Statistical Methods for Discrete Response, Time Series, and Panel Data (W271): Lab 3

K.C. Tobin, Weixing Sun, Winston Lin

August 6, 2017

Question 1: EDA

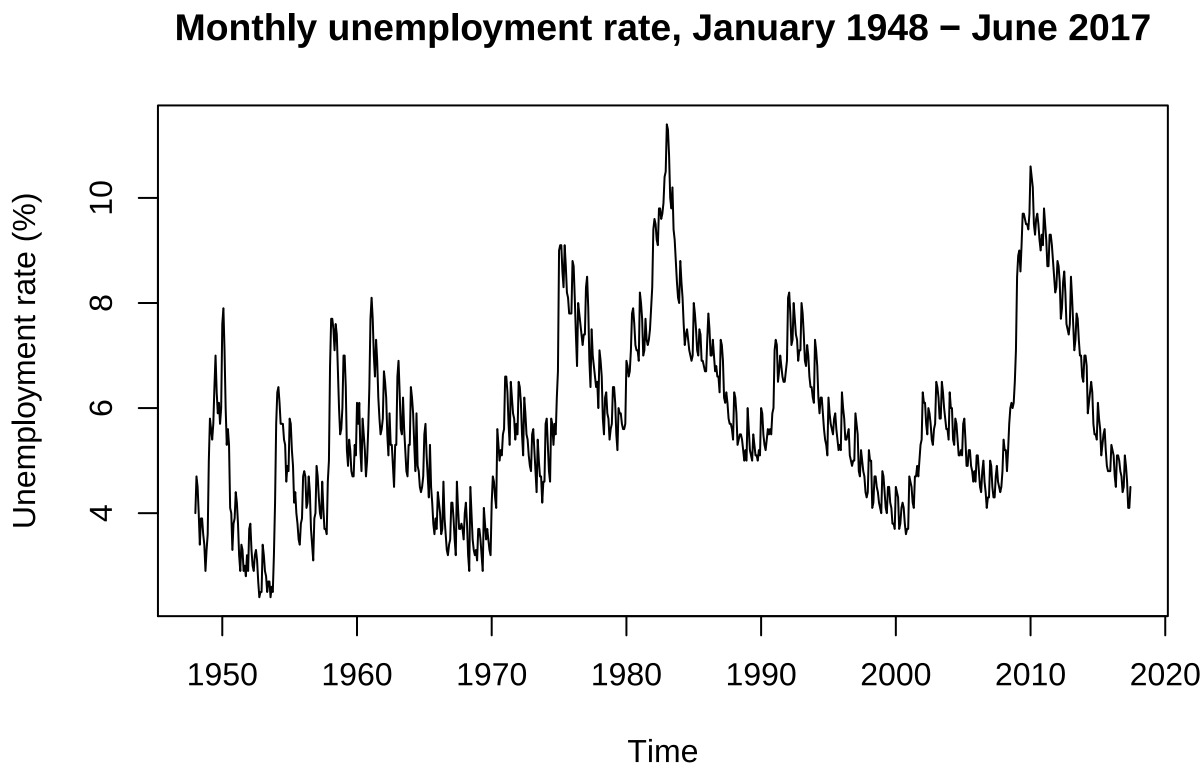
During your EDA, you notice that your data exhibits both seasonality (different months have different heights) AND that there is a clear linear trend. How many order of non-seasonal and seasonal differencing would it take to make this time-series stationary in the mean? Why?

Behavior of ACF and PACF for ARMA models (from SS2016, p.71): - AR(p): ACF tails off, PACF cuts off after lag p - MA(q): ACF cuts off after lag q, PACF tails off - ARMA(p,q): both ACF and PACF tail off

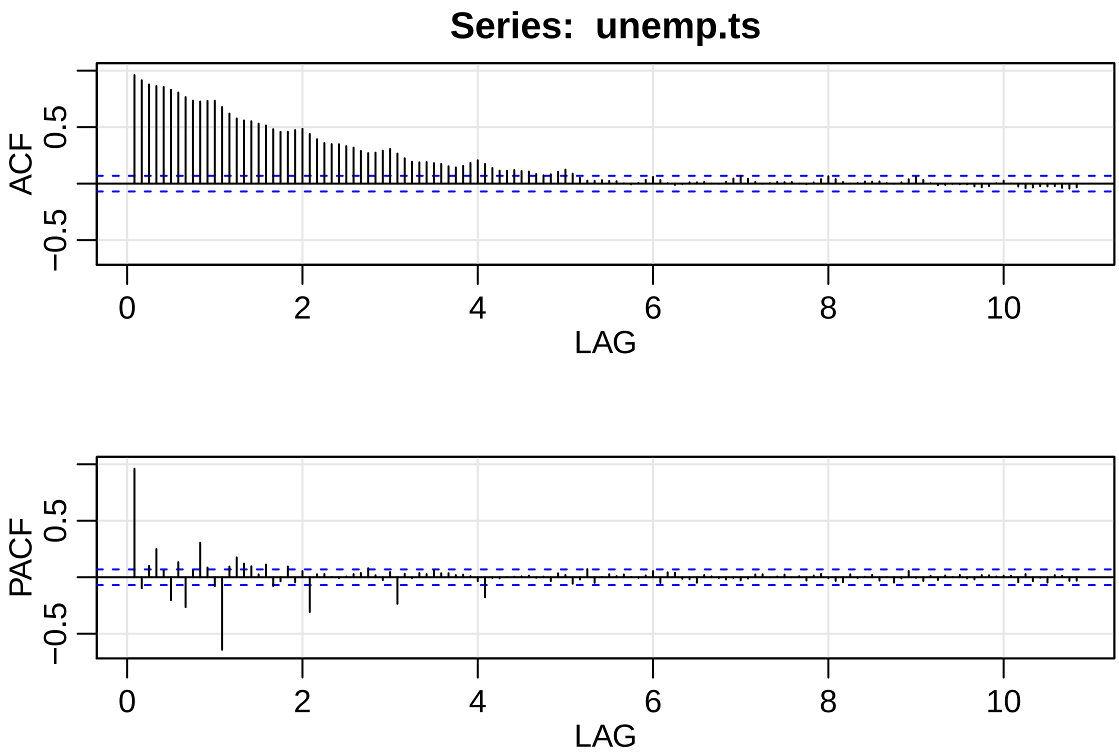
```
### 1. EDA for unemployment rate
unemp = read.csv("UNRATENSA.csv")
head(unemp); tail(unemp)
```

```
##         DATE UNRATENSA
## 1 1948-01-01       4.0
## 2 1948-02-01       4.7
## 3 1948-03-01       4.5
## 4 1948-04-01       4.0
## 5 1948-05-01       3.4
## 6 1948-06-01       3.9

##           DATE UNRATENSA
## 829 2017-01-01       5.1
## 830 2017-02-01       4.9
## 831 2017-03-01       4.6
## 832 2017-04-01       4.1
## 833 2017-05-01       4.1
## 834 2017-06-01       4.5
```

```
unemp.ts = ts(unemp$UNRATENSA, start = c(1948, 1), frequency=12)

# plot series
plot.ts(unemp.ts, main = 'Monthly unemployment rate, January 1948 - June 2017', ylab = 'Unemployment ra
```



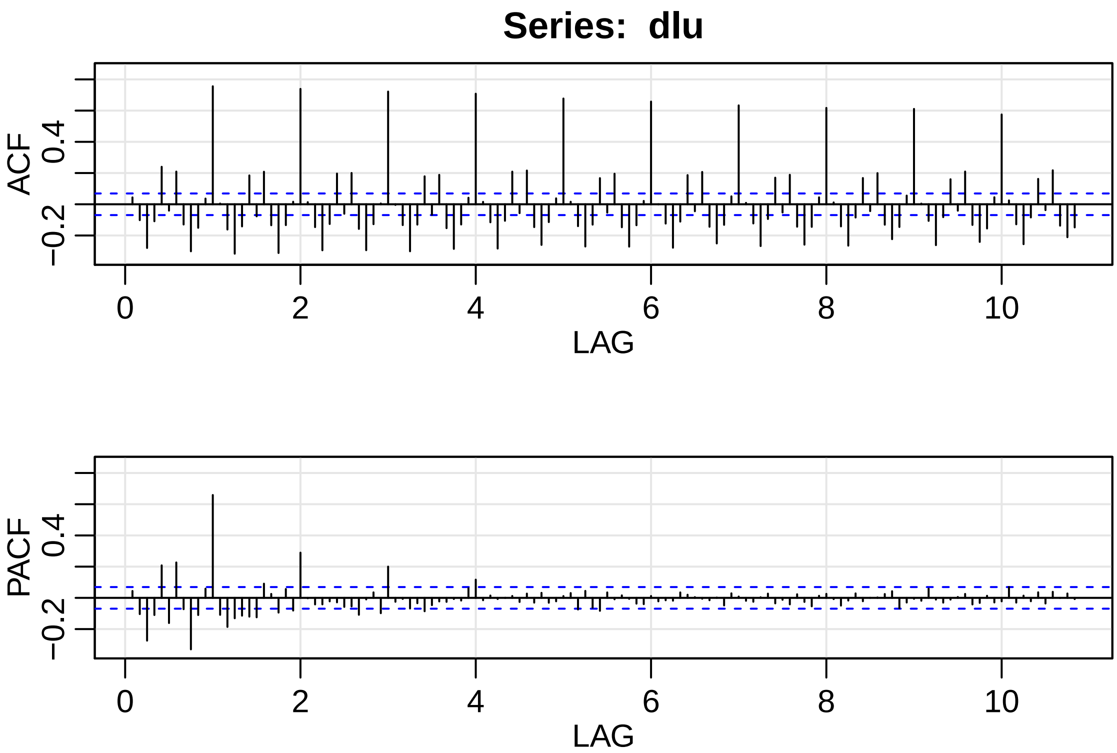

```
invisible(acf2(unemp.ts, 130))   # ACF: trend, PACF: seasonality
```




```r
# test null hypothesis of non-stationarity (unit root)
adf.test(unemp.ts)  # p = 0.33 > 0.05, series is non-stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  unemp.ts
## Dickey-Fuller = -2.5911, Lag order = 9, p-value = 0.3281
## alternative hypothesis: stationary
```

```r
### 2. transform data to stationary series

# log transform to stabilize variance
lu = log(unemp.ts)

# diff by 1 lag to remove monthly trend
dlu = diff(lu)
invisible(acf2(dlu, 130))
```

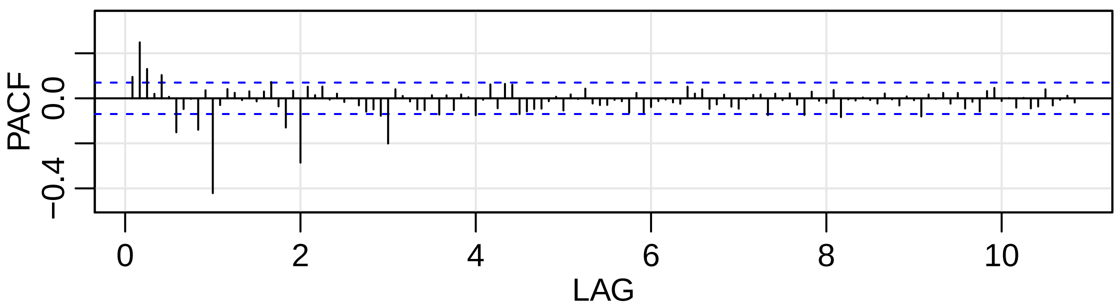



```r
# ACF: seasonal trend tails off --> seasonal MA
# PACF: cutoff after 3 seasonal lags --> seasonal AR, max order 3

# test null hypothesis of non-stationarity (unit root)
adf.test(dlu)
```

```
## Warning in adf.test(dlu): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  dlu
## Dickey-Fuller = -13.715, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

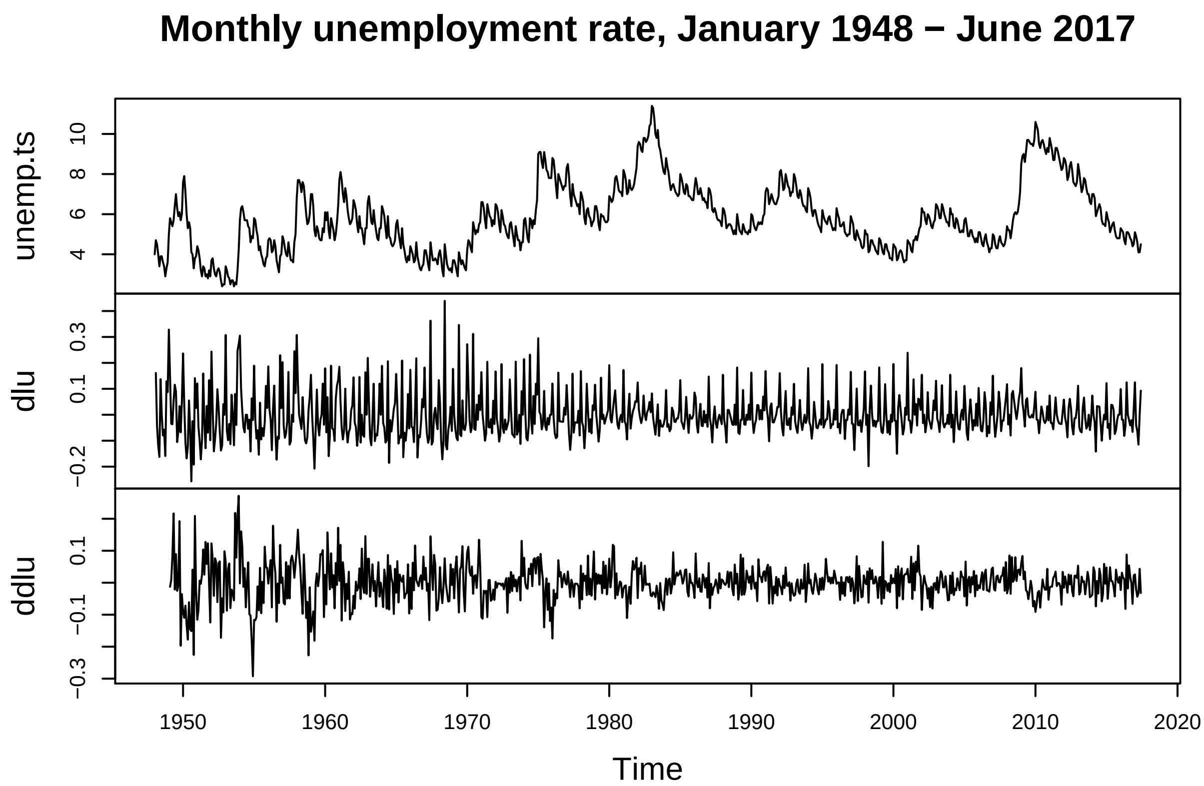
```r
# p = 0.01 < 0.05 --> series is stationary, but seasonal MA from PACF

