Project 2:

Oai Tran, Andrea Fleming, Ashtin Riad, Katherine Yu

2020-12-09

Abstract

This paper will study the relationship between the unemployment rate and consumer spending, particularly the total car sales in the US. Since a car purchase is an expensive purchase we think that people who are unemployed would probably not make a new car purchase. Empirical data shows that during a recession all consumption drops due to the rise of unemployment and we want to see what happens in particular to a large durable good purchase like a car.

0.1 Introduction:

Background on our interests:
During the recession, we see a dramatic drop in consumer level spending, due to consumers deciding to re-allocate their capital and be more selective on the type of goods and services they would be consuming. This leads us to want to know what level of change the Unemployment rate (UNRATENSA) would have in relation to a particular section, such as the Total Vehicle Sales (TOTALNSA) data. We will use the data from 1976-01-01 to 2020-10-01 from FRED to study the effect of the unemployment rate on total vehicle sales. We want to understand any relationship between the two, and if there is any correlation, what it would be. Furthermore, we will try to forecast the next level of the unemployment rate and total car sales.

Details of the data sets:
The two time series we will be using are Total Car Sales (TOTALNSA) and Unemployment rate, both not seasonally adjusted, (UNRATENSA) in the U.S. from 1976-01-01 to 2020-10-01. The frequency been use on both data are monthly (12). Total car sales not seasonally adjusted (TOTALNSA) unit is in Thousands of Units, Unemployment rate not seasonally adjusted (UNRATENSA) unit is in Percent.

0.2 Methodology and Data Analysis with Results:

Prep data to explore:
Using wo() function we tested for seasonality and it is present in both data sets.

```
une = read.csv("UNRATENSA.csv")
cars = read.csv("TOTALNSA.csv")

cars_ts = ts(cars$TOTALNSA, start = c(1976,1), frequency = 12)
summary(wo(cars_ts))
```

```
## Test used:  WO
##
## Test statistic:  1
## P-value:  0 0 0
##
## The WO - test identifies seasonality
```

```
une_ts = ts(une$UNRATENSA, start = c(1976,1), frequency = 12)
summary(wo(une_ts))
```

```
## Test used:  WO
##
## Test statistic:  1
## P-value:  0 0 0
##
## The WO - test identifies seasonality
```

```r
logcars = log(cars_ts)
```

This is a test to see if our data sets are stationary and we found out from the results using Dickey-Fuller test, ADF test, below both data are non-stationary and we do this just to give us an insight of the data before explore the two time series.

```r
#test for stationarity
adf.test(cars_ts, alternative="stationary", k=0)
```

```
## Warning in adf.test(cars_ts, alternative = "stationary", k = 0): p-value smaller
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cars_ts
## Dickey-Fuller = -10.348, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

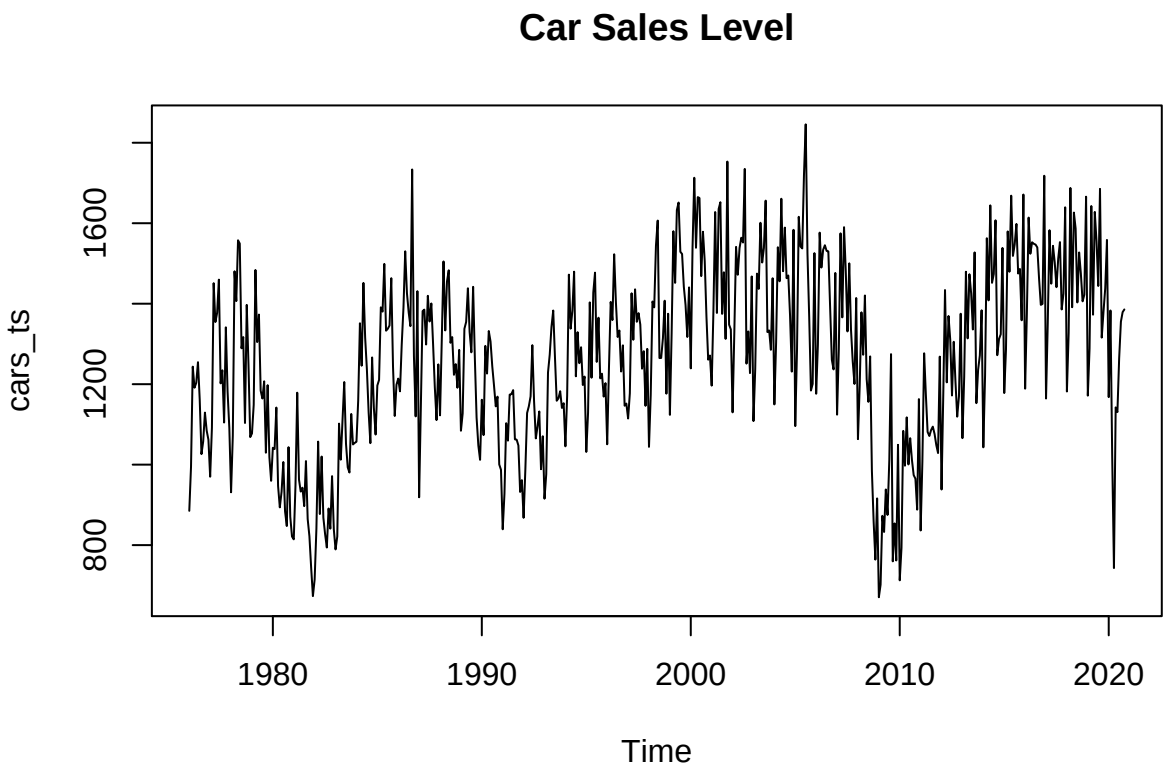
```r
adf.test(une_ts, alternative="stationary", k=0)
```

```
## Warning in adf.test(une_ts, alternative = "stationary", k = 0): p-value smaller
## than printed p-value
```

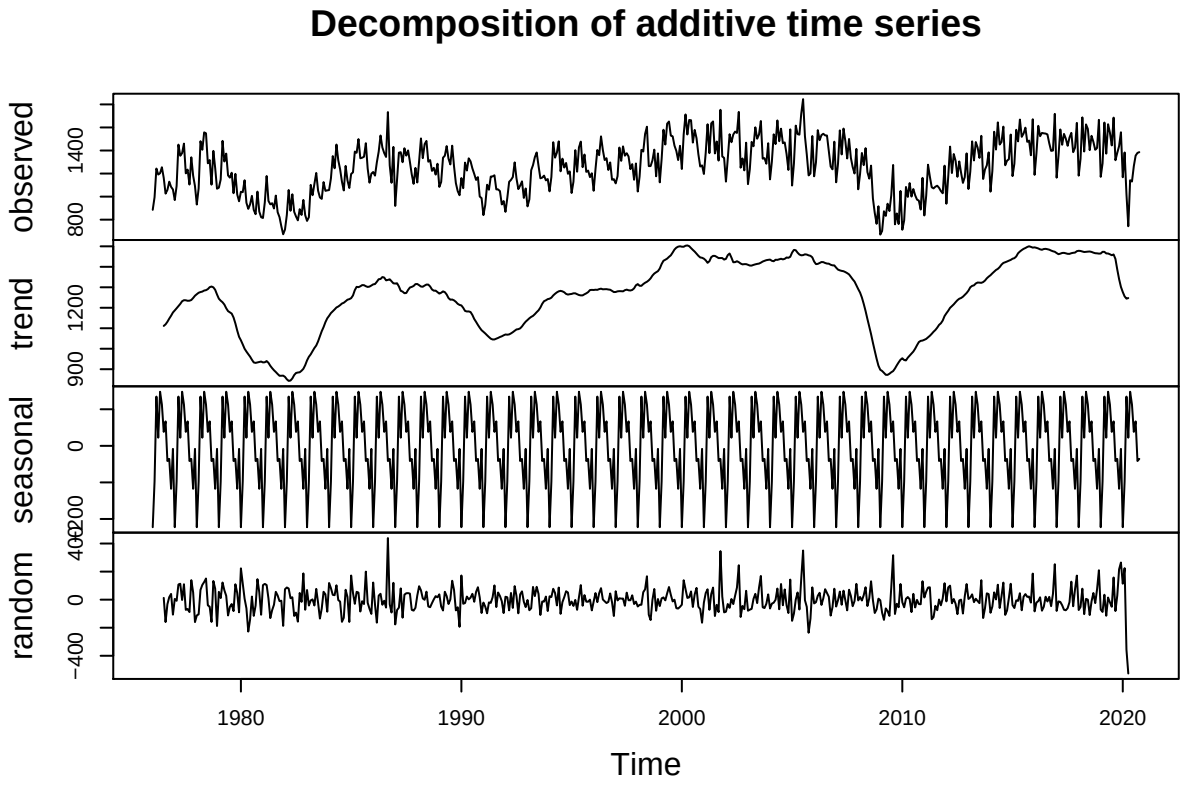
```
##
##  Augmented Dickey-Fuller Test
##
## data:  une_ts
## Dickey-Fuller = -4.2017, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

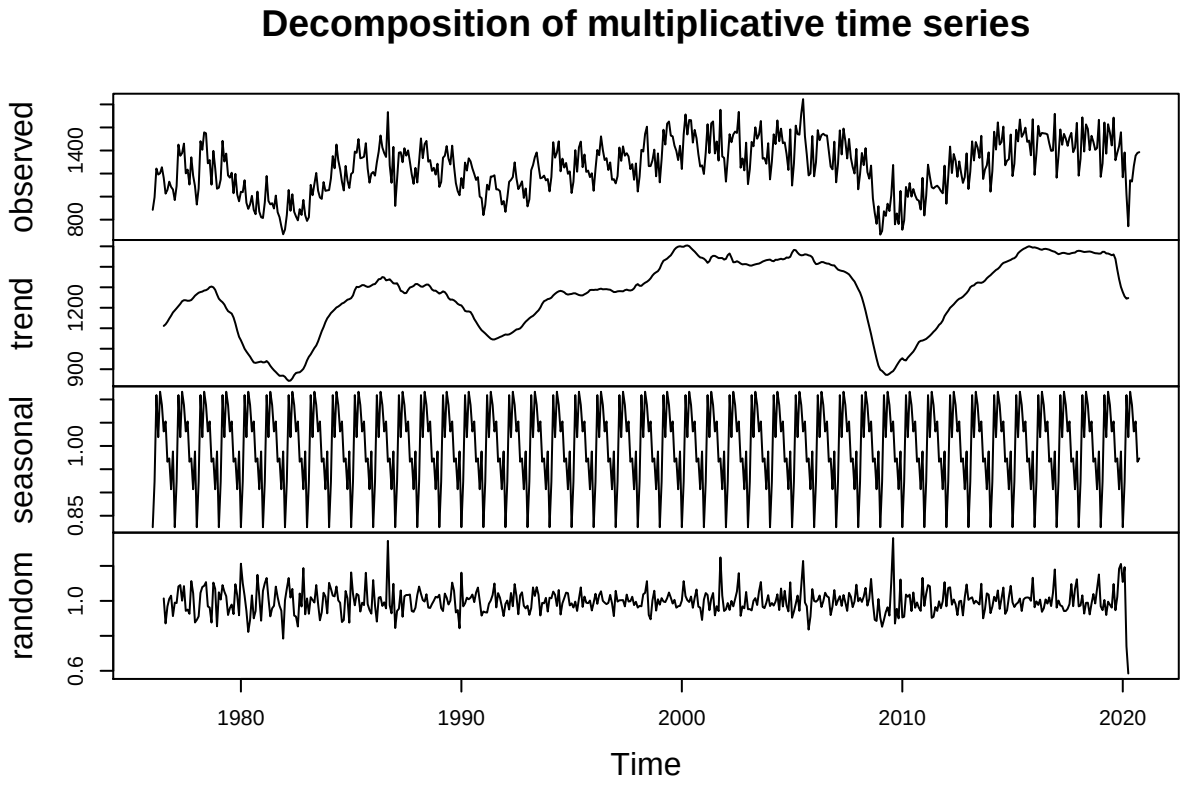
0.2.1 1 Produce a time-series plot of your data including the respective ACF and PACF plots.

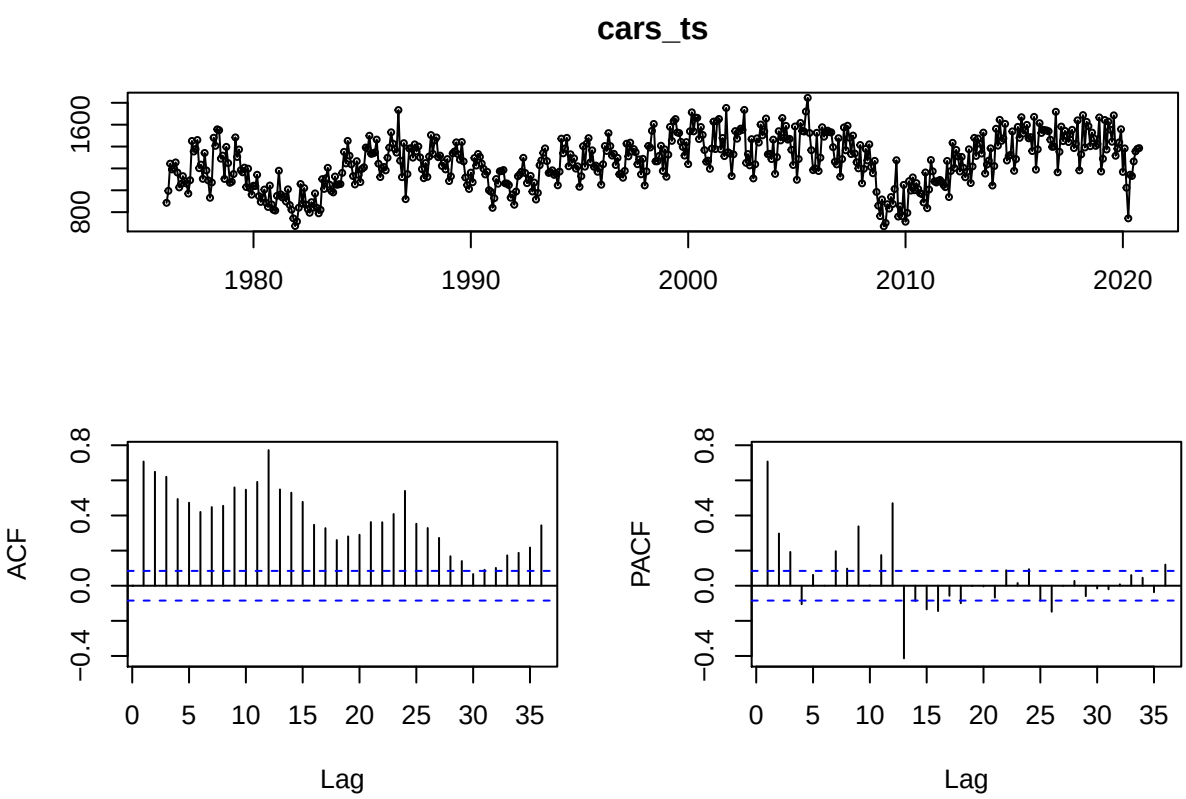
plot ACF and PACF individually and use it as way for us to build our model for number 2.

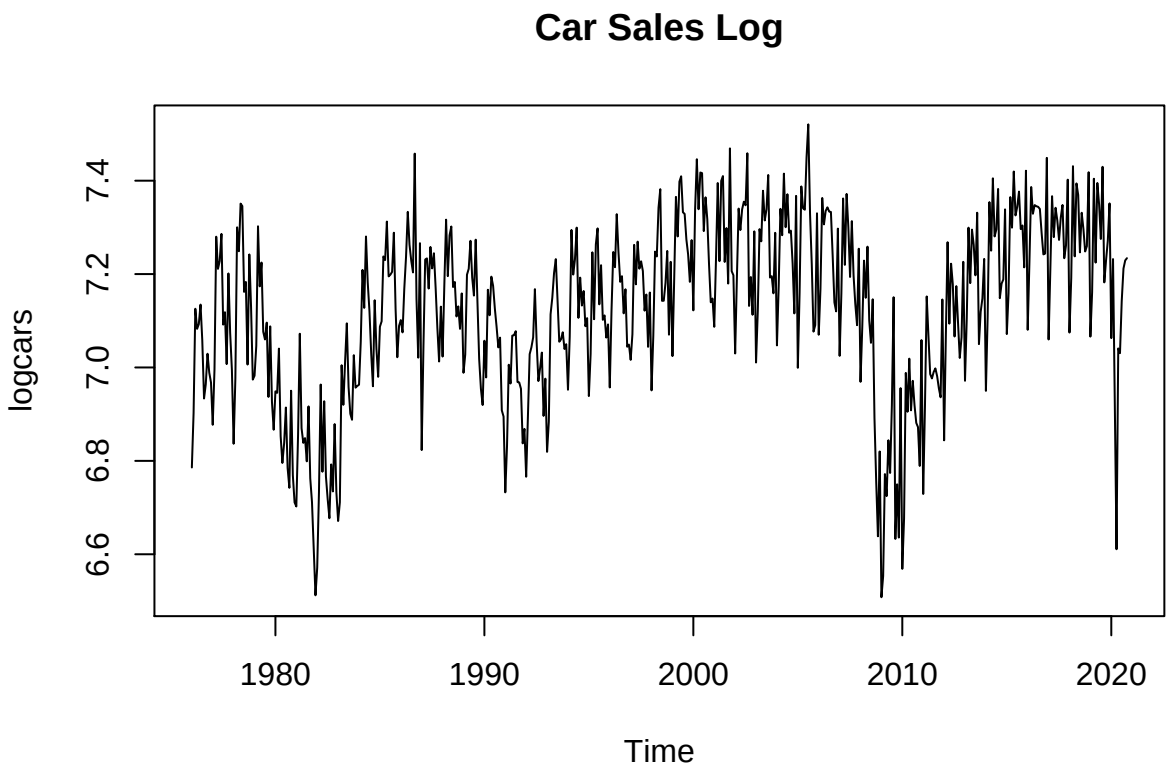


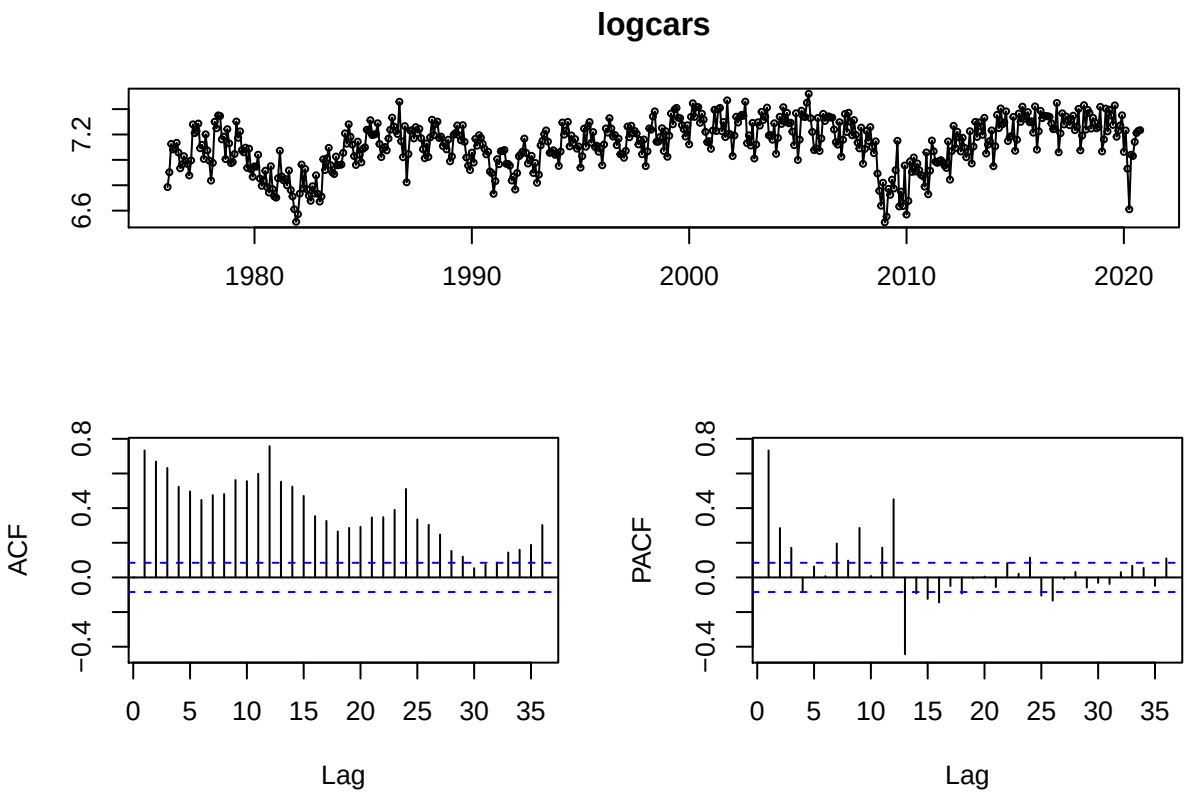
Exploring the data:

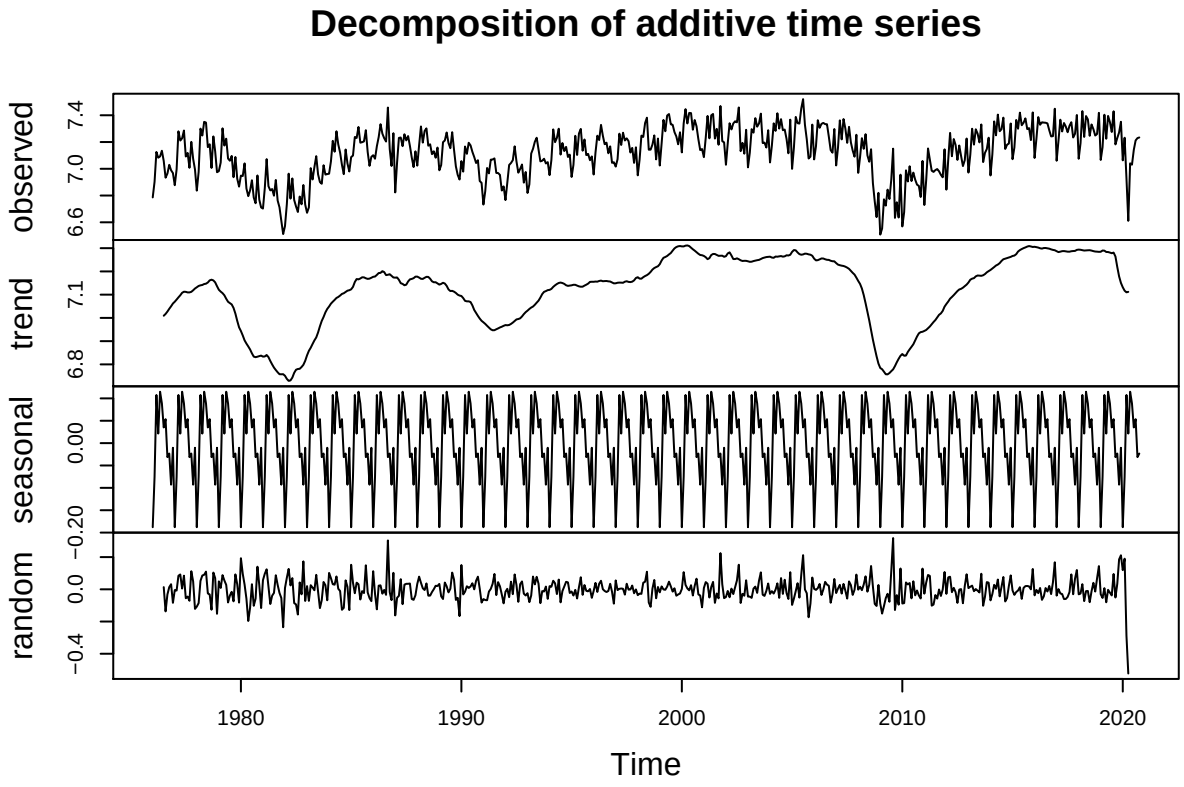

Decomposition of additive time series
observed
trend
seasonal
random
800
1400
900
1200
0
200
400
−400
0
1980
1990
2000
2010
2020
Time

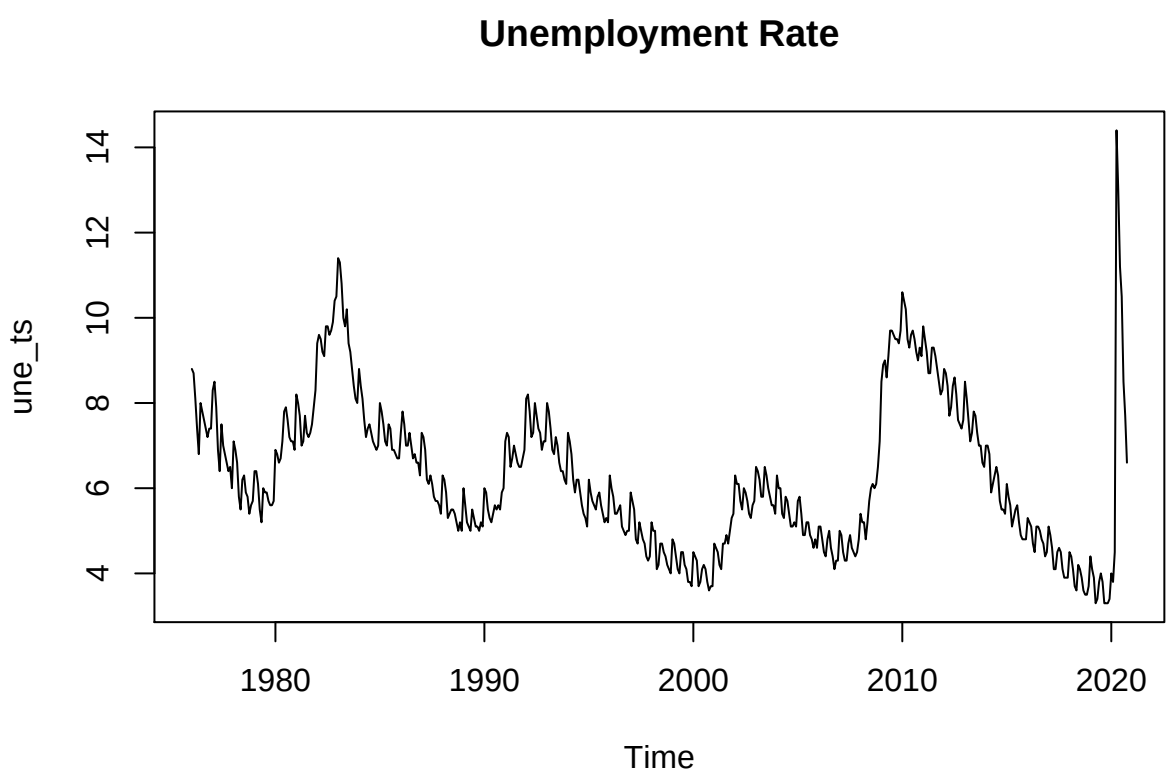

Decomposition of multiplicative time series
observed
trend
seasonal
random
800
1400
900
1200
0.85
1.00
0.6
1.0
1980
1990
2000
2010
2020
Time

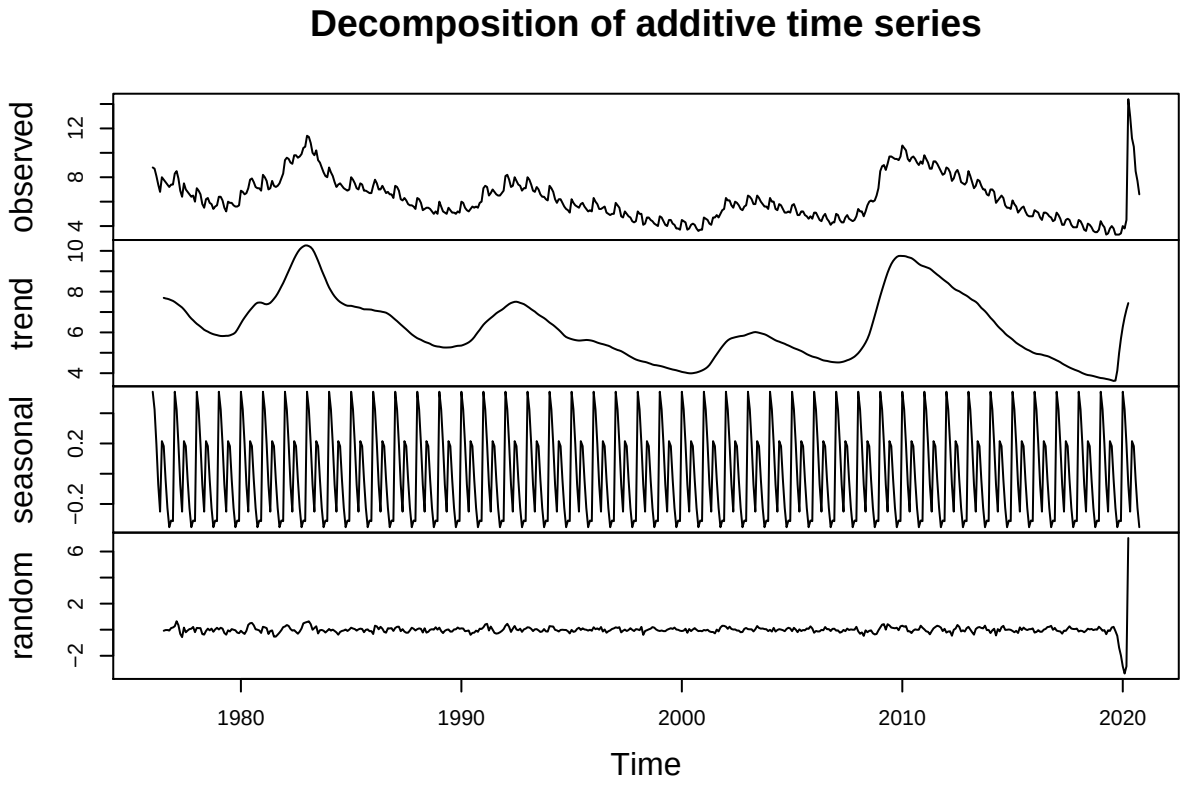

cars_ts
800
1600
1980
1990
2000
2010
2020
ACF
PACF
−0.4
0.0
0.4
0.8
0
5
10
15
20
25
30
35
Lag

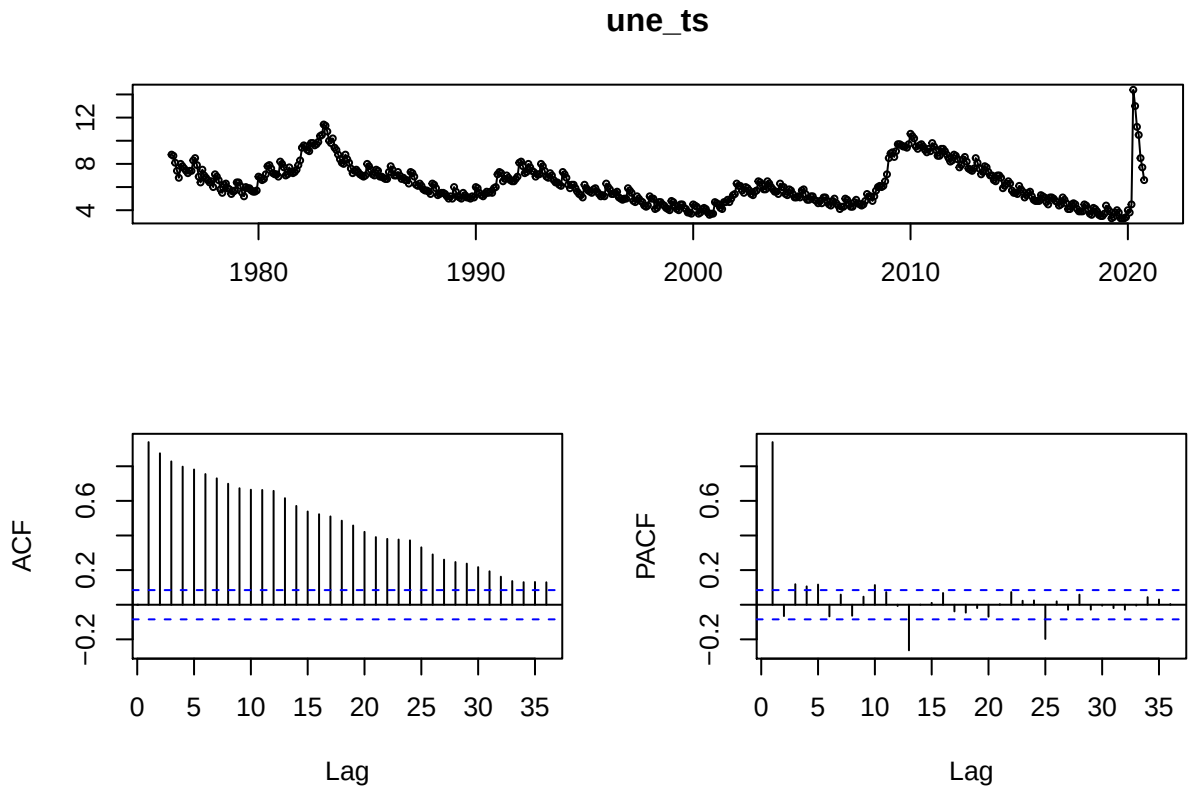

Car Sales Log
logcars
6.6
6.8
7.0
7.2
7.4
1980
1990
2000
2010
2020
Time


logcars
6.6
7.2
1980
1990
2000
2010
2020
ACF
PACF
−0.4
0.0
0.4
0.8
0
5
10
15
20
25
30
35
Lag
Lag


Decomposition of additive time series
observed
trend
seasonal
random
6.6
7.0
7.4
6.8
7.1
−0.20
0.00
0.20
−0.4
0.0
0.4
1980
1990
2000
2010
2020
Time


Unemployment Rate
une_ts
4
6
8
10
12
14
1980
1990
2000
2010
2020
Time


Decomposition of additive time series
observed
trend
seasonal
random
4
6
8
10
12
4
6
8
−0.2
0.2
−2
2
6
1980
1990
2000
2010
2020
Time



When we plot the decomposition of car sales we see that the trend varies widely and has no obvious pattern. In the decomposition this is captured in the trend so the residuals seem normal but we will later capture this trend in the cycles using an ARIMA model instead. The tsdisplay of CARS shows that there are signs of an AR(2) or AR(3) model with seasonality. There is are some pikes in the PACF in the later lags that are not due to seasonality that may be due to also having an MA component. We decide to take the log of CARS since the scale is very large.

When we plot the decomposition of UNEMPLOYMENT we see that there is strong seasonality. We also see that the trend would be rather hard to fit to a linear/quadratic/sinusoidal trend like we have learned in class. Like CARS, in the decomposition this is captured in the trend so the residuals seem normal but we will later capture this using an ARIMA model instead. The tsdisplay of UNEMPLOYMENT shows that there are signs of an AR(1) model with seasonality.

0.2.2 2. Fit an ARIMA model to each series and comment on the fit. For the next questions, you will instead use the model estimated in (3) for their respective answers.

We will use auto.arima here to get the base model. We also compare to see how the auto.arima model compares to some other selected models as well.

For log cars the auto.arima gives us : ARIMA(1,0,2)(0,1,2)[12] For unemployment the auto.arima gives us : ARIMA(1,1,1)(2,0,0)[12]

We are surprised that the model selected for log cars doesn't include differencing since the trend from the decomposition wasn't a good fit for a linear function.

For Total Car Sales Data: We decided that a ARIMA(1,0,2)(0,1,2)[12] model would fit to our car_sales_ts the best. From the ACF and PACF graphs above, we know that it is an AR(1) using PACF

from the original PACF(before we take the first diff()). Furthermore, after takes the first diff(), the ACF and PACF suggest to us there is a MA(2) with some non-seasonal MA(2) because there lot of spikes and it is not showing an decay as AR() model should. These keys characteristics tell us that we cam use the combination of AR and MA model to fit our data. We also use auto.arima() function to check after using ACF and PCF, we notice that a combination of both,ARIMA(1,0,2)(0,1,2)[12], from auto.arima() works best due to AIC=6320.89, and BIC=6346.48 have the smallest values. Not only that by looking at the fit of models individually, the last one fit the best out of the three model, which is ARIMA(1,0,2)(0,1,2)[12].
ARMA(1,1) -> AIC = 6922.7.
ARMA(1,4) -> AIC = 6893.5.
ARIMA(1,0,2)(0,1,2)[12] -> AIC=6321.05.

For Unemployment rate: For our UNRATE_not_ts time series, we will fit ARIMA(1,1,2)(2,0,0)[12]. Here I repeated the same steps as above model car_sales_ts. Using ACF and PACF suggesting us the model would be a combination of AR and MA such as ARMA(1,2) or ARMA(1,4). But by using auto.arima() we learned in class, we found that ARIMA(1,1,2)(2,0,0)[12], gives us a better fit in the forecast than the other models with an AIC= 828.27.
ARMA(1,2) -> AIC = 956.6.
ARMA(1,4) -> AIC = 949.26.
ARIMA(1,1,2)(2,0,0)[12] -> AIC= 828.27.

Using auto.arima() to find a model and use it as a benchmark to help us understand the both of the data better and compare them with other models such as AR and or MA methods. Below are the codes and different models that we are testing.

```
cars <- auto.arima(cars_ts)
summary(cars)
```

```
## Series: cars_ts
## ARIMA(1,0,2)(0,1,2)[12]
##
## Coefficients:
##          ar1      ma1      ma2     sma1     sma2
##       0.9666  -0.5078  -0.1152  -0.5622  -0.2496
## s.e.  0.0139   0.0463   0.0460   0.0454   0.0438
##
## sigma^2 estimated as 9358:  log likelihood=-3154.44
## AIC=6320.89   AICc=6321.05   BIC=6346.48
##
## Training set error measures:
##                     ME    RMSE      MAE        MPE     MAPE     MASE
## Training set 2.232457 95.1972 68.09466 -0.3297174 5.608962 0.630223
##                       ACF1
## Training set -0.001769225
```

```
carslog = auto.arima(logcars)
summary(carslog)
```

```
## Series: logcars
## ARIMA(1,0,2)(0,1,2)[12]
##
## Coefficients:
```

```
##          ar1      ma1      ma2     sma1     sma2
##       0.9635  -0.4622  -0.1103  -0.5885  -0.2198
## s.e.  0.0144   0.0460   0.0450   0.0457   0.0436
##
## sigma^2 estimated as 0.006239:  log likelihood=585.74
## AIC=-1159.48   AICc=-1159.31   BIC=-1133.88
##
## Training set error measures:
##                       ME       RMSE        MAE        MPE      MAPE      MASE
## Training set 0.001886521 0.07772692 0.05508625 0.01659669 0.7774167 0.6093331
##                      ACF1
## Training set 0.0003468893
```

```
unrate <- auto.arima(une_ts)
summary(unrate)
```

```
## Series: une_ts
## ARIMA(1,1,1)(2,0,0)[12]
##
## Coefficients:
##          ar1     ma1    sar1    sar2
##       -0.8533  0.9289  0.4070  0.3715
## s.e.   0.0597  0.0450  0.0889  0.0894
##
## sigma^2 estimated as 0.2658:  log likelihood=-409.14
## AIC=828.27   AICc=828.39   BIC=849.7
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE    MAPE      MASE
## Training set 0.004412565 0.5131745 0.2024289 0.02101432 3.03623 0.2443828
##                     ACF1
## Training set -0.028649
```

```
#fit the model into the data
#AR1 and MA(2)
arma1 <- arma(cars_ts, order = c(1,1))
summary(arma1)
```

```
##
## Call:
## arma(x = cars_ts, order = c(1, 1))
##
## Model:
## ARMA(1,1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -447.259 -103.384   -7.173   97.096  392.259
##
## Coefficient(s):
##            Estimate  Std. Error  t value Pr(>|t|)
## ar1         0.91222     0.02314   39.416  < 2e-16 ***
## ma1        -0.45912     0.05336   -8.604  < 2e-16 ***
```

```
## intercept 111.44802    29.41013    3.789 0.000151 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 22437,  Conditional Sum-of-Squares = 12025986,  AIC = 6922.7
```

```
arma1.2 <- arma(cars_ts, order = c(1,4))
#result: ma2 is not significant
summary(arma1.2)
```

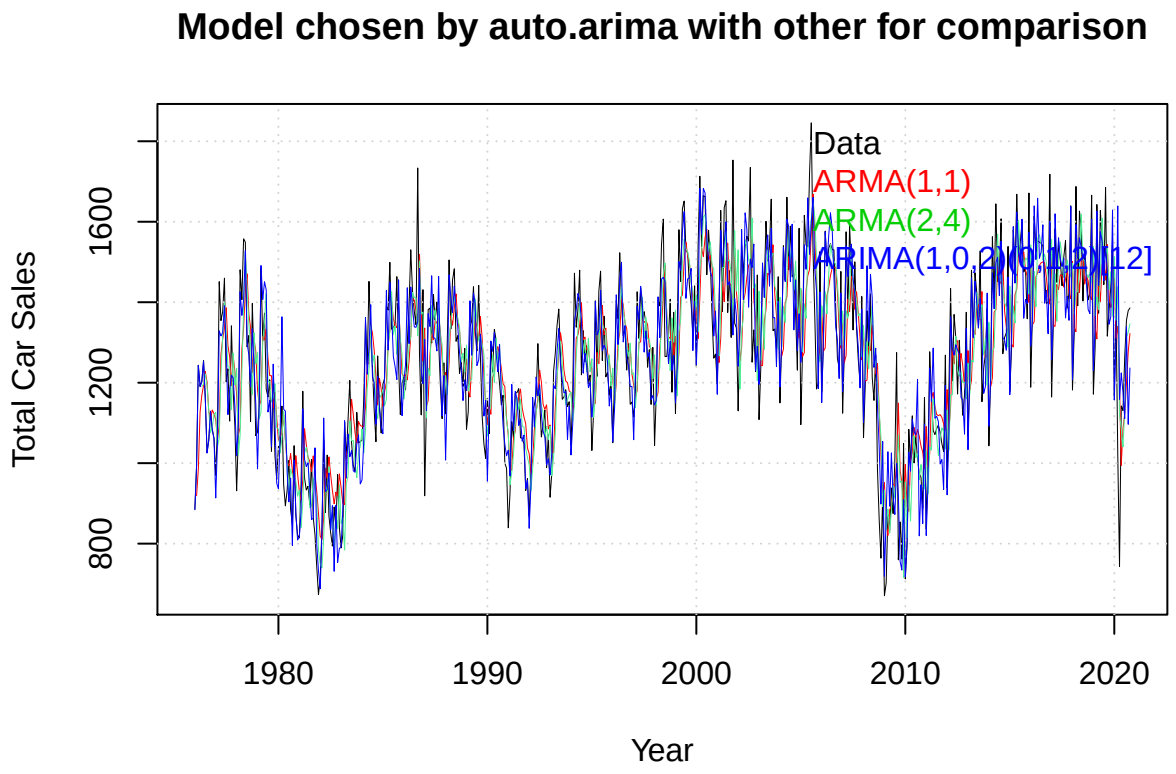
```
##
## Call:
## arma(x = cars_ts, order = c(1, 4))
##
## Model:
## ARMA(1,4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -415.280  -93.087   -3.433   96.766  417.159
##
## Coefficient(s):
##            Estimate  Std. Error  t value Pr(>|t|)
## ar1         1.00369     0.01009   99.440  < 2e-16 ***
## ma1        -0.54991     0.04343  -12.663  < 2e-16 ***
## ma2        -0.05901     0.04463   -1.322    0.186
## ma3         0.20130     0.04495    4.478 7.54e-06 ***
## ma4        -0.31009     0.04079   -7.602 2.93e-14 ***
## intercept  -4.62252    12.87253   -0.359    0.720
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 21015,  Conditional Sum-of-Squares = 11201402,  AIC = 6893.5
```

```
# test out the ARMA
arma1.1 <- auto.arima(cars_ts)
arma1.1
```

```
## Series: cars_ts
## ARIMA(1,0,2)(0,1,2)[12]
##
## Coefficients:
##          ar1      ma1      ma2     sma1     sma2
##       0.9666  -0.5078  -0.1152  -0.5622  -0.2496
## s.e.  0.0139   0.0463   0.0460   0.0454   0.0438
##
## sigma^2 estimated as 9358:  log likelihood=-3154.44
## AIC=6320.89   AICc=6321.05   BIC=6346.48
```

Here are the graphs with all the models:

```r
#plot
plot(cars_ts, xlab="Year", ylab="Total Car Sales", col="black", lwd=.5, main = 'Model chosen by auto.ar
grid()
lines(arma1$fitted.values, col="red", lwd=.5, lty=1)
lines(arma1.2$fitted.values, col="seagreen2", lwd=.5, lty=1)
lines(fitted(arma1.1), col="blue", lwd=.5, lty=1)
legend("topright", legend=c("Data", "ARMA(1,1)", "ARMA(2,4)", "ARIMA(1,0,2)(0,1,2)[12]"), text.col=1:4,
```



Unemployment rate:

```r
#AR1 and MA(2)
arma2 <- arma(une_ts, order = c(1,2))
summary(arma2)
```

```
##
## Call:
## arma(x = une_ts, order = c(1, 2))
##
## Model:
## ARMA(1,2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76966 -0.25046 -0.09711  0.14741  9.75355
```

```
##
## Coefficient(s):
##            Estimate  Std. Error  t value Pr(>|t|)
## ar1         0.95421     0.01534   62.202  < 2e-16 ***
## ma1         0.02486     0.04991    0.498  0.61837
## ma2        -0.16081     0.05640   -2.851  0.00435 **
## intercept   0.28435     0.09845    2.888  0.00387 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 0.3414,  Conditional Sum-of-Squares = 182.65,  AIC = 956.6
```

```
arma2.2 <- arma(une_ts, order = c(1,4))
#result: ma1 is not significant
summary(arma2.2)
```

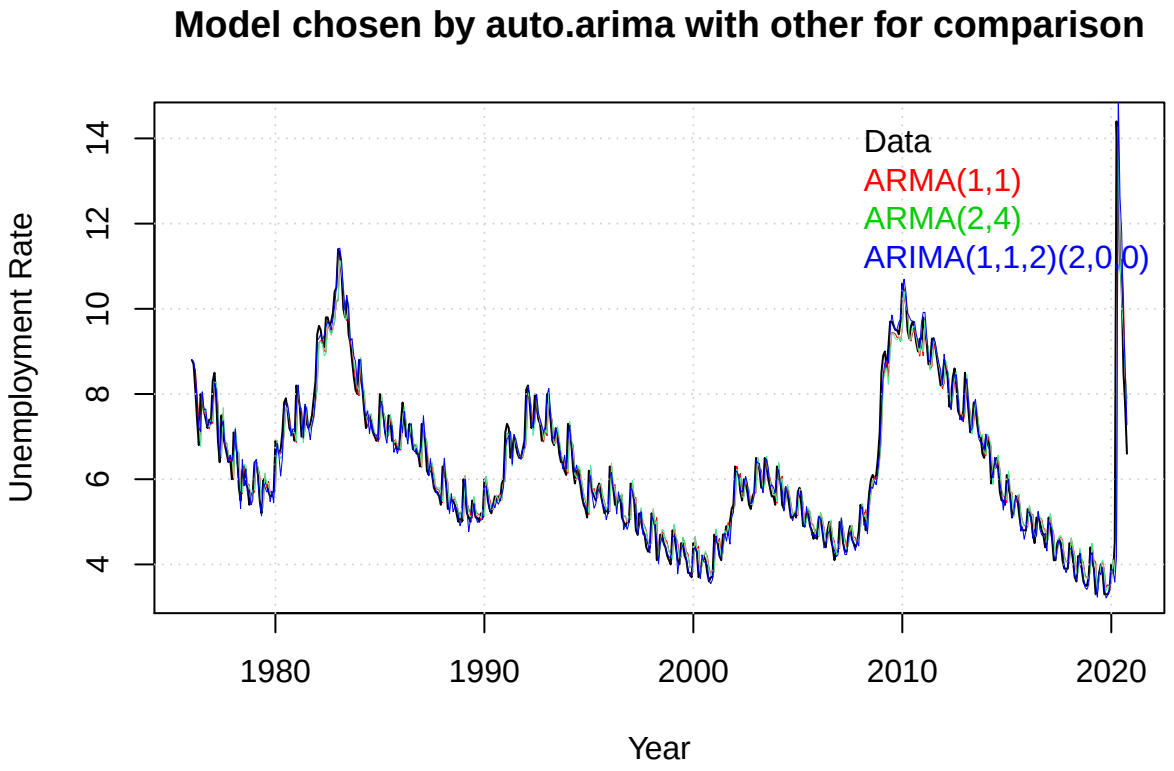
```
##
## Call:
## arma(x = une_ts, order = c(1, 4))
##
## Model:
## ARMA(1,4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2799 -0.2581 -0.1083  0.1554  9.8270
##
## Coefficient(s):
##            Estimate  Std. Error  t value Pr(>|t|)
## ar1         0.97380     0.01149   84.746  < 2e-16 ***
## ma1         0.02178     0.04535    0.480 0.631104
## ma2        -0.15550     0.04434   -3.507 0.000453 ***
## ma3        -0.10549     0.04569   -2.309 0.020955 *
## ma4        -0.11227     0.04172   -2.691 0.007121 **
## intercept   0.16420     0.07364    2.230 0.025766 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 0.3343,  Conditional Sum-of-Squares = 178.17,  AIC = 949.26
```

```
#plug in the seasonal
arma2.1 <- auto.arima(une_ts)
summary(arma2.1)
```

```
## Series: une_ts
## ARIMA(1,1,1)(2,0,0)[12]
##
## Coefficients:
##          ar1     ma1    sar1    sar2
##      -0.8533  0.9289  0.4070  0.3715
## s.e.  0.0597  0.0450  0.0889  0.0894
```

```
##
## sigma^2 estimated as 0.2658:  log likelihood=-409.14
## AIC=828.27   AICc=828.39   BIC=849.7
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE    MAPE      MASE
## Training set 0.004412565 0.5131745 0.2024289 0.02101432 3.03623 0.2443828
##                    ACF1
## Training set -0.028649
```

```
#plot
plot(une_ts, xlab="Year", ylab="Unemployment Rate", col="black", lwd=1, main = 'Model chosen by auto.ar
grid()
lines(arma2$fitted.values, col="red", lwd=.5, lty=1)
lines(arma2.2$fitted.values, col="seagreen2", lwd=.5, lty=1)
lines(fitted(arma2.1), col="blue", lwd=.5, lty=1)
legend("topright", legend=c("Data", "ARMA(1,1)", "ARMA(2,4)", "ARIMA(1,1,2)(2,0,0)"), text.col=1:4, bty
```



0.3 3. Fit a model that includes, trend, seasonality and cyclical components. Make sure to discuss your model in detail.

