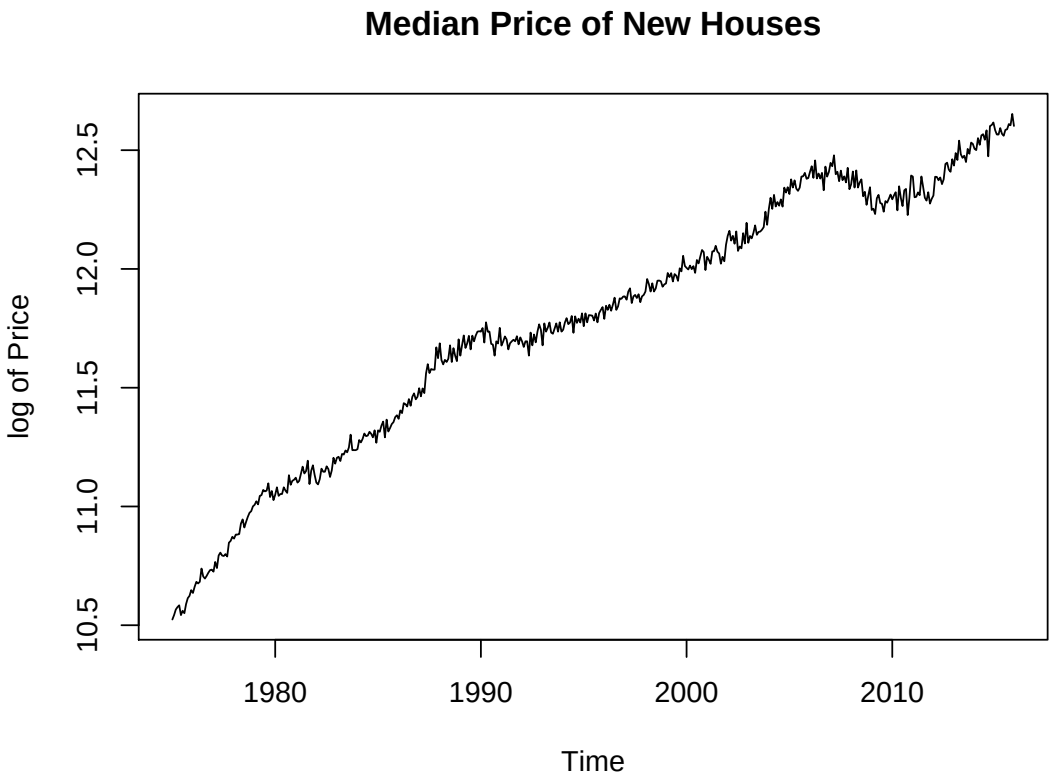
Next we forecast unemployment rate.

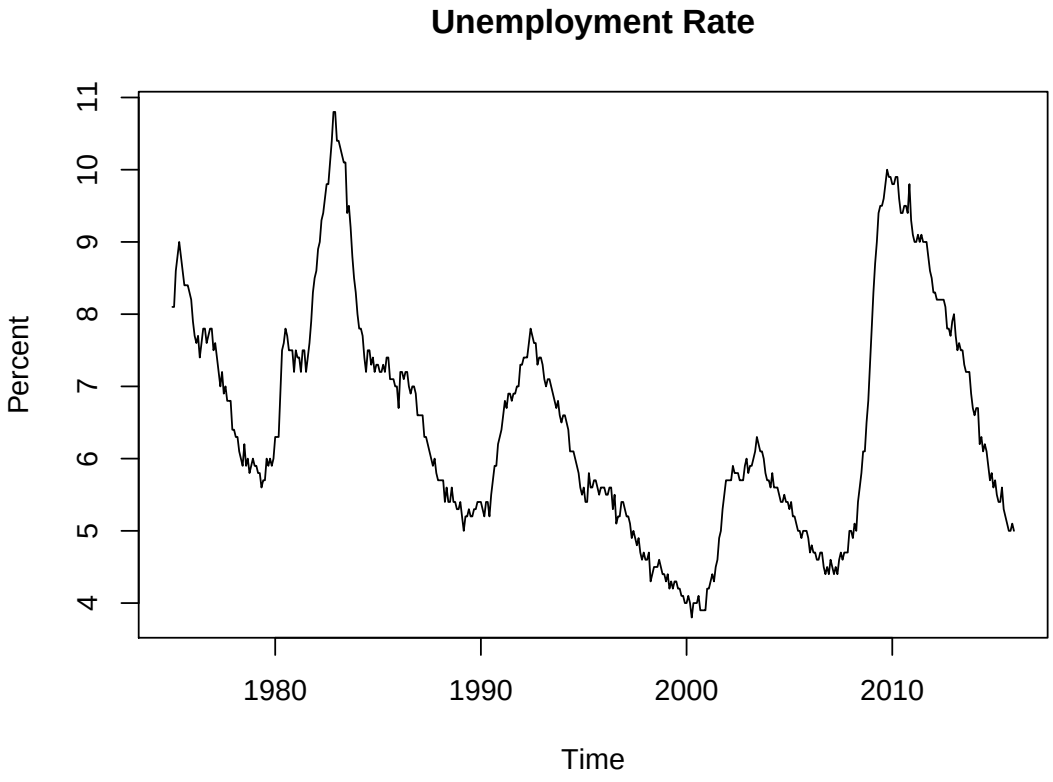
```
u5 = Arima(unrate.ts, order = c(2,0,0), include.drift = , seasonal = list(order = c(0,0,0)))

# forecast 12-steps ahead with m4
plot(forecast(u5, h = 12))
lines(u4$fitted, col = "red")
legend("topleft", c("Actual", "Fitted"), col = c("black", "red"), lty = c(1, 1))
```



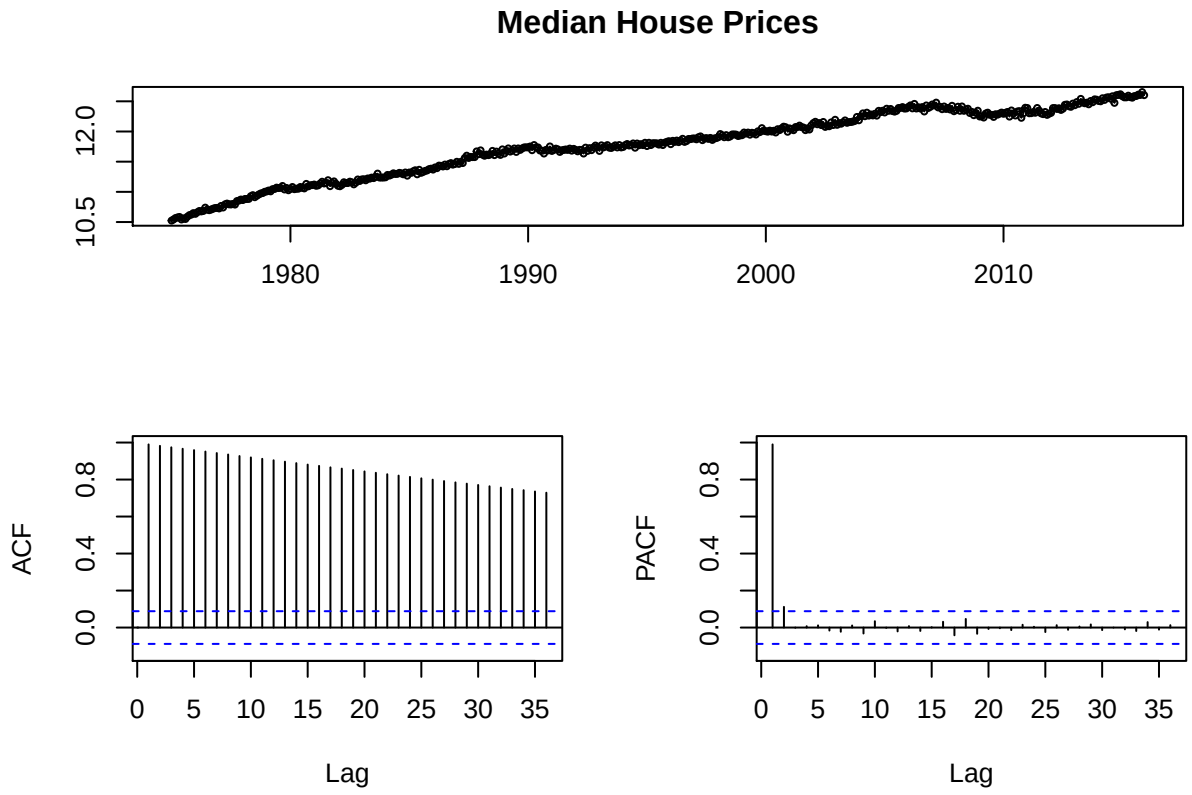
(j) Fit an appropriate VAR model using your two variables. Make sure to show the relevant plots and discuss your results from the fit.

```
# read in and clean house price data
price = read.csv("MSP FRED.csv", header = TRUE)
date = as.Date(price$DATE, format = "%Y-%m-%d")
price = cbind(date, price)
price = price[, c(1,3)]
price = subset(price, price$date >= "1975-01-01" & price$date <= "2015-12-01")

# read in and clean unrate data
unrate = read.csv("UNRATE FRED.csv", header = TRUE)
date = as.Date(unrate$DATE, format = "%Y-%m-%d")
unrate = cbind(date, unrate)
unrate = unrate[, c(1,3)]
unrate = subset(unrate, unrate$date >= "1975-01-01" & unrate$date <= "2015-12-01")

# create ts object for each series
price.ts = ts(log(price$MSPNHSUS), start = 1975, frequency = 12)
unrate.ts = ts(unrate$UNRATE, start = 1975, frequency = 12)

par(mfrow=c(2,1))
plot(price.ts, ylab = "log of Price", main = "Median Price of New Houses")
plot(unrate.ts, ylab = "Percent", main = "Unemployment Rate")
```