# diff by 12 lags to remove seasonality
ddlu = diff(dlu, lag=12)
invisible(acf2(ddlu, 130))
```




```r
# ACF: cutoff after 1 seasonal lag --> seasonal MA, max order 1
# ACF: cutoff after 3 lags --> MA, max order 3
# PACF: cutoff after 3 seasonal lags --> seasonal

# test null hypothesis of non-stationarity (unit root)
adf.test(ddlu)
```

```
## Warning in adf.test(ddlu): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  ddlu
## Dickey-Fuller = -9.7378, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
# p =0.01 < 0.05, series is stationary

# plot transformed data
plot.ts(cbind(unemp.ts,dlu,ddlu), main='Monthly unemployment rate, January 1948 - June 2017')
```




```r
# start w/ d=1, D=1 from ADF tests for stationarity, max of 3 seasonal and 3 non-seasonal lags from ACF
```

The variance of the series appears to decrease over time, so we take the log of the series to stabilize the variance. The ACF decays slowly, so we take the difference with lag 1 (1 month) to remove the trend. The resulting ACF shows seasonal autocorrelation every 12 months that decays gradually. Now, we take the difference with lag 12 (1 year) to remove the seasonal trend. The ACF now cuts off after lag 12, and the PACF decays slowly. This hints that a seasonal MA(1) model might be a good place to start fitting. So to remove trend and seasonality, we would perform non-seasonal differencing of order 1 and seasonal differencing of order 1, with a 12-month seasonal period.

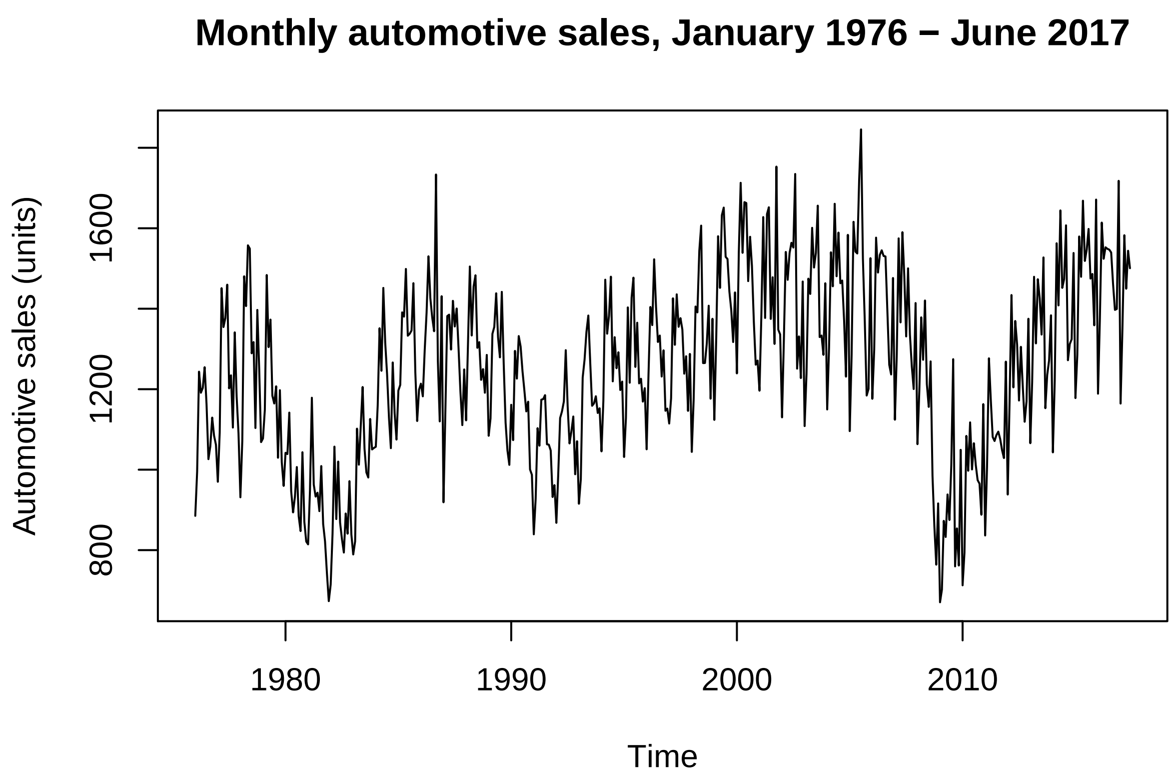
```r
### 1. EDA for automotive sales
auto = read.csv("TOTALNSA.csv")
head(auto); tail(auto)
```

```
##         DATE TOTALNSA
## 1 1976-01-01    885.2
## 2 1976-02-01    994.7
## 3 1976-03-01   1243.6
## 4 1976-04-01   1191.2
## 5 1976-05-01   1203.2
## 6 1976-06-01   1254.7
```

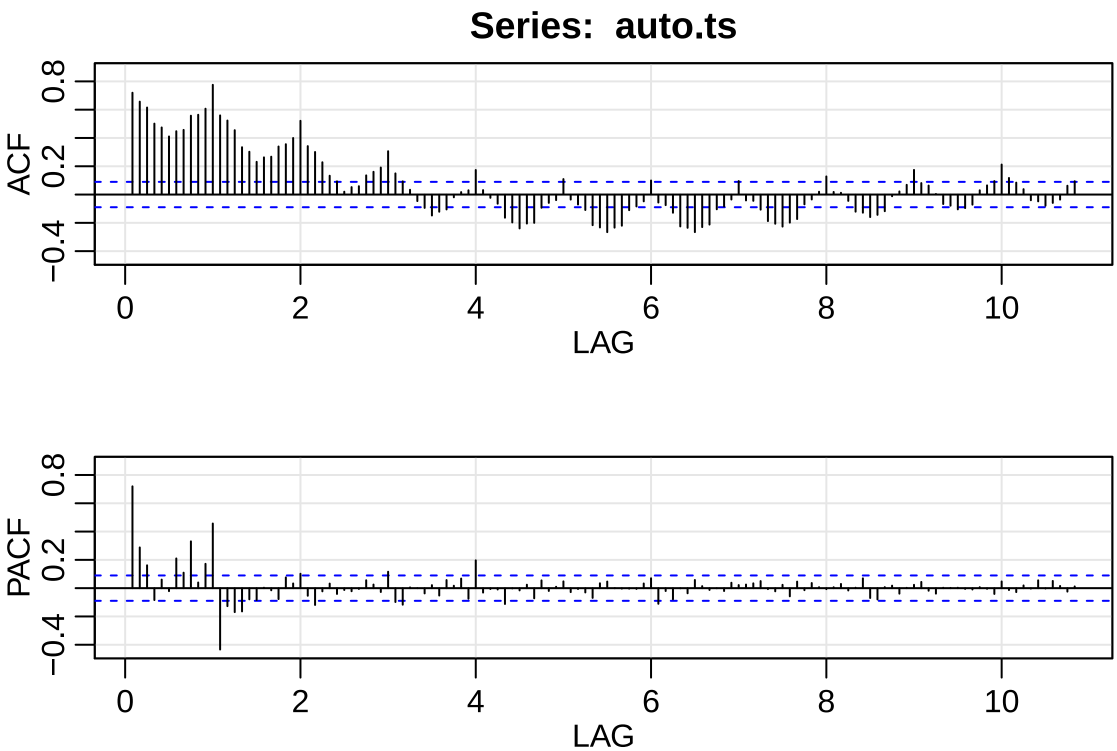
```
##           DATE TOTALNSA
## 493 2017-01-01   1164.3
## 494 2017-02-01   1352.1
## 495 2017-03-01   1582.7
## 496 2017-04-01   1449.7
## 497 2017-05-01   1544.1
```

```
## 498 2017-06-01   1500.6
```

```
auto.ts = ts(auto$TOTALNSA, start = c(1976, 1), frequency=12)

# plot raw data
plot.ts(auto.ts, main = 'Monthly automotive sales, January 1976 - June 2017', ylab = 'Automotive sales
```



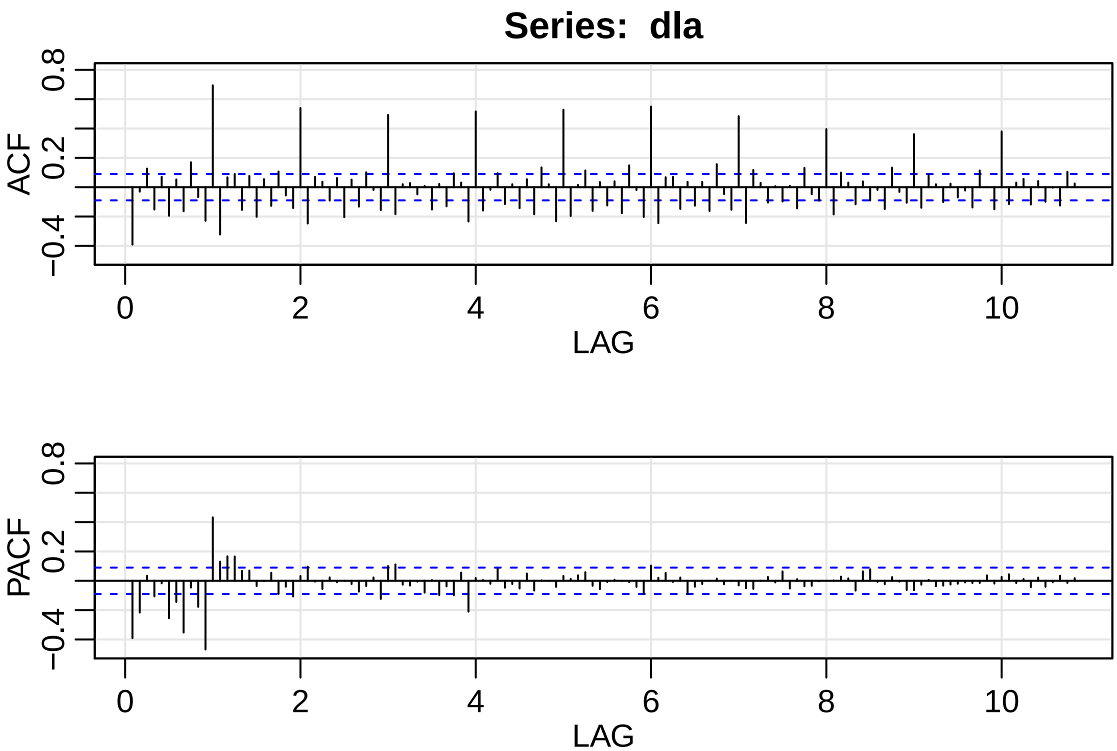

```
invisible(acf2(auto.ts, 130))  # ACF: trend and seasonality
```




```r
# test null hypothesis of non-stationarity (unit root)
adf.test(auto.ts)   # p = 0.04 < 0.05, series is stationary, no transform needed
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  auto.ts
## Dickey-Fuller = -3.5662, Lag order = 7, p-value = 0.03595
## alternative hypothesis: stationary
```

```r
## 2. transform data to stationary series

# diff by 1 lag to remove monthly trend
dla = diff(auto.ts)
invisible(acf2(dla, 130))
```

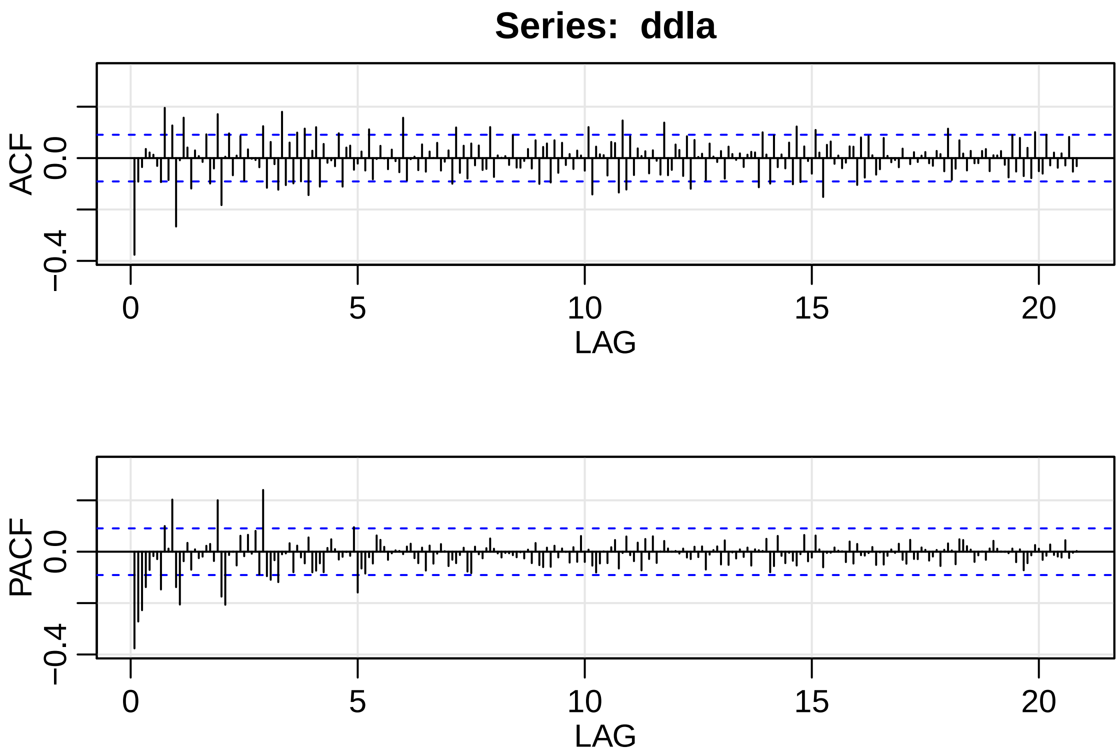



```r
# test null hypothesis of non-stationarity (unit root)
adf.test(dla)   # p = 0.01 < 0.05, series is stationary
```