From the very first ACF and PACF of the original data we get these results (copied again here for convenience):

When we plot the decomposition In the decomposition this is captured in the trend so the residuals seem normal but we will later capture this using an ARIMA model instead. The tsdisplay of CARS shows that there are signs of an AR(2) or AR(3) model with seasonality. There is are some pikes in the PACF in the later lags that are not due to seasonality that may be due to also having an MA component. We decide to take the log of CARS since the scale is very large.

When we plot the decomposition of UNEMPLOYMENT we see that there is strong seasonality. We also see that the trend would be rather hard to fit to a linear/quadratic/sinusoidal trend like we have learned in class. Like CARS, in the decomposition this is captured in the trend so the residuals seem normal but we will later capture this using an ARIMA model instead. The tsdisplay of UNEMPLOYMENT shows that there are signs of an AR(1) model with seasonality.

Down below, we are testing our difference AR, MA, ARIMA() models to select the best one to use for our time series. Un on testing, we decided to go with Total car sales final model: diff2.5 model which is ARIMA(1,1,1)x(0,1,1) [12], and unemployment final model: diff3.7 model which is ARIMA(1,1,1)x(2,0,0)[12].

Below are our method of model selection:

Fitting the models: testing different model to fit in
Here is for fitting a non-differenced model for cars: We end up choose quadratic for trend in this test

```
#fitting a non-differenced model for cars

plot(logcars, main = 'Car Sales Log',xlab="Time", lwd=3, col='skyblue3')

#create time sequence
t = (as.Date(seq(as.Date("1976/01/01"), by = "month", length.out = length(cars_ts))))
t = decimal_date(t)
sint = sin(t)
cost = cos(t)
t2 = t^2

t1 = tslm(logcars~trend)
summary(t1)
```

```
##
## Call:
## tslm(formula = logcars ~ trend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66685 -0.09421  0.02363  0.13365  0.39291
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.012e+00  1.536e-02 456.460  < 2e-16 ***
## trend       4.096e-04  4.939e-05   8.294  8.9e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1779 on 536 degrees of freedom
## Multiple R-squared:  0.1137, Adjusted R-squared:  0.1121
```

```
## F-statistic: 68.79 on 1 and 536 DF,  p-value: 8.903e-16
```

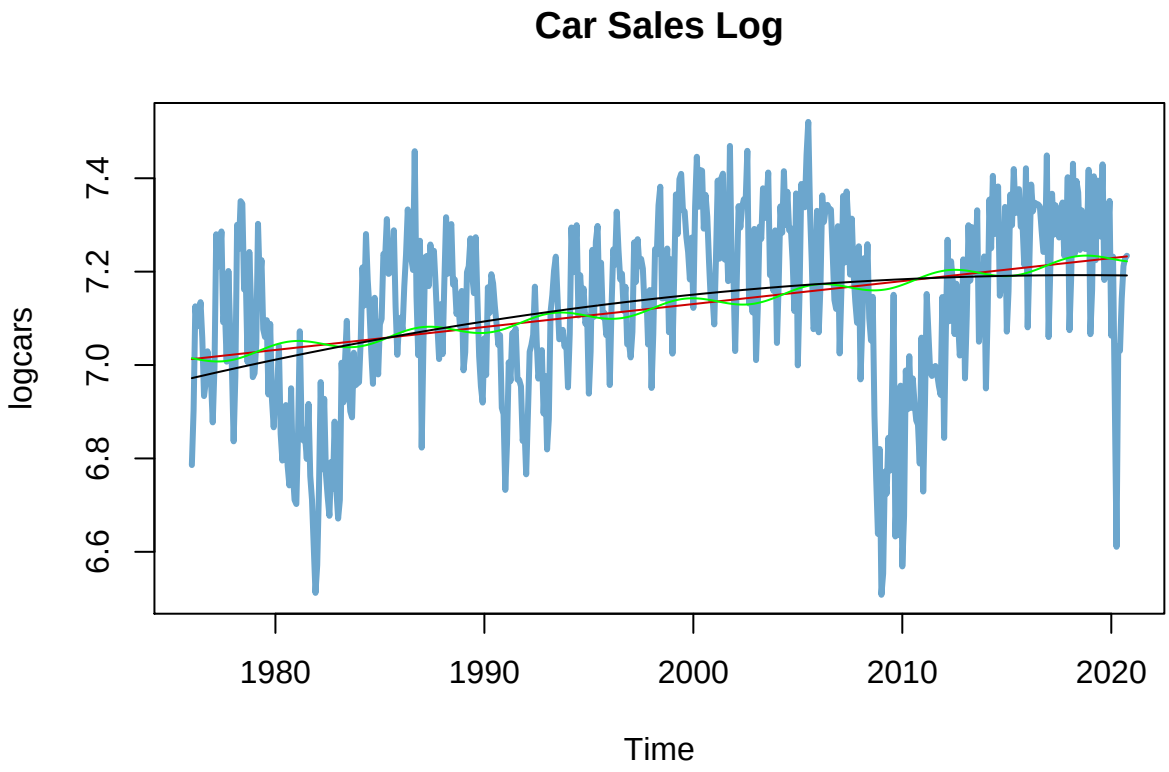
```
lines(t,t1$fit,col="red3",lwd=1)

t1.1 = tslm(logcars~t+sint+cost)
summary(t1.1)
```

```
##
## Call:
## tslm(formula = logcars ~ t + sint + cost)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65274 -0.09833  0.02058  0.13266  0.37940
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.5639212  1.1907246  -2.153   0.0317 *
## t            0.0048472  0.0005958   8.135  2.9e-15 ***
## sint         0.0134773  0.0109756   1.228   0.2200
## cost         0.0004429  0.0107895   0.041   0.9673
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.178 on 534 degrees of freedom
## Multiple R-squared:  0.1162, Adjusted R-squared:  0.1112
## F-statistic:  23.4 on 3 and 534 DF,  p-value: 3.026e-14
```

```
lines(t,t1.1$fit,col="green",lwd=1)

t1.3 = tslm(logcars~t+t2)
summary(t1.3)
```

```
##
## Call:
## tslm(formula = logcars ~ t + t2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67352 -0.09502  0.02715  0.13131  0.38918
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.907e+02  2.036e+02  -2.411   0.0163 *
## t            4.933e-01  2.037e-01   2.421   0.0158 *
## t2          -1.222e-04  5.097e-05  -2.397   0.0169 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1771 on 535 degrees of freedom
## Multiple R-squared:  0.1231, Adjusted R-squared:  0.1198
## F-statistic: 37.56 on 2 and 535 DF,  p-value: 5.444e-16
```

```
lines(t,t1.3$fit,col="black",lwd=1)
```




```
AIC(t1, t1.1, t1.3)
```

```
##      df       AIC
## t1    3 -326.8988
## t1.1  5 -324.3974
## t1.3  4 -330.6259
```

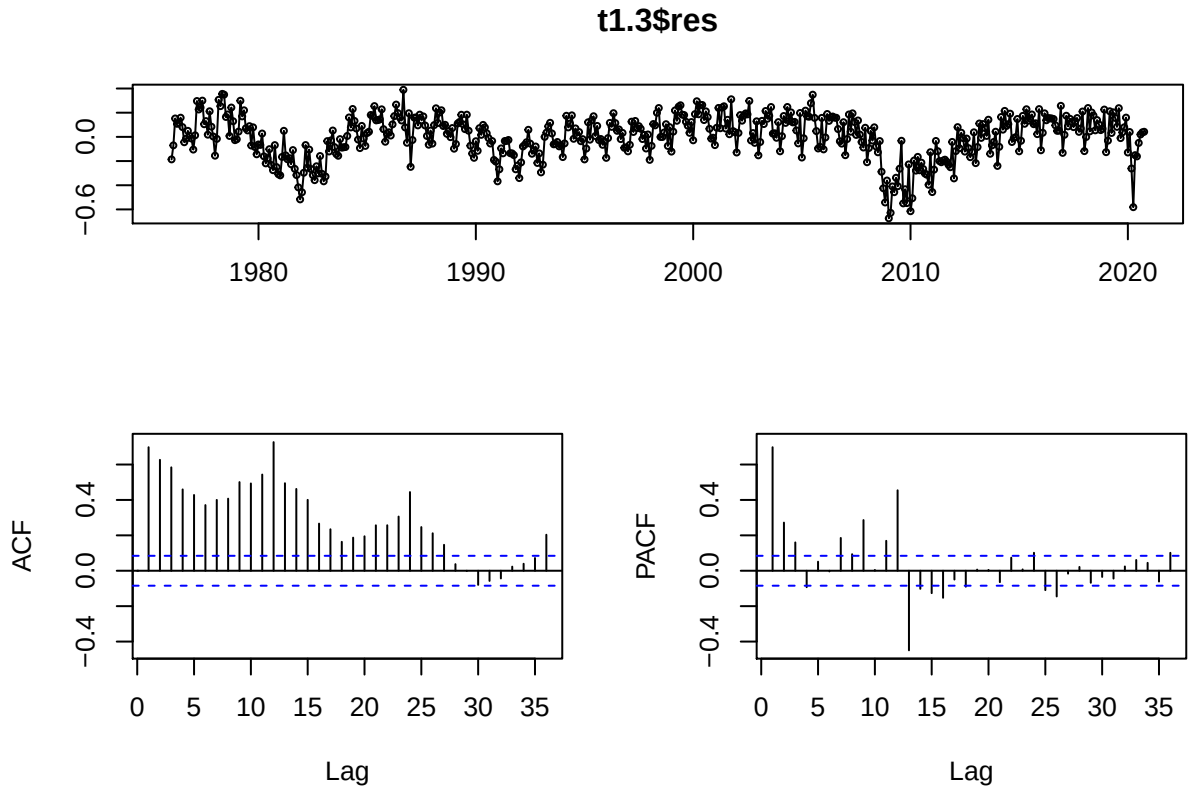
```
#We end up choosing the quadratic for trend.
```

From the result above, we end up choosing the quadratic for trend since the AIC is the lowest for our total car sales.

Next, we try to fit an ARIMA model while keeping the trend and not first differencing.

Look at residuals of quadratic trend and ACF and PACF with season dummies:

```
#look at residuals of quadratic trend
tsdisplay(t1.3$res) #might be S-AR(1) and AR(1) or AR(3) from looking at the ACF and PACF.
```

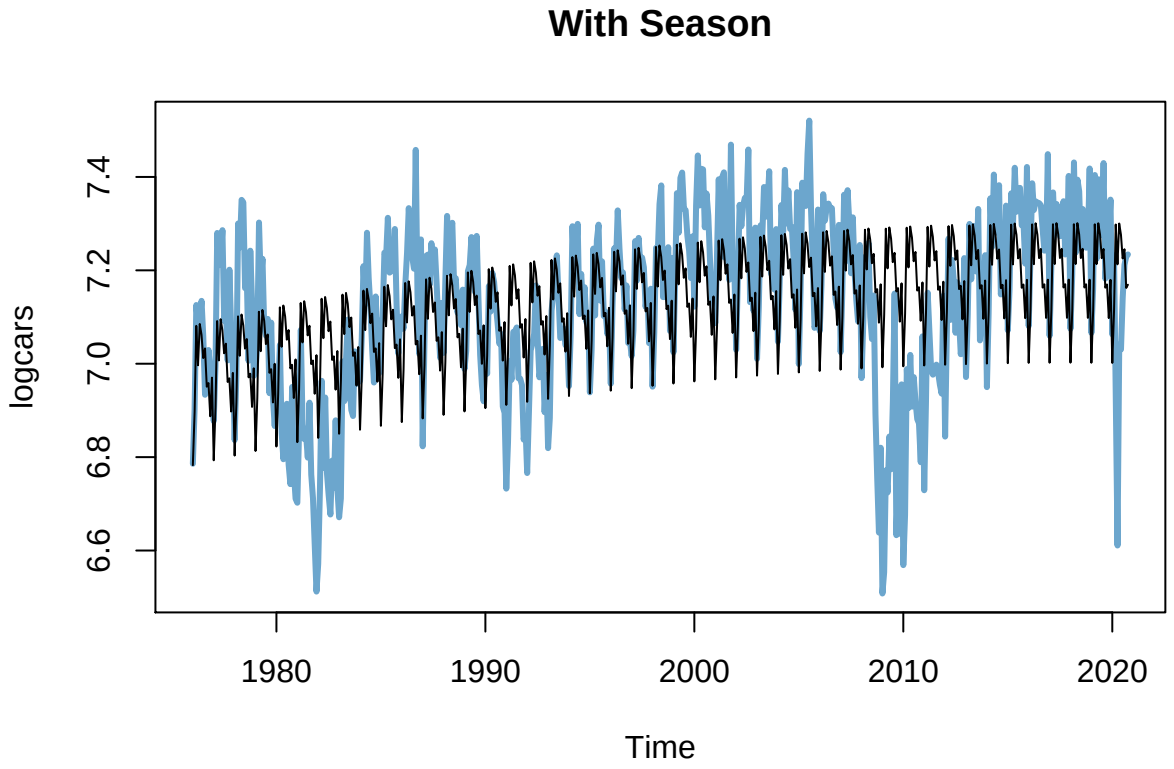



```
#look at ACF and PACF with season dummies
quadseason = tslm(logcars~t+t2+season)
plot(logcars, xlab="Time", lwd=3, col='skyblue3', main="With Season")
summary(quadseason)
```

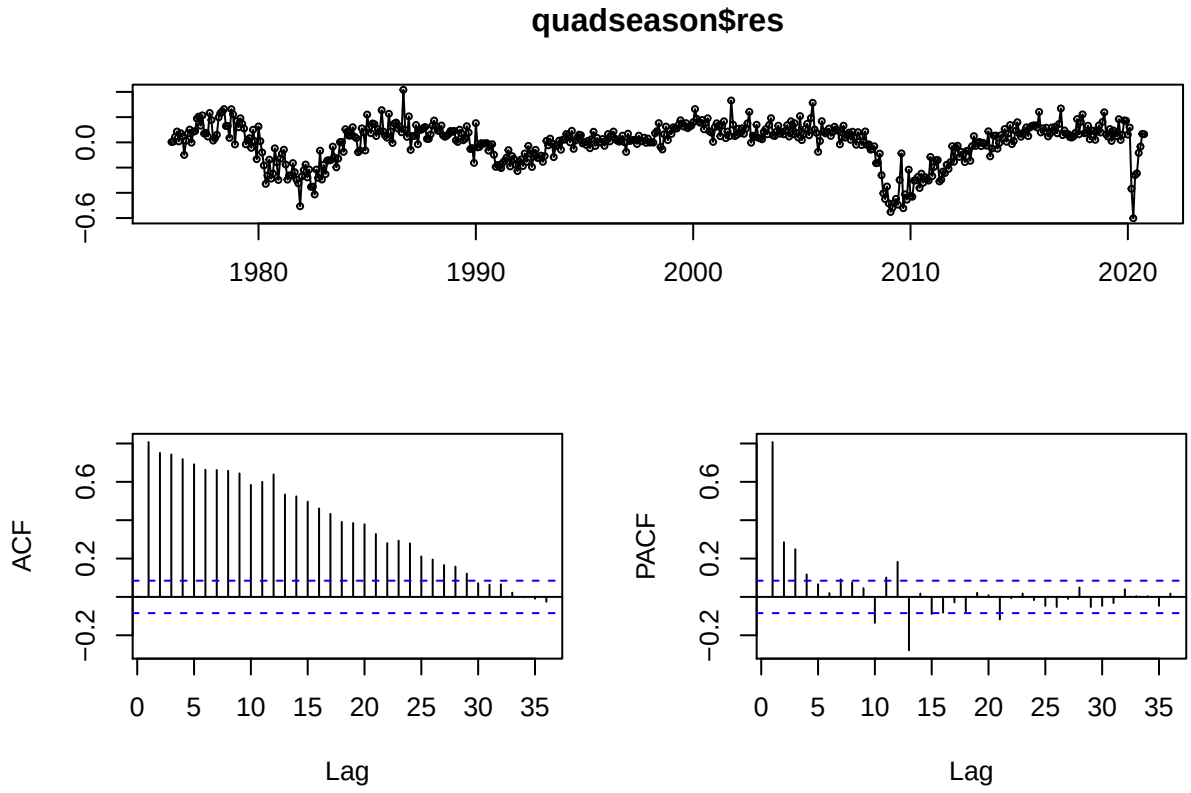
```
##
## Call:
## tslm(formula = logcars ~ t + t2 + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.60209 -0.05961  0.03780  0.09340  0.41661
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.016e+02  1.801e+02  -2.785 0.005549 **
## t            5.041e-01  1.803e-01   2.796 0.005359 **
## t2          -1.249e-04  4.510e-05  -2.769 0.005818 **
## season2      1.127e-01  3.304e-02   3.413 0.000693 ***
## season3      2.966e-01  3.304e-02   8.977  < 2e-16 ***
## season4      2.113e-01  3.304e-02   6.395 3.56e-10 ***
## season5      2.988e-01  3.304e-02   9.045  < 2e-16 ***
## season6      2.747e-01  3.304e-02   8.316 7.89e-16 ***
## season7      2.241e-01  3.304e-02   6.783 3.19e-11 ***
## season8      2.439e-01  3.304e-02   7.382 6.18e-13 ***
## season9      1.610e-01  3.304e-02   4.874 1.45e-06 ***
```

```
## season10     1.680e-01  3.304e-02   5.085 5.13e-07 ***
## season11     9.597e-02  3.323e-02   2.888 0.004032 **
## season12     1.779e-01  3.323e-02   5.355 1.28e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1567 on 524 degrees of freedom
## Multiple R-squared:  0.3278, Adjusted R-squared:  0.3111
## F-statistic: 19.66 on 13 and 524 DF,  p-value: < 2.2e-16
```

```
lines(t,quadseason$fit,col="black",lwd=1)
```



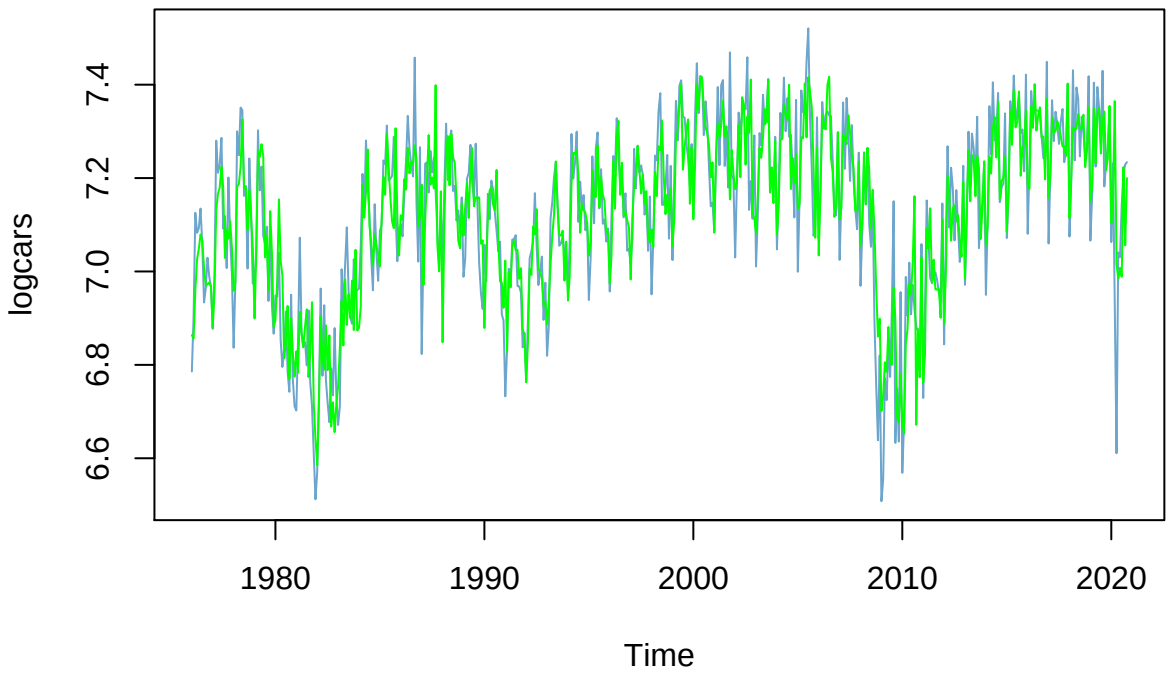

```
tsdisplay(quadseason$res) #since spikes in 1-3 persist in PACF from this we try
```




```
#to fit AR(3) for cycles
```

The comments on the ACF and PACF are also included in the code. When we look at the residuals for the quadratic trend model we see that there could be S-AR(1) and AR(1) or AR(3) from looking at the ACF and PACF. After trying trend with seasonal dummies, from the residuals we get that spikes in 1-3 persist in PACF from this we try to fit AR(3) for cycles.

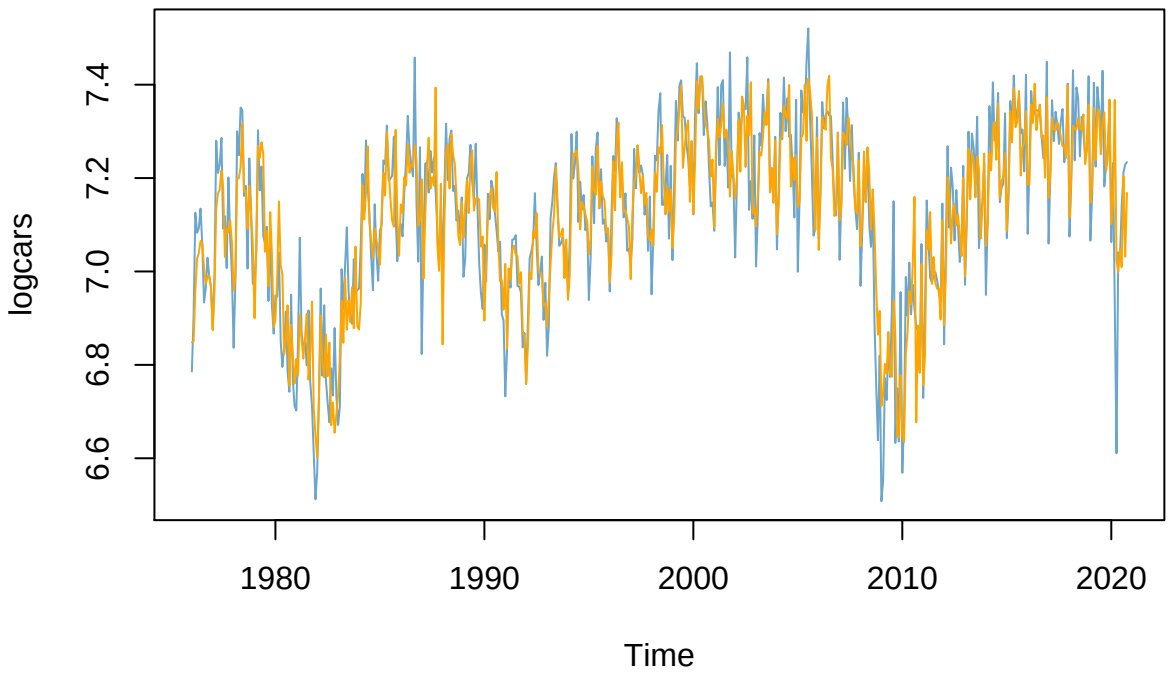
```
#fit ar models
ar1=arima(logcars,order=c(3,0,0),xreg = cbind(t, t2),seasonal=list(order=c(1,0,0)))
plot(logcars, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(ar1),col="green")
```




```
summary(ar1)
```

```
##
## Call:
## arima(x = logcars, order = c(3, 0, 0), seasonal = list(order = c(1, 0, 0)),
##     xreg = cbind(t, t2))
##
## Coefficients:
##          ar1     ar2     ar3    sar1  intercept       t      t2
##       0.4863  0.1571  0.1333  0.7139  -731.9582  0.7351  -2e-04
## s.e.  0.0427  0.0471  0.0428  0.0304    88.3592  0.0874   0e+00
##
## sigma^2 estimated as 0.007244:  log likelihood = 557.36,  aic = -1098.72
##
## Training set error measures:
##                         ME       RMSE        MAE         MPE      MAPE      MASE
## Training set 0.0002710538 0.08511308 0.06217704 -0.01027597 0.8782087 0.5725039
##                      ACF1
## Training set -0.009834658
```

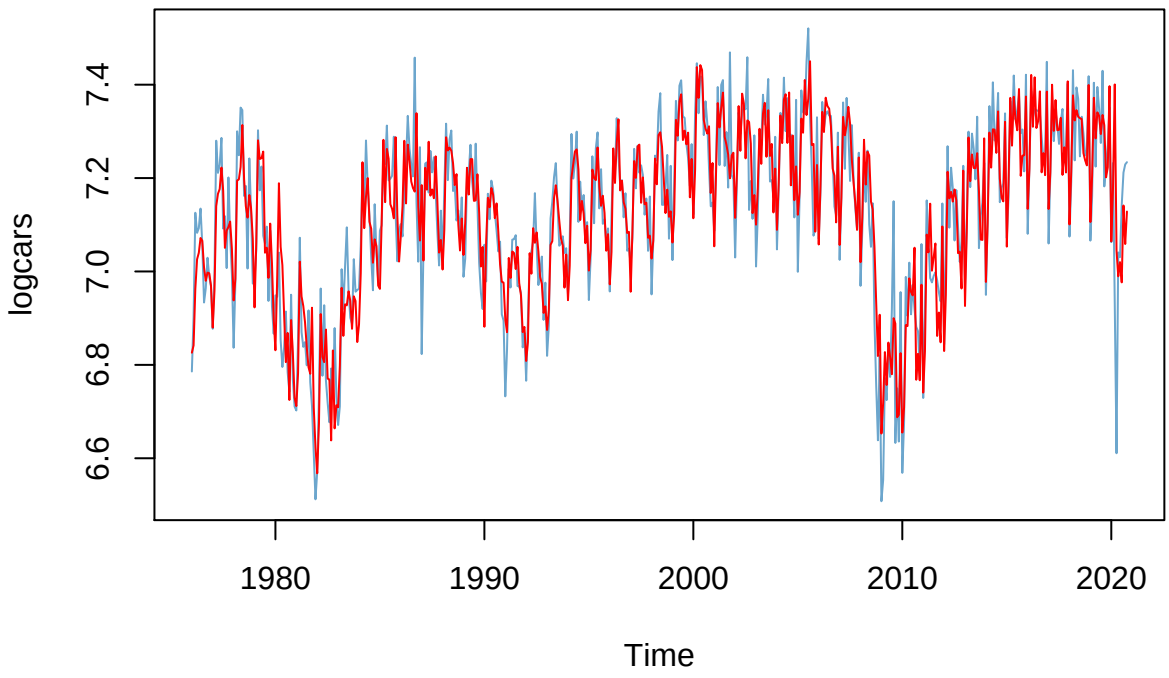
```
ar2=arima(logcars,order=c(3,0,1),xreg = cbind(t, t2),seasonal=list(order=c(1,0,0)))
plot(logcars, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(ar2),col="orange")
```




```
summary(ar2)
```

```
##
## Call:
## arima(x = logcars, order = c(3, 0, 1), seasonal = list(order = c(1, 0, 0)),
##     xreg = cbind(t, t2))
##
## Coefficients:
##          ar1      ar2     ar3      ma1    sar1  intercept       t      t2
##       1.0762  -0.1488  0.0008  -0.6072  0.7210  -919.3084  0.9226  -2e-04
## s.e.  0.1672   0.0949  0.0703   0.1620  0.0301   373.6983  0.3738   1e-04
##
## sigma^2 estimated as 0.007165:  log likelihood = 560.01,  aic = -1102.01
##
## Training set error measures:
##                         ME      RMSE        MAE          MPE      MAPE      MASE
## Training set 0.0003310505 0.0846461 0.06163431 -0.008747071 0.8704652 0.5675067
##                      ACF1
## Training set 0.001277254
```

```
ar3=arima(logcars,order=c(3,0,1),xreg = cbind(t, t2),seasonal=list(order=c(1,0,1)))
plot(logcars, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(ar3),col="red")
```

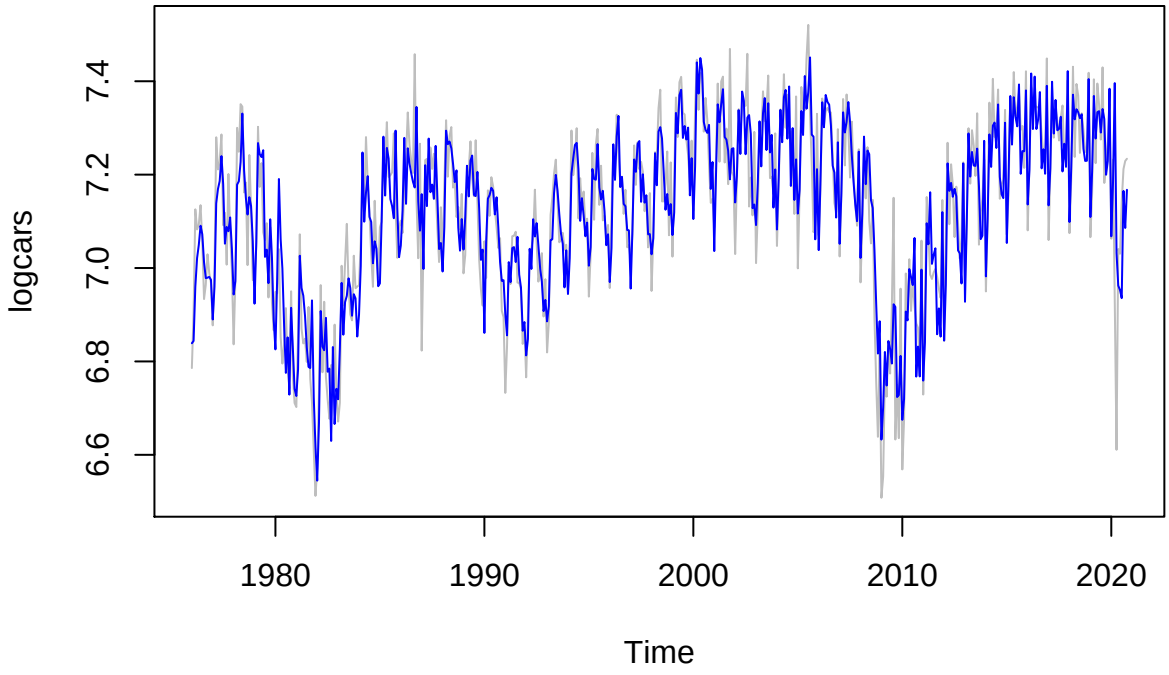



```
summary(ar3)
```

```
##
## Call:
## arima(x = logcars, order = c(3, 0, 1), seasonal = list(order = c(1, 0, 1)),
##     xreg = cbind(t, t2))
##
## Coefficients:
##          ar1      ar2     ar3      ma1     sar1     sma1  intercept       t
##       1.0542  -0.1311  0.0409  -0.5677  0.9783  -0.7387  -426.1685  0.4293
## s.e.  0.1309   0.0853  0.0684   0.1255  0.0100   0.0482   390.3897  0.3871
##           t2
##       -1e-04
## s.e.   1e-04
##
## sigma^2 estimated as 0.006399:  log likelihood = 586.2,  aic = -1152.39
##
## Training set error measures:
##                       ME      RMSE        MAE          MPE      MAPE      MASE
## Training set 0.0006362609 0.0799948 0.05905147 -0.002733308 0.8335184 0.5437249
##                      ACF1
## Training set 0.002527851
```

```
ar4=arima(logcars,order=c(3,0,0),xreg = cbind(t, t2),seasonal=list(order=c(1,0,1)))
plot(logcars, xlab="Time", lwd=1, col='grey')
```

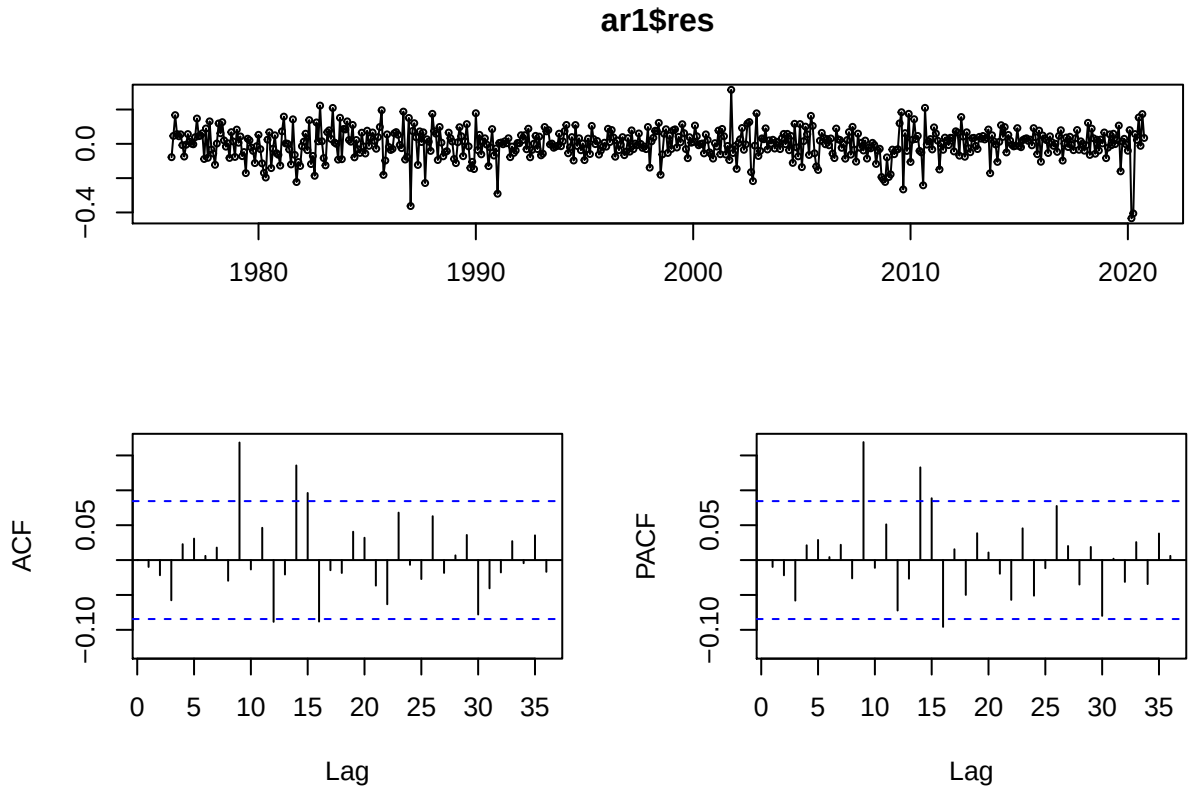
```
lines(fitted(ar4),col="blue")
```



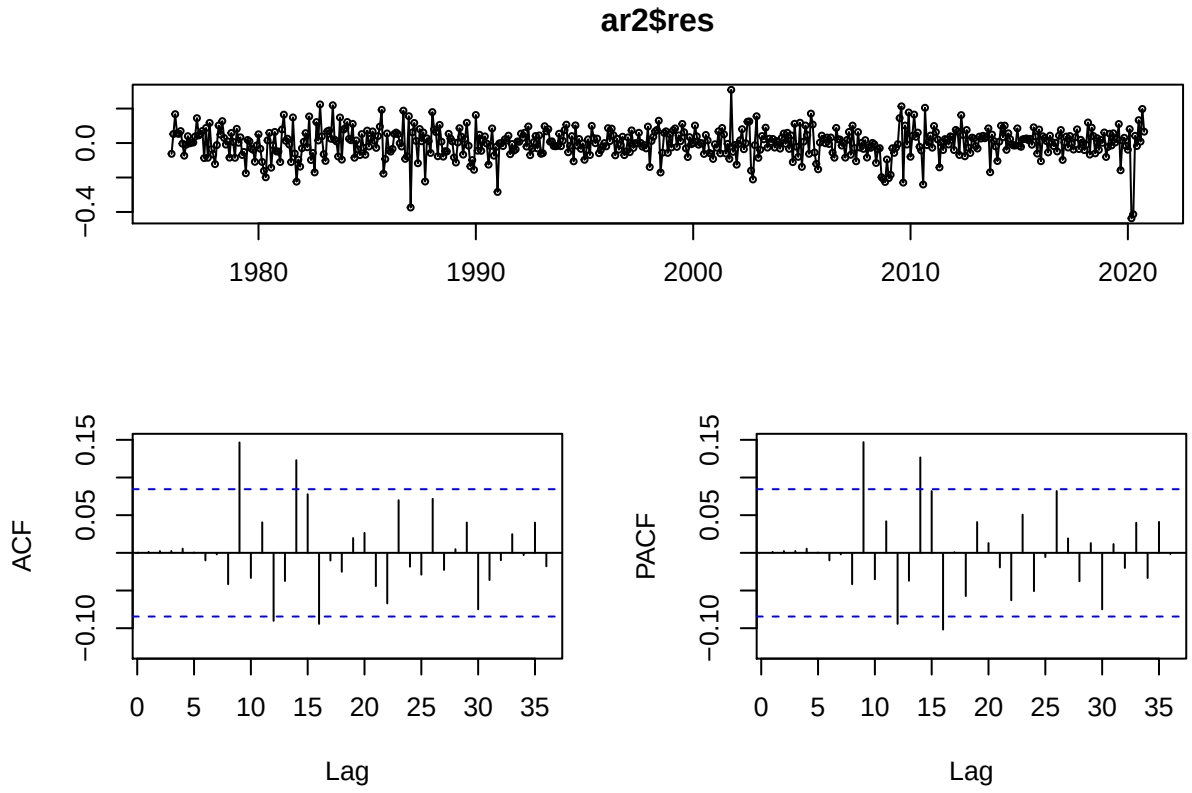

```
summary(ar4)
```

```
##
## Call:
## arima(x = logcars, order = c(3, 0, 0), seasonal = list(order = c(1, 0, 1)),
##     xreg = cbind(t, t2))
##
## Coefficients:
##          ar1     ar2     ar3    sar1     sma1  intercept       t  t2
##       0.5156  0.1744  0.2049  0.9763  -0.7305  -199.5462  0.2020   0
## s.e.  0.0424  0.0472  0.0423  0.0108   0.0497   182.2329  0.1764   0
##
## sigma^2 estimated as 0.006498:  log likelihood = 582.52,  aic = -1147.04
##
## Training set error measures:
##                        ME       RMSE        MAE          MPE      MAPE
## Training set 0.0005637843 0.08061257 0.05977711 -0.004153504 0.8438867
##                    MASE        ACF1
## Training set 0.5504063 -0.01498409
```

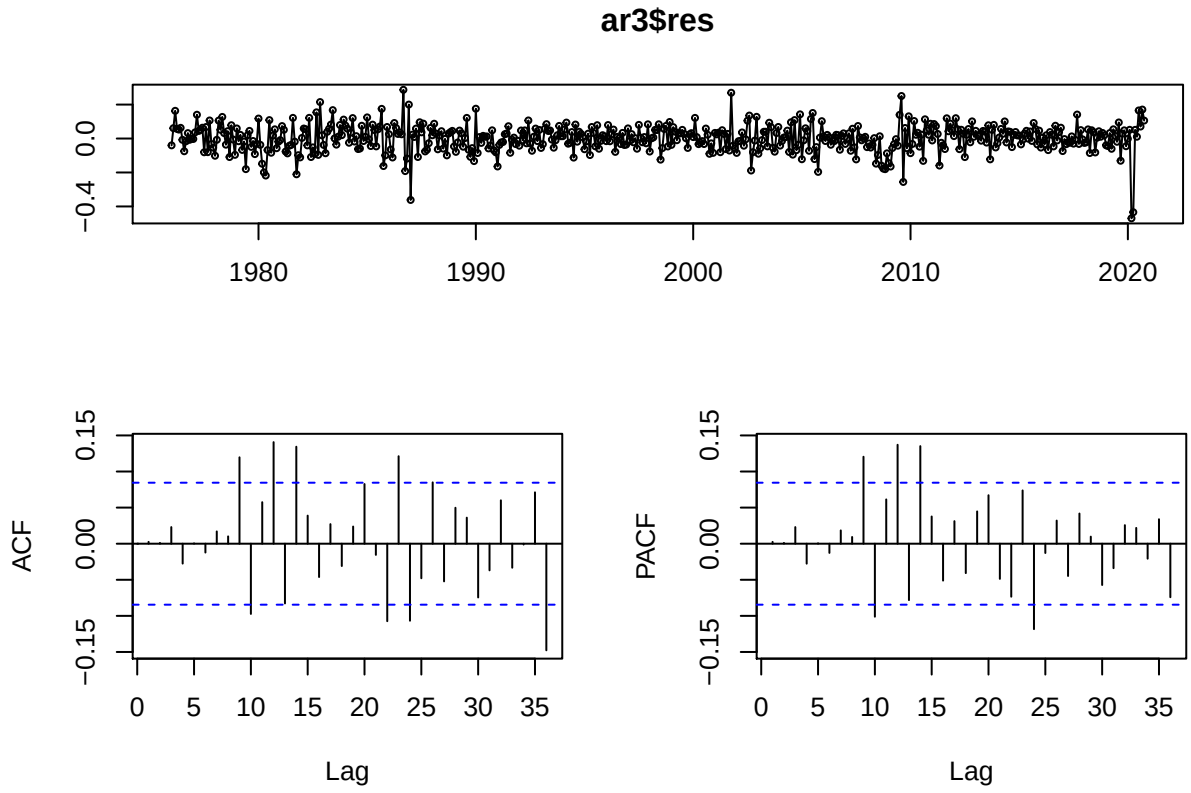
```
tsdisplay(ar1$res)
```



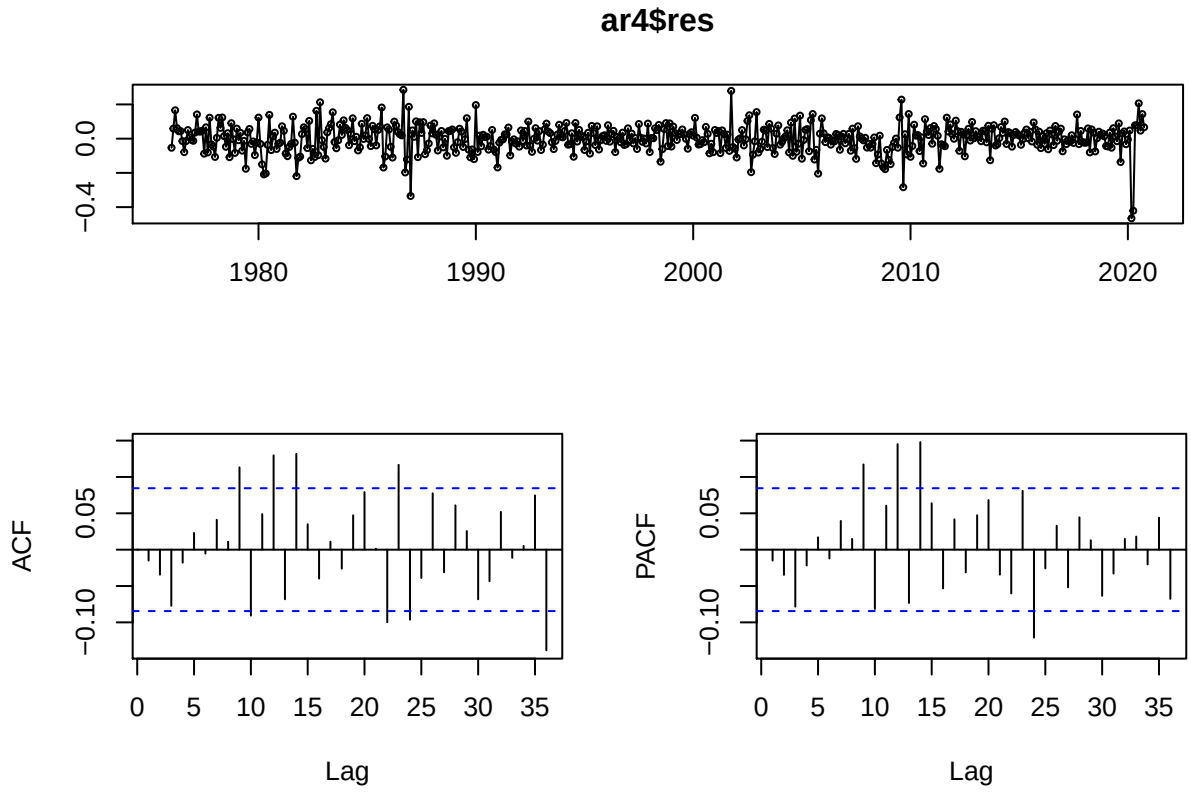

```
tsdisplay(ar2$res)
```




```
tsdisplay(ar3$res)
```




```
tsdisplay(ar4$res)
```




```
AIC(ar1, ar2, ar3, ar4)
```

```
##     df       AIC
## ar1  8 -1098.719
## ar2  9 -1102.012
## ar3 10 -1152.393
## ar4  9 -1147.038
```

```
AIC(ar1, ar2, ar3, ar4,k = log(length(cars_ts)))
```

```
##     df       AIC
## ar1  8 -1064.416
## ar2  9 -1063.422
## ar3 10 -1109.514
## ar4  9 -1108.448
```

```
#from using the trend and season and building up the model without differencing
#we discover that the ar3 model is the best in this group.

Box.test(ar3$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
```

```
## data:  ar3$residuals
## X-squared = 0.003457, df = 1, p-value = 0.9531
```

```
#residuals is not serially correlated from the results of this test
```

After trying difference combinations we think might be useful with guidance from our auto.arima model and our previous residuals, Model AR3 which is ARIMA(3,0,1)(1,0,1)[12] ends up being the best model chosen thorugh AIC and BIC of this method of building up trend and seasonality. The Ljung-Box test of the residuals are not serially correlated.