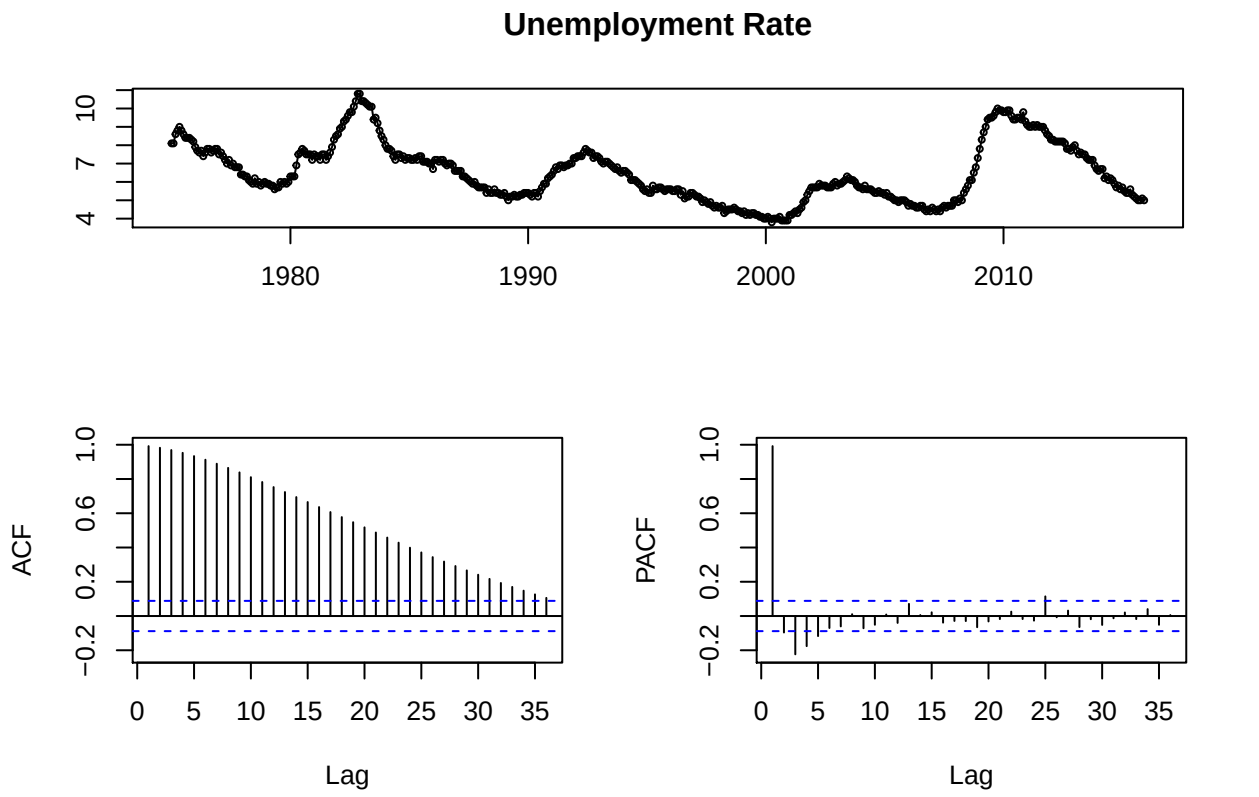

Median Price of New Houses
log of Price
10.5
11.0
11.5
12.0
12.5
1980
1990
2000
2010
Time


Unemployment Rate
Percent
4
5
6
7
8
9
10
11
1980
1990
2000
2010
Time

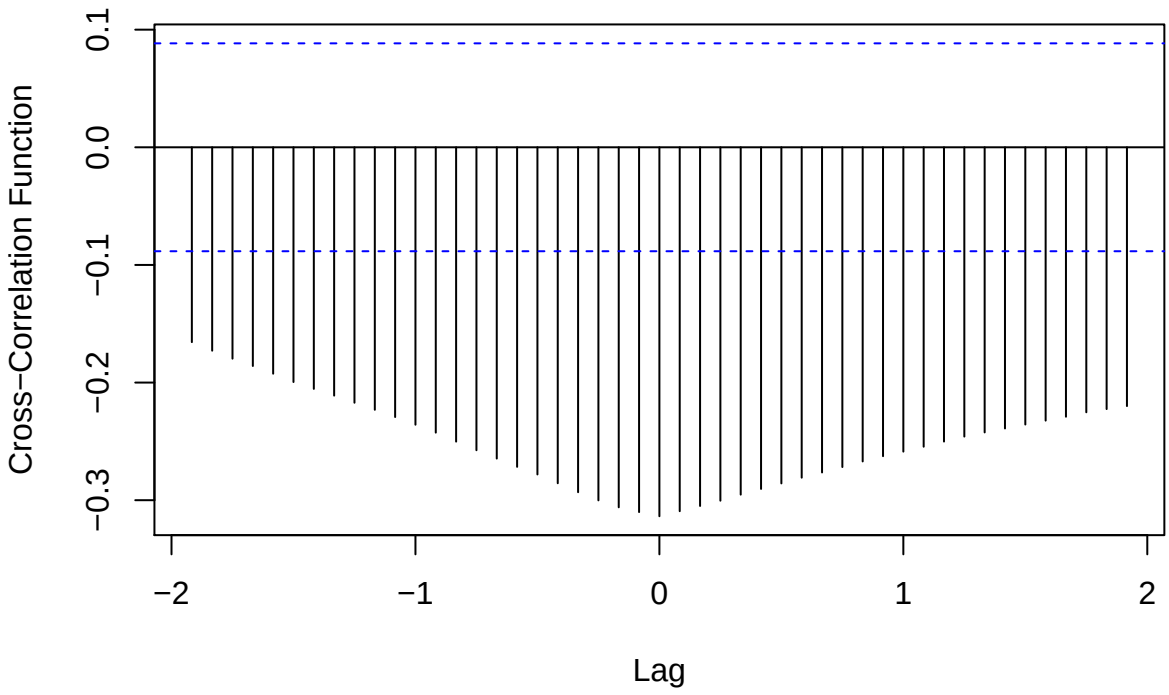
```r
# Look at the ACF, PACF, and CCF (cros-correlation function)

tsdisplay(price.ts,main="Median House Prices")
```




```r
tsdisplay(unrate.ts,main="Unemployment Rate")
```




```
ccf(price.ts,unrate.ts,ylab="Cross-Correlation Function",
    main = "Median House Prices and Unemployment rate CCF")
```



From the cross-correlation figure it appears that the peak negative correlation between median sales price and unemployment rate is at lag zero.

Next we fit a VAR model using our two time series. To determine the order of (p) in our VAR model, we use the following algorithm:
A) Start with p=1, compute VAR(1)
B) Continue with p=2, and keep the one with lowest AIC
C) Continue with p= 3, 4, etc.
D) After a certain value of p, AIC & BIC will worsen, therefore, decide on the model with lowest overall AIC and BIC.

We create a loop to fit a VAR model to the data with p = 1,...10. Then we compare the AIC and BIC of all ten fits to determine the appropriate order of p in the VAR model.

```
# Fit a VAR(p) model to the data
y=cbind(price.ts, unrate.ts)
y.tot=data.frame(y)

order = c(1:10)
AIC = c()
BIC = c()

for(i in 1:10) {

  y.model = VAR(y.tot, p = i)
  AIC[i] = -2*logLik(y.model)[1] +
    2*(length(y.model$varresult$price.ts$coefficients) + length(y.model$varresult$unrate.ts$coefficients
```

```
  BIC[i] = -2*logLik(y.model)[1] +
  (length(y.model$varresult$price.ts$coefficients) + length(y.model$varresult$unrate.ts$coefficients))*l

}

order = cbind(order, AIC, BIC)
order
```

```
##       order       AIC       BIC
##  [1,]     1 -2128.517 -2103.326
##  [2,]     2 -2284.560 -2242.575
##  [3,]     3 -2330.919 -2272.140
##  [4,]     4 -2354.830 -2279.257
##  [5,]     5 -2364.055 -2271.688
##  [6,]     6 -2380.484 -2271.323
##  [7,]     7 -2377.164 -2251.209
##  [8,]     8 -2377.436 -2234.687
##  [9,]     9 -2367.441 -2207.899
## [10,]    10 -2353.718 -2177.381
```

When comparing the AIC and BIC, we see that AIC suggests to use a VAR(6) model while BIC suggests to use a VAR(4) model. Because AIC is biased towards overparameterized models, it appears that the most appropriate model is to use a VAR(4) model.

```
varmodel = VAR(y.tot, p = 4)
summary(varmodel)
```