```
## Warning in adf.test(dla): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  dla
## Dickey-Fuller = -16.651, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
# diff by 12 lags to remove seasonal trend
ddla = diff(dla, lag=12)
invisible(acf2(ddla, 250))
```

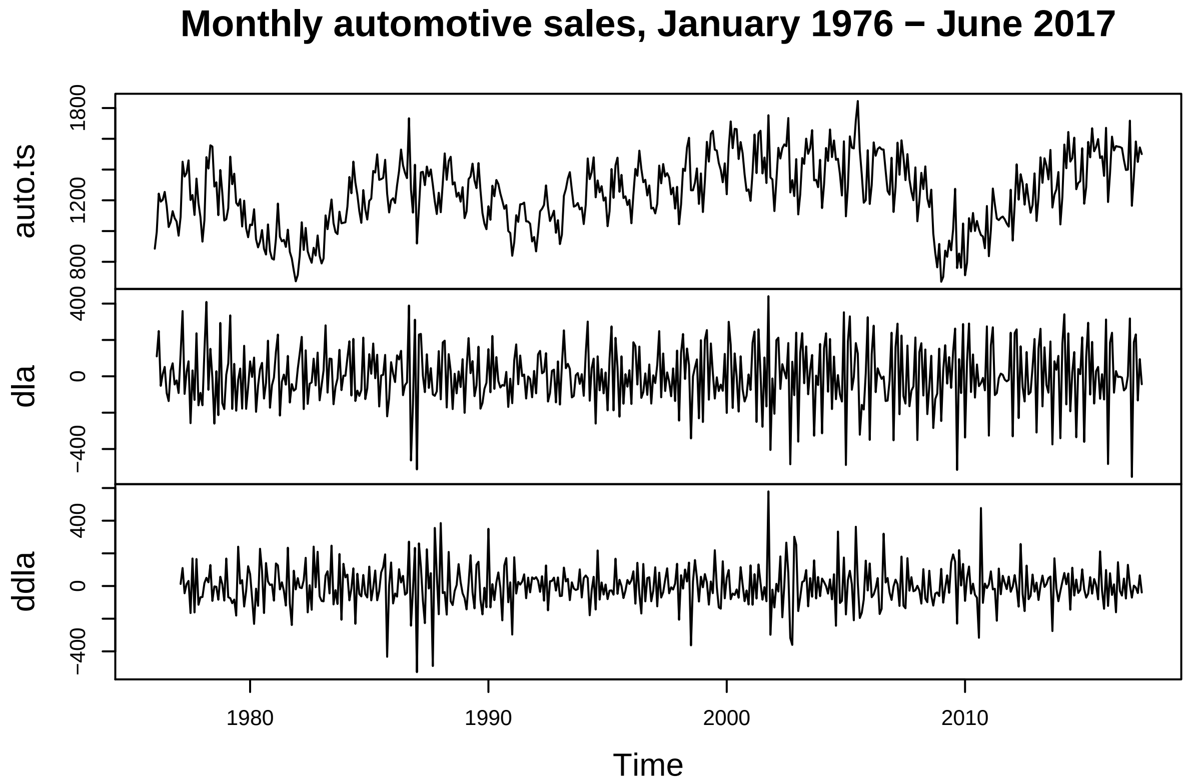



```r
# test null hypothesis of non-stationarity (unit root)
adf.test(ddla)  # p = 0.01 < 0.05, series is stationary
```

```
## Warning in adf.test(ddla): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ddla
## Dickey-Fuller = -11.281, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
# plot transformed data
plot.ts(cbind(auto.ts,dla,ddla), main='Monthly automotive sales, January 1976 - June 2017')
```



The variance of the series appears to relatively constant so no need to apply a transformation. The ACF decays slowly, so we take the difference with lag 1 (1 month) to remove the trend. The resulting ACF shows seasonal autocorrelation every 12 months that decays gradually. **Add analysis here**

Question 2: SARIMA

It is Dec. 31, 2016, and you work for a non-partisan think tank focusing on the state of the U.S. economy. You are interested in forecasting the unemployment rate through 2017 (and then 2020) to use it as a benchmark against the incoming administration's economic performance. Use the dataset UNRATENSA.csv and answer the following:

a. Build a SARIMA model using the unemployment data and produce a 1-year forecast and then a 4-year forecast. Because it is Dec. 31, 2016, leave out 2016 as your test data.

Remember, here are the steps to building an ARIMA model! 1. Conduct and EDA to determine if you need to transform the data in order to make it stationary. 2. Transform the data if needed. 3. Estimate several Arima(p,d,q) models. Remember, you set the value of d in the first step! So really, you are trying to find the apprioriate values of p and q. 4. Evaluate the residuals of models with the lowest AIC/BIC values and simpler models. Select the model where the residuals resemble white nose. 5. If you still have some candidate models remaining, then conduct an out of sample test and select the model with the lowest forecasting error. 6. Answer your question / generate forecasts!

```
### 3.a. fit seasonal models and select final candidate models with lowest AIC/BIC

# split data into training and test sets
unemp.train = window(unemp.ts, end=c(2015,12), frequency=12)
unemp.test = window(unemp.ts, start=c(2016,1), frequency=12)
```

```r
# try models w/ max 3 seasonal AR, MA lags based on ACF, PACF
# start w/ D=1 from EDA and transforms
for (P in 0:3) {
        for(Q in 0:3) {
                fit = Arima(unemp.train, order=c(0,1,0), seasonal=list(order=c(P,1,Q)), method='ML')
                print(c(P,Q,fit$aic))
        }
}
```

```
## [1]   0.0000   0.0000 427.2262
## [1]  0.00000  1.00000 47.12622
## [1]  0.00000  2.00000 49.10322
## [1]  0.00000  3.00000 51.10067
## [1]   1.0   0.0 210.4
## [1]  1.00000  1.00000 49.10312
## [1]  1.00000  2.00000 51.12536
## [1]  1.00000  3.00000 52.14965
## [1]   2.0000   0.0000 138.9131
## [1]  2.00000  1.00000 51.10083
## [1]  2.00000  2.00000 52.21085
## [1]  2.00000  3.00000 42.59912
## [1]  3.0000  0.0000 85.7188
## [1]  3.00000  1.00000 52.90777
## [1]  3.00000  2.00000 53.35645
## [1]  3.00000  3.00000 44.58169
```

```r
# candidate seasonal models (P,D,Q):
# (0,1,1) AIC = 47.1 --> low AIC, largest decrease in AIC, most parsimonious model
# (2,1,3) AIC = 42.6 --> lowest AIC, not much improvement in AIC, more complex model
```

Based on the AIC outputs it appears that a seasonal order of (0,1,1) produces relatively good results and makes for a parsimonious choice to move forward with.

```r
### 3.b. fit non-seasonal models (given seasonal models) and select final candidate models with lowest A

# try models w/ max of 3 non-seasonal AR, MA lags based on ACF, PACF
for (p in 0:3) {
        for(q in 0:3) {
                try(fit <- Arima(unemp.train, order=c(p,1,q), seasonal=list(order=c(0,1,1)), method='ML
                print(c(p,q,fit$aic))
        }
}
```

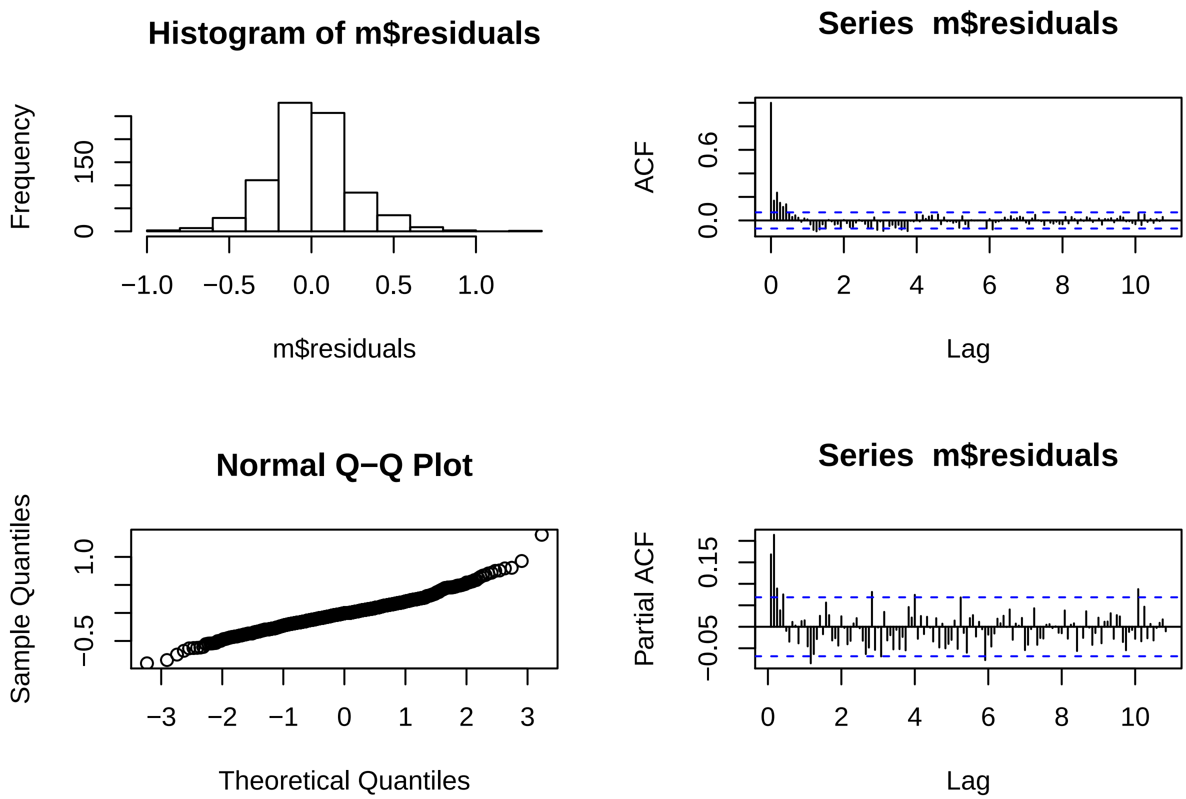
```
## [1]  0.00000  0.00000 47.12622
## [1]  0.00000  1.00000 32.70797
## [1] 0.000000 2.000000 1.640301
## [1]  0.000000  3.000000 -2.766081
## [1]  1.00000  0.00000 25.49529
## [1]   1.00000   1.00000 -12.69638
## [1]   1.00000   2.00000 -18.01948
## [1]   1.00000   3.00000 -16.24685
## [1]   2.00000   0.00000 -10.45144
## [1]   2.00000   1.00000 -18.34985
## [1]   2.00000   2.00000 -16.39718
## [1]   2.00000   3.00000 -14.39707
```

```
## [1]   3.00000   0.00000 -15.31482
## [1]   3.00000   1.00000 -16.39161
## [1]   3.0000   2.0000 -14.3607
## [1]   3.00000   3.00000 -15.81192
```

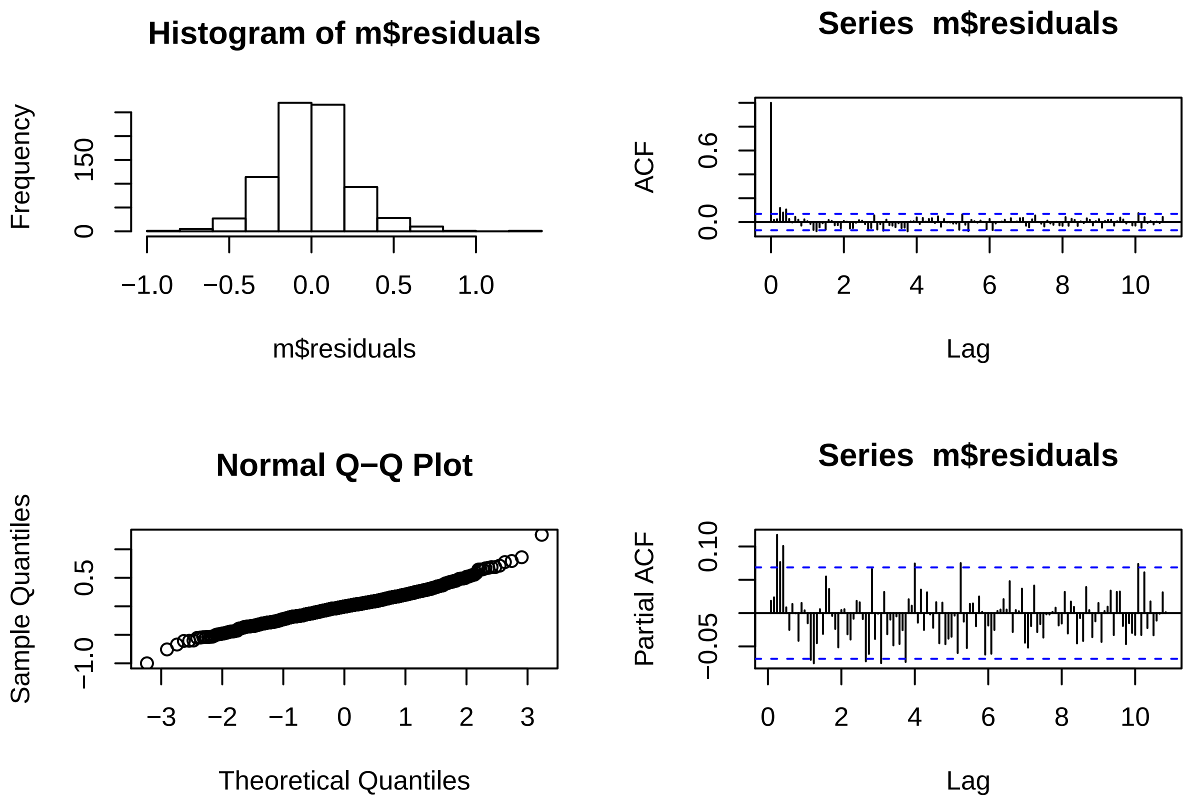
```r
# candidate non-seasonal models (p,d,q):
# (0,1,0) AIC = 47.1 --> baseline model
# (0,1,2) AIC = 1.6 --> low AIC, largest decrease in AIC, most parsimonious model
# (1,1,1) AIC = -12.7  --> low AIC, large decrease in AIC
# (1,1,2) AIC = -18.0  --> 2nd lowest AIC, slight improvement
# (2,1,1) AIC = -18.3  --> lowest AIC, slight improvement
# (2,1,2) AIC = -16.4
```

Several AR/MA terms are close in nature so we will continue to evaluate them all below.

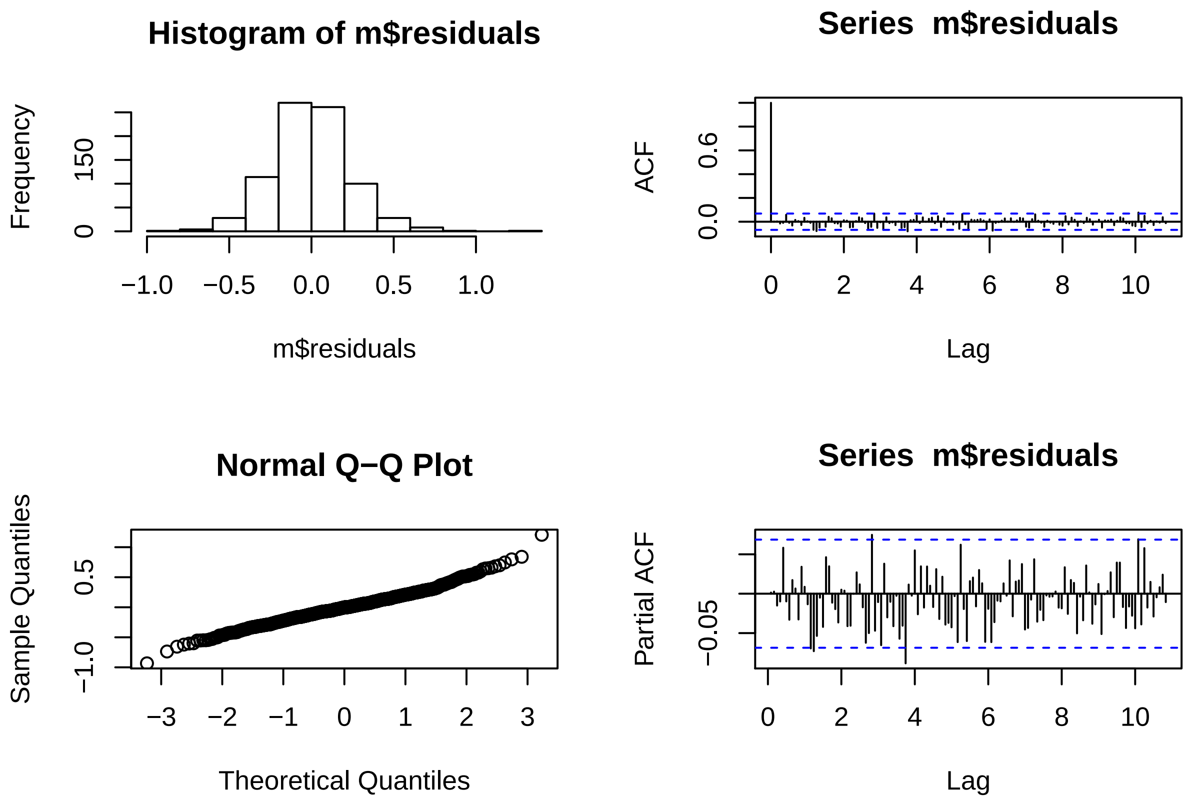
```r
### 4. check candidate models for residuals ~ white noise
sarima_diag = function(train, p,d,q, P,D,Q) {
        m = Arima(train, order=c(p,d,q), seasonal=list(order=c(P,D,Q)), method='ML')