Next we move on to fitting a non-differentiated model for unemployment.

```
#look at trends for unemployment
ut1 = tslm(une_ts~trend)
summary(ut1)
```

```
##
## Call:
## tslm(formula = une_ts ~ trend)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6001 -1.2227 -0.3889  0.7538  8.8763
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.0616124  0.1459960  48.369  < 2e-16 ***
## trend       -0.0028909  0.0004694  -6.159 1.44e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.691 on 536 degrees of freedom
## Multiple R-squared:  0.06609,    Adjusted R-squared:  0.06435
## F-statistic: 37.93 on 1 and 536 DF,  p-value: 1.437e-09
```

```
plot(une_ts, xlab="Time", lwd=3, col='skyblue3')
lines(t,ut1$fit,col="red3",lwd=1)

ut1.1 = tslm(une_ts~t+sint+cost)
summary(ut1.1)
```

```
##
## Call:
## tslm(formula = une_ts ~ t + sint + cost)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4193 -1.2311 -0.5058  0.8451  8.8504
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 72.486144  11.247984   6.444  2.6e-10 ***
```

```
## t           -0.033129   0.005628  -5.886  7.0e-09 ***
## sint        -0.286128   0.103679  -2.760  0.00598 **
## cost         0.067234   0.101922   0.660  0.50975
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.681 on 534 degrees of freedom
## Multiple R-squared:  0.07989,    Adjusted R-squared:  0.07472
## F-statistic: 15.45 on 3 and 534 DF,  p-value: 1.181e-09
```

```
lines(t,ut1.1$fit,col="green",lwd=1)
t2 = t^2

ut1.3 = tslm(une_ts~t+t2)
summary(ut1.3)
```

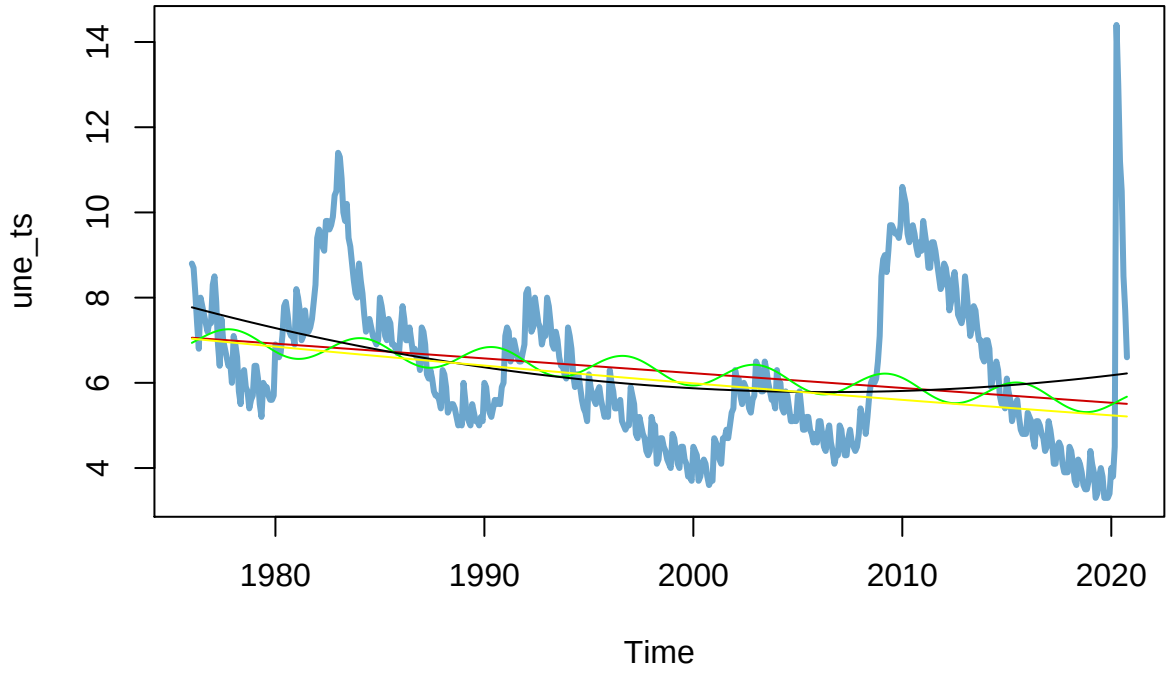
```
##
## Call:
## tslm(formula = une_ts ~ t + t2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8665 -1.1705 -0.3575  0.7037  8.2093
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.639e+03  1.909e+03   4.525 7.45e-06 ***
## t           -8.606e+00  1.911e+00  -4.503 8.21e-06 ***
## t2           2.145e-03  4.781e-04   4.485 8.91e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.661 on 535 degrees of freedom
## Multiple R-squared:  0.09994,    Adjusted R-squared:  0.09657
## F-statistic:  29.7 on 2 and 535 DF,  p-value: 5.861e-13
```

```
lines(t,ut1.3$fit,col="black",lwd=1)

ut1.4  = lm(log(UNRATENSA)~t, data=une)
summary(ut1.4)
```

```
##
## Call:
## lm(formula = log(UNRATENSA) ~ t, data = une)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5040 -0.1821 -0.0336  0.1296  1.0132
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.2163658  1.7073478   8.912  < 2e-16 ***
## t           -0.0067132  0.0008544  -7.858 2.16e-14 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2565 on 536 degrees of freedom
## Multiple R-squared:  0.1033, Adjusted R-squared:  0.1016
## F-statistic: 61.74 on 1 and 536 DF,  p-value: 2.156e-14
```

```
lines(t,exp((as.numeric(ut1.4$coefficients[1]))+(as.numeric(ut1.4$coefficients[2]))*t),col="yellow",lwd=
```




```
#undo exponential
exponential_fit = exp((as.numeric(ut1.4$coefficients[1]))+(as.numeric(ut1.4$coefficients[2]))*t)
calc_res = (une$UNRATENSA-exponential_fit)^2
exp_res = sum(calc_res)/536
exp_res = sqrt(exp_res)
print("Exponential Level RMSE:")
```

```
## [1] "Exponential Level RMSE:"
```

```
print(exp_res)
```

```
## [1] 1.701222
```

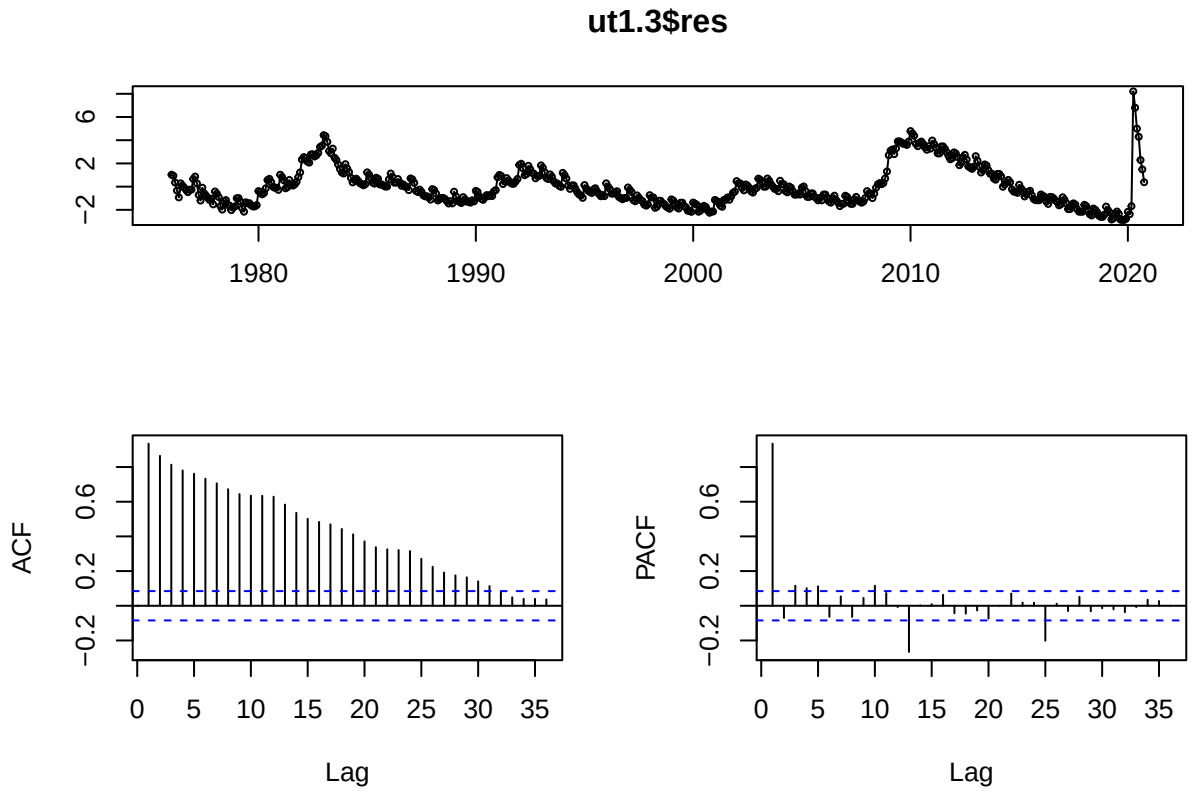
```
AIC(ut1, ut1.1, ut1.3, ut1.4) #quadratic turns out to be best
```

```
##        df        AIC
## ut1     3 2095.90177
## ut1.1   5 2091.89717
## ut1.3   4 2078.04336
## ut1.4   3   66.61474
```

We do the same thing for unemployment and look at different linear functions of time. From the results and looking at AIC for these trend models, the quadratic has the lowest AIC = 2078.04336 besides the exponential model, which is only very low because the scale of y variable is different. However if we look at RMSE = 1.701222 for exponential model it's a bit higher than the quadratic model RMS = 1.661, so we choose to go with the quadratic.

We look at residuals of the fitted trend model:

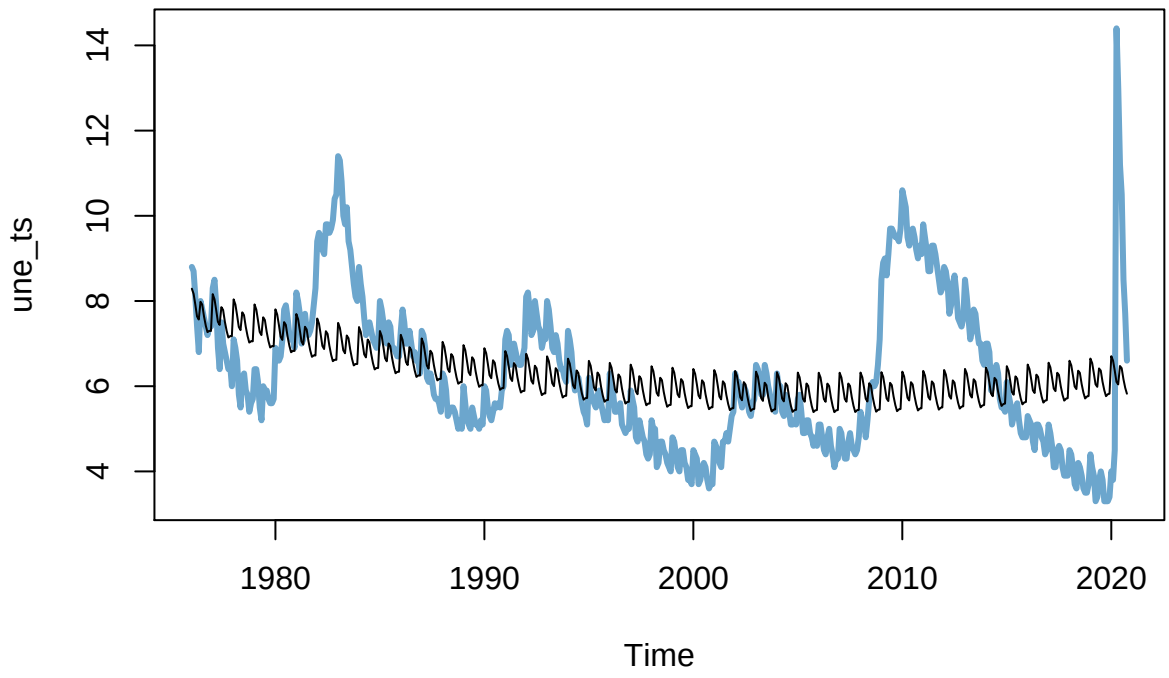
```
#look at residuals of trend
tsdisplay(ut1.3$res) #might be S-AR(2) since there are spikes at factors of 12
```



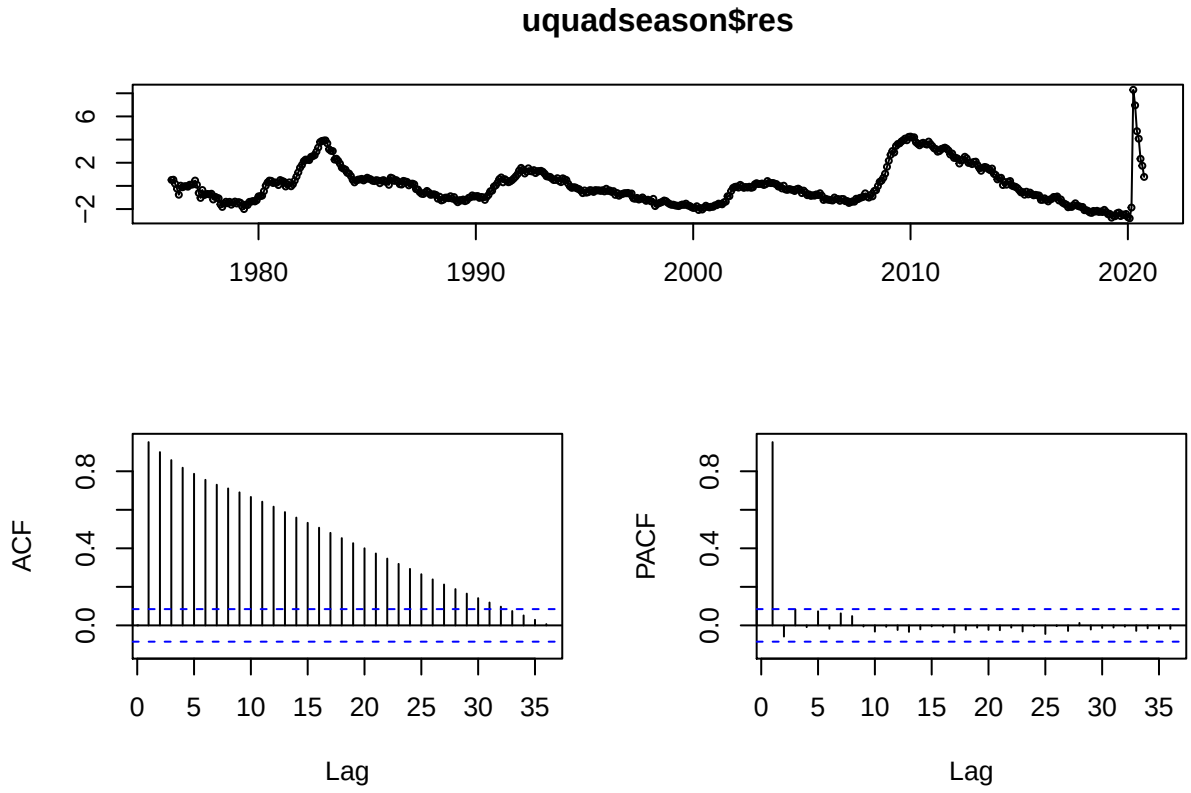

```
#in the PACF

#look at residuals with season dummies
uquadseason = tslm(une_ts~t+t2+season)
plot(une_ts, xlab="Time", lwd=3, col='skyblue3')
summary(uquadseason)
```

```
##
## Call:
## tslm(formula = une_ts ~ t + t2 + season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7993 -1.1666 -0.3464  0.5772  8.2988
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.563e+03  1.897e+03   4.515 7.84e-06 ***
## t           -8.529e+00  1.898e+00  -4.493 8.65e-06 ***
## t2           2.125e-03  4.750e-04   4.475 9.39e-06 ***
## season2     -1.103e-01  3.479e-01  -0.317  0.75135
## season3     -3.431e-01  3.479e-01  -0.986  0.32448
## season4     -6.179e-01  3.479e-01  -1.776  0.07630 .
## season5     -6.817e-01  3.479e-01  -1.959  0.05059 .
## season6     -2.543e-01  3.479e-01  -0.731  0.46507
## season7     -3.093e-01  3.479e-01  -0.889  0.37436
## season8     -5.753e-01  3.479e-01  -1.654  0.09878 .
## season9     -7.814e-01  3.479e-01  -2.246  0.02512 *
## season10    -9.254e-01  3.479e-01  -2.660  0.00806 **
## season11    -8.832e-01  3.499e-01  -2.524  0.01189 *
## season12    -8.759e-01  3.499e-01  -2.503  0.01261 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.65 on 524 degrees of freedom
## Multiple R-squared:  0.1303, Adjusted R-squared:  0.1087
## F-statistic:  6.04 on 13 and 524 DF,  p-value: 1.511e-10
```

```r
lines(t,uquadseason$fit,col="black",lwd=1)
```




```
tsdisplay(uquadseason$res)
```

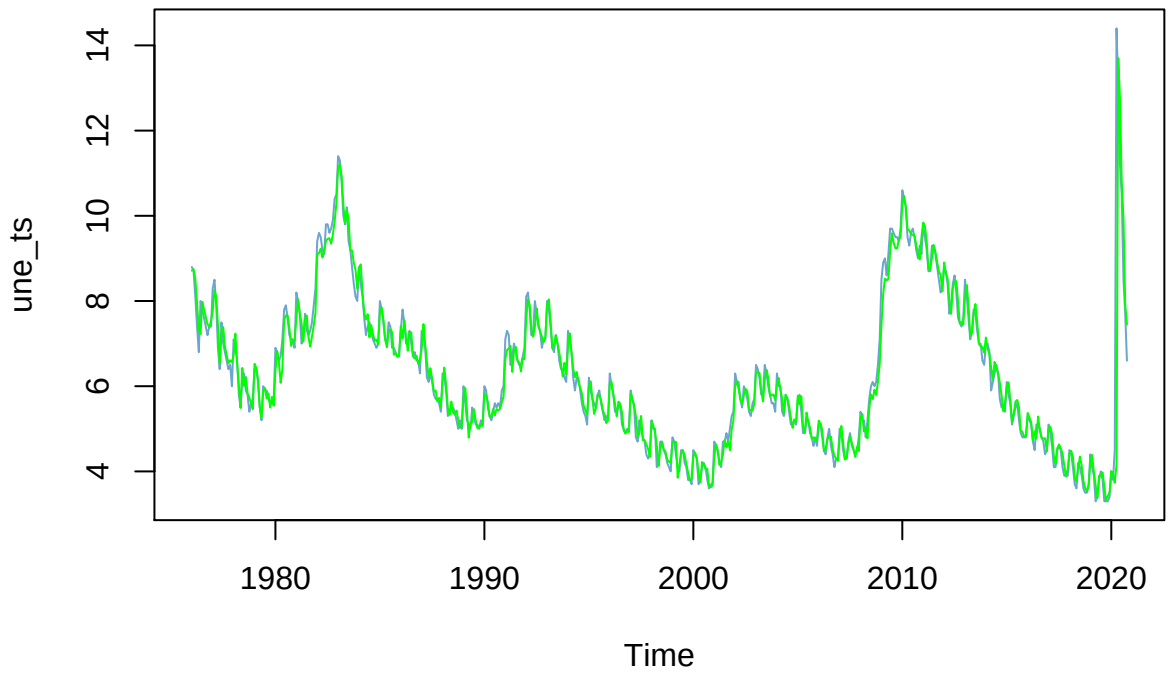



```
#since spikes in 1 persist in PACF we try to fit AR(1) for cycles
```

Comments of the ACF and PACF displayed are in the code too. When we look at the ACF and PACF of the residuals after fitting a quadratic trend, we see that there might be S-AR(2) since there are spikes at factors of 12 in the PACF. When we look at the residuals with seasonal dummies we see that a spike still persist in the PACF at 1 so this could indicate an AR(1) component to the model. We deduce that there is S-AR(2) and AR(1) as possible components.

Below we combine the trend component with ARIMA for unemployment.

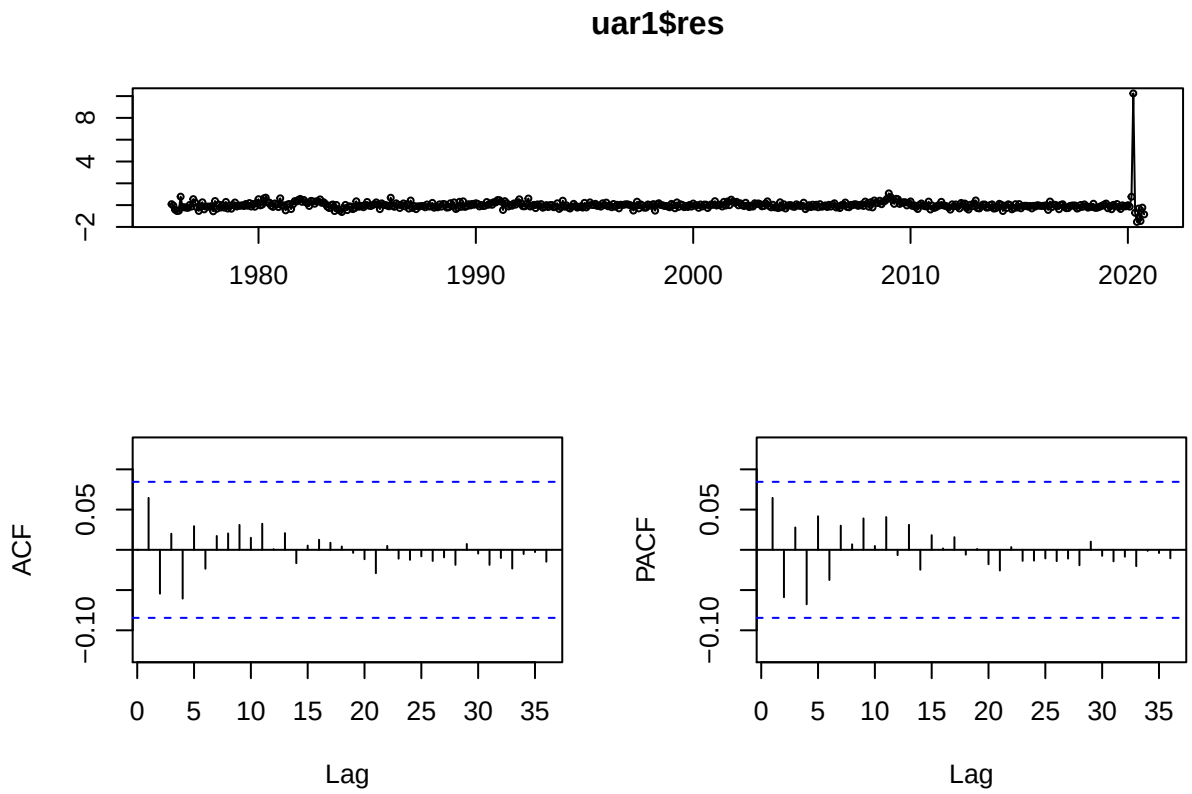
```
#we try to fit ar models without differencing (using trend)

#we had to transform the time variables due to the computational limits of
#the ARIMA function
at = t-1976
at2 = at^2

uar1=arima(une_ts,order=c(1,0,0),xreg = cbind(at, at2),seasonal=list(order=c(2,0,0)))
plot(une_ts, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(uar1),col="green")
```



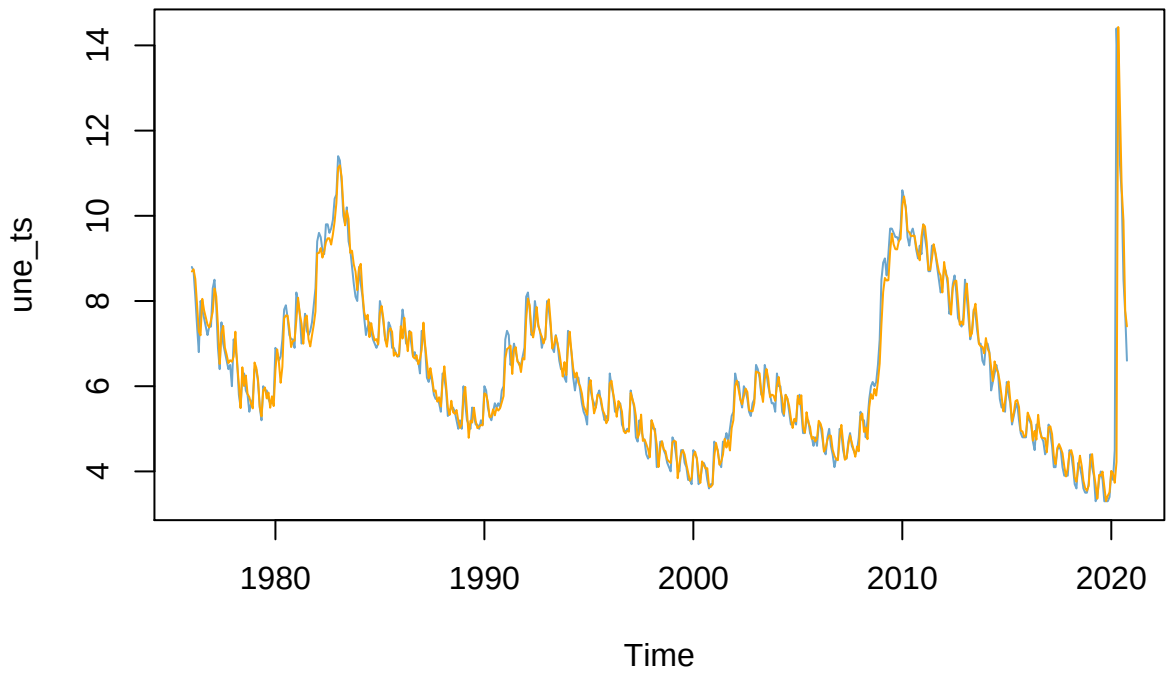

```
summary(uar1)
```

```
##
## Call:
## arima(x = une_ts, order = c(1, 0, 0), seasonal = list(order = c(2, 0, 0)), xreg = cbind(at,
##     at2))
##
## Coefficients:
##          ar1    sar1    sar2  intercept       at     at2
##       0.9257  0.4611  0.2824     8.3771  -0.2323  0.0048
## s.e.  0.0169  0.0909  0.0946     2.2119   0.2216  0.0047
##
## sigma^2 estimated as 0.2601:  log likelihood = -406.52,  aic = 827.04
##
## Training set error measures:
##                        ME      RMSE       MAE        MPE     MAPE     MASE
## Training set -0.000698526 0.5099691 0.2032364 -0.3599391 3.078243 0.625074
##                    ACF1
## Training set 0.0645737
```

```
tsdisplay(uar1$res)
```



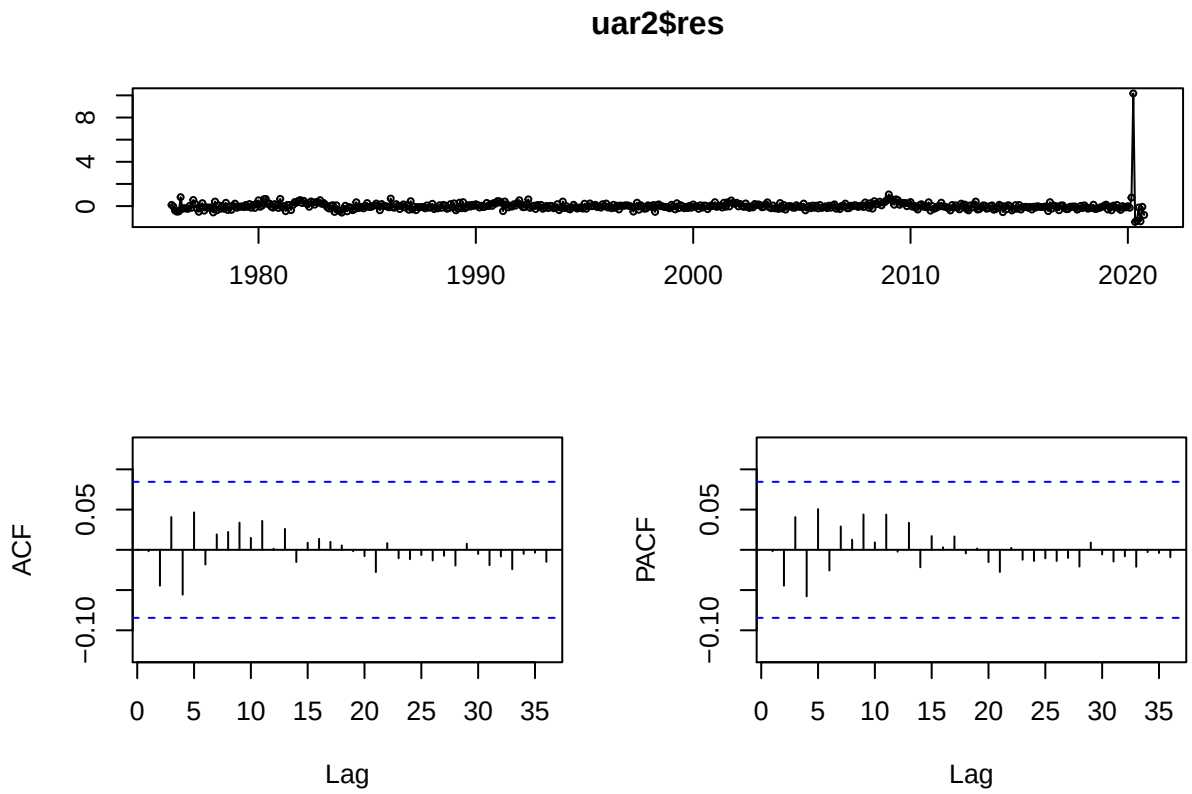

```
uar2=arima(une_ts,order=c(1,0,1),xreg = cbind(at, at2),seasonal=list(order=c(2,0,0)))
plot(une_ts, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(uar2),col="orange")
```



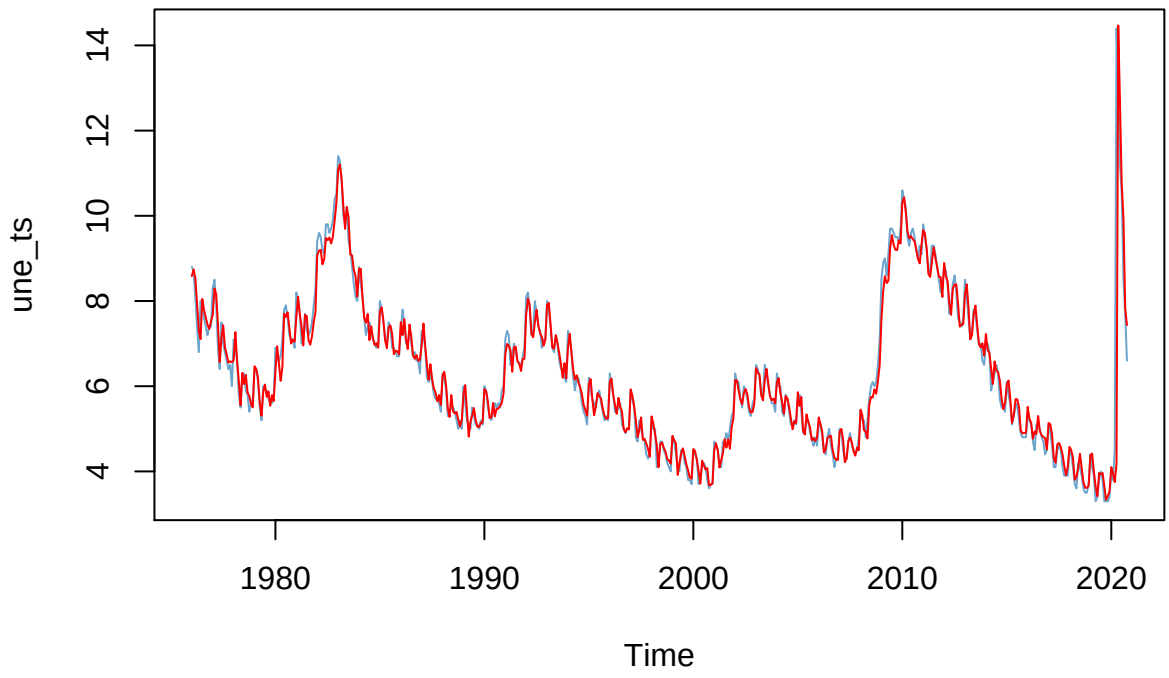

```
summary(uar2)
```

```
##
## Call:
## arima(x = une_ts, order = c(1, 0, 1), seasonal = list(order = c(2, 0, 0)), xreg = cbind(at,
##     at2))
##
## Coefficients:
##          ar1     ma1    sar1    sar2  intercept       at     at2
##       0.9140  0.0837  0.4564  0.2873     8.3078  -0.2175  0.0044
## s.e.  0.0199  0.0499  0.0911  0.0951     2.0873   0.2092  0.0045
##
## sigma^2 estimated as 0.2588:  log likelihood = -405.12,  aic = 826.24
##
## Training set error measures:
##                        ME      RMSE       MAE       MPE     MAPE      MASE
## Training set -0.001343354 0.5086779 0.2049554 -0.381165 3.110022 0.6303612
##                      ACF1
## Training set -0.001737998
```

```
tsdisplay(uar2$res)
```



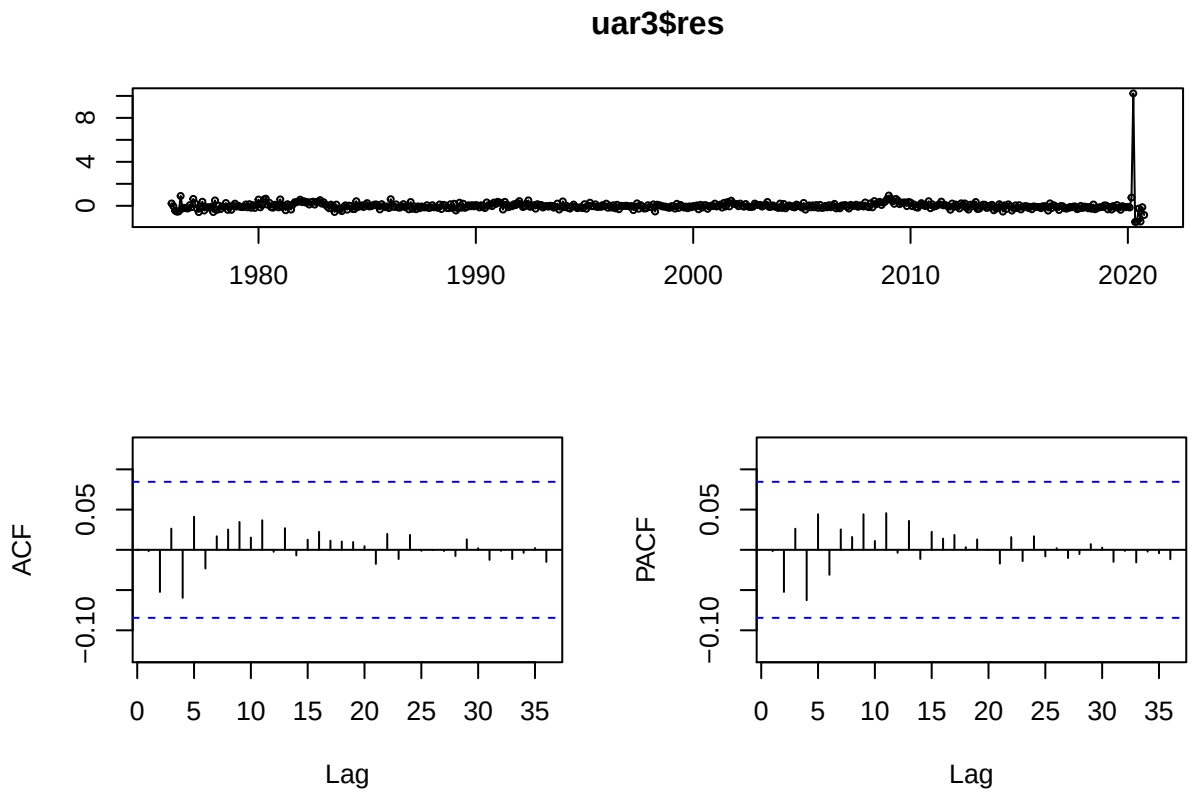

```
uar3=arima(une_ts,order=c(1,0,1),xreg = cbind(at, at2),seasonal=list(order=c(2,0,1)))
plot(une_ts, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(uar3),col="red")
```



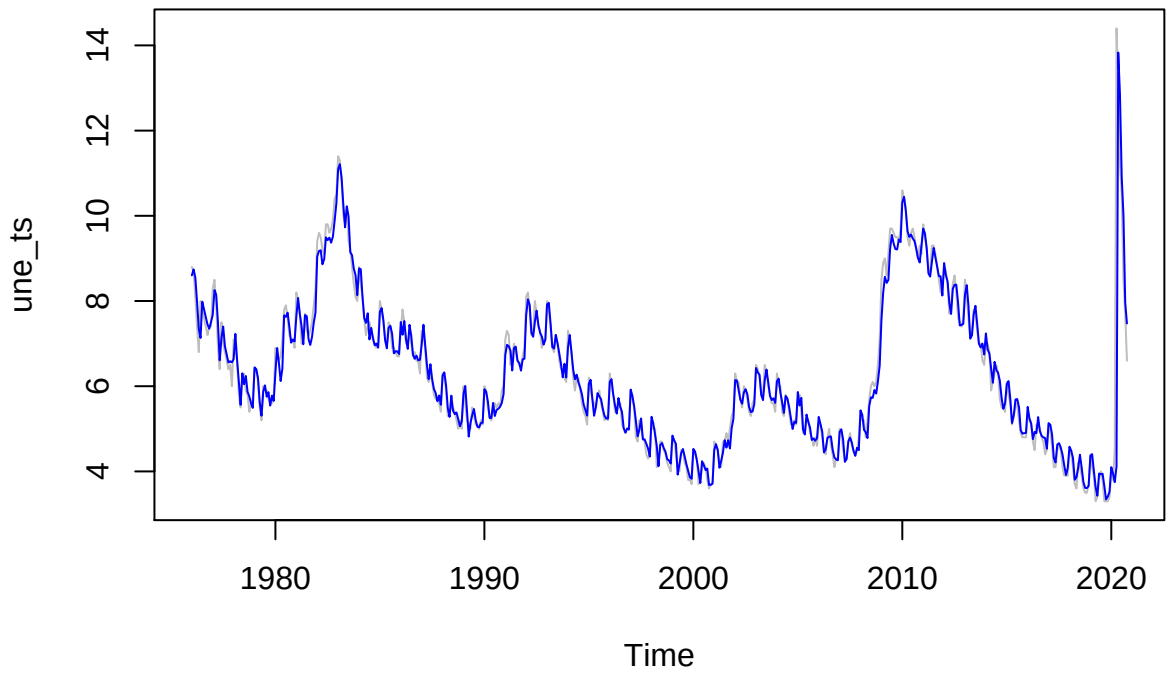

```
summary(uar3)
```

```
##
## Call:
## arima(x = une_ts, order = c(1, 0, 1), seasonal = list(order = c(2, 0, 1)), xreg = cbind(at,
##     at2))
##
## Coefficients:
##          ar1     ma1    sar1     sar2     sma1  intercept       at     at2
##       0.9237  0.0712  1.2064  -0.2365  -0.8240     7.8522  -0.1494  0.0027
## s.e.  0.0209  0.0498  0.1952   0.1739   0.1329     1.9969   0.1659  0.0035
##
## sigma^2 estimated as 0.2569:  log likelihood = -403.24,  aic = 824.47
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE    MAPE      MASE
## Training set -0.00182827 0.5068746 0.1970543 -0.4534406 2.98626 0.6060606
##                      ACF1
## Training set -0.001691921
```

```
tsdisplay(uar3$res)
```