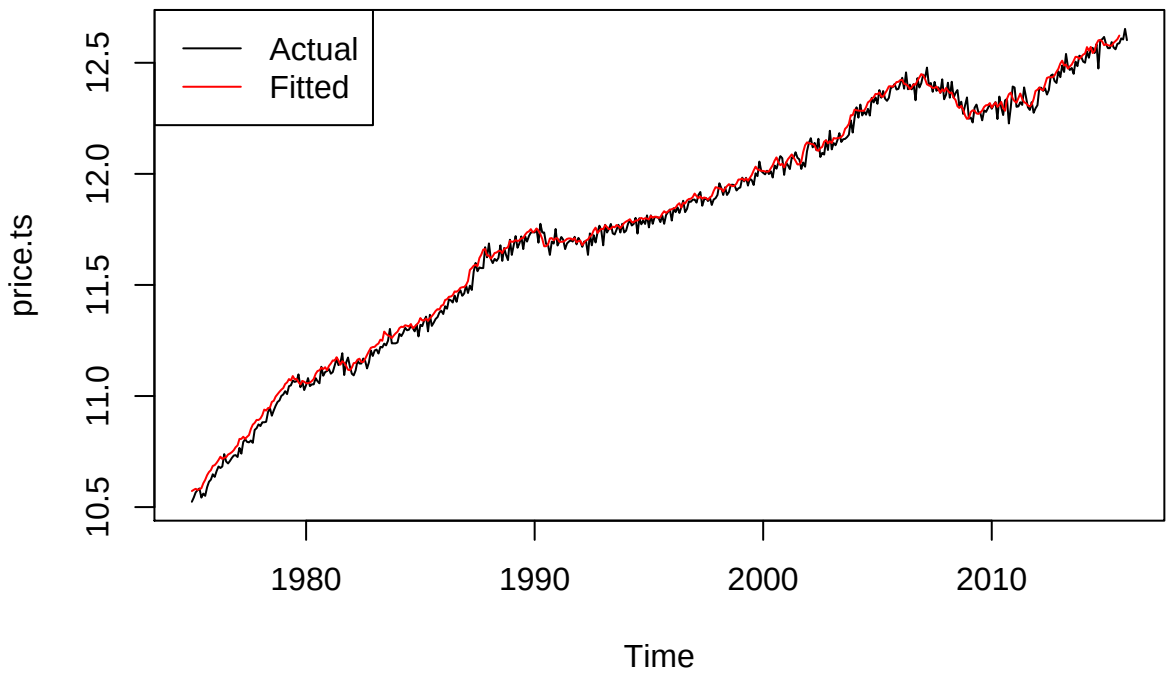
```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: price.ts, unrate.ts
## Deterministic variables: const
## Sample size: 488
## Log Likelihood: 1195.415
## Roots of the characteristic polynomial:
## 0.9971 0.9771 0.7487 0.5837 0.5607 0.5607 0.5036 0.5036
## Call:
## VAR(y = y.tot, p = 4)
##
##
## Estimation results for equation price.ts:
## =========================================
## price.ts = price.ts.l1 + unrate.ts.l1 + price.ts.l2 + unrate.ts.l2 + price.ts.l3 + unrate.ts.l3 + pri
##
##               Estimate Std. Error t value Pr(>|t|)
## price.ts.l1   0.330462   0.045213   7.309 1.14e-12 ***
## unrate.ts.l1 -0.014611   0.008627  -1.694 0.090989 .
## price.ts.l2   0.367646   0.047191   7.791 4.17e-14 ***
## unrate.ts.l2  0.011903   0.012544   0.949 0.343150
## price.ts.l3   0.166999   0.047073   3.548 0.000427 ***
## unrate.ts.l3 -0.013790   0.012493  -1.104 0.270224
## price.ts.l4   0.128834   0.045279   2.845 0.004626 **
## unrate.ts.l4  0.016866   0.008514   1.981 0.048178 *
## const         0.077553   0.035367   2.193 0.028803 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0316 on 479 degrees of freedom
## Multiple R-Squared: 0.9966,  Adjusted R-squared: 0.9966
## F-statistic: 1.762e+04 on 8 and 479 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation unrate.ts:
## ==========================================
## unrate.ts = price.ts.l1 + unrate.ts.l1 + price.ts.l2 + unrate.ts.l2 + price.ts.l3 + unrate.ts.l3 + p
##
##               Estimate Std. Error t value Pr(>|t|)
## price.ts.l1  -0.483708   0.233490  -2.072   0.0388 *
## unrate.ts.l1  1.030476   0.044552  23.130  < 2e-16 ***
## price.ts.l2  -0.235013   0.243706  -0.964   0.3354
## unrate.ts.l2  0.152785   0.064782   2.358   0.0188 *
## price.ts.l3   0.225343   0.243096   0.927   0.3544
## unrate.ts.l3  0.005042   0.064518   0.078   0.9377
## price.ts.l4   0.487652   0.233831   2.085   0.0376 *
## unrate.ts.l4 -0.199660   0.043971  -4.541  7.1e-06 ***
## const         0.143306   0.182647   0.785   0.4331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.1632 on 479 degrees of freedom
## Multiple R-Squared: 0.9894,  Adjusted R-squared: 0.9892
## F-statistic:  5580 on 8 and 479 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##             price.ts unrate.ts
## price.ts   0.0009985 -0.000272
## unrate.ts -0.0002720  0.026630
##
## Correlation matrix of residuals:
##           price.ts unrate.ts
## price.ts   1.00000  -0.05276
## unrate.ts -0.05276   1.00000
```

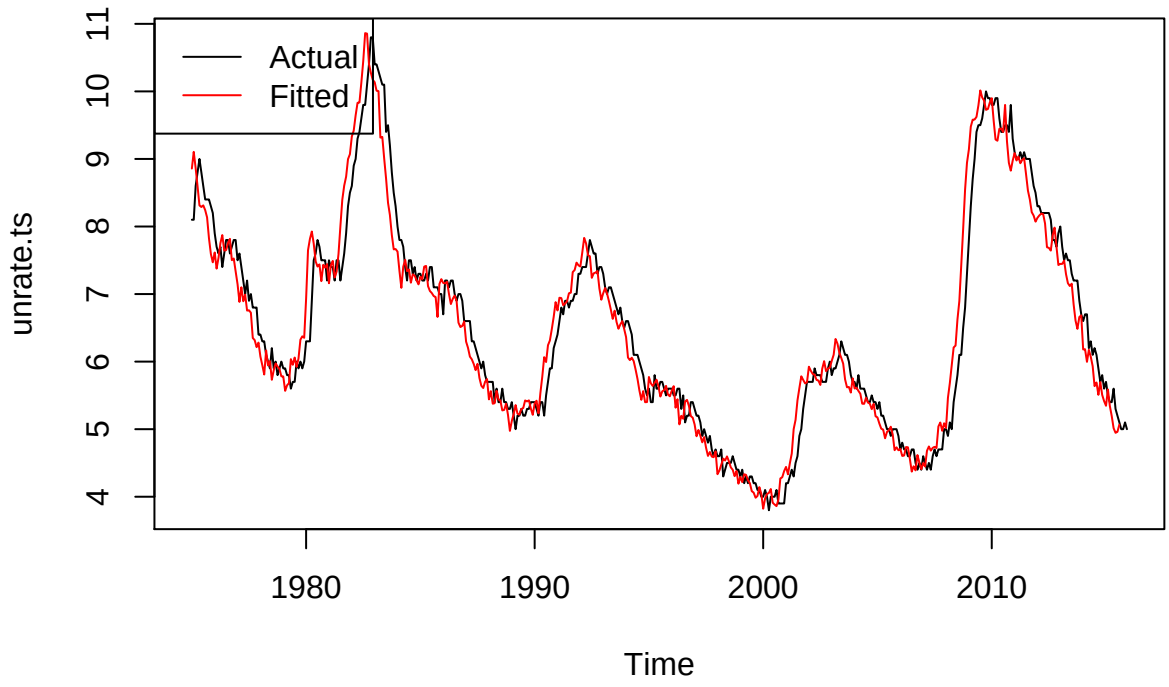
From the summary output, we can see that lags one, two, three, and four of sales price are significant when regressed on sales price. Additionally, lag four of unemployment rate is significant when regressed on sales price. From the summary output, we can see that lags one, two, and four of unemployment rate are significant when regressed on unemployment rate. Additionally, lags one and four of sales price is significant when regressed on unemployment rate. Based on these results, it appears that US unemployment rate has some explanatory power in predicting medians sales price of new houses in the US.

```
# plot of actual and fitted values for Median House Price
fit.price = fitted(varmodel)[,1]
fit.price.ts = ts(fit.price, start = 1975, frequency = 12)

plot(price.ts)
lines(fit.price.ts, col = "red")
```

```
legend("topleft", c("Actual", "Fitted"), lty = c(1,1), col = c("black", "red"))
```