        # residual diagnostics
        par(mfcol=c(2,2))
        hist(m$residuals)
        qqnorm(m$residuals)
        acf(m$residuals, 130)
        pacf(m$residuals, 130)
}

# diagnostics for candidate models
sarima_diag(unemp.train, 0,1,0, 0,1,1)  # residuals are serially correlated, normally distributed
```



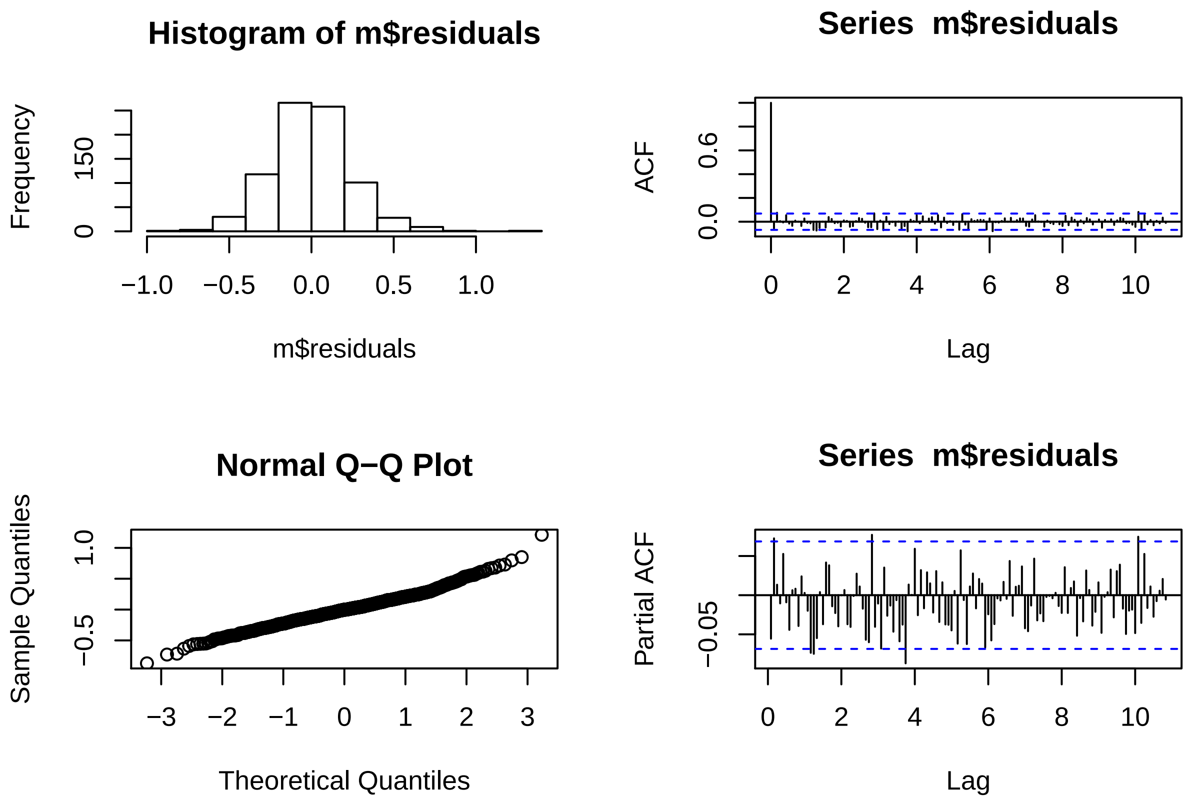

```
sarima_diag(unemp.train, 0,1,2, 0,1,1)  # residuals are serially uncorrelated, normally distributed
```



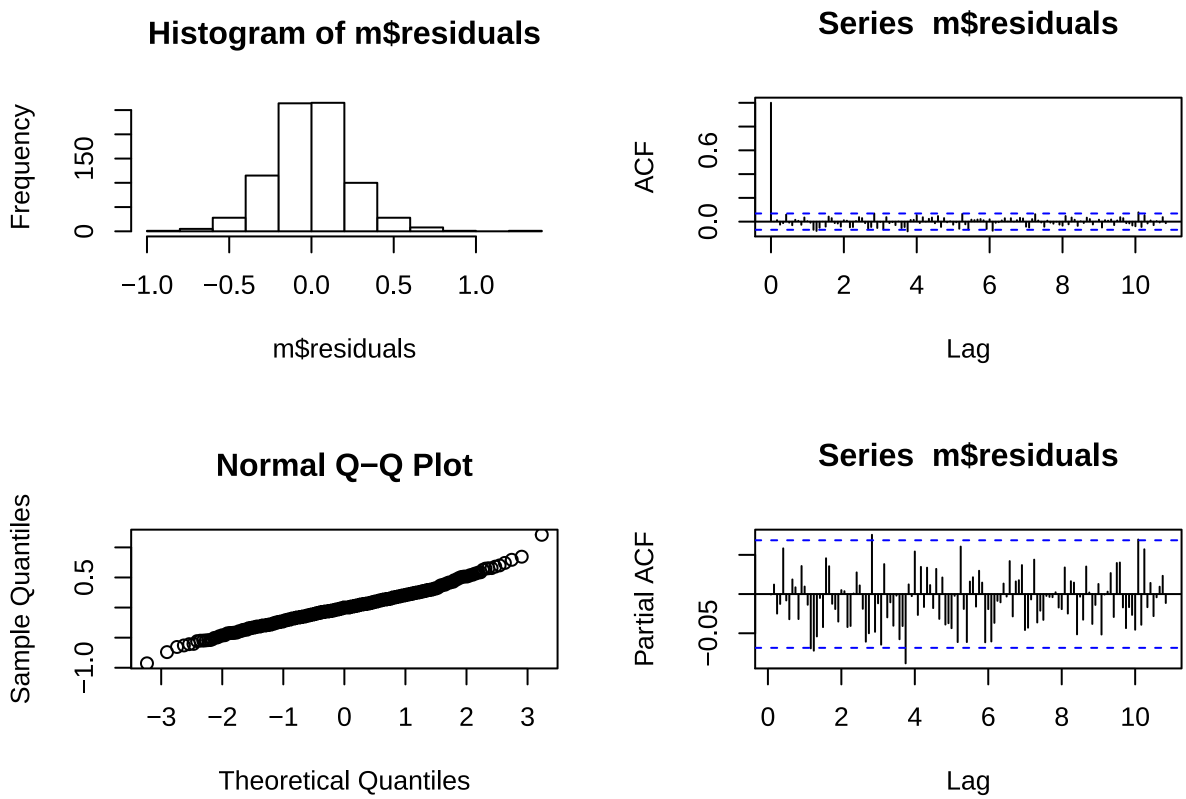

```r
sarima_diag(unemp.train, 2,1,1, 0,1,1)  # residuals are serially uncorrelated, normally distributed
```



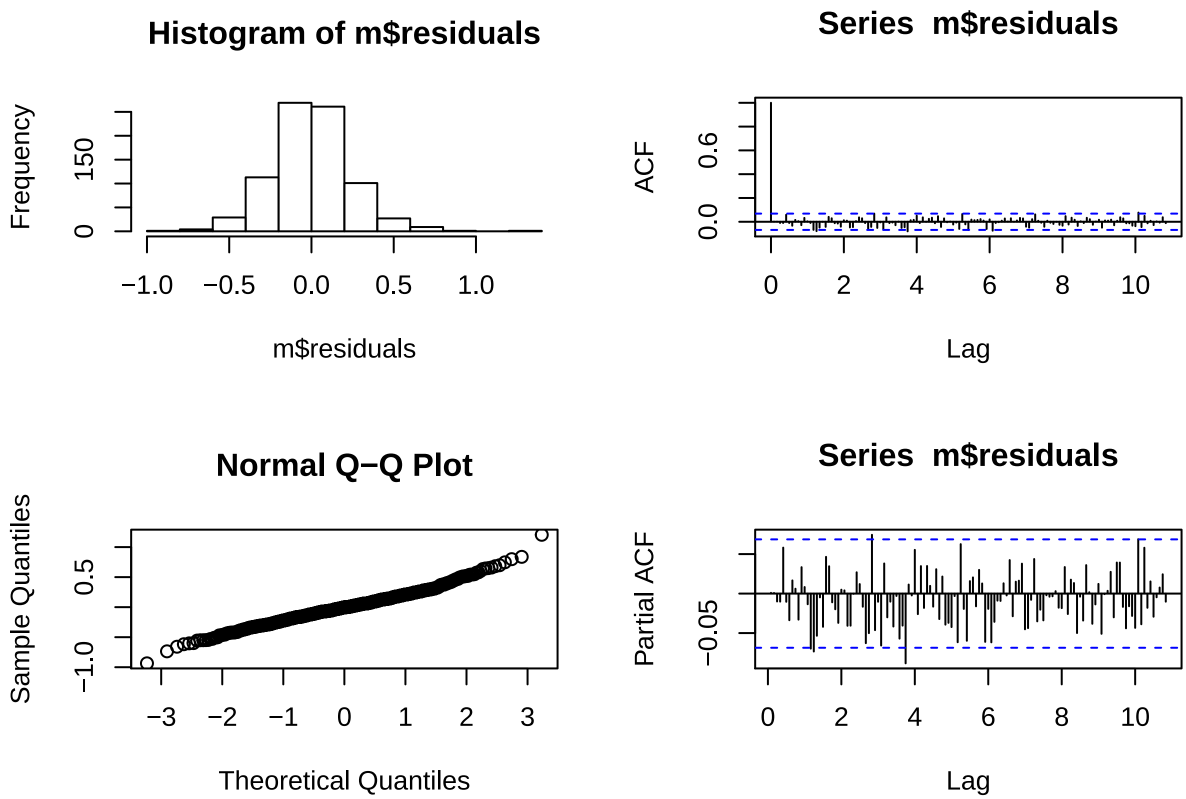

```
sarima_diag(unemp.train, 1,1,1, 0,1,1)  # residuals are serially uncorrelated, normally distributed
```




```
sarima_diag(unemp.train, 1,1,2, 0,1,1)  # residuals are serially uncorrelated, normally distributed
```




```
sarima_diag(unemp.train, 2,1,2, 0,1,1)  # residuals are serially uncorrelated, normally distributed
```



All models appear to produce residuals that resemble white noise and lack correlation with the exception of that (0,1,0) model. We will now evaluate the various models in terms of their forecasting power.

