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

Kari Ross and Jan Forslow

April 2, 2017

Question 1

ECOMPCTNSA.csv, contains quarterly data of E-Commerce Retail Sales as a Percent of Total Sales. The data can be found at: https://fred.stlouisfed.org/series/ECOMPCTNSA.

Build a Seasonal ARIMA model and generate quarterly forecast for 2017. Make sure you use all the steps of building a univariate time series model between lecture 6 and 9, such as checking the raw data, conducting a thorough EDA, justifying all modeling decisions (including transformation), testing model assumptions, and clearly articulating why you chose your given model. Measure and discuss your model's performance. Use both in-sample and out-of-sample model performance. When training your model, exclude the series from 2015 and 2016. For the out-of-sample forecast, measure your model's performance in forecasting the quarterly E-Commerce retail sales in 2015 and 2016. Discuss the model performance. Also forecast beyond the observed time-period of the series. Specifically, generate quarterly forecast for 2017.

Initialization

Loading the libraries used in the analysis and setting working directory.

```
# Clean up the workspace before we begin
rm(list = ls())

# Libraries
library(astsa)  # Time series package by Shummway and Stoffer
library(forecast)  # Time series forecast package by Rob Hyndman
library(zoo)  # time series package
library(dplyr)
library(Hmisc)
library(ggplot2)
library(psych)  # describe
library(tseries)  # time series creation

# Insert the function to *tidy up* the code when they are printed out
library(knitr)
opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE, warning = FALSE)

# Set working directory
wd <- "C:/Users/jfors/Documents/Berkeley/W271_Statistical_Methods/Lab3"
setwd(wd)
```

Common Functions

These are common functions that are being used in the later analysis.

```r
# Function to check that time series are continious without
# gap.
elapsed_months <- function(end_date, start_date) {
    ed <- as.POSIXlt(end_date)
    sd <- as.POSIXlt(start_date)
    12 * (ed$year - sd$year) + (ed$mon - sd$mon)
}

# Calculate RMSE
calculate_rmse <- function(fcast, test) {
    rmse <- sqrt(mean((fcast - test)^2))
}
```

EDA

The dataset contains 69 observations of 2 variables (DATE and ECOMPCTNSA). ECOMPCTNSA stands for E-Commerce Retail Sales as a Percent of Total Sales. The time series starts with DATE 1999-10-01 and ends with 2016-04-01. It is a continuous quarterly series with no missing Date and NA values as Elapsed_months check = 66 quarters which matches with 69 observations (last observation omitted). Mean is 3.83. Min is 0.7 (positive number) and max is 9.5. Even if the latter is an outlier versus other observations it is still reasonable to occur and we will not take any action.

Looking first at the complete time series. It has a steady upwards trend with seasonal variation that is steadily increasing over time (evaluate log transformation). ACF is oscillating (no MA component), while PACF has a non-seasonal AR(1) and a seasonal AR(1) at lag 4 in this case. The histogram shows a positively skewed distribution with strong negative curtosis.

```r
# Load the data from a csv file into an R data frame
df <- read.csv("ECOMPCTNSA.csv", header = TRUE, stringsAsFactors = FALSE)

# Examine the data structure
str(df)
```

```
## 'data.frame':    69 obs. of  2 variables:
##  $ DATE      : chr  "1999-10-01" "2000-01-01" "2000-04-01" "2000-07-01" ...
##  $ ECOMPCTNSA: num  0.7 0.8 0.8 0.9 1.1 1.1 1 1 1.3 1.3 ...
```

```r
names(df)
```

```
## [1] "DATE"       "ECOMPCTNSA"
```

```r
head(df)
```

```
##         DATE ECOMPCTNSA
## 1 1999-10-01        0.7
## 2 2000-01-01        0.8
## 3 2000-04-01        0.8
## 4 2000-07-01        0.9
## 5 2000-10-01        1.1
## 6 2001-01-01        1.1
```

```r
tail(df)
```

```
##          DATE ECOMPCTNSA
## 64 2015-07-01        6.8
## 65 2015-10-01        8.7
```

```
## 66 2016-01-01        7.7
## 67 2016-04-01        7.5
## 68 2016-07-01        7.7
## 69 2016-10-01        9.5
```

```
View(df)
describe(df)
```

```
##            vars  n mean   sd median trimmed  mad min  max range skew
## DATE*         1 69  NaN   NA     NA     NaN   NA Inf -Inf  -Inf   NA
## ECOMPCTNSA    2 69 3.83 2.21    3.6    3.69 2.52 0.7  9.5   8.8 0.49
##            kurtosis   se
## DATE*            NA   NA
## ECOMPCTNSA    -0.67 0.27
```

```
elapsed_months("2016-04-01", "1999-10-01")/3
```

```
## [1] 66
```

```
# Convert a column of the data frame into a time-series
# object
ecompctnsa <- ts(df$ECOMPCTNSA, start = c(1999, 4), frequency = 4)
str(ecompctnsa)
```

```
##  Time-Series [1:69] from 2000 to 2017: 0.7 0.8 0.8 0.9 1.1 1.1 1 1 1.3 1.3 ...
```

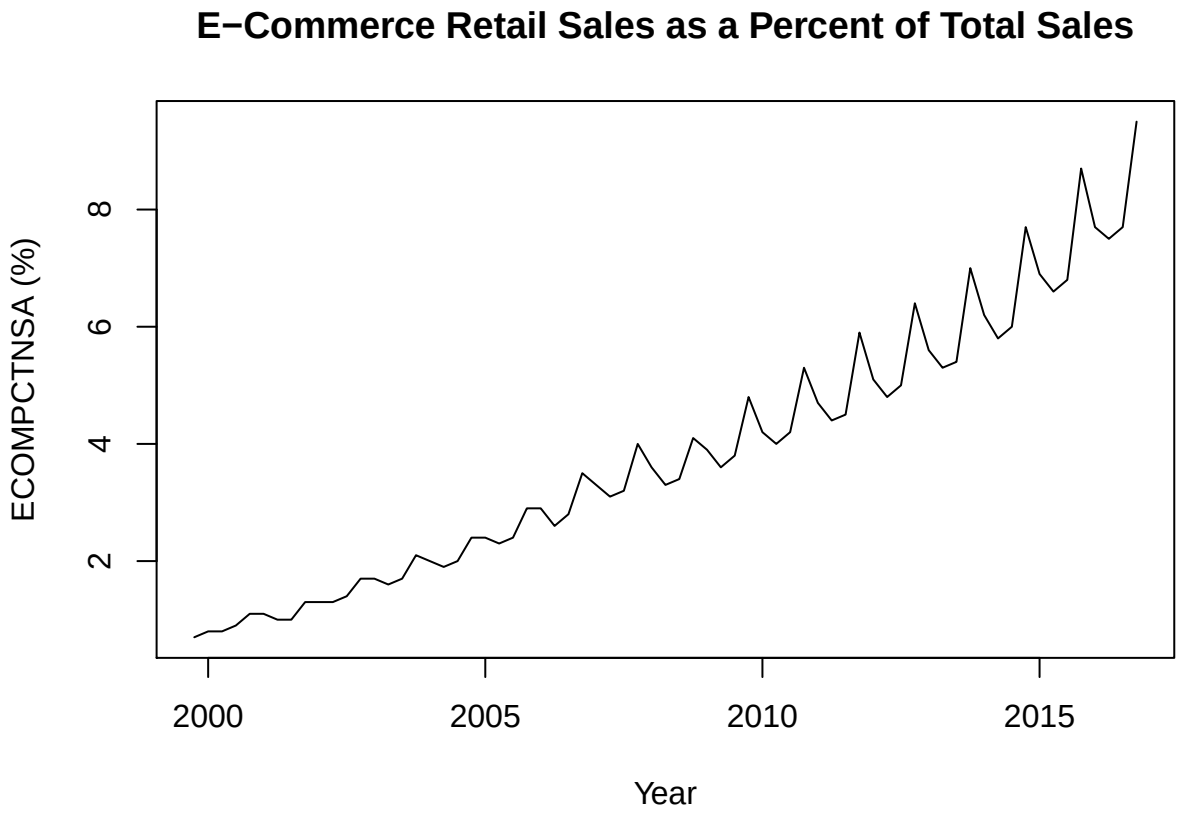
```
head(cbind(time(ecompctnsa), ecompctnsa), 8)
```

```
##      time(ecompctnsa) ecompctnsa
## [1,]          1999.75        0.7
## [2,]          2000.00        0.8
## [3,]          2000.25        0.8
## [4,]          2000.50        0.9
## [5,]          2000.75        1.1
## [6,]          2001.00        1.1
## [7,]          2001.25        1.0
## [8,]          2001.50        1.0
```

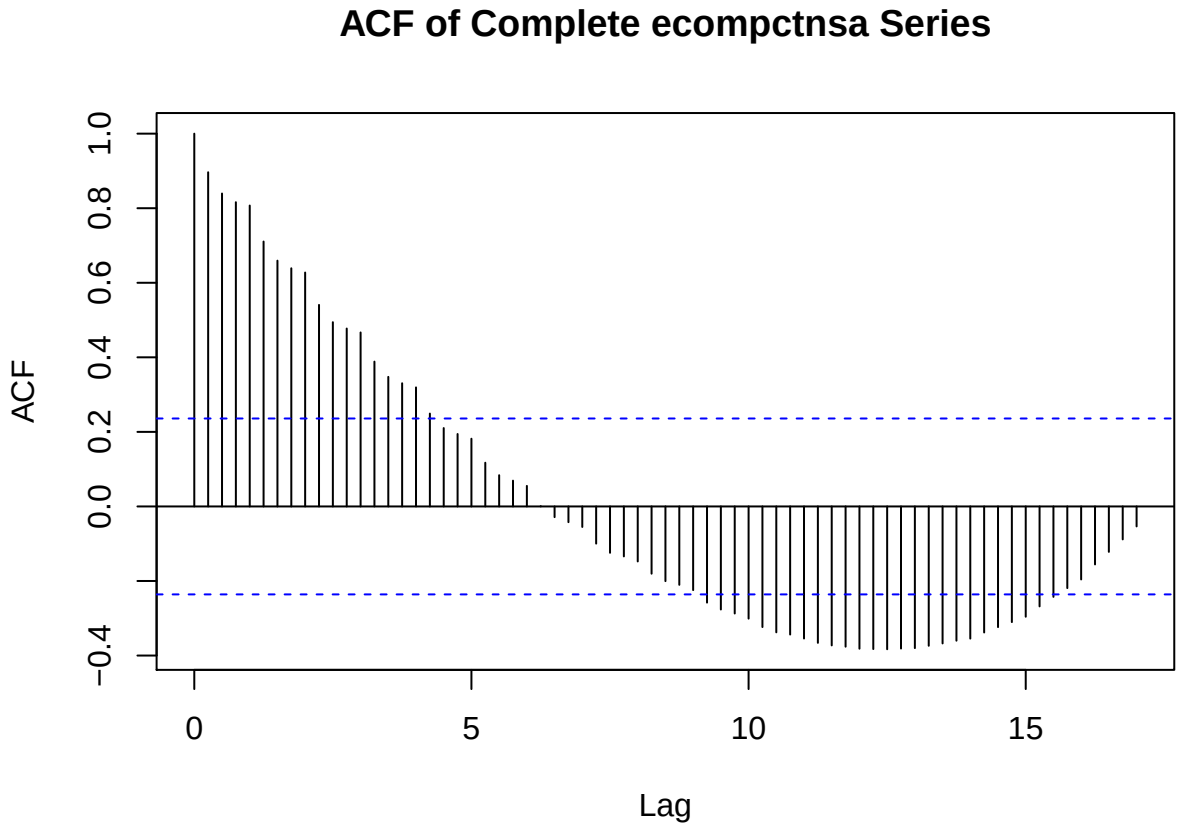
```
tail(cbind(time(ecompctnsa), ecompctnsa), 8)
```

```
##       time(ecompctnsa) ecompctnsa
## [62,]          2015.00        6.9
## [63,]          2015.25        6.6
## [64,]          2015.50        6.8
## [65,]          2015.75        8.7
## [66,]          2016.00        7.7
## [67,]          2016.25        7.5
## [68,]          2016.50        7.7
## [69,]          2016.75        9.5
```

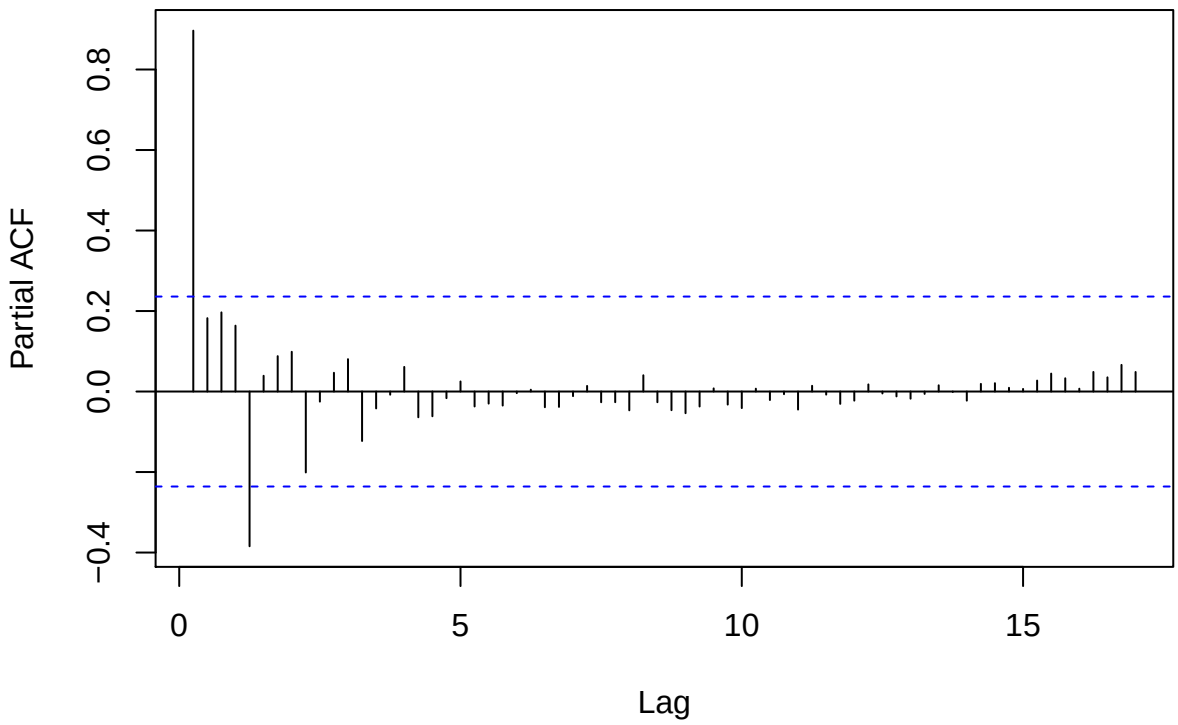
```
# Visualize the original time series
par(mar = c(4, 4, 4, 2))
ts.plot(ecompctnsa, main = "E-Commerce Retail Sales as a Percent of Total Sales",
    ylab = "ECOMPCTNSA (%)", xlab = "Year")
```




```
acf(ecompctnsa, lag.max = 156, main = "ACF of Complete ecompctnsa Series")
```

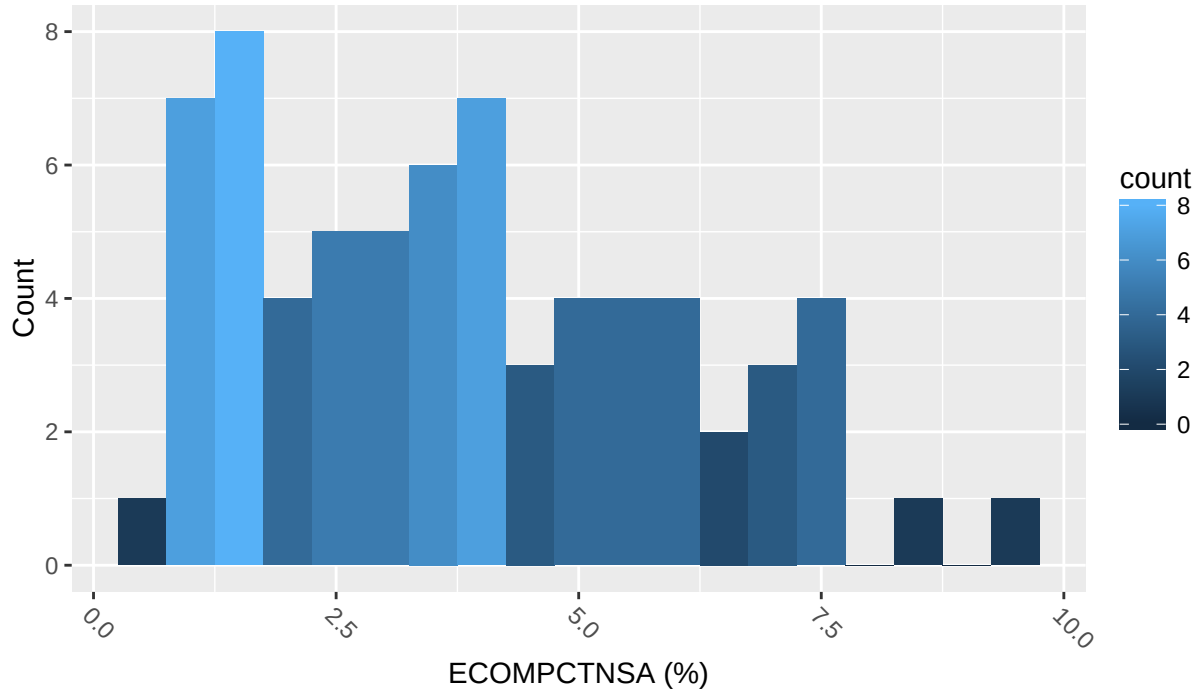



```
pacf(ecompctnsa, lag.max = 156, main = "PACF of Complete ecompctnsa Series")
```




```
ggplot(df, aes(x = ecompctnsa)) + geom_histogram(aes(fill = ..count..),
    binwidth = 0.5) + ggtitle(expression(atop("E-Commerce Retail Sales as a Percent of Total Sales",
    atop(italic("Quarterly 1991-10-01 to 2016-04-01"), "")))) +
    xlab("ECOMPCTNSA (%)") + ylab("Count") + theme(axis.text.x = element_text(angle = -45,
    hjust = 0, vjust = 1), plot.title = element_text(size = 18,
    face = "bold", colour = "black", vjust = -1))
```

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



We take the natural log transformation in order to get a stable variance across time. The time series has now stable variance but still with an increasing trend (should apply differentiating in initial model). The PACF is oscillates slowly (root close to one). The PACF shows strong non-seasonal AR(1) but not any strong seasonal lag.

The histogram of the log transform shows a negatively skewed distribution with strong negative curtosis. No extreme outlier.

```
# Take log transform of ecompctnsa as time-series object
logecompctnsa <- ts(log(df$ECOMPCTNSA), start = c(1999, 4), frequency = 4)
str(logecompctnsa)
```

```
##  Time-Series [1:69] from 2000 to 2017: -0.3567 -0.2231 -0.2231 -0.1054 0.0953 ...
```

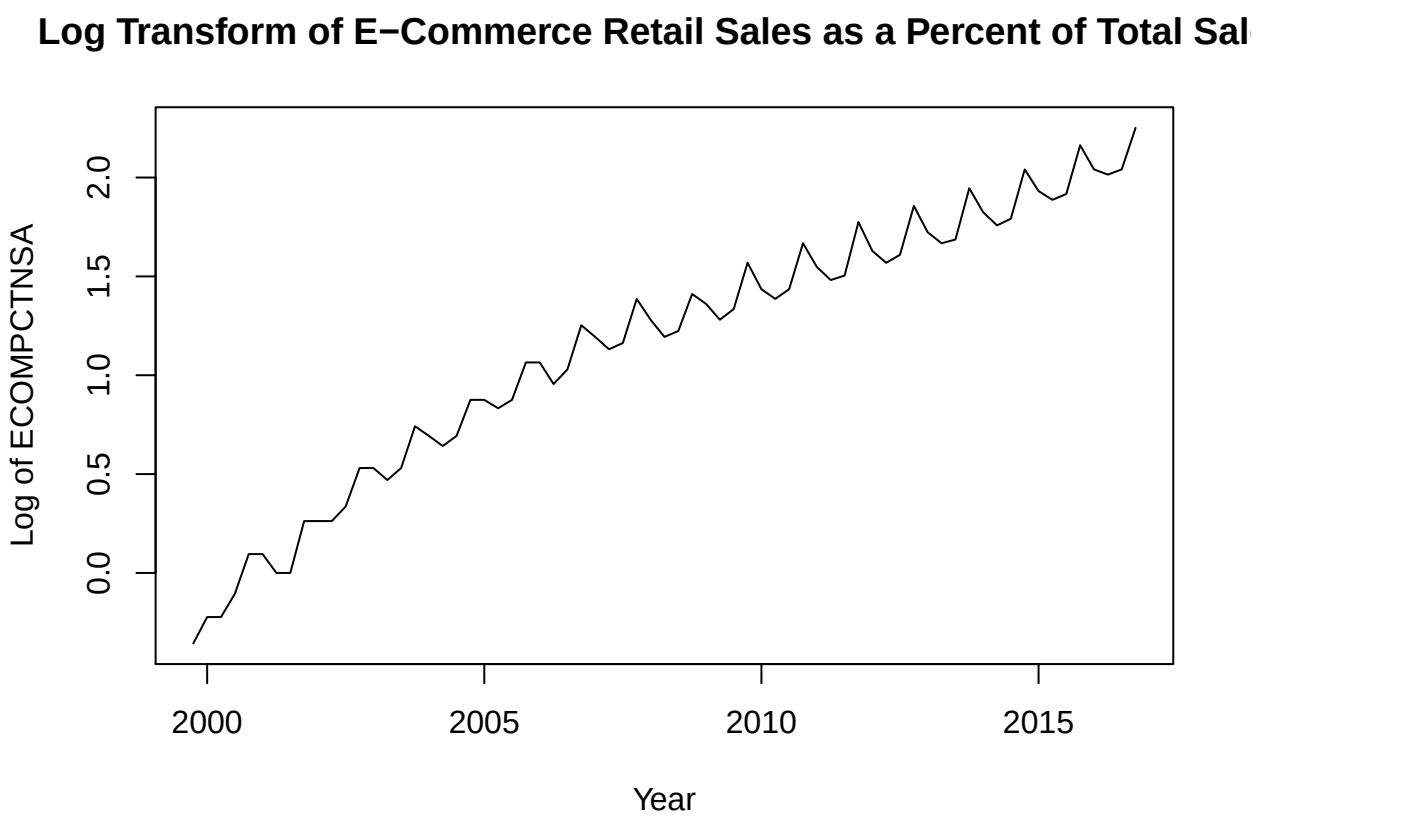
```
# Validate that no data was lost in log transform
head(cbind(time(logecompctnsa), logecompctnsa), 8)
```

```
##      time(logecompctnsa) logecompctnsa
## [1,]             1999.75   -0.35667494
## [2,]             2000.00   -0.22314355
## [3,]             2000.25   -0.22314355
## [4,]             2000.50   -0.10536052
## [5,]             2000.75    0.09531018
## [6,]             2001.00    0.09531018
## [7,]             2001.25    0.00000000
## [8,]             2001.50    0.00000000
```

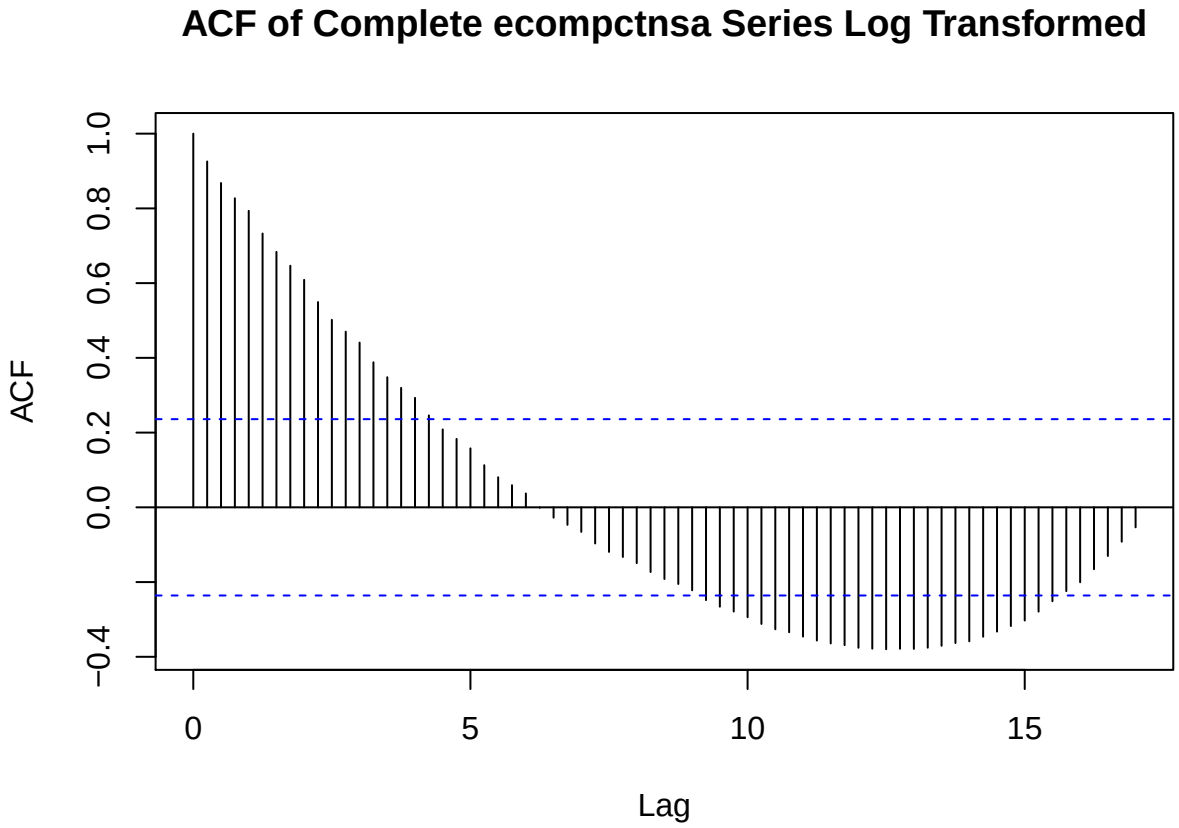
```
tail(cbind(time(logecompctnsa), logecompctnsa), 8)
```

```
##       time(logecompctnsa) logecompctnsa
## [62,]             2015.00      1.931521
## [63,]             2015.25      1.887070
## [64,]             2015.50      1.916923
## [65,]             2015.75      2.163323
## [66,]             2016.00      2.041220
## [67,]             2016.25      2.014903
## [68,]             2016.50      2.041220
## [69,]             2016.75      2.251292
```

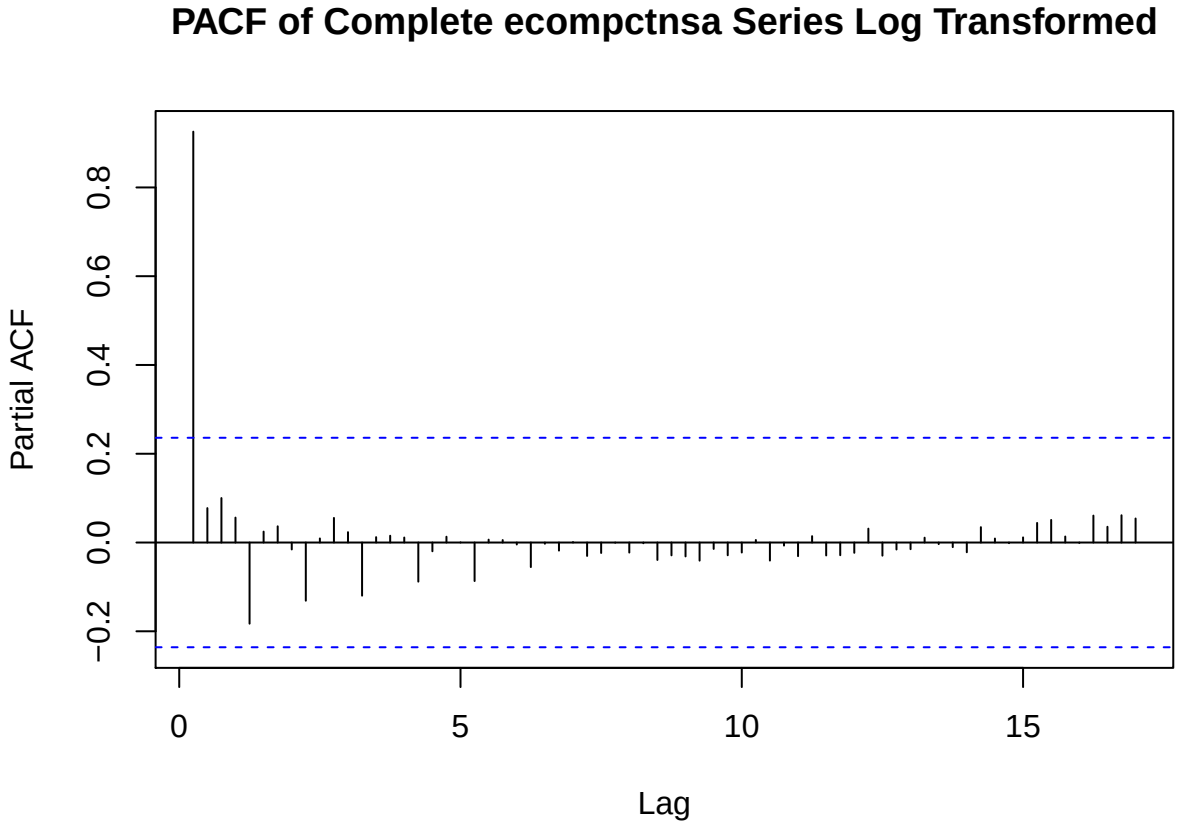
```r
# Visualize the log transformed time series
par(mar = c(4, 4, 4, 2))
ts.plot(logecompctnsa, main = "Log Transform of E-Commerce Retail Sales as a Percent of Total Sales",
    ylab = "Log of ECOMPCTNSA", xlab = "Year")
```




```r
acf(logecompctnsa, lag.max = 156, main = "ACF of Complete ecompctnsa Series Log Transformed")
```