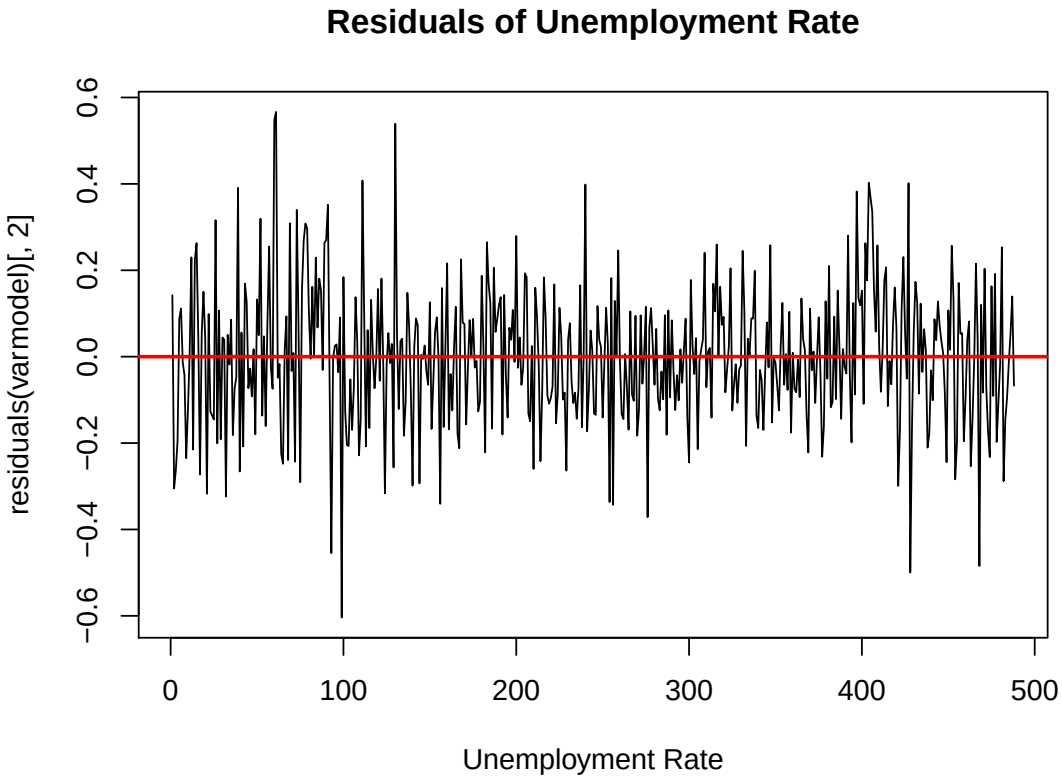



```
# plot of actual and fitted values for Unemployment Rate
fit.unrate = fitted(varmodel)[,2]
fit.unrate.ts = ts(fit.unrate, start = 1975, frequency = 12)

plot(unrate.ts)
lines(fit.unrate.ts, col = "red")
legend("topleft", c("Actual", "Fitted"), lty = c(1,1), col = c("black", "red"))
```

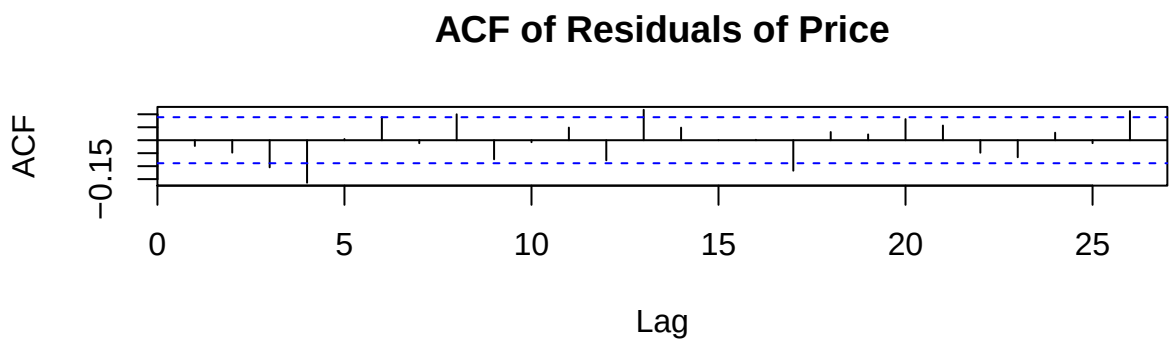


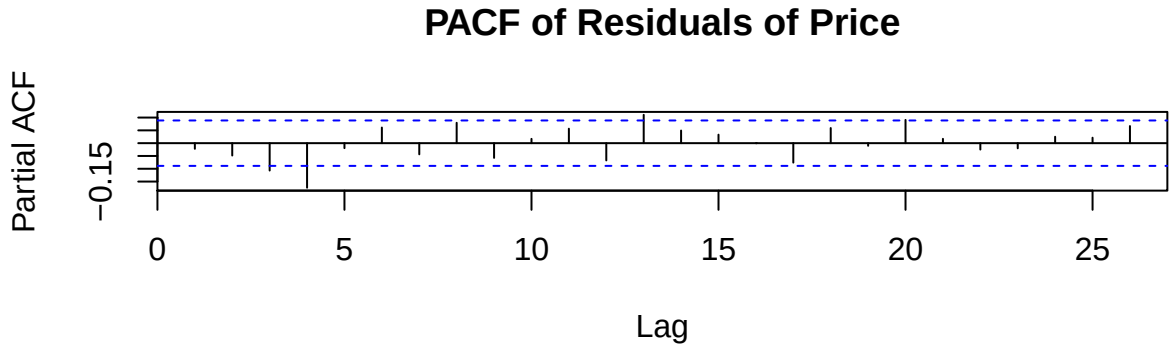

```
# plot residuals from VAR fit
par(mfrow=c(2,1))

plot(residuals(varmodel)[,1], type = "l", xlab = "Price", main = "Residuals of Median House Price")
abline(h = 0, col = "red", lwd = 2)

plot(residuals(varmodel)[,2], type = "l", xlab = "Unemployment Rate",
     main = "Residuals of Unemployment Rate")
abline(h = 0, col = "red", lwd = 2)
```


Residuals of Median House Price
residuals(varmodel)[, 1]
0.10
0.05
0.00
−0.05
−0.10
0
100
200
300
400
500
Price


Residuals of Unemployment Rate
residuals(varmodel)[, 2]
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
100
200
300
400
500
Unemployment Rate

```r
# ACF and PACf of residuals of price
par(mfrow=c(2,1))
acf(residuals(varmodel)[,1], main = "ACF of Residuals of Price")
pacf(residuals(varmodel)[,1], main = "PACF of Residuals of Price")
```

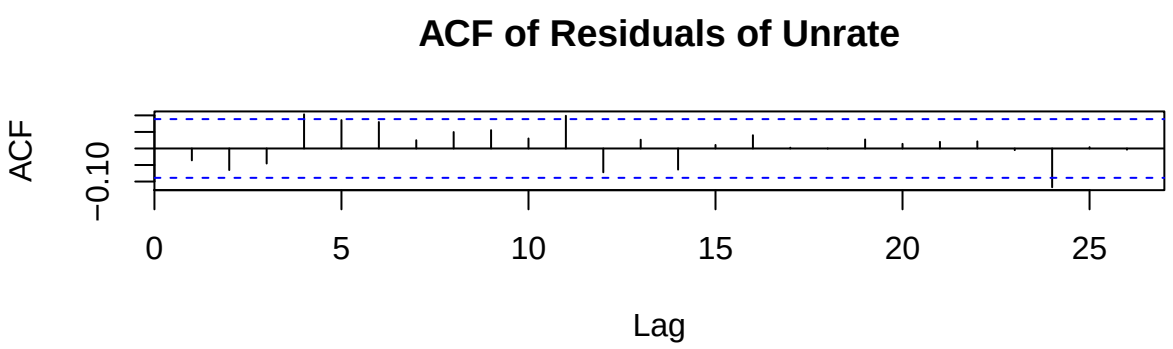


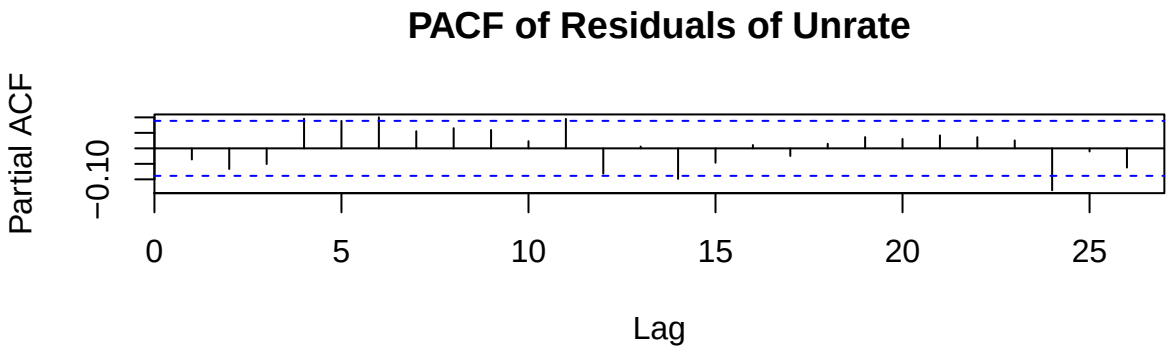



```r
Box.test(residuals(varmodel)[,1])
```

```
##
##  Box-Pierce test
##
## data:  residuals(varmodel)[, 1]
## X-squared = 0.23809, df = 1, p-value = 0.6256
```

```r
# ACF and PACF of residuals of unrate
par(mfrow=c(2,1))
acf(residuals(varmodel)[,2], main = "ACF of Residuals of Unrate")
pacf(residuals(varmodel)[,2], main = "PACF of Residuals of Unrate")
```