```r
### 5. check candidate models for out of sample accuracy

# candidate models
m0 = Arima(unemp.train, order=c(0,1,0), seasonal=list(order=c(0,1,1)), method='ML')
m1 = Arima(unemp.train, order=c(0,1,2), seasonal=list(order=c(0,1,1)), method='ML')
m2 = Arima(unemp.train, order=c(1,1,1), seasonal=list(order=c(0,1,1)), method='ML')
m3 = Arima(unemp.train, order=c(1,1,2), seasonal=list(order=c(0,1,1)), method='ML')
m4 = Arima(unemp.train, order=c(2,1,1), seasonal=list(order=c(0,1,1)), method='ML')
m5 = Arima(unemp.train, order=c(2,1,2), seasonal=list(order=c(0,1,1)), method='ML')

# forecasts for test period
for0 = forecast(m0, h=length(unemp.test))
for1 = forecast(m1, h=length(unemp.test))
for2 = forecast(m2, h=length(unemp.test))
for3 = forecast(m3, h=length(unemp.test))
for4 = forecast(m4, h=length(unemp.test))
for5 = forecast(m5, h=length(unemp.test))

# check accuracy against test set
accuracy(for0, unemp.test)  # RMSE = 0.152
```

```
##                        ME      RMSE       MAE         MPE     MAPE
## Training set -0.006361257 0.2447366 0.1825672 -0.08723024 3.367421
## Test set     -0.018362544 0.1518747 0.1272645 -0.40584864 2.699796
```

```
##                    MASE      ACF1 Theil's U
## Training set 0.2063602 0.1685593        NA
## Test set     0.1438503 0.4257728 0.4583755
```

```r
accuracy(for1, unemp.test)  # RMSE = 0.153
```

```
##                        ME      RMSE       MAE         MPE     MAPE
## Training set -0.004965367 0.2374515 0.1793519 -0.05758716 3.329680
## Test set      0.026112182 0.1530490 0.1196016  0.55779723 2.523776
##                    MASE       ACF1 Theil's U
## Training set 0.2027259 0.01851124        NA
## Test set     0.1351887 0.39501668  0.454131
```

```r
accuracy(for2, unemp.test)  # RMSE = 0.155
```

```
##                        ME      RMSE       MAE          MPE     MAPE
## Training set -0.003388788 0.2354166 0.1791475 -0.004004458 3.334639
## Test set      0.034894521 0.1547726 0.1217408  0.750543864 2.567508
##                    MASE        ACF1 Theil's U
## Training set 0.2024949 -0.05554858        NA
## Test set     0.1376066  0.38914276 0.4598686
```

```r
accuracy(for3, unemp.test)  # RMSE = 0.151 --> best fit, use SARIMA(1,1,2)(0,1,1) for final forecast
```

```
##                        ME      RMSE       MAE         MPE     MAPE
## Training set -0.003534511 0.2343297 0.1778710 -0.01030263 3.320626
## Test set      0.015050078 0.1506237 0.1193972  0.32696023 2.518034
##                    MASE          ACF1 Theil's U
## Training set 0.2010520 -0.0004805123        NA
## Test set     0.1349576  0.3868569701 0.4453878
```

```r
accuracy(for4, unemp.test)  # RMSE = 0.152
```

```
##                        ME      RMSE       MAE         MPE     MAPE
## Training set -0.003512207 0.2342802 0.1777658 -0.00941441 3.318480
## Test set      0.017207310 0.1510337 0.1194946  0.37283282 2.520191
##                    MASE        ACF1 Theil's U
## Training set 0.2009331 0.001441395        NA
## Test set     0.1350677 0.387328131  0.446624
```

```r
accuracy(for5, unemp.test)  # RMSE = 0.151
```

```
##                        ME      RMSE       MAE          MPE     MAPE
## Training set -0.003499461 0.2342739 0.1777561 -0.008939808 3.317892
## Test set      0.018172236 0.1512012 0.1195267  0.393400238 2.520930
##                    MASE         ACF1 Theil's U
## Training set 0.2009221 0.0009387735        NA
## Test set     0.1351040 0.3873410172 0.4471562
```

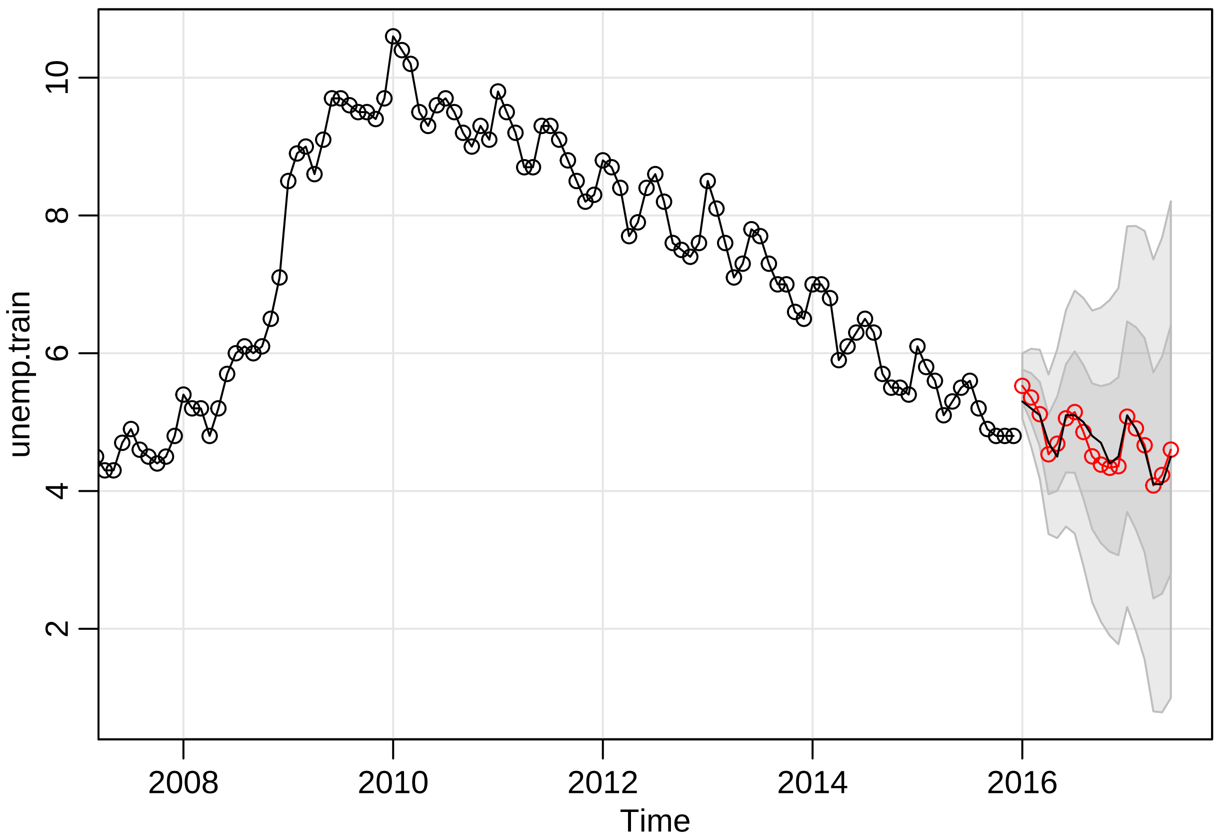
Based on forecasting power Sarima (1,1,2)(0,1,1) performs slightly better and will be our choice for modeling unemployment rates.

- How well does your model predict the unemployment rate up until June 2017?

The SARIMA(1,1,2) (0,1,1) model predicts unemployment rate well, with a RMSE of 0.151 on the test data set. Forecast seems to reflect the trend and seasonality and does not deviate significantly from the actual values in the forecast horizon.

```r
### 6.a. forecast through June 2017
```

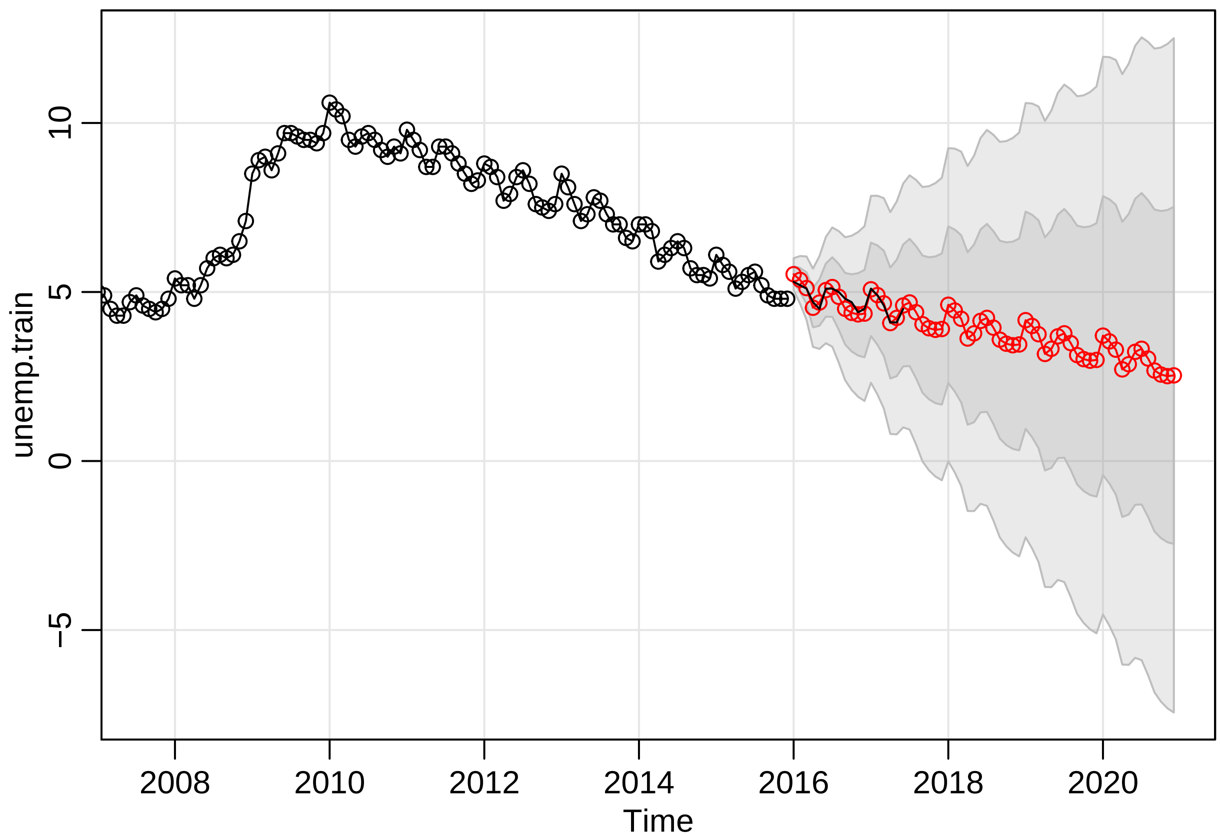
```
# plot test set and forecasts for models 1-3
# plot(unemp.test)
# lines(for1$mean, col = "red")
# lines(for2$mean, col = "blue")
# lines(for3$mean, col = "green")
# legend(1000, 10, c('Actual', 'SARIMA(0,1,2)(0,1,1)', 'SARIMA(0,1,2)(0,1,1)', 'SARIMA(0,1,2)(0,1,1)'),

# final forecast with model 3: SARIMA(1,1,2) (0,1,1) and CIs
for2y = sarima.for(unemp.train, length(unemp.test), 1,1,2, 0,1,1, 12)
lines(unemp.test, col='black')
```



- What does the unemployment rate look like at the end of 2020? How credible is this estimate?

The model predicts an unemployment rate of 2.53% at the end of 2020. The point estimate appears credible based on the current downward trend but the 95% confidence interval widens and contains 0 and negative values in December 2020.

```
### 6.a. forecast through December 2020

# final forecast with model 3: SARIMA(1,1,2) (0,1,1) and CIs
for5y = sarima.for(unemp.train, 60, 1,1,2, 0,1,1, 12)
lines(unemp.test, col='black')
```




```r
tail(for5y)
```

```
## $pred
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2016 5.526246 5.357901 5.115507 4.533402 4.686756 5.055630 5.146009
## 2017 5.079234 4.908464 4.664247 4.080770 4.233092 4.601191 4.690987
## 2018 4.622763 4.451914 4.207637 3.624115 3.776404 4.144477 4.234254
## 2019 4.165983 3.995132 3.750852 3.167329 3.319617 3.687689 3.777465
## 2020 3.709193 3.538341 3.294062 2.710539 2.862826 3.230898 3.320675
##           Aug      Sep      Oct      Nov      Dec
## 2016 4.856990 4.502899 4.383385 4.337381 4.360586
## 2017 4.401529 4.047109 3.927347 3.881156 3.904221
## 2018 3.944782 3.590351 3.470581 3.424384 3.447444
## 2019 3.487993 3.133561 3.013791 2.967594 2.990654
## 2020 3.031202 2.676771 2.557000 2.510803 2.533863
##
## $se
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2016 0.2362185 0.3531598 0.4680527 0.5789404 0.6849151 0.7856838 0.8812959
## 2017 1.3821818 1.4693835 1.5555925 1.6401072 1.7225385 1.8027004 1.8805359
## 2018 2.3161690 2.3945203 2.4731011 2.5512692 2.6285951 2.7048032 2.7797268
## 2019 3.2118738 3.2898950 3.3684326 3.4469108 3.5249268 3.6022056 3.6785662
## 2020 4.1243982 4.2045009 4.2851985 4.3659668 4.4464315 4.5263308 4.6054863
##            Aug       Sep       Oct       Nov       Dec
## 2016 0.9719833 1.0580661 1.1398982 1.2178360 1.2922212
## 2017 1.9560681 2.0293674 2.1005307 2.1696680 2.2368937
```

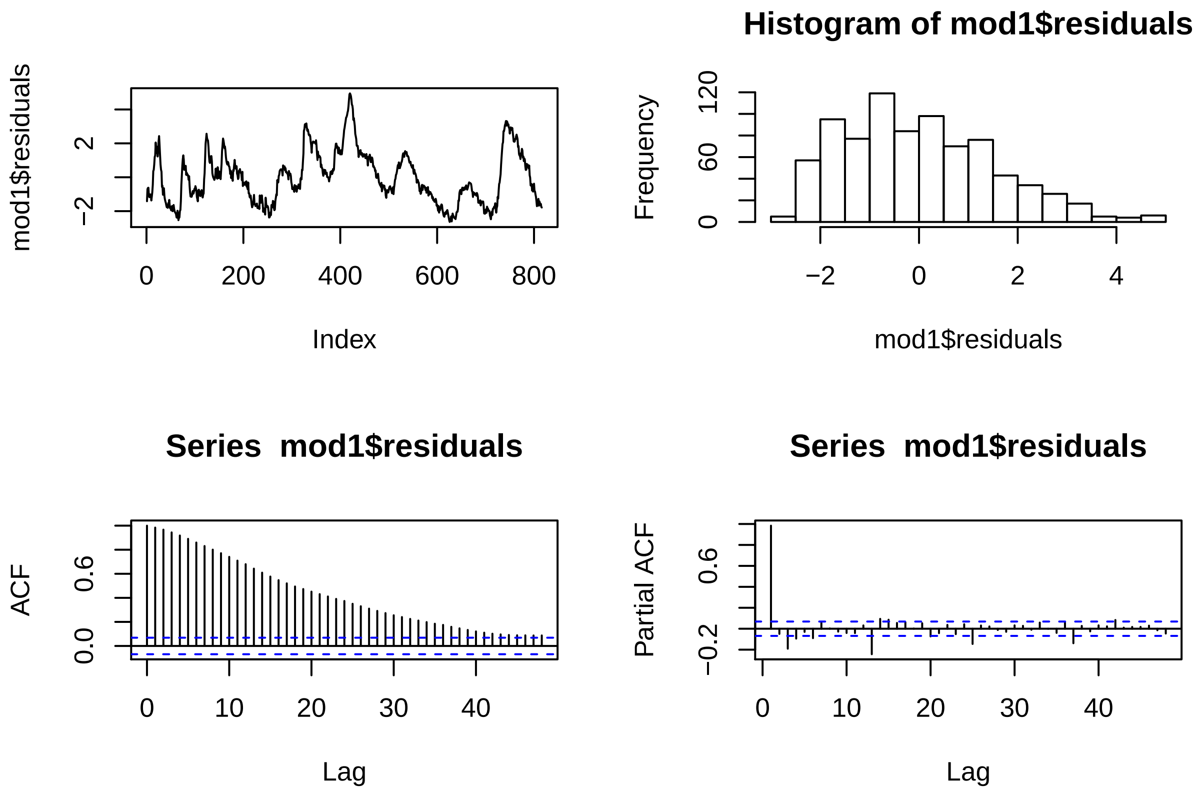
```
## 2018 2.8532757 2.9254117 2.9961322 3.0654579 3.1334243
## 2019 3.7538955 3.8281286 3.9012349 3.9732072 4.0440551
## 2020 4.6837810 4.7611418 4.8375278 4.9129208 4.9873183
```

b. Build a linear time regression and incorporate seasonal effects. Be sure to evaluate the residuals and assess this model on the bases of the assumptions of the classical linear model, and the produce a 1-year and 4-year forecast.