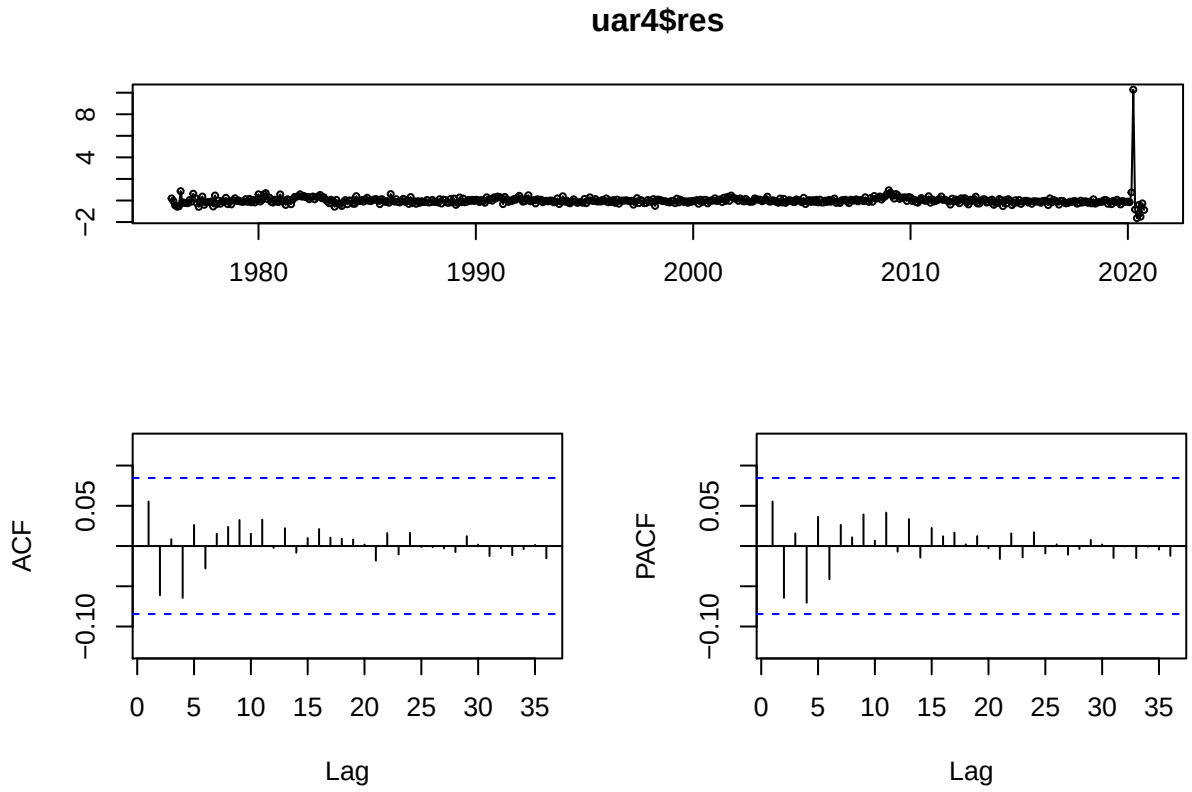

```
uar4=arima(une_ts,order=c(1,0,0),xreg = cbind(at, at2),seasonal=list(order=c(2,0,1)))
plot(une_ts, xlab="Time", lwd=1, col='grey')
lines(fitted(uar4),col="blue")
```




```
summary(uar4)
```

```
##
## Call:
## arima(x = une_ts, order = c(1, 0, 0), seasonal = list(order = c(2, 0, 1)), xreg = cbind(at,
##     at2))
##
## Coefficients:
##          ar1    sar1     sar2     sma1  intercept       at     at2
##       0.9332  1.2063  -0.2349  -0.8294     7.8922  -0.1565  0.0029
## s.e.  0.0177  0.1782   0.1607   0.1176     2.1037   0.1723  0.0036
##
## sigma^2 estimated as 0.2578:  log likelihood = -404.26,  aic = 824.51
##
## Training set error measures:
##                        ME      RMSE      MAE        MPE    MAPE      MASE
## Training set -0.001886045 0.5077624 0.194955 -0.4451893 2.95013 0.5996038
##                     ACF1
## Training set 0.05536119
```

```
tsdisplay(uar4$res)
```




```
AIC(uar1, uar2, uar3, uar4) #uar3 has the lowest AIC score
```

```
##      df      AIC
## uar1  7 827.0385
## uar2  8 826.2423
## uar3  9 824.4742
## uar4  8 824.5115
```

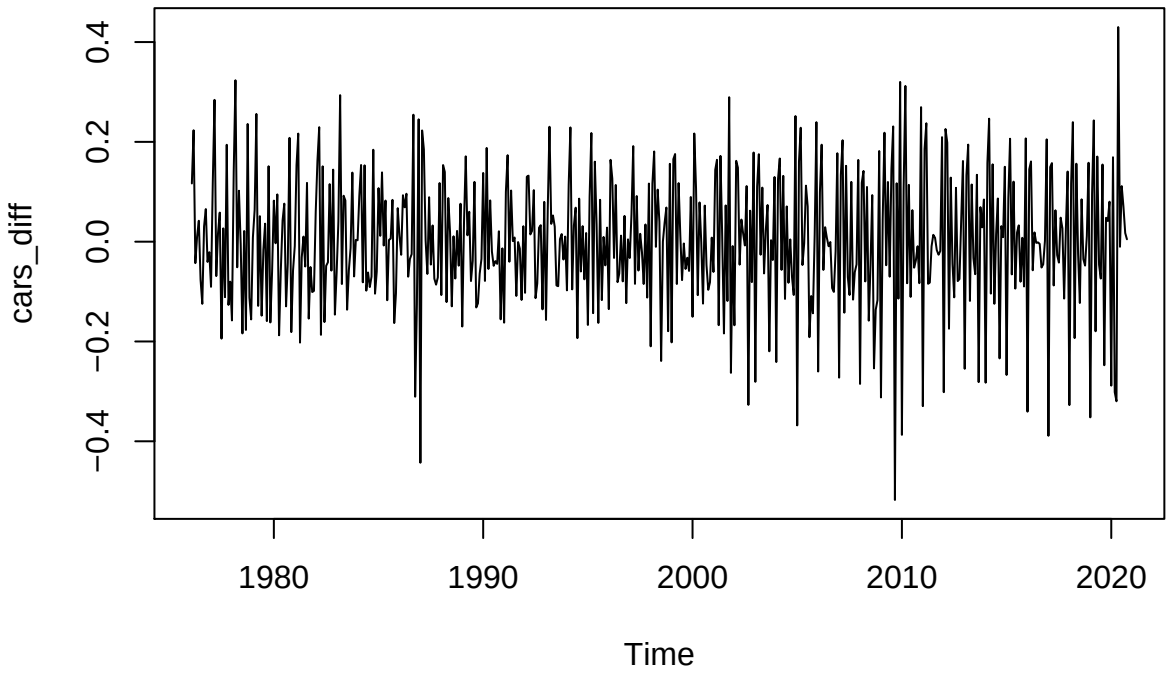
```
Box.test(uar3$res) #residuals are not serially correlated
```

```
##
##  Box-Pierce test
##
## data:  uar3$res
## X-squared = 0.0015401, df = 1, p-value = 0.9687
```

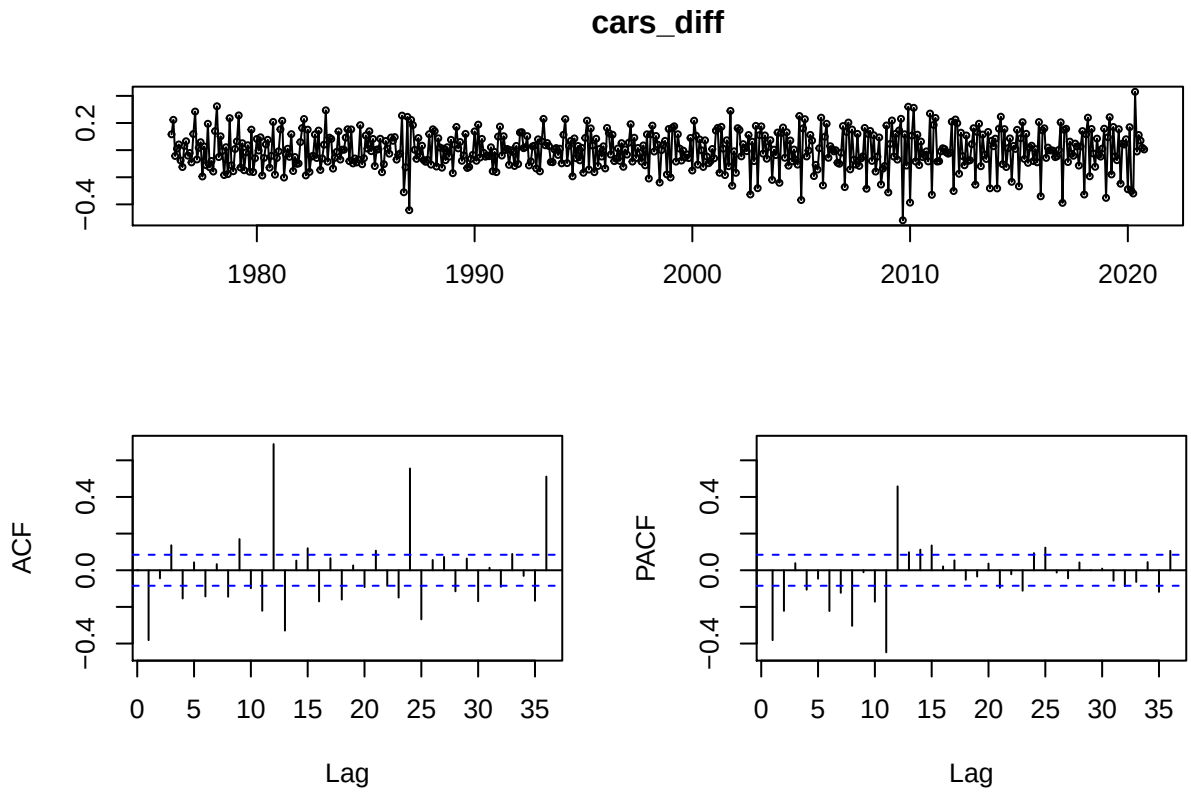
After trying out combinations with guidance from auto.arima model and what we learned in class, using AIC we get that UAR3 is the best model. uar3's AIC = 824.4742. The Ljung Box test we find the residuals are not serially correlated. The best model is therefore ARIMA(1,0,1)(2,0,1) with trend of t+t^2.

Below we will use a differenced model instead and then compare the results that we got with the trend/undifferenced model

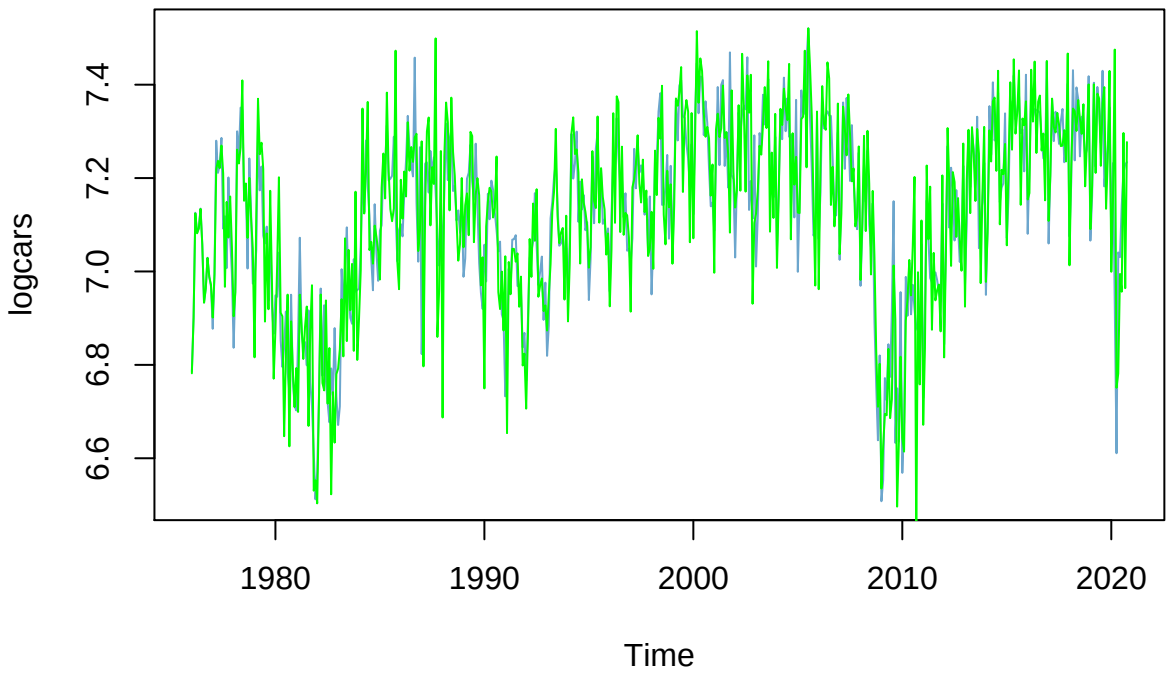
```r
#look at ACF and PACF of differenced residuals
logcars = log(cars_ts)
cars_diff = diff(logcars)
plot(cars_diff)
```




```r
tsdisplay(cars_diff) #looking at this ACF and PACF there is definitely
```



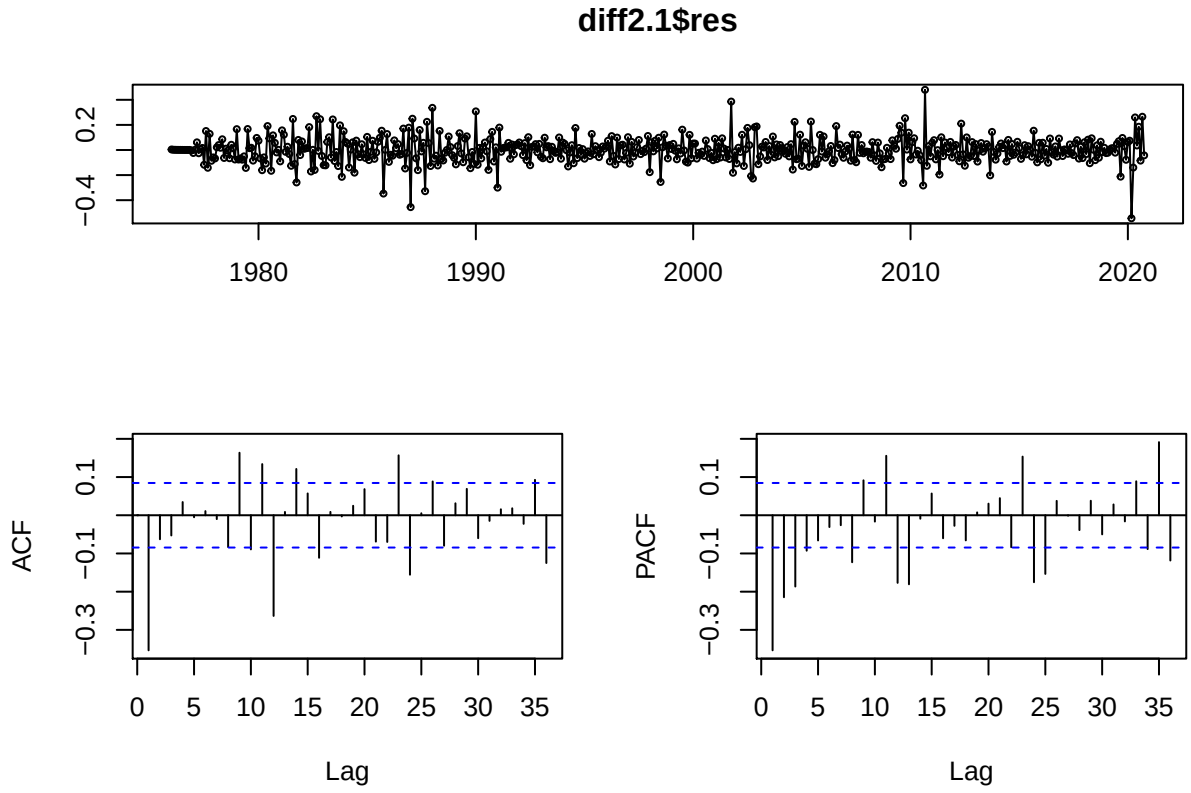

```
#seasonality that needs to be taken care of

#Since the trend is rather hard to identify we also try fitting a differened arima
#model. We include seasonal differencing as well since we have a strong
#seasonal pattern.

diff2.1=arima(logcars,order=c(0,1,0),seasonal=list(order=c(0,1,0)))
plot(logcars, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(diff2.1),col="green")
```




```
summary(diff2.1)
```

```
##
## Call:
## arima(x = logcars, order = c(0, 1, 0), seasonal = list(order = c(0, 1, 0)))
##
##
## sigma^2 estimated as 0.01082:  log likelihood = 443.23,  aic = -884.46
##
## Training set error measures:
##                         ME      RMSE        MAE          MPE     MAPE      MASE
## Training set -0.0001925617 0.1027601 0.07429065 -0.006944572 1.047855 0.6840418
##                     ACF1
## Training set -0.3534869
```

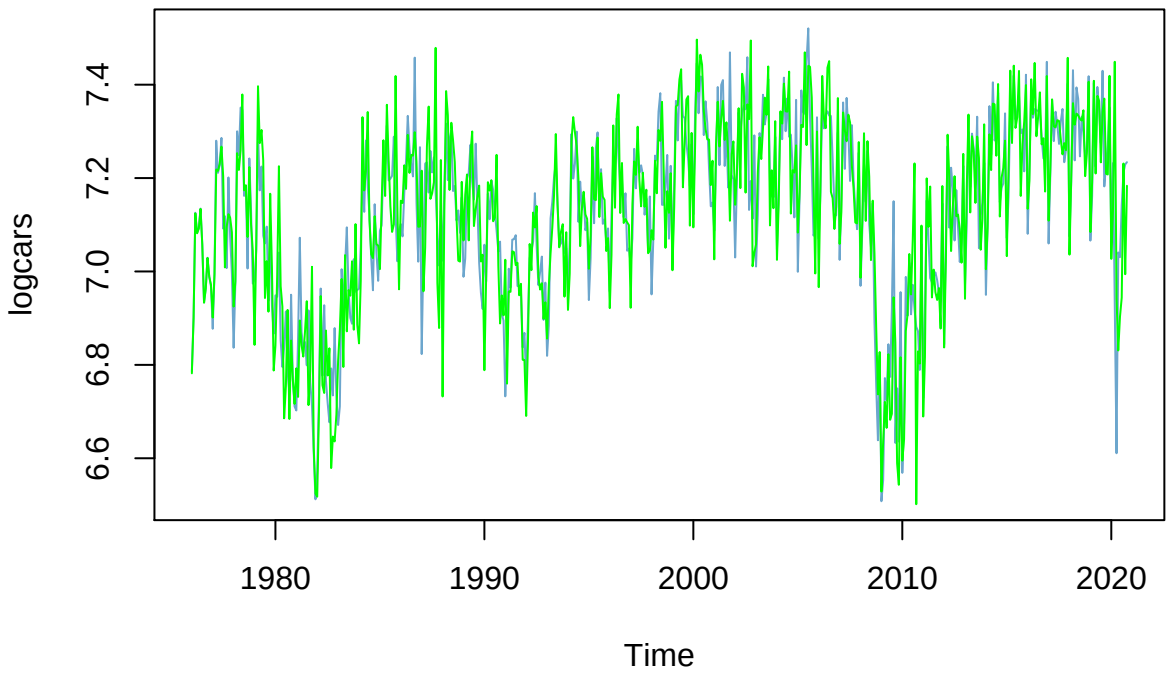
```
tsdisplay(diff2.1$res)
```




```
Box.test(diff2.1$residuals, type = "Ljung-Box") #does not pass Ljung Box test
```

```
##
##  Box-Ljung test
##
## data:  diff2.1$residuals
## X-squared = 67.6, df = 1, p-value = 2.22e-16
```

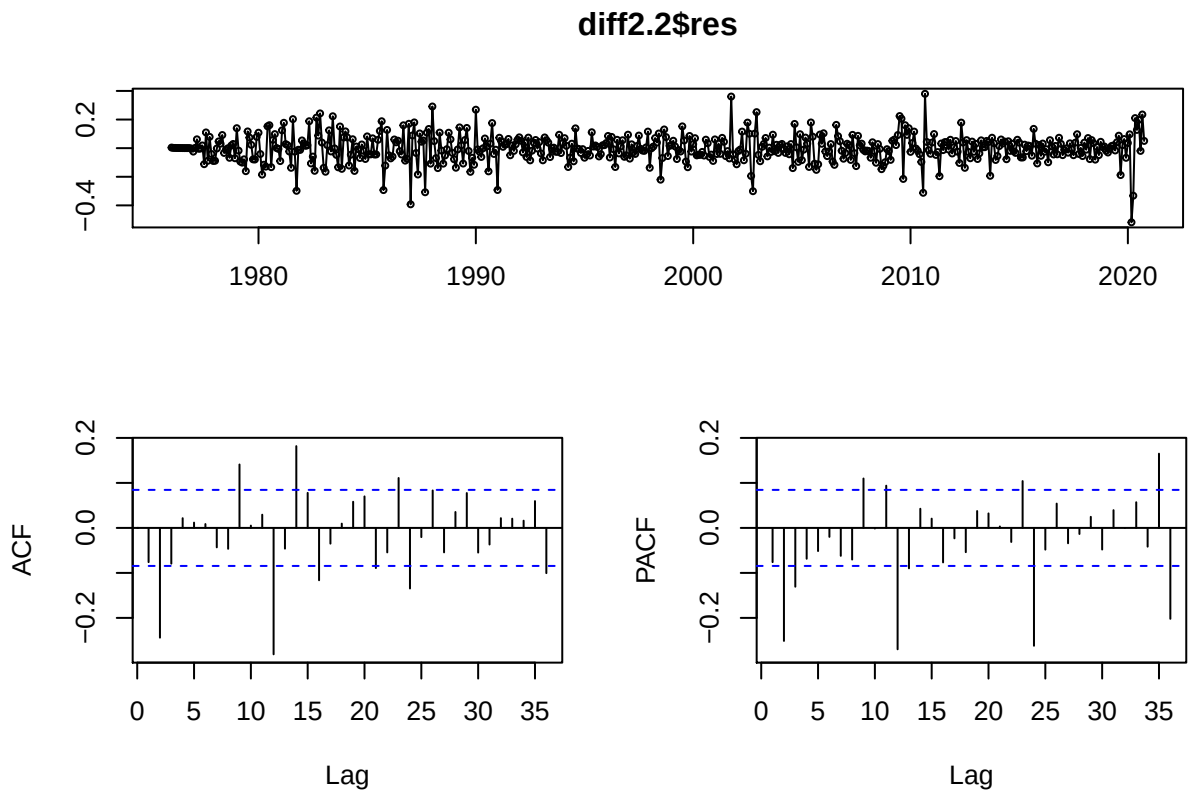
```
#we try adding AR(1) to the differenced model since spike in PACF for last model
diff2.2=arima(logcars,order=c(1,1,0),seasonal=list(order=c(0,1,0)))
plot(logcars, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(diff2.2),col="green")
```




```
summary(diff2.2)
```

```
##
## Call:
## arima(x = logcars, order = c(1, 1, 0), seasonal = list(order = c(0, 1, 0)))
##
## Coefficients:
##           ar1
##       -0.3530
## s.e.   0.0408
##
## sigma^2 estimated as 0.009467:  log likelihood = 478.22,  aic = -952.44
##
## Training set error measures:
##                         ME       RMSE       MAE         MPE      MAPE      MASE
## Training set -0.0002219065 0.09612261 0.0693568 -0.00814911 0.9783577 0.6386126
##                      ACF1
## Training set -0.07603501
```

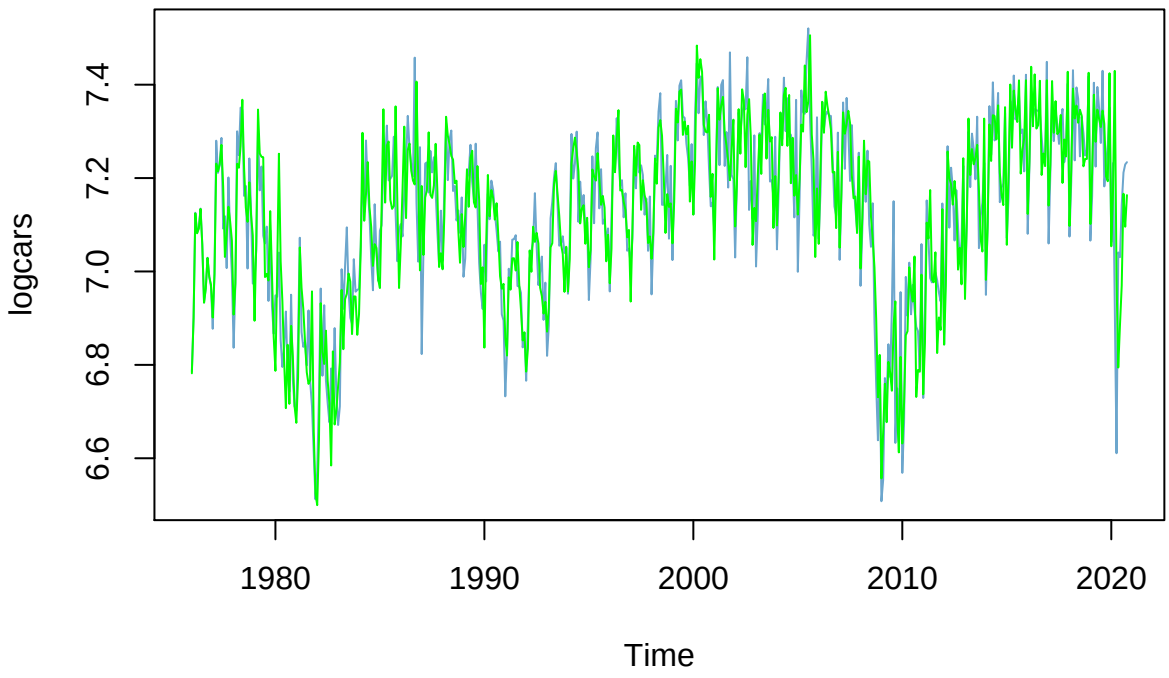
```
tsdisplay(diff2.2$res)
```




```
Box.test(diff2.2$residuals, type = "Ljung-Box") #doesn't pass at 10%
```

```
##
##  Box-Ljung test
##
## data:  diff2.2$residuals
## X-squared = 3.1277, df = 1, p-value = 0.07697
```

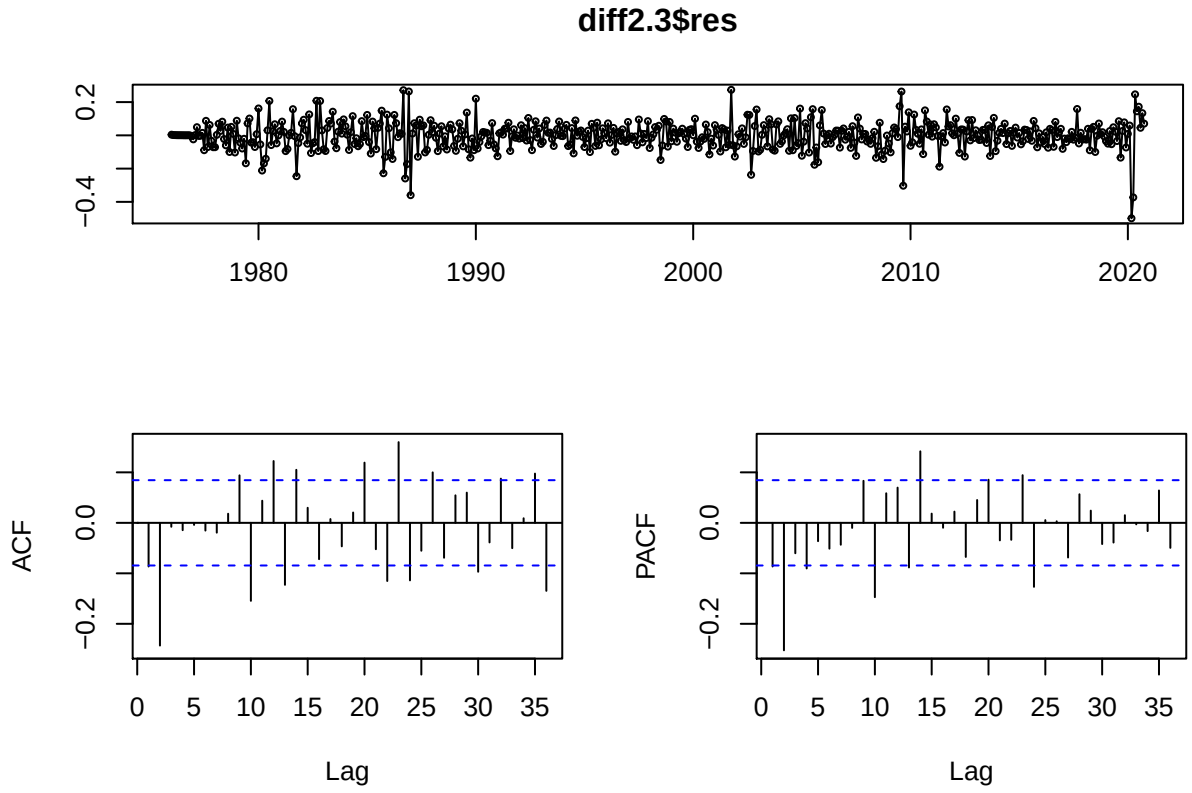
```
#since there is strong spike in ACF at 12 we try adding MA to
#seasonality
diff2.3=arima(logcars,order=c(1,1,0),seasonal=list(order=c(0,1,1)))
plot(logcars, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(diff2.3),col="green")
```




```
summary(diff2.3)
```

```
##
## Call:
## arima(x = logcars, order = c(1, 1, 0), seasonal = list(order = c(0, 1, 1)))
##
## Coefficients:
##           ar1     sma1
##       -0.3539  -0.7567
## s.e.   0.0409   0.0376
##
## sigma^2 estimated as 0.007194:  log likelihood = 545.19,  aic = -1084.38
##
## Training set error measures:
##                         ME       RMSE        MAE         MPE     MAPE      MASE
## Training set -0.0004564484 0.08379489 0.06078584 -0.01371992 0.856558 0.5596943
##                      ACF1
## Training set -0.08650328
```

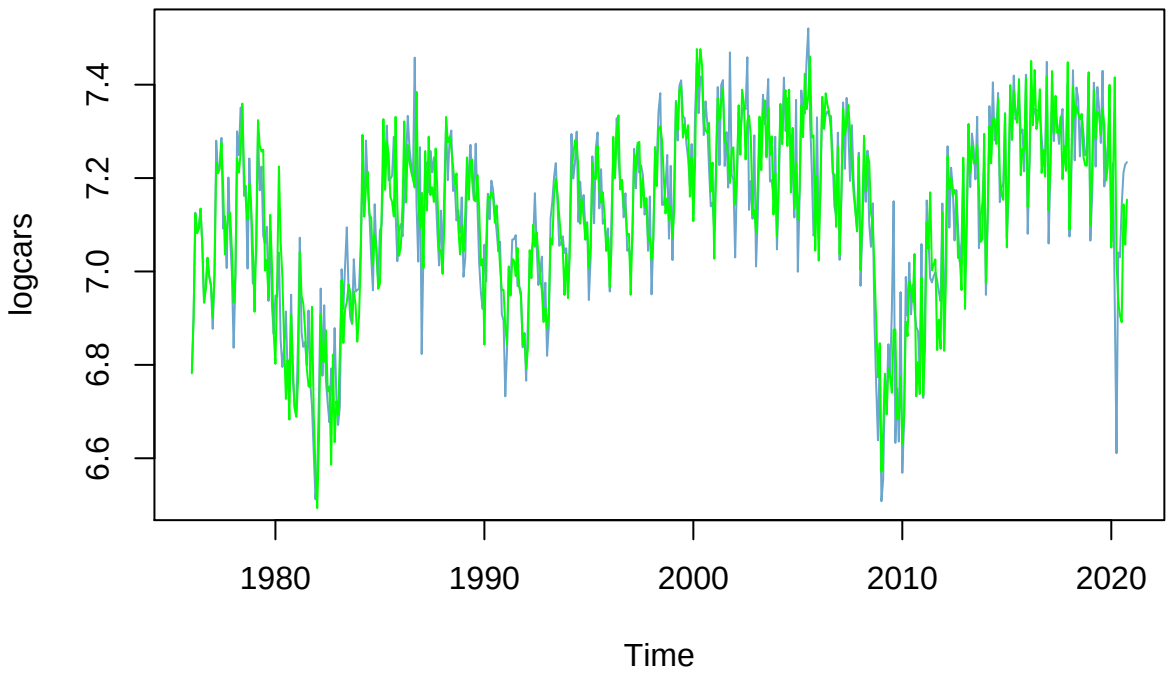
```
tsdisplay(diff2.3$res)
```




```
Box.test(diff2.3$residuals, type = "Ljung-Box") #doesn't pass test
```

```
##
##  Box-Ljung test
##
## data:  diff2.3$residuals
## X-squared = 4.0482, df = 1, p-value = 0.04422
```

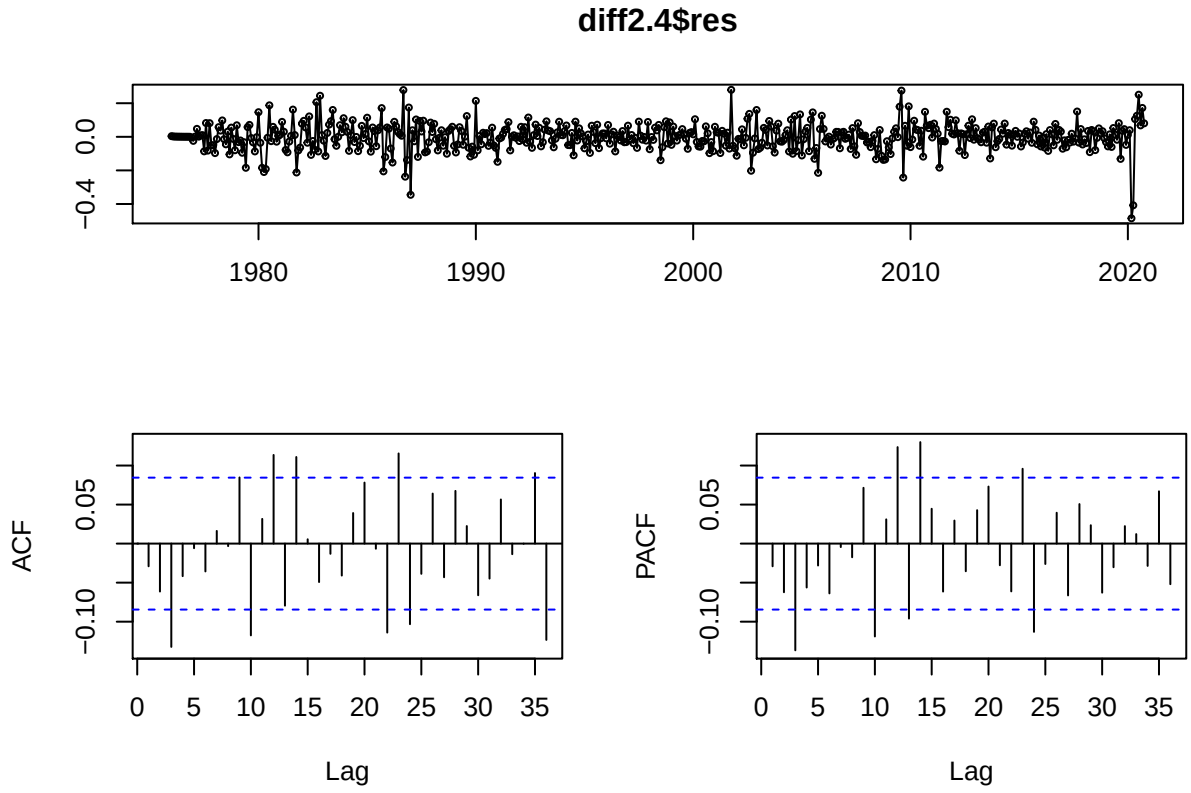
```
#we change to AR(2) because of a strong spike in PACF at 2 in last model
diff2.4=arima(logcars,order=c(2,1,0),seasonal=list(order=c(0,1,1)))
plot(logcars, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(diff2.4),col="green")
```




```
summary(diff2.4)
```

```
##
## Call:
## arima(x = logcars, order = c(2, 1, 0), seasonal = list(order = c(0, 1, 1)))
##
## Coefficients:
##           ar1      ar2     sma1
##       -0.4398  -0.2436  -0.7669
## s.e.   0.0424   0.0424   0.0371
##
## sigma^2 estimated as 0.006762:  log likelihood = 561.18,  aic = -1114.36
##
## Training set error measures:
##                        ME       RMSE        MAE         MPE      MAPE     MASE
## Training set -0.0006576228 0.08123724 0.05849043 -0.01657129 0.8249399 0.538559
##                     ACF1
## Training set -0.0290561
```

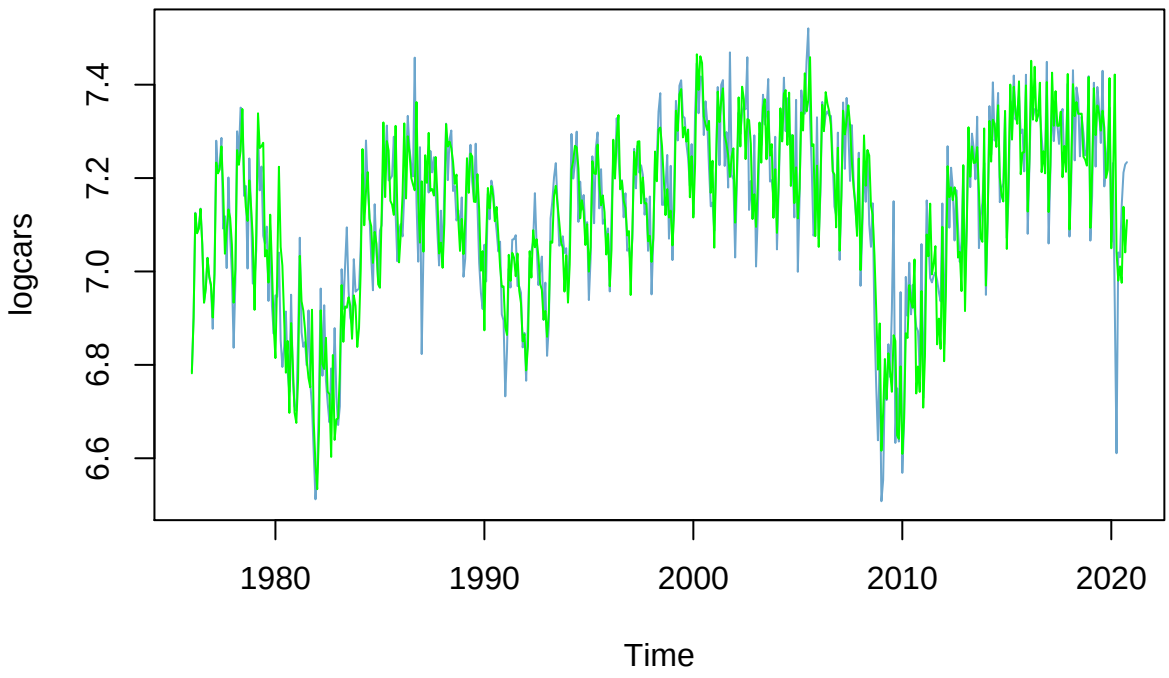
```
tsdisplay(diff2.4$res)
```




```
Box.test(diff2.4$residuals, type = "Ljung-Box") #passes test
```

```
##
##  Box-Ljung test
##
## data:  diff2.4$residuals
## X-squared = 0.45675, df = 1, p-value = 0.4991
```

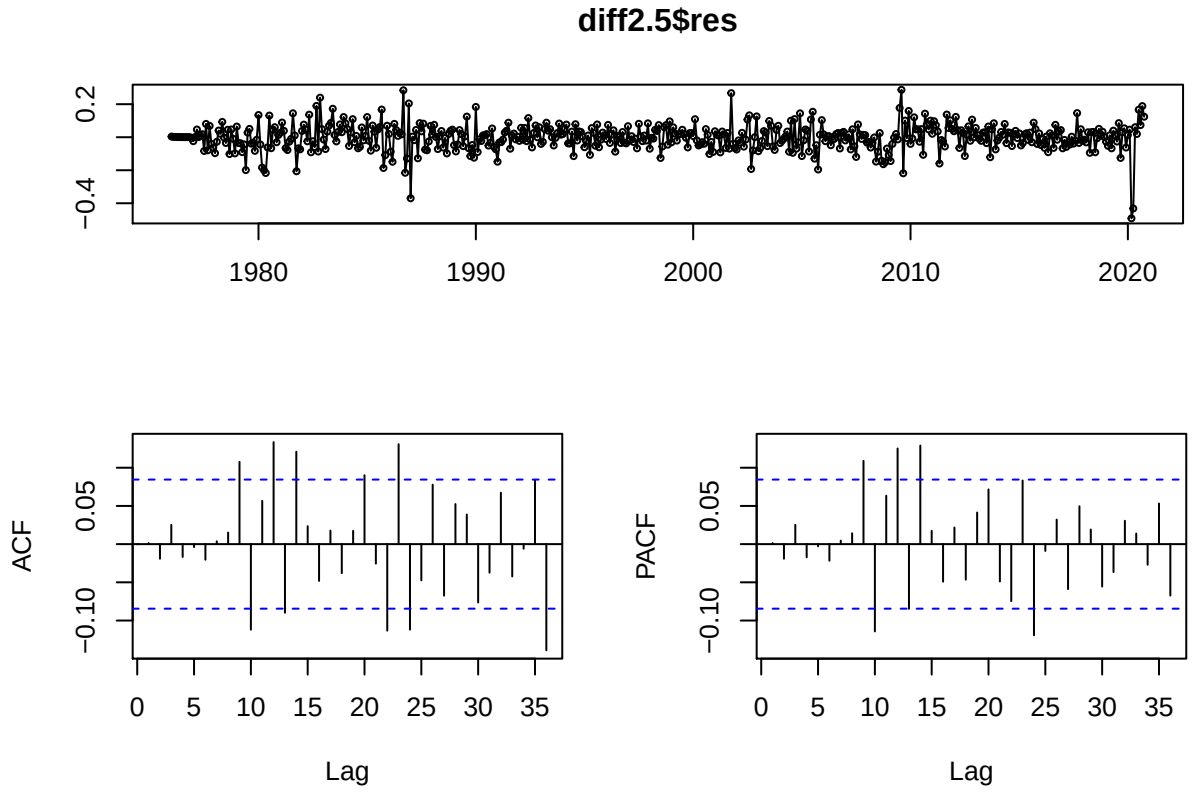
```
#we try AR(1) and MA(1) as well becuase our results from auto.arima indicate
#there is MA component
diff2.5=arima(logcars,order=c(1,1,1),seasonal=list(order=c(0,1,1)))
plot(logcars, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(diff2.5),col="green")
```




```
summary(diff2.5)
```

```
##
## Call:
## arima(x = logcars, order = c(1, 1, 1), seasonal = list(order = c(0, 1, 1)))
##
## Coefficients:
##          ar1      ma1     sma1
##       0.1727  -0.6674  -0.7677
## s.e.  0.0742   0.0554   0.0381
##
## sigma^2 estimated as 0.006567:  log likelihood = 568.77,  aic = -1129.54
##
## Training set error measures:
##                       ME       RMSE        MAE         MPE      MAPE      MASE
## Training set -0.001258827 0.08005903 0.05747208 -0.02527777 0.8109746 0.5291824
##                      ACF1
## Training set 0.001306447
```

```
tsdisplay(diff2.5$res)
```




```
Box.test(diff2.5$residuals, type = "Ljung-Box") #passes test
```

```
##
##  Box-Ljung test
##
## data:  diff2.5$residuals
## X-squared = 0.00092339, df = 1, p-value = 0.9758
```

```
accuracy(diff2.1)
```

```
##                         ME      RMSE        MAE          MPE     MAPE      MASE
## Training set -0.0001925617 0.1027601 0.07429065 -0.006944572 1.047855 0.6840418
##                     ACF1
## Training set -0.3534869
```

```
accuracy(diff2.2)
```

```
##                         ME       RMSE       MAE         MPE      MAPE      MASE
## Training set -0.0002219065 0.09612261 0.0693568 -0.00814911 0.9783577 0.6386126
##                     ACF1
## Training set -0.07603501
```

```r
accuracy(diff2.3)
```

```
##                         ME       RMSE        MAE         MPE     MAPE      MASE
## Training set -0.0004564484 0.08379489 0.06078584 -0.01371992 0.856558 0.5596943
##                      ACF1
## Training set -0.08650328
```

```r
accuracy(diff2.4)
```

```
##                         ME       RMSE        MAE         MPE      MAPE     MASE
## Training set -0.0006576228 0.08123724 0.05849043 -0.01657129 0.8249399 0.538559
##                     ACF1
## Training set -0.0290561
```

```r
accuracy(diff2.5)
```

```
##                          ME       RMSE        MAE         MPE      MAPE      MASE
## Training set -0.001258827 0.08005903 0.05747208 -0.02527777 0.8109746 0.5291824
##                      ACF1
## Training set 0.001306447
```