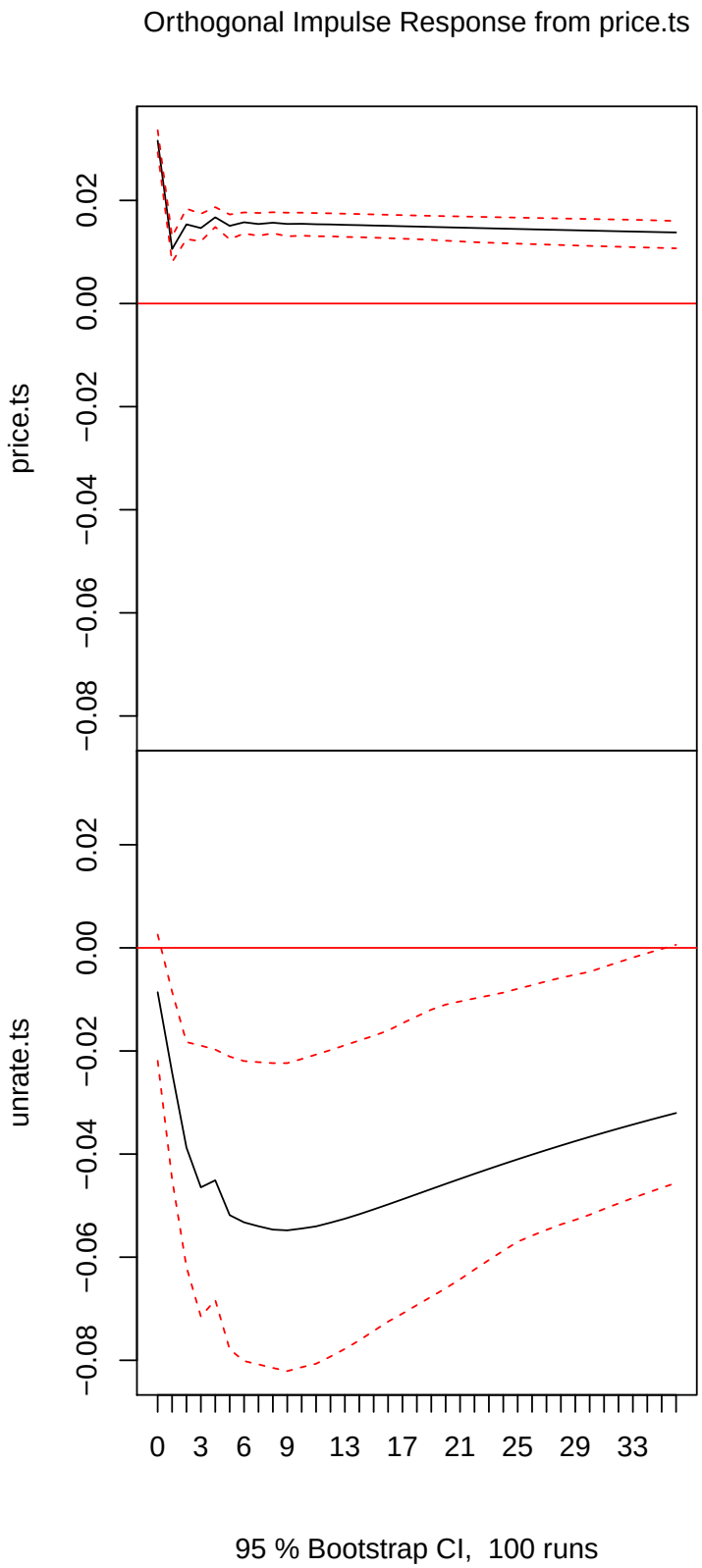



```
Box.test(residuals(varmodel)[,2])
```

```
##
##  Box-Pierce test
##
## data:  residuals(varmodel)[, 2]
## X-squared = 0.62859, df = 1, p-value = 0.4279
```

(k) Compute, plot, and interpret the respective impulse response functions.

```
# Impulse Response Function
plot(irf(varmodel, n.ahead=36))
```


Orthogonal Impulse Response from price.ts
price.ts
unrate.ts
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0 3 6 9 13 17 21 25 29 33
95 % Bootstrap CI, 100 runs

The first figure shows the own variable response and cross variable response from a shock in price. The top graph in this figure indicates that the effect of price's shock on subsequent prices has a small initial effect and then decays very slowly. The bottom graph in this figure indicates that the effect of prices's shock on subsequent unemployment rates has a slow build up to a negative shock that peaks around 10 months.

The second figure shows the cross variable response and the own variable response from a shock in unemployment rate. The top graph in this figure indicates that their is no effect of unemployments shock on price. The bottom graph in this figure indicates that the effect of unemployments shock on subsequent unemployment is positive and builds up to a peak at 12 months and slowly decays.

The results of the impulse response function support our conclution about the unemployment rates ability to help predict the median sales price of new houses in the US.

(l) Perform a Granger-Causality test on your variables and discuss your results from the test.

```
# does median house price granger-cause unemployment
grangertest(unrate.ts ~ price.ts, order = 4)
```

```
## Granger causality test
##
## Model 1: unrate.ts ~ Lags(unrate.ts, 1:4) + Lags(price.ts, 1:4)
## Model 2: unrate.ts ~ Lags(unrate.ts, 1:4)
##   Res.Df Df      F Pr(>F)
## 1    479
## 2    483 -4 2.0568 0.0854 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The null hypothesis for the granger test is that the variable does not granger-cause the other varaible. The p-value is greater than 0.05, indicating that we fail to reject the null hypothesis. So median sales price of new houses in the US does not granger-cause unemployment.

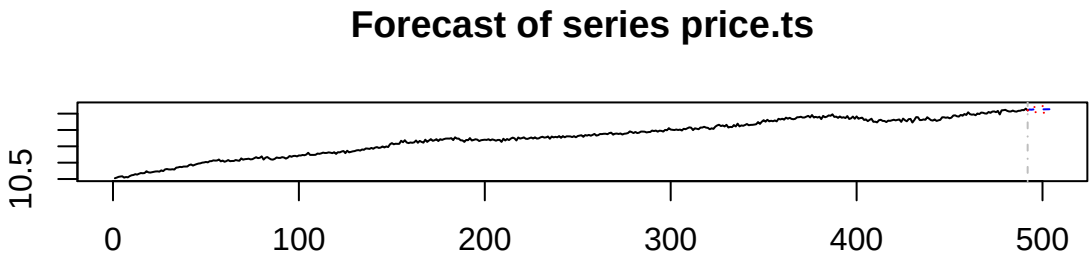
```
# does unemployment rate granger-cause median house price
grangertest(price.ts ~ unrate.ts, order = 4)
```

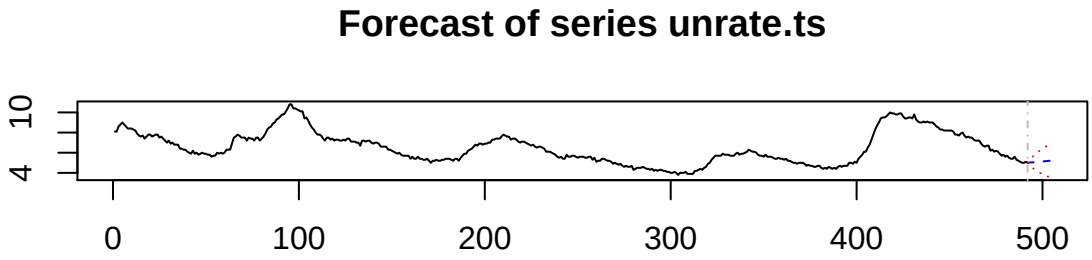
```
## Granger causality test
##
## Model 1: price.ts ~ Lags(price.ts, 1:4) + Lags(unrate.ts, 1:4)
## Model 2: price.ts ~ Lags(price.ts, 1:4)
##   Res.Df Df      F  Pr(>F)
## 1    479
## 2    483 -4 2.3762 0.05118 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value is greater than 0.05, indicating that we fail to reject the null hypothesis. So the US unemployment rate does not granger-cause median sales price of new houses in the US. However, the p-value is very close to 0.05.

(m) Use your VAR model to forecast 12-steps ahead. Your forecast should include the respective error bands. Comment on the differences between the two forecasts (VAR vs. ARMA).

```
var.predict = predict(object=varmodel, n.ahead=12)
plot(var.predict)
```





The forecast of median sales price appears to be very similar for both the VAR and the ARMA model that we fit. The confidence and prediction intervals appear to be very similar in size. However, because it appears from the granger-causality test that the unemployment rate does not granger-cause median sales price, we would choose the ARMA model over the VAR model when forecasting.

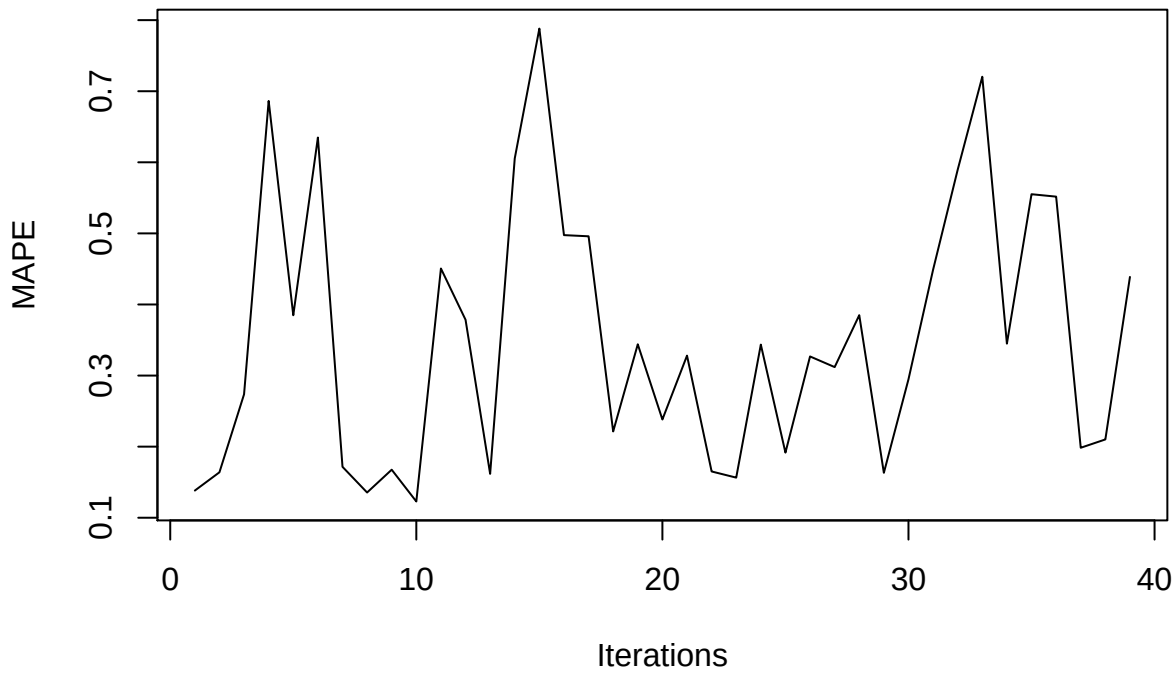
However, the confidence and prediction intervals are larger when using the ARMA model to forecast the US unemployment rate compared to the VAR model. This is an interesting result because we also found from the granger-causality test that the median sales price does not granger-cause the unemployment rate so we would have expected the VAR model to perform worse than the ARMA model.