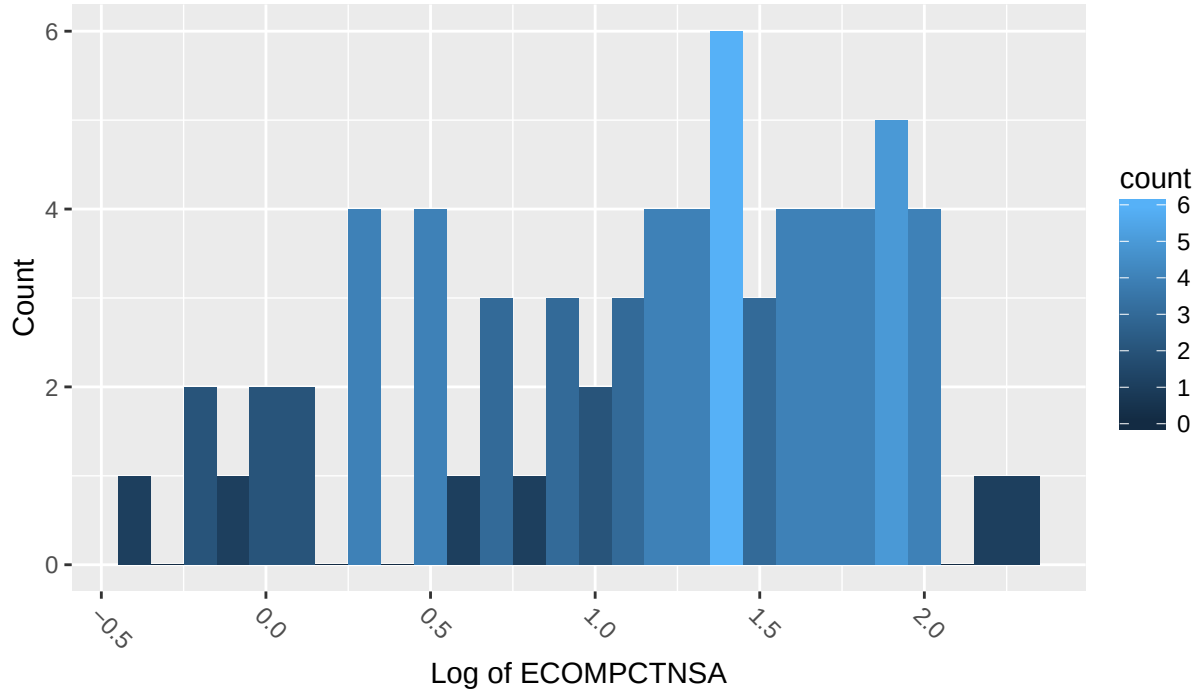



```
pacf(logecompctnsa, lag.max = 156, main = "PACF of Complete ecompctnsa Series Log Transformed")
```




```
ggplot(df, aes(x = logecompctnsa)) + geom_histogram(aes(fill = ..count..),
    binwidth = 0.1) + ggtitle(expression(atop("Log Transform of ecompctnsa",
    atop(italic("Quarterly 1999-10-01 to 2016-04-01"), ""))) +
    xlab("Log of ECOMPCTNSA") + ylab("Count") + theme(axis.text.x = element_text(angle = -45,
    hjust = 0, vjust = 1), plot.title = element_text(size = 18,
    face = "bold", colour = "black", vjust = -1))
```

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



Create Training and Test Data

We divide the data up in training and test dataset based on the question, i.e. 2015 and 2016 observations are kept for in-sample test.

```
# Create a subset for training data. Could also use window.
fp.training <- ts(subset(logecompctnsa, time(logecompctnsa) >=
    1999 & time(logecompctnsa) < 2015), start = c(1999, 4), frequency = 4)
str(fp.training)
```

```
##  Time-Series [1:61] from 2000 to 2015: -0.3567 -0.2231 -0.2231 -0.1054 0.0953 ...
```

```
# Ensure that we have the correct observations selected
head(cbind(time(fp.training), fp.training), 8)
```

```
##      time(fp.training) fp.training
## [1,]           1999.75 -0.35667494
## [2,]           2000.00 -0.22314355
## [3,]           2000.25 -0.22314355
## [4,]           2000.50 -0.10536052
## [5,]           2000.75  0.09531018
## [6,]           2001.00  0.09531018
## [7,]           2001.25  0.00000000
## [8,]           2001.50  0.00000000
```

```
tail(cbind(time(fp.training), fp.training), 8)
```

```
##       time(fp.training) fp.training
## [54,]           2013.00    1.722767
## [55,]           2013.25    1.667707
## [56,]           2013.50    1.686399
## [57,]           2013.75    1.945910
## [58,]           2014.00    1.824549
## [59,]           2014.25    1.757858
## [60,]           2014.50    1.791759
## [61,]           2014.75    2.041220
```

```
fp.test <- ts(subset(logecompctnsa, time(logecompctnsa) >= 2015 &
    time(logecompctnsa) <= 2017), start = c(2015, 1), frequency = 4)
str(fp.test)
```

```
##  Time-Series [1:8] from 2015 to 2017: 1.93 1.89 1.92 2.16 2.04 ...
```

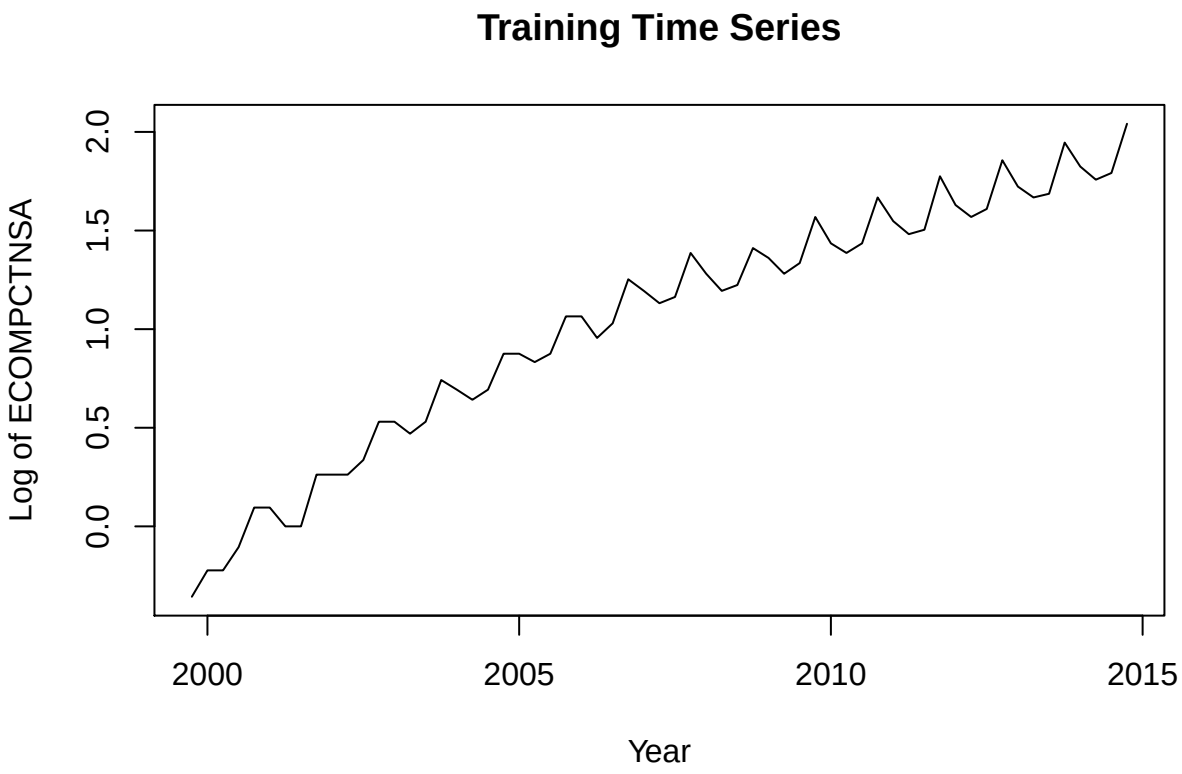
```
head(cbind(time(fp.test), fp.test), 8)
```

```
##      time(fp.test)  fp.test
## [1,]       2015.00 1.931521
## [2,]       2015.25 1.887070
## [3,]       2015.50 1.916923
## [4,]       2015.75 2.163323
## [5,]       2016.00 2.041220
## [6,]       2016.25 2.014903
## [7,]       2016.50 2.041220
## [8,]       2016.75 2.251292
```

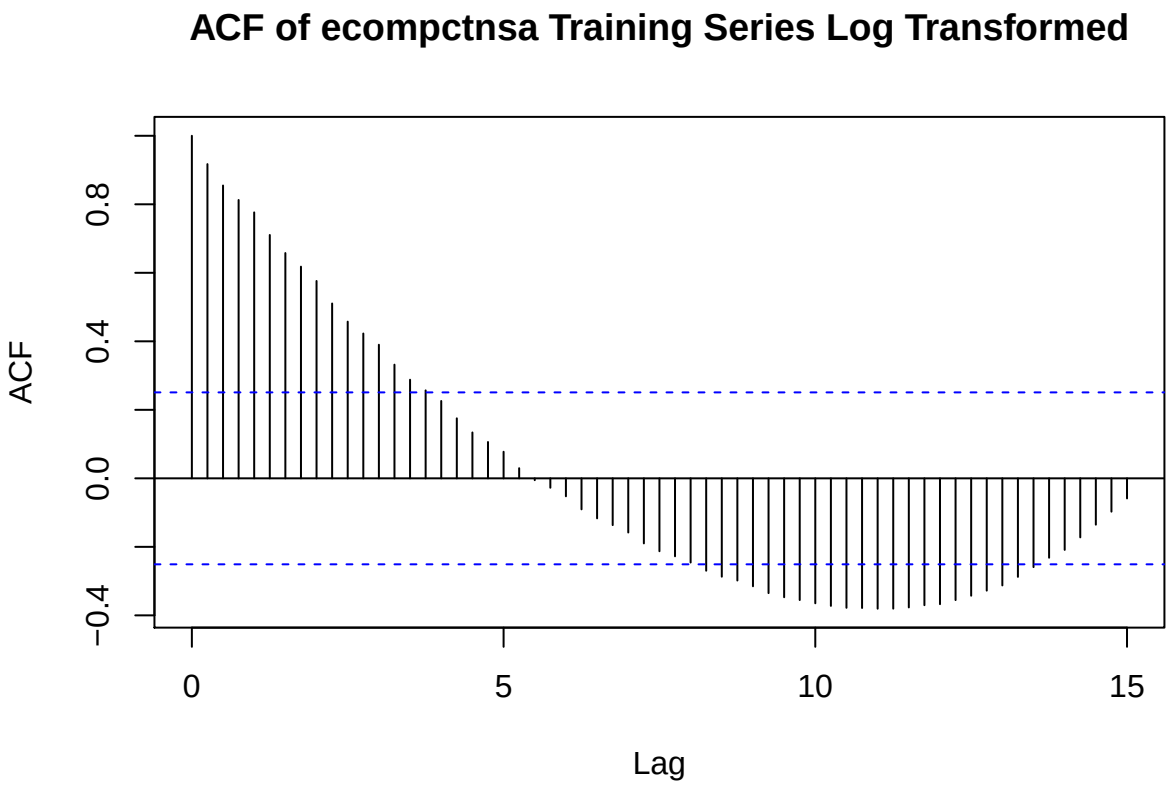
```
tail(cbind(time(fp.test), fp.test), 8)
```

```
##      time(fp.test)  fp.test
## [1,]       2015.00 1.931521
## [2,]       2015.25 1.887070
## [3,]       2015.50 1.916923
## [4,]       2015.75 2.163323
## [5,]       2016.00 2.041220
## [6,]       2016.25 2.014903
## [7,]       2016.50 2.041220
## [8,]       2016.75 2.251292
```

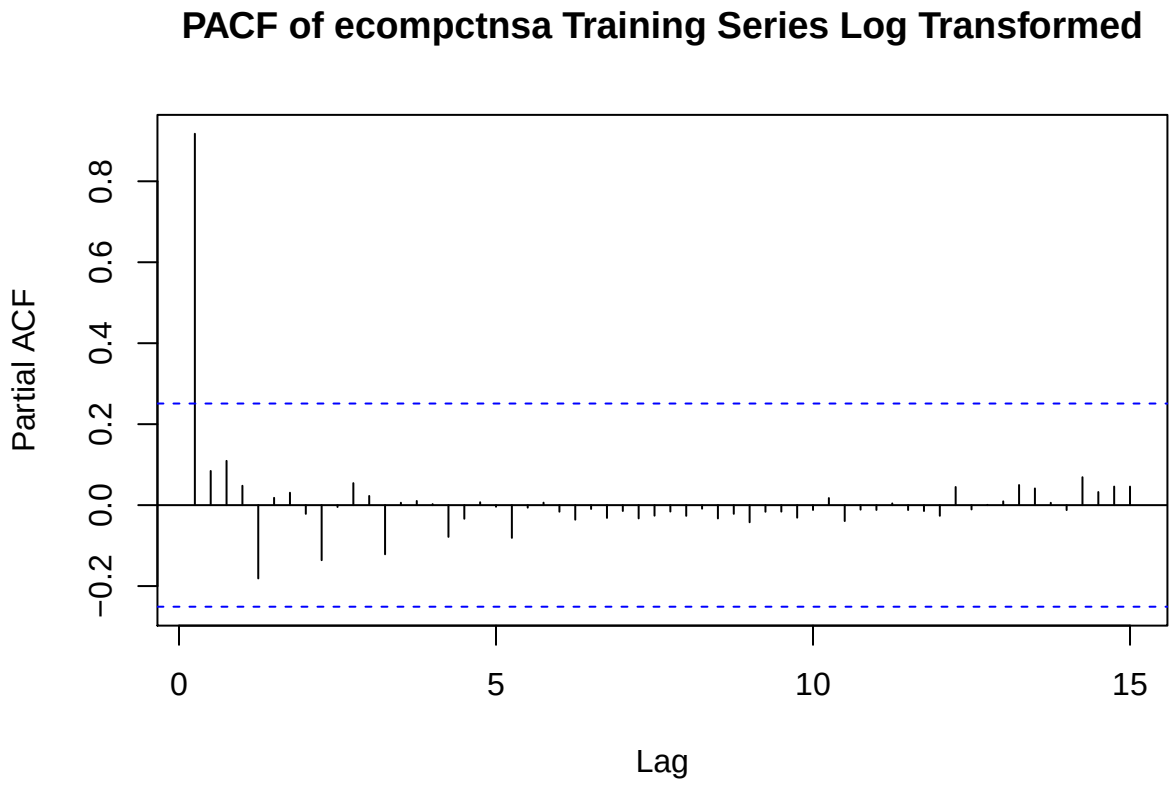
```
# Review time series plot for training data.
ts.plot(fp.training, ylab = "Log of ECOMPCTNSA", xlab = "Year",
    main = "Training Time Series")
```



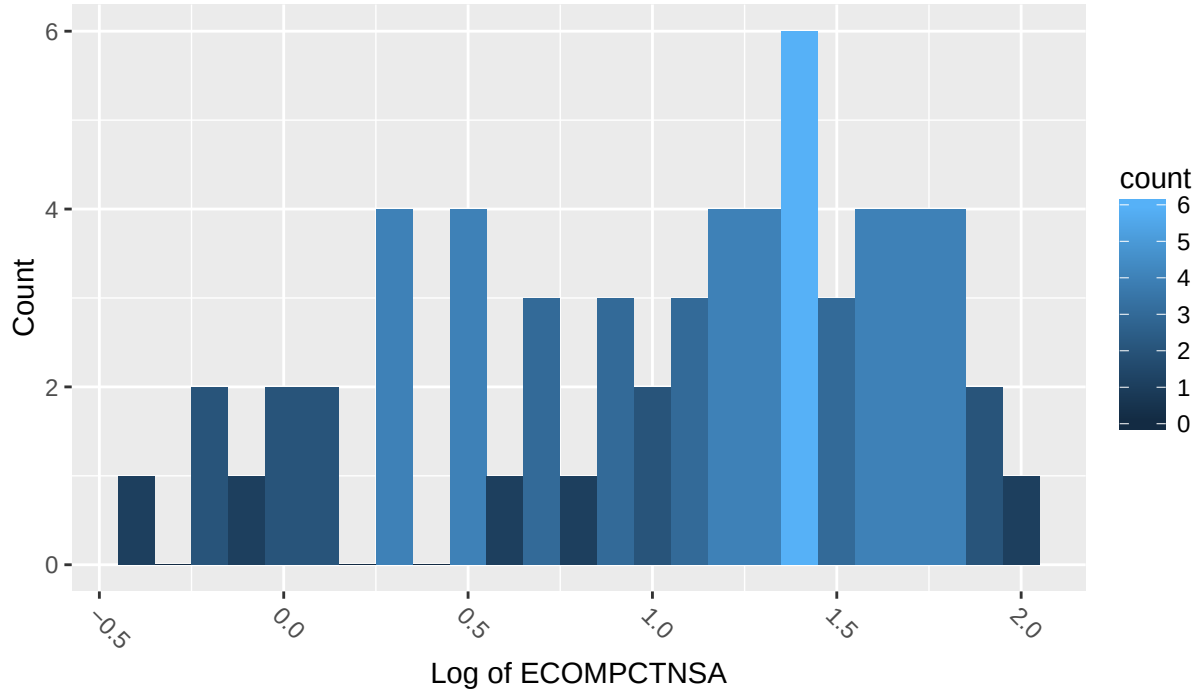

```
acf(fp.training, lag.max = 156, main = "ACF of ecompctnsa Training Series Log Transformed")
```




```
pacf(fp.training, lag.max = 156, main = "PACF of ecompctnsa Training Series Log Transformed")
```




```
ggplot(fp.training, aes(x = y)) + geom_histogram(aes(fill = ..count..),
    binwidth = 0.1) + ggtitle(expression(atop("Log Transform of ecompctnsa",
    atop(italic("Quarterly 1999-10-01 to 2014-10-01"), ""))) +
    xlab("Log of ECOMPCTNSA") + ylab("Count") + theme(axis.text.x = element_text(angle = -45,
    hjust = 0, vjust = 1), plot.title = element_text(size = 18,
    face = "bold", colour = "black", vjust = -1))
```



Model based on Graph Analysis and In-Sample Method

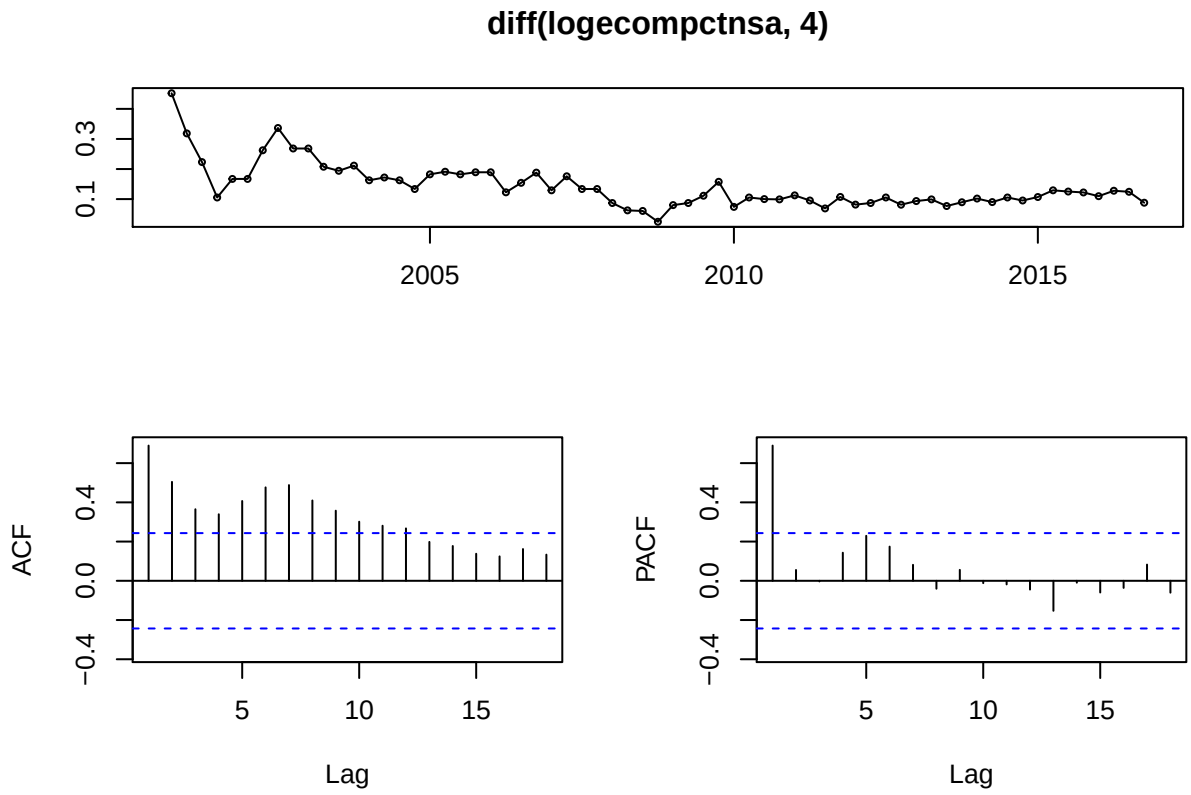
We create a model fit1.in based on visual examination from EDA and full sample set. We first need to create a stationary time series by detrending with differentiation. We end up needing to take both a seasonal [4] and a non-seasonal differentiation. From the resulting PACF we see a a seasonal AR(1) at lag 4 and from ACF a non-seasonal AR(1). This leads us to a model: ARIMA(0,1,0)(1,1,1)[4].

The model coefficients sar1 and sma1 are both significant. We record an AICc=-241.16, BIC=-235.08 and RMSE = 0.033.

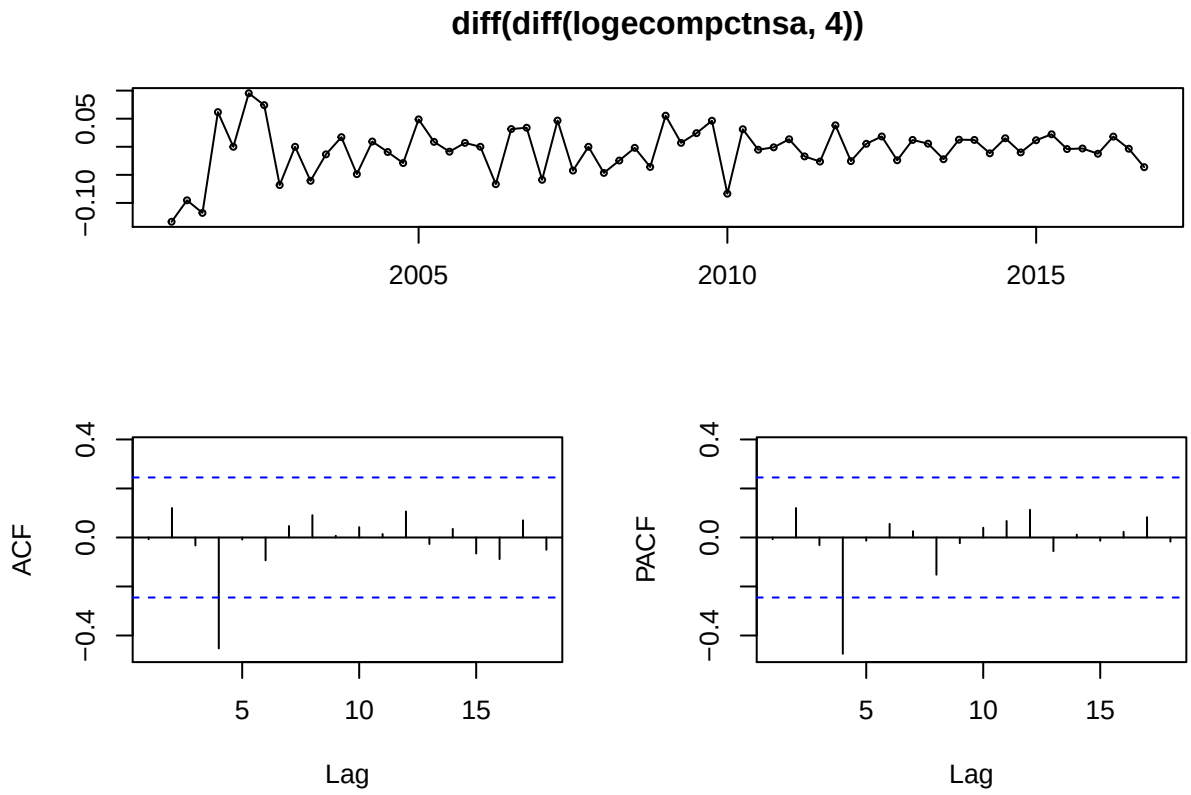
The Ljung-Box test shows us that we cannot reject the null hypothesis that the residual series comes from a white noise process (p-value = 0.1531). Also, the residual plots show characteristics of white noise with ACF being uncorrelated between time observations after the initial lag and non-significant oscillating PACF.

The histogram does not have a perfect normal distribution and a Shapiro-Wilk test gives p-value = 0.000929 < 0.05, which is statistically significant so the distribution is significantly different than the normal distribution. In this regression, we have a large sample size (> 30 data points), so we can use the central limit theorem (sample distribution of our coefficients is normal) and we don't worry about this.

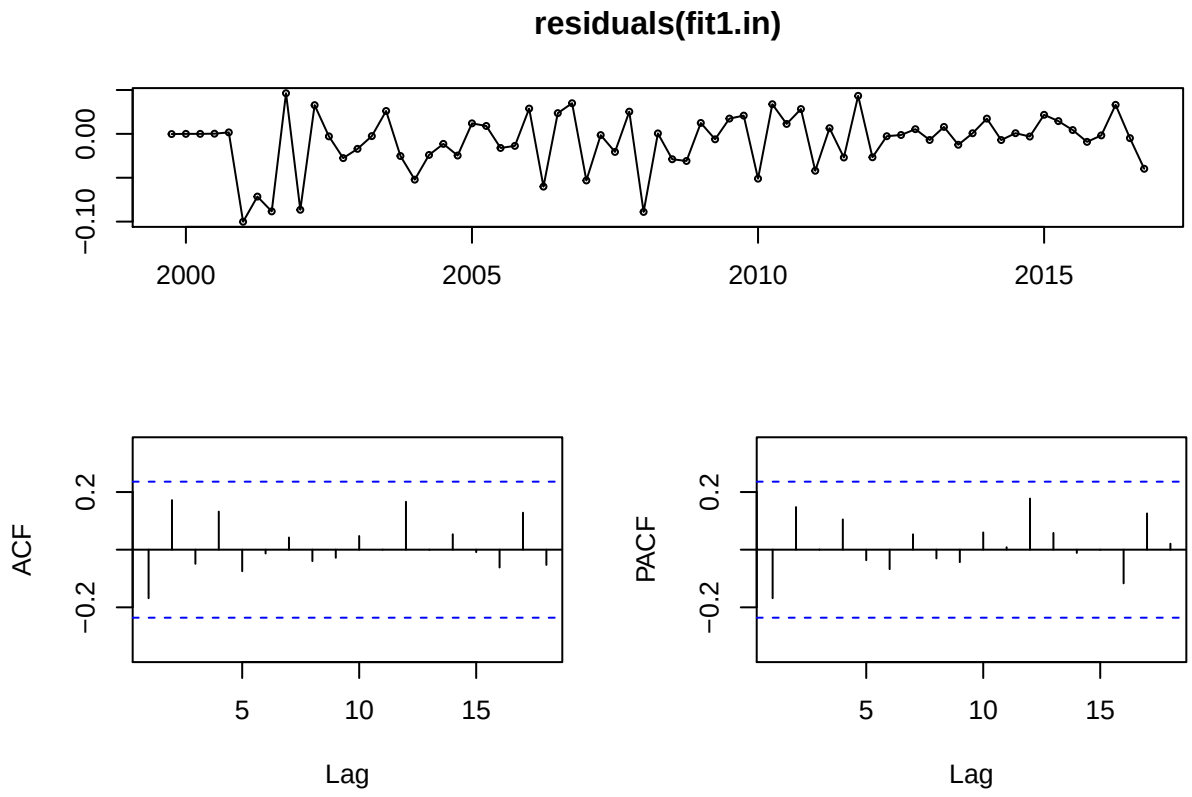
```
# Make seasonal differentiated plot at year level. We still
# have a trend remaining.
tsdisplay(diff(logecompctnsa, 4))
```




```
# As not detrended we also take a first non-seasonal diff.
# Both ACF and PACF show one spike at lag 4 so seasonal MA(1)
# and AR(1).  No non-seasonal spikes.
tsdisplay(diff(diff(logecompctnsa, 4)))
```




```
# We will use model ARIMA(0,1,0)(1,1,1)[4]
fit1.in <- Arima(logecompctnsa, order = c(0, 1, 0), seasonal = c(1,
    1, 1))
tsdisplay(residuals(fit1.in))
```



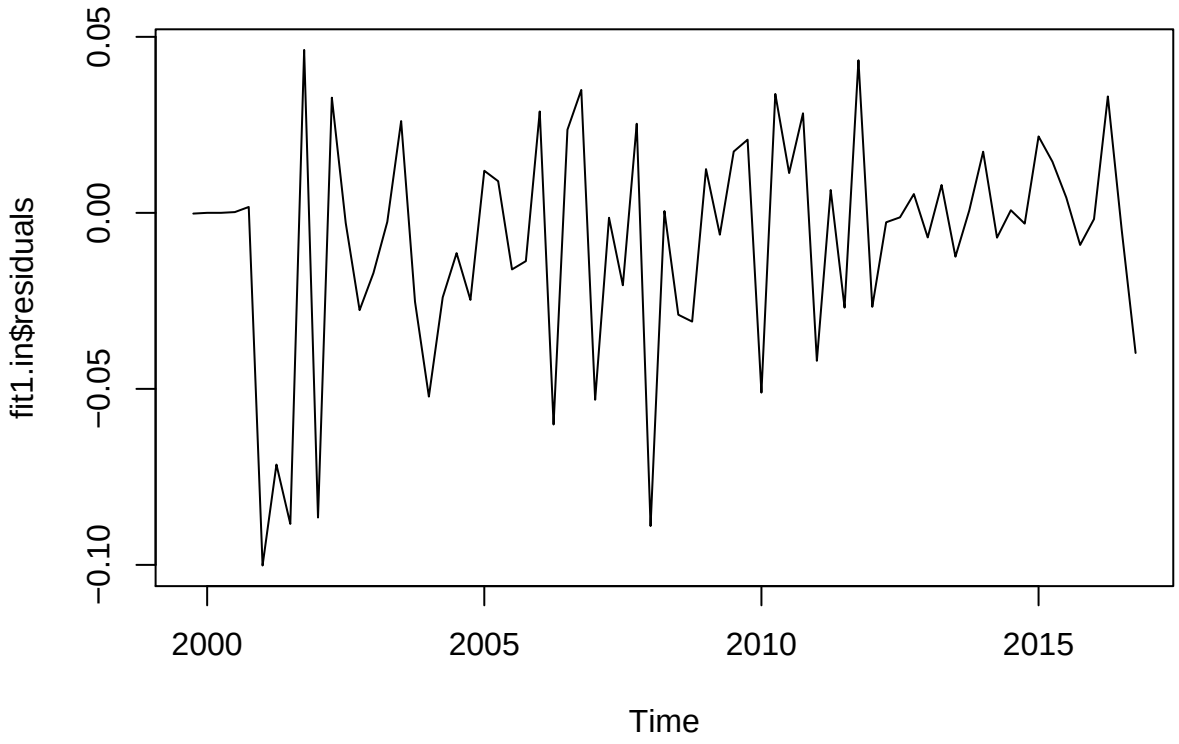

```
summary(fit1.in)  # BIC=-199.42
```

```
## Series: logecompctnsa
## ARIMA(0,1,0)(1,1,1)[4]
##
## Coefficients:
##          sar1     sma1
##       -0.5416  -0.1993
## s.e.   0.1773   0.1705
##
## sigma^2 estimated as 0.001217:  log likelihood=123.78
## AIC=-241.56   AICc=-241.16   BIC=-235.08
##
## Training set error measures:
##                       ME       RMSE        MAE  MPE MAPE      MASE
## Training set -0.008252534 0.03307048 0.02334702 -Inf  Inf 0.1639369
##                    ACF1
## Training set -0.1683233
```