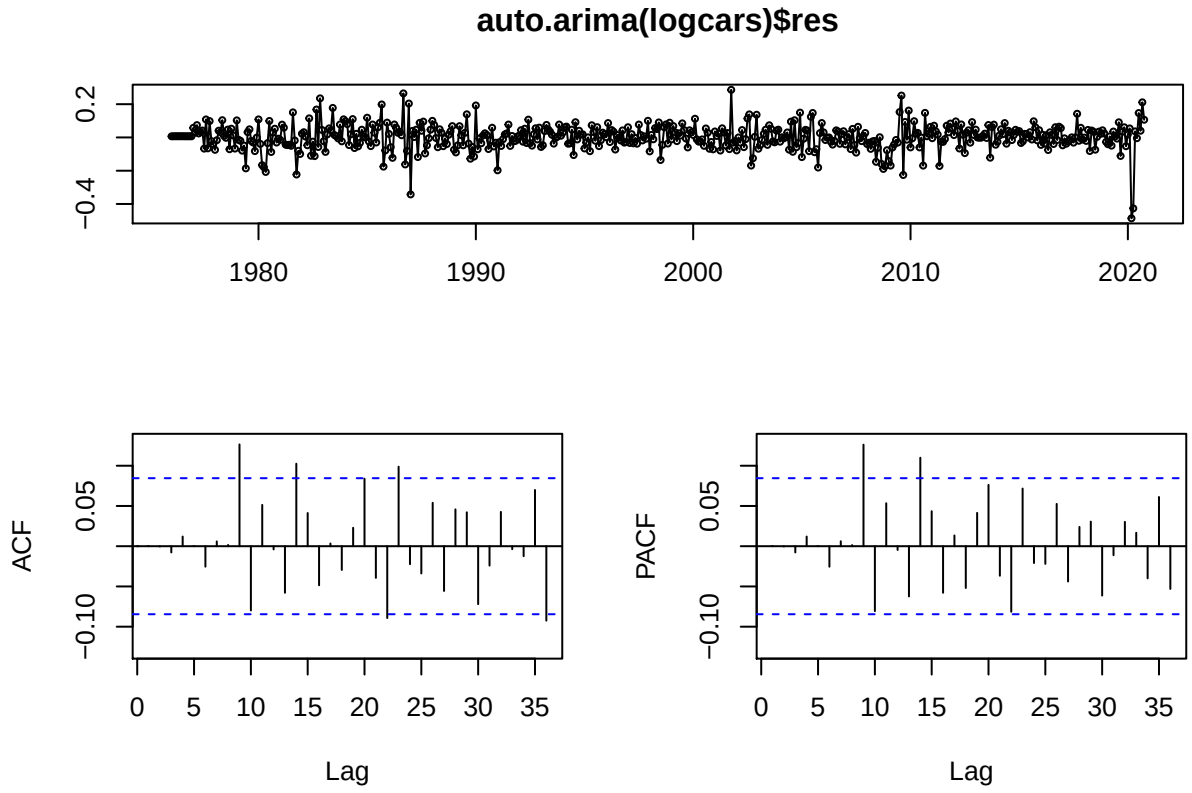
```r
#from using differencing we get that diff2.5 model is the best.

#p-values for diff2.5 - all are significant
(1-pnorm(abs(diff2.5$coef)/sqrt(diag(diff2.5$var.coef))))*2
```

```
##        ar1        ma1       sma1
## 0.01992611 0.00000000 0.00000000
```

```r
#best non-differenced/trend model from before accuracy
accuracy(ar3)
```

```
##                         ME      RMSE        MAE          MPE      MAPE      MASE
## Training set 0.0006362609 0.0799948 0.05905147 -0.002733308 0.8335184 0.5437249
##                      ACF1
## Training set 0.002527851
```

```r
#for comparison
tsdisplay(auto.arima(logcars)$res)
```

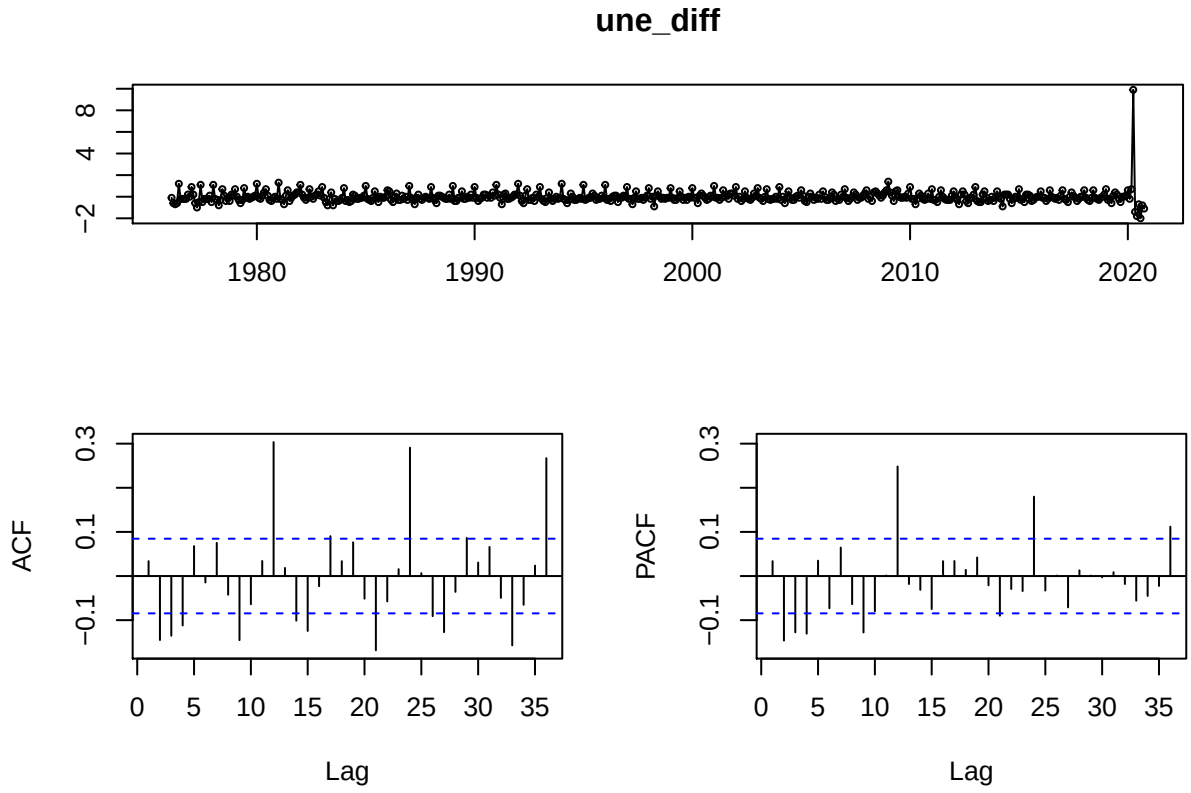


Comments on ACF and PACF for deciding which model to iterate next are in the code too. We first take the difference of the data and then look at the residuals. Looking at ACF and PACF for the residuals there is definitely seasonality that needs to be taken care of. Since the trend is rather hard to identify we also try fitting a differened arima model. We include seasonal differencing as well since we have a strong seasonal pattern. Our first iteration of ARIMA(0,1,0)(0,1,0) doesn't pass the Ljung-Box test. Then we try adding AR(1) to the differenced model since there was a spike in PACF for last model. Then since there is strong spike in ACF at 12 we try adding MA(1) to seasonality. In the next iteration we change to AR(2) because of a strong spike in PACF at 2 in last model. As a final iteration we try AR(1) and MA(1) as well becuase our results from auto.arima indicate there is an MA component.

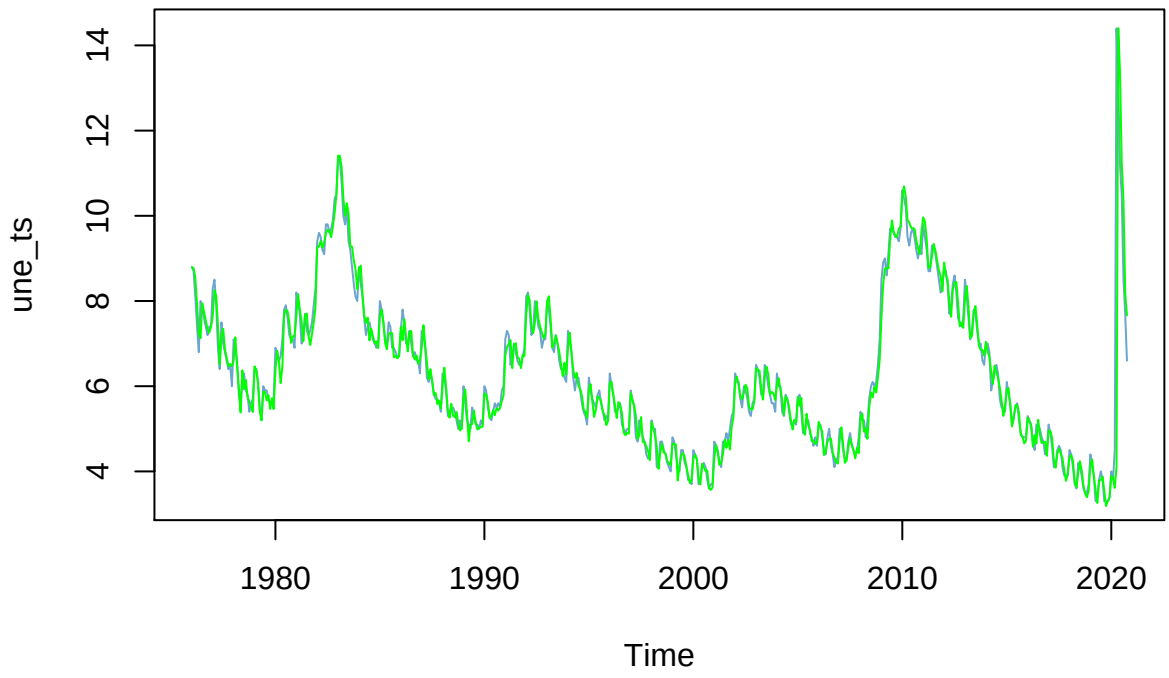
When compared to the previous model built without differencing, it performs better so we will use ARIMA(1,1,1)x(0,1,1)[12]. The auto.arima ARIMA(1,0,2)(0,1,2)[12] so ours is a lower amount of df. The autoarima does not use first differencing for the trend and seasonal but we thought this would make more sense for our data since it is hard to identify a linear form of a trend in the decomposition earlier and we have strong consistent seasonality. The p-values for our selected model are significant and it's residuals pass the Ljung Box test. The residuals in the model from the auto arima also tend to spike when compared, aka the residuals are similar. Looking at the accuracy measures we see that RMSE and MAPE are the lowest for our selected model when comparing the best model with the trend and all the other differenced models.

Below we try a differenced model for unemployment.

```
une_diff = diff(une_ts)
tsdisplay(une_diff) #there is strong seasonality in the PACF that could be
```



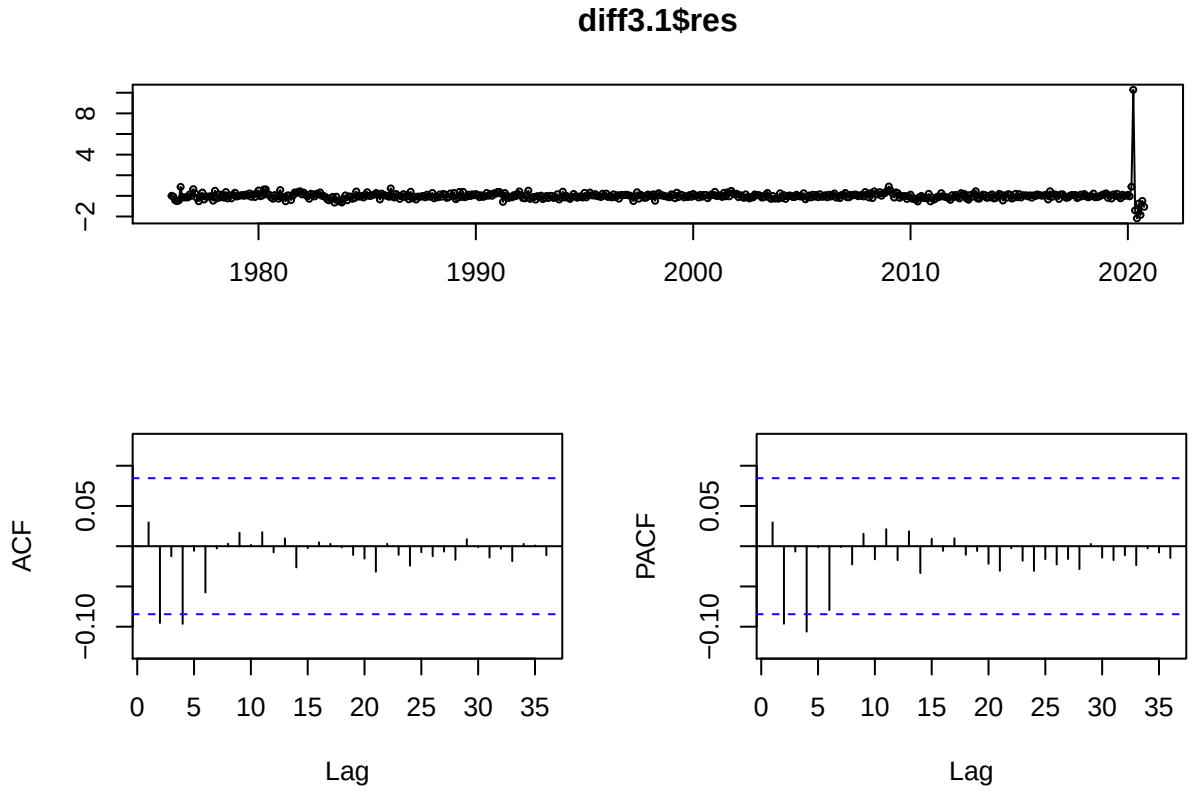

```
#S-AR(2) since there are 2 spikes in the PACF which agrees withe auto.arima
#model, so we will keep seasonality at S-AR(2).

diff3.1=arima(une_ts,order=c(0,1,0),seasonal=list(order=c(2,0,0)))
plot(une_ts, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(diff3.1),col="green")
```




```
summary(diff3.1)
```

```
##
## Call:
## arima(x = une_ts, order = c(0, 1, 0), seasonal = list(order = c(2, 0, 0)))
##
## Coefficients:
##         sar1    sar2
##       0.4055  0.3558
## s.e.  0.0880  0.0900
##
## sigma^2 estimated as 0.2692:  log likelihood = -414.23,  aic = 834.46
##
## Training set error measures:
##                       ME      RMSE       MAE         MPE     MAPE      MASE
## Training set 0.004146867 0.5183283 0.202014 0.007171978 3.018738 0.6213146
##                     ACF1
## Training set 0.02956826
```

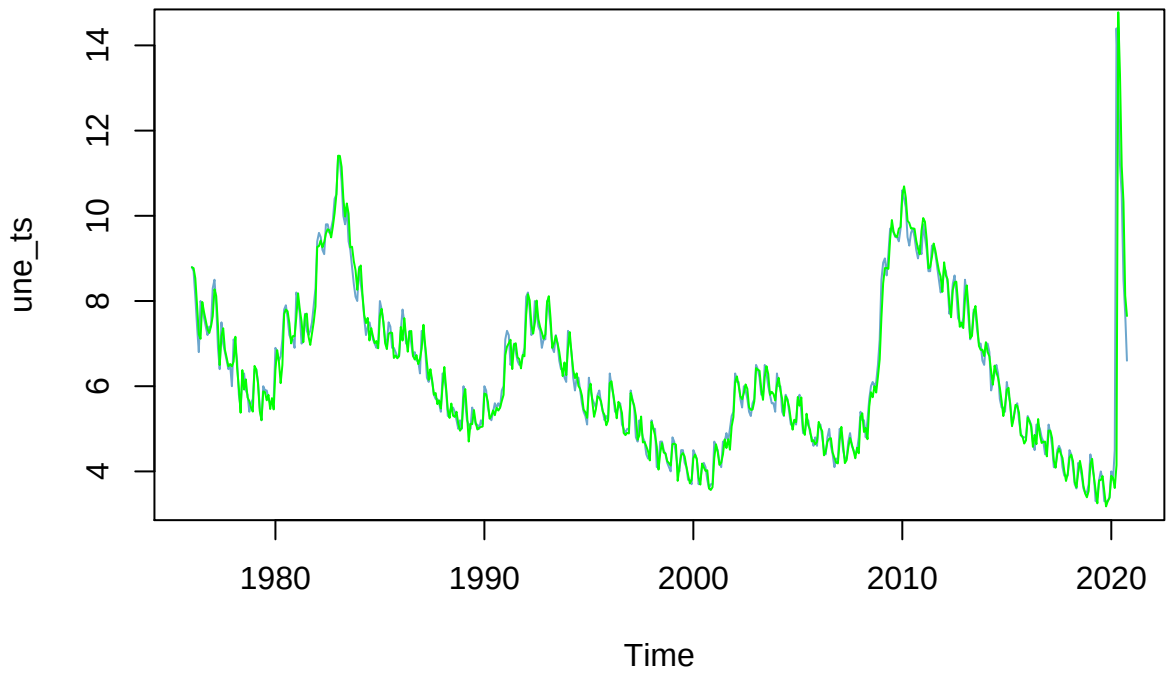
```
tsdisplay(diff3.1$res)
```




```
Box.test(diff3.1$residuals, type = "Ljung-Box") #passes test
```

```
##
##  Box-Ljung test
##
## data:  diff3.1$residuals
## X-squared = 0.47299, df = 1, p-value = 0.4916
```

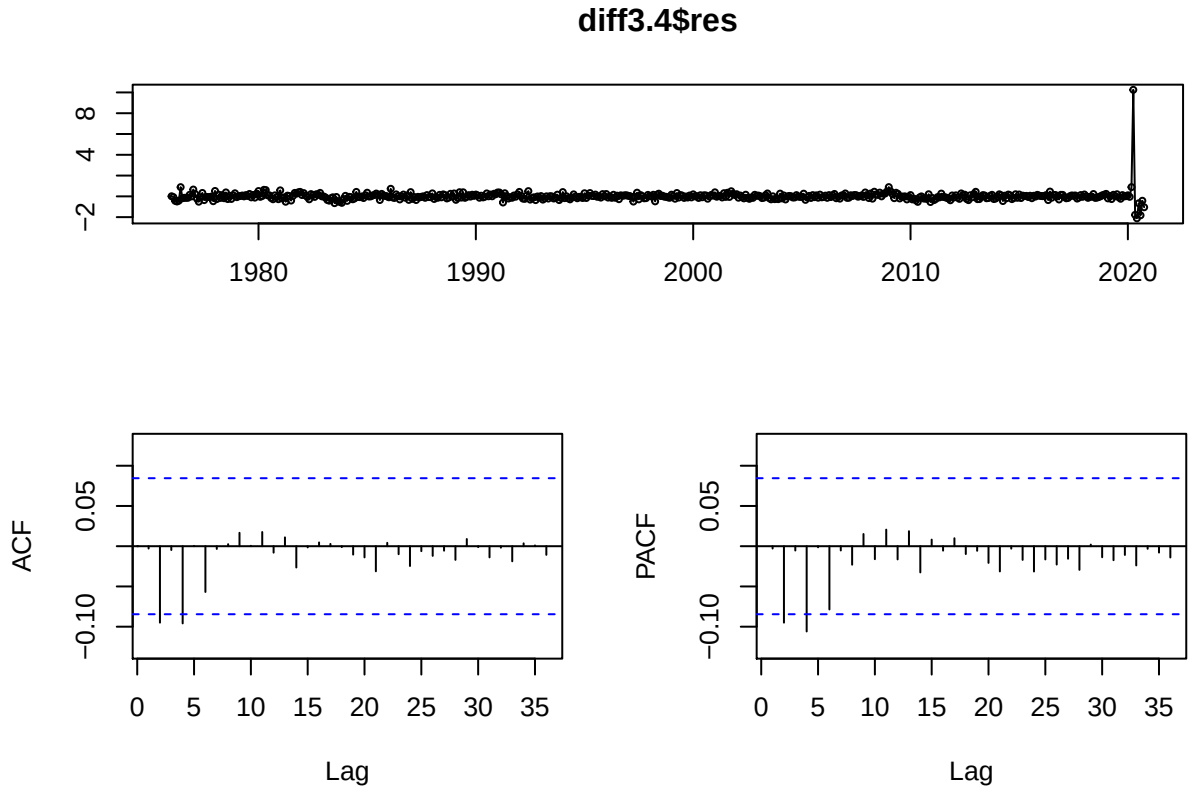
```
#we try to do AR(1) also because of the PACF of the original data but are met
#with an error with non-finite values below

#diff3.5=arima(une_ts,order=c(1,1,0),seasonal=list(order=c(2,0,0)))
#plot(une_ts, xlab="Time", lwd=1, col='skyblue3')
#lines(fitted(diff3.5),col="green")
#summary(diff3.5)
#tsdisplay(diff3.5$res)
#Box.test(diff3.5$residuals, type = "Ljung-Box")

#we add MA(1) instead to see if it will calculate.
diff3.4=arima(une_ts,order=c(0,1,1),seasonal=list(order=c(2,0,0)))
plot(une_ts, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(diff3.4),col="green")
```




```
summary(diff3.4)
```

```
##
## Call:
## arima(x = une_ts, order = c(0, 1, 1), seasonal = list(order = c(2, 0, 0)))
##
## Coefficients:
##          ma1    sar1    sar2
##       0.0366  0.3999  0.3621
## s.e.  0.0484  0.0882  0.0903
##
## sigma^2 estimated as 0.2688:  log likelihood = -413.95,  aic = 835.89
##
## Training set error measures:
##                       ME      RMSE       MAE         MPE     MAPE      MASE
## Training set 0.003964558 0.5180213 0.2020776 0.007814707 3.022188 0.6215103
##                      ACF1
## Training set -0.002913738
```

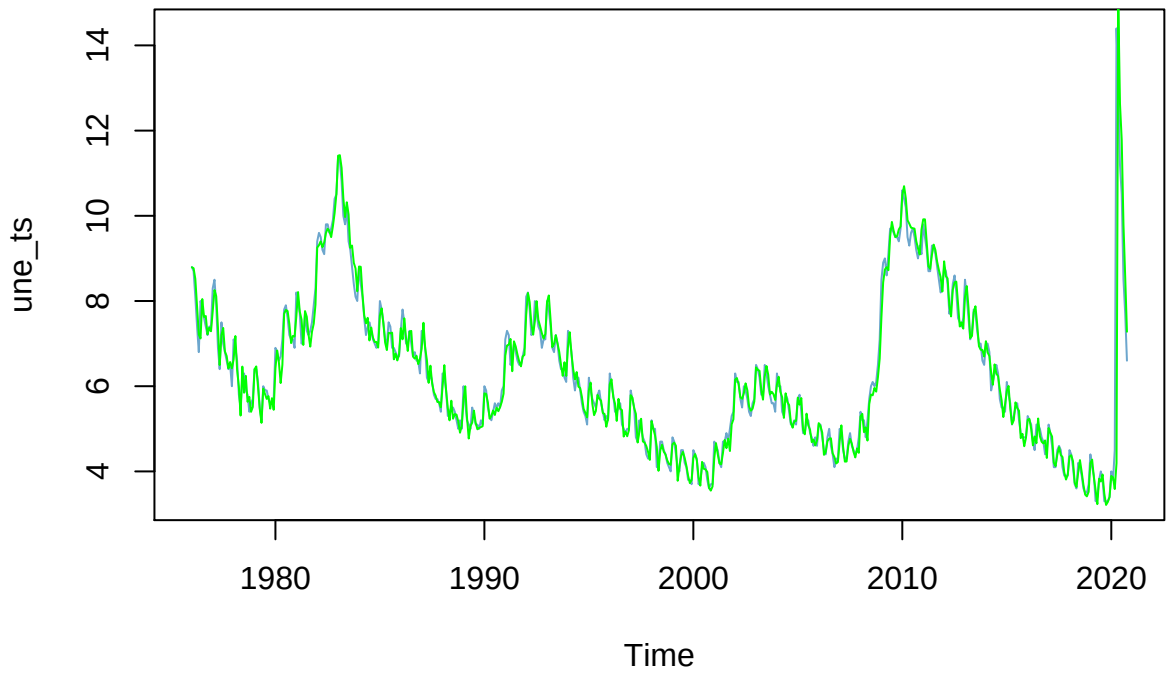
```
tsdisplay(diff3.4$res)
```




```
Box.test(diff3.4$residuals, type = "Ljung-Box") #passes test
```

```
##
##  Box-Ljung test
##
## data:  diff3.4$residuals
## X-squared = 0.0045931, df = 1, p-value = 0.946
```

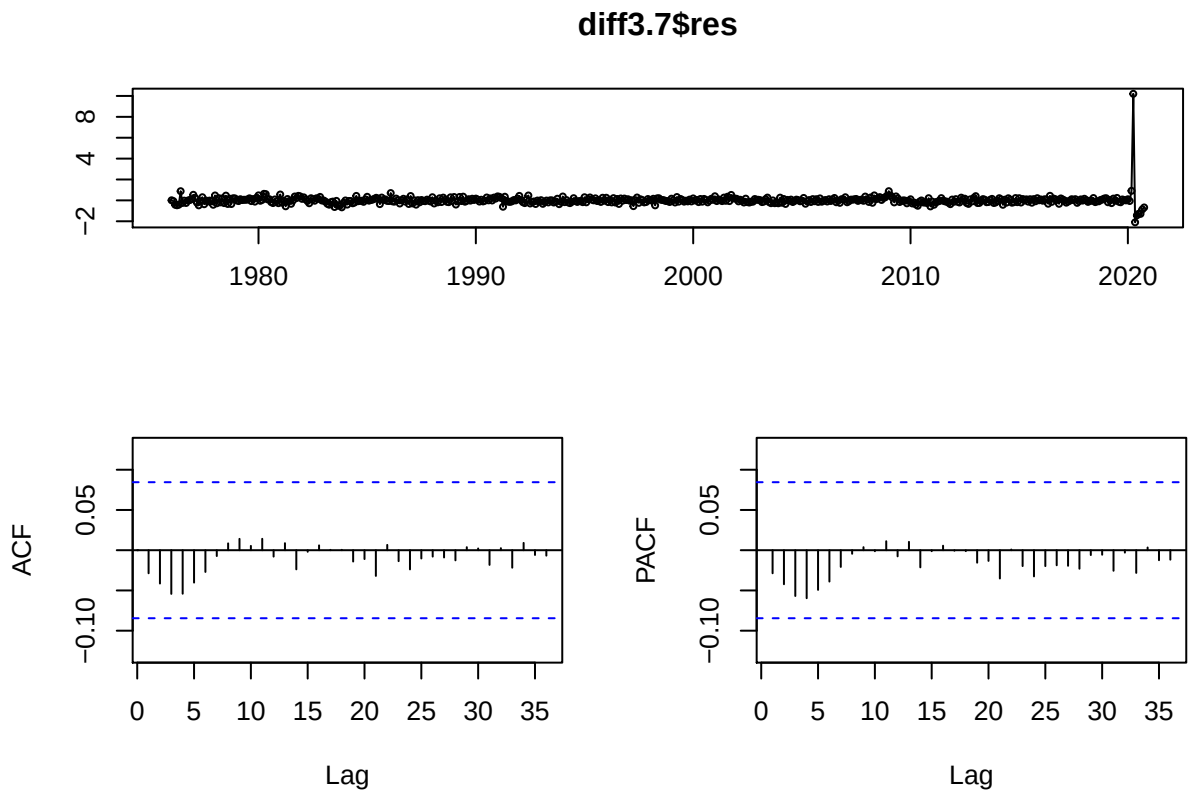
```
#It works but we see some major spikes in the PACF of residuals.


#Since we saw some spikes still we add AR(1) since that was what we thought
#it would be originally
diff3.7=arima(une_ts,order=c(1,1,1),seasonal=list(order=c(2,0,0)))
plot(une_ts, xlab="Time", lwd=1, col='skyblue3')
lines(fitted(diff3.7),col="green")
```




```
summary(diff3.7)
```

```
##
## Call:
## arima(x = une_ts, order = c(1, 1, 1), seasonal = list(order = c(2, 0, 0)))
##
## Coefficients:
##           ar1     ma1    sar1    sar2
##       -0.8533  0.9289  0.4070  0.3715
## s.e.   0.0597  0.0450  0.0889  0.0894
##
## sigma^2 estimated as 0.2638:  log likelihood = -409.14,  aic = 828.27
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE    MAPE      MASE
## Training set 0.004412565 0.5131745 0.2024289 0.02101432 3.03623 0.6225905
##                    ACF1
## Training set -0.028649
```

```
tsdisplay(diff3.7$res)
```




```
Box.test(diff3.7$residuals, type = "Ljung-Box") #passes test
```

```
##
##  Box-Ljung test
##
## data:  diff3.7$residuals
## X-squared = 0.44404, df = 1, p-value = 0.5052
```

```
accuracy(diff3.1)
```

```
##                       ME      RMSE       MAE          MPE     MAPE      MASE
## Training set 0.004146867 0.5183283 0.202014 0.007171978 3.018738 0.6213146
##                     ACF1
## Training set 0.02956826
```

```
accuracy(diff3.4)
```

```
##                       ME      RMSE       MAE          MPE     MAPE      MASE
## Training set 0.003964558 0.5180213 0.2020776 0.007814707 3.022188 0.6215103
##                      ACF1
## Training set -0.002913738
```

```
accuracy(diff3.7)
```

```
##                       ME      RMSE       MAE        MPE    MAPE      MASE
## Training set 0.004412565 0.5131745 0.2024289 0.02101432 3.03623 0.6225905
##                    ACF1
## Training set -0.028649
```

```
AIC(diff3.1, diff3.4, diff3.7) #diff3.7 is lowest
```

```
##         df      AIC
## diff3.1  3 834.4629
## diff3.4  4 835.8935
## diff3.7  5 828.2748
```

```
accuracy(uar3)
```

```
##                       ME      RMSE       MAE         MPE    MAPE      MASE
## Training set -0.00182827 0.5068746 0.1970543 -0.4534406 2.98626 0.6060606
##                      ACF1
## Training set -0.001691921
```

We start of with S-AR(2) and an AR(0,1,0) since this is what most likely could be a good model when looking at the ACF and PACF of residuals from the differenced data, which is also in line with what the auro arima suggests.

Comments on ACF and PACF for deciding which model to iterate next are in the code also. For our next iteration we try to do AR(1) also because of the PACF of the original data but are met with an error with non-finite values. Then we try to add an MA(1) to see if it will calculate with guidance from the auto.arima so that we get finite values. Since we sitll see spikes in the residuals of that iteration we add AR(1) since that was we thought a plausiable model would be for the original data in the beginning.

We eventually end up with ARIMA(1,1,1)(2,0,0)[12]. This lines up with what we get from the auto arima. Looking at the accuracy measures we see that ARIMA(1,1,1)(2,0,0)[12] our chosen model has the lowst RMSE, but not the lowest MAPE, although all the values are very close together. Nevertheless we will stick with the chosen model since the AIC is lowest and it is suggested by the auto.arima as well.

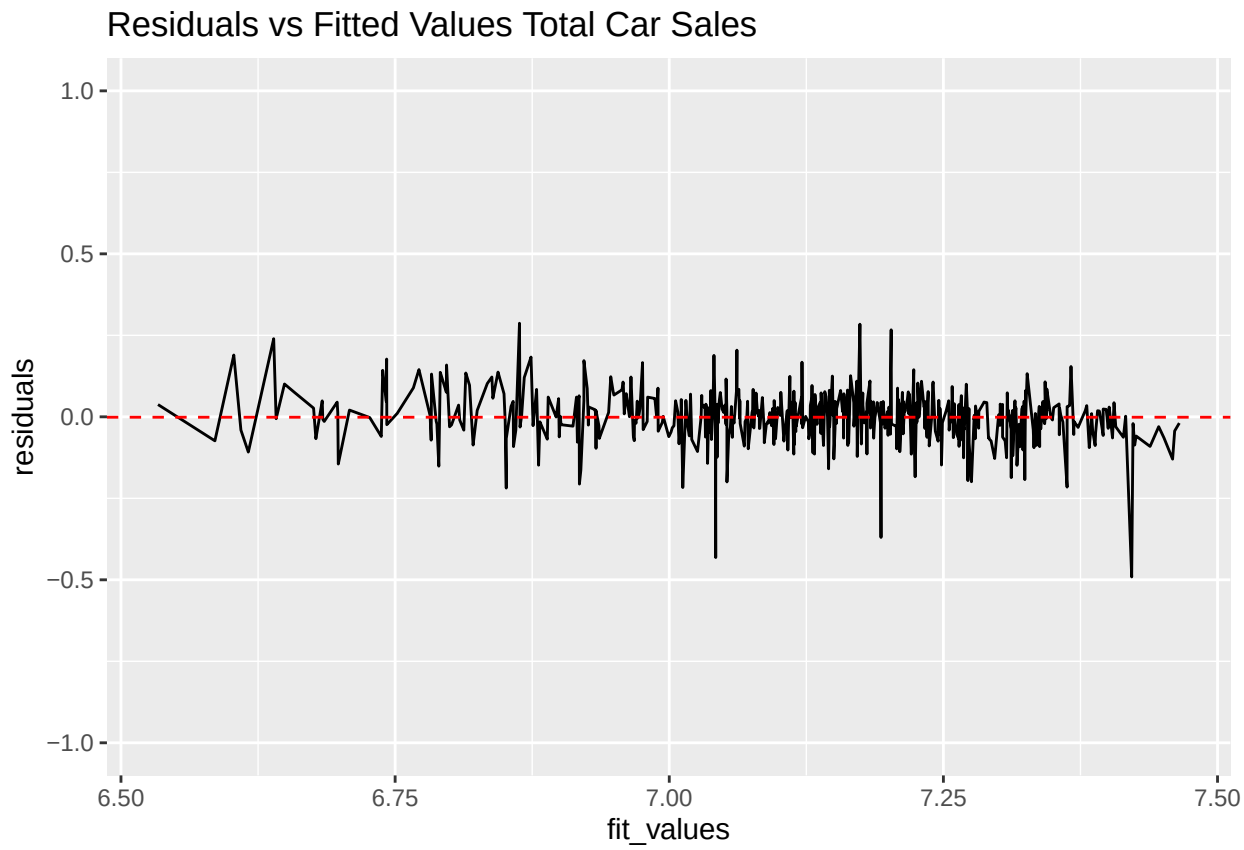
When we compare accuracy with the trend model with no differencing that we selected before, which is UAR3, we see that RMSE and MAPE are both a tiny bit lower. However, since we were dubious about the trend since the data doesn't seem to show a very obvious trend, we decide on the ARIMA(1,1,1)(2,0,0)[12] as our final model for unemployment.

0.4 4. Plot the respective residuals vs. fitted values and discuss your observations.

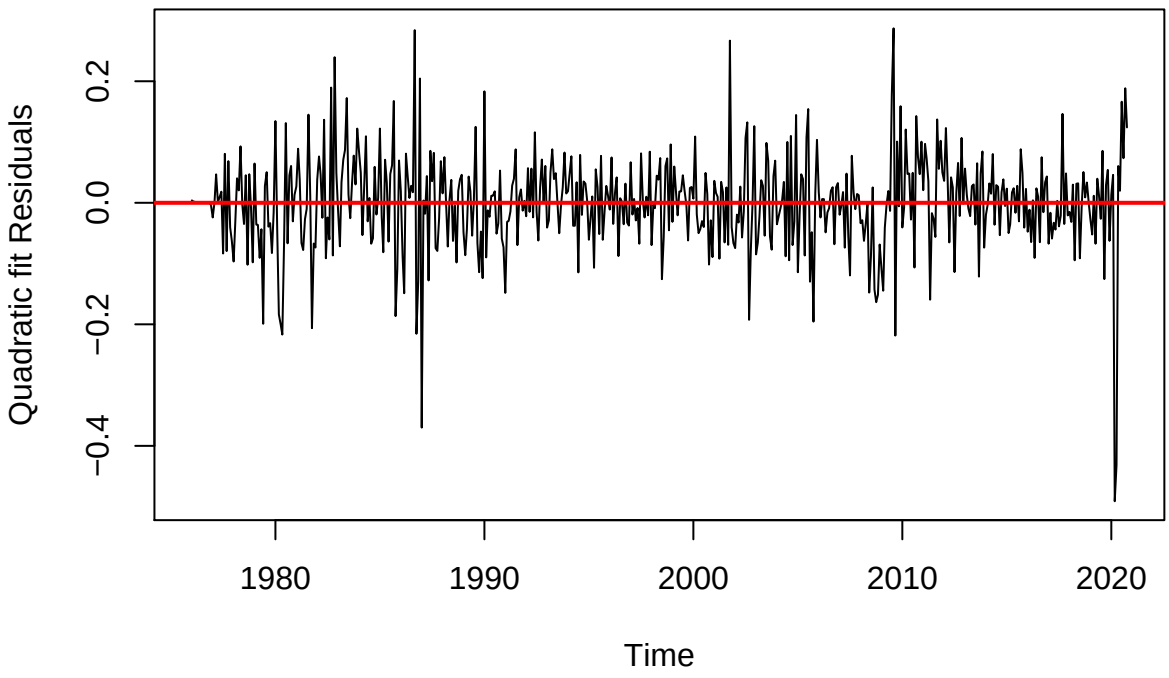
From the residual vs. fit values graph for Total car sales below, we se an even spread across all the values. The mean is near 0, which there is no significant correlation in the residuals series. It also shows the variation of the residuals stay pretty much constant across historical data, except a few large noticable spikes at fit_values of 7.03, 7.21, and 7.37. These spikes are not large enough to consider as outlier or enought to impact the over all weight of our residuals, which we will treat as constant.

```
library(ggplot2)
total_car  <- data.frame(residuals=diff2.5$residuals, fit_values=fitted(diff2.5))
ggplot(total_car, aes(x=fit_values, y=residuals))+
  geom_line()+
  ggtitle("Residuals vs Fitted Values Total Car Sales")+
  geom_hline(yintercept = mean(diff2.5$residuals), linetype="dashed", color="red")+
  ylim(-1,1)
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```




```
#plot(logcars)
#lines(t,diff2.5$fit,col="red3",lwd=2)
plot(t, diff2.5$res, ylab="Quadratic fit Residuals",xlab="Time", type = 'l')
abline(a=0, b=0, lwd=2, col="red")
```

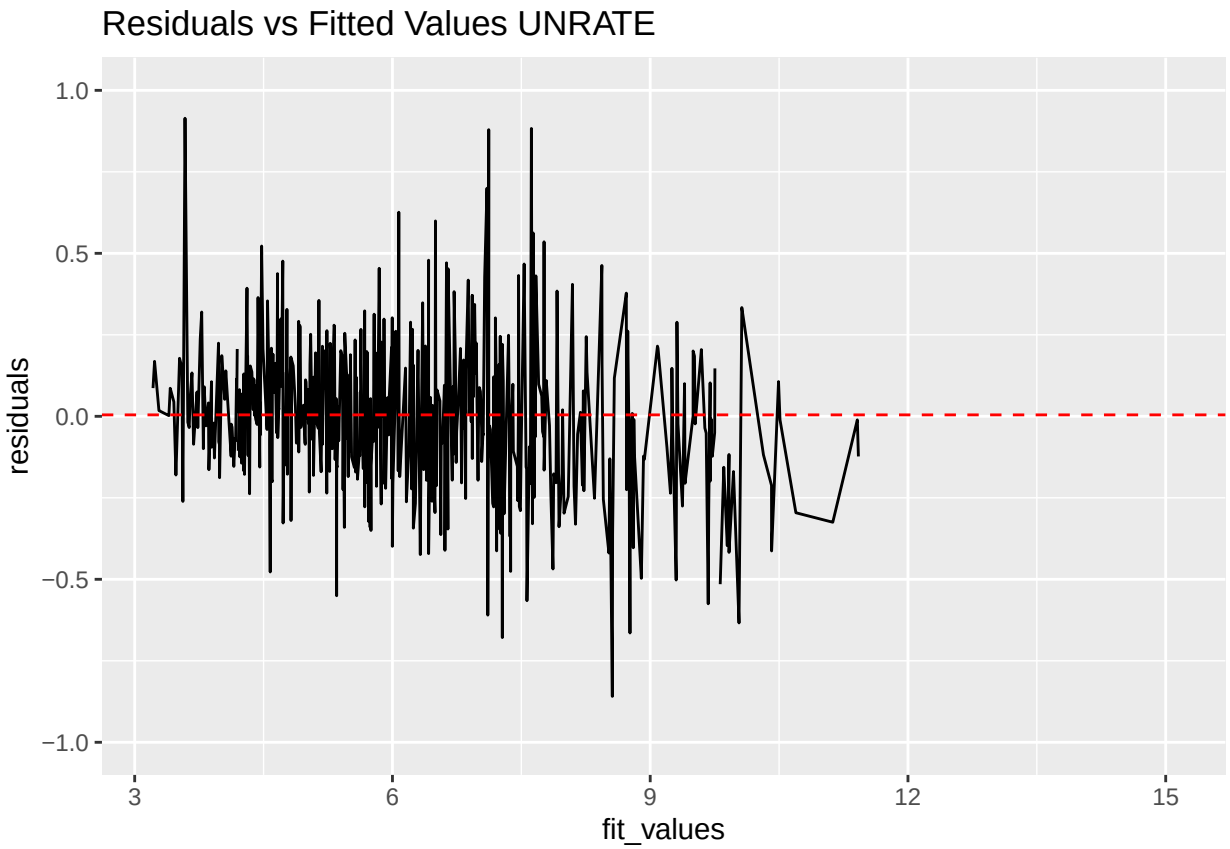


From the residual vs. fit values graph for Unemployment rate below, we see pretty even spread across all the values. However, near the lower fitted value there seems to be a bit less variation in error. Overall, if we look at the band between [-0.25 and 0.25], most seems to fall in that window. This makes us believe that variance is pretty constant here. The mean is 0, and from the Ljung Box test previously conducted we know there is no serial correlation in the residuals. We see a large spike at the end over time which is from the COVID-19 pandemic.

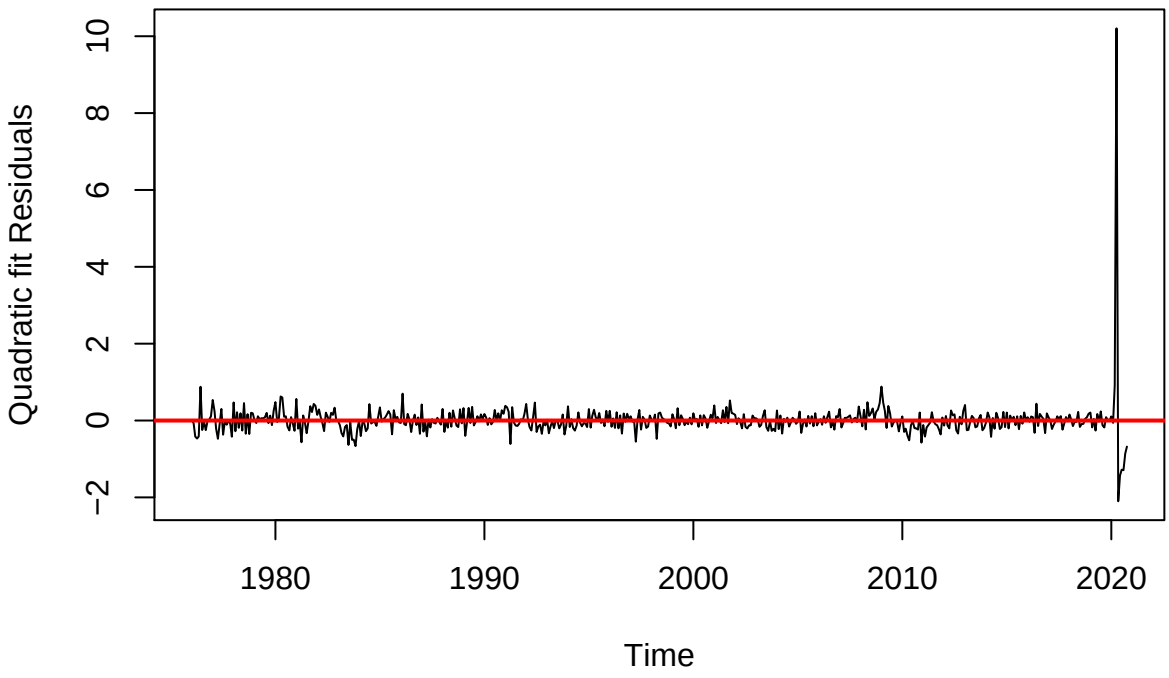
```
library(ggplot2)
total_unemploy  <- data.frame(residuals=diff3.7$residuals, fit_values=fitted(diff3.7))
ggplot(total_unemploy, aes(x=fit_values, y=residuals))+
  geom_line()+
  ggtitle("Residuals vs Fitted Values UNRATE ")+
  geom_hline(yintercept = mean(diff3.7$residuals), linetype="dashed", color="red")+
  ylim(-1,1)
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

```
## Warning: Removed 3 row(s) containing missing values (geom_path).
```




```
#plot(une_ts)
#lines(t, diff3.7$fit, col="red", lwd=2)
plot(t,diff3.7$res, ylab="Quadratic fit Residuals",xlab="Time",type = 'l')
abline(a=0, b=0, lwd=2, col="red")
```

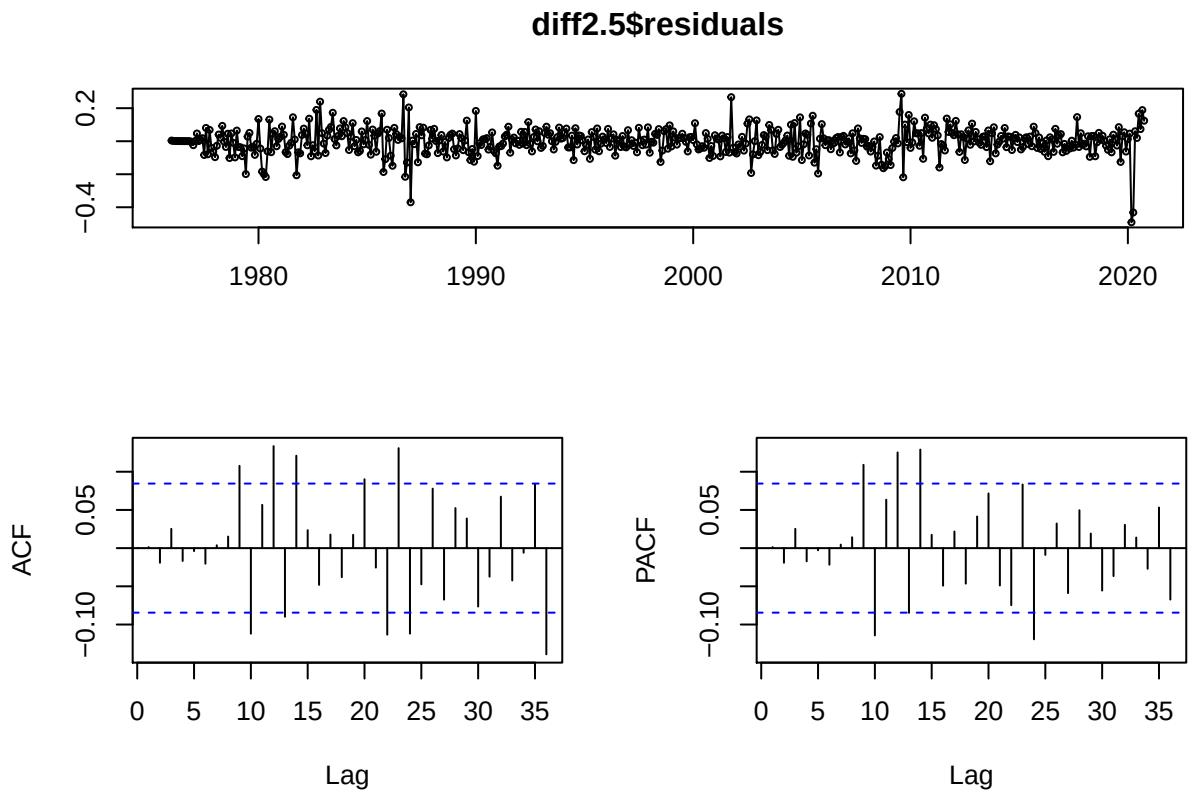


0.5 5. Plot the ACF and PACF of the respective residuals and interpret the plots.