(n) Backtest your ARMA model. Begin by partitioning your data set into an estimation set and a prediction set.

(a) Use a recursive backtesting scheme, and forecast 12-steps ahead at each iteration. Compute the mean absolute percentage error at each step. Provide a plot showing the MAPE over each iteration.

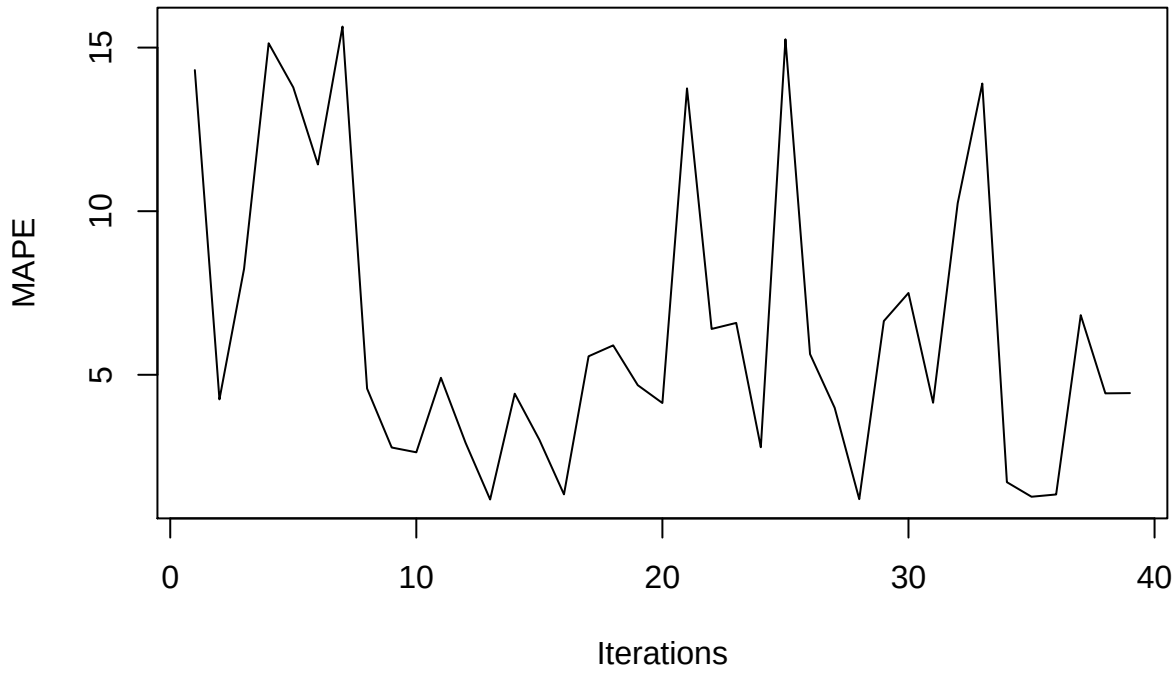
```
# sales price data

c = 1
MAPE =c() #create empty vector to store MAPE
MAPE = as.data.frame(MAPE)

#loop
for (i in 39:1){

  estindex = i*12
```

```
  splitdata = splitTrainTest(price.ts,length(price.ts)-estindex) #split data into training and testing
  estset = splitdata$train #create estimation set
  predset = splitdata$test #create prediction set

  #fit model to estimation set
  fit <- Arima(estset,order=c(3,0,0), include.drift = TRUE, seasonal=list(order=c(1,0,1)))
  fc <- forecast(fit,h=12) #forecast ahead 12 steps
  j= accuracy(fc,predset) #find MAPE

  #store MAPE
  MAPE[c,1]=i
  MAPE[c,2]=j[2,5]

  c=c+1



}

plot(1:39,MAPE$V2,type='l',xlab="Iterations",ylab="MAPE") #plot
```



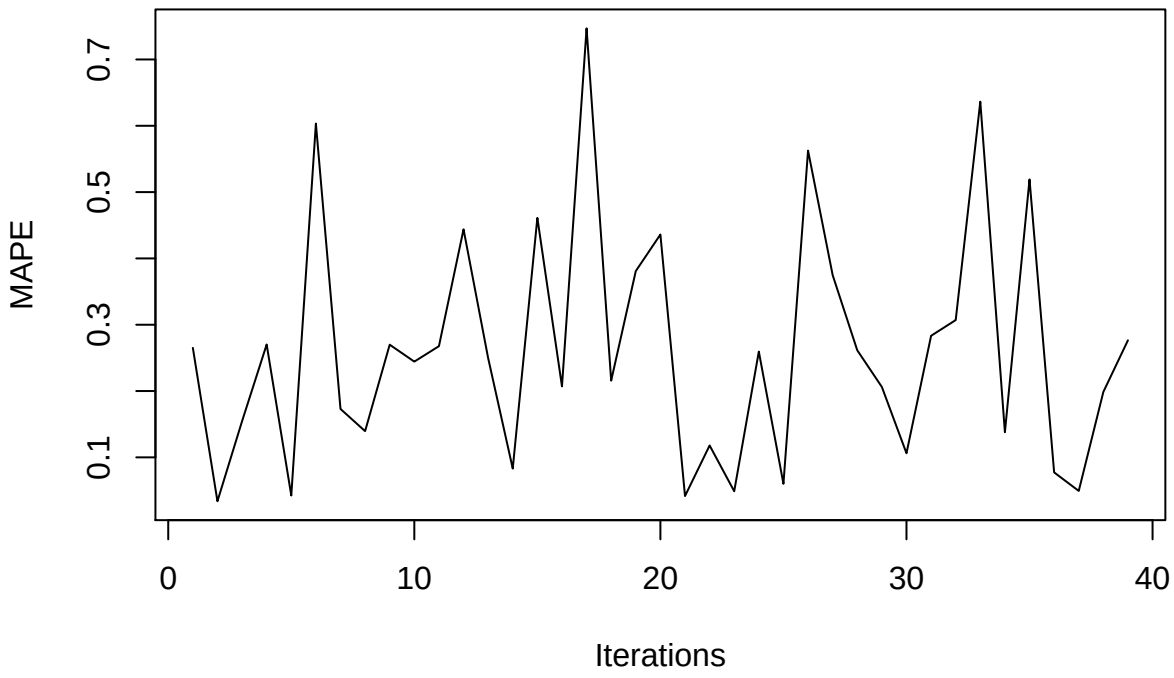

```
# unemployment rate data
c = 1
UMAPE =c() #create empty vector to store MAPE
UMAPE = as.data.frame(UMAPE)
```

```r
#loop
for (i in 39:1){

  estindex = i*12
  splitdata = splitTrainTest(unrate.ts,length(unrate.ts)-estindex) #split data into training and testin
  estset = splitdata$train #create estimation set
  predset = splitdata$test #create prediction set

  #fit model to estimation set
  fit <- Arima(estset,order=c(4,1,0), include.drift = FALSE, seasonal=list(order=c(1,0,1)))
  fc <- forecast(fit,h=12) #forecast ahead 12 steps
  j= accuracy(fc,predset) #find MAPE

  #store MAPE
  UMAPE[c,1]=i
  UMAPE[c,2]=j[2,5]

  c=c+1



}

plot(1:39,UMAPE$V2,type='l',xlab="Iterations",ylab="MAPE") #plot
```