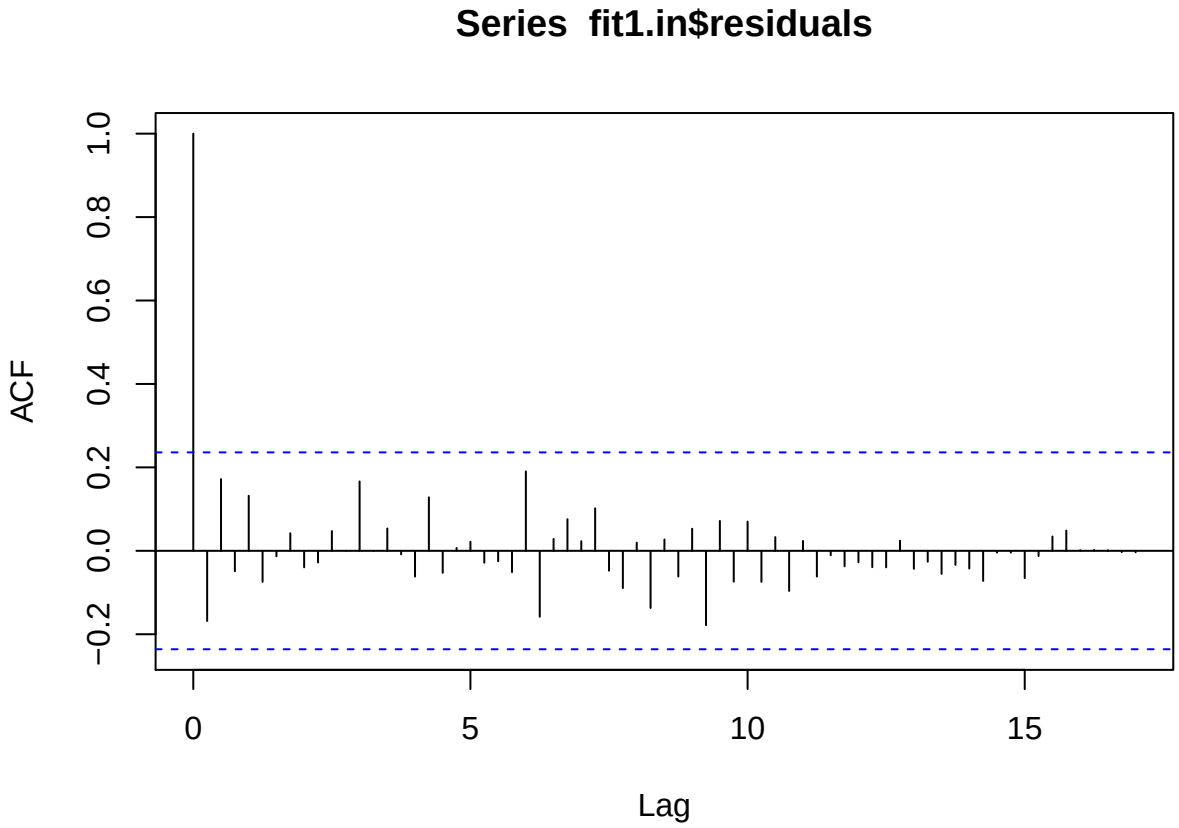
```
Box.test(fit1.in$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit1.in$residuals
## X-squared = 2.0412, df = 1, p-value = 0.1531
```

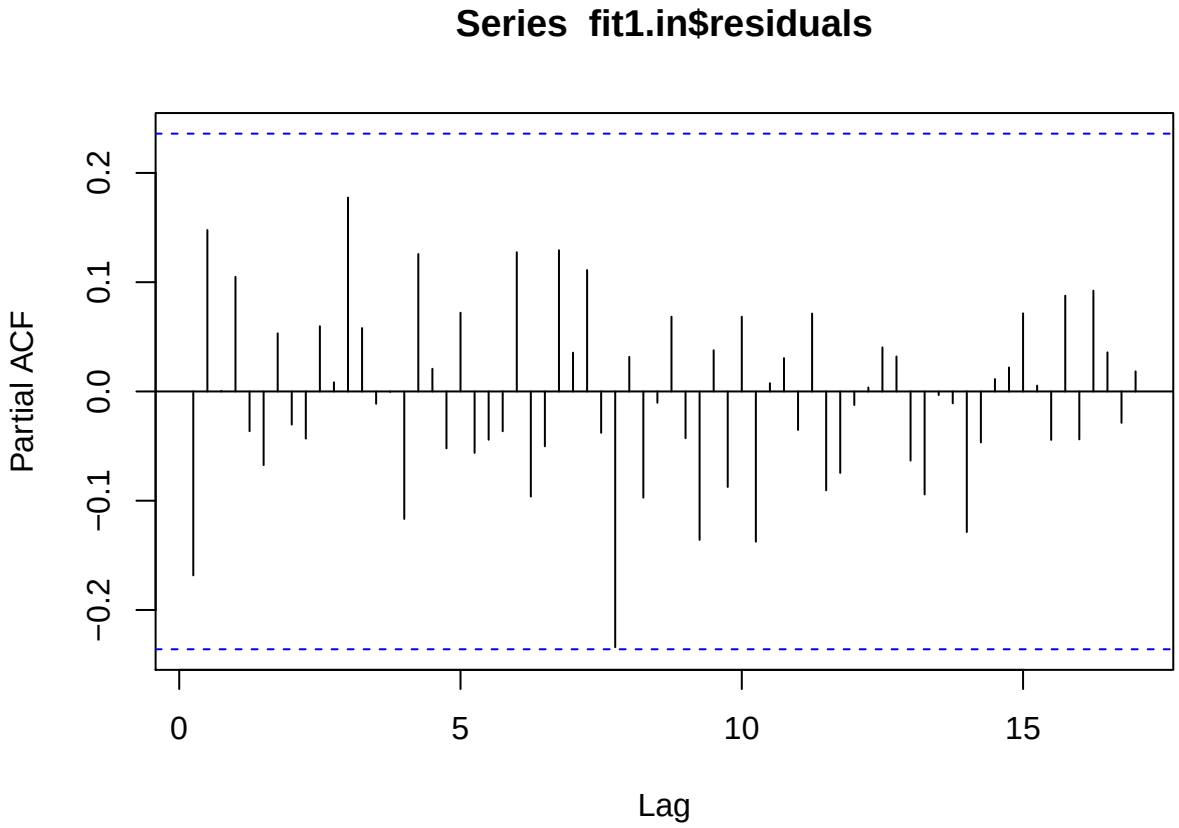
```r
# Residual plots not using tdisplay ToDo: Could be removed
# before submission as duplicate of above but gives larger
# graphs.
par(mar = c(4, 4, 4, 2))
plot(fit1.in$residuals)  # Has characteristics of white noise.
```



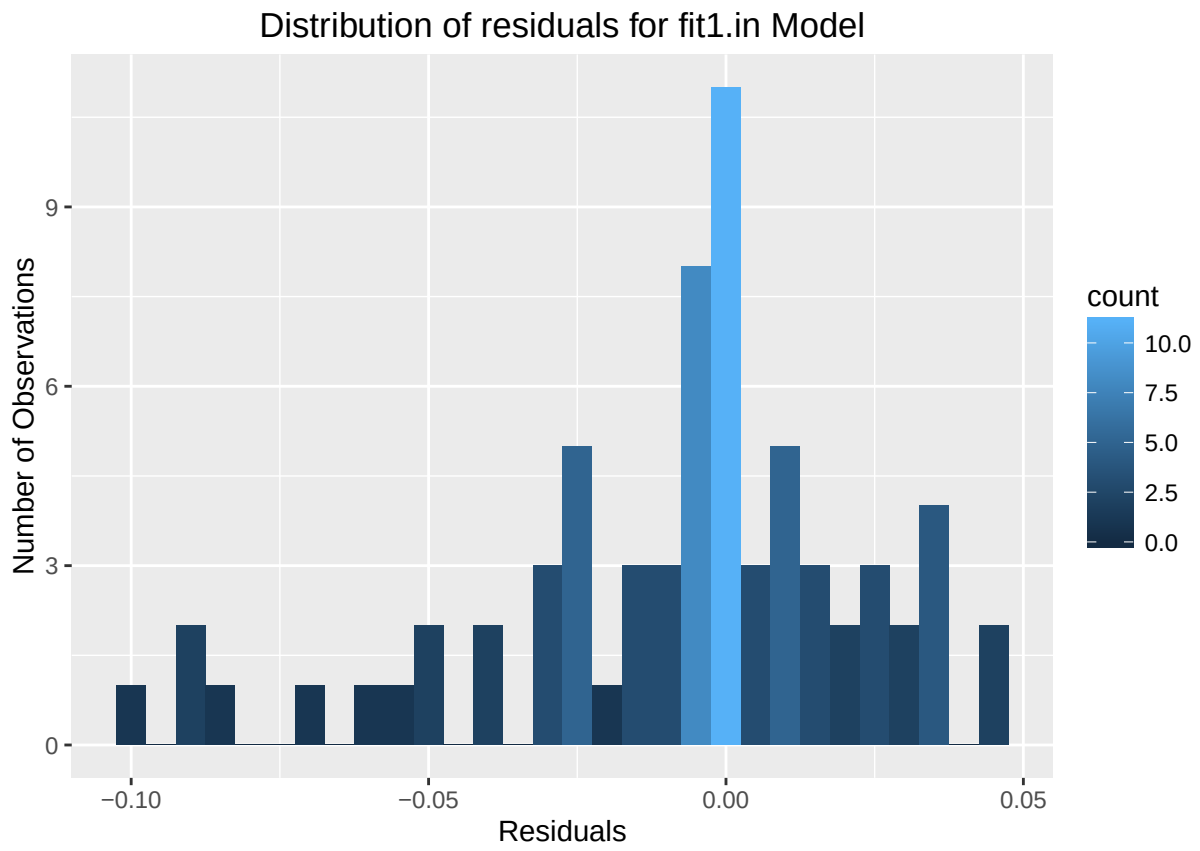

```r
acf(fit1.in$residuals, lag.max = 156)  # Uncorrelated after initial peak.
```




```
pacf(fit1.in$residuals, lag.max = 156)  # Oscillating pacf as expected for white noise
```

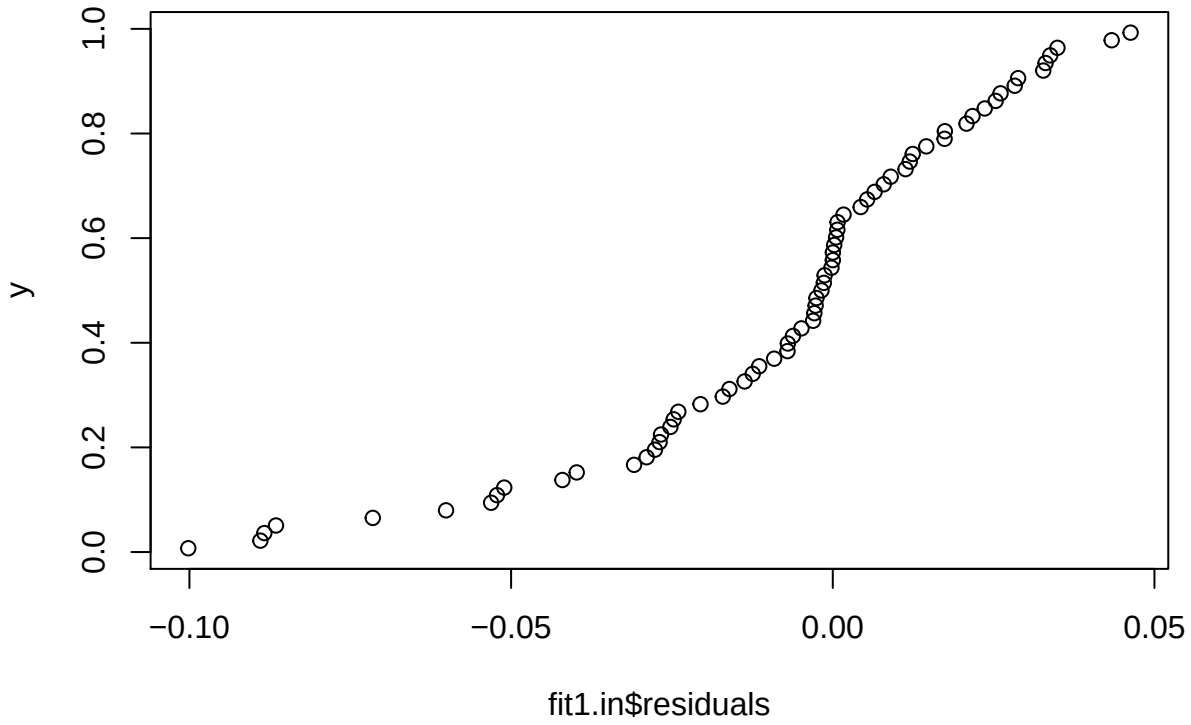



```
# Histogram of residuals. We don't have a perfect normal
# distribution.
ggplot(fit1.in$residuals, aes(x = y)) + geom_histogram(aes(fill = ..count..),
    binwidth = 0.005) + ggtitle("Distribution of residuals for fit1.in Model") +
    xlab("Residuals") + ylab("Number of Observations")
```




```
y <- qunif(ppoints(length(fit1.in$residuals)))

# Normal distribution test of residuals
qqplot(fit1.in$residuals, y)  # Not perfectly linear.
```




```
shapiro.test(fit1.in$residuals)  # Normal distribution test.
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit1.in$residuals
## W = 0.93116, p-value = 0.000929
```

Model based on Graph Analysis and Out-of-Sample Split Method

We create next a model fit1 using the split in training and test data and visual examination from EDA of the fp.training set. We arrive at the same model as when using the complete sample, i.e. ARIMA(0,1,0)(1,1,1)[4].

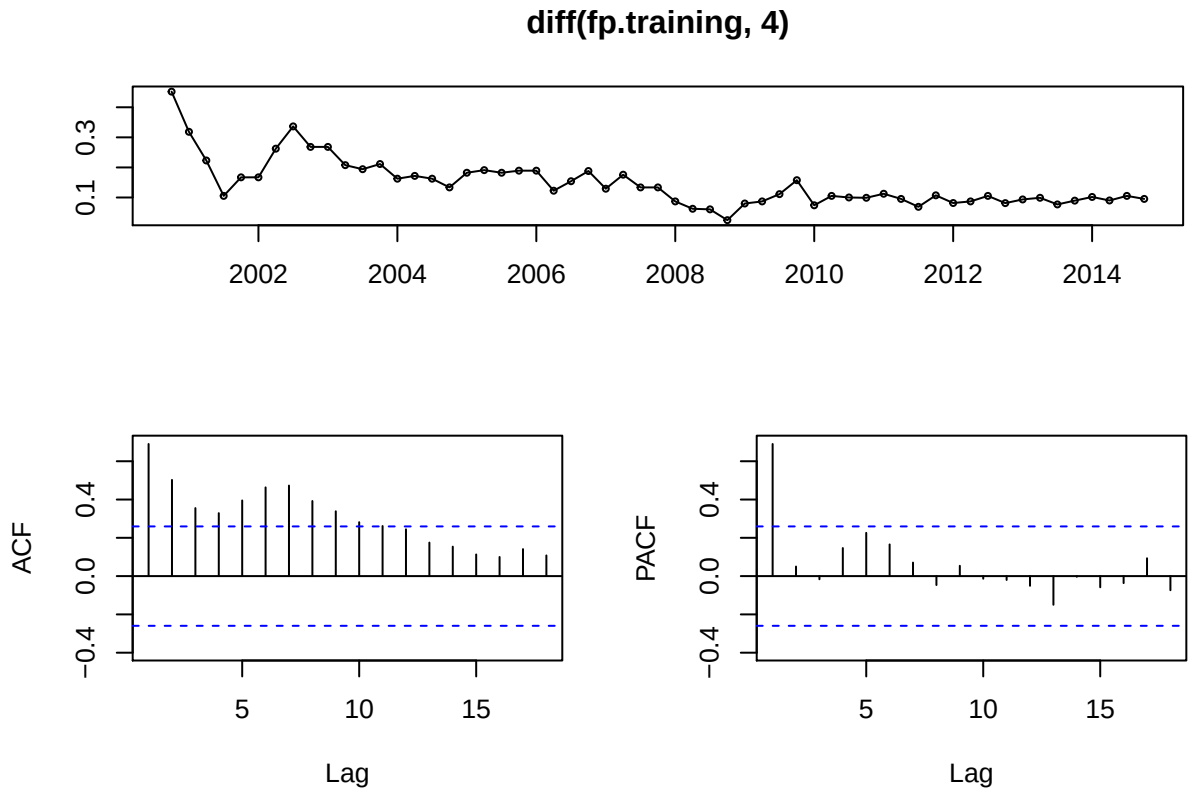
The model coefficients sar1 and sma1 are both significant. We record an AICc=-205.04 and BIC=-199.42.

The Ljung-Box test shows us that we cannot reject the null hypothesis that the residual series comes from a white noise process (p-value = 0.09831). Also, the residual plots show characteristics of white noise with ACF being uncorrelated between time observations after the initial lag and non-significant oscillating PACF.

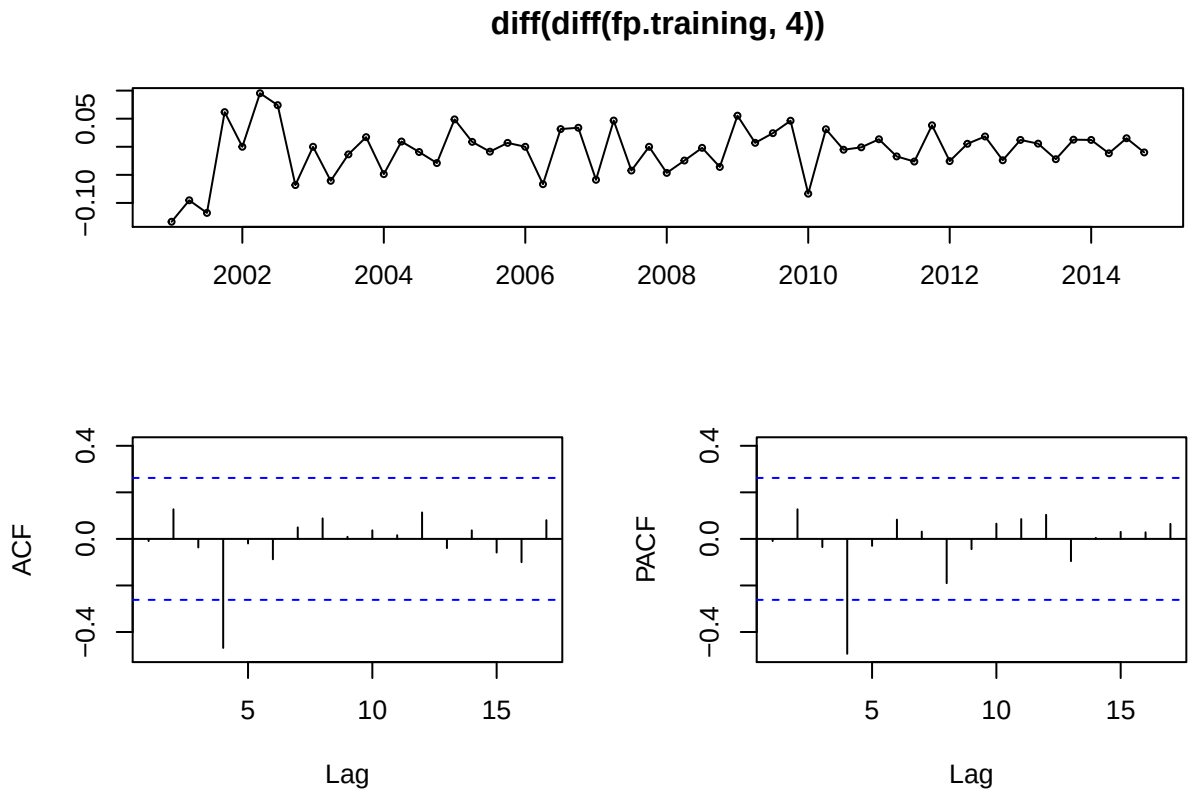
The histogram does not have a perfect normal distribution and a Shapiro-Wilk test gives p-value = 0.002509 < 0.05, which is statistically significant so the distribution is significantly different than the normal distribution. In this regression, we have a large sample size (> 30 data points), so we can use the central limit theorem (sample distribution of our coefficients is normal) and we don't worry about this.

Going forward we will use the out-of-sample assessment as the in-sample procedure is known to draw an overly optimistic picture of the model's forecasting ability, since common fitting algorithms (e.g. using squared error or likelihood criteria) tend to take pains to avoid large prediction errors, and are thus susceptible to overfitting - mistaking noise for signal in the data.

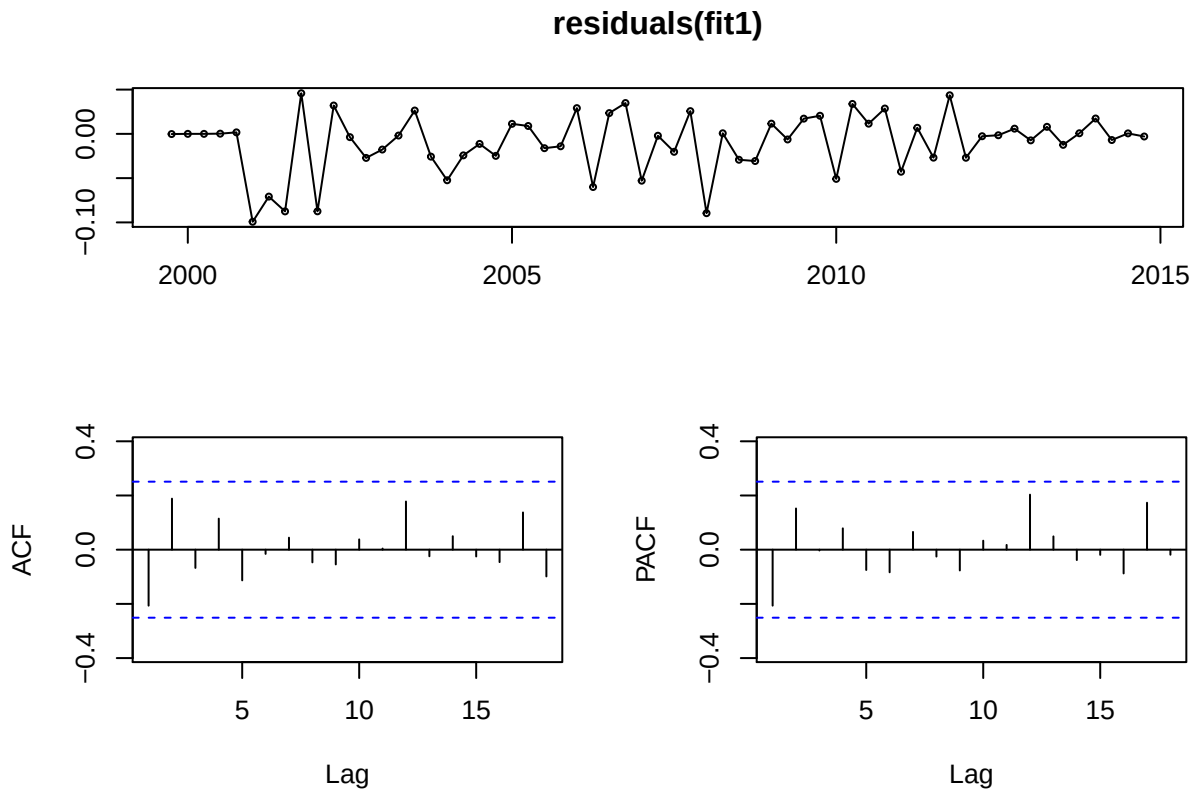
```r
# Make seasonal differentiated plot at year level. We still
# have a trend remaining.
tsdisplay(diff(fp.training, 4))
```




```r
# As not detrended we also take a first non-seasonal diff.
# Both ACF and PACF show one spike at lag 4 so seasonal MA(1)
# and AR(1). No non-seasonal spikes.
tsdisplay(diff(diff(fp.training, 4)))
```




```
# We will use model ARIMA(0,1,0)(1,1,1)[4]
fit1 <- Arima(fp.training, order = c(0, 1, 0), seasonal = c(1,
    1, 1))
tsdisplay(residuals(fit1))
```



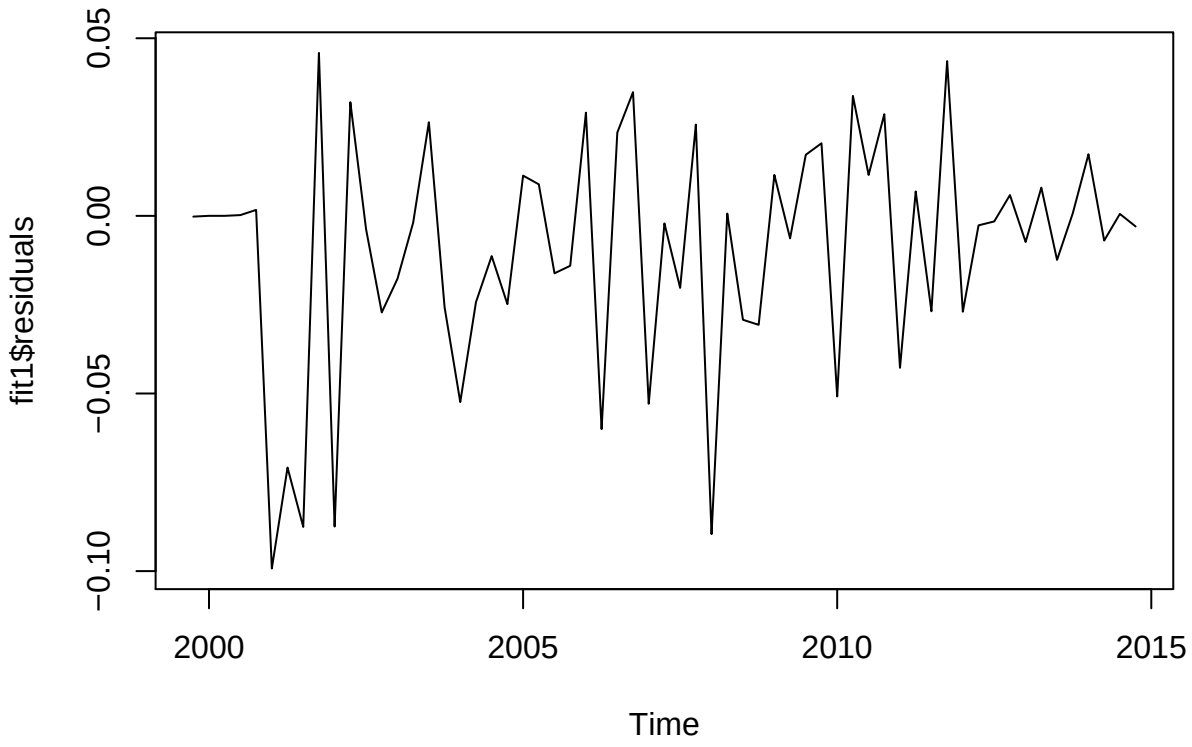

```
summary(fit1)  # BIC=-199.42
```

```
## Series: fp.training
## ARIMA(0,1,0)(1,1,1)[4]
##
## Coefficients:
##          sar1     sma1
##       -0.5480  -0.2046
## s.e.   0.1816   0.1730
##
## sigma^2 estimated as 0.001331:  log likelihood=105.75
## AIC=-205.5   AICc=-205.04   BIC=-199.42
##
## Training set error measures:
##                       ME       RMSE        MAE  MPE MAPE     MASE
## Training set -0.009692995 0.03432324 0.02431327 -Inf  Inf 0.166495
##                    ACF1
## Training set -0.2065581
```

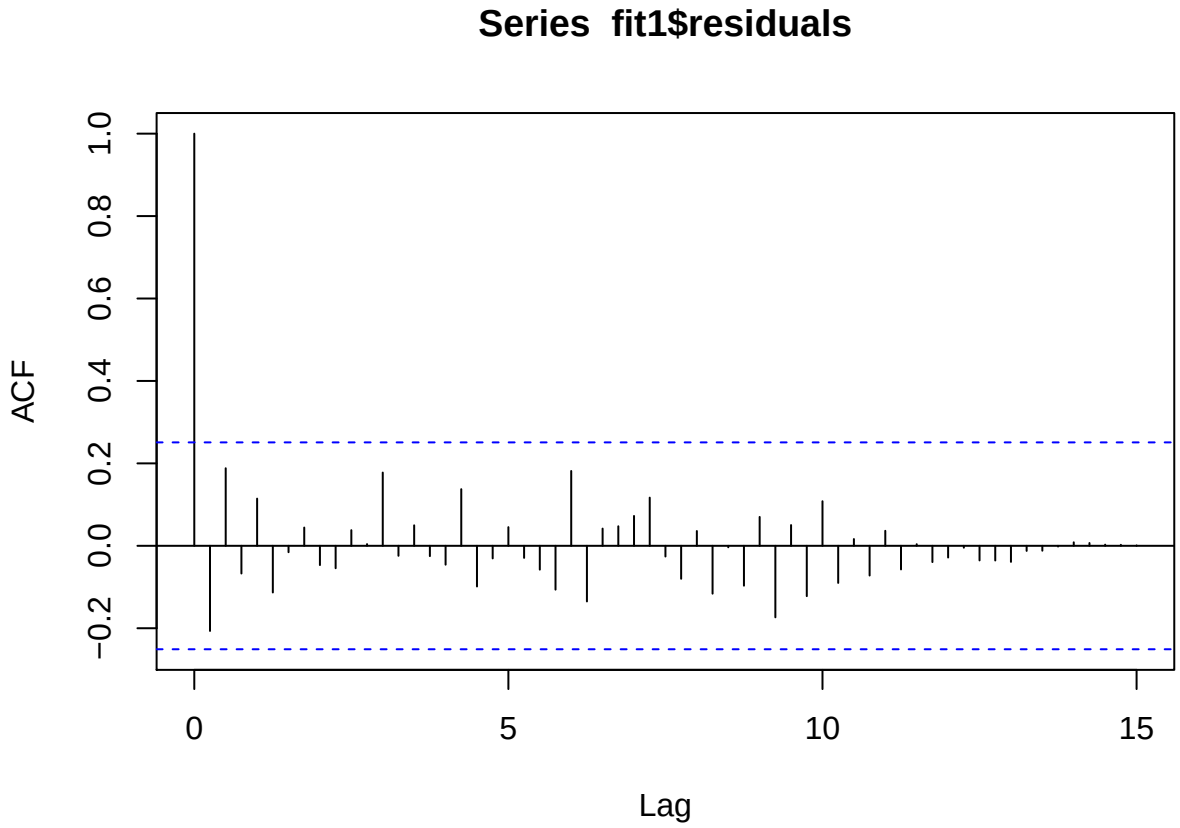
```
Box.test(fit1$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit1$residuals
## X-squared = 2.7328, df = 1, p-value = 0.09831
```