Interpret model for Total Car Sales, which is diff2.5

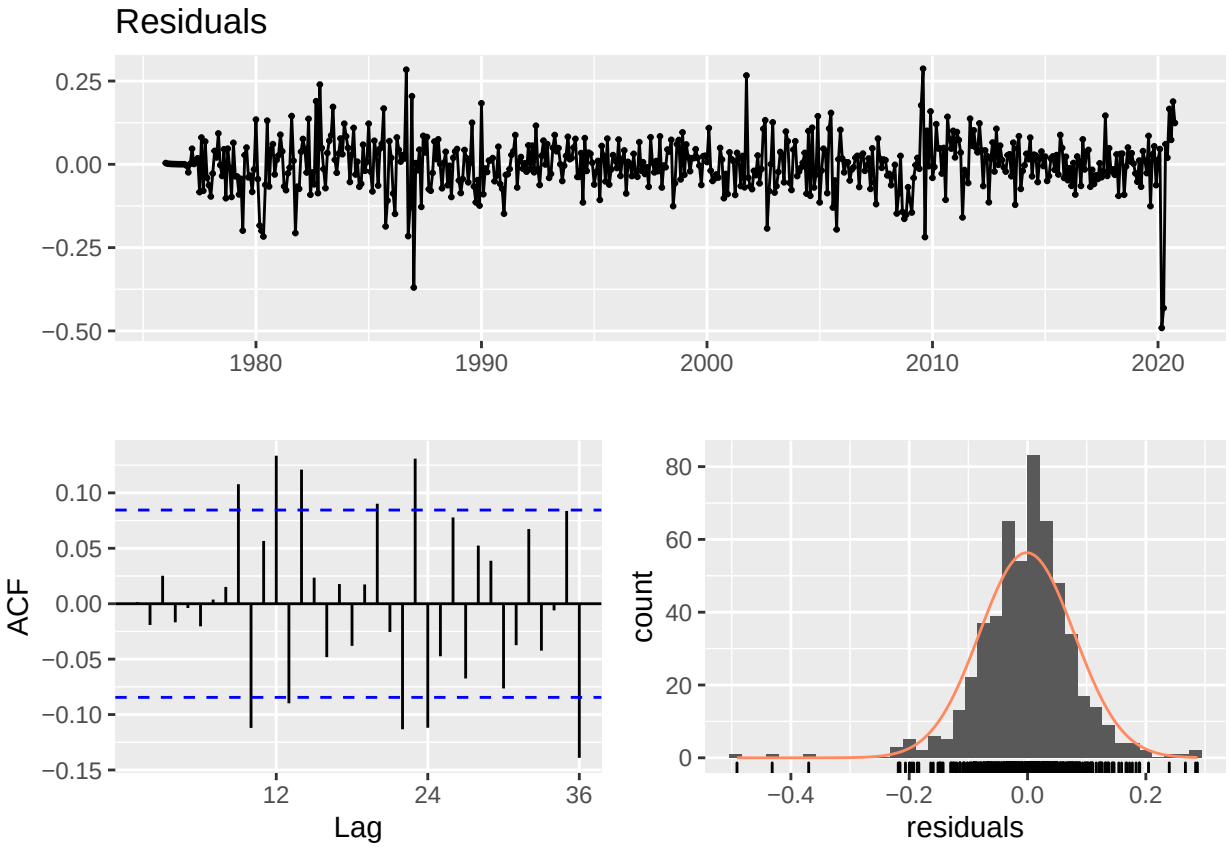
As seen below, the Ljung-Box test tells us that the residuals are not serially correlated. We also see the residuals are normally distributed pretty much. When we look at the PACF and ACF we see that most of the lags fall under the lines (are near 0) so they are not significant. When compared to the residuals of model chosen my auto.arima, we find that the auto.arima also has a similar PACF and ACF (numbers of lags that are technically significant). Since they are close to 0 here (same as auto.arima), we accept this model as the best model still.

```
tsdisplay(diff2.5$residuals)
```




```
checkresiduals(diff2.5$residuals)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



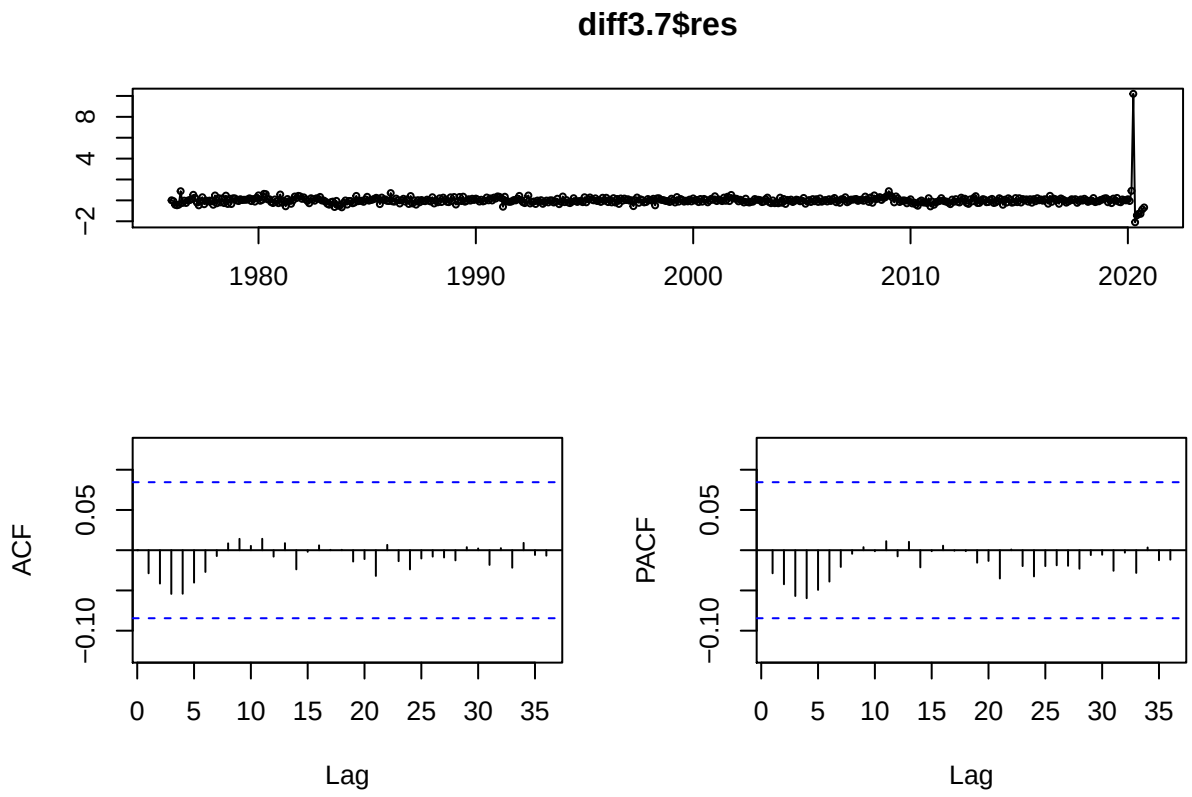

```
Box.test(diff2.5$residuals, type = 'Ljung-Box')
```

```
##
##  Box-Ljung test
##
## data:  diff2.5$residuals
## X-squared = 0.00092339, df = 1, p-value = 0.9758
```

Interpret model for Unemployment which is diff3.7

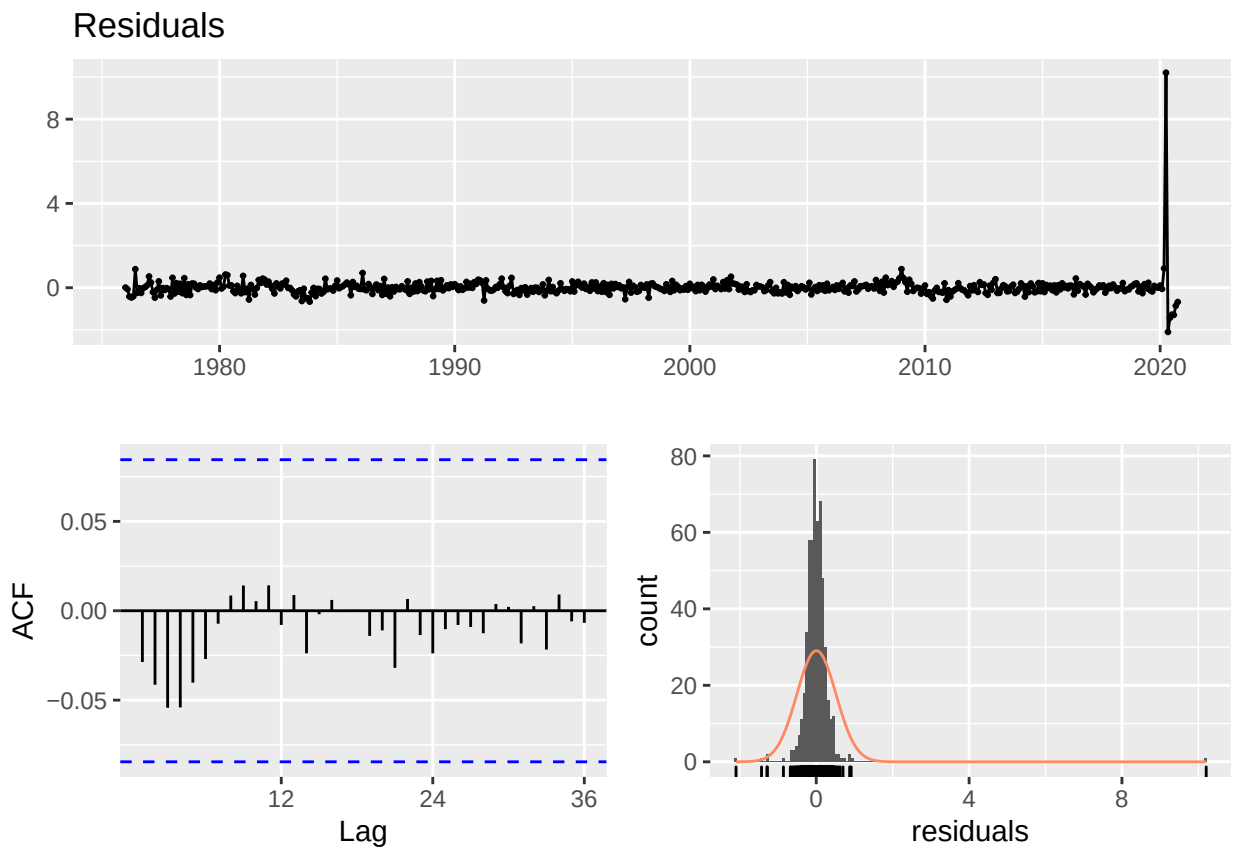
Our best model here is the same at the auto.arima model. We see that it passes the Ljung-Box test for residuals. It seems like our residuals are a bit narrower than the normal distribution, but that is alright since they are narrower at the mean of the normal distribution. All the ACF and PACF lags are insiginificant.

```
tsdisplay(diff3.7$res)
```




```
checkresiduals(diff3.7$res)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
Box.test(diff3.7$residuals, type = 'Ljung-Box')
```

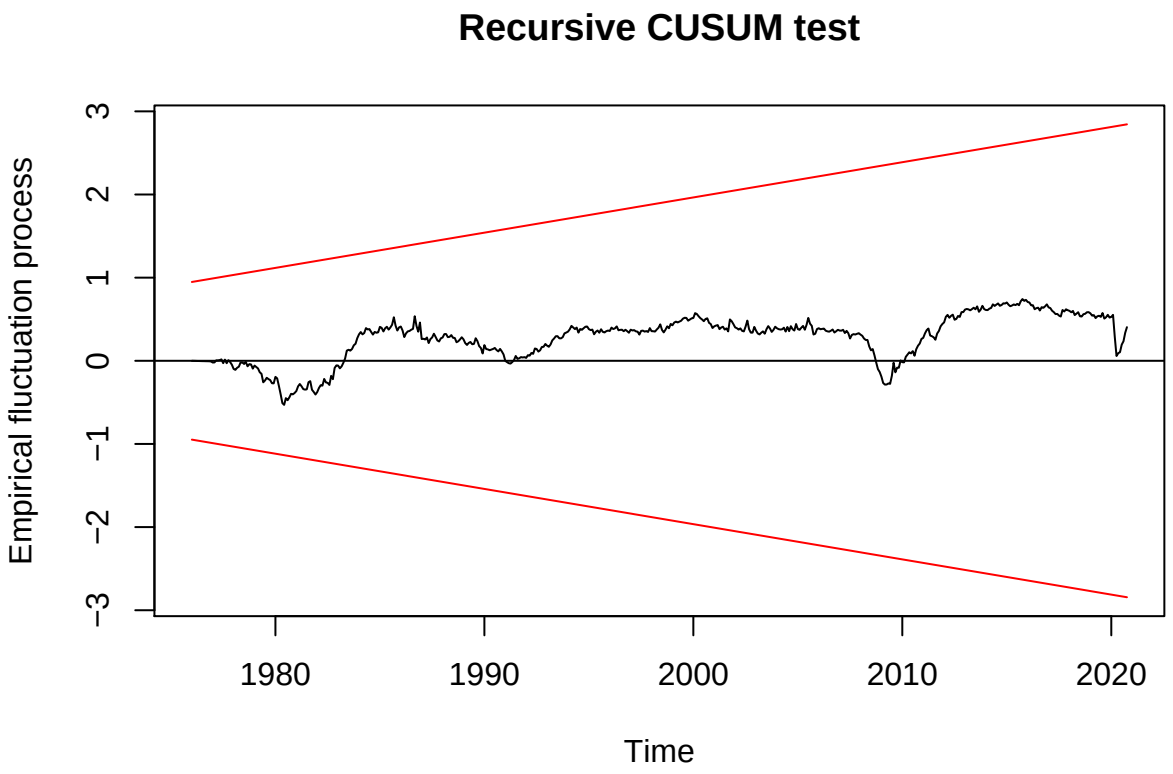
```
##
##  Box-Ljung test
##
## data:  diff3.7$residuals
## X-squared = 0.44404, df = 1, p-value = 0.5052
```

0.6 6. Plot the respective CUSUM and interpret the plot.

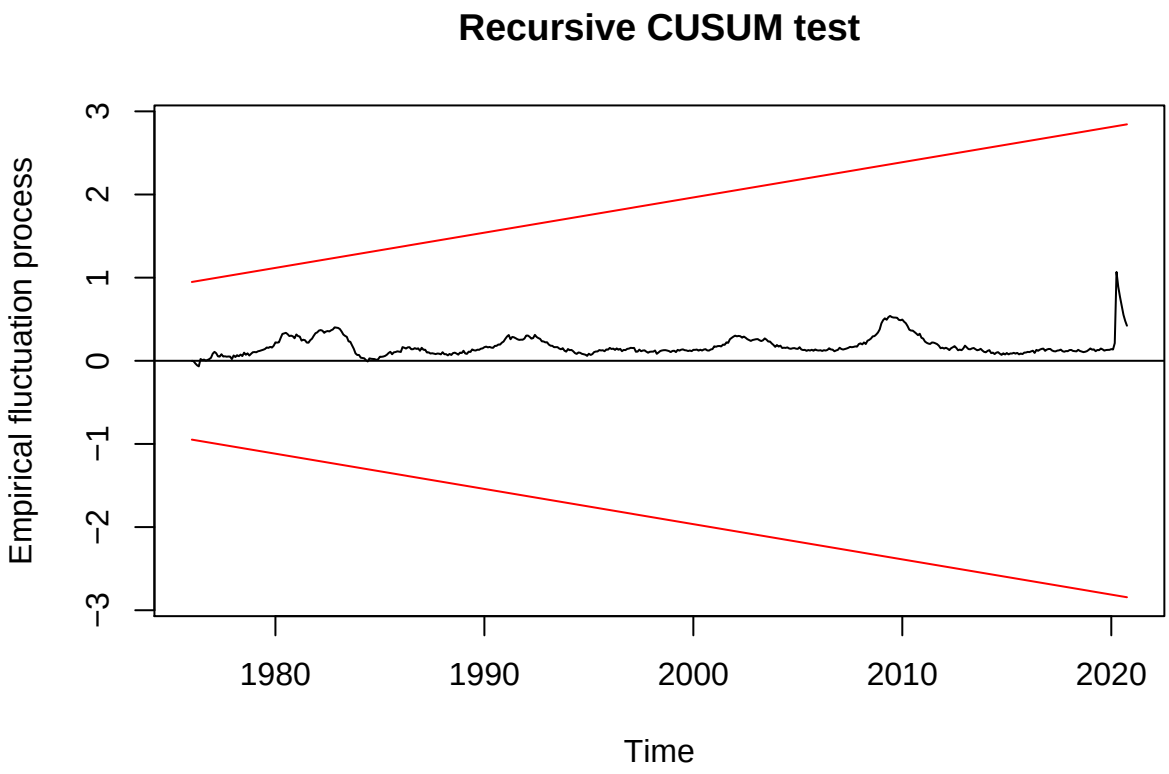
We are going to plot the CUSUM for our choosen models: for Total car sales is ARIMA(1,1,1)x(0,1,1) [12] which is diff2.5 model. While Unemployment rate is ARIMA(1,1,1)x(2,0,0)[12] which is diff3.7 model.

From the plotted CUSUM, we see that the cumulative sum of residuals stays mostly constant.There are no upward or downward drifts of the cumulative sum that cross a boundary line. This does not indicate an out of control process.

```
#total car sales
plot(efp(diff2.5$res~1, type = "Rec-CUSUM"))
```




```
#unemployment rate
plot(efp(diff3.7$res~1, type = "Rec-CUSUM"))
```

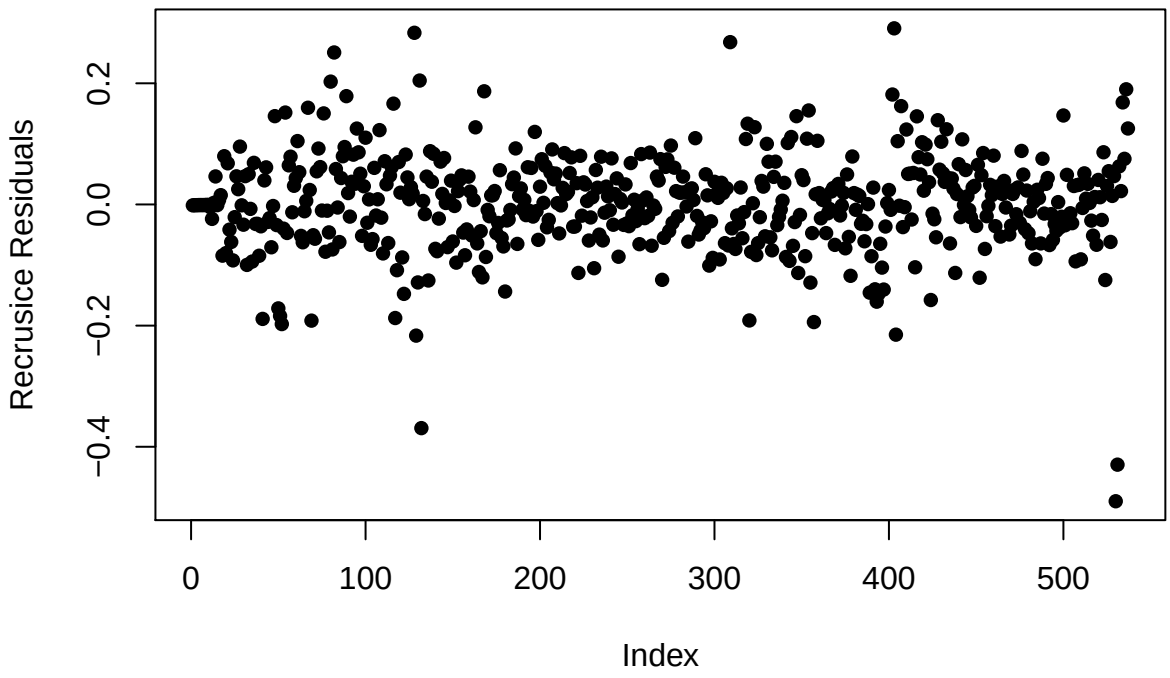


0.7 7. Plot the respective Recursive Residuals and interpret the plot.

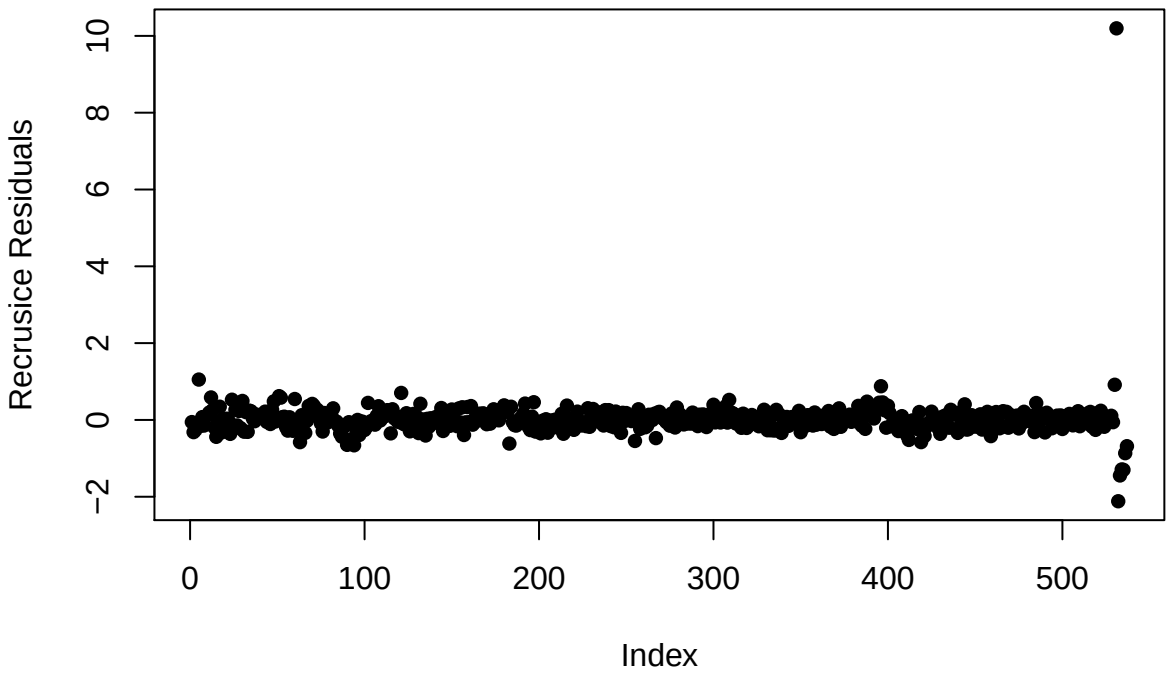
We are going to plot the the Recursive Residuals for our choosen models: for Total car sales is ARIMA(1,1,1)x(0,1,1) [12] which is diff2.5 model. While Unemployment rate is ARIMA(1,1,1)x(2,0,0)[12] which is diff3.7 model.

The plot of recursive residuals shows that our models are accurate in predicting true values as we do not see much dispersion in residuals. Our model for Car Sales Data is not as reliable for accurate predictions as the one for the Unemployment Data due to the dispersion of the residuals being tighter for the unemployment data. Further, we see some stray residual outliers in higher index values but these are not seen much in the graph overall.

```
##Question 7, recursive plot residuals for Toral Car sales
car_sales_diff2.5 <- recresid(diff2.5$residuals ~ 1)
plot(car_sales_diff2.5, pch=16, ylab="Recrusice Residuals")
```




```
##Question 7, recursive plot residuals for Unemployment
unemployment_diff3.7 <- recresid(diff3.7$residuals ~ 1)
plot(unemployment_diff3.7, pch=16, ylab="Recrusice Residuals")
```



0.8 8. For your model, discuss the associated diagnostic statistics.

```
accuracy(diff2.5)
```

```
##                       ME       RMSE        MAE         MPE      MAPE      MASE
## Training set -0.001258827 0.08005903 0.05747208 -0.02527777 0.8109746 0.5291824
##                     ACF1
## Training set 0.001306447
```

```
accuracy(diff3.7)
```

```
##                        ME      RMSE       MAE        MPE    MAPE      MASE
## Training set 0.004412565 0.5131745 0.2024289 0.02101432 3.03623 0.6225905
##                    ACF1
## Training set -0.028649
```

Previously in question 3 we already compared AIC and BIC when choosing model as well as the measures in the accuracy function. Refer to question 3 for model comparison.

When looking at the measures for our optimal model, we see that RMSE is low given the scale of the variables (both have numbers around 10 for the scale). The MAPE for our cars model is very low, the forecast is only off by 0.8%, which means we could either have a very good model, or there was some overfitting. The MAPE

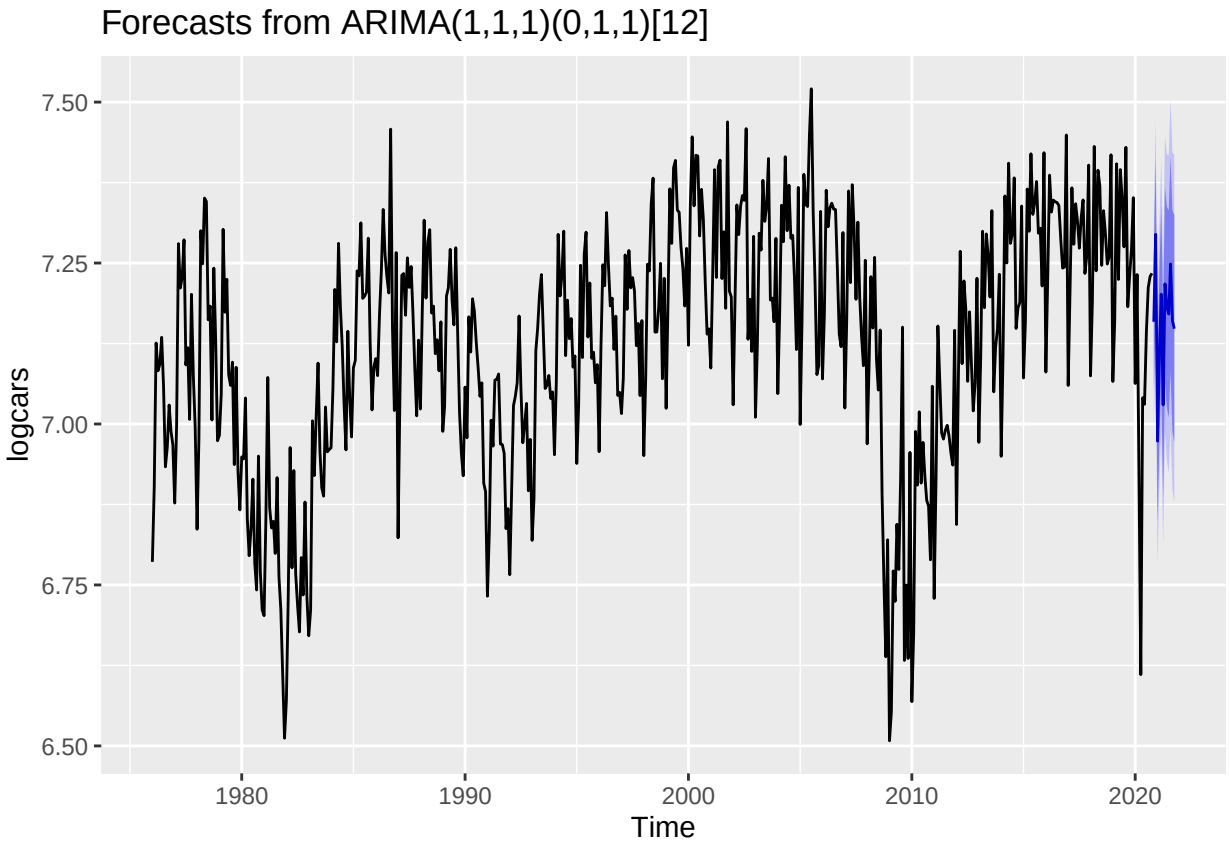
for unemployment is also low, the forecast is off by 5%. The other diagnostics in the accuracy function are also low for both models, and we therefore conclude that the models we chose do a good job of predicting.

Other statistics for the residuals in particular are in questions 4 and 5 above.

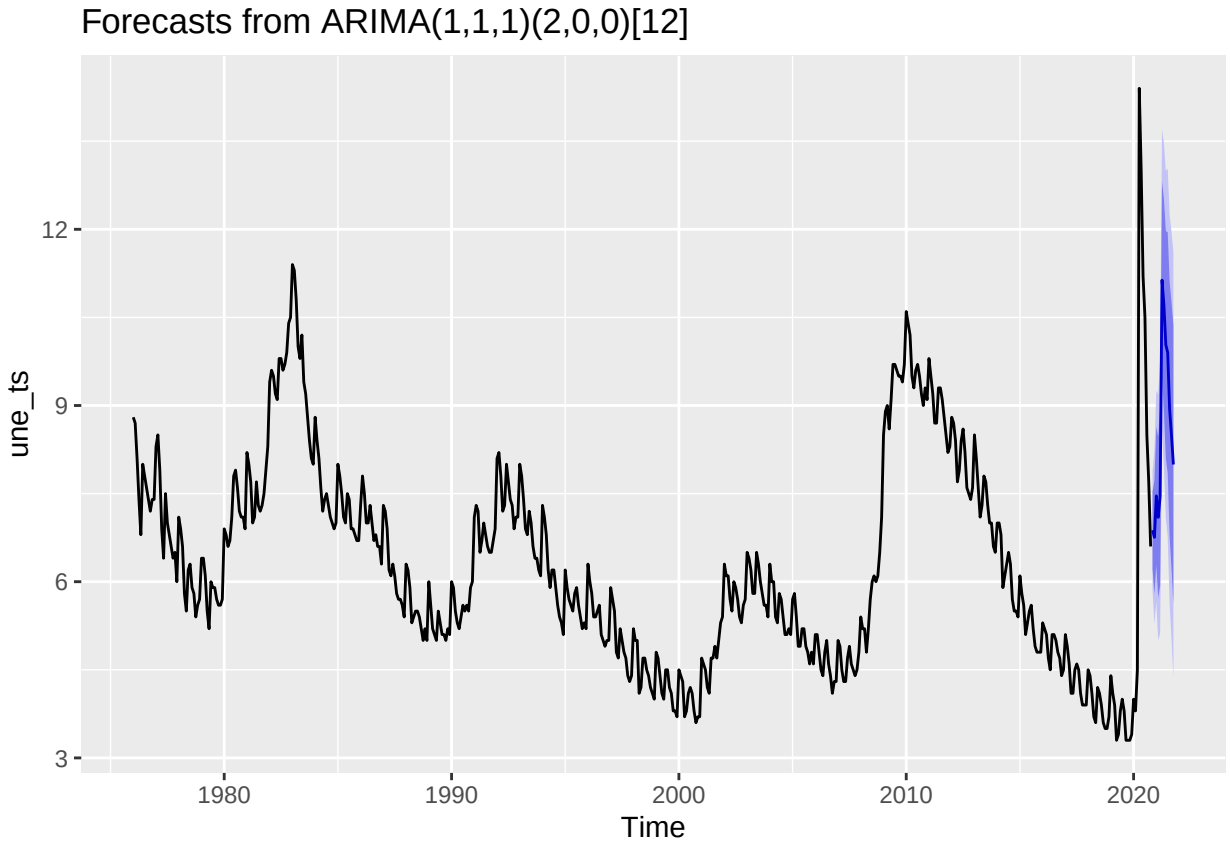
0.9 9. Use your model to forecast 12-steps ahead. Your forecast should include the respective error bands.

The data forecasted looks very similar to the original data.

```
#Total Car Sales
autoplot(forecast(diff2.5,h=12))
```




```
#Unemployment rate
autoplot(forecast(diff3.7,h=12))
```



0.10 10. Fit an appropriate VAR model using your two variables. Make sure to show the relevant plots and discuss your results from the.

```
une = read.csv("UNRATENSA.csv")
cars = read.csv("TOTALNSA.csv")
cars_ts = ts(cars$TOTALNSA, start = c(1976,1), frequency = 12)
une_ts = ts(une$UNRATENSA, start = c(1976,1), frequency = 12)
```

```
# prep the data for fitting a VAR model
y = cbind(cars_ts,une_ts)
#head(y)

y_tot <- data.frame(y)
#head(y_tot)
```

```
VARselect(y_tot,25)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     18     13     13     18
##
```

```
## $criteria
##                  1           2           3           4           5           6
## AIC(n)    8.899565    8.841068    8.775880    8.745774    8.729588    8.732502
## HQ(n)     8.919004    8.873467    8.821238    8.804092    8.800865    8.816739
## SC(n)     8.949159    8.923725    8.891599    8.894556    8.911432    8.947409
## FPE(n) 7328.785409 6912.381051 6476.163547 6284.122945 6183.262246 6201.360875
##                  7           8           9          10          11          12
## AIC(n)    8.707703    8.662529    8.449765    8.349281    8.286033    8.036372
## HQ(n)     8.804899    8.772684    8.572880    8.485355    8.435067    8.198365
## SC(n)     8.955672    8.943561    8.763860    8.696438    8.666253    8.449654
## FPE(n) 6049.532506 5782.418560 4674.292502 4227.525800 3968.539248 3091.854546
##                 13          14          15          16          17          18
## AIC(n)    7.840426    7.839335    7.835584    7.821454    7.814305    7.811937
## HQ(n)     8.015379    8.027247    8.036456    8.035285    8.041096    8.051686
## SC(n)     8.286771    8.318742    8.348054    8.366987    8.392900    8.423594
## FPE(n) 2541.782582 2539.128253 2529.758325 2494.416281 2476.817831 2471.148825
##                 19          20          21          22          23          24
## AIC(n)    7.824606    7.829917    7.840655    7.842915    7.842738    7.841158
## HQ(n)     8.077316    8.095585    8.119283    8.134502    8.147285    8.158664
## SC(n)     8.469327    8.507700    8.551501    8.586823    8.619709    8.651191
## FPE(n) 2502.872362 2516.438589 2543.875300 2549.924289 2549.795858 2546.120034
##                 25
## AIC(n)    7.821095
## HQ(n)     8.151561
## SC(n)     8.664191
## FPE(n) 2495.920431
```

From the result above using AIC as a method of selection we choose 18 as the lag, p=18, and BIC with lag of 13 to construct the multivariate methods, VAR model. However, we will go with AIC selection and choose VAR(18).

Below will be our graph for the VAR(18) model and the summary for the model. We will look at Correlation matrix of residuals, where it is -0.2456, which if unemployment rate increase we should expected a decrease in Total car sales. This does make sense because people are loosing their jobs and buying cars would not be their top priority.

```
#Fit the graph
y_model = VAR(y_tot, p=18)
summary(y_model)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: cars_ts, une_ts
## Deterministic variables: const
## Sample size: 520
## Log Likelihood: -3430.903
## Roots of the characteristic polynomial:
## 0.9933 0.9933 0.9901 0.9901 0.9806 0.9806 0.9776 0.9776 0.9774 0.9774 0.9702 0.9702 0.9663 0.9544 0.9
## Call:
## VAR(y = y_tot, p = 18)
##
```

```
##
## Estimation results for equation cars_ts:
## ========================================
## cars_ts = cars_ts.l1 + une_ts.l1 + cars_ts.l2 + une_ts.l2 + cars_ts.l3 + une_ts.l3 + cars_ts.l4 + un
##
##               Estimate Std. Error t value Pr(>|t|)
## cars_ts.l1     0.38828    0.04648   8.355 7.02e-16 ***
## une_ts.l1    -27.18509    9.32607  -2.915 0.003723 **
## cars_ts.l2     0.07060    0.05052   1.397 0.162976
## une_ts.l2     27.26635   12.24068   2.228 0.026373 *
## cars_ts.l3     0.16466    0.05049   3.261 0.001189 **
## une_ts.l3      5.91368   12.25916   0.482 0.629749
## cars_ts.l4     0.07315    0.05101   1.434 0.152200
## une_ts.l4    -10.36346   12.31076  -0.842 0.400305
## cars_ts.l5     0.03354    0.05090   0.659 0.510262
## une_ts.l5     27.37012   12.28367   2.228 0.026329 *
## cars_ts.l6     0.04035    0.05022   0.804 0.422051
## une_ts.l6    -14.47226   11.89706  -1.216 0.224405
## cars_ts.l7    -0.01014    0.04312  -0.235 0.814231
## une_ts.l7     -9.74732   19.69750  -0.495 0.620930
## cars_ts.l8     0.01401    0.04261   0.329 0.742382
## une_ts.l8    -27.09749   23.16428  -1.170 0.242660
## cars_ts.l9     0.19500    0.04310   4.524 7.64e-06 ***
## une_ts.l9     77.92780   23.07423   3.377 0.000791 ***
## cars_ts.l10   -0.09215    0.04269  -2.158 0.031391 *
## une_ts.l10   -80.35796   22.37793  -3.591 0.000363 ***
## cars_ts.l11    0.06682    0.04344   1.538 0.124708
## une_ts.l11    42.86645   23.43659   1.829 0.068010 .
## cars_ts.l12    0.52639    0.04353  12.092  < 2e-16 ***
## une_ts.l12   -21.09653   23.50753  -0.897 0.369933
## cars_ts.l13   -0.24166    0.04922  -4.910 1.25e-06 ***
## une_ts.l13    25.04055   23.70113   1.057 0.291262
## cars_ts.l14    0.09305    0.05154   1.805 0.071633 .
## une_ts.l14    46.50354   24.05118   1.934 0.053756 .
## cars_ts.l15   -0.02210    0.05194  -0.426 0.670607
## une_ts.l15   -32.41209   23.40828  -1.385 0.166801
## cars_ts.l16   -0.11962    0.05166  -2.316 0.021000 *
## une_ts.l16     8.90926   24.33933   0.366 0.714493
## cars_ts.l17   -0.03224    0.05191  -0.621 0.534856
## une_ts.l17    -6.26994   24.41151  -0.257 0.797409
## cars_ts.l18   -0.12161    0.04917  -2.473 0.013735 *
## une_ts.l18   -21.13832   17.91162  -1.180 0.238523
## const       -106.31407  100.98402  -1.053 0.292968
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 97.31 on 483 degrees of freedom
## Multiple R-Squared: 0.8284,  Adjusted R-squared: 0.8156
## F-statistic: 64.76 on 36 and 483 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation une_ts:
## ========================================
```

```
## une_ts = cars_ts.l1 + une_ts.l1 + cars_ts.l2 + une_ts.l2 + cars_ts.l3 + une_ts.l3 + cars_ts.l4 + une
##
##               Estimate Std. Error t value Pr(>|t|)
## cars_ts.l1  -1.476e-03  2.341e-04  -6.307 6.43e-10 ***
## une_ts.l1    8.805e-01  4.697e-02  18.745  < 2e-16 ***
## cars_ts.l2   2.814e-04  2.545e-04   1.106 0.269333
## une_ts.l2   -1.127e-01  6.165e-02  -1.829 0.068084 .
## cars_ts.l3  -1.643e-05  2.543e-04  -0.065 0.948506
## une_ts.l3    6.043e-02  6.175e-02   0.979 0.328233
## cars_ts.l4  -1.471e-04  2.569e-04  -0.573 0.567177
## une_ts.l4   -4.537e-02  6.201e-02  -0.732 0.464703
## cars_ts.l5   2.352e-04  2.564e-04   0.918 0.359329
## une_ts.l5    5.372e-02  6.187e-02   0.868 0.385691
## cars_ts.l6  -1.928e-04  2.529e-04  -0.762 0.446219
## une_ts.l6   -4.296e-02  5.992e-02  -0.717 0.473751
## cars_ts.l7  -3.423e-04  2.172e-04  -1.576 0.115678
## une_ts.l7    1.275e-01  9.921e-02   1.285 0.199359
## cars_ts.l8   3.097e-04  2.146e-04   1.443 0.149659
## une_ts.l8   -5.177e-03  1.167e-01  -0.044 0.964627
## cars_ts.l9  -2.946e-05  2.171e-04  -0.136 0.892124
## une_ts.l9   -1.112e-01  1.162e-01  -0.957 0.339080
## cars_ts.l10  5.372e-04  2.150e-04   2.498 0.012818 *
## une_ts.l10   1.093e-01  1.127e-01   0.970 0.332508
## cars_ts.l11 -9.721e-05  2.188e-04  -0.444 0.657062
## une_ts.l11   1.924e-02  1.180e-01   0.163 0.870581
## cars_ts.l12 -3.432e-04  2.193e-04  -1.565 0.118181
## une_ts.l12   4.642e-01  1.184e-01   3.920 0.000101 ***
## cars_ts.l13  1.166e-03  2.479e-04   4.703 3.36e-06 ***
## une_ts.l13  -5.210e-01  1.194e-01  -4.364 1.56e-05 ***
## cars_ts.l14 -7.642e-04  2.596e-04  -2.944 0.003397 **
## une_ts.l14  -4.070e-02  1.211e-01  -0.336 0.737011
## cars_ts.l15  1.238e-04  2.616e-04   0.473 0.636379
## une_ts.l15  -3.975e-02  1.179e-01  -0.337 0.736145
## cars_ts.l16  5.584e-04  2.602e-04   2.146 0.032376 *
## une_ts.l16   2.314e-01  1.226e-01   1.887 0.059721 .
## cars_ts.l17 -4.196e-04  2.614e-04  -1.605 0.109200
## une_ts.l17  -8.122e-02  1.230e-01  -0.661 0.509212
## cars_ts.l18 -1.220e-04  2.477e-04  -0.493 0.622551
## une_ts.l18  -3.616e-02  9.022e-02  -0.401 0.688731
## const        1.501e+00  5.086e-01   2.951 0.003317 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.4902 on 483 degrees of freedom
## Multiple R-Squared: 0.9274,  Adjusted R-squared: 0.9219
## F-statistic: 171.3 on 36 and 483 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##         cars_ts   une_ts
## cars_ts 9470.15 -11.7132
## une_ts   -11.71   0.2403
```