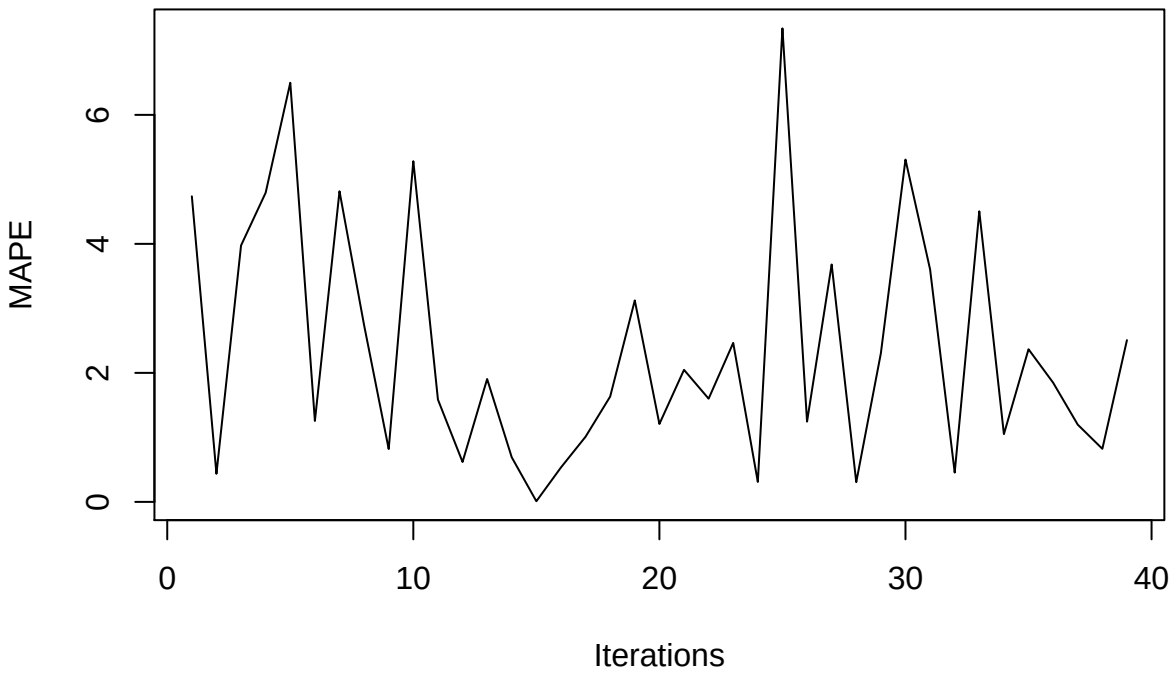


(b) Shorten your forecast horizon to only 1-step ahead. Compute the absolute percentage error at each iteration, and plot.

```
# sales price data

c = 1
MAPE1 =c()
MAPE1 = as.data.frame(MAPE1)
for (i in 39:1){

  estindex = i*12
  splitdata = splitTrainTest(price.ts,length(price.ts)-estindex)
  estset = splitdata$train
  predset = splitdata$test

  fit <- Arima(estset,order=c(3,0,0), include.drift = TRUE, seasonal=list(order=c(1,0,1)))
  fc <- forecast(fit,h=1) #forecast 1 step ahead
  j= accuracy(fc,predset)


  MAPE1[c,1]=i
  MAPE1[c,2]=j[2,5]

  c=c+1

}

plot(1:39,MAPE1$V2,type='l',xlab="Iterations",ylab="MAPE")
```




```
# unemployment rate data

c = 1
UMAPE1 =c()
UMAPE1 = as.data.frame(UMAPE1)
for (i in 39:1){

  estindex = i*12
  splitdata = splitTrainTest(unrate.ts,length(unrate.ts)-estindex)
  estset = splitdata$train
  predset = splitdata$test

  fit <- Arima(estset,order=c(4,1,0), include.drift = FALSE, seasonal=list(order=c(1,0,1)))
  fc <- forecast(fit,h=1) #forecast 1 step ahead
  j= accuracy(fc,predset)


  UMAPE1[c,1]=i
  UMAPE1[c,2]=j[2,5]

  c=c+1

}

plot(1:39,UMAPE1$V2,type='l',xlab="Iterations",ylab="MAPE")
```



(c) Based on your findings above, does your model perform better at longer or shorter horizon forecasts?

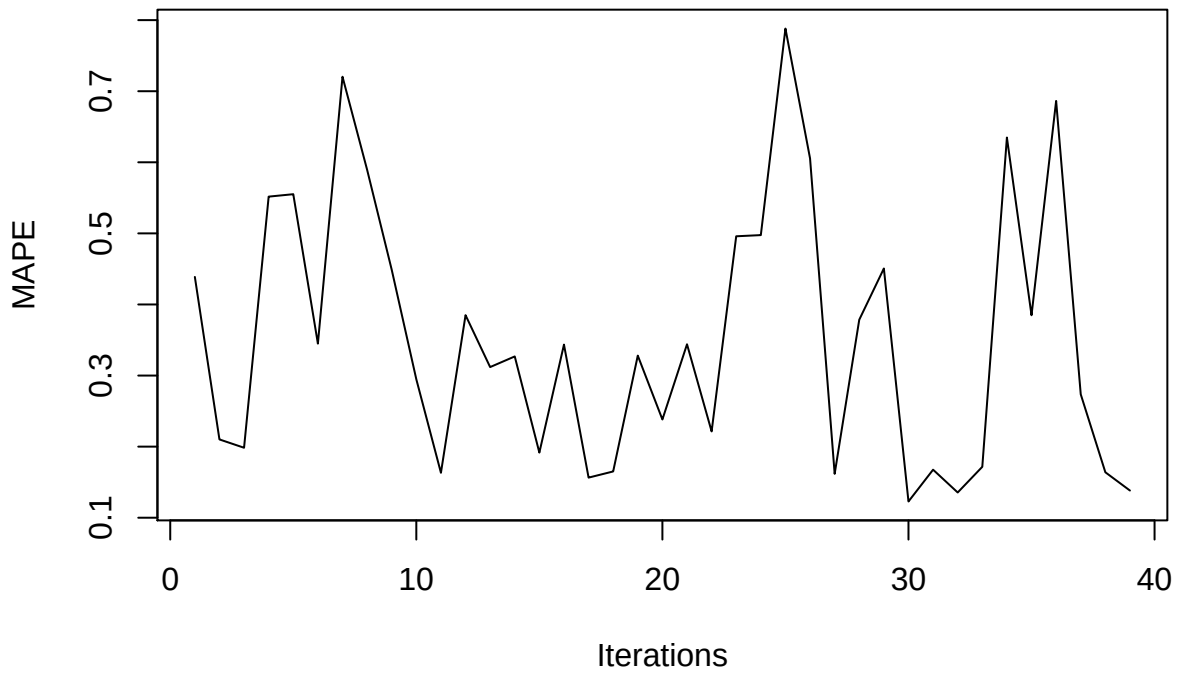
We can see that the range of MAPE is smaller when the forecasst horizon is shorter therefore our model performs better at longer horizon forecasts.

(d) Now test your model using a moving window backtesting scheme. Forecast out 12-steps ahead at each iteration, and plot the forecast errors observed at each iteration. Repeat for a 1-step ahead forecast horizon. Provide plots of both.

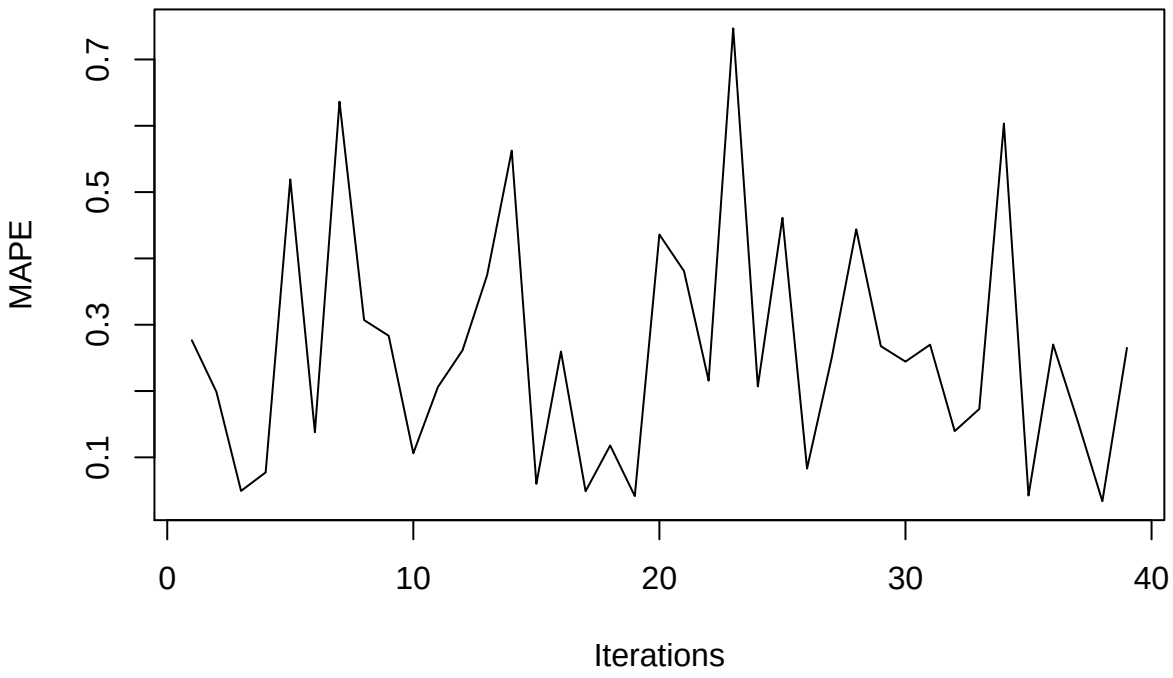
```
# sales price data

#initialize parameters
c = 1
MAPE =c()
MAPE = as.data.frame(MAPE)
k=13
newdata = price.ts

for (i in 1:39){


  estindex = length(newdata)-12
  splitdata = splitTrainTest(price.ts,estindex) #split data into training and testing
  estset = splitdata$train
  predset = splitdata$test
```

```
  #fit model
  fit <- Arima(estset,order=c(3,0,0), include.drift = TRUE, seasonal=list(order=c(1,0,1)))
  fc <- forecast(fit,h=12)#forecast 12 steps ahead
  j= accuracy(fc,predset)

  #store MAPE
  MAPE[c,1]=i
  MAPE[c,2]=j[2,5]

  c=c+1

#subset new data for new window
  newdata = subset(newdata,start=k,end=492) #setup data for next window
  k=k+12


}




c = 1
MAPE1 =c()
MAPE1 = as.data.frame(MAPE1)
k=13
newdata = price.ts

for (i in 1:39){
  #dataset
  #make first 12 estimation

  estindex = length(newdata)-12
  splitdata = splitTrainTest(price.ts,estindex)
  estset = splitdata$train
  predset = splitdata$test



  fit <- Arima(estset,order=c(3,0,0), include.drift = TRUE, seasonal=list(order=c(1,0,1)))
  fc <- forecast(fit,h=1) #forecast 1 step ahead
  j= accuracy(fc,predset)


  MAPE1[c,1]=i
  MAPE1[c,2]=j[2,5]

  c=c+1


  newdata = subset(newdata,start=k,end=492)
  k=k+12
```

```
}


par(mfrow=c(1,1))

#plots
plot(1:39,MAPE$V2,type='l',xlab="Iterations",ylab="MAPE")
```