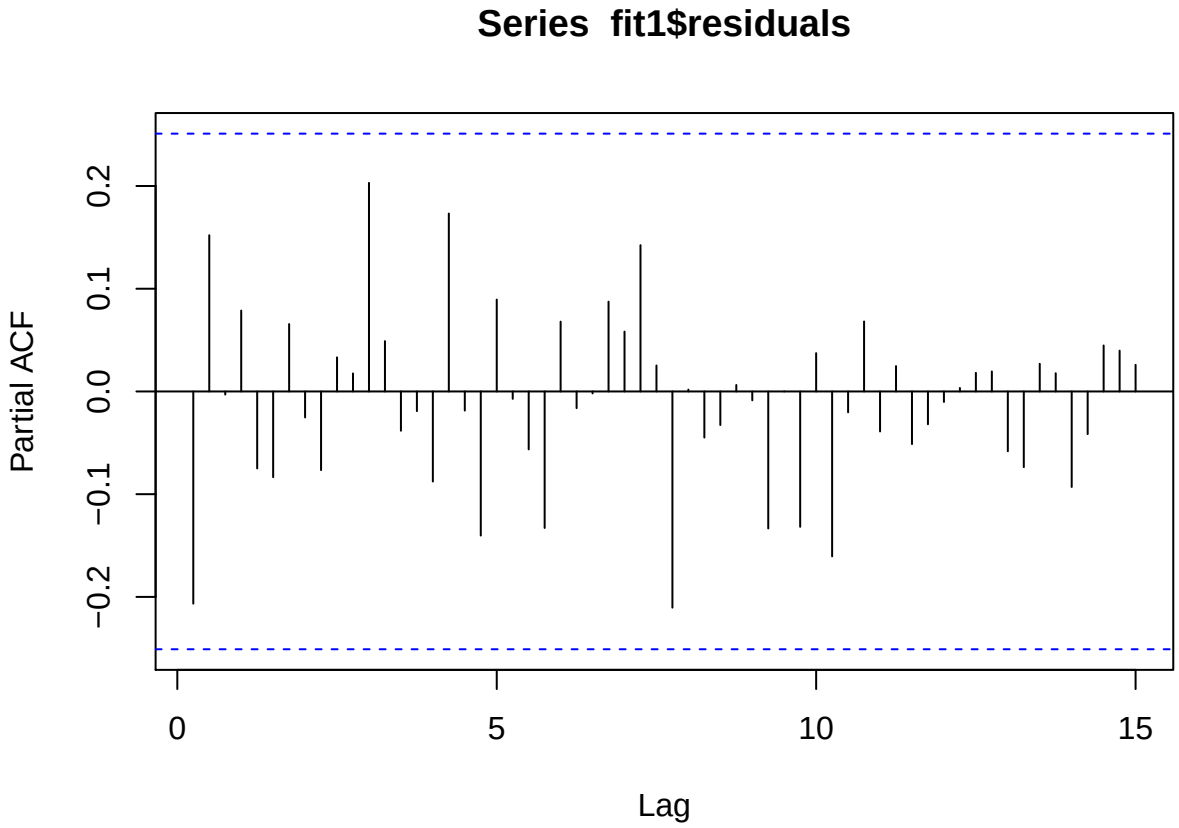
```r
# Residual plots not using tdisplay ToDo: Could be removed
# before submission as duplicate of above but gives larger
# graphs.
par(mar = c(4, 4, 4, 2))
plot(fit1$residuals)  # Has characteristics of white noise.
```



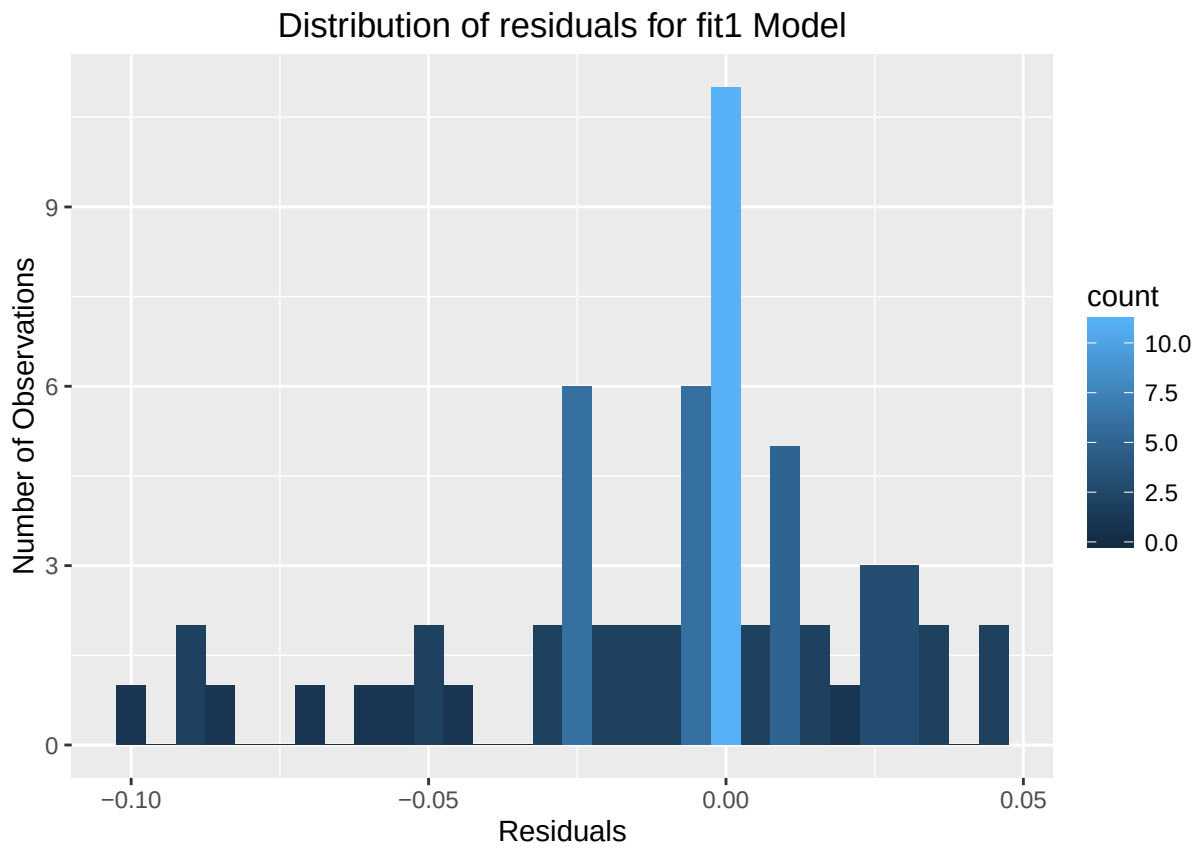

```r
acf(fit1$residuals, lag.max = 156)  # Uncorrelated after initial peak.
```




```
pacf(fit1$residuals, lag.max = 156)  # Oscillating pacf as expected for white noise
```

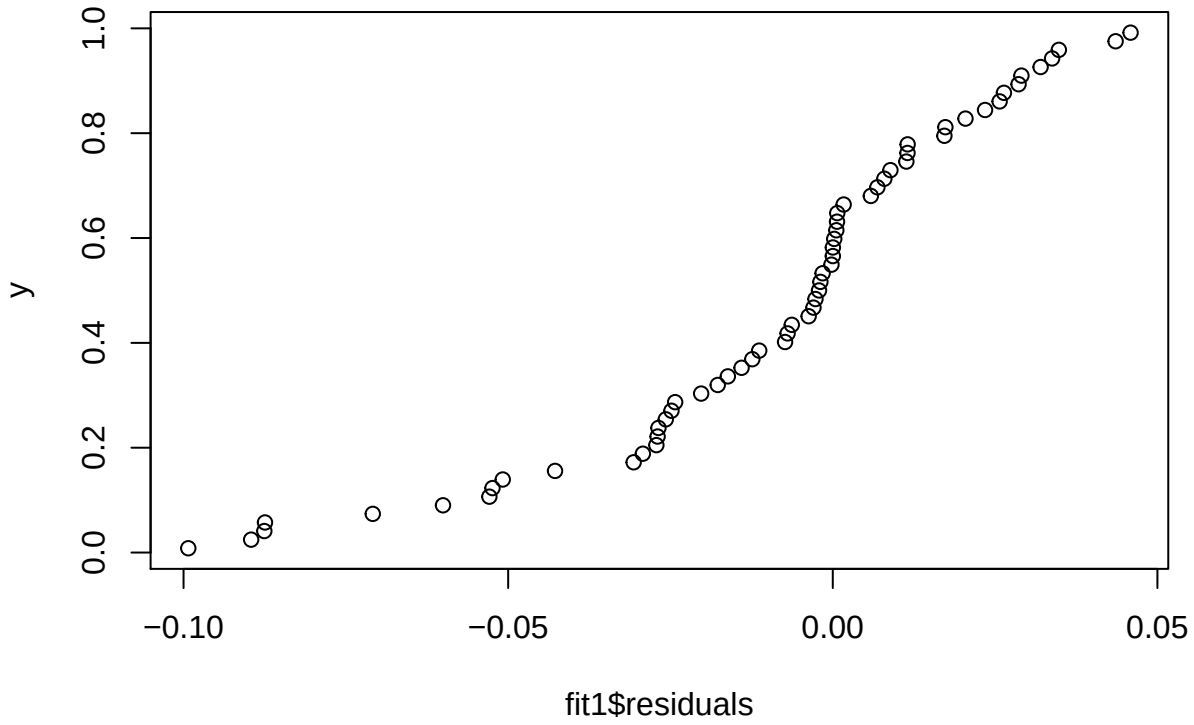



```
# Histogram of residuals
ggplot(fit1$residuals, aes(x = y)) + geom_histogram(aes(fill = ..count..),
    binwidth = 0.005) + ggtitle("Distribution of residuals for fit1 Model") +
    xlab("Residuals") + ylab("Number of Observations")  # Not perfect normal distr.
```




```
y <- qunif(ppoints(length(fit1$residuals)))

# Normal distribution test of residuals
qqplot(fit1$residuals, y)  # Not perfectly linear.
```




```
shapiro.test(fit1$residuals)  # Normal distribution test.
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit1$residuals
## W = 0.93335, p-value = 0.002509
```

Reviewing the forecast error using the Root-Mean-Square-Error (RMSE), which is 0.045 for the out-of-sample forecast test observations and 0.034 for the training set. With the accuracy function we also record the Mean Error (ME) for the out-of-sample test set to be 0.0416 and the Mean Absolute Percentage Error (MAPE) = 2.061. Plotting the forecast alongside actual test sample values shows good match even if forecast seems to over-estimate the changes.

```
# ME, RMSE and MAPE
accuracy(forecast.Arima(fit1, h = 8), fp.test)
```

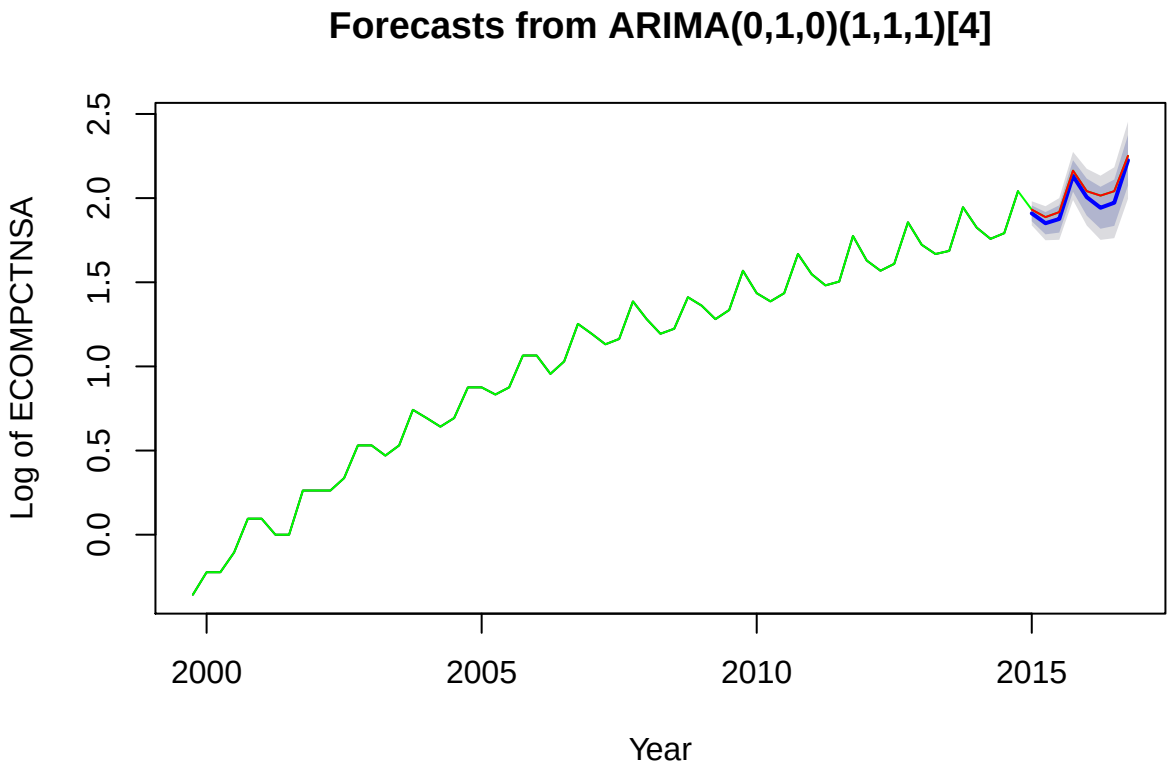
```
##                       ME       RMSE        MAE      MPE     MAPE
## Training set -0.009692995 0.03432324 0.02431327     -Inf      Inf
## Test set      0.041624846 0.04515920 0.04162485 2.060984 2.060984
##                    MASE       ACF1 Theil's U
## Training set 0.1664950 -0.2065581        NA
## Test set     0.2850431  0.1646683 0.3538649
```

```
# Alternative way to calculate RMSE using own common function
# defined on top
fcast.fit1.out <- forecast.Arima(fit1, h = 8)
rmse.fit1.out <- calculate_rmse(fcast.fit1.out$mean, fp.test)
```

```
rmse.fit1.out
```

```
## [1] 0.0451592
```

```
# Plot forecast and compare with out-of-sample observed
# values.
plot(forecast.Arima(fit1, h = 8), ylab = "Log of ECOMPCTNSA",
    xlab = "Year")
par(new = TRUE)
lines(logecompctnsa, col = "green")
par(new = TRUE)
lines(fp.test, col = "red")
```



Model based on Iterative Algorithm Selection

We attempt to find an alternative model using an iterative approach across both seasonal and non-seasonal components and see if can get lower BIC/AICc, lower Forecast Error as well as more normally distributed residuals. As we assess the models we do review both BIC and AICc. We give slight extra weight to AICc as small sample. We get the following contenders:

p	d	q	P	D	Q	AICc	BIC	RMSE	Comments
2	1	0	1	0	0	-187.336	-179.686	0.021	lowest MRSE, high AICc and BIC
0	1	0	2	0	0	-208.775	-202.921	0.030	selected as fit2 on AICc and RMSE
0	1	0	1	1	0	-205.992	-202.168	0.032	
0	1	0	2	1	0	-206.038	-200.424	0.035	simple auto.arima selection

p	d	q	P	D	Q	AICc	BIC	RMSE	Comments
0	1	0	1	0	1	-204.180	-198.326	0.027	good contender given low MRSE
0	1	1	2	0	1	-205.535	-196.174	0.032	
0	1	0	0	1	1	-201.312	-197.487	0.029	
0	1	0	1	1	1	-205.037	-199.423	0.033	fit1 from initial graph analysis
0	1	0	1	0	2	-209.389	-201.739	0.038	
0	1	0	0	1	2	-206.759	-201.144	0.040	auto.arima (stepwise = FALSE)
0	1	0	2	0	2	-209.592	-200.232	0.034	

```
# Algorithm to validate models in an iterative approach.  for
# (Q in 0:2){ for (D in 0:1){ for(P in 0:2){ for (q in 0:2){
# for(d in 1){ for(p in 0:2){ mod <- Arima(fp.training, order
# = c(p,d,q), seasonal = list(order = c(P,D,Q), 4), method =
# 'ML') fcast <- forecast.Arima(mod, h = 4) rmse <-
# calculate_rmse(fcast$mean, fp.test) print(round(c(p, d, q,
# P, D, Q, mod$aicc, mod$bic, rmse),3)) } } } } } }
```

We select to go forward with ARIMA(0,1,0)(2,0,0)[4] as our second model fit2 based on its AICc = -208.775, BIC = -202.921 and RMSE = 0.030. This has a good combination of low AICs/BIC and RMSE.It is also a rather simple model which is preferred for easier interpretation.

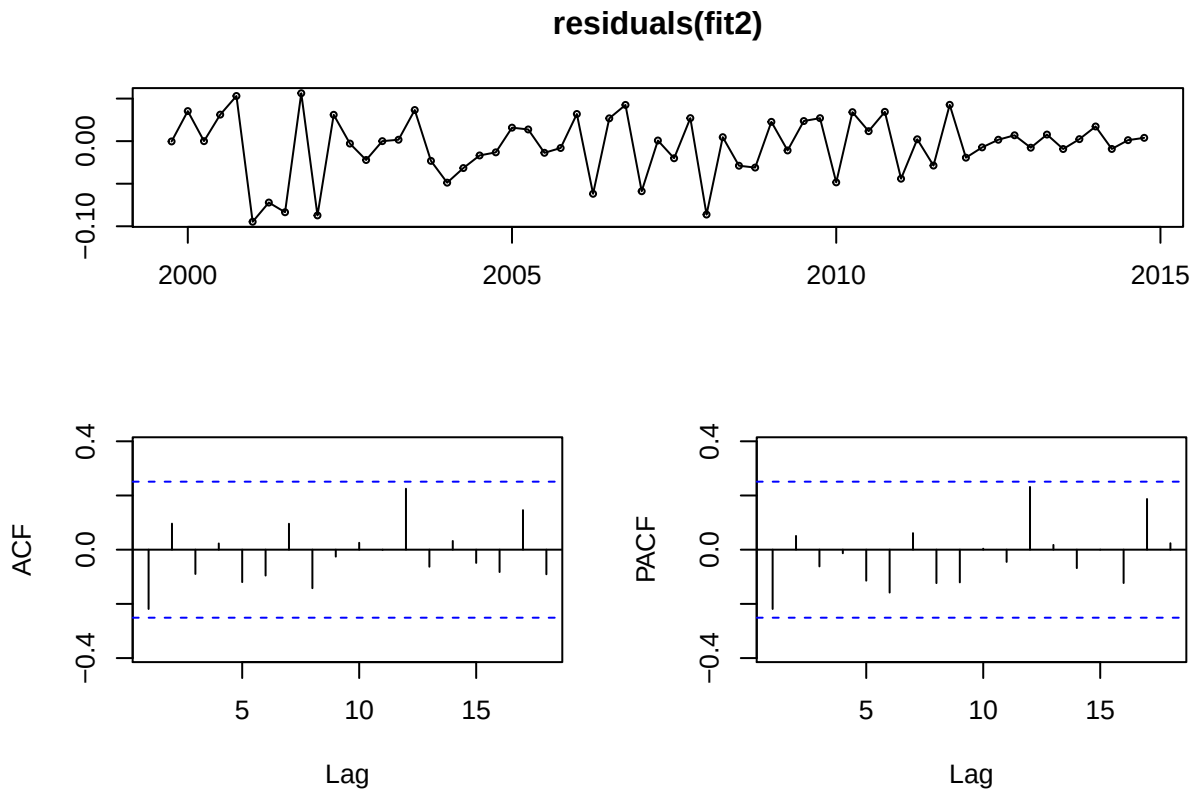
The model coefficients sar1 and sma1 are both significant.

The Ljung-Box test shows us that we cannot reject the null hypothesis that the residual series comes from a white noise process (p-value = 0.0801). Also, the residual plots show characteristics of white noise with ACF being uncorrelated between time observations after the initial lag and non-significant oscillating PACF.

The histogram does not have a perfect normal distribution and a Shapiro-Wilk test gives p-value = 0.01425 > 0.05, which is statistically non-significant so the distribution is close to normal distribution.

The histogram does not have a perfect normal distribution and a Shapiro-Wilk test gives a higher p-value = 0.01425 but it is still smaller than 0.05 critera, so the distribution is significantly different than the normal distribution. As before in this regression, we have a large sample size (> 30 data points), so we can use the central limit theorem (sample distribution of our coefficients is normal) and we don't worry about this.

```
# Fit model 2 for selected ARIMA(0,1,0)(2,0,0)[4]
fit2 <- Arima(fp.training, order = c(0, 1, 0), seasonal = c(2,
    0, 0))
tsdisplay(residuals(fit2))
```



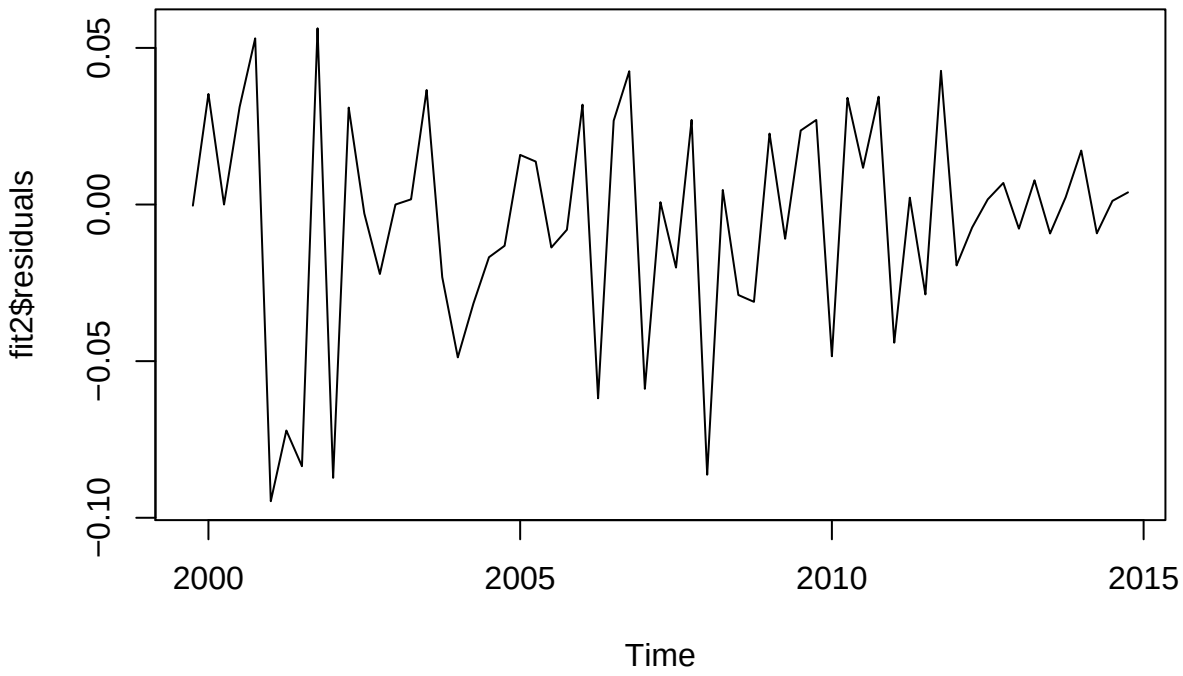

```
summary(fit2)
```

```
## Series: fp.training
## ARIMA(0,1,0)(2,0,0)[4]
##
## Coefficients:
##         sar1    sar2
##       0.3247  0.6535
## s.e.  0.1202  0.1221
##
## sigma^2 estimated as 0.001353:  log likelihood=107.6
## AIC=-209.2   AICc=-208.78   BIC=-202.92
##
## Training set error measures:
##                       ME       RMSE       MAE  MPE MAPE      MASE
## Training set -0.005625774 0.03587087 0.0268497 -Inf  Inf 0.1838643
##                    ACF1
## Training set -0.2186788
```

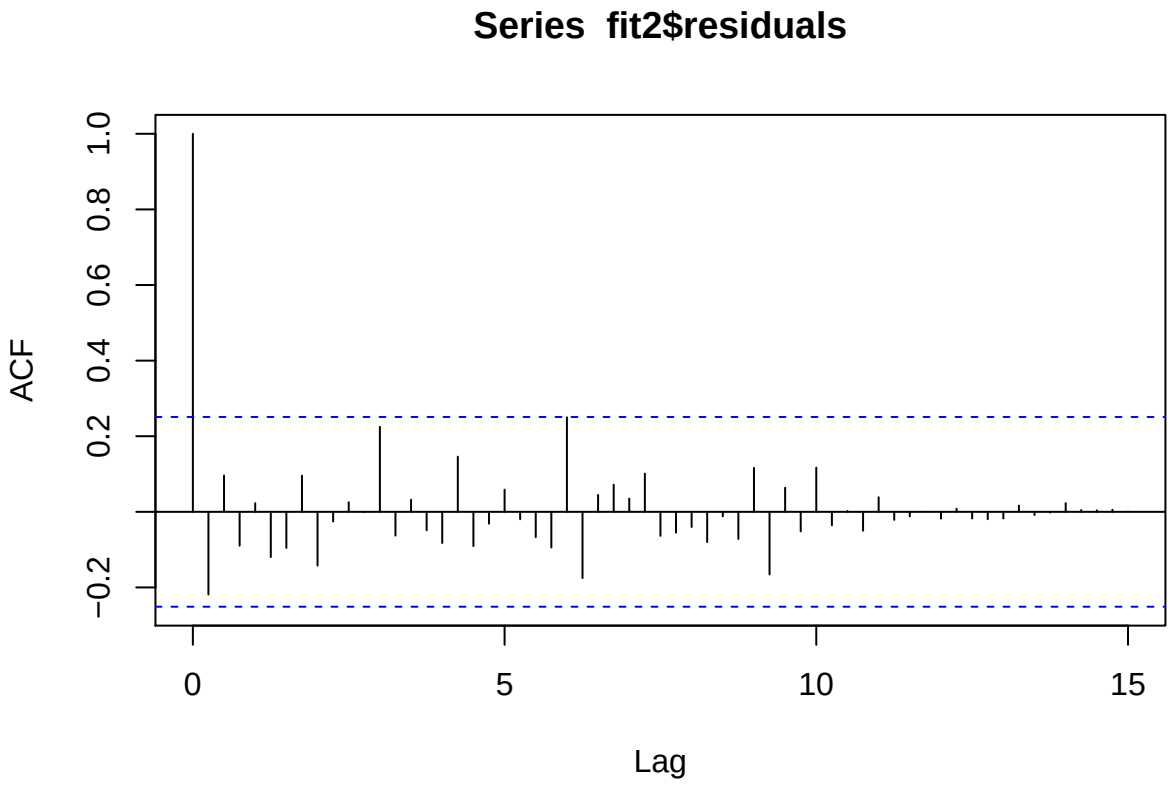
```
Box.test(fit2$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  fit2$residuals
## X-squared = 3.0629, df = 1, p-value = 0.0801
```

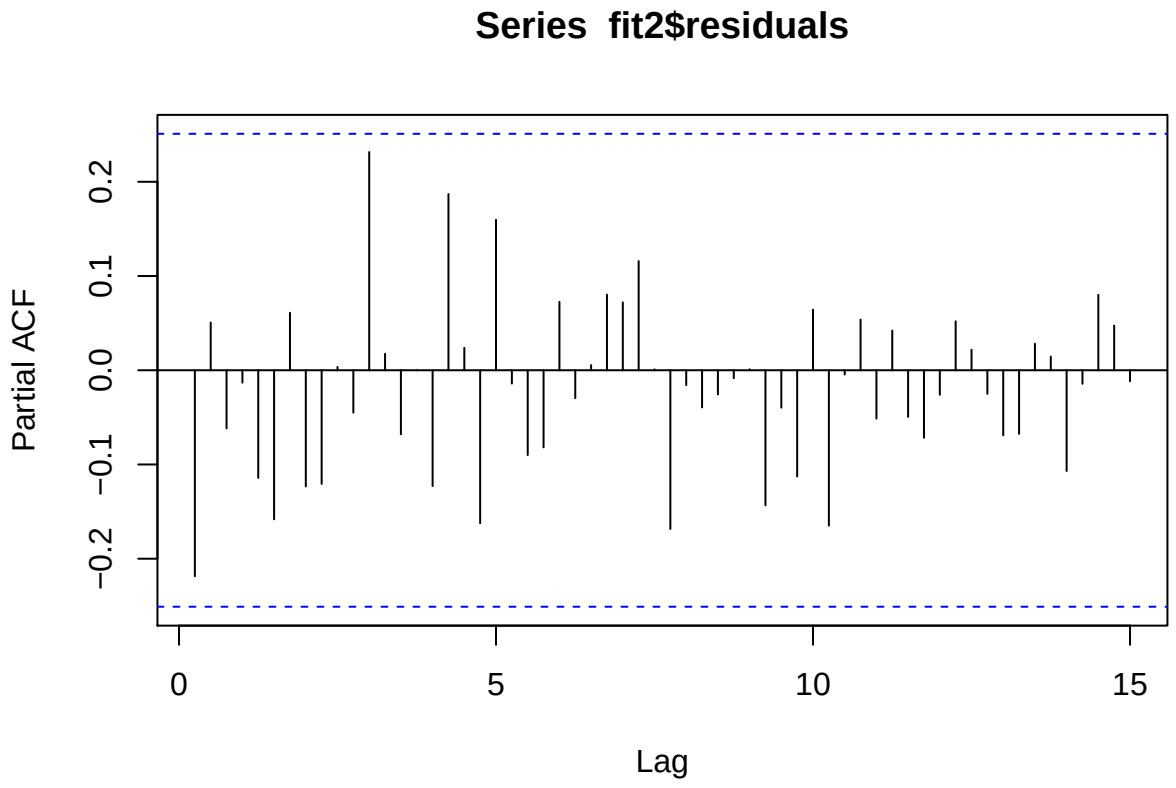
```r
# Residual plots not using tdisplay
plot(fit2$residuals)  # Has characteristics of white noise.
```




```r
acf(fit2$residuals, lag.max = 156)  # Uncorrelated after initial peak.
```



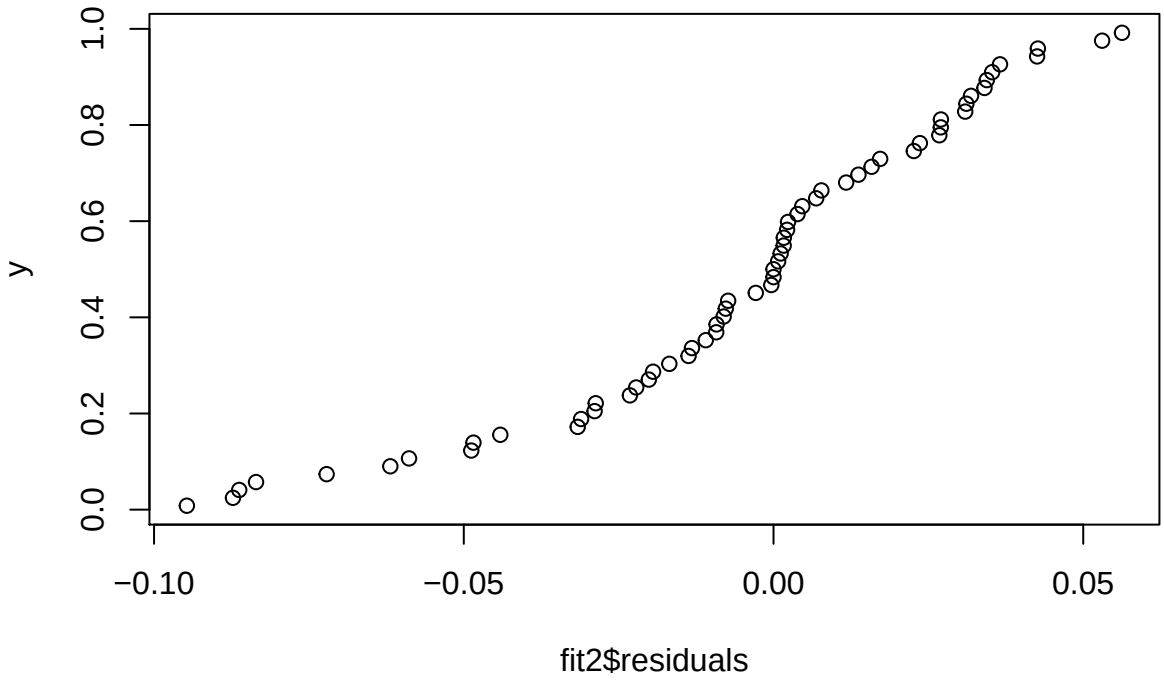

```
pacf(fit2$residuals, lag.max = 156)  # Oscillating pacf as expected for white noise
```




```
# Histogram of residuals
ggplot(fit2$residuals, aes(x = y)) + geom_histogram(aes(fill = ..count..),
    binwidth = 0.005) + ggtitle("Distribution of residuals for fit2 Model") +
    xlab("Residuals") + ylab("Number of Observations")  # Not a perfect normal distr.
```


```
y <- qunif(ppoints(length(fit2$residuals)))

# Normal distribution test of residuals
qqplot(fit2$residuals, y)  # Not perfectly linear.
```




```
shapiro.test(fit2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit2$residuals
## W = 0.94989, p-value = 0.01425
```

Reviewing the forecast error using the Root-Mean-Square-Error (RMSE), which is slightly 0.042 (lower than fit1 model's 0.045) for the out-of-sample test, while 0.036 (> 0.034 fit1 model) for the training set. We put more emphasis on the test RMSE.