```
##
## Correlation matrix of residuals:
##         cars_ts  une_ts
## cars_ts  1.0000 -0.2456
## une_ts  -0.2456  1.0000
```

```
#plot(y_model)
```

```
quartz()
plot(y_model)
```

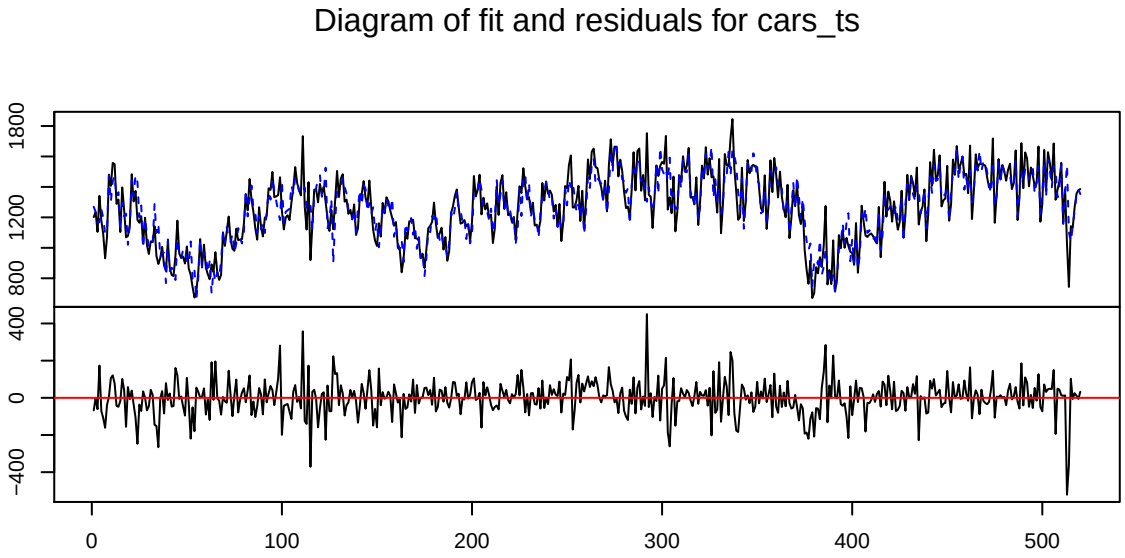


figure margins too large

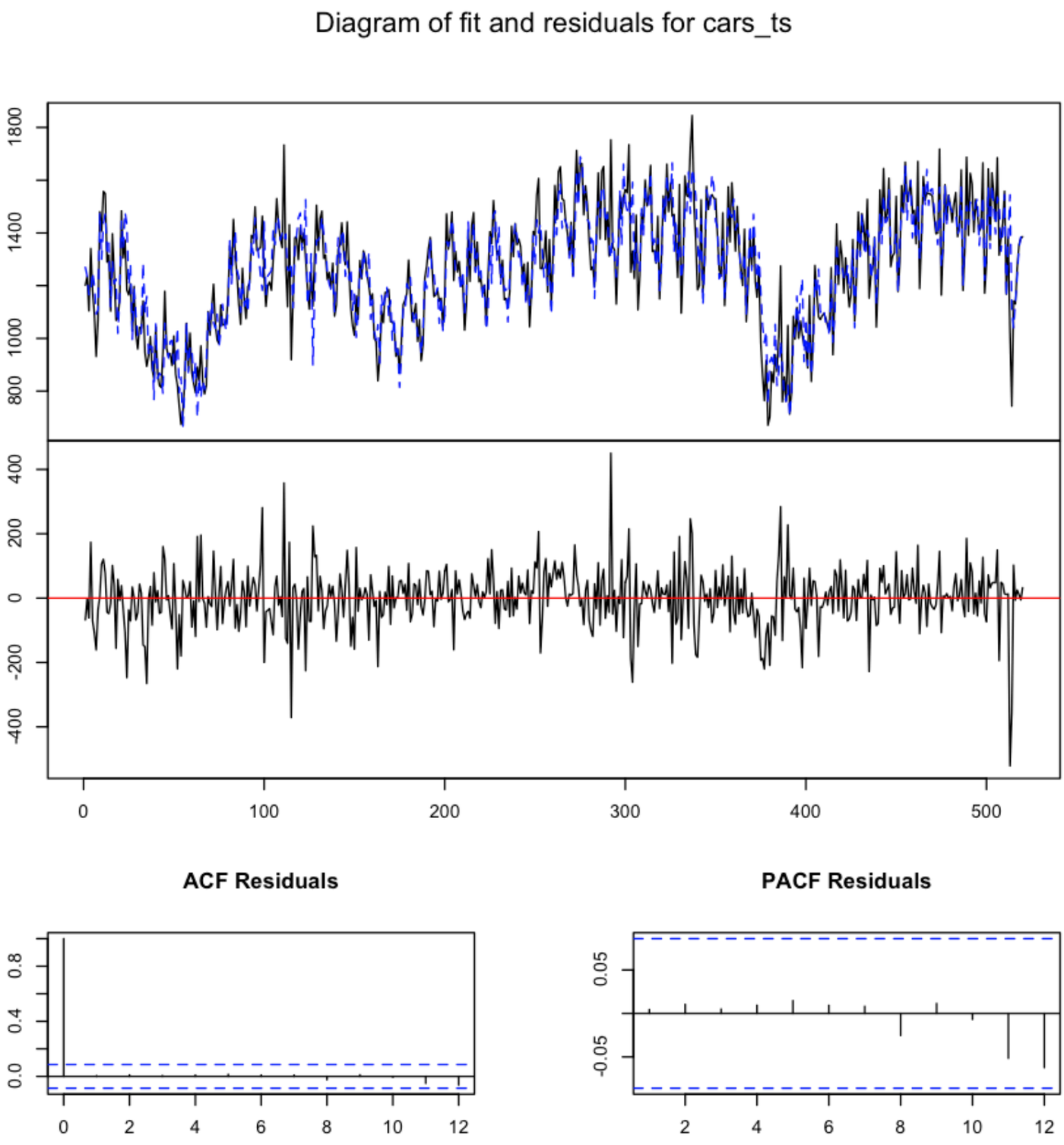
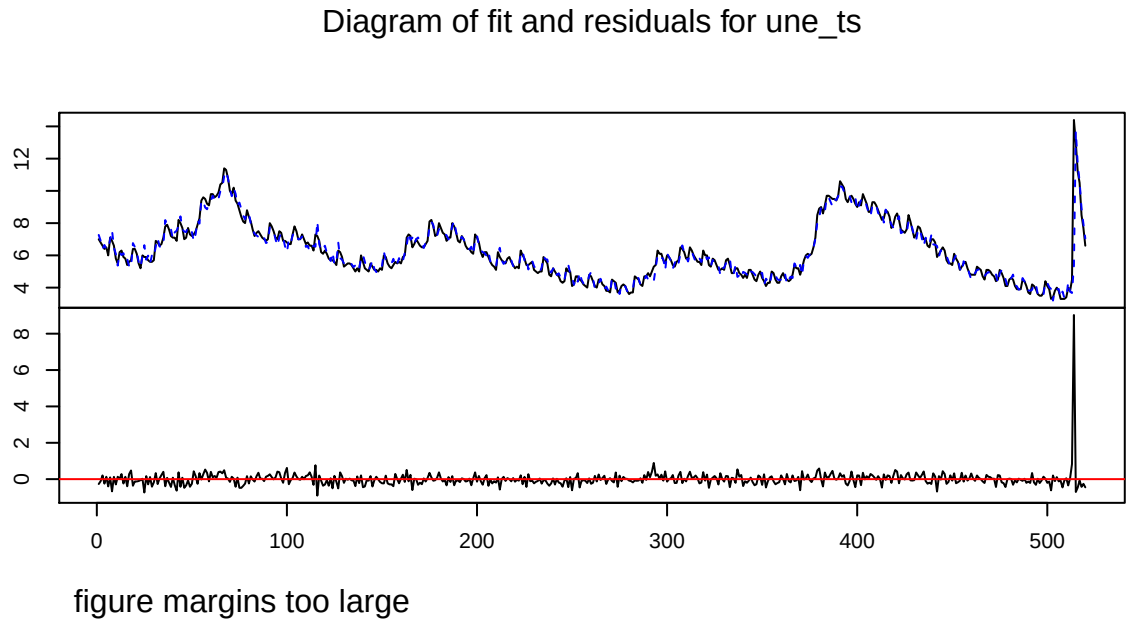


Diagram of fit and residuals for cars_ts
1800
1400
1000
800
400
200
0
-200
-400
0
100
200
300
400
500

ACF Residuals
0.8
0.4
0.0
0
2
4
6
8
10
12

PACF Residuals
0.05
-0.05
2
4
6
8
10
12



0.10.1 11. Compute, plot, interpret the respective impulse response functions.

Explanation:

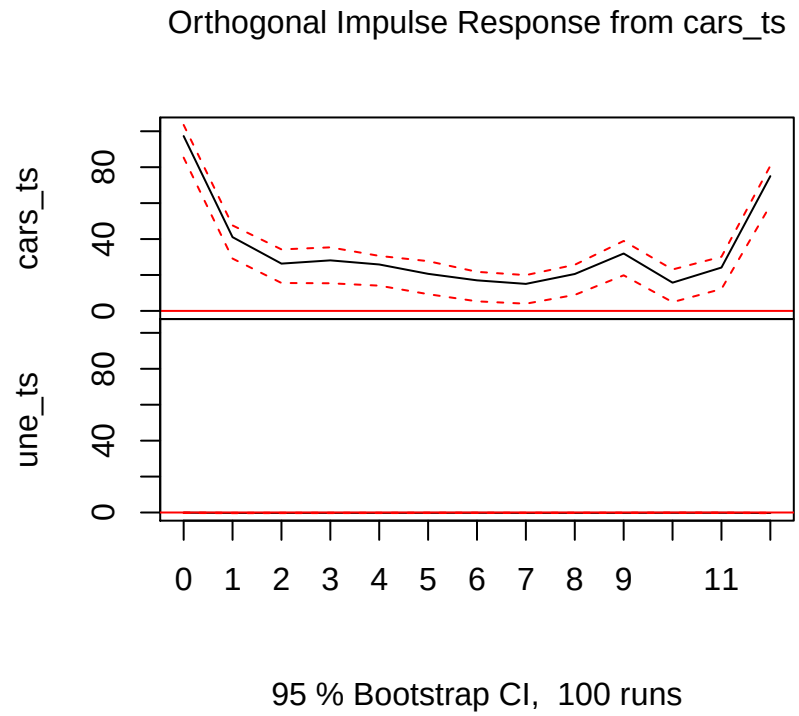
```
library(vars)
irf(y_model)
```

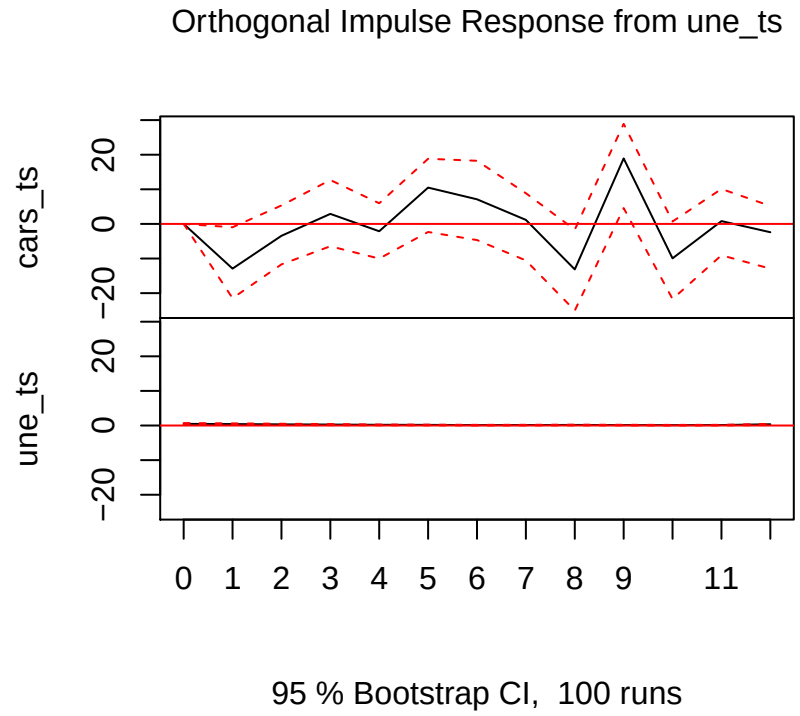
```
##
## Impulse response coefficients
## $cars_ts
##         cars_ts     une_ts
##  [1,] 97.31470 -0.1203646
##  [2,] 41.05769 -0.2496599
##  [3,] 26.31716 -0.2394939
##  [4,] 28.13211 -0.2189083
##  [5,] 25.85210 -0.2245006
##  [6,] 20.63555 -0.1964428
##  [7,] 17.01408 -0.1948993
##  [8,] 15.04524 -0.2294999
##  [9,] 20.54427 -0.2184111
## [10,] 31.94344 -0.2157360
## [11,] 15.71778 -0.1780105
##
```

```
## $une_ts
##          cars_ts    une_ts
##  [1,]   0.000000 0.4751496
##  [2,] -12.916985 0.4183838
##  [3,]  -3.433641 0.3339029
##  [4,]   2.895383 0.2769917
##  [5,]  -2.120787 0.2049523
##  [6,]  10.495172 0.1818619
##  [7,]   7.114436 0.1220070
##  [8,]   1.165442 0.1410330
##  [9,] -13.143144 0.1697253
## [10,]  18.922364 0.1335322
## [11,]  -9.945515 0.1099079
##
##
## Lower Band, CI= 0.95
## $cars_ts
##         cars_ts     une_ts
##  [1,] 85.146999 -0.2627092
##  [2,] 29.780706 -0.3933374
##  [3,] 14.334483 -0.3754645
##  [4,] 15.829549 -0.3381417
##  [5,] 14.671840 -0.2935192
##  [6,] 10.554491 -0.2695848
##  [7,]  5.251804 -0.2901884
##  [8,]  6.336132 -0.3191388
##  [9,]  9.224756 -0.3071884
## [10,] 19.365029 -0.3077205
## [11,]  5.654723 -0.2536797
##
## $une_ts
##           cars_ts     une_ts
##  [1,]   0.0000000 0.24128717
##  [2,] -21.6146168 0.20646990
##  [3,] -10.9906012 0.14878314
##  [4,]  -5.8126760 0.11990093
##  [5,] -12.7293511 0.07790528
##  [6,]  -0.2377529 0.06877602
##  [7,]  -3.6037667 0.03549390
##  [8,]  -9.1100166 0.04858749
##  [9,] -24.4462063 0.06609888
## [10,]   3.8844694 0.05010149
## [11,] -17.9836819 0.03582063
##
##
## Upper Band, CI= 0.95
## $cars_ts
##         cars_ts      une_ts
##  [1,] 102.52177 -0.02683033
##  [2,]  48.93639 -0.15249018
##  [3,]  35.08985 -0.15213102
##  [4,]  35.31136 -0.13773328
##  [5,]  33.84183 -0.13973576
##  [6,]  28.62335 -0.12132379
```

```
##  [7,]  25.00317 -0.11787758
##  [8,]  23.22789 -0.14712925
##  [9,]  28.99749 -0.13406393
## [10,]  38.68540 -0.12844463
## [11,]  23.60168 -0.08837630
##
## $une_ts
##          cars_ts    une_ts
##  [1,]  0.0000000 0.7001471
##  [2,] -0.9092774 0.6178630
##  [3,]  5.2292132 0.4842668
##  [4,] 11.0885910 0.4206238
##  [5,]  5.8634704 0.3273656
##  [6,] 21.9020121 0.2848205
##  [7,] 13.3105779 0.2094541
##  [8,]  8.4972352 0.2435200
##  [9,] -2.6695402 0.2957894
## [10,] 29.7064219 0.2388722
## [11,]  3.1884442 0.1903070
```

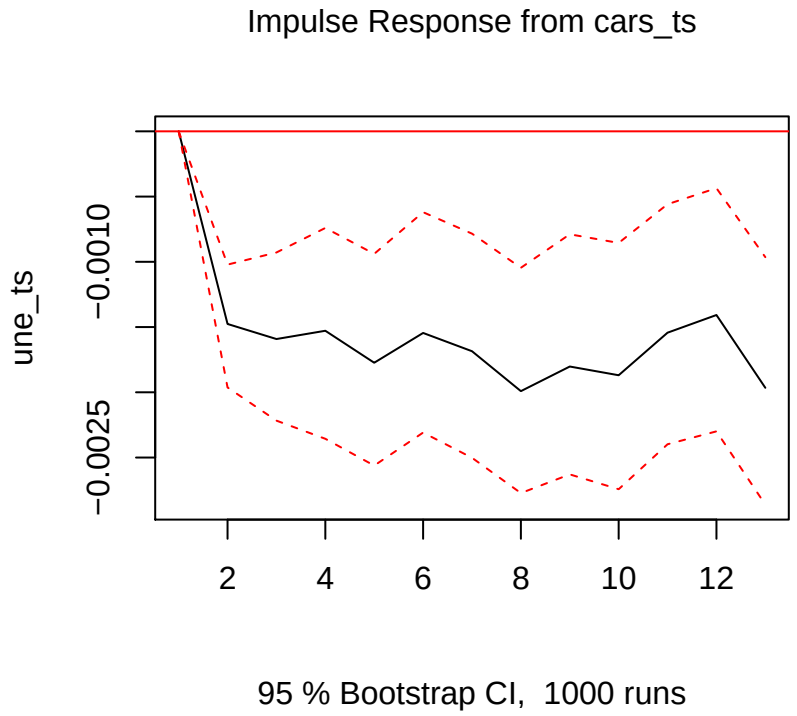
```
plot(irf(y_model, n.ahead=12))
```





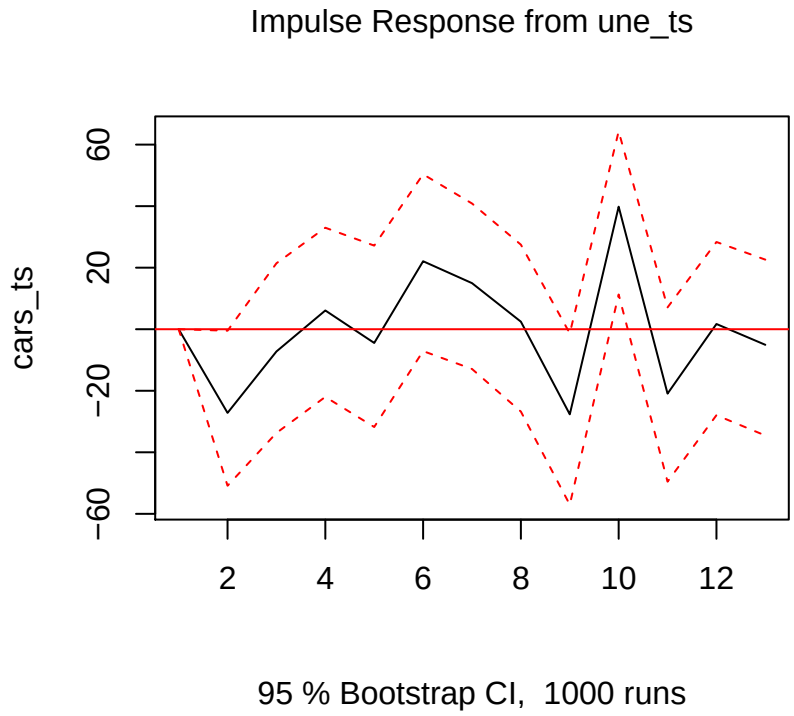
Using the graphs below, we saw Orthogonal Impulse Response from cars_ts decline rapidly from period 0 to 2 then smooth out from 2 to 11, not till period 11 to 12 we saw a sharp decrease. Notice that is not much in response to unemployment when we have an increase in car sales is nothing much at all, almost close to zero.

```
feir1 <- irf(y_model, impulse = "cars_ts", response = "une_ts",
             n.ahead = 12, ortho = FALSE, runs = 1000)
plot(feir1)
```



Here, when we increase unemployment from period 0 to 2, we saw a large decrease in total car sales. Then the number of car sales will increase back again as unemployment keeps increasing (period 2 to 6). This keeps repeating which tells us that the initial shocks of raise in unemployment will have the most impact on the number of car sales but then as time goes on, it does not weighted as much.

```
feir2 <- irf(y_model, impulse ="une_ts" , response = "cars_ts",
             n.ahead = 12, ortho = FALSE, runs = 1000)
plot(feir2)
```



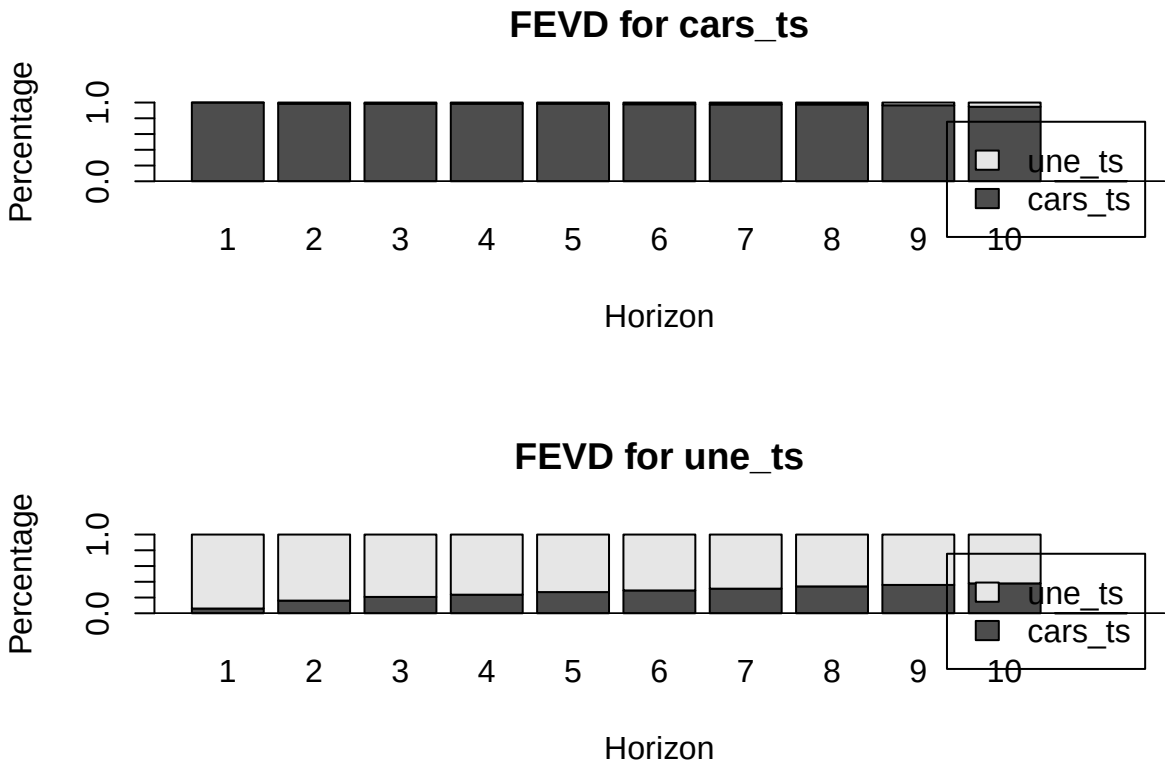
0.10.2 12. Perform a Granger-Causality test on your variables and discuss your results from the test

Using Granger-Causality test and the graphs below, we are more interst in the FEVD for UNRATE_not_ts than FEVD for car_Sales_ts because in FEVD for car_Sales_ts does not tell us much about the relationship or causation on unemploymnet rate. However, in FEVD for UNRATE_not_ts in decomposition at 1 horizon fluctuation almost as much as 96% by umeploymnet and about 4% by car sales. The percentage increase as we go up to 10. In another words that Unemployment rate is more useful to determining the forecast for Total Car Sales.

```
# test to see if umeployment granger-cause Total car sales
grangertest(une_ts ~ cars_ts, order=18)
```

```
## Granger causality test
##
## Model 1: une_ts ~ Lags(une_ts, 1:18) + Lags(cars_ts, 1:18)
## Model 2: une_ts ~ Lags(une_ts, 1:18)
##   Res.Df  Df      F    Pr(>F)
## 1    483
## 2    501 -18 4.7262 9.932e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#plot the variance
plot(fevd(y_model, n.head=12))
```



0.10.3 13. 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. ARIMA).

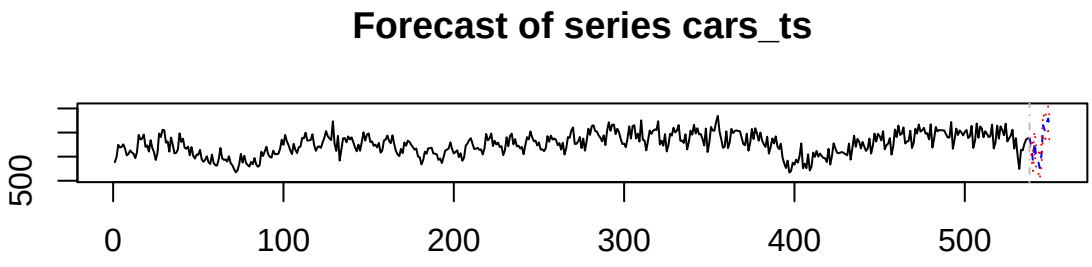
Looking at both VAR(18) and ARIMA() models for both Total car Sales and Umeployment rate, we noticed that VAR(18) model is a better forecasting since it gives us a larger intervals in both Total car Sales and Unemploymnet rate. In forecast, having a good intervals allows us to understand the future better, so in this case VAR(18) would be a better choice.

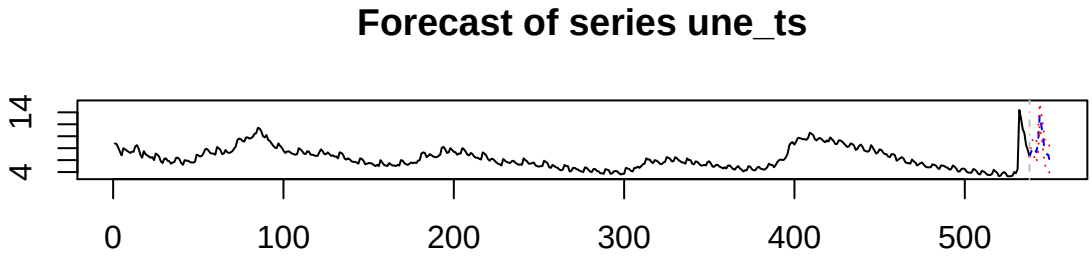
```
var.predict <- predict(object = y_model, n.ahead=12, ci=0.95)
var.predict
```

```
## $cars_ts
##           fcst     lower     upper       CI
##  [1,] 1221.5823 1030.8490 1412.3156 190.7333
##  [2,]  911.3825  702.8260 1119.9390 208.5565
##  [3,] 1307.4060 1092.4602 1522.3517 214.9457
##  [4,]  954.8779  732.9002 1176.8555 221.9776
```

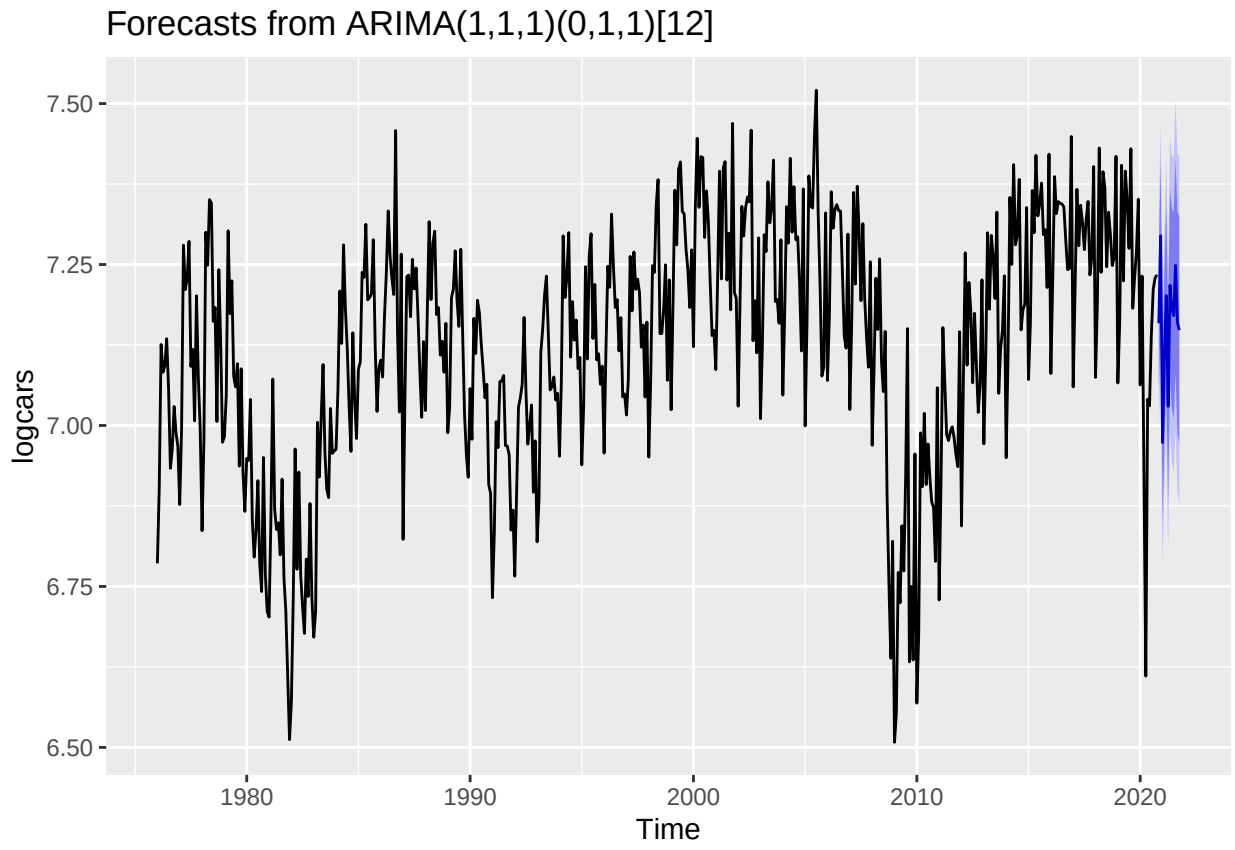
```
##  [5,]  923.3625  695.6374 1151.0876 227.7251
##  [6,]  763.1885  530.9868  995.3902 232.2017
##  [7,] 1030.7215  795.7234 1265.7195 234.9981
##  [8,] 1653.1840 1416.3320 1890.0360 236.8520
##  [9,] 1538.5448 1296.9174 1780.1722 241.6274
## [10,] 1708.2123 1455.8652 1960.5593 252.3470
## [11,] 1788.8441 1533.8774 2043.8108 254.9667
## [12,] 1520.8502 1261.5262 1780.1741 259.3240
##
## $une_ts
##             fcst     lower     upper        CI
##  [1,]  7.390158  6.429466  8.350849 0.9606919
##  [2,]  8.290238  6.935691  9.644784 1.3545462
##  [3,]  8.024035  6.448148  9.599922 1.5758870
##  [4,]  7.327984  5.606868  9.049100 1.7211158
##  [5,]  8.792595  6.971273 10.613917 1.8213218
##  [6,] 13.609840 11.714450 15.505231 1.8953907
##  [7,] 11.720477  9.772245 13.668709 1.9482323
##  [8,]  9.676019  7.657518 11.694521 2.0185013
##  [9,]  7.247799  5.157761  9.337836 2.0900377
## [10,]  7.009297  4.860915  9.157678 2.1483816
## [11,]  6.905496  4.718335  9.092657 2.1871613
## [12,]  5.670977  3.440509  7.901446 2.2304685
```

```r
#plot
plot(var.predict)
```

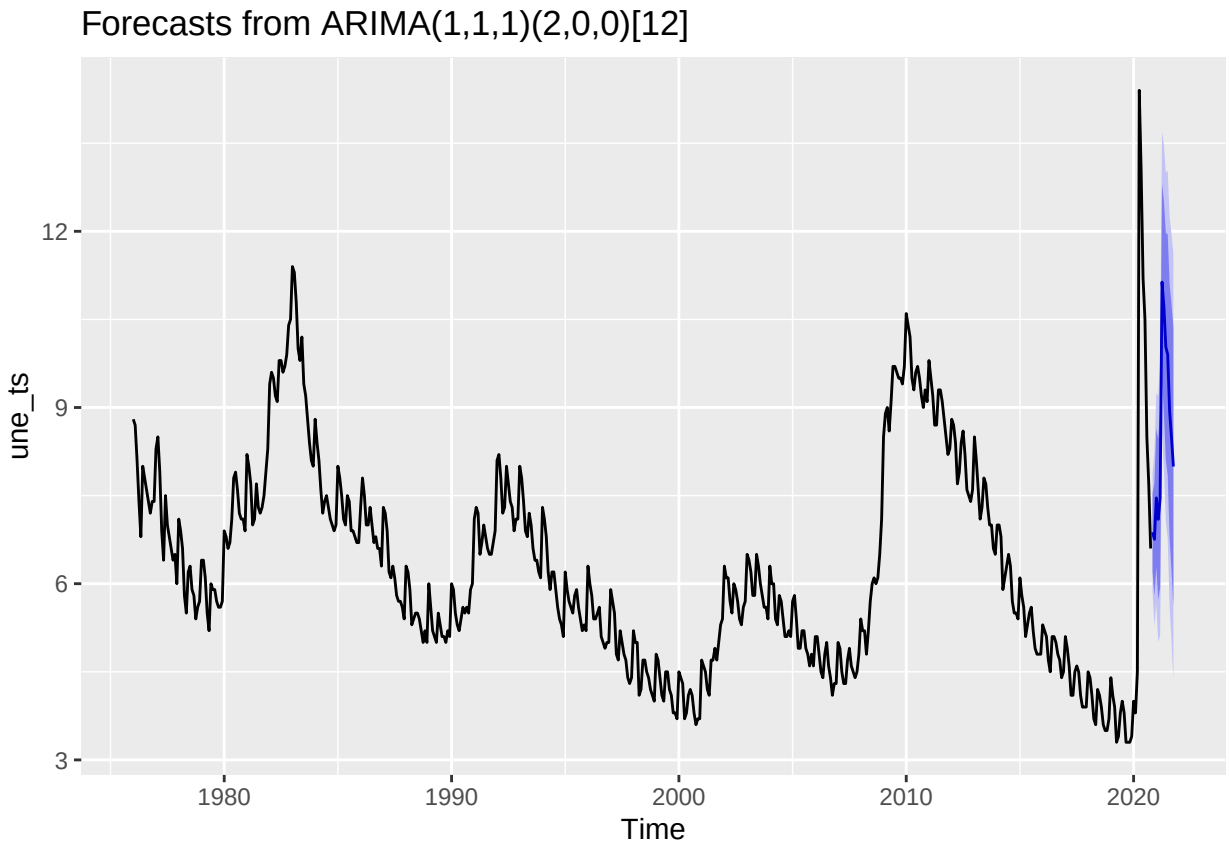




```
#Total Car Sales
autoplot(forecast(diff2.5,h=12))
```




```
#Unemployment rate
autoplot(forecast(diff3.7,h=12))
```



Look for the R-squared for both variables:

```
summary(y_model$varresult$cars_ts)$adj.r.squared
```

```
## [1] 0.8155869
```

```
# result: adjust-R.squared = 0.8136422

summary(y_model$varresult$une_ts)$adj.r.squared
```

```
## [1] 0.921947
```

```
# adjust R.squared = 0.9221744

accuracy(y_model$varresult[[1]])
```

```
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -5.913059e-15 93.78867 67.23122 -0.5934308 5.577726 0.3640269
```

Look for the R-squared for both variables:

```r
summary(diff2.5)
```

```
##
## Call:
## arima(x = logcars, order = c(1, 1, 1), seasonal = list(order = c(0, 1, 1)))
##
## Coefficients:
##          ar1      ma1     sma1
##       0.1727  -0.6674  -0.7677
## s.e.  0.0742   0.0554   0.0381
##
## sigma^2 estimated as 0.006567:  log likelihood = 568.77,  aic = -1129.54
##
## Training set error measures:
##                        ME       RMSE        MAE         MPE      MAPE      MASE
## Training set -0.001258827 0.08005903 0.05747208 -0.02527777 0.8109746 0.5291824
##                      ACF1
## Training set 0.001306447
```

```r
# result: adjust-R.squared = 0.8136422

summary(diff3.7)
```

```
##
## Call:
## arima(x = une_ts, order = c(1, 1, 1), seasonal = list(order = c(2, 0, 0)))
##
## Coefficients:
##           ar1     ma1    sar1    sar2
##       -0.8533  0.9289  0.4070  0.3715
## s.e.   0.0597  0.0450  0.0889  0.0894
##
## sigma^2 estimated as 0.2638:  log likelihood = -409.14,  aic = 828.27
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE    MAPE      MASE
## Training set 0.004412565 0.5131745 0.2024289 0.02101432 3.03623 0.6225905
##                     ACF1
## Training set -0.028649
```

```r
# adjust R.squared = 0.9221744
```

0.10.4 14: Backtest your ARIMA model. Begin by partitioning your data set into an estimation set and a

prediction set.

To set up our test/train, we use 8/2 ratio, where around 80% of the data set will be use for estimation and around 20% for prediction. Some of our computers were having computation memory issues with a 2/3 split so we decided to lessen the amount of predictions being backtested.

part 14a.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.

Recursive back test for Total Car Sales, we are using the log of Total Car Sales here because we want to reduce the scales.

To calculate MAPE, we take the errors at step h = 12 only and calculate the absolute value divided by the actuals values, and then take the cumulative mean of those numbers.

```
library(MTS)
```

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

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

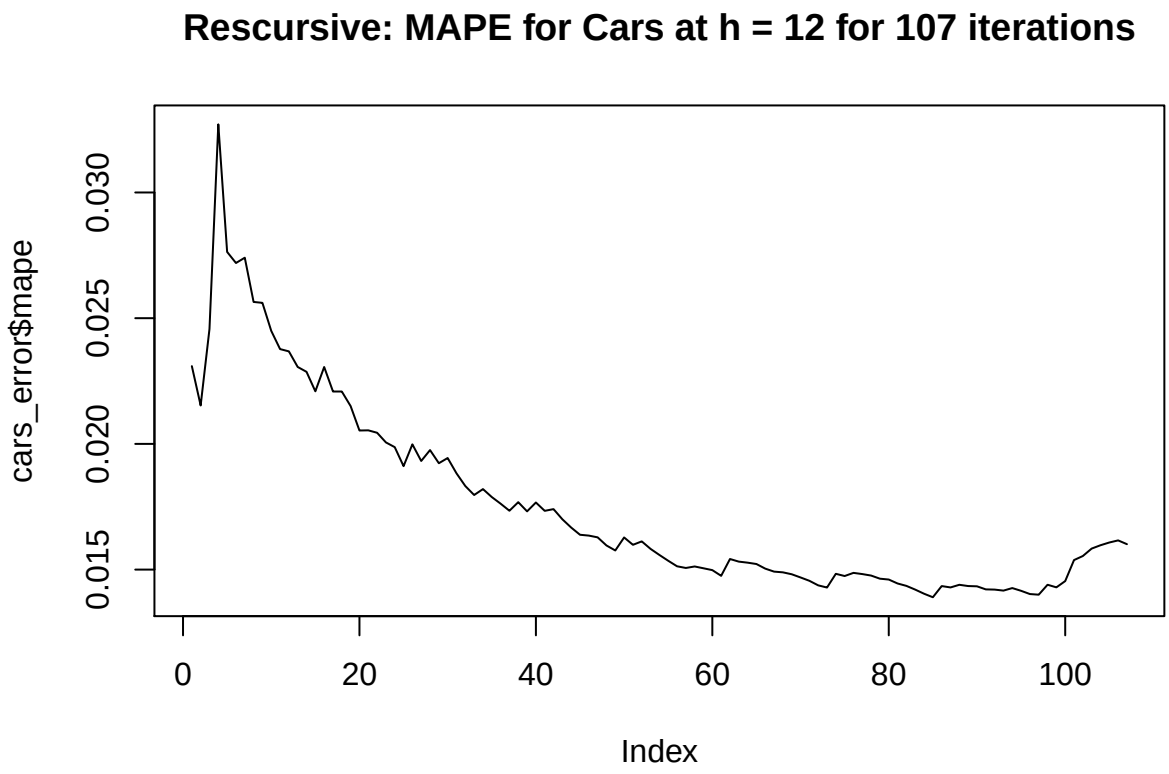
```
## The following object is masked from 'package:tidyquant':
##
##     VAR
```

```
## The following object is masked from 'package:TTR':
##
##     VMA
```

```
Cars <- backtest(diff2.5, logcars, 420, h=12)
```

```
## [1] "RMSE of out-of-sample forecasts"
##  [1] 0.1371647 0.1441078 0.1562963 0.1506019 0.1552046 0.1585596 0.1495004
##  [8] 0.1562013 0.1511889 0.1571492 0.1520695 0.1596973
## [1] "Mean absolute error of out-of-sample forecasts"
##  [1] 0.09246954 0.09826550 0.10675677 0.10503760 0.10548264 0.11032751
##  [7] 0.10160933 0.10882049 0.10413453 0.11170649 0.10381579 0.11451867
```

```
#Cars$error

cars_error = data.frame(Cars$error)
cars_error = cars_error%>% slice(1:107)
cars_error$actual = logcars[432:538]

cars_error$abs = abs(cars_error$X12/cars_error$actual)
cars_error$mape = cummean(cars_error$abs)

plot(cars_error$mape, type = 'l', main = 'Rescursive: MAPE for Cars at h = 12 for 107 iterations')
```

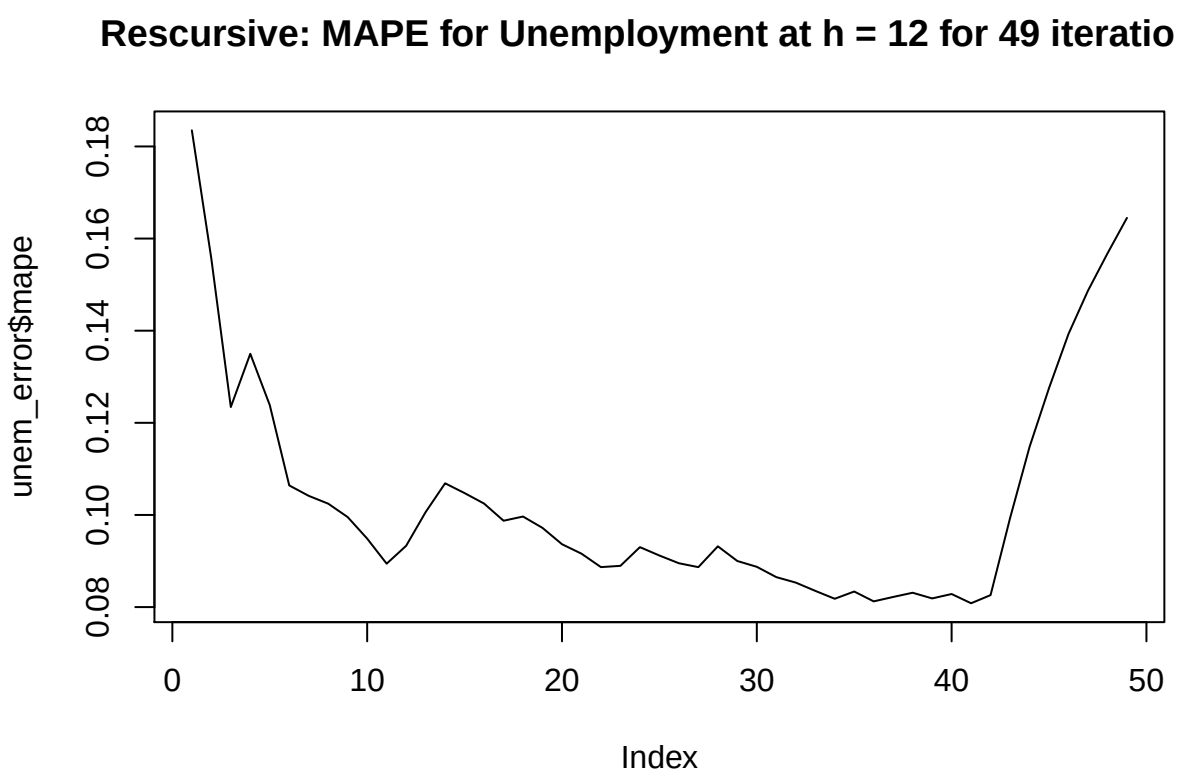


Recurive back test for Unemployment rate: Due to computation difficulties giving non-finite values we had to change the train test to around 90% and the test set to around 10% of the data.