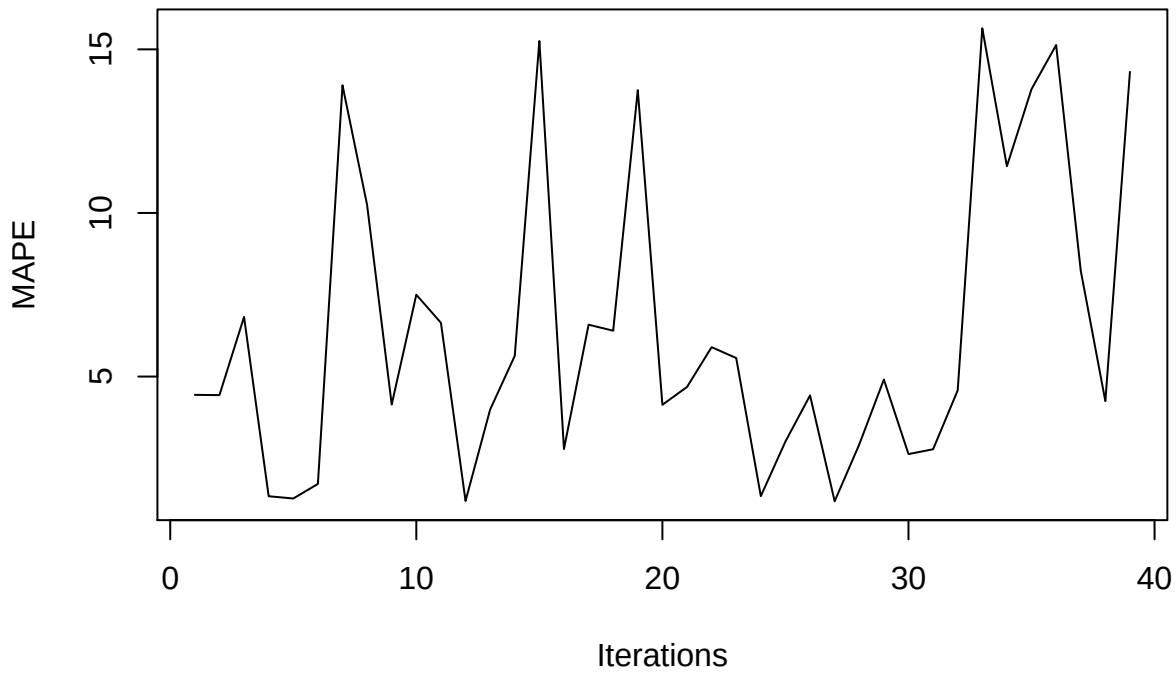



```
plot(1:39,MAPE1$V2,type='l',xlab="Iterations",ylab="MAPE")
```

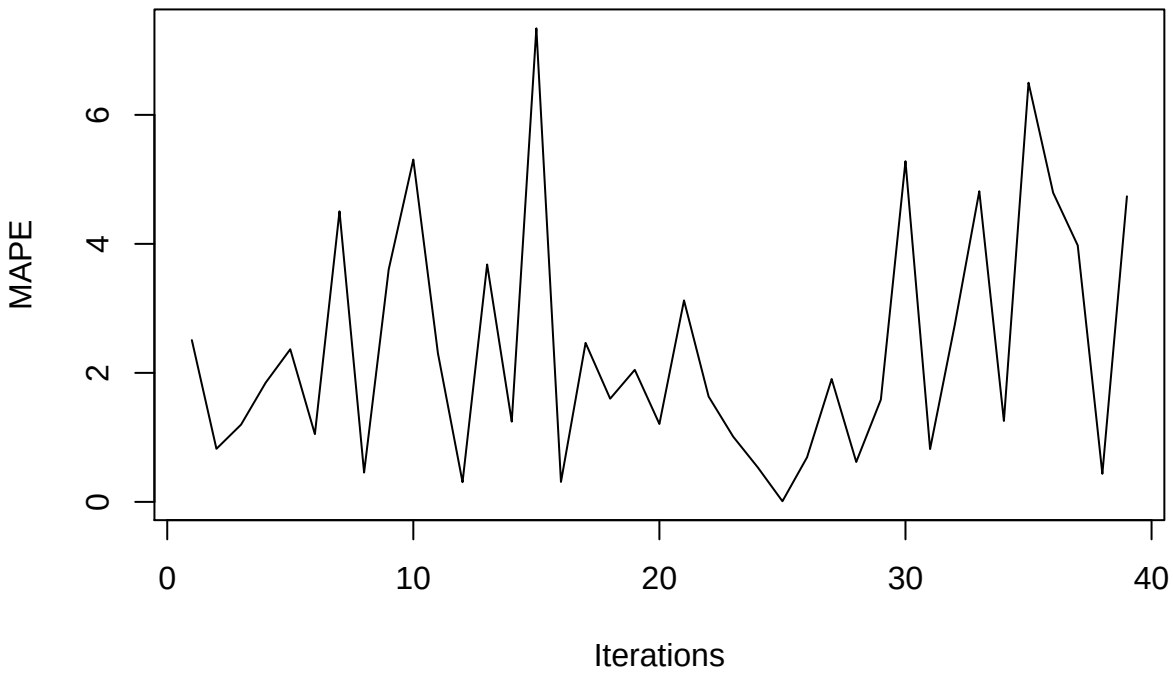



```
# unemployment rate data

#initialize parameters
c = 1
UMAPE =c()
UMAPE = as.data.frame(UMAPE)
k=13
newdata = unrate.ts

for (i in 1:39){


  estindex = length(newdata)-12
  splitdata = splitTrainTest(unrate.ts,estindex) #split data into training and testing
  estset = splitdata$train
  predset = splitdata$test


  #fit model
  fit <- Arima(estset,order=c(4,1,0), include.drift = FALSE, seasonal=list(order=c(1,0,1)))
  fc <- forecast(fit,h=12)#forecast 12 steps ahead
  j= accuracy(fc,predset)

  #store MAPE
  UMAPE[c,1]=i
  UMAPE[c,2]=j[2,5]
```

```
  c=c+1

#subset new data for new window
  newdata = subset(newdata,start=k,end=492) #setup data for next window
  k=k+12


}




c = 1
UMAPE1 =c()
UMAPE1 = as.data.frame(UMAPE1)
k=13
newdata = unrate.ts

for (i in 1:39){
  #dataset
  #make first 12 estimation

  estindex = length(newdata)-12
  splitdata = splitTrainTest(unrate.ts,estindex)
  estset = splitdata$train
  predset = splitdata$test



  fit <- Arima(estset,order=c(4,1,0), include.drift = FALSE, seasonal=list(order=c(1,0,1)))
  fc <- forecast(fit,h=1) #forecast 1 step ahead
  j= accuracy(fc,predset)


  UMAPE1[c,1]=i
  UMAPE1[c,2]=j[2,5]

  c=c+1


  newdata = subset(newdata,start=k,end=492)
  k=k+12


}


par(mfrow=c(1,1))

#plots
plot(1:39,UMAPE$V2,type='l',xlab="Iterations",ylab="MAPE")
```




```
plot(1:39,UMAPE1$V2,type='l',xlab="Iterations",ylab="MAPE")
```



(e) How do the errors found using a recursive backtesting scheme compare with the errors observed using a moving average backtesting scheme? Which scheme showed higher errors overall, and what does that tell you about your model?

Comparing the rolling window and the recursive backtesting schemes, we see that the errors are very similar for both time series. This indicates that our model is not dependent on observations that occur farther in the past because if we use an increasing test set or a rolling test set we get fairly consistent errors.

III. Conclusions and Future Work

Our final model to forecast the median sales price of new houses in the US includes a quadratic trend, seasonal AR(1) and seasonal MA(1), as well as ARIMA(3,0,0) to capture the cycles. Our final model to forecast the US unemployment rate includes no trend, seasonal AR(1) and seasonal MA(1), and ARIMA(4,1,0) to capture the cycles. The results of the VAR model output, as well as the granger-causality tests, indicate that the unemployment rate does not granger-cause the median sales price of new homes. This was a suprising result as we expected the unemployment rate to have some degree of preditive ability towards the median sales price. The next step in this analysis would be to incorporate another variable into the VAR model instead of the unemployment rate, such as housing supply.

IV. References (include the source of your data and any other resources).

Series 1: Median Sales Price for New Houses Sold in the United States (MSPNHSUS) https://fred.stlouisfed.org/series/MSPNHSUS

Series 2: Civilian Unemployment Rate (UNRATE) https://fred.stlouisfed.org/series/UNRATE