With the accuracy function we also record the Mean Error (ME) for the out-of-sample test set to be 0.038 (lower than 0.042 for fit1) and the Mean Absolute Percentage Error (MAPE) = 1.893 (lower than 2.061 for fit1 model). Plotting the forecast alongside actual test sample values and it shows similar behavior as fit1 model.

```
# ME, RMSE and MAPE
accuracy(forecast.Arima(fit2, h = 8), fp.test)
```

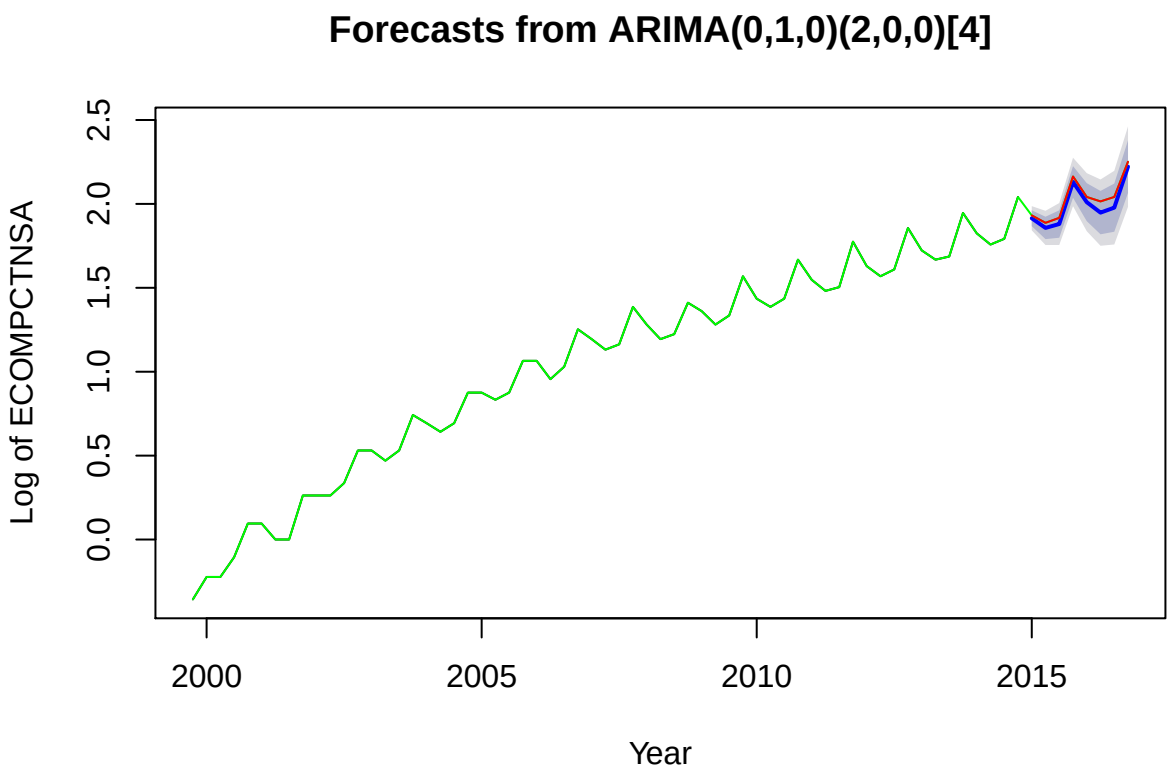
```
##                       ME       RMSE        MAE     MPE    MAPE      MASE
## Training set -0.005625774 0.03587087 0.02684970    -Inf     Inf 0.1838643
## Test set      0.038384784 0.04173597 0.03838478 1.89346 1.89346 0.2628555
##                    ACF1 Theil's U
## Training set -0.2186788        NA
## Test set      0.2402407 0.3285158
```

```
# Alternative way to calculate RMSE using own common function
# defined on top
```

```
fcast.fit2.out <- forecast.Arima(fit2, h = 8)
rmse.fit2.out <- calculate_rmse(fcast.fit2.out$mean, fp.test)
rmse.fit2.out
```

```
## [1] 0.04173597
```

```
# Plot forecast and compare with out-of-sample observed
# values.
plot(forecast.Arima(fit2, h = 8), ylab = "Log of ECOMPCTNSA",
    xlab = "Year")
par(new = TRUE)
lines(logecompctnsa, col = "green")
par(new = TRUE)
lines(fp.test, col = "red")
```



Comparison of the Two Models

When doing a Diebold-Mariano test of forecast accuracy comparison between fit1 and fit2, it is not significant (p-value = 0.123) so the alternative hypothesis that fit1 and fit2 have different levels of accuracy cannot be confirmed. If anything, the indication is that fit1 is slightly more accurate.

```
# Diebold-Mariano test of forecast accuracy comparison
dm.test(residuals(fit1), residuals(fit2), alternative = c("two.sided"),
    h = 4)
```

```
##
##  Diebold-Mariano Test
```

```
##
## data:  residuals(fit1)residuals(fit2)
## DM = -1.5633, Forecast horizon = 4, Loss function power = 2,
## p-value = 0.1232
## alternative hypothesis: two.sided
```

```
dm.test(residuals(fit1), residuals(fit2), alternative = c("greater"),
    h = 4)
```

```
##
##  Diebold-Mariano Test
##
## data:  residuals(fit1)residuals(fit2)
## DM = -1.5633, Forecast horizon = 4, Loss function power = 2,
## p-value = 0.9384
## alternative hypothesis: greater
```

```
dm.test(residuals(fit1), residuals(fit2), alternative = c("less"),
    h = 4)
```

```
##
##  Diebold-Mariano Test
##
## data:  residuals(fit1)residuals(fit2)
## DM = -1.5633, Forecast horizon = 4, Loss function power = 2,
## p-value = 0.06162
## alternative hypothesis: less
```

Model Selection with auto.arima

We also validate with auto.arima function.

Basic auto.arima recommends ARIMA(0,1,0)(2,1,0)[4] with AICc= - 206.5, BIC=-200.42 and RMSE = 0.034.

Auto arima with stepwise = FALSE, recommends ARIMA(0,1,0)(0,1,2)[4] with AICc=-206.76, BIC=-201.14 and RMSE = 0.034.

```
fit3 <- auto.arima(fp.training)
summary(fit3)
```

```
## Series: fp.training
## ARIMA(0,1,0)(2,1,0)[4]
##
## Coefficients:
##          sar1     sar2
##       -0.8012  -0.2491
## s.e.   0.1506   0.1618
##
## sigma^2 estimated as 0.001303:  log likelihood=106.25
## AIC=-206.5   AICc=-206.04   BIC=-200.42
##
## Training set error measures:
##                       ME       RMSE        MAE  MPE MAPE      MASE
## Training set -0.01010292 0.03396238 0.02369047 -Inf  Inf 0.1622302
##                    ACF1
## Training set -0.2087758
```

```r
fit4 <- auto.arima(fp.training, stepwise = FALSE, approximation = FALSE)
summary(fit4)
```

```
## Series: fp.training
## ARIMA(0,1,0)(0,1,2)[4]
##
## Coefficients:
##          sma1    sma2
##       -0.7714  0.4047
## s.e.   0.1552  0.1343
##
## sigma^2 estimated as 0.001281:  log likelihood=106.61
## AIC=-207.22   AICc=-206.76   BIC=-201.14
##
## Training set error measures:
##                       ME       RMSE        MAE  MPE MAPE      MASE
## Training set -0.008356378 0.03367401 0.02346029 -Inf  Inf 0.1606539
##                     ACF1
## Training set -0.1868566
```

When doing a Diebold-Mariano test of forecast accuracy comparison between fit1 and the two auto-arima models (fit3 and fit3), these are not significant so the alternative hypothesis that models have different levels of accuracy cannot be confirmed.

```r
# Diebold-Mariano test of forecast accuracy comparison
dm.test(residuals(fit1), residuals(fit3), alternative = c("two.sided"),
    h = 4)
```

```
##
##  Diebold-Mariano Test
##
## data:  residuals(fit1)residuals(fit3)
## DM = 0.65753, Forecast horizon = 4, Loss function power = 2,
## p-value = 0.5134
## alternative hypothesis: two.sided
```

```r
dm.test(residuals(fit1), residuals(fit3), alternative = c("greater"),
    h = 4)
```

```
##
##  Diebold-Mariano Test
##
## data:  residuals(fit1)residuals(fit3)
## DM = 0.65753, Forecast horizon = 4, Loss function power = 2,
## p-value = 0.2567
## alternative hypothesis: greater
```

```r
dm.test(residuals(fit1), residuals(fit4), alternative = c("two.sided"),
    h = 4)
```

```
##
##  Diebold-Mariano Test
##
## data:  residuals(fit1)residuals(fit4)
## DM = 0.88546, Forecast horizon = 4, Loss function power = 2,
## p-value = 0.3794
## alternative hypothesis: two.sided
```

```
dm.test(residuals(fit1), residuals(fit4), alternative = c("greater"),
    h = 4)
```

```
##
##  Diebold-Mariano Test
##
## data:  residuals(fit1)residuals(fit4)
## DM = 0.88546, Forecast horizon = 4, Loss function power = 2,
## p-value = 0.1897
## alternative hypothesis: greater
```

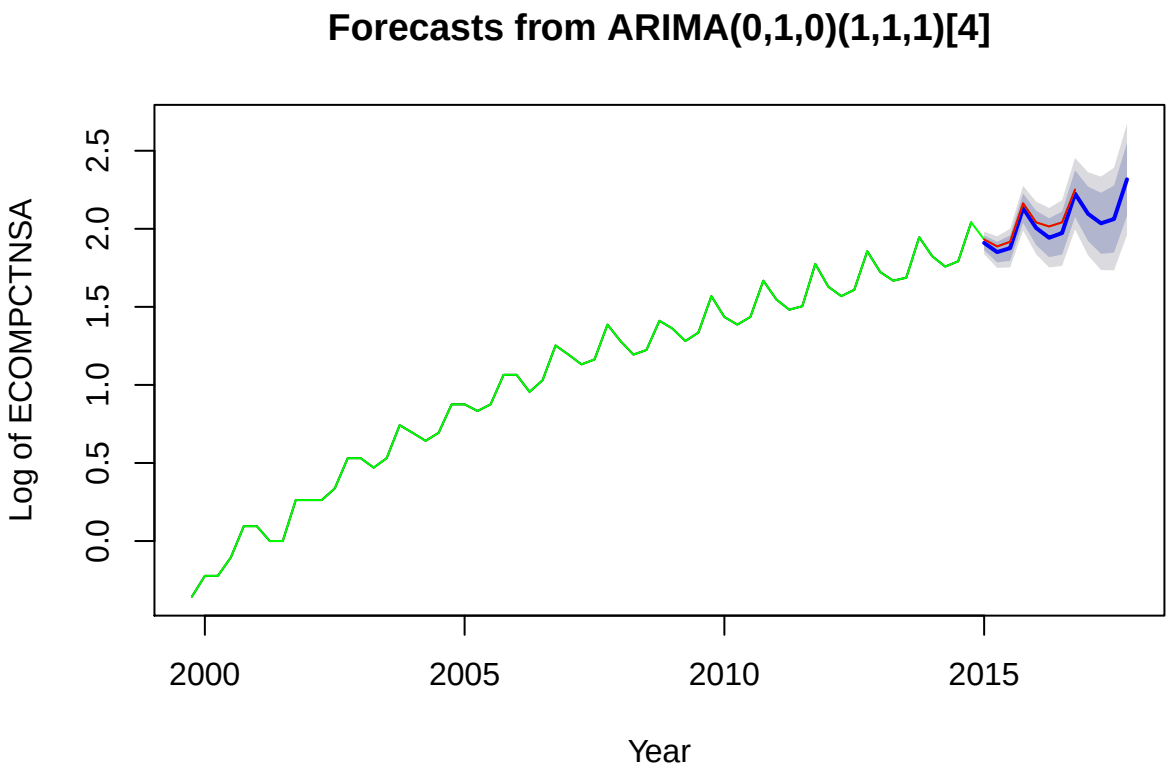
Make a Forecast for 2017

We make a forecast for 2017 using our fit1 model. The forecast for 2017 after inversing the log transform is shown the last table.

```
# Make 4-step ahead forecast (2017)
fc <- forecast.Arima(fit1, h = 12)
fc
```

```
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015 Q1       1.909642 1.862891 1.956393 1.838143 1.981142
## 2015 Q2       1.850748 1.784632 1.916864 1.749632 1.951864
## 2015 Q3       1.876204 1.795228 1.957179 1.752362 2.000045
## 2015 Q4       2.131781 2.038278 2.225283 1.988781 2.274780
## 2016 Q1       2.005801 1.895603 2.116000 1.837267 2.174336
## 2016 Q2       1.942635 1.817956 2.067313 1.751955 2.133314
## 2016 Q3       1.972718 1.835074 2.110362 1.762210 2.183226
## 2016 Q4       2.224944 2.075455 2.374433 1.996321 2.453567
## 2017 Q1       2.095897 1.921835 2.269958 1.829693 2.362101
## 2017 Q2       2.035071 1.839501 2.230642 1.735972 2.334171
## 2017 Q3       2.062619 1.847681 2.277557 1.733899 2.391338
## 2017 Q4       2.316681 2.083982 2.549380 1.960798 2.672563
```

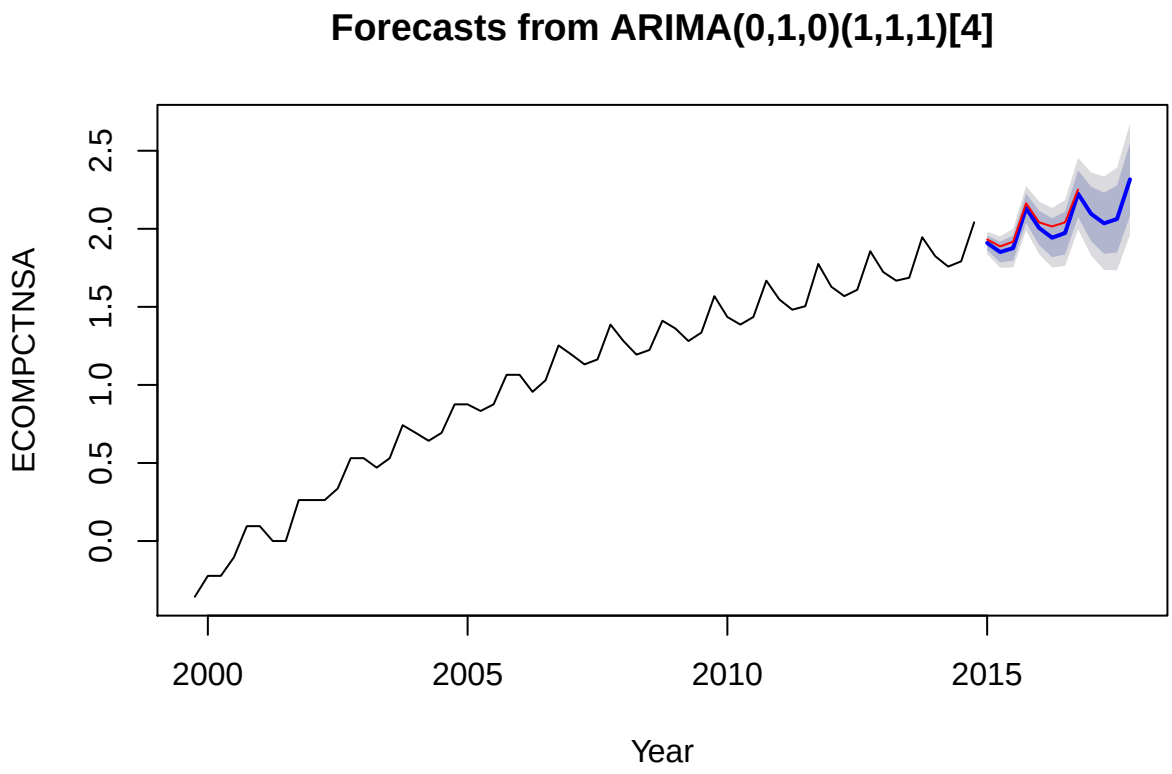
```
# Plot forecast and compare with out-of-sample observed
# values.
plot(fc, ylab = "Log of ECOMPCTNSA", xlab = "Year")
par(new = TRUE)
lines(logecompctnsa, col = "green")
par(new = TRUE)
lines(fp.test, col = "red")
```




```
# Transform back to ECOMPCTNSA', xlab = 'Year')
fc.unlog <- fc
fc.unlog$mean <- exp(fc$mean)
fc.unlog$upper <- exp(fc$upper)
fc.unlog$lower <- exp(fc$lower)
fc.unlog$x <- exp(fc$x)
fc.unlog
```

```
##         Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
## 2015 Q1       6.750674 6.442336  7.073769 6.284854  7.251019
## 2015 Q2       6.364580 5.957388  6.799604 5.752487  7.041802
## 2015 Q3       6.528672 6.020849  7.079328 5.768214  7.389387
## 2015 Q4       8.429865 7.677381  9.256103 7.306624  9.725782
## 2016 Q1       7.432048 6.656560  8.297880 6.279355  8.796340
## 2016 Q2       6.977109 6.159256  7.903561 5.765864  8.442803
## 2016 Q3       7.190195 6.265601  8.251229 5.825299  8.874894
## 2016 Q4       9.252964 7.968173 10.744916 7.361919 11.629759
## 2017 Q1       8.132730 6.833489  9.678995 6.231972 10.613223
## 2017 Q2       7.652797 6.293395  9.305837 5.674440 10.320897
## 2017 Q3       7.866543 6.345086  9.752823 5.662692 10.928107
## 2017 Q4      10.141956 8.036405 12.799165 7.104998 14.477032
```

```
plot(fc, ylab = "ECOMPCTNSA", xlab = "Year")
# par(new=TRUE) lines(df$ECOMPCTNSA, col = 'green')
par(new = TRUE)
lines(fp.test, col = "red")
```



Question 2

Imagine that you are working in a data science team and you are asked to examine the relationship between the search team on unemployment rate and related terms on *google correlate* and the actual unemployment rate.

Clearly, there are a lot of search terms your team is considering. However, for this exercise, you will only use two data series: the official monthly unemployment rate (*unemployment_rate.csv*) and the relative google search activity of the phrase "unemployment" (*correlate-unemployment-data.csv*).

You will need to build a *Vector Autoregressive (VAR) model* and generate monthly forecast for the next 6 months. Make sure that your steps of building the model are very clearly conducted. All the model building steps covered in lecture 6 - 10 are applicable.

As always, checking the raw data, conducting a thorough EDA, justifying all modeling decisions (including transformation), testing model assumptions, and clearly articulating why you chose your given model. Measure and discuss your model's performance. Use both in-sample and out-of-sample model performance. When training your model, exclude all the observations in 2016 and 2017. For the out-of-sample forecast, measure your model's performance in forecasting the official monthly unemployment rate in 2015 and 2016. Discuss the model performance. Also forecast beyond the observed time-period of the series. Specifically, generate a 6-month forecast beyond the last observed month in the unemployment series.

Initial EDA

For the investigation, two data sets are being investigated. 1. The Official Monthly Unemployment Rate 2. The Search Activity of the phrase "unemployment", as provided by *Google Correlate*

The *unemployment_rate.csv* contains 85 observations of 2 variables (DATE and UNRATENSA). The time series starts with 2010-01-01 and ends with 2017-01-01 and is monthly. The mean is 7.17 with min 4.4 and max 10.6. There is no missing DATE or NA value.

The *google_search_unemployment.csv* contains 145 observations of 2 variables (Date and unemployment). The time series starts with 2004-01-01 and ends with 2016-01-01 and is monthly. The mean is 0 with min -1.13 and max 2.91 (normalized search activity). There is no missing Date and no NA observations.

The data for the official unemployment rate begins with January 2010 and ends with January 2017. The data from Google beings with January 2004 and ends with January 2016. For the analysis in this report, the data will be restricted to the intersection of the data ranges, which is January 2010 to January 2016.

```
# base_path <-'C:/Users/Kari/source_code/MIDS/W271/lab3_kr/'
base_path <- "C:/Users/jfors/Documents/Berkeley/W271_Statistical_Methods/Lab3/"
require(forecast)
require(tseries)
require(psych)
library(car)
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:psych':
##
##     logit
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(dplyr)
library(Hmisc)
library(astsa)
library(vars)
```

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

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: strucchange
```

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

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

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

```
# Read in both dataframes
official_rate.df <- read.csv(paste(base_path, "unemployment_rate.csv",
    sep = ""), header = TRUE)
google_search.df <- read.csv(paste(base_path, "google_search_unemployment.csv",
```

```
    sep = ""), header = TRUE)

# Review the data
str(official_rate.df)
```

```
## 'data.frame':    85 obs. of  2 variables:
##  $ DATE     : Factor w/ 85 levels "2010-01-01","2010-02-01",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ UNRATENSA: num  10.6 10.4 10.2 9.5 9.3 9.6 9.7 9.5 9.2 9 ...
```

```
View(official_rate.df)
describe(official_rate.df)
```

```
##           vars  n  mean    sd median trimmed   mad min  max range skew
## DATE*        1 85 43.00 24.68   43.0   43.00 31.13 1.0 85.0  84.0 0.00
## UNRATENSA    2 85  7.17  1.76    7.3    7.14  2.52 4.4 10.6   6.2 0.06
##           kurtosis   se
## DATE*        -1.24 2.68
## UNRATENSA    -1.34 0.19
```

```
head(official_rate.df)
```

```
##         DATE UNRATENSA
## 1 2010-01-01      10.6
## 2 2010-02-01      10.4
## 3 2010-03-01      10.2
## 4 2010-04-01       9.5
## 5 2010-05-01       9.3
## 6 2010-06-01       9.6
```

```
tail(official_rate.df)
```

```
##          DATE UNRATENSA
## 80 2016-08-01       5.0
## 81 2016-09-01       4.8
## 82 2016-10-01       4.7
## 83 2016-11-01       4.4
## 84 2016-12-01       4.5
## 85 2017-01-01       5.1
```

```
anyNA(official_rate.df)
```

```
## [1] FALSE
```

```
elapsed_months("2017-01-01", "2010-01-01")  # Gives 84 which matches with 85 observations
```

```
## [1] 84
```

```
str(google_search.df)
```

```
## 'data.frame':    145 obs. of  2 variables:
##  $ Date        : Factor w/ 145 levels "2004-01-01","2004-02-01",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ unemployment: num  -0.684 -0.802 -0.903 -0.956 -1.003 ...
```

```
View(google_search.df)
describe(google_search.df)
```

```
##             vars   n mean sd median trimmed   mad   min    max  range
## Date*          1 145   73 42  73.00   73.00 53.37  1.00 145.00 144.00
## unemployment   2 145    0  1  -0.29   -0.08  1.14 -1.13   2.91   4.04
```

```
##              skew kurtosis   se
## Date*        0.00    -1.22 3.49
## unemployment 0.51    -0.98 0.08
```

```
head(google_search.df)
```

```
##         Date unemployment
## 1 2004-01-01       -0.684
## 2 2004-02-01       -0.802
## 3 2004-03-01       -0.903
## 4 2004-04-01       -0.956
## 5 2004-05-01       -1.003
## 6 2004-06-01       -0.981
```

```
tail(google_search.df)
```

```
##           Date unemployment
## 140 2015-08-01       -0.684
## 141 2015-09-01       -0.762
## 142 2015-10-01       -0.776
## 143 2015-11-01       -0.705
## 144 2015-12-01       -0.630
## 145 2016-01-01       -0.288
```

```
anyNA(google_search.df)
```

```
## [1] FALSE
```

```
elapsed_months("2016-01-01", "2004-01-01")  #Gives 144 which matches with 145 observations
```

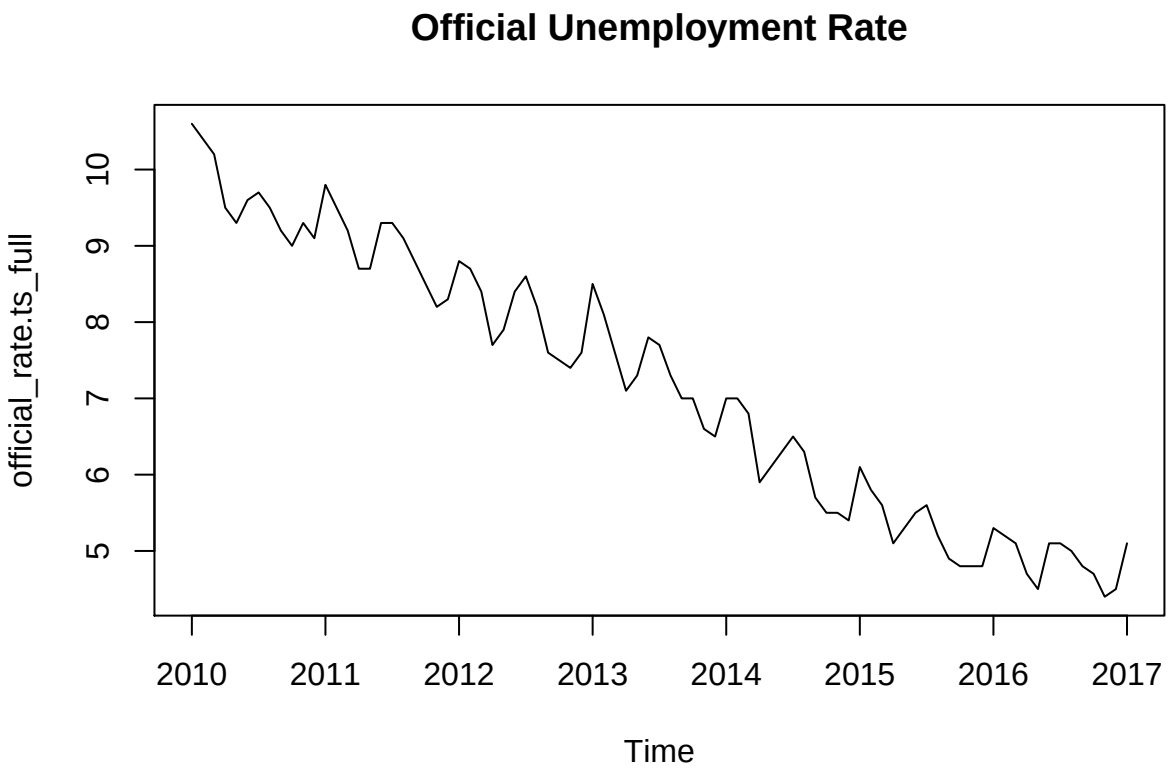
```
## [1] 144
```

Convert each of Data Frame to a Time Series

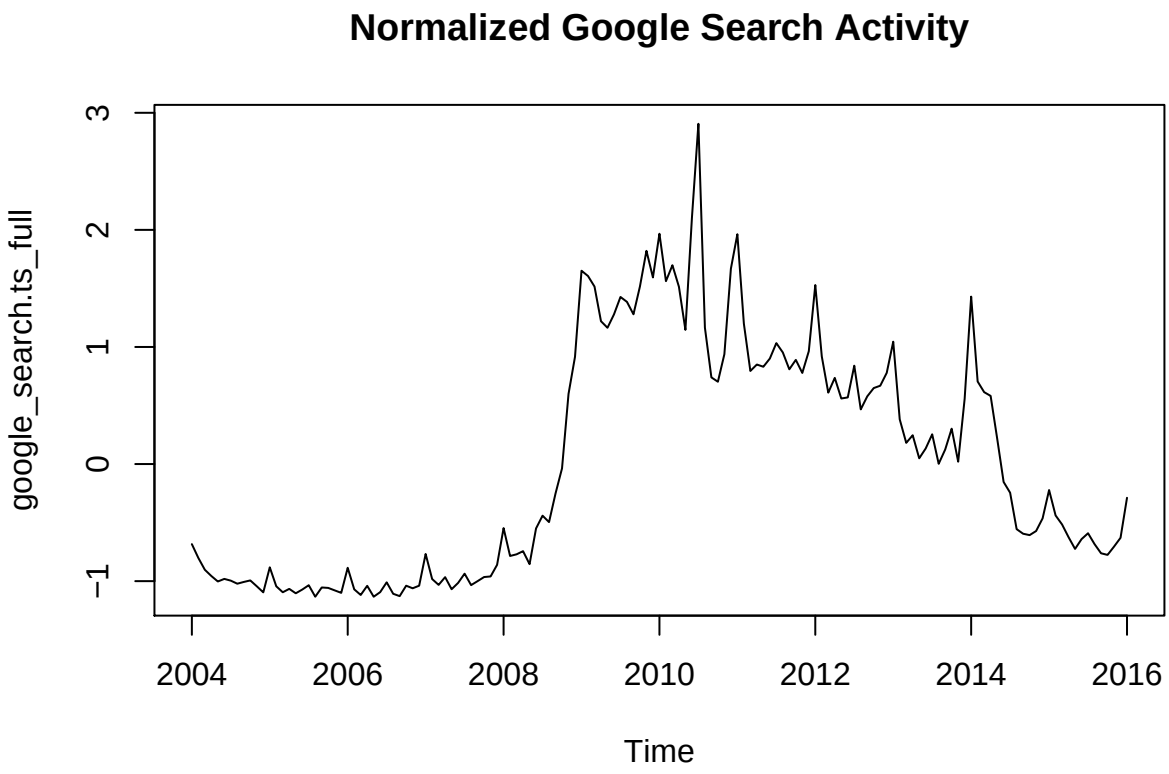
By using the decompose function, we can see that both time series have a downward trend for the period and a 6 month seasonal component. The seasonal spike is lower for mid-year as compared to beginning of year for the normalized Google Search correlate activity.

Note specifically how the the beginning of the longer time series for the Google Search shows that we over a longer time period does not have a permanent trend. The same really applies to Unemployment Rate also over a longer time period and we will use this knowledge in model selection later on.