```
### fit linear model w/ shorter window for data set, account for stochastic trend due to event
unemp$DATE <- as.Date(unemp$DATE)
unemp$time <- rank(unemp$DATE)
unemp$month <- months(unemp$DATE)
unemp$year <- ceiling(unemp$time/12)
unemp_test <- unemp[unemp$DATE>"2015-12-01",]
unemp_train <- unemp[unemp$DATE<"2016-01-01",]

mod1 <- lm(UNRATENSA ~ time + month, data = unemp_train)
summary(mod1)
```

```
##
## Call:
## lm(formula = UNRATENSA ~ time + month, data = unemp_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.637 -1.156 -0.158  1.068  4.949
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.6084708  0.2087918  22.072  < 2e-16 ***
## time             0.0026124  0.0002291  11.403  < 2e-16 ***
## monthAugust     -0.0178027  0.2643577  -0.067  0.94633
## monthDecember   -0.1547230  0.2643625  -0.585  0.55853
## monthFebruary    0.7934602  0.2643565   3.001  0.00277 **
## monthJanuary     0.8048961  0.2643570   3.045  0.00240 **
## monthJuly        0.2906921  0.2643570   1.100  0.27183
## monthJune        0.4535987  0.2643565   1.716  0.08657 .
## monthMarch       0.5143771  0.2643562   1.946  0.05203 .
## monthMay        -0.1467301  0.2643562  -0.555  0.57902
## monthNovember   -0.1932870  0.2643610  -0.731  0.46490
## monthOctober    -0.3642040  0.2643597  -1.378  0.16868
## monthSeptember  -0.1954151  0.2643586  -0.739  0.46000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.541 on 803 degrees of freedom
## Multiple R-squared:  0.1833, Adjusted R-squared:  0.1711
## F-statistic: 15.02 on 12 and 803 DF,  p-value: < 2.2e-16
```

```
# check residuals - normally distributed (plot, histogram, Q-Q), white noise (ACF, PACF)
par(mfrow = c(2,2))
plot(mod1$residuals, type = "l")
hist(mod1$residuals)
acf(mod1$residuals, lag.max = 48)
pacf(mod1$residuals, lag.max = 48)
```




```r
# plot fitted values on training set
plot(unemp_train$UNRATENSA~unemp_train$time, type = "l") + lines(predict(mod1,newdata = unemp_train), c
```

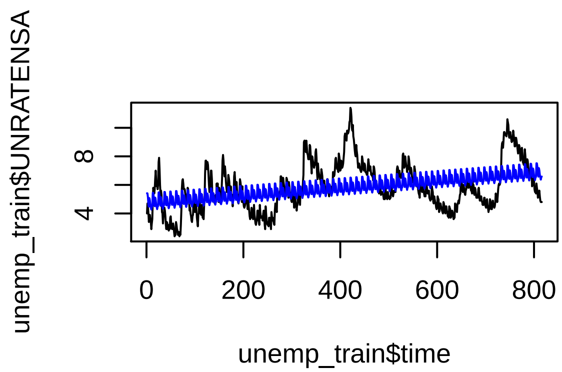
```
## numeric(0)
```

```r
# forecast input data
test <- data.frame(time = 817:876, month = rep(c("January", "February", "March", "April", "May", "June"

### forecast through June 2017
accuracy(predict(mod1, newdata = test[1:18,]),unemp_test$UNRATENSA)
```

```
##                 ME    RMSE      MAE      MPE    MAPE
## Test set -2.237503 2.26136 2.237503 -47.4506 47.4506
```

```r
### forecast through December 2020
accuracy(predict(mod1, newdata = test[1:60,]),unemp_test$UNRATENSA)
```

```
##                 ME    RMSE      MAE      MPE    MAPE
## Test set -2.237503 2.26136 2.237503 -47.4506 47.4506
```



- How well does your model predict the unemployment rate up until June 2017?

The linear model does not predict the unemployment rate well, with a RMSE of 2.26 on the test data set.

- What does the unemployment rate look like at the end of 2020? How credible is this estimate?

The linear model predicts an unemployment rate of 6.56% at the end of 2020.

- Compare this forecast to the one produced by the SARIMA model. What do you notice?

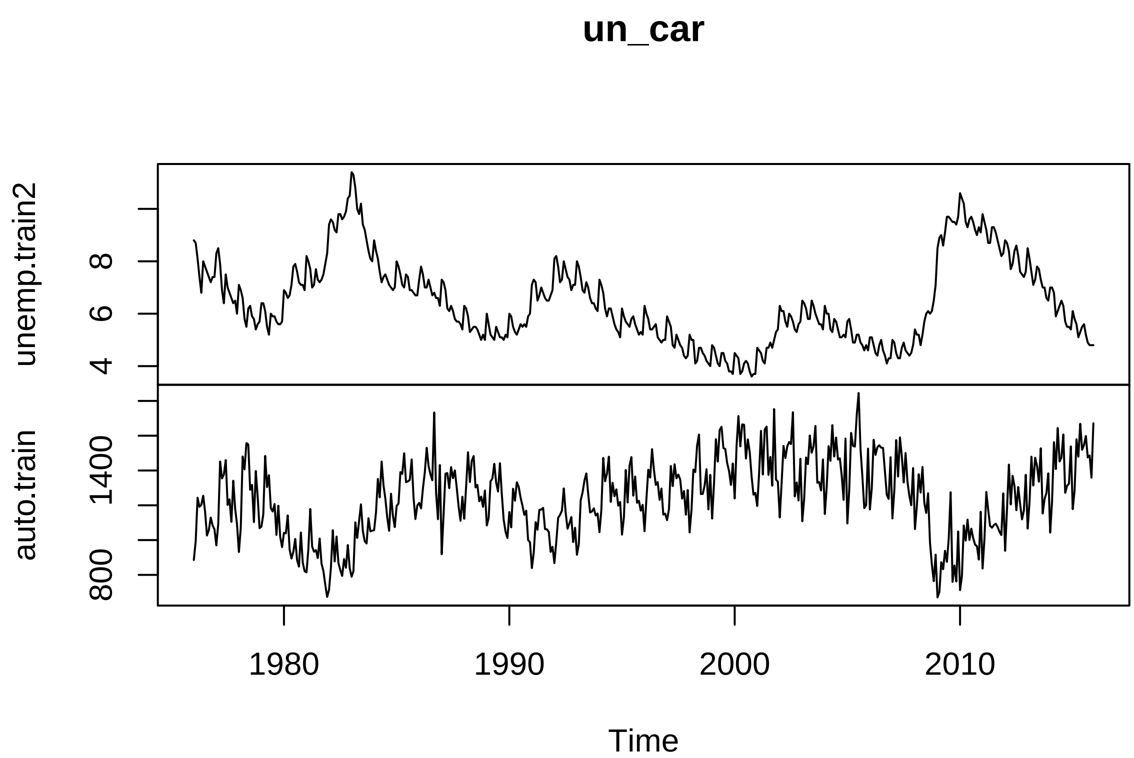
The forecast from the linear model does not contain any trend and contains more uniform predictions compared to those of the SARIMA model. The forecast is also higher than the test set.

Question 3: VAR.

You also have data on automotive car sales.

- Use a VAR model to produce a 1-year forecast on both the unemployment rate and automotive sales for 2017 in the U.S.

```r
# split data into training and test sets
unemp.train2 = window(unemp.train, start=c(1976,1), end=c(2015,12), frequency=12)
unemp.test = window(unemp.ts, start=c(2016,1), frequency=12)
auto.train = window(auto.ts, end=c(2015,12), frequency=12)
auto.test = window(auto.ts, start=c(2016,1), frequency=12)

un_car = cbind(unemp.train2, auto.train)
plot(un_car)
```




```r
# use ddlx and ddy to examine if the differenced data is stationary
ddlu.train = window(ddlu, start=c(1977,2), end=c(2015,12), frequency=12)
ddlu.test = window(ddlu, start=c(2016,1), frequency=12)
ddla.train = window(ddla, end=c(2015,12), frequency=12)
ddla.test = window(ddla, start=c(2016,1), frequency=12)

# differenced and seasonally differenced data is stationary
adf.test(ddlu.train)  # p = 0.01 < 0.05, data is stationary
```

```
## Warning in adf.test(ddlu.train): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ddlu.train
## Dickey-Fuller = -6.1878, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
pp.test(ddlu.train)  # p = 0.01 < 0.05, data is stationary
```

```
## Warning in pp.test(ddlu.train): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  ddlu.train
## Dickey-Fuller Z(alpha) = -615.36, Truncation lag parameter = 5,
## p-value = 0.01
```

```
## alternative hypothesis: stationary
```

```r
adf.test(ddla.train)  # p = 0.01 < 0.05, data is stationary
```

```
## Warning in adf.test(ddla.train): p-value smaller than printed p-value
```

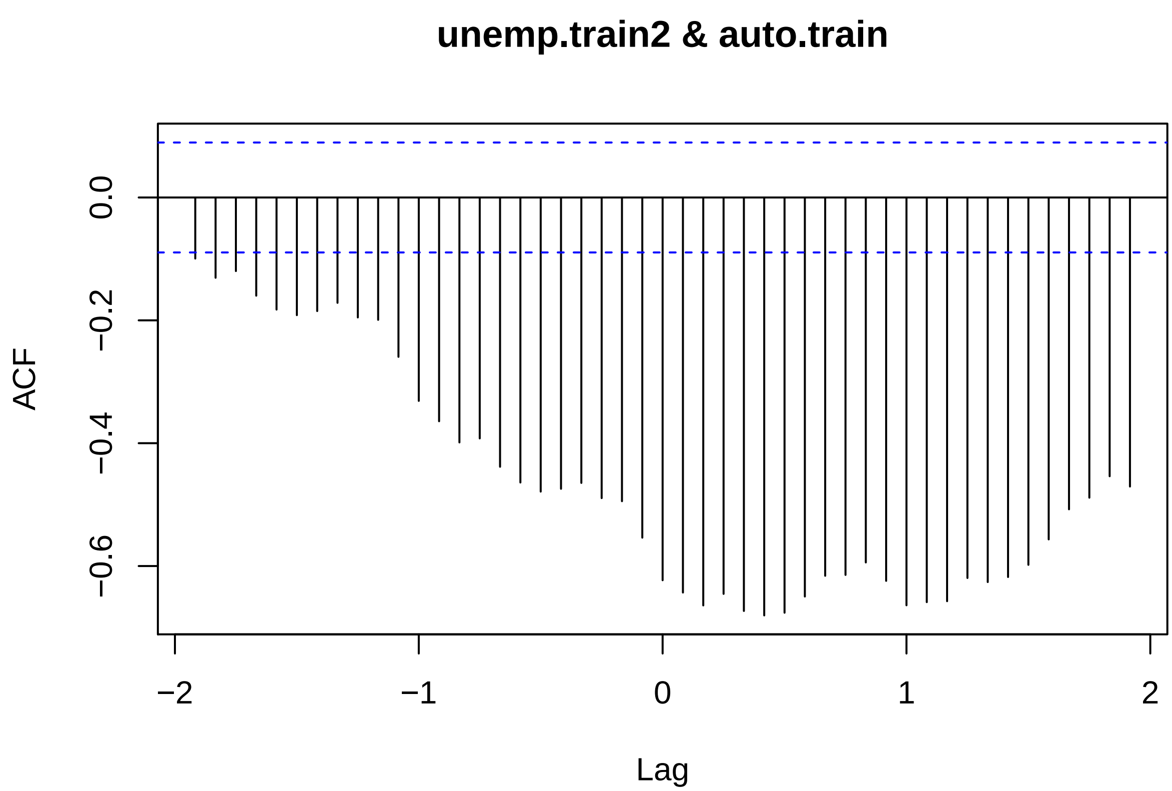
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ddla.train
## Dickey-Fuller = -11.046, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
pp.test(ddla.train)  # p = 0.01 < 0.05, data is stationary
```