```
library(MTS)
Unem <- backtest(diff3.7, une_ts, 478, h=12)
```

```
## [1] "RMSE of out-of-sample forecasts"
##  [1] 1.953339 2.949984 3.648571 4.168774 4.477712 3.850027 2.676294 2.707819
##  [9] 2.729808 2.778752 2.818844 2.878576
## [1] "Mean absolute error of out-of-sample forecasts"
##  [1] 0.7143138 1.0781061 1.3480646 1.5262616 1.6129268 1.4225636 1.1391118
##  [8] 1.1929793 1.1852515 1.2053711 1.2489741 1.2903491
```

```
#Unem$error

unem_error = data.frame(Unem$error)
unem_error = unem_error%>% slice(1:49)
unem_error$actual = une_ts[490:538]

unem_error$abs = abs(unem_error$X12/unem_error$actual)
unem_error$mape = cummean(unem_error$abs)

plot(unem_error$mape, type = 'l', main = 'Rescursive: MAPE for Unemployment at h = 12 for 49 iterations
```



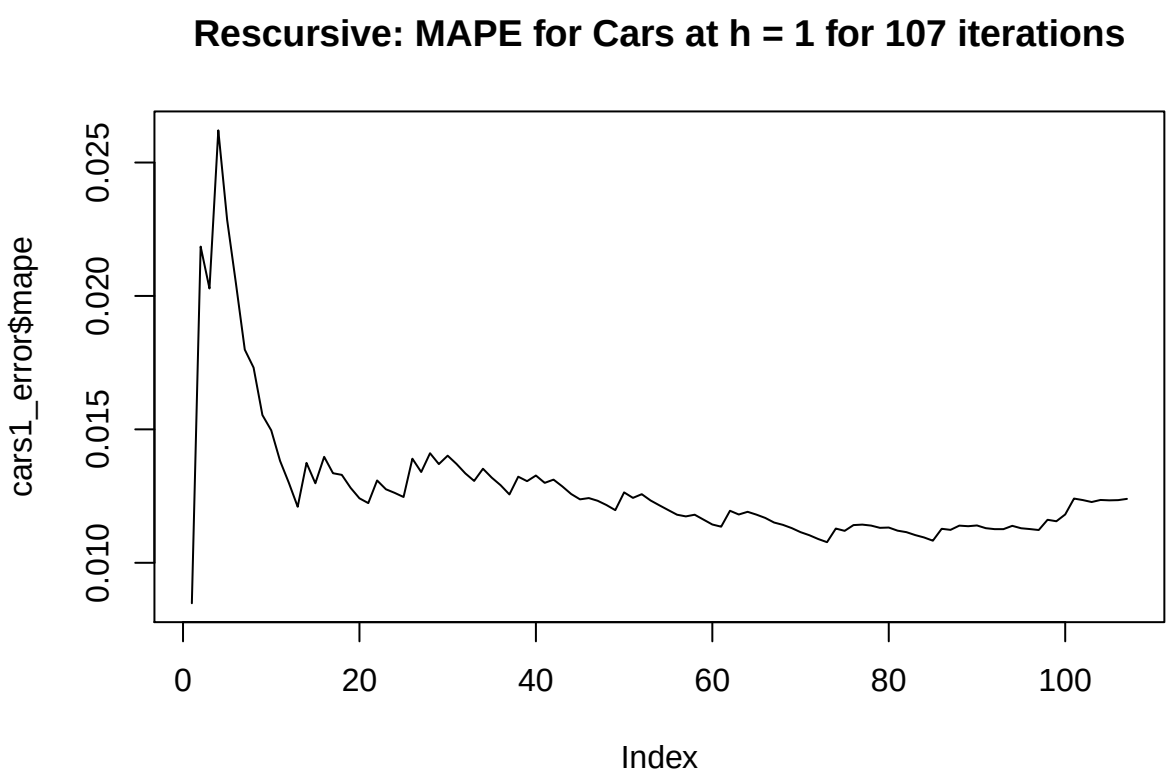
part 14b.Shorten your forecast horizon to only 1-step ahead. Compute the absolute percentage error at each iteration, and plot.

Here is for Total cars sales:

```
Cars1 <- backtest(diff2.5, logcars, 431, h=1)
```

```
## [1] "RMSE of out-of-sample forecasts"
## [1] 0.1340932
## [1] "Mean absolute error of out-of-sample forecasts"
## [1] 0.08860184
```

```
#Cars1$error

cars1_error = data.frame(Cars1$error)
cars1_error = cars1_error%>% slice(1:107)
cars1_error$actual = logcars[432:538]

cars1_error$abs = abs(cars1_error$Cars1.error/cars1_error$actual)
cars1_error$mape = cummean(cars1_error$abs)

plot(cars1_error$mape, type = 'l', main = 'Rescursive: MAPE for Cars at h = 1 for 107 iterations')
```

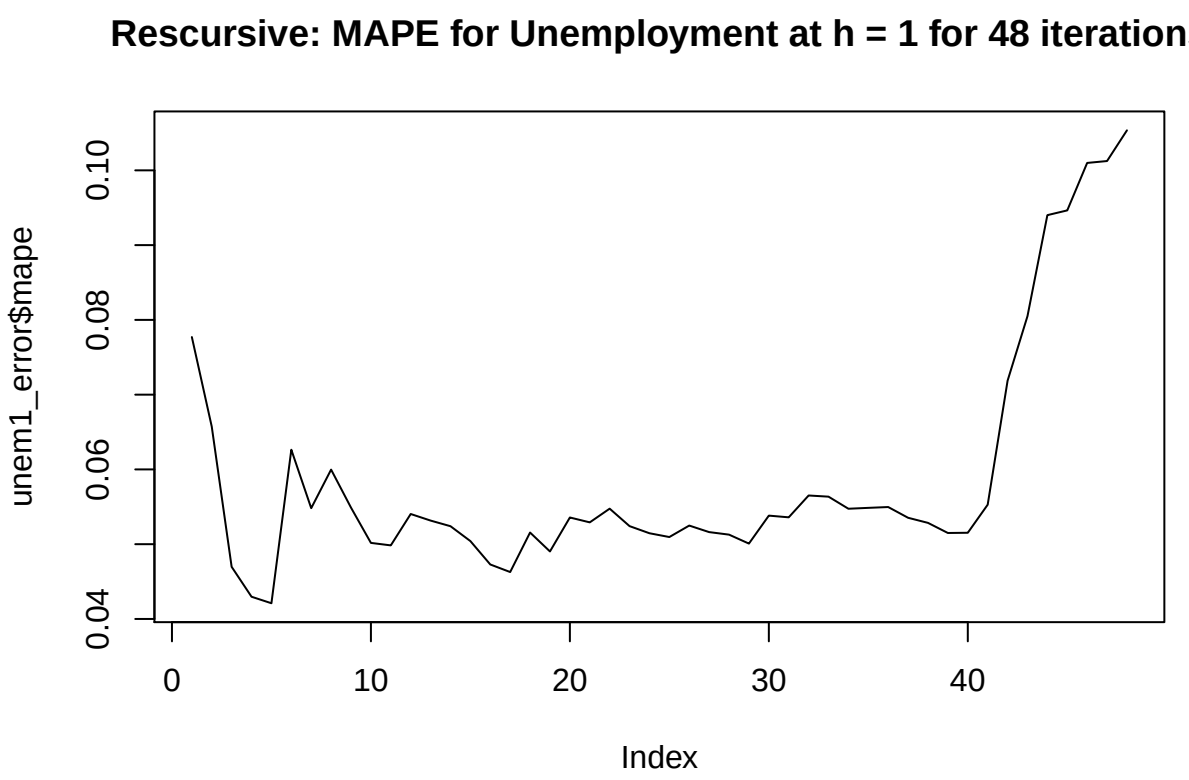


Below is to look for unemployment rate. We had an issue where if we forecasted 49 iterations for h = 12, the backtest using h = 1 would throw an error for h = 1 of non-finite finite values. This was the case with 50,51,52,53 as well, so we decided to settle on 48 iterations for h = 1 and 49 for h = 12. We believe we have enough observations so that this difference will not have a huge impact.

```
Unem1 <- backtest(diff3.7, une_ts, 490, h=1)
```

```
## [1] "RMSE of out-of-sample forecasts"
## [1] 2.1812
## [1] "Mean absolute error of out-of-sample forecasts"
## [1] 0.8452364
```

```
#Unem1$error

unem1_error = data.frame(Unem1$error)
unem1_error$actual = une_ts[491:538]

unem1_error$abs = abs(unem1_error$Unem1.error/unem1_error$actual)
unem1_error$mape = cummean(unem1_error$abs)

plot(unem1_error$mape, type = 'l', main = 'Rescursive: MAPE for Unemployment at h = 1 for 48 iterations
```



part 14c.Based on your findings above, does your model perform better at longer or shorter horizon forecasts?

The Mean Absolute Percentage Error (MAPE) is a measure of how accurate a forecast is. Based on our findings, our model performs better at lower h values for both our Unemployment Data and our Car Sales Data. If we look at both MAPE graphs, we see lower error percentage numbers for both datasets at shorter horizon forecasts. However, the difference in MAPE is not large between h = 1 and h = 12.

part 14d.
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.

```
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)
}
```

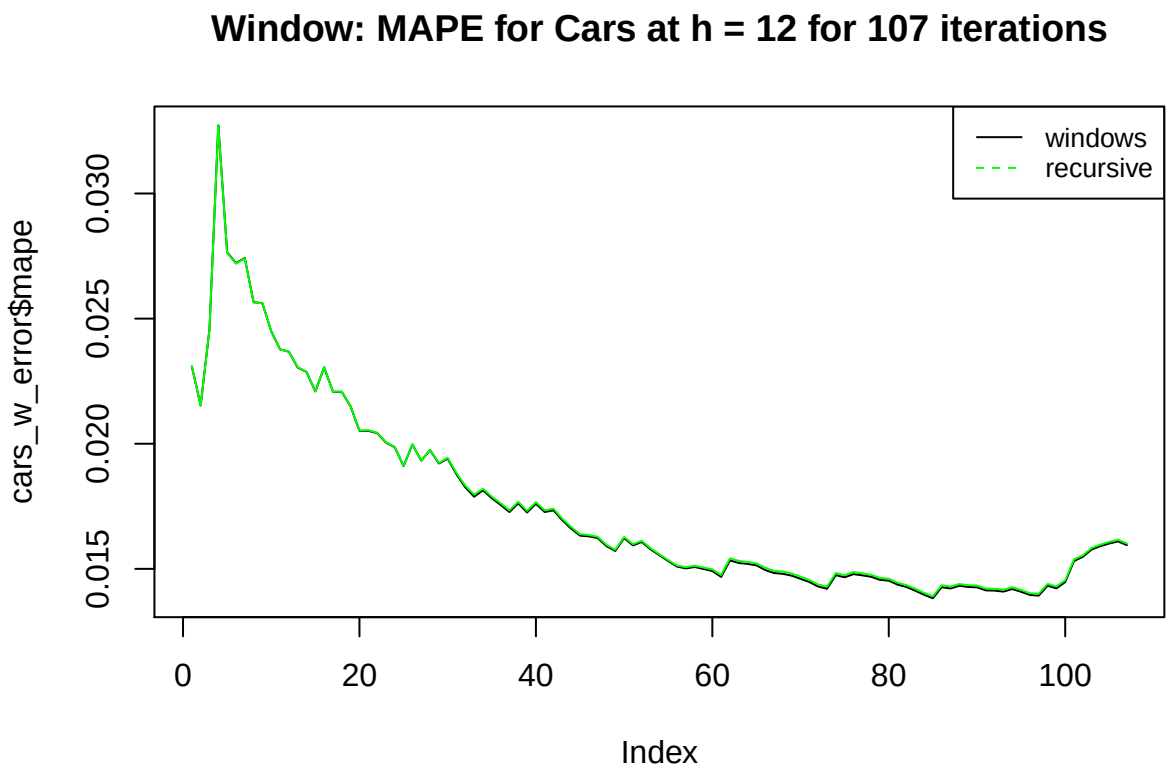
Window backtesting for cars using h = 12 and corresponding MAPE plot.

```
Carsw <- window_backtesting(diff2.5, logcars, 420, h=12)
```

```
## [1] "RMSE of out-of-sample forecasts"
```

```
##  [1] 0.1370922 0.1439405 0.1565998 0.1505141 0.1552564 0.1583157 0.1492850
##  [8] 0.1559644 0.1509335 0.1569022 0.1519412 0.1595020
## [1] "Mean absolute error of out-of-sample forecasts"
##  [1] 0.09228188 0.09832791 0.10706988 0.10514608 0.10570835 0.11034457
##  [7] 0.10173984 0.10876527 0.10391814 0.11154782 0.10344100 0.11404431
```

```
cars_w_error = data.frame(Carsw$error)

cars_w_error = cars_w_error%>% slice(1:107)
cars_w_error$actual = logcars[432:538]

cars_w_error$abs = abs(cars_w_error$X12/cars_w_error$actual)
cars_w_error$mape = cummean(cars_w_error$abs)

plot(cars_w_error$mape, type = 'l', main = 'Window: MAPE for Cars at h = 12 for 107 iterations')
lines(cars_error$mape, type = 'l', col = 'green')
legend('topright',legend=c("windows", "recursive"),
       col=c("black", "green"), lty=1:2, cex=0.8)
```



Window testing for cars using h = 1 and the MAPE plot.

```
Cars1w <- window_backtesting(diff2.5, logcars, 431, h=1)
```

```
## [1] "RMSE of out-of-sample forecasts"
## [1] 0.1352218
## [1] "Mean absolute error of out-of-sample forecasts"
## [1] 0.08926636
```

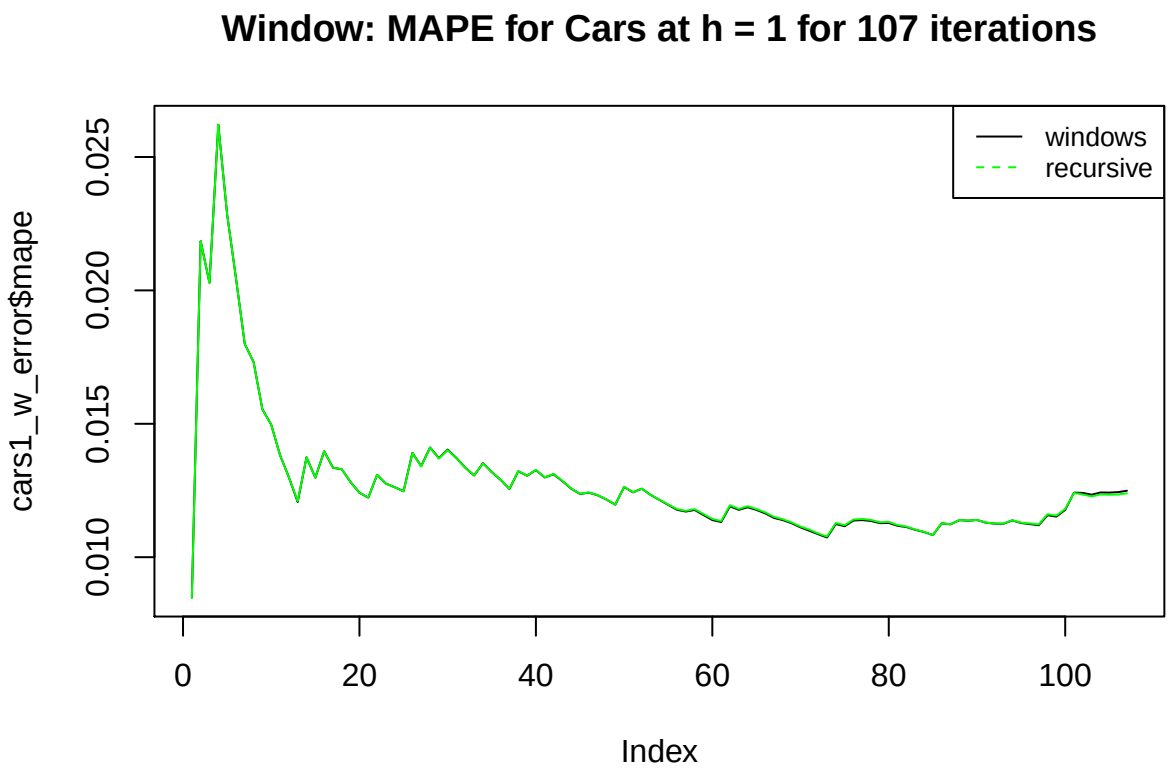
```
Cars1w$error
```

```
##                 [,1]
##   [1,]  0.0606233811
##   [2,] -0.2410058716
##   [3,]  0.1212281547
##   [4,]  0.3197329098
##   [5,] -0.0667893717
##   [6,]  0.0616107603
##   [7,]  0.0225547037
##   [8,] -0.0890443283
##   [9,]  0.0097059136
##  [10,] -0.0692991628
##  [11,] -0.0169843523
##  [12,]  0.0272833247
##  [13,]  0.0070921048
##  [14,] -0.2476989010
##  [15,]  0.0159497111
##  [16,]  0.2105109839
##  [17,] -0.0247655584
##  [18,]  0.0898229111
##  [19,]  0.0300066762
##  [20,] -0.0351727189
##  [21,]  0.0631043643
##  [22,] -0.2183110084
##  [23,]  0.0394859659
##  [24,]  0.0681391548
##  [25,] -0.0638640629
##  [26,] -0.3464883961
##  [27,] -0.0035817772
##  [28,]  0.2428908533
##  [29,]  0.0178707930
##  [30,]  0.1728912838
##  [31,] -0.0305054184
##  [32,] -0.0153652141
##  [33,]  0.0276175361
##  [34,] -0.2063287825
##  [35,]  0.0119662262
##  [36,]  0.0209086271
##  [37,] -0.0005927629
##  [38,] -0.2676810963
##  [39,] -0.0481723934
##  [40,]  0.1581432411
##  [41,]  0.0138168906
##  [42,]  0.1341204786
##  [43,] -0.0147924927
##  [44,]  0.0035790579
##  [45,] -0.0266048060
##  [46,] -0.1069163308
##  [47,]  0.0568254870
##  [48,] -0.0329600269
##  [49,]  0.0208971844
##  [50,] -0.3197740371
```

```
##  [51,] -0.0165849830
##  [52,]  0.1442492005
##  [53,] -0.0021330664
##  [54,]  0.0164318068
##  [55,]  0.0131773699
##  [56,]  0.0118581110
##  [57,] -0.0591087040
##  [58,] -0.1109049275
##  [59,] -0.0039946951
##  [60,] -0.0023368679
##  [61,]  0.0508453708
##  [62,] -0.3381366509
##  [63,] -0.0249591766
##  [64,]  0.1325154560
##  [65,] -0.0390447074
##  [66,]  0.0238739970
##  [67,] -0.0013466361
##  [68,] -0.0433343819
##  [69,] -0.0234298909
##  [70,]  0.0033647173
##  [71,] -0.0150485148
##  [72,]  0.0126226921
##  [73,] -0.0136698780
##  [74,] -0.3411279684
##  [75,] -0.0359230880
##  [76,]  0.2036101791
##  [77,] -0.0915038077
##  [78,]  0.0647314494
##  [79,]  0.0368652538
##  [80,] -0.0860715420
##  [81,] -0.0189063811
##  [82,] -0.0538278439
##  [83,]  0.0180184063
##  [84,]  0.0295533769
##  [85,]  0.0080574149
##  [86,] -0.3440309938
##  [87,] -0.0584816689
##  [88,]  0.1845811200
##  [89,] -0.0684770247
##  [90,]  0.1021440070
##  [91,]  0.0178939382
##  [92,] -0.0560690295
##  [93,]  0.0813344928
##  [94,] -0.1662801156
##  [95,]  0.0137919779
##  [96,]  0.0551443735
##  [97,] -0.0515946843
##  [98,] -0.3401151309
##  [99,]  0.0439442066
## [100,] -0.2575740454
## [101,] -0.5084783611
## [102,] -0.0784386865
## [103,]  0.0390833762
## [104,]  0.1504739656
```

```
## [105,]  0.0860581270
## [106,]  0.1033929198
## [107,]  0.1318721765
```

```
cars1_w_error = data.frame(Cars1w$error)
cars1_w_error = cars1_w_error%>% slice(1:107)
cars1_w_error$actual = logcars[432:538]

cars1_w_error$abs = abs(cars1_w_error$Cars1w.error/cars1_w_error$actual)
cars1_w_error$mape = cummean(cars1_w_error$abs)

plot(cars1_w_error$mape, type = 'l', main = 'Window: MAPE for Cars at h = 1 for 107 iterations')
lines(cars1_error$mape, type = 'l', col = 'green')
legend('topright',legend=c("windows", "recursive"),
       col=c("black", "green"), lty=1:2, cex=0.8)
```



Window backtesting for unemployment using h = 12 and corresponding MAPE plot.

```
Unemw <- window_backtesting(diff3.7, une_ts, 478, h=12)
```

```
## [1] "RMSE of out-of-sample forecasts"
##  [1] 2.022267 3.121930 3.883273 4.482884 4.849386 4.105410 2.676356 2.707576
##  [9] 2.729417 2.777987 2.819612 2.880626
## [1] "Mean absolute error of out-of-sample forecasts"
##  [1] 0.7205089 1.1193875 1.3899999 1.5963254 1.6967906 1.4700912 1.1419906
##  [8] 1.1965788 1.1886787 1.2069590 1.2512933 1.2932489
```

```
Unemw$error
```

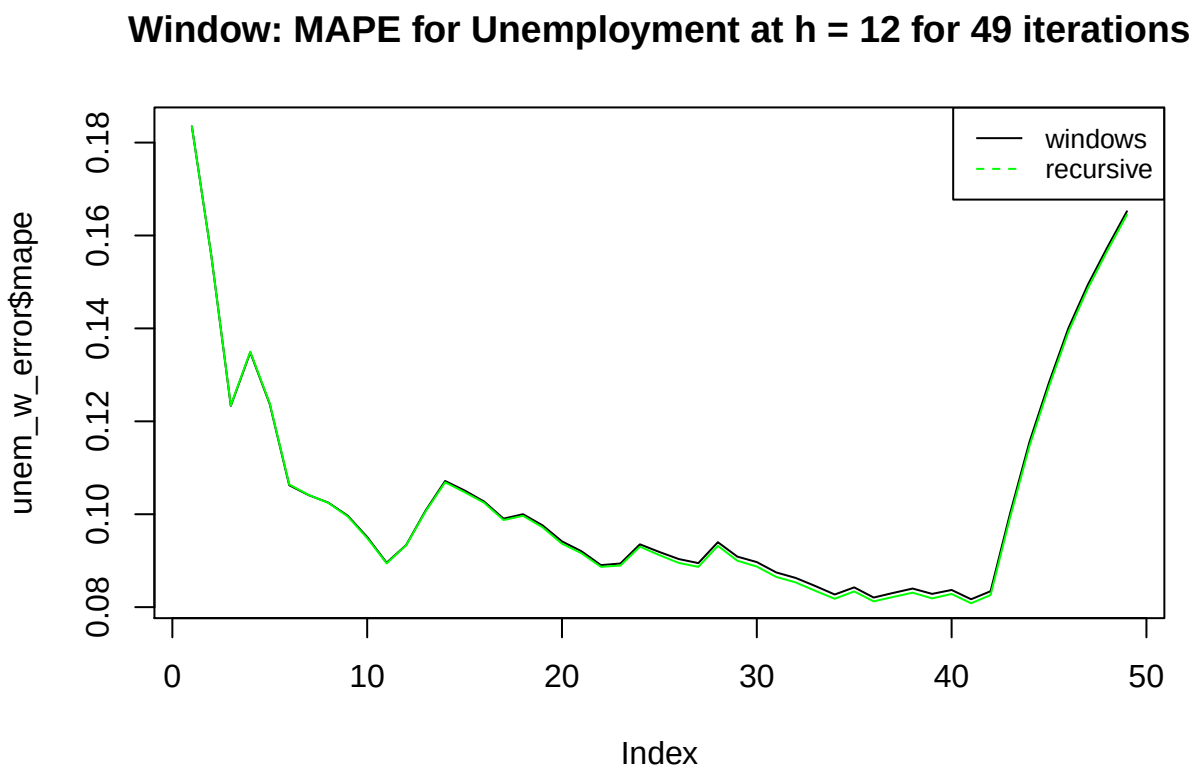
```
##               [,1]          [,2]          [,3]         [,4]          [,5]
##  [1,]  0.169640497   0.257913232   0.203188684   0.24483104   0.325256286
##  [2,]  0.169640497   0.757913232   0.103188684   0.14483104  -0.074743714
##  [3,] -0.082006655  -0.063693481  -0.004621367   0.20244361  -0.198021647
##  [4,] -0.182006655  -0.163693481  -0.404621367   0.00244361   0.401978353
##  [5,]  0.064363909   0.276701138  -0.119384904   0.28615780   0.141085936
##  [6,] -0.335636091   0.076701138   0.480615096   0.28615780   0.041085936
##  [7,] -0.427839677  -0.050241969  -0.220458071  -0.07664616   0.096402706
##  [8,]  0.172160323  -0.050241969  -0.320458071  -0.27664616  -0.003597294
##  [9,] -0.176002811  -0.037728201   0.130284014   0.13473645  -0.191119252
## [10,] -0.276002811  -0.237728201   0.030284014  -0.16526355  -0.091119252
## [11,]  0.167492050   0.171531002  -0.154680604  -0.03467493   0.006213643
## [12,]  0.067492050  -0.128468998  -0.054680604   0.56532507  -0.193786357
## [13,] -0.342315639  -0.237198791  -0.207334852  -0.26130377  -0.454648549
## [14,] -0.242315639   0.362801209  -0.407334852  -0.56130377  -0.954648549
## [15,]  0.050383135   0.012581016  -0.166093355  -0.27294596  -0.270433683
## [16,] -0.149616865  -0.287418984  -0.666093355  -0.27294596   0.129566317
## [17,] -0.178726459  -0.286473554  -0.284510830  -0.26224856  -0.211275772
## [18,] -0.678726459  -0.286473554   0.115489170  -0.16224856  -0.311275772
## [19,]  0.035014401   0.083583929   0.159346063   0.28924131   0.123672067
## [20,]  0.435014401   0.183583929   0.059346063  -0.11075869  -0.076327933
## [21,]  0.066466305   0.187983048   0.015146437  -0.08966082   0.059426489
## [22,] -0.033533695  -0.212016952  -0.184853563  -0.08966082   0.059426489
## [23,] -0.194488671  -0.318416349  -0.186045418  -0.24481409  -0.150738336
## [24,] -0.394488671  -0.318416349  -0.186045418   0.35518591  -0.250738336
## [25,]  0.169568820   0.141126726   0.263369838   0.31735671   0.218270415
## [26,]  0.169568820   0.741126726   0.163369838   0.01735671  -0.181729585
## [27,]  0.109779247   0.152298369   0.042892897   0.05245245   0.037031928
## [28,]  0.009779247  -0.147701631  -0.357107103  -0.04754755   0.637031928
## [29,] -0.129560736  -0.136298422  -0.159905668  -0.01077523  -0.164394755
## [30,] -0.529560736  -0.236298422   0.440094332  -0.11077523  -0.364394755
## [31,] -0.010984579   0.150342852   0.006664415  -0.10136238  -0.126949139
## [32,]  0.589015421   0.050342852  -0.193335585  -0.40136238  -0.226949139
## [33,] -0.163255012  -0.288292842  -0.328555519  -0.30352935  -0.185445572
## [34,] -0.363255012  -0.588292842  -0.428555519  -0.30352935   0.014554428
## [35,] -0.008577581   0.044388699   0.187378209   0.35764830   0.532440413
## [36,] -0.108577581   0.044388699   0.387378209   1.05764830   0.232440413
## [37,]  0.139667675   0.303688379   0.469682299   0.30938448   0.384972039
## [38,]  0.339667675   1.003688379   0.169682299   0.10938448  -0.215027961
## [39,]  0.130137940  -0.061539776  -0.013386495  -0.22881156  -0.099472466
## [40,] -0.169862060  -0.261539776  -0.613386495  -0.12881156   0.300527534
## [41,]  0.061345100  -0.143803800  -0.002936519  -0.06706642   0.133418937
## [42,] -0.538654900  -0.043803800   0.397063481   0.13293358  -0.066581063
## [43,]  0.161612900   0.116779121   0.333051956   0.28050820   0.101542816
## [44,]  0.561612900   0.316779121   0.133051956  -0.21949180   0.101542816
## [45,]  0.205909307   0.144409914  -0.042076245   0.09091594   0.091214948
## [46,]  0.005909307  -0.355590086  -0.042076245   0.09091594   0.191214948
## [47,] -0.198012259  -0.075171760  -0.084322923  -0.08702991  -0.083109249
## [48,] -0.198012259  -0.075171760   0.015677077   0.51297009  -0.283109249
## [49,] -0.005759577  -0.005101578   0.001968946  -0.01748198   0.902576005
## [50,]  0.094240423   0.594898422  -0.198031054   0.68251802  10.802576005
```

```
## [51,]  0.007557459  -0.012253285   0.908470111  11.26022506   9.854375725
## [52,] -0.192442541   0.687746715  10.808470111   9.86022506   8.054375725
## [53,]  0.923174778  11.279651362   9.874266212   7.62420130   6.871555064
## [54,] 10.823174778   9.879651362   8.074266212   6.92420130   4.871555064
## [55,] -6.516069477 -12.401507802 -15.783194238 -19.52951582 -21.386653305
## [56,] -8.316069477 -13.101507802 -17.783194238 -20.32951582 -22.486653305
## [57,] -0.676824922  -2.492822581  -2.974512930  -4.03061809   0.000000000
## [58,] -2.676824922  -3.292822581  -4.074512930   0.00000000   0.000000000
## [59,] -0.867566131  -1.623532528   0.000000000   0.00000000   0.000000000
## [60,] -1.967566131   0.000000000   0.000000000   0.00000000   0.000000000
##               [,6]        [,7]        [,8]          [,9]        [,10]
##  [1,]   0.55259983  0.16855453  0.58469507  0.4489819686  0.626315495
##  [2,]   0.35259983  0.76855453  0.58469507  0.3489819686  0.426315495
##  [3,]   0.20362786  0.05535514  0.22158312  0.4128626018  0.437208573
##  [4,]   0.20362786 -0.04464486  0.02158312  0.3128626018  0.137208573
##  [5,]   0.30961600  0.50391338  0.53071009  0.2241350719  0.361479467
##  [6,]   0.10961600  0.40391338  0.23071009  0.3241350719  0.961479467
##  [7,]   0.10512687 -0.21704158 -0.09340777 -0.0501462820 -0.088394078
##  [8,]  -0.19487313 -0.11704158  0.50659223 -0.2501462820 -0.388394078
##  [9,]  -0.07079669 -0.02993642 -0.07092918 -0.2530352001 -0.361774780
## [10,]   0.52920331 -0.22993642 -0.37092918 -0.7530352001 -0.361774780
## [11,]  -0.03512735 -0.21759915 -0.32694727 -0.3315886781 -0.312470948
## [12,]  -0.33512735 -0.71759915 -0.32694727  0.0684113219 -0.212470948
## [13,]  -0.57421956 -0.58519564 -0.57450112 -0.5335592463 -0.429732977
## [14,]  -0.57421956 -0.18519564 -0.47450112 -0.6335592463 -0.829732977
## [15,]  -0.25032195 -0.19953257 -0.08869393 -0.2715154120 -0.386189525
## [16,]  -0.15032195 -0.29953257 -0.48869393 -0.4715154120 -0.386189525
## [17,]  -0.10034659 -0.28346621 -0.39820514 -0.2595555087 -0.310776490
## [18,]  -0.50034659 -0.48346621 -0.39820514 -0.2595555087  0.289223510
## [19,]   0.02525466  0.17974901  0.14109230  0.2532543946  0.298111113
## [20,]   0.02525466  0.17974901  0.74109230  0.1532543946 -0.001888887
## [21,]   0.01568295  0.12318268  0.16426600  0.0530195204  0.062310962
## [22,]   0.61568295  0.02318268 -0.13573400 -0.3469804796 -0.037689038
## [23,]  -0.12124128 -0.24264772 -0.24230323 -0.2649607990 -0.111839807
## [24,]  -0.42124128 -0.64264772 -0.34230323  0.3350392010 -0.211839807
## [25,]   0.23684109  0.22875862  0.39962489  0.2591796608  0.155924848
## [26,]   0.13684109  0.82875862  0.29962489  0.0591796608 -0.144075152
## [27,]   0.20088418  0.05467768 -0.05384528 -0.0760328730 -0.036414389
## [28,]   0.10088418 -0.14532232 -0.35384528 -0.1760328730 -0.036414389
## [29,]  -0.28134873 -0.31523140 -0.28426105 -0.1601423911 -0.006677991
## [30,]  -0.58134873 -0.41523140 -0.28426105  0.0398576089  0.693322009
## [31,]  -0.08883879  0.04059141  0.19968294  0.3654656869  0.198525662
## [32,]  -0.08883879  0.24059141  0.89968294  0.0654656869 -0.001474338
## [33,]  -0.03678034  0.11890218 -0.05572509  0.0066960528 -0.198075663
## [34,]   0.66321966 -0.18109782 -0.25572509 -0.5933039472 -0.098075663
## [35,]   0.37466789  0.45182879  0.26009639  0.4136377162  0.368472553
## [36,]   0.17466789 -0.14817121  0.36009639  0.8136377162  0.568472553
## [37,]   0.19370344  0.34420992  0.29430105  0.5002445951  0.439585245
## [38,]   0.29370344  0.74420992  0.49430105  0.3002445951 -0.060414755
## [39,]  -0.16849455  0.02120230 -0.05376548 -0.2500311095 -0.125374014
## [40,]   0.03150545 -0.17879770 -0.55376548 -0.2500311095 -0.125374014
## [41,]   0.06694518 -0.12451361  0.00397101  0.0005716526  0.001109571
## [42,]  -0.43305482 -0.12451361  0.00397101  0.1005716526  0.601109571
## [43,]   0.24115837  0.24745028  0.25702858  0.2726115627  0.263055115
```

```
## [44,]   0.24115837  0.34745028  0.85702858  0.0726115627  0.963055115
## [45,]   0.09583214  0.10681596  0.09290197  1.0159697613 11.372721755
## [46,]   0.69583214 -0.09318404  0.79290197 10.9159697613  9.972721755
## [47,]  -0.10444309  0.81355128 11.16444639  9.7558005404  7.505024205
## [48,]   0.59555691 10.71355128  9.76444639  7.9558005404  6.805024205
## [49,]  11.25466879  9.84792796  7.59849693  6.8452251380  5.023824090
## [50,]   9.85466879  8.04792796  6.89849693  4.8452251380  4.223824090
## [51,]   7.60428599  6.85219789  5.03098805  4.5938292713  3.535954687
## [52,]   6.90428599  4.85219789  4.23098805  3.4938292713  0.000000000
## [53,]   5.05215293  4.61849469  3.56108343  0.0000000000  0.000000000
## [54,]   4.25215293  3.51849469  0.00000000  0.0000000000  0.000000000
## [55,] -23.43017492  0.00000000  0.00000000  0.0000000000  0.000000000
## [56,]   0.00000000  0.00000000  0.00000000  0.0000000000  0.000000000
## [57,]   0.00000000  0.00000000  0.00000000  0.0000000000  0.000000000
## [58,]   0.00000000  0.00000000  0.00000000  0.0000000000  0.000000000
## [59,]   0.00000000  0.00000000  0.00000000  0.0000000000  0.000000000
## [60,]   0.00000000  0.00000000  0.00000000  0.0000000000  0.000000000
##              [,11]        [,12]
##  [1,]  0.828784782  0.862412077
##  [2,]  0.728784782  0.562412077
##  [3,]  0.128581762  0.263911095
##  [4,]  0.228581762  0.863911095
##  [5,]  0.414577128  0.387447501
##  [6,]  0.214577128  0.087447501
##  [7,] -0.268077756 -0.374385939
##  [8,] -0.768077756 -0.374385939
##  [9,] -0.365836904 -0.347097755
## [10,]  0.034163096 -0.247097755
## [11,] -0.265591025 -0.154290279
## [12,] -0.365591025 -0.554290279
## [13,] -0.619391918 -0.740575622
## [14,] -0.819391918 -0.740575622
## [15,] -0.246745974 -0.298510682
## [16,] -0.246745974  0.301489318
## [17,] -0.211105492 -0.176458710
## [18,] -0.311105492 -0.476458710
## [19,]  0.190093384  0.202197787
## [20,] -0.209906616  0.102197787
## [21,]  0.047801108  0.208076333
## [22,]  0.647801108  0.108076333
## [23,] -0.264464872 -0.378758692
## [24,] -0.464464872 -0.678758692
## [25,]  0.138792144  0.182695754
## [26,]  0.038792144  0.182695754
## [27,]  0.087244176  0.247786523
## [28,]  0.287244176  0.947786523
## [29,]  0.154225648 -0.016578745
## [30,] -0.145774352 -0.216578745
## [31,]  0.267453843  0.068053310
## [32,] -0.332546157  0.168053310
## [33,] -0.055861991 -0.111267897
## [34,]  0.344138009  0.088732103
## [35,]  0.577729851  0.518523114
## [36,]  0.377729851  0.018523114
```

```
## [37,]  0.256016860  0.391907913
## [38,]  0.256016860  0.391907913
## [39,] -0.133344691 -0.133552391
## [40,] -0.033344691  0.466447609
## [41,]  0.008479281 -0.007060935
## [42,] -0.191520719  0.692939065
## [43,]  1.189487069 11.549610603
## [44,] 11.089487069 10.149610603
## [45,]  9.967451365  7.721793760
## [46,]  8.167451365  7.021793760
## [47,]  6.750013384  4.927642204
## [48,]  4.750013384  4.127642204
## [49,]  4.587130468  3.528787151
## [50,]  3.487130468  0.000000000
## [51,]  0.000000000  0.000000000
## [52,]  0.000000000  0.000000000
## [53,]  0.000000000  0.000000000
## [54,]  0.000000000  0.000000000
## [55,]  0.000000000  0.000000000
## [56,]  0.000000000  0.000000000
## [57,]  0.000000000  0.000000000
## [58,]  0.000000000  0.000000000
## [59,]  0.000000000  0.000000000
## [60,]  0.000000000  0.000000000
```

```
unem_w_error = data.frame(Unemw$error)
unem_w_error = unem_w_error%>% slice(1:49)
unem_w_error$actual = une_ts[490:538]

unem_w_error$abs = abs(unem_w_error$X12/unem_w_error$actual)
unem_w_error$mape = cummean(unem_w_error$abs)

plot(unem_w_error$mape, type = 'l', main = 'Window: MAPE for Unemployment at h = 12 for 49 iterations')
lines(unem_error$mape, type = 'l', col = 'green')
legend('topright',legend=c("windows", "recursive"),
       col=c("black", "green"), lty=1:2, cex=0.8)
```



Window backtesting for unemployment using h = 1 and corresponding MAPE plot.

```
Unem1w <- window_backtesting(diff3.7, une_ts, 490, h=1)
```

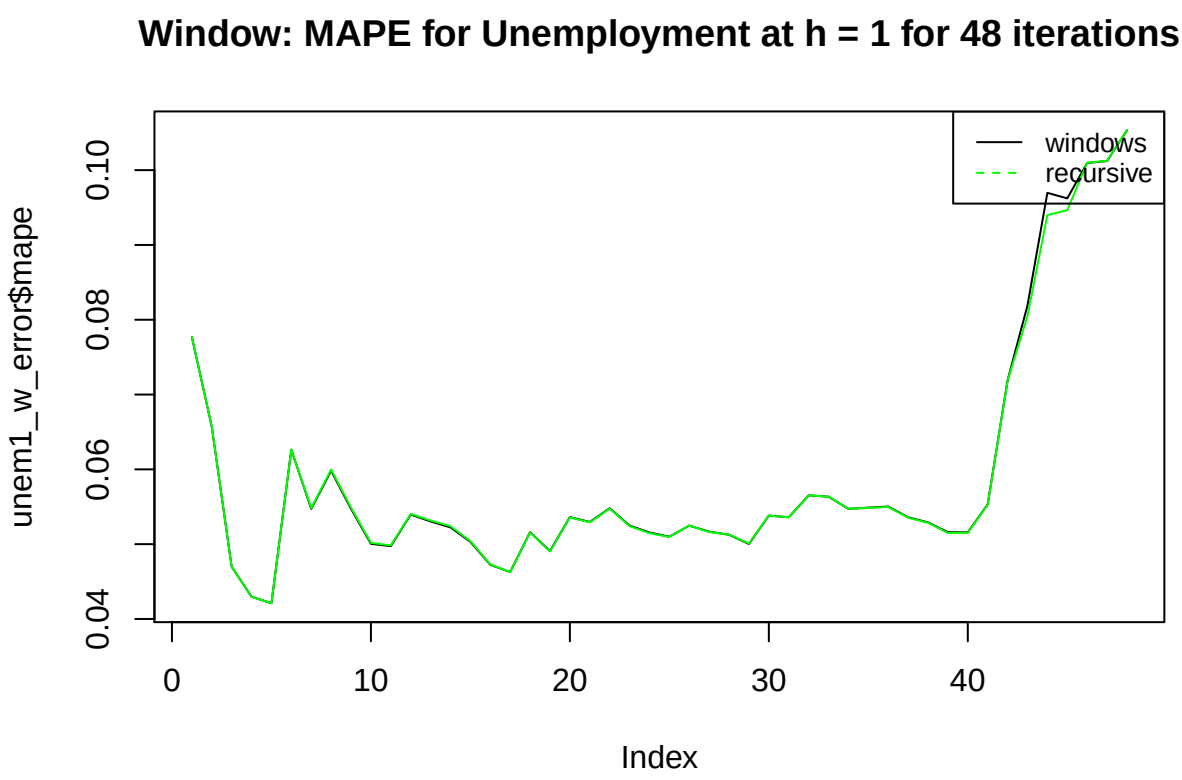
```
## [1] "RMSE of out-of-sample forecasts"
## [1] 2.26045
## [1] "Mean absolute error of out-of-sample forecasts"
## [1] 0.8518161
```

```
Unem1w$error
```

```
##               [,1]
##  [1,] -0.341899026
##  [2,] -0.241899026
##  [3,]  0.048481094
##  [4,] -0.151518906
##  [5,] -0.177673958
##  [6,] -0.677673958
##  [7,]  0.029672348
##  [8,]  0.429672348
##  [9,]  0.064563216
## [10,] -0.035436784
## [11,] -0.191416592
## [12,] -0.391416592
## [13,]  0.163662260
```

```
## [14,]  0.163662260
## [15,]  0.105076037
## [16,]  0.005076037
## [17,] -0.126116964
## [18,] -0.526116964
## [19,] -0.013267670
## [20,]  0.586732330
## [21,] -0.164117256
## [22,] -0.364117256
## [23,] -0.005716776
## [24,] -0.105716776
## [25,]  0.131254427
## [26,]  0.331254427
## [27,]  0.134755966
## [28,] -0.165244034
## [29,]  0.059142319
## [30,] -0.540857681
## [31,]  0.159572197
## [32,]  0.559572197
## [33,]  0.203571119
## [34,]  0.003571119
## [35,] -0.199165430
## [36,] -0.199165430
## [37,] -0.005921729
## [38,]  0.094078271
## [39,]  0.006689760
## [40,] -0.193310240
## [41,]  0.922202936
## [42,] 10.822202936
## [43,] -6.534755295
## [44,] -8.334755295
## [45,] -0.669158997
## [46,] -2.669158997
## [47,] -0.868555531
## [48,] -1.968555531
```

```
unem1_w_error = data.frame(Unem1w$error)
unem1_w_error$actual = une_ts[491:538]

unem1_w_error$abs = abs(unem1_w_error$Unem1w.error/unem1_w_error$actual)
unem1_w_error$mape = cummean(unem1_w_error$abs)

plot(unem1_w_error$mape, type = 'l', main = 'Window: MAPE for Unemployment at h = 1 for 48 iterations')
lines(unem1_error$mape, type = 'l', col = 'green')
legend('topright',legend=c("windows", "recursive"),
       col=c("black", "green"), lty=1:2, cex=0.8)
```



Part 14e. 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?

The errors between the two schemes are very similar as shown by our plots above, which means that there are not major irregularities in our data that are particular to any one time instance. The windows backtesting showed very very slightly higher errors, but since they are essentially the same we believe that our models chosen are robust.

0.11 Conclusions and Future Work

Using ARIMA, VAR(18), and both Recursive and Window backtesting methods, we get that our models are overall robust. However, using VAR(18) allows us to further comprehend the correlation between our two time series (Total Car Sales and Unemployment Rate). Nonetheless, this does not mean that one forecast method is better than the other, we should use and choose the one that has the smallest RSME, MAPE etc in order to assure that we are maximizing efficiency. In our case study for predicting car sales, our ARIMA model had lower MAPE 0.8% and our VAR model had a MAPE of 5.6%. Our ARIMA model is better in terms of forecasting accuracy but there is a possibility for overfitting the data as well. Overall, both model have a pretty good MAPE, as they are both under 10%. Our unemployment ARIMA model had a MAPE of 5%, which is also under 10% and good as well. We conclude that there is a correlation between Unemployment Rate and the Total Car Sales on a monthly basis. We confirm our hypothesis and are able to see the forecast and the negative relationship between the two time series is important.

For future works, we can look into how specifically covid-19 has affected the models since the government

are heavily subsidizing personal income. We can also look into consumption other than car sales. We can also look at personal income instead of unemployment to see if it differs from out current results.

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

The data we use come from the FRED website

https://fred.stlouisfed.org/series/TOTALNSA https://fred.stlouisfed.org/series/UNRATENSA

Other resources included class notes and online forums.