```
official_rate.ts_full <- ts(official_rate.df$UNRATENSA, start = c(2010,
    1), freq = 12)
google_search.ts_full <- ts(google_search.df$unemployment, start = c(2004,
    1), freq = 12)
plot(official_rate.ts_full)
title("Official Unemployment Rate")
```

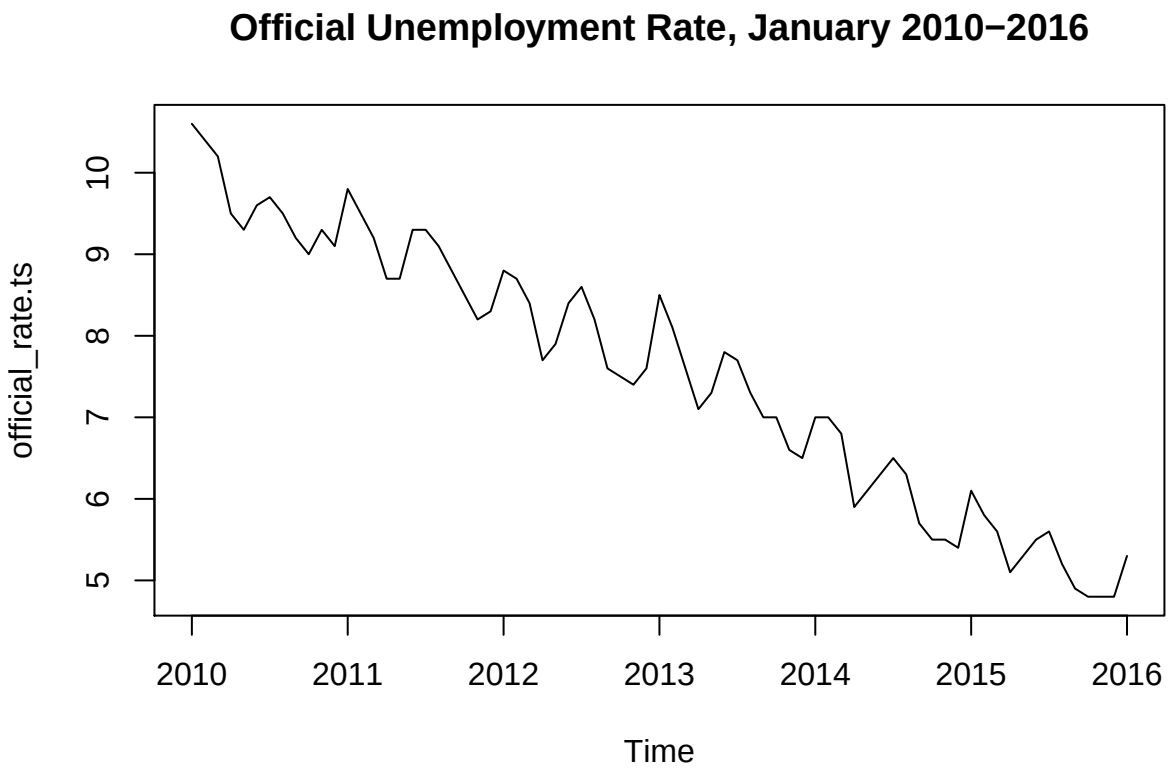



```
plot(google_search.ts_full)
title("Normalized Google Search Activity")
```

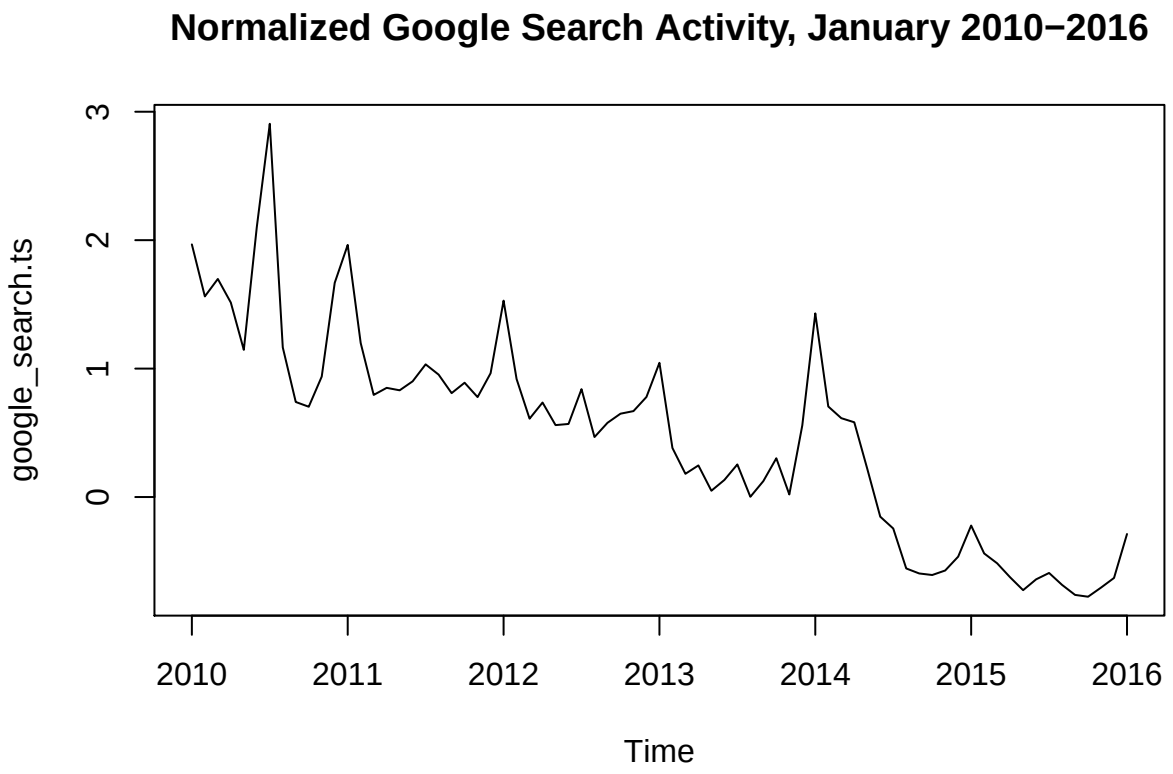



```
official_rate.ts <- window(official_rate.ts_full, start = c(2010,
    1), end = c(2016, 1))
google_search.ts <- window(google_search.ts_full, start = c(2010,
    1), end = c(2016, 1))

plot(official_rate.ts)
title("Official Unemployment Rate, January 2010-2016")
```

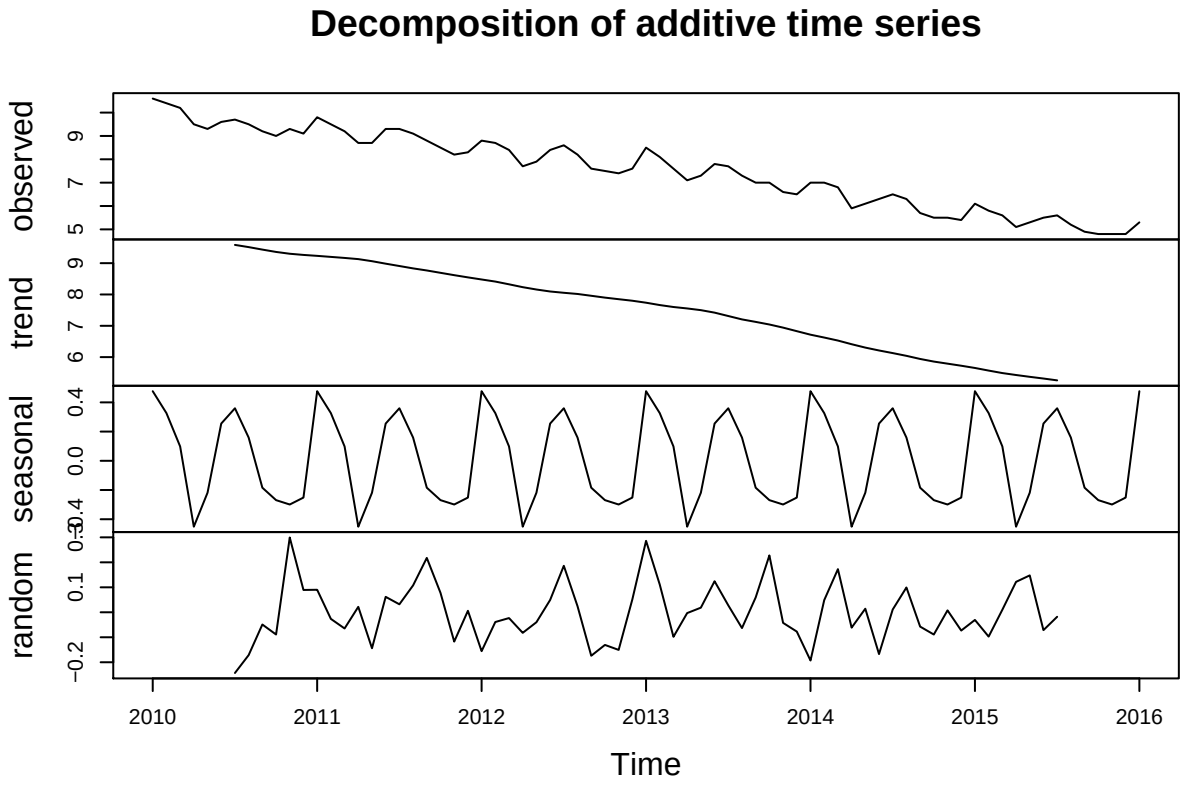



```
plot(google_search.ts)
title("Normalized Google Search Activity, January 2010-2016")
```



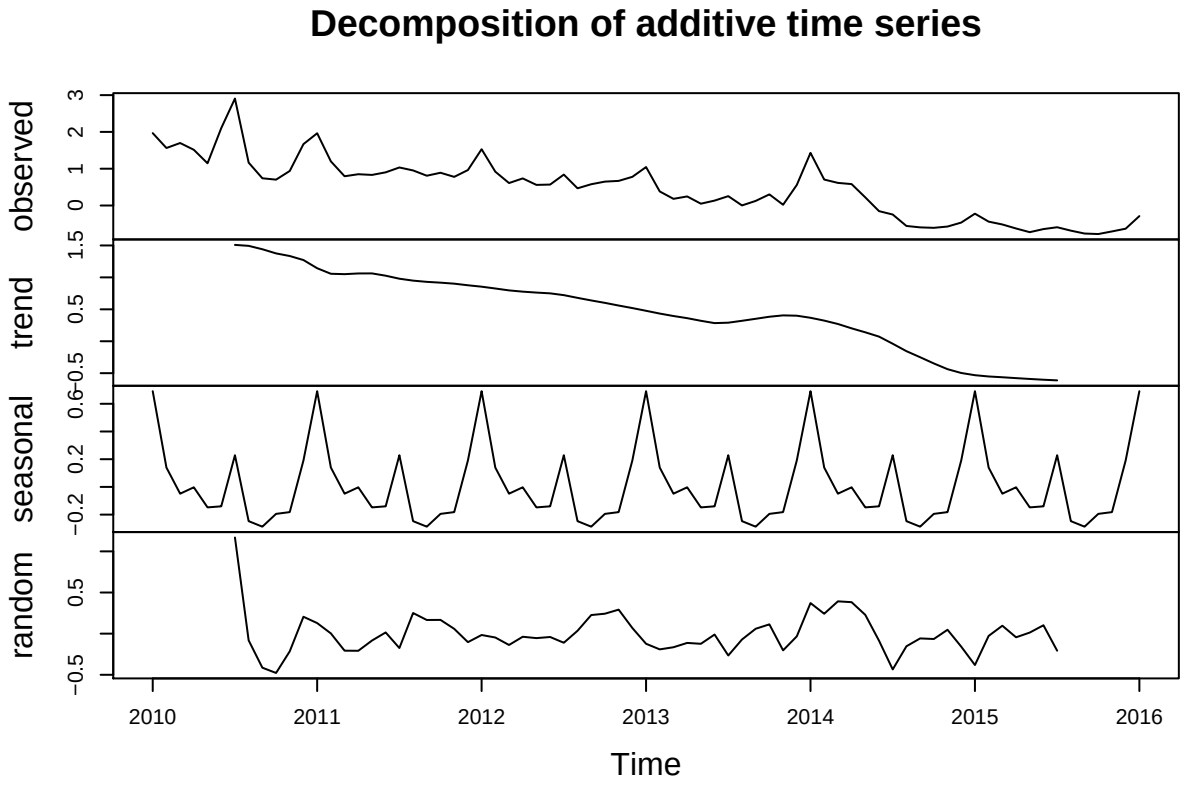

```
official_rate_decomp <- decompose(official_rate.ts)
google_search_decomp <- decompose(google_search.ts)

plot(official_rate_decomp)
title(sub = "Official Unemployment Rate, January 2010-2016")
```



Official Unemployment Rate, January 2010−2016

```
plot(google_search_decomp)
title(sub = "Normalized Google Search Activity, January 2010-2016")
```




```
# Double check that the series are now the same length, i.e.
# 73
t_o <- c(1:length(official_rate.ts))
length(t_o)
```

```
## [1] 73
```

```
t_g <- c(1:length(google_search.ts))
length(t_g)
```

```
## [1] 73
```

Basic Correlation of the Series between 2010 and 2016

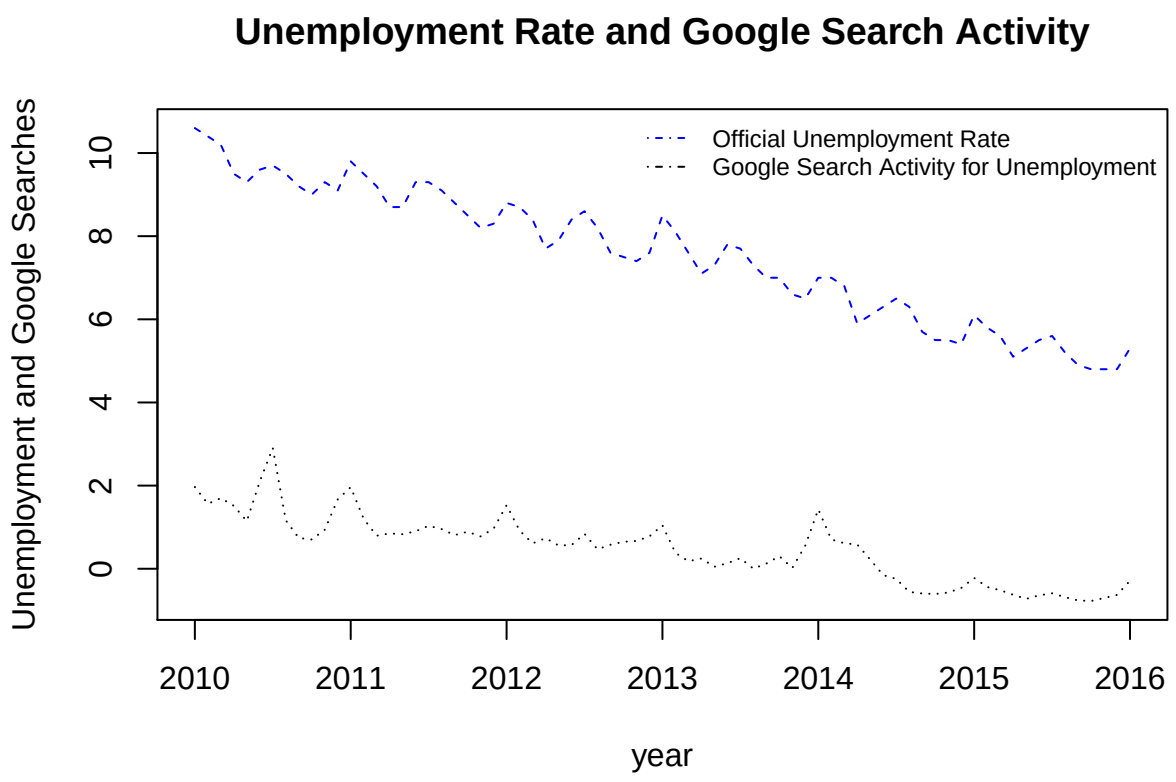
The Unemployment Rate series shows a strong seasonal variations and a downward trend for the period the observations were taken. The Google Search time series does have more irregularities in its seasonal trend (a few spikes that are significantly higher than others). It does have a downward trend as well for this time period just not showing as pronounced in a combined graph as measured closer to zero. The correlation is also high (0.893).

```
cor(official_rate.ts, google_search.ts)
```

```
## [1] 0.8933178
```

```
# plot the series
ts.plot(official_rate.ts, google_search.ts, gpars = list(xlab = "year",
    ylab = "Unemployment and Google Searches", lty = c(2:3),
    pch = c(1, 4), col = c("blue", "black")))
```

```
title("Unemployment Rate and Google Search Activity")
leg.txt <- c("Official Unemployment Rate", "Google Search Activity for Unemployment")
legend("topright", leg.txt, lty = 16, col = c("blue", "black"),
    bty = "n", cex = 0.75)
```

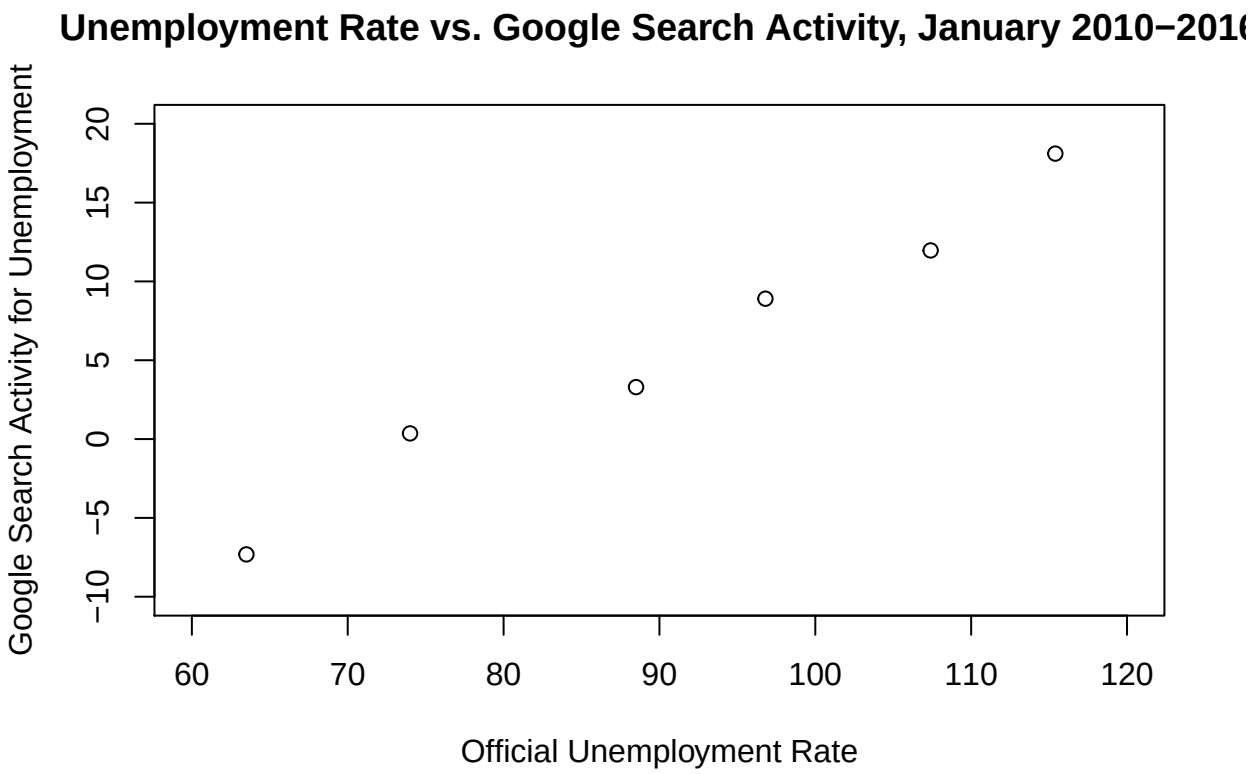


Aggregate the Data into an Annual Series

When aggregating the time series to a yearly basis, the correlation went from a 0.893 at a monthly series to 0.986 when aggregated. In other words the linear relationship became stronger at a yearly level.

```
tempx <- aggregate(official_rate.ts)
tempy <- aggregate(google_search.ts)

plot(as.vector(aggregate(official_rate.ts)), as.vector(aggregate(google_search.ts)),
    xlab = "Official Unemployment Rate", ylab = "Google Search Activity for Unemployment",
    main = "Unemployment Rate vs. Google Search Activity, January 2010-2016",
    xlim = c(60, 120), ylim = c(-10, 20))
```




```
cor(tempx, tempy)
```

```
## [1] 0.986728
```

ACF and PACF of the Series

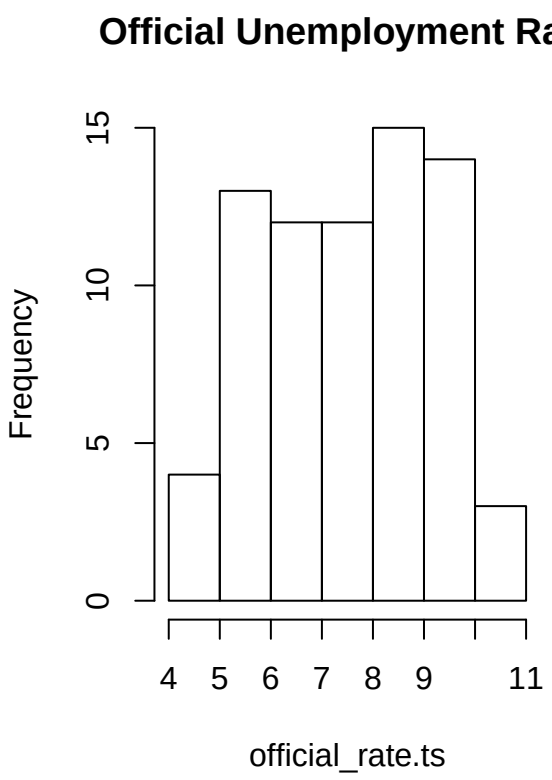
The autocorrelation functions for both the Correlate Google Search and the Official Unemployment Rate show some seasonality and a quite slow dampening over time. It appears that the ACF Google Search lags the Official Unemployment Rate, which is reasonable as likely an Unemployment Rate increase will trigger more Google Search for unemployment.

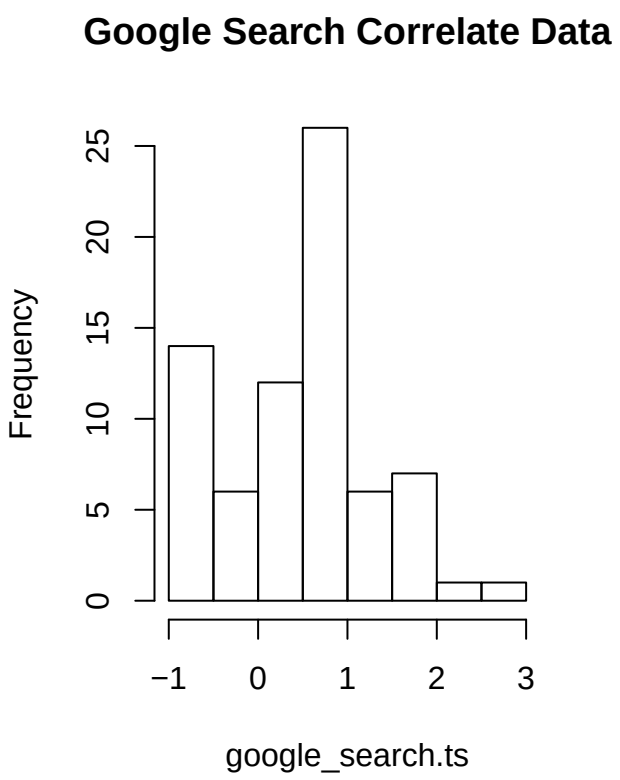
The ACF is indicating a non-stationary series. We have an alternating dependence structure, which causes time series to oscillate. We have a spike in lag 1 in the PACF, that is an indication of a Random Walk. We do have a Random Walk with drift as we have a trend. There is a also a 6-month seasonality for Correlate Google Search and a 12-month seasonality for Official Unemployment Rate. Note that this is a slightly different result than when we saw visually in the decompose function, where a 6 month seasonality was considered likely for both.

From the histograms, we can see that the actual distribution of each series is slightly different in that the Official Unemployment Rate has quite a negative kurtosis vs. the Correlate Google Search time series. Given that the Google Search graph shows the correlation of change in Google Searches to a change in Unemployment Rate, it is a reasonable finding that the graph has a more normal distribution profile. We can also see that negatively correlated data points seems to gather more towards the left while a positively correlated data points are more spread out. This can be interpreted as a negative signal to the market in terms of Unemployment Rate brings a more unified response in Google Search activity as compared to positive signal.

```
par(mfrow = c(2, 2))
acf(official_rate.ts, 156, main = "")
title("ACF of Offical Unemployment Rate")
acf(google_search.ts, 156, main = "")
title("ACF of Google Search Data")
pacf(official_rate.ts, 156, main = "")
title("PACF of Offical Unemployment Rate")
pacf(google_search.ts, 156, main = "")
title("PACF of Google Search Data")
```


```
par(mfrow = c(1, 2))
hist(official_rate.ts, main = "")
title("Official Unemployment Rate")
hist(google_search.ts, main = "")
title("Google Search Correlate Data")
```





Contemporaneous Correlation and Density of Series

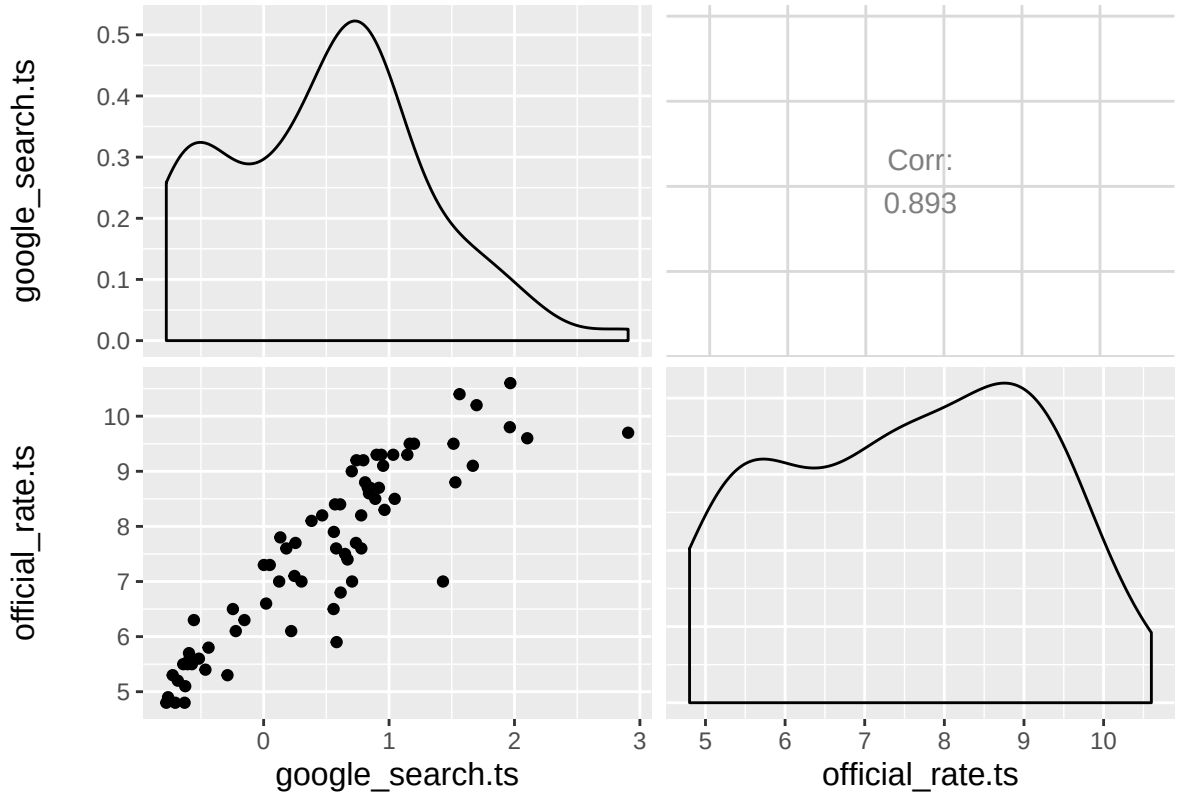
The scatter plot shows a high correlation (0.893 on a monthly basis) between the two series as we have identified before. We can in the graph also see two outliers: one close to Correlate Google Search = 3 and one close to 1.5. We will still leave these in as there is no good reason to remove or alter. Apart from that the observations falls into a straight line as expected. The data distribution graphs show same information as the histograms we displaced before.

```
library(GGally)
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
unemployment.ts <- cbind(google_search.ts, official_rate.ts)
ggpairs(unemployment.ts)
```



Unit Root Test

Augmunted Dicky-Fuller Test

We perform an Augmented Dicky-Fuller Test to assess if a unit root exists. The null hypothesis is that the series has a unit root. For both of these series, we cannot reject the null hypothesis as non-significant p-value (> 0.05). This result is not surprising from our exploratory analysis, in that the data was trending. The detrended time series (one diff) both have p-value = 0.01 (<0.05), so one can reject the null hypothesis of unit root.

```
adf.test(official_rate.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  official_rate.ts
## Dickey-Fuller = -2.4637, Lag order = 4, p-value = 0.3865
## alternative hypothesis: stationary
```

```
adf.test(google_search.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  google_search.ts
## Dickey-Fuller = -2.9907, Lag order = 4, p-value = 0.1718
```

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

Phillips-Perron Unit Root Test

We also evaluate the alternative Philips-Perron Test as only 73 observations. PP is semi-parametric and estimates the auto-correlation in the stationary process instead of assuming AR approximation. In this case, we get a significant p-value = 0.01 even without differentiation of the time series. However, we rely on the previous visuals of the time series with slowly oscillating ACF and spike at lag 1 for PACF and continue based on the basis that the original time series are non-stationary as the ADF test indicated.

```
pp.test(official_rate.ts)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  official_rate.ts
## Dickey-Fuller Z(alpha) = -32.915, Truncation lag parameter = 3,
## p-value = 0.01
## alternative hypothesis: stationary
```

```
pp.test(google_search.ts)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  google_search.ts
## Dickey-Fuller Z(alpha) = -36.024, Truncation lag parameter = 3,
## p-value = 0.01
## alternative hypothesis: stationary
```

Cointegration and Casual Relationship

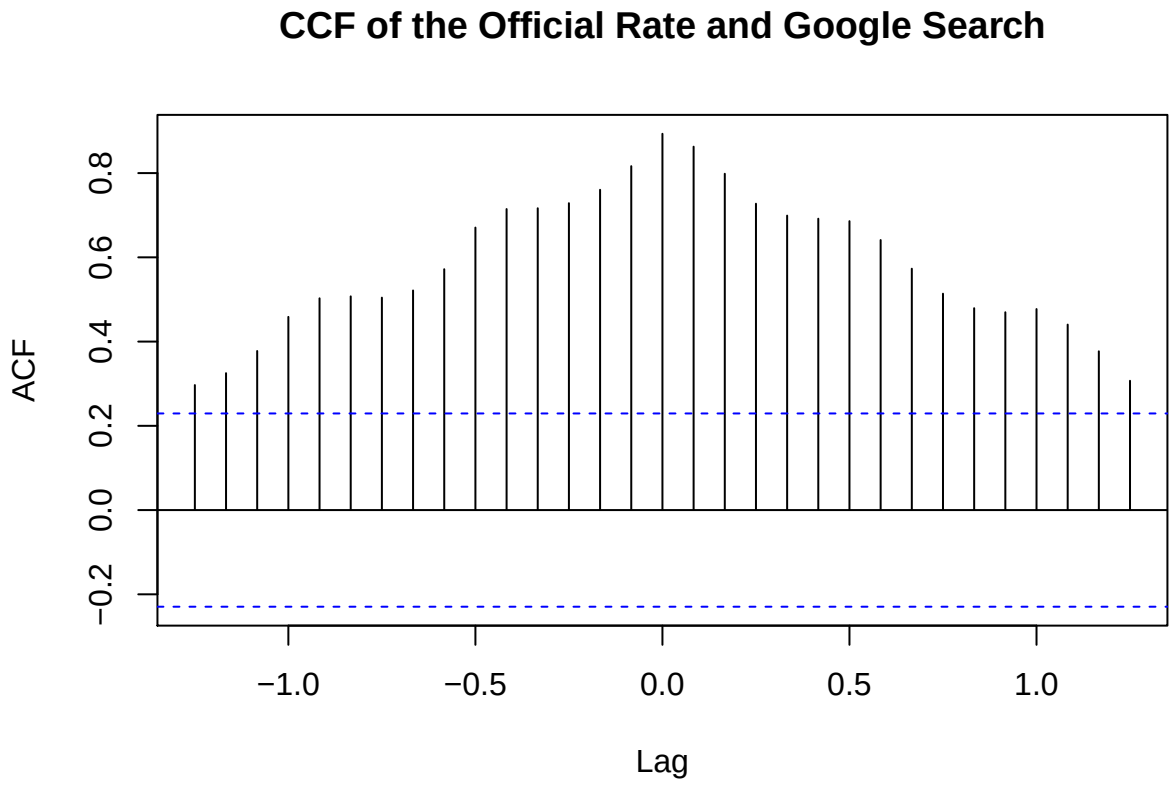
We test for cointegration of the two series as many series can show high correlation due to chance in case of unit root nd random walk. However, it is quite possible for two series to contain unit roots and be related also. Such series are said to be cointegrated, i.e. a linear combination of the series produces a stationary time series. We investigate this by reviewing the cross-correlation and with a Phllips-Ouliaris Test.

We get a high cross-correlation of 0.89. The graph also shows cross-correlation across many lags.