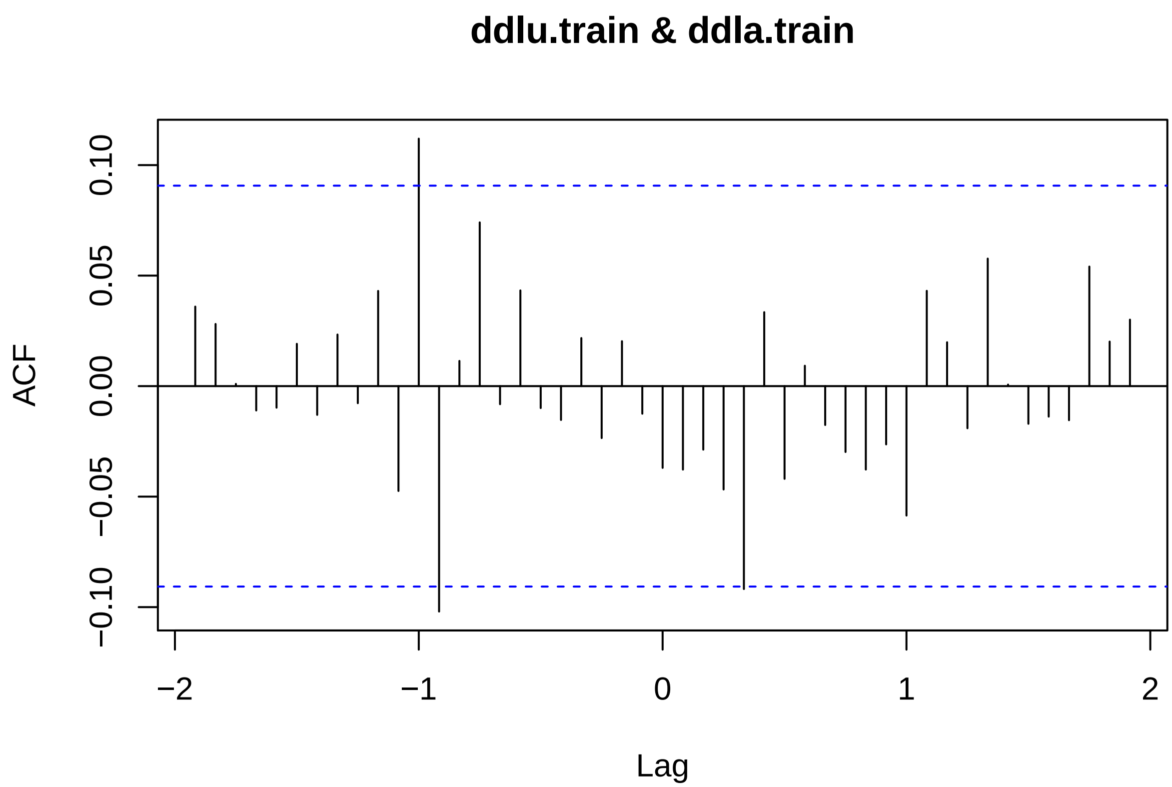
```
## Warning in pp.test(ddla.train): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  ddla.train
## Dickey-Fuller Z(alpha) = -518.5, Truncation lag parameter = 5,
## p-value = 0.01
## alternative hypothesis: stationary
```

```r
# ccf for original data and differenced data
ccf(unemp.train2, auto.train)
```



```r
ccf(ddlu.train, ddla.train)
```

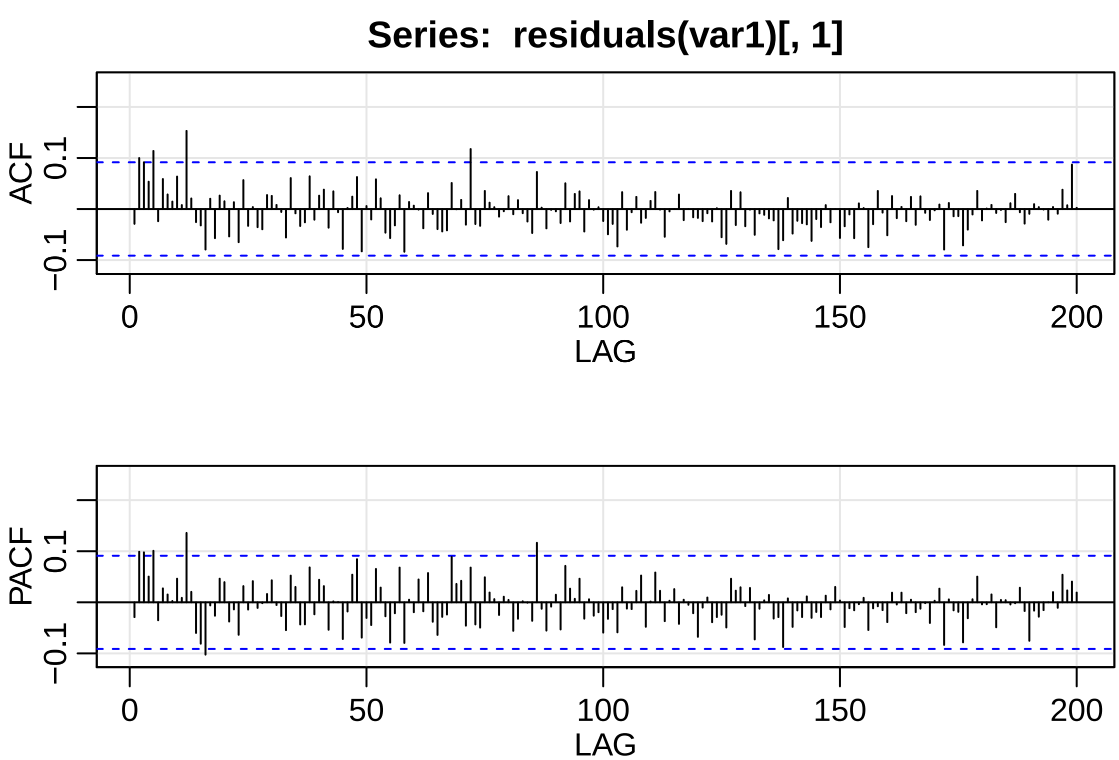



```r
# examine optimal VAR order by AIC, p = 13 shows smallest AIC
VARselect(un_car, type='both', lag.max = 30, season=12)
```

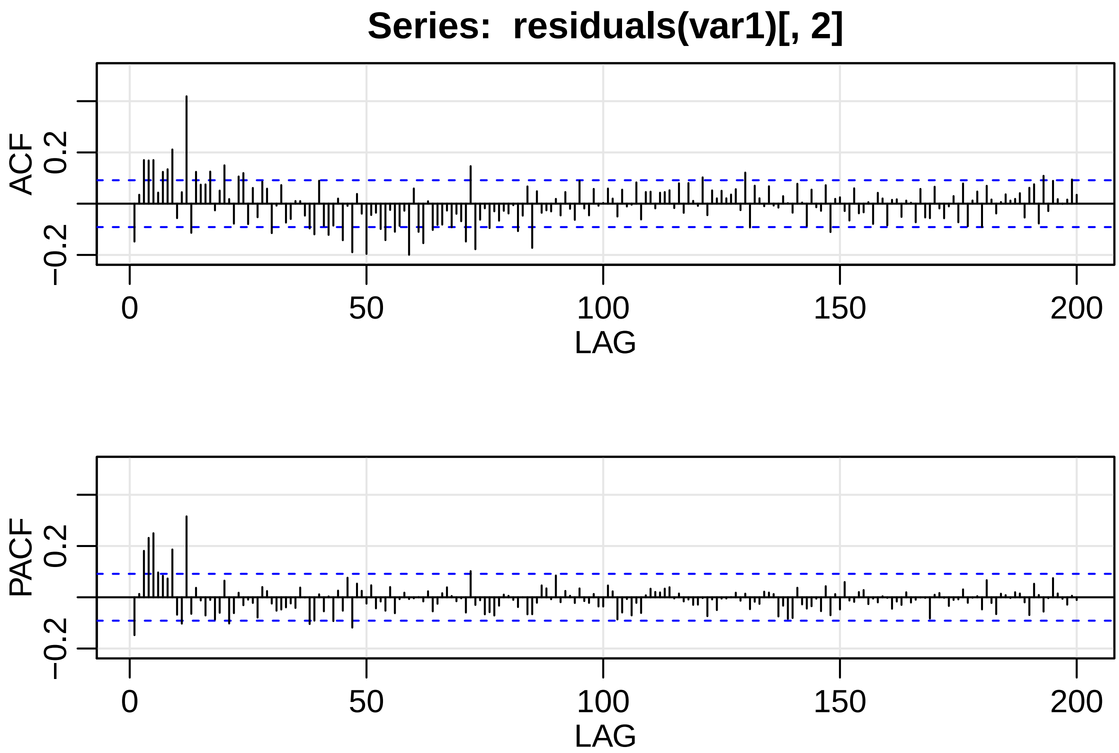
```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     13      4      4     13
##
## $criteria
##                 1          2          3          4          5          6
## AIC(n)   5.955596   5.884010   5.792751   5.736120   5.731808   5.716316
## HQ(n)    6.063570   6.006381   5.929518   5.887284   5.897368   5.896272
## SC(n)    6.229546   6.194487   6.139754   6.119650   6.151865   6.172899
## FPE(n) 385.925893 359.272890 327.946908 309.901948 308.581674 303.853351
##                 7          8          9         10         11         12
## AIC(n)   5.715299   5.712470   5.717751   5.717756   5.716798   5.645467
## HQ(n)    5.909652   5.921220   5.940897   5.955298   5.968737   5.911802
## SC(n)    6.208409   6.242107   6.283914   6.320445   6.356014   6.321210
## FPE(n) 303.562558 302.725999 304.352924 304.381877 304.121591 283.216273
##                13         14         15         16         17         18
## AIC(n)   5.613129   5.620887   5.621708   5.633582   5.640832   5.638495
## HQ(n)    5.893861   5.916015   5.931233   5.957503   5.979150   5.991209
## SC(n)    6.325398   6.369683   6.407031   6.455431   6.499208   6.533397
## FPE(n) 274.239200 276.414214 276.684368 280.037424 282.128007 281.527002
##                19         20         21         22         23         24
```

```
## AIC(n)    5.644440   5.655061   5.649709   5.663132   5.675042   5.683417
## HQ(n)     6.011551   6.036569   6.045613   6.073433   6.099739   6.122510
## SC(n)     6.575870   6.623017   6.654192   6.704142   6.752578   6.797479
## FPE(n) 283.268788 286.362313 284.907811 288.838660 292.387163 294.940892
##                25         26         27         28         29         30
## AIC(n)    5.681346   5.690357   5.702597   5.716520   5.729388   5.743720
## HQ(n)     6.134836   6.158244   6.184881   6.213199   6.240464   6.269193
## SC(n)     6.831936   6.877473   6.926240   6.976689   7.026084   7.076943
## FPE(n) 294.432166 297.206300 300.984183 305.330690 309.421621 314.034847
```

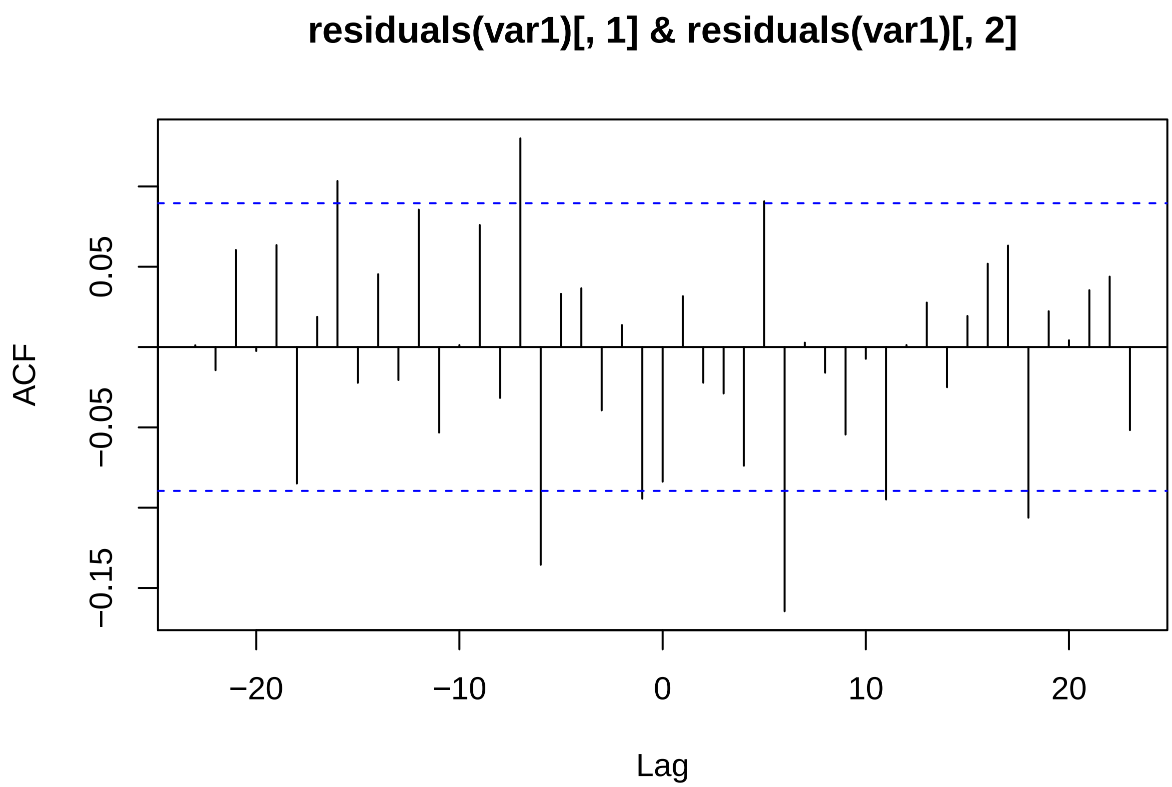
```
# season is assigned. QUESTION: should we also assign const or trend?
var1 <- VAR(un_car, p = 1, type='both', season = 12)
var13 <- VAR(un_car, p = 13, type='both', season = 12)

invisible(acf2(residuals(var1)[,1], 200))
```