```
cat("Correlation Matrix:", cor(official_rate.ts, google_search.ts))
```

```
## Correlation Matrix: 0.8933178
```

```
par(mfrow = c(1, 1))
ccf(official_rate.ts, google_search.ts, main = "CCF of the Official Rate and Google Search")
```



The results from the Phillips-Ouliaris Test also shows a statistically significant result (p-value = 0.01 < 0.05), meaning that we can reject the null hypothesis that the series are not cointegrated.

```
po.test(cbind(google_search.ts, official_rate.ts))
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  cbind(google_search.ts, official_rate.ts)
## Phillips-Ouliaris demeaned = -35.464, Truncation lag parameter =
## 0, p-value = 0.01
```

VAR Model Creation

Several models are created and then evaluated and compared with each other. The sets of models include: 1. VAR model(s) using the entire sample set (in-sample) 2. VAR model using entire sample set, with differencing 3. VAR model using the sample broken into training and testing data (out-of-sample)

VAR Model 1 based on In-Sample Method and with Seasonality

We use the VARSelect function to determine a model to use. We apply a constant as we know that the Unemployment Rate will never go down to zero. We don't apply a trend in the model even if our shorter time series actually has this. The reason is that the official Unemployment Rate will go up and down over time and we don't want to have a fixed trend in our model when doing forecasting (could lead to a negative unemployment rate). We add a seasonal component of one year (season = 12) as this is what our PACF

showed for the Unemployment Rate time series. Given the result with cross-correlation for many lags, we do use a higher lag.max in the model selection.

Even if higher order lags show slightly improved results for AIC, we see that a smaller 1-lag model is recommended by SC (BIC = -4.967) and that seems wise given that we are close to a turning point for the unemployment rate. The Portmanteau test for this model is not statistically significant with p-value = 0.3216 (> 0.05) and same for the Breusch-Godfrey Test (p-value = 0.07242), which means that we cannot reject the null hypothesis of no serial correlation in the residuals. The covariance of residuals are also small = -0.0007 and so are the correlation of residuals = -0.01716. The ARCH test is highly significant with p-value = 1.424e-06, meaning that we can reject the null hypothesis of no ARCH effects (no heteroscedasticity).

We can see that the constant in this model for predicting the official Unemployment Rate is not significant after all. The same applies to the Google Search variable. The latter may be reasonable as likely human behavior is that the search lags the official unemployment rate. On the other hand the lag1 of the official rate as well as a number of the seasonal dummy variables are highly significant in model 1.

```
for (max_lag in 3:12) {
    # VARselect(unemployment.ts, lag.max = max_lag, type =
    # 'const', season = 12)
    p <- VARselect(unemployment.ts, lag.max = max_lag, type = "const",
        season = 12)$selection
    print(max_lag)
    print(p)
}
```

```
## [1] 3
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      3      1      3
## [1] 4
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      1      4
## [1] 5
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      5      4      1      5
## [1] 6
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      5      4      1      5
## [1] 7
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      5      1      1      5
## [1] 8
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      6      2      1      6
## [1] 9
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      6      1      1      6
## [1] 10
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      6      1      1      6
## [1] 11
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     11      1      1      6
## [1] 12
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     11      1      1      8
```

```r
# Estimate the model (based on the order chosen above)
VARselect(unemployment.ts, lag.max = 1, type = "const", season = 12)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      1      1      1      1
##
## $criteria
##                  1
## AIC(n) -6.01103658
## HQ(n)  -5.65856810
## SC(n)  -5.12566642
## FPE(n)  0.00247626
```

```r
var.mod.lag1 <- VAR(unemployment.ts, p = 1, type = "const", season = 12)
summary(var.mod.lag1)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: google_search.ts, official_rate.ts
## Deterministic variables: const
## Sample size: 72
## Log Likelihood: 40.07
## Roots of the characteristic polynomial:
## 0.9993 0.6005
## Call:
## VAR(y = unemployment.ts, p = 1, type = "const", season = 12L)
##
##
## Estimation results for equation google_search.ts:
## =================================================
## google_search.ts = google_search.ts.l1 + official_rate.ts.l1 + const + sd1 + sd2 + sd3 + sd4 + sd5 +
##
##                     Estimate Std. Error t value Pr(>|t|)
## google_search.ts.l1  0.62534    0.10017   6.243 5.40e-08 ***
## official_rate.ts.l1  0.17671    0.04936   3.580 0.000703 ***
## const               -1.18659    0.33159  -3.578 0.000706 ***
## sd1                  0.25120    0.15602   1.610 0.112809
## sd2                 -0.70910    0.16198  -4.378 5.08e-05 ***
## sd3                 -0.47579    0.15547  -3.060 0.003346 **
## sd4                 -0.34029    0.15469  -2.200 0.031811 *
## sd5                 -0.42381    0.15459  -2.741 0.008120 **
## sd6                 -0.17579    0.15351  -1.145 0.256856
## sd7                 -0.11570    0.15436  -0.750 0.456554
## sd8                 -0.73946    0.15463  -4.782 1.23e-05 ***
## sd9                 -0.46491    0.15566  -2.987 0.004127 **
## sd10                -0.30251    0.15387  -1.966 0.054090 .
## sd11                -0.30933    0.15316  -2.020 0.048053 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.2652 on 58 degrees of freedom
```

```
## Multiple R-Squared: 0.909,   Adjusted R-squared: 0.8887
## F-statistic: 44.59 on 13 and 58 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation official_rate.ts:
## ================================================
## official_rate.ts = google_search.ts.l1 + official_rate.ts.l1 + const + sd1 + sd2 + sd3 + sd4 + sd5 +
##
##                     Estimate Std. Error t value Pr(>|t|)
## google_search.ts.l1  0.05261    0.05934   0.887  0.37897
## official_rate.ts.l1  0.97447    0.02924  33.324  < 2e-16 ***
## const                0.09407    0.19643   0.479  0.63380
## sd1                  0.63426    0.09242   6.863 4.98e-09 ***
## sd2                 -0.21943    0.09596  -2.287  0.02588 *
## sd3                 -0.26196    0.09210  -2.844  0.00614 **
## sd4                 -0.61090    0.09163  -6.667 1.06e-08 ***
## sd5                  0.10696    0.09158   1.168  0.24761
## sd6                  0.40356    0.09094   4.438 4.12e-05 ***
## sd7                  0.10607    0.09144   1.160  0.25079
## sd8                 -0.28641    0.09160  -3.127  0.00276 **
## sd9                 -0.36908    0.09221  -4.003  0.00018 ***
## sd10                -0.12532    0.09115  -1.375  0.17444
## sd11                -0.06483    0.09073  -0.715  0.47778
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.1571 on 58 degrees of freedom
## Multiple R-Squared: 0.9919,  Adjusted R-squared: 0.9901
## F-statistic: 544.8 on 13 and 58 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##                  google_search.ts official_rate.ts
## google_search.ts        0.0703390       -0.0007151
## official_rate.ts       -0.0007151        0.0246829
##
## Correlation matrix of residuals:
##                  google_search.ts official_rate.ts
## google_search.ts          1.00000         -0.01716
## official_rate.ts         -0.01716          1.00000
```

```r
# Test for serial correlation in residuals with both
# Portmanteau and BG Test.  Null hypothesis: No serial
# correlation of any order up to lag p
serial.test(var.mod.lag1, lags.pt = 16, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.mod.lag1
## Chi-squared = 64.521, df = 60, p-value = 0.3216
```

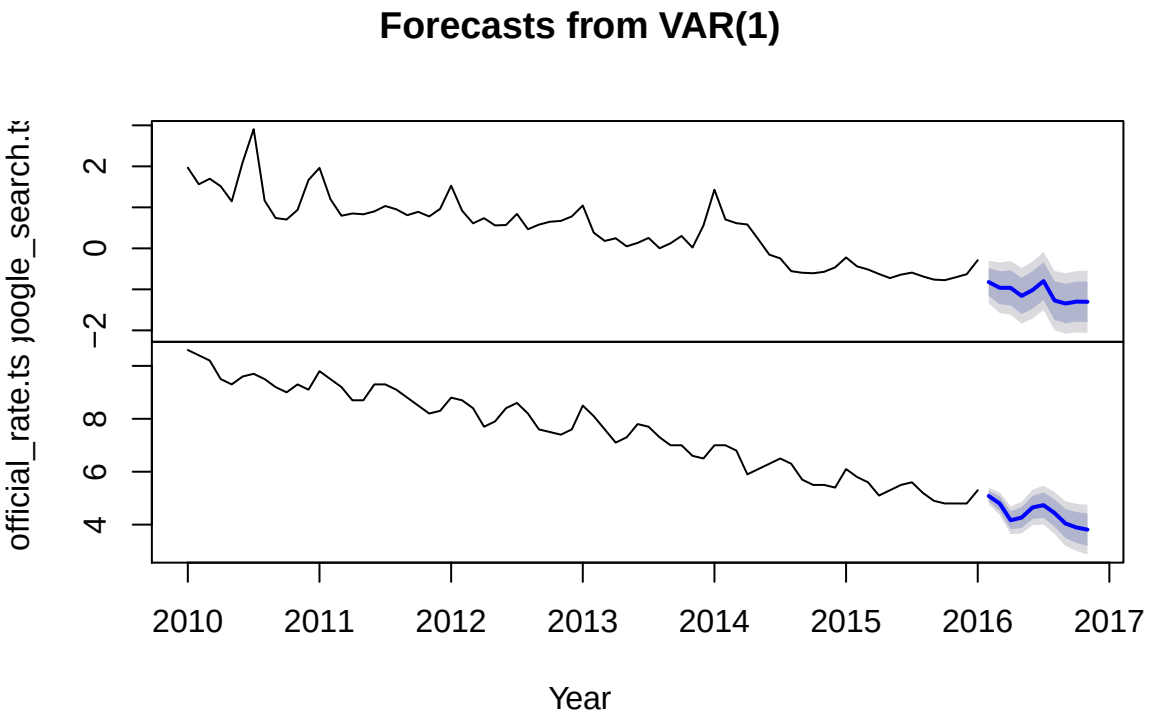
```
serial.test(var.mod.lag1, lags.bg = 16, type = "BG")
```

```
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object var.mod.lag1
## Chi-squared = 81.176, df = 64, p-value = 0.07242
```

```
# Test for heteroscedasticity.  Null hypothesis: No ARCH
# effect present
arch.test(var.mod.lag1)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var.mod.lag1
## Chi-squared = 104.02, df = 45, p-value = 1.424e-06
```

```
fcst.mod1 <- forecast(var.mod.lag1)
plot(fcst.mod1, xlab = "Year")
```



VAR Model 2 based on In-Sample with No Seasonality

We test a model without inbuilt seasonality. In this case we don't get a smaller SC (BIC = -4.1295) and also the Portmanteau Test is has a significant p-value = 0.00039 (< 0.05), meaning we can reject the null hypothesis of no serial correlation in the residuals. We will not take this model forward.

```
for (max_lag in 3:12) {
    # VARselect(unemployment.ts, lag.max = max_lag, type =
    # 'const')
    p <- VARselect(unemployment.ts, lag.max = max_lag, type = "const")$selection
    print(max_lag)
    print(p)
}
```

```
## [1] 3
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      1      1      3
## [1] 4
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      4      4
## [1] 5
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      5      5      5      5
## [1] 6
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      6      6      4      6
## [1] 7
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      7      7      7      7
## [1] 8
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      8      7      7      7
## [1] 9
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      7      7      9
## [1] 10
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     10      9      7      9
## [1] 11
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      7      7      9
## [1] 12
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      8      8      7      8
```

```
# Estimate the model (based on the order chosen above)
VARselect(unemployment.ts, lag.max = 4, type = "const")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      4      4
##
## $criteria
##                 1           2           3            4
## AIC(n) -4.24322973 -4.22782441 -4.31867260 -4.712313616
## HQ(n)  -4.16615628 -4.09936867 -4.13883455 -4.481093271
## SC(n)  -4.04895960 -3.90404086 -3.86537563 -4.129503223
## FPE(n)  0.01436271  0.01459151  0.01333622  0.009010865
```

```
var.mod.lag4 <- VAR(unemployment.ts, p = 4, type = "const")
summary(var.mod.lag4)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: google_search.ts, official_rate.ts
## Deterministic variables: const
## Sample size: 69
## Log Likelihood: -15.239
## Roots of the characteristic polynomial:
## 0.9949 0.881 0.881 0.7806 0.7806 0.7768 0.7768 0.7241
## Call:
## VAR(y = unemployment.ts, p = 4, type = "const")
##
##
## Estimation results for equation google_search.ts:
## ================================================
## google_search.ts = google_search.ts.l1 + official_rate.ts.l1 + google_search.ts.l2 + official_rate.ts
##
##                       Estimate Std. Error t value Pr(>|t|)
## google_search.ts.l1  0.780675   0.131283   5.947 1.51e-07 ***
## official_rate.ts.l1 -0.034605   0.133398  -0.259 0.796206
## google_search.ts.l2 -0.442554   0.154812  -2.859 0.005842 **
## official_rate.ts.l2  0.007624   0.182481   0.042 0.966814
## google_search.ts.l3  0.411453   0.154217   2.668 0.009797 **
## official_rate.ts.l3 -0.085188   0.181214  -0.470 0.639992
## google_search.ts.l4 -0.360048   0.131487  -2.738 0.008118 **
## official_rate.ts.l4  0.381694   0.127224   3.000 0.003926 **
## const               -1.823337   0.508331  -3.587 0.000673 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.3212 on 60 degrees of freedom
## Multiple R-Squared: 0.8483,  Adjusted R-squared: 0.8281
## F-statistic: 41.95 on 8 and 60 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation official_rate.ts:
## ================================================
## official_rate.ts = google_search.ts.l1 + official_rate.ts.l1 + google_search.ts.l2 + official_rate.ts
##
##                       Estimate Std. Error t value Pr(>|t|)
## google_search.ts.l1  0.096363   0.117189   0.822    0.414
## official_rate.ts.l1  0.895547   0.119077   7.521 3.23e-10 ***
## google_search.ts.l2  0.009783   0.138193   0.071    0.944
## official_rate.ts.l2 -0.203674   0.162891  -1.250    0.216
## google_search.ts.l3 -0.111465   0.137661  -0.810    0.421
## official_rate.ts.l3 -0.238911   0.161760  -1.477    0.145
## google_search.ts.l4  0.050954   0.117371   0.434    0.666
## official_rate.ts.l4  0.517046   0.113566   4.553 2.64e-05 ***
## const                0.072625   0.453760   0.160    0.873
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
```

```
## Residual standard error: 0.2867 on 60 degrees of freedom
## Multiple R-Squared: 0.9683,  Adjusted R-squared: 0.9641
## F-statistic: 229.4 on 8 and 60 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##                   google_search.ts official_rate.ts
## google_search.ts          0.10317          0.03783
## official_rate.ts          0.03783          0.08221
##
## Correlation matrix of residuals:
##                   google_search.ts official_rate.ts
## google_search.ts           1.0000           0.4107
## official_rate.ts           0.4107           1.0000
```

```
# Test for serial correlation in residuals with both
# Portmanteau and BG Test.  Null hypothesis: No serial
# correlation of any order up to lag p
serial.test(var.mod.lag4, lags.pt = 16, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.mod.lag4
## Chi-squared = 87.856, df = 48, p-value = 0.0003943
```

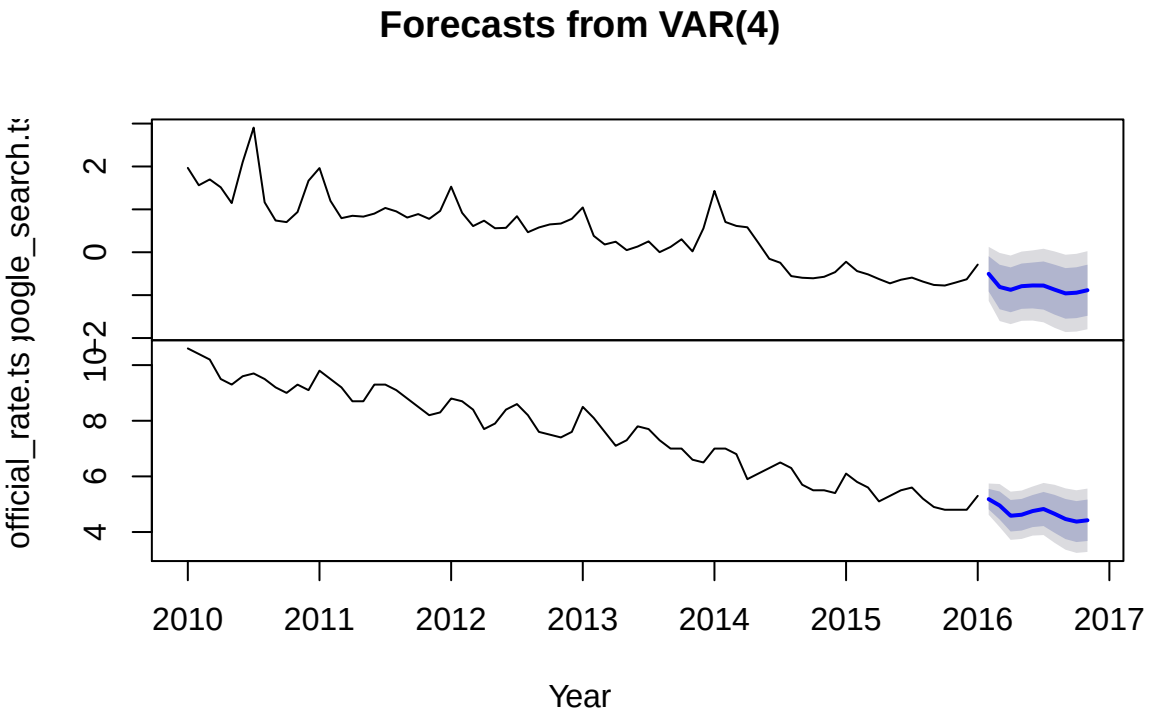
```
serial.test(var.mod.lag4, lags.bg = 16, type = "BG")
```

```
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object var.mod.lag4
## Chi-squared = 111.26, df = 64, p-value = 0.0002316
```

```
# Test for heteroscedasticity.  Null hypothesis: No ARCH
# effect present
arch.test(var.mod.lag4)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var.mod.lag4
## Chi-squared = 61.144, df = 45, p-value = 0.0547
```

```
fcst.mod2 <- forecast(var.mod.lag4)
plot(fcst.mod2, xlab = "Year")
```



VAR Model 3 based on Manually Detrended Series

As a downward trend is persistent over the time period we have in the sample, we will also test to detrend manually by differentiation and then make our model without the trend for future forecasting. The VAR model 3 was created using the full sample series but using differencing.

Determine Stationarity and lags

By using the adf test it is shown that a 1st difference will achieve stationarity.

```
ndiffs(official_rate.ts, alpha = 0.05, test = c("adf"))
```

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

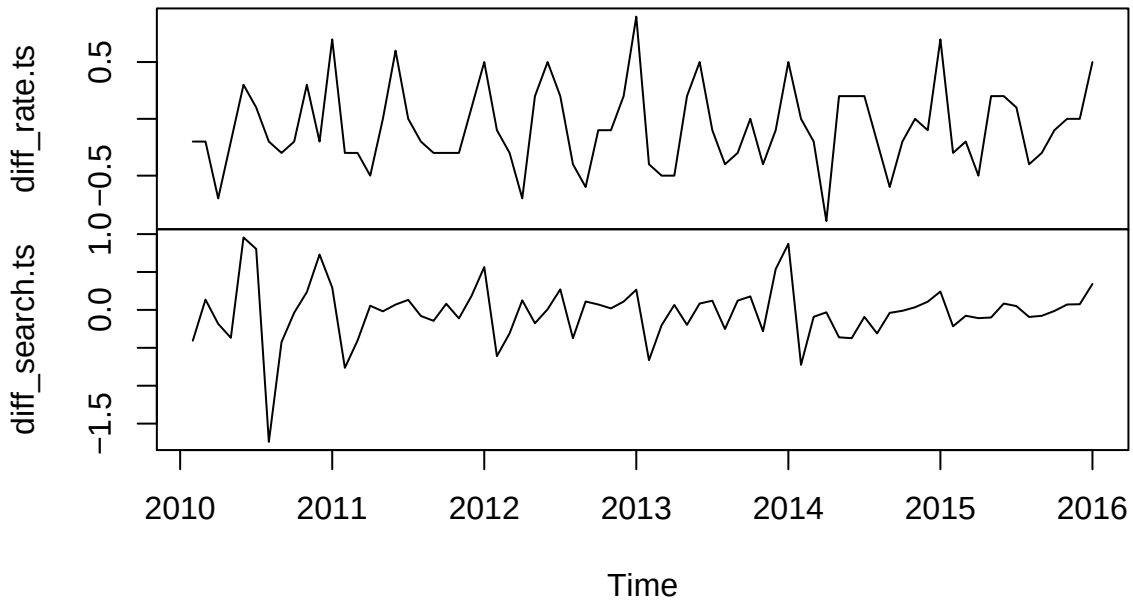
```
ndiffs(google_search.ts, alpha = 0.05, test = c("adf"))
```

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

Detrending by taking differentiation on both series and then combining into one dataframe.

```
diff_rate.ts <- diff(official_rate.ts)
diff_search.ts <- diff(google_search.ts)

diff_unemployment.ts <- cbind(diff_rate.ts, diff_search.ts)
plot(diff_unemployment.ts)
title(sub = "Detrended Unemployment Rate and Google Search Correlate Activity")
```



Detrended Unemployment Rate and Google Search Correlate Activity

Augmented Dicky-Fuller Test

The ADF test for the differentiated times series is now statistically significant.

```
adf.test(diff_rate.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_rate.ts
## Dickey-Fuller = -7.3376, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff_search.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_search.ts
## Dickey-Fuller = -6.1743, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

Phillips-Perron Unit Root Test

Same for the Phillips-Perron Unit Root Test.

```
pp.test(diff_rate.ts)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  diff_rate.ts
## Dickey-Fuller Z(alpha) = -44.814, Truncation lag parameter = 3,
## p-value = 0.01
## alternative hypothesis: stationary
```

```
pp.test(diff_search.ts)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  diff_search.ts
## Dickey-Fuller Z(alpha) = -58.196, Truncation lag parameter = 3,
## p-value = 0.01
## alternative hypothesis: stationary
```

Difference Model Creation and Results

Similar to the first two VAR models, a variable number of max_lags were explored to determine which how many lags to explore. By using the Schwartz Criterion, a model of 4 was selected and evaluated. On all accounts, the VAR model 3 (created from a differenced series) was inferior to the VAR models created above. VAR model 3 was excluded from any further investiation.

```
for (max_lag in 3:10) {
    p <- VARselect(diff_unemployment.ts, lag.max = max_lag, type = "const",
        season = 12)$selection
    print(max_lag)
    print(p)
}
```

```
## [1] 3
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      3      2      3
## [1] 4
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      2      4
## [1] 5
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      3      4
## [1] 6
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      4      4
## [1] 7
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      5      4      1      4
## [1] 8
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      5      4      1      4
## [1] 9
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      1      4
```

```
## [1] 10
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      1      4
```

```
# Estimate the model (based on the order chosen above)
VARselect(diff_unemployment.ts, lag.max = 6, type = "const",
    season = 12)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      4      4      4      4
##
## $criteria
##                  1            2            3            4            5
## AIC(n) -6.014473841 -6.134624949 -6.290881694 -6.437514262 -6.392049605
## HQ(n)  -5.647403742 -5.715116266 -5.818934424 -5.913128407 -5.815225165
## SC(n)  -5.085529405 -5.072974165 -5.096524561 -5.110450782 -4.932279777
## FPE(n)  0.002475302  0.002209632  0.001906361  0.001664511  0.001765911
##                  6
## AIC(n) -6.288869634
## HQ(n)  -5.659606608
## SC(n)  -4.696393458
## FPE(n)  0.001990967
```

```
var_diff.mod1 <- VAR(diff_unemployment.ts, p = 4, type = "const",
    season = 12)
summary(var_diff.mod1)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: diff_rate.ts, diff_search.ts
## Deterministic variables: const
## Sample size: 68
## Log Likelihood: 56.516
## Roots of the characteristic polynomial:
## 0.8077 0.8077 0.7403 0.7403 0.711 0.711 0.4936 0.4936
## Call:
## VAR(y = diff_unemployment.ts, p = 4, type = "const", season = 12L)
##
##
## Estimation results for equation diff_rate.ts:
## =============================================
## diff_rate.ts = diff_rate.ts.l1 + diff_search.ts.l1 + diff_rate.ts.l2 + diff_search.ts.l2 + diff_rate
##
##                   Estimate Std. Error t value Pr(>|t|)
## diff_rate.ts.l1   -0.24244    0.13193  -1.838  0.07232 .
## diff_search.ts.l1 -0.09252    0.07089  -1.305  0.19807
## diff_rate.ts.l2   -0.26529    0.12294  -2.158  0.03596 *
## diff_search.ts.l2  0.07689    0.07204   1.067  0.29119
## diff_rate.ts.l3   -0.34059    0.12523  -2.720  0.00907 **
## diff_search.ts.l3 -0.20825    0.06820  -3.053  0.00368 **
## diff_rate.ts.l4   -0.21676    0.12859  -1.686  0.09834 .
## diff_search.ts.l4  0.08978    0.07383   1.216  0.22993
## const             -0.15155    0.03160  -4.796 1.61e-05 ***
```

```
## sd1               -0.64048    0.14736  -4.346 7.16e-05 ***
## sd2               -0.73760    0.15409  -4.787 1.66e-05 ***
## sd3               -0.95438    0.14616  -6.530 3.87e-08 ***
## sd4               -0.66028    0.17274  -3.822  0.00038 ***
## sd5               -0.41674    0.12249  -3.402  0.00136 **
## sd6               -0.54725    0.10927  -5.008 7.84e-06 ***
## sd7               -0.82616    0.10976  -7.527 1.15e-09 ***
## sd8               -0.82452    0.14346  -5.748 6.08e-07 ***
## sd9               -0.66614    0.13637  -4.885 1.19e-05 ***
## sd10              -0.93097    0.11874  -7.841 3.84e-10 ***
## sd11              -0.76774    0.09794  -7.839 3.86e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.1358 on 48 degrees of freedom
## Multiple R-Squared: 0.8999,  Adjusted R-squared: 0.8603
## F-statistic: 22.71 on 19 and 48 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation diff_search.ts:
## ===============================================
## diff_search.ts = diff_rate.ts.l1 + diff_search.ts.l1 + diff_rate.ts.l2 + diff_search.ts.l2 + diff_ra
##
##                   Estimate Std. Error t value Pr(>|t|)
## diff_rate.ts.l1   -0.29732    0.25845  -1.150 0.255687
## diff_search.ts.l1  0.02194    0.13888   0.158 0.875142
## diff_rate.ts.l2   -0.23192    0.24084  -0.963 0.340390
## diff_search.ts.l2 -0.51224    0.14113  -3.630 0.000688 ***
## diff_rate.ts.l3    0.30520    0.24533   1.244 0.219522
## diff_search.ts.l3  0.00625    0.13361   0.047 0.962886
## diff_rate.ts.l4    0.14093    0.25190   0.559 0.578448
## diff_search.ts.l4 -0.26147    0.14464  -1.808 0.076908 .
## const             -0.06218    0.06191  -1.004 0.320192
## sd1               -0.65383    0.28868  -2.265 0.028069 *
## sd2               -0.35250    0.30187  -1.168 0.248686
## sd3               -1.01262    0.28634  -3.536 0.000910 ***
## sd4               -0.92001    0.33840  -2.719 0.009094 **
## sd5               -0.48955    0.23995  -2.040 0.046855 *
## sd6               -0.04276    0.21406  -0.200 0.842511
## sd7               -0.71758    0.21502  -3.337 0.001641 **
## sd8               -0.68934    0.28103  -2.453 0.017854 *
## sd9               -0.90926    0.26714  -3.404 0.001351 **
## sd10              -0.52267    0.23260  -2.247 0.029272 *
## sd11              -0.18383    0.19186  -0.958 0.342782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.266 on 48 degrees of freedom
## Multiple R-Squared: 0.6737,  Adjusted R-squared: 0.5446
## F-statistic: 5.217 on 19 and 48 DF,  p-value: 1.787e-06
##
##
```

```
##
## Covariance matrix of residuals:
##                diff_rate.ts diff_search.ts
## diff_rate.ts      0.0184439     -0.0005088
## diff_search.ts   -0.0005088      0.0707819
##
## Correlation matrix of residuals:
##                diff_rate.ts diff_search.ts
## diff_rate.ts        1.00000       -0.01408
## diff_search.ts     -0.01408        1.00000
```

```
# Test for serial correlation in residuals with both
# Portmanteau and BG Test.  Null hypothesis: No serial
# correlation of any order up to lag p
serial.test(var_diff.mod1, lags.pt = 16, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var_diff.mod1
## Chi-squared = 62.761, df = 48, p-value = 0.07472
```

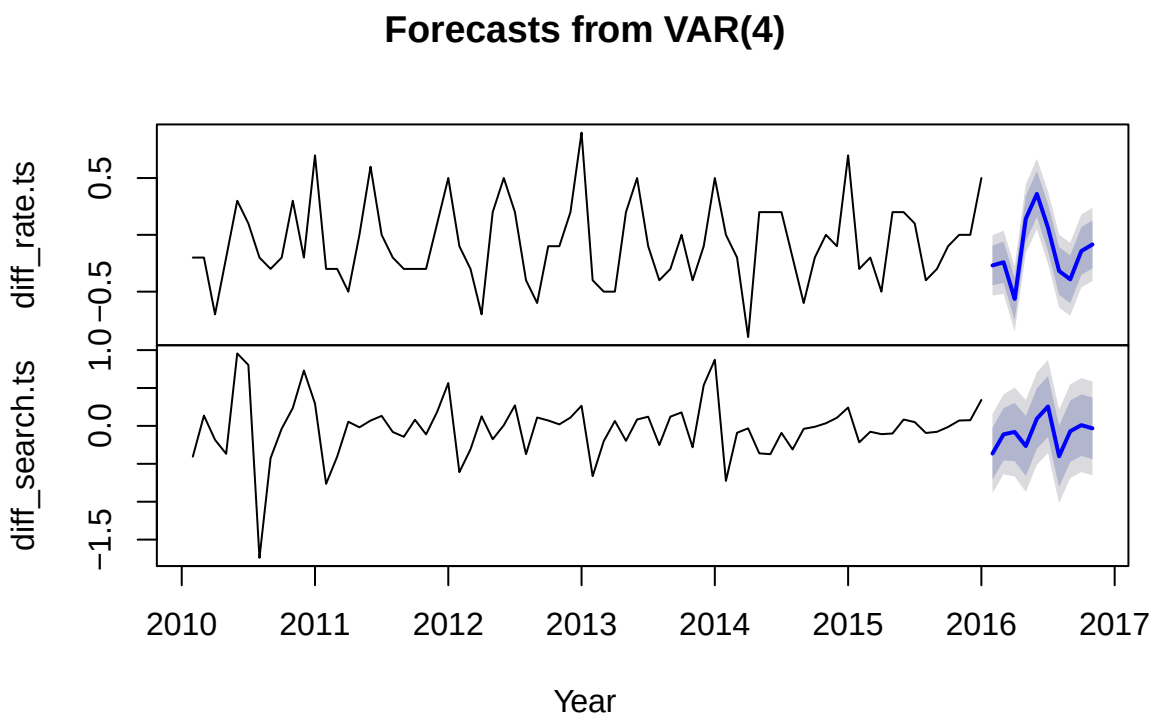
```
serial.test(var_diff.mod1, lags.bg = 16, type = "BG")
```

```
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object var_diff.mod1
## Chi-squared = 101.69, df = 64, p-value = 0.001897
```

```
# Test for heteroscedasticity.  Null hypothesis: No ARCH
# effect present
arch.test(var_diff.mod1)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var_diff.mod1
## Chi-squared = 50.024, df = 45, p-value = 0.2807
```

```
fcst_diff.mod1 <- forecast(var_diff.mod1)
plot(fcst_diff.mod1, xlab = "Year")
```