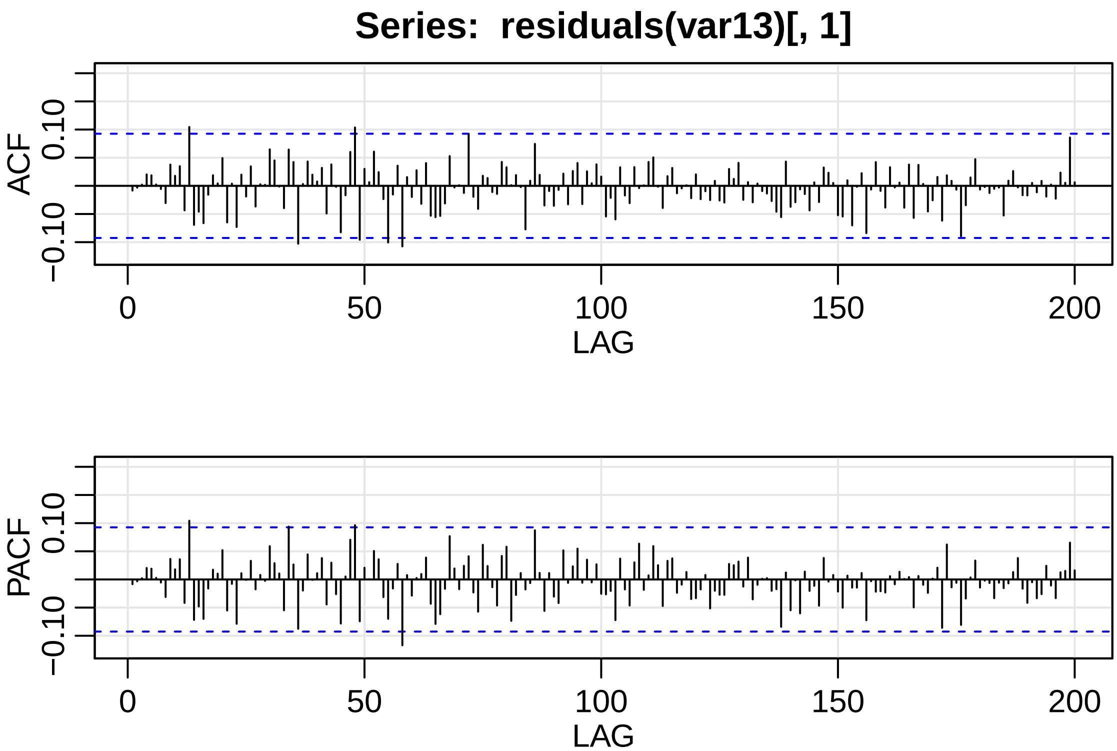

```
invisible(acf2(residuals(var1)[,2], 200))
```



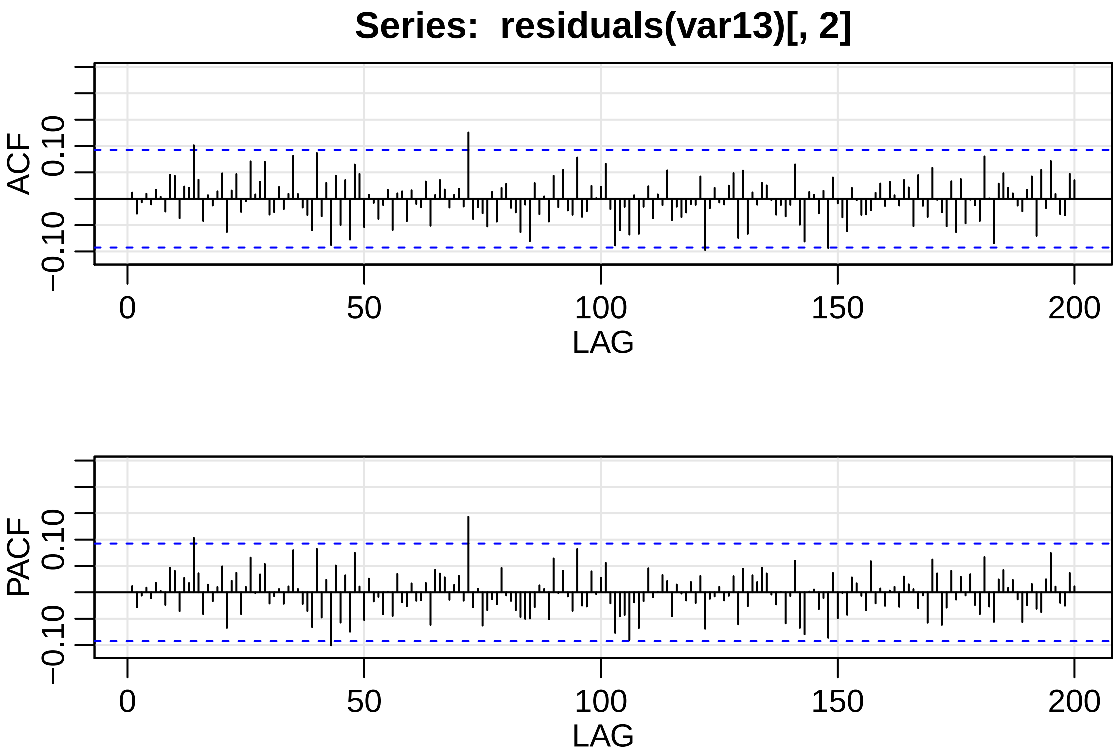

```
ccf(residuals(var1)[,1], residuals(var1)[,2])
```



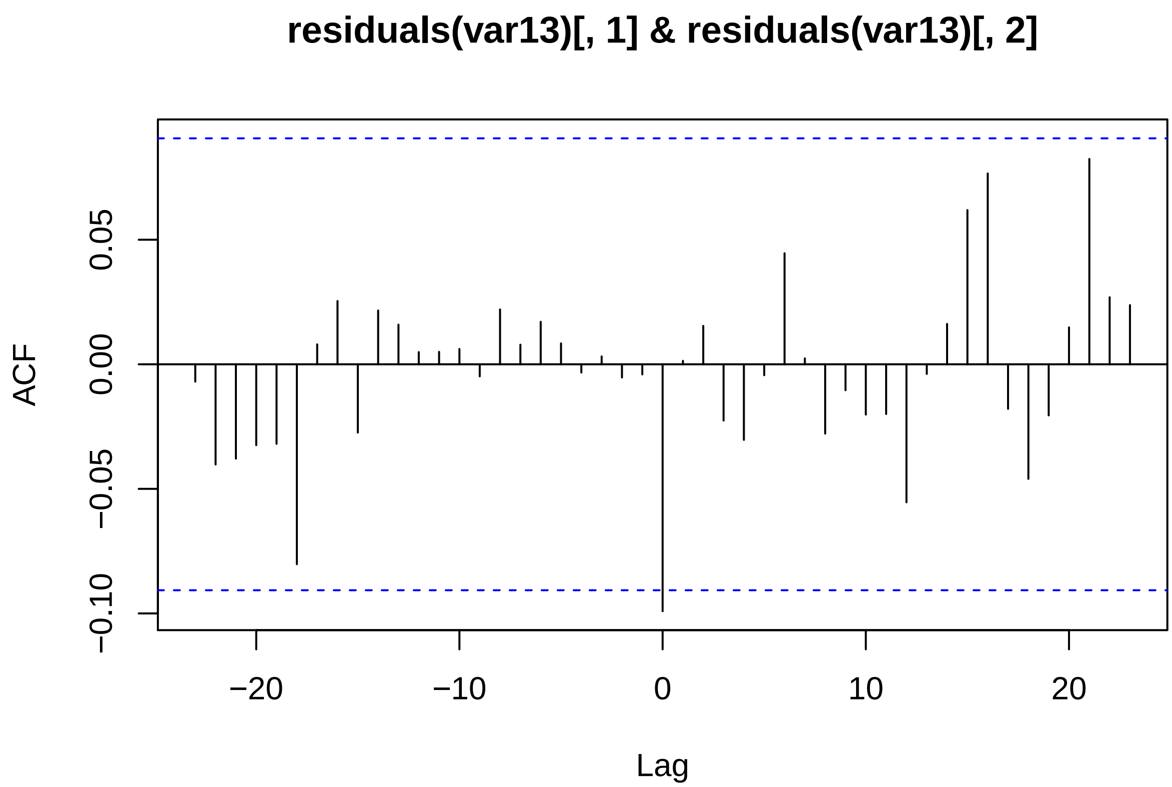

```
# var13, apparently, var13 beats var1
invisible(acf2(residuals(var13)[,1], 200))
```



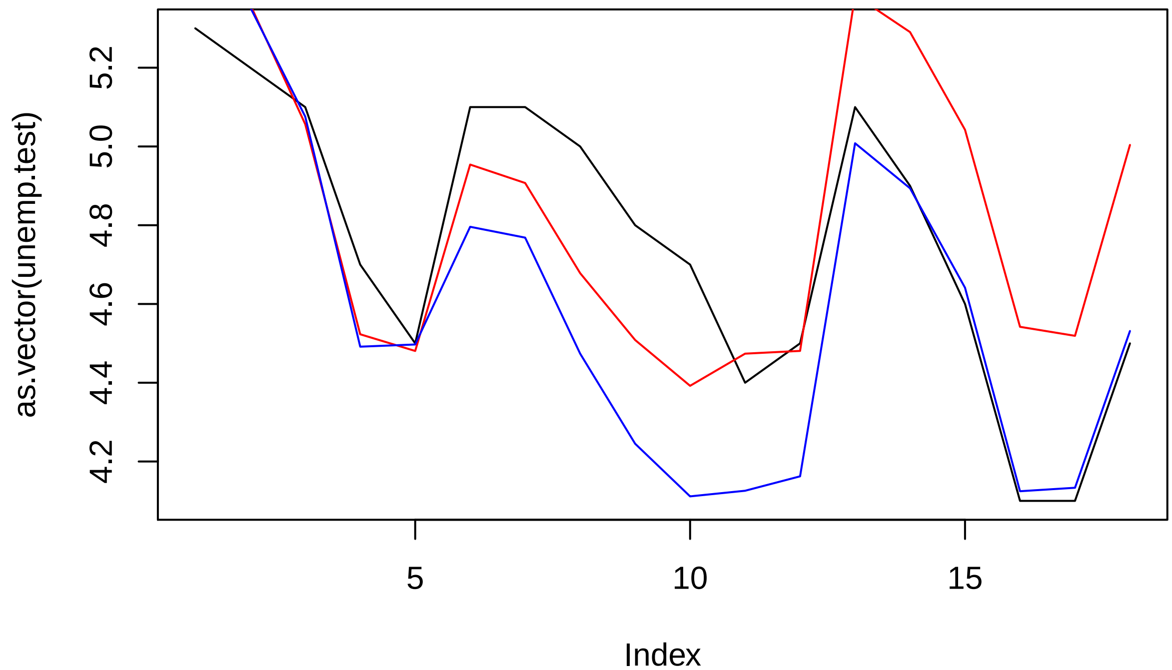

```
invisible(acf2(residuals(var13)[,2], 200))
```



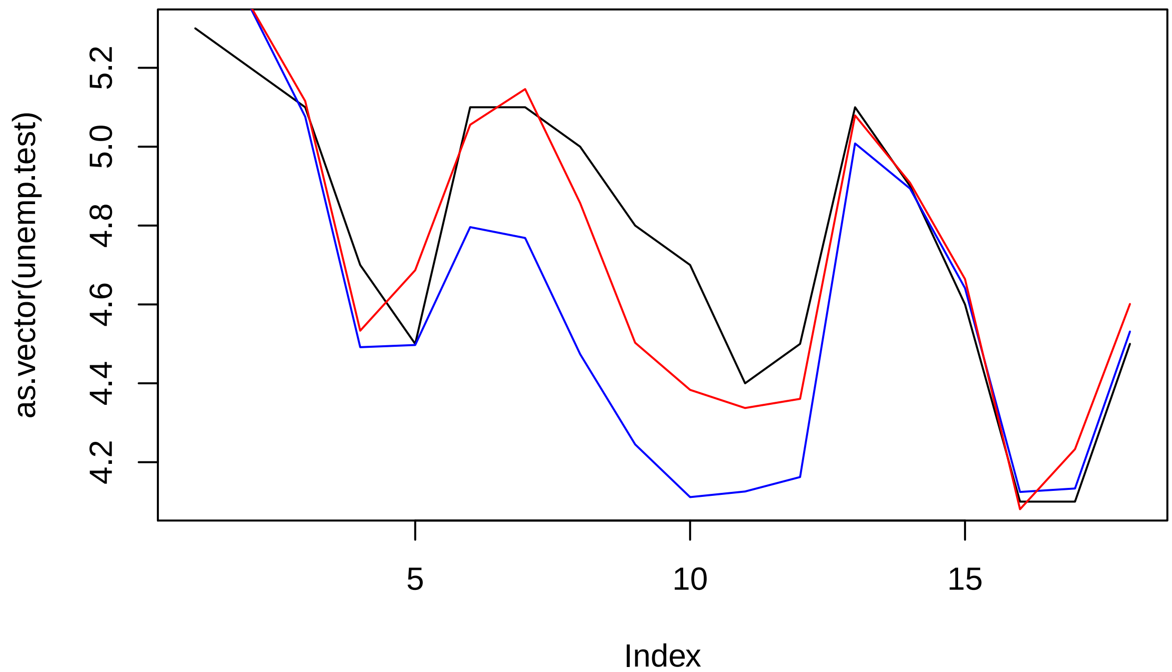

```
ccf(residuals(var13)[,1], residuals(var13)[,2])
```




```
### Forecasts
f1 <- predict(var1, n.ahead = length(unemp.test))
f13 <- predict(var13, n.ahead = length(auto.test))

# plot x.test vs forecasts, var30 shows best fitting, corresponding to residuals analysis
plot(as.vector(unemp.test), type = 'l')
lines(f1$fcst$unemp.train2[,1], col = 'red')
lines(f13$fcst$unemp.train2[,1], col = 'blue')
```




```
# compare SARIMA and VAR
# VAR is not as good fitting as SARIMA, because car sales has little causal correlation with unemployme
plot(as.vector(unemp.test), type = 'l')
lines(f13$fcst$unemp.train2[,1], col = 'blue')
lines(as.vector(for3$mean), col = 'red')
```




```r
# check accuracy vs. SARIMA model
accuracy(f13$fcst$unemp.train2[,1], unemp.test)  # RMSE = 0.285
```

```
##                  ME      RMSE       MAE      MPE     MAPE     ACF1
## Test set 0.1514737 0.2848653 0.2096747 3.178804 4.343368 0.710593
##          Theil's U
## Test set 0.8722405
```

```r
accuracy(for3, unemp.test)  # RMSE = 0.151
```

```
##                        ME      RMSE       MAE         MPE     MAPE
## Training set -0.003534511 0.2343297 0.1778710 -0.01030263 3.320626
## Test set      0.015050078 0.1506237 0.1193972  0.32696023 2.518034
##                   MASE          ACF1 Theil's U
## Training set 0.2010520 -0.0004805123        NA
## Test set     0.1349576  0.3868569701 0.4453878
```

```
### add plots with VAR and SARIMA forecast
```

- Compare the 1-year forecast for unemployment produced by the VAR and SARIMA models, examining both the accuracy AND variance of the forecast. Do you think the addition of the automotive sales data helps? Why or why not?

Adding automotive sales data does not help the unemployment forecast. The forecast accuracy of the VAR model on the test is less than that of the SARIMA model since RMSE increases from 0.151 to 0.285.