VAR Model: Breaking the Sample Data into training and test sets

The final VAR model was created by breaking the dataset into a training and test set. The model training set was created by using a subset from the data, consisting of the series from 2010 through 2014.The model test set was created by using a subset of the sample date, consisting of the series from 2015 to 2016

```
unemployment.ts.train <- window(unemployment.ts, start = c(2010,
    1), end = c(2014, 12))
unemployment.ts.test <- window(unemployment.ts, start = c(2015,
    1), end = c(2016, 1))

str(unemployment.ts.train)
```

```
##  Time-Series [1:60, 1:2] from 2010 to 2015: 1.97 1.56 1.7 1.51 1.15 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "google_search.ts" "official_rate.ts"
```

```
head(unemployment.ts.train)
```

```
##      google_search.ts official_rate.ts
## [1,]            1.967             10.6
## [2,]            1.562             10.4
## [3,]            1.698             10.2
## [4,]            1.514              9.5
## [5,]            1.146              9.3
```

```
## [6,]           2.102             9.6
```

```
tail(unemployment.ts.train)
```

```
##       google_search.ts official_rate.ts
## [55,]           -0.245             6.5
## [56,]           -0.556             6.3
## [57,]           -0.595             5.7
## [58,]           -0.607             5.5
## [59,]           -0.572             5.5
## [60,]           -0.464             5.4
```

```
str(unemployment.ts.test)
```

```
##  Time-Series [1:13, 1:2] from 2015 to 2016: -0.222 -0.439 -0.516 -0.625 -0.725 -0.641 -0.591 -0.684 -
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "google_search.ts" "official_rate.ts"
```

```
head(unemployment.ts.test)
```

```
##      google_search.ts official_rate.ts
## [1,]          -0.222             6.1
## [2,]          -0.439             5.8
## [3,]          -0.516             5.6
## [4,]          -0.625             5.1
## [5,]          -0.725             5.3
## [6,]          -0.641             5.5
```

```
tail(unemployment.ts.test)
```

```
##       google_search.ts official_rate.ts
##  [8,]           -0.684             5.2
##  [9,]           -0.762             4.9
## [10,]           -0.776             4.8
## [11,]           -0.705             4.8
## [12,]           -0.630             4.8
## [13,]           -0.288             5.3
```

Root Mean Square Error function is used to assess the optimal model for the out-of-sample model. RMSE is gradually lower with higher lags. There is a risk of overfitting though by merely following this criteria.

```
# Select optimal order accoring to out-of-sample forecast
for (order in 1:12) {
    var.trained <- VAR(unemployment.ts.train, p = order, type = c("const"),
        season = 12)
    var.trained.forecast <- predict(var.trained, n.ahead = 13)
    var.trained.forecast
    var.trained.forecast$fcst
    test.error <- calculate_rmse(var.trained.forecast$fcst$official_rate.ts[,
        1], unemployment.ts.test[, 1])
    print(c(order, test.error))
}
```

```
## [1] 1.000000 5.656808
## [1] 2.000000 5.655736
## [1] 3.000000 5.654218
## [1] 4.000000 5.534442
```

```
## [1] 5.000000 5.608397
## [1] 6.000000 5.364221
## [1] 7.000000 5.337244
## [1] 8.000000 5.381465
## [1] 9.000000 5.414937
## [1] 10.000000  5.433323
## [1] 11.000000  5.458826
## [1] 12.000000  5.296301
```

VAR Modelling with Seasonal Component and In-sample Training for VAR(4)

We select to use a lag4 model for the training data based on minimum RMSE. The Portmanteau Test for the training data is non-significant with p-value = 0.1044, which means that we cannot reject the null hypothesis of no serial correlation in the residuals.

```
var.trained.l4 <- VAR(unemployment.ts.train, p = 4, type = c("const"),
    season = 12)
summary(var.trained.l4)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: google_search.ts, official_rate.ts
## Deterministic variables: const
## Sample size: 56
## Log Likelihood: 42.514
## Roots of the characteristic polynomial:
## 1.009 0.8229 0.8229 0.7195 0.7195 0.434 0.434 0.0905
## Call:
## VAR(y = unemployment.ts.train, p = 4, type = c("const"), season = 12L)
##
##
## Estimation results for equation google_search.ts:
## ================================================
## google_search.ts = google_search.ts.l1 + official_rate.ts.l1 + google_search.ts.l2 + official_rate.ts
##
##                      Estimate Std. Error t value Pr(>|t|)
## google_search.ts.l1  0.84786    0.16889   5.020 1.41e-05 ***
## official_rate.ts.l1 -0.24934    0.24992  -0.998  0.32510
## google_search.ts.l2 -0.50088    0.20668  -2.424  0.02052 *
## official_rate.ts.l2  0.05573    0.31484   0.177  0.86049
## google_search.ts.l3  0.27945    0.20566   1.359  0.18266
## official_rate.ts.l3  0.46740    0.31547   1.482  0.14715
## google_search.ts.l4 -0.08018    0.15340  -0.523  0.60440
## official_rate.ts.l4 -0.05676    0.27190  -0.209  0.83583
## const               -1.50928    0.52760  -2.861  0.00700 **
## sd1                  0.23737    0.22756   1.043  0.30385
## sd2                 -0.45323    0.34730  -1.305  0.20017
## sd3                 -0.07408    0.33568  -0.221  0.82659
## sd4                 -0.63534    0.35391  -1.795  0.08102 .
## sd5                 -0.79144    0.24802  -3.191  0.00294 **
## sd6                 -0.26559    0.21962  -1.209  0.23443
## sd7                  0.11881    0.23763   0.500  0.62012
## sd8                 -0.58257    0.31826  -1.830  0.07547 .
## sd9                 -0.39196    0.29544  -1.327  0.19296
```

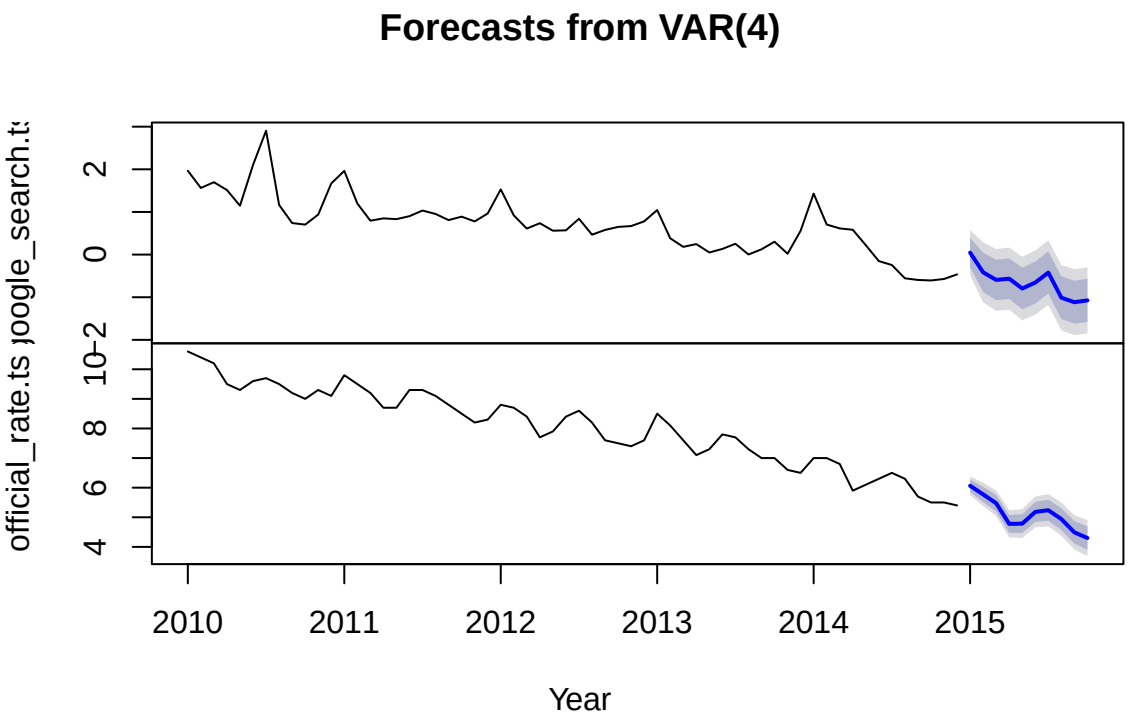
```
## sd10                -0.70815    0.23605  -3.000  0.00488 **
## sd11                -0.52133    0.19721  -2.644  0.01207 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.2706 on 36 degrees of freedom
## Multiple R-Squared: 0.8929,  Adjusted R-squared: 0.8363
## F-statistic: 15.79 on 19 and 36 DF,  p-value: 3.802e-12
##
##
## Estimation results for equation official_rate.ts:
## ================================================
## official_rate.ts = google_search.ts.l1 + official_rate.ts.l1 + google_search.ts.l2 + official_rate.ts
##
##                      Estimate Std. Error t value Pr(>|t|)
## google_search.ts.l1 -0.134541   0.098686  -1.363  0.18125
## official_rate.ts.l1  0.743175   0.146033   5.089 1.14e-05 ***
## google_search.ts.l2  0.190913   0.120764   1.581  0.12265
## official_rate.ts.l2 -0.033848   0.183964  -0.184  0.85505
## google_search.ts.l3 -0.267484   0.120168  -2.226  0.03237 *
## official_rate.ts.l3  0.017137   0.184337   0.093  0.92645
## google_search.ts.l4  0.268973   0.089635   3.001  0.00487 **
## official_rate.ts.l4  0.265599   0.158875   1.672  0.10325
## const               -0.129419   0.308287  -0.420  0.67713
## sd1                  0.863037   0.132966   6.491 1.54e-07 ***
## sd2                  0.185514   0.202931   0.914  0.36671
## sd3                 -0.009978   0.196144  -0.051  0.95971
## sd4                 -0.309893   0.206794  -1.499  0.14271
## sd5                 -0.187136   0.144920  -1.291  0.20483
## sd6                  0.301873   0.128330   2.352  0.02424 *
## sd7                  0.274704   0.138850   1.978  0.05557 .
## sd8                  0.066528   0.185966   0.358  0.72263
## sd9                 -0.192234   0.172632  -1.114  0.27285
## sd10                -0.041331   0.137928  -0.300  0.76616
## sd11                -0.245189   0.115231  -2.128  0.04027 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.1581 on 36 degrees of freedom
## Multiple R-Squared: 0.9892,  Adjusted R-squared: 0.9834
## F-statistic: 172.8 on 19 and 36 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##                   google_search.ts official_rate.ts
## google_search.ts          0.07320         -0.00352
## official_rate.ts         -0.00352          0.02499
##
## Correlation matrix of residuals:
##                   google_search.ts official_rate.ts
## google_search.ts           1.0000          -0.0823
```

```
## official_rate.ts          -0.0823           1.0000
```

```
serial.test(var.trained.14, lags.pt = 16, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.trained.14
## Chi-squared = 60.627, df = 48, p-value = 0.1044
```

```
fcst.mod3 <- forecast(var.trained.14)
plot(fcst.mod3, xlab = "Year")
title(sub = "Forecasts using Training and Test Data")
```



VAR Modelling with Seasonal Component and In-sample Training for VAR(7)

We also assess a VAR(7) model for comparison with some risk of overfitting and a significant Portmanteau Test p-value = 3.393e-05 for the training data.

```
var.trained.17 <- VAR(unemployment.ts.train, p = 7, type = c("const"),
    season = 12)
summary(var.trained.17)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: google_search.ts, official_rate.ts
## Deterministic variables: const
```

```
## Sample size: 53
## Log Likelihood: 71.596
## Roots of the characteristic polynomial:
## 1.017 0.8389 0.8389 0.8319 0.8319 0.8281 0.8281 0.7784 0.7784 0.7331 0.7331 0.4968 0.4773 0.4773
## Call:
## VAR(y = unemployment.ts.train, p = 7, type = c("const"), season = 12L)
##
##
## Estimation results for equation google_search.ts:
## ================================================
## google_search.ts = google_search.ts.l1 + official_rate.ts.l1 + google_search.ts.l2 + official_rate.ts
##
##                      Estimate Std. Error t value Pr(>|t|)
## google_search.ts.l1  0.59156    0.16050   3.686  0.00101 **
## official_rate.ts.l1  0.06557    0.30530   0.215  0.83156
## google_search.ts.l2 -0.19562    0.20673  -0.946  0.35240
## official_rate.ts.l2  0.12798    0.39109   0.327  0.74602
## google_search.ts.l3  0.09106    0.21627   0.421  0.67705
## official_rate.ts.l3  0.22803    0.33886   0.673  0.50671
## google_search.ts.l4 -0.15234    0.24008  -0.635  0.53106
## official_rate.ts.l4 -0.17227    0.27893  -0.618  0.54200
## google_search.ts.l5  0.16282    0.27360   0.595  0.55673
## official_rate.ts.l5 -0.07299    0.27295  -0.267  0.79118
## google_search.ts.l6 -0.05289    0.25358  -0.209  0.83635
## official_rate.ts.l6  0.04473    0.29453   0.152  0.88043
## google_search.ts.l7 -0.15930    0.16945  -0.940  0.35552
## official_rate.ts.l7  0.04408    0.26053   0.169  0.86691
## const               -1.67470    0.61376  -2.729  0.01105 *
## sd1                  0.39903    0.32284   1.236  0.22710
## sd2                 -0.51803    0.45949  -1.127  0.26949
## sd3                 -0.49033    0.38582  -1.271  0.21461
## sd4                 -0.50025    0.39569  -1.264  0.21695
## sd5                 -0.43041    0.31983  -1.346  0.18958
## sd6                 -0.34347    0.26378  -1.302  0.20387
## sd7                 -0.08345    0.29560  -0.282  0.77986
## sd8                 -0.66156    0.33162  -1.995  0.05623 .
## sd9                 -0.62066    0.35507  -1.748  0.09183 .
## sd10                -0.45251    0.36487  -1.240  0.22557
## sd11                -0.27052    0.26471  -1.022  0.31588
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.2207 on 27 degrees of freedom
## Multiple R-Squared: 0.9215,  Adjusted R-squared: 0.8488
## F-statistic: 12.67 on 25 and 27 DF,  p-value: 2.482e-09
##
##
## Estimation results for equation official_rate.ts:
## ================================================
## official_rate.ts = google_search.ts.l1 + official_rate.ts.l1 + google_search.ts.l2 + official_rate.ts
##
##                      Estimate Std. Error t value Pr(>|t|)
## google_search.ts.l1 -0.15030    0.10077  -1.492  0.14740
```

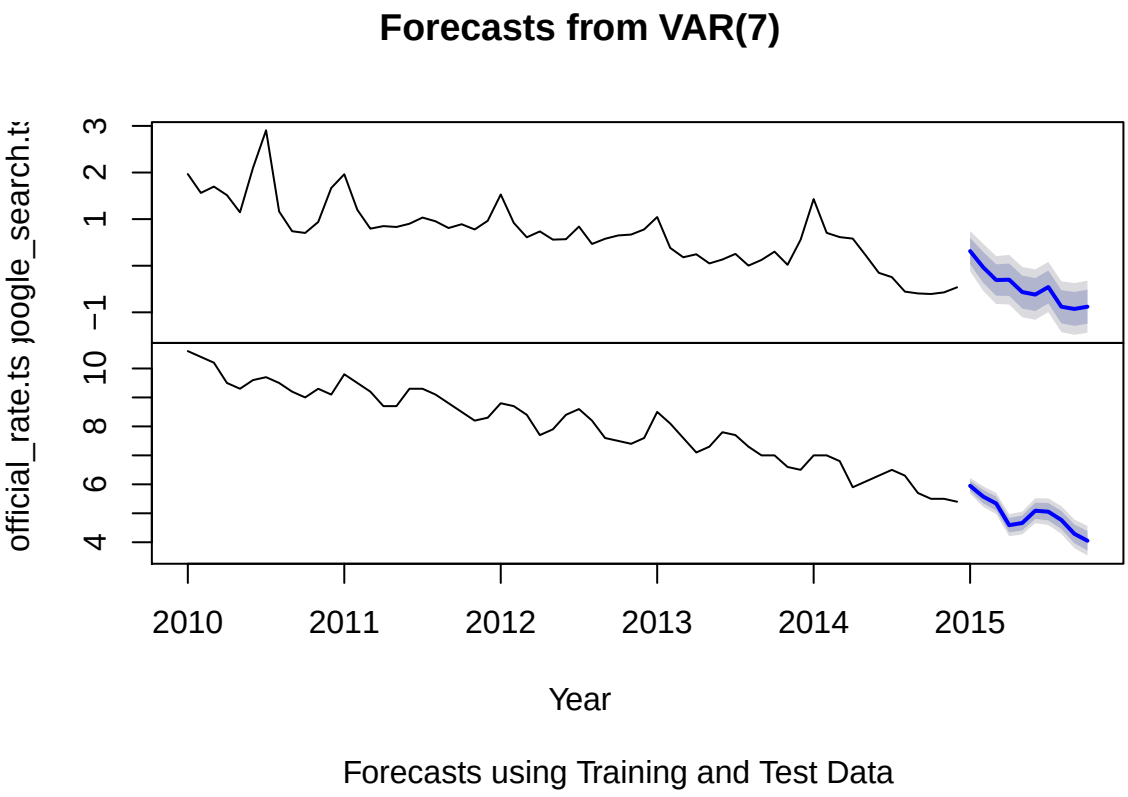
```
## official_rate.ts.l1  0.85454    0.19167   4.458  0.00013 ***
## google_search.ts.l2  0.21703    0.12979   1.672  0.10603
## official_rate.ts.l2 -0.36268    0.24553  -1.477  0.15121
## google_search.ts.l3 -0.38923    0.13578  -2.867  0.00795 **
## official_rate.ts.l3  0.26123    0.21274   1.228  0.23007
## google_search.ts.l4  0.52501    0.15072   3.483  0.00171 **
## official_rate.ts.l4  0.04092    0.17512   0.234  0.81699
## google_search.ts.l5 -0.40654    0.17177  -2.367  0.02537 *
## official_rate.ts.l5  0.34891    0.17136   2.036  0.05166 .
## google_search.ts.l6  0.29666    0.15920   1.863  0.07331 .
## official_rate.ts.l6 -0.20832    0.18491  -1.127  0.26983
## google_search.ts.l7 -0.08619    0.10639  -0.810  0.42494
## official_rate.ts.l7  0.10244    0.16356   0.626  0.53637
## const               -0.47112    0.38533  -1.223  0.23203
## sd1                  0.57228    0.20268   2.823  0.00881 **
## sd2                  0.03296    0.28847   0.114  0.90989
## sd3                  0.08491    0.24223   0.351  0.72865
## sd4                 -0.42078    0.24842  -1.694  0.10181
## sd5                 -0.11108    0.20080  -0.553  0.58468
## sd6                  0.16295    0.16560   0.984  0.33386
## sd7                  0.08665    0.18558   0.467  0.64431
## sd8                 -0.06715    0.20819  -0.323  0.74951
## sd9                 -0.10482    0.22292  -0.470  0.64199
## sd10                -0.16192    0.22907  -0.707  0.48571
## sd11                -0.46185    0.16619  -2.779  0.00980 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.1385 on 27 degrees of freedom
## Multiple R-Squared: 0.993,   Adjusted R-squared: 0.9866
## F-statistic: 153.6 on 25 and 27 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##                   google_search.ts official_rate.ts
## google_search.ts         0.048688         0.006938
## official_rate.ts         0.006938         0.019190
##
## Correlation matrix of residuals:
##                   google_search.ts official_rate.ts
## google_search.ts             1.000            0.227
## official_rate.ts             0.227            1.000
```

```
serial.test(var.trained.l7, lags.pt = 16, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.trained.l7
## Chi-squared = 80.076, df = 36, p-value = 3.393e-05
```

```
fcst.mod4 <- forecast(var.trained.l7)
plot(fcst.mod4, xlab = "Year")
```

```
title(sub = "Forecasts using Training and Test Data")
```



VAR Model Diagnostics with Out-of-Sample

The VAR(4) and VAR(7) model based on training data is then used to assess fit with out-of-sample test data.

The VAR(4) model diagnostics are not-significant. The JB-Test is not signficant and as such we cannot reject the null hypothesis of normality for the residuals. Same applies for Skewness and Kurtosis. The Portmanteau Test is not significant, while the BG Test is marginally significant p-value = 0.002395. which creates some uncertainty in terms of null hypothesis of no serial correlation in the residuals. The ACF graphs for both the official rate and the google search have a spike at lag = 1 only.

```
var.trained.l4.norm <- normality.test(var.trained.l4, multivariate.only = TRUE)
names(var.trained.l4.norm)
```

```
## [1] "resid"  "jb.mul"
```

```
var.trained.l4.norm
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object var.trained.l4
## Chi-squared = 3.5149, df = 4, p-value = 0.4756
##
```

```
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.trained.l4
## Chi-squared = 1.0945, df = 2, p-value = 0.5785
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.trained.l4
## Chi-squared = 2.4203, df = 2, p-value = 0.2981
```

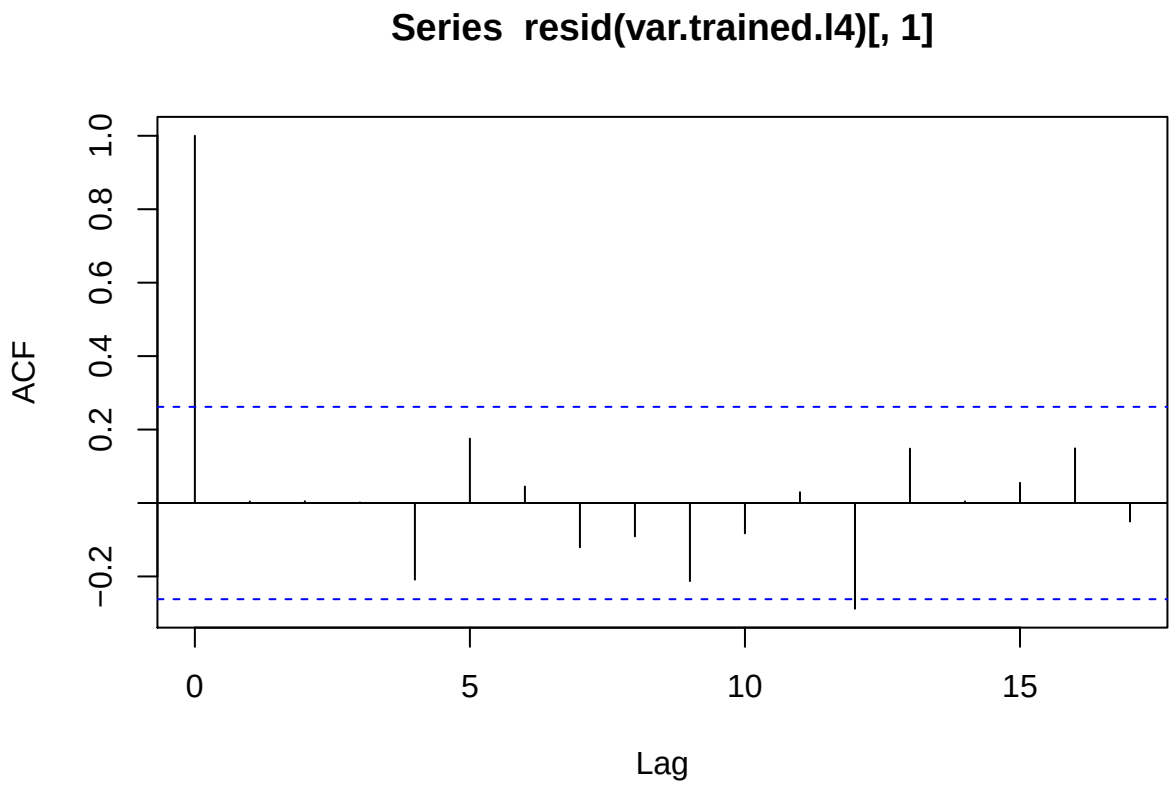
```
var.trained.l4.pt.asy <- serial.test(var.trained.l4, lags.pt = 16,
    type = "PT.asymptotic")
var.trained.l4.pt.asy
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.trained.l4
## Chi-squared = 60.627, df = 48, p-value = 0.1044
```

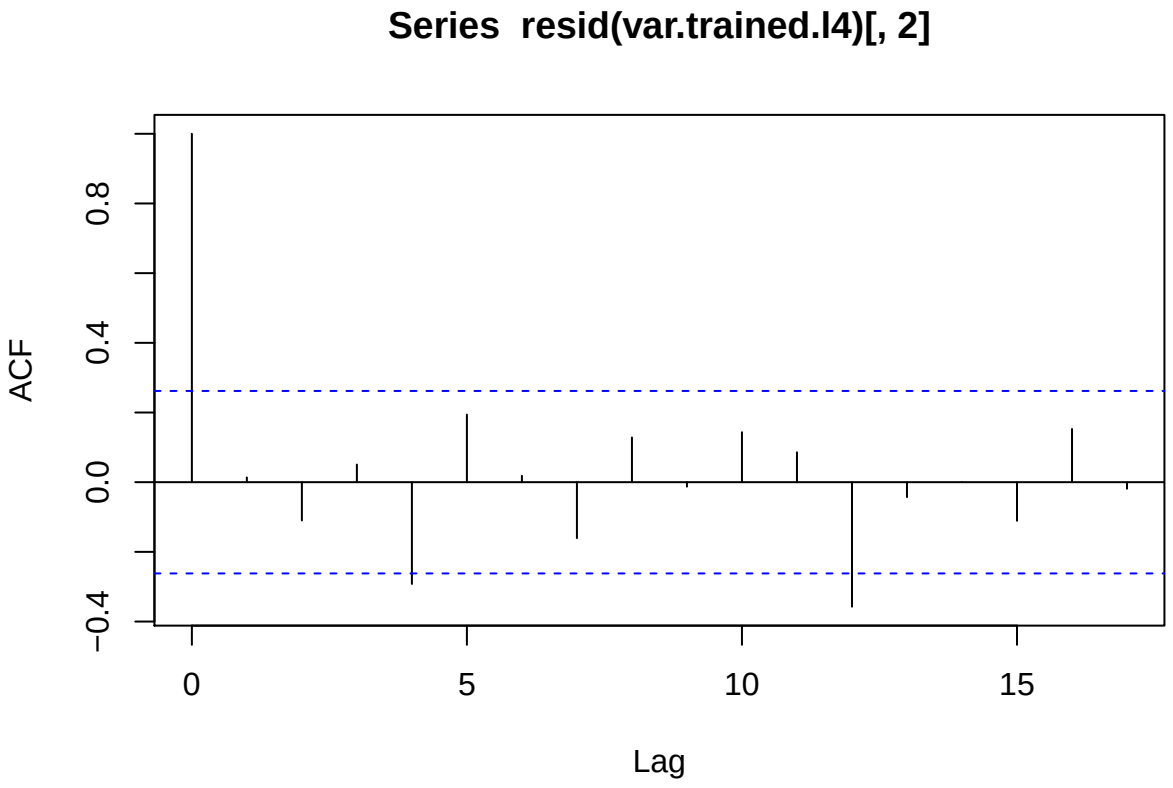
```
serial.test(var.trained.l4, lags.bg = 16, type = "BG")
```

```
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object var.trained.l4
## Chi-squared = 100.56, df = 64, p-value = 0.002395
```

```
acf(resid(var.trained.l4)[, 1])
```




```
acf(resid(var.trained.l4)[, 2])
```



The VAR(7) model we get similar non-significant p-values for the JB-Test, Skewness and Kurtosis Tests, while Portmanteau Test and BG Test have significant p-values. The latter means that we can reject the null hypothesis of no serial correlation in the residuals. We can also see that in the ACF which has multiple significant lags.

```
var.trained.l7.norm <- normality.test(var.trained.l7, multivariate.only = TRUE)
names(var.trained.l7.norm)
```

```
## [1] "resid"  "jb.mul"
```

```
var.trained.l7.norm
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object var.trained.l7
## Chi-squared = 7.9785, df = 4, p-value = 0.09237
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.trained.l7
## Chi-squared = 6.9243, df = 2, p-value = 0.03136
##
##
```

```
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.trained.l7
## Chi-squared = 1.0541, df = 2, p-value = 0.5903
```

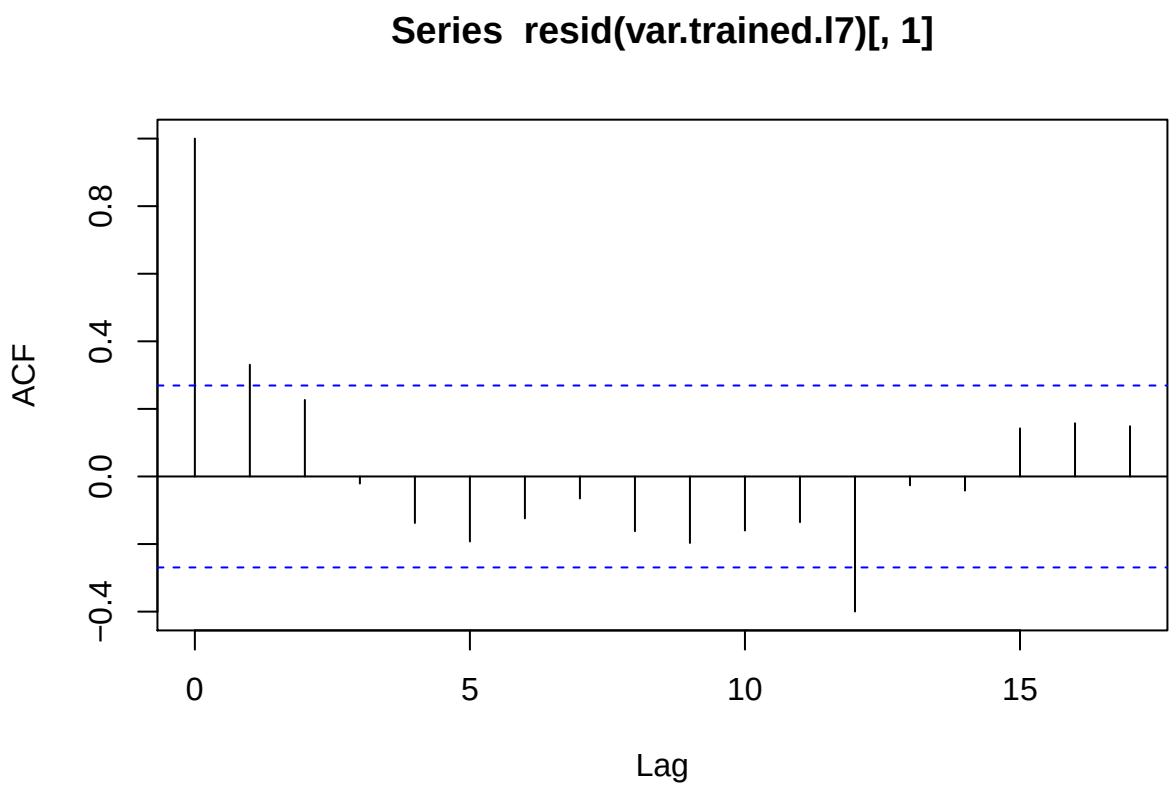
```
var.trained.l7pt.asy <- serial.test(var.trained.l7, lags.pt = 16,
    type = "PT.asymptotic")
var.trained.l7pt.asy
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.trained.l7
## Chi-squared = 80.076, df = 36, p-value = 3.393e-05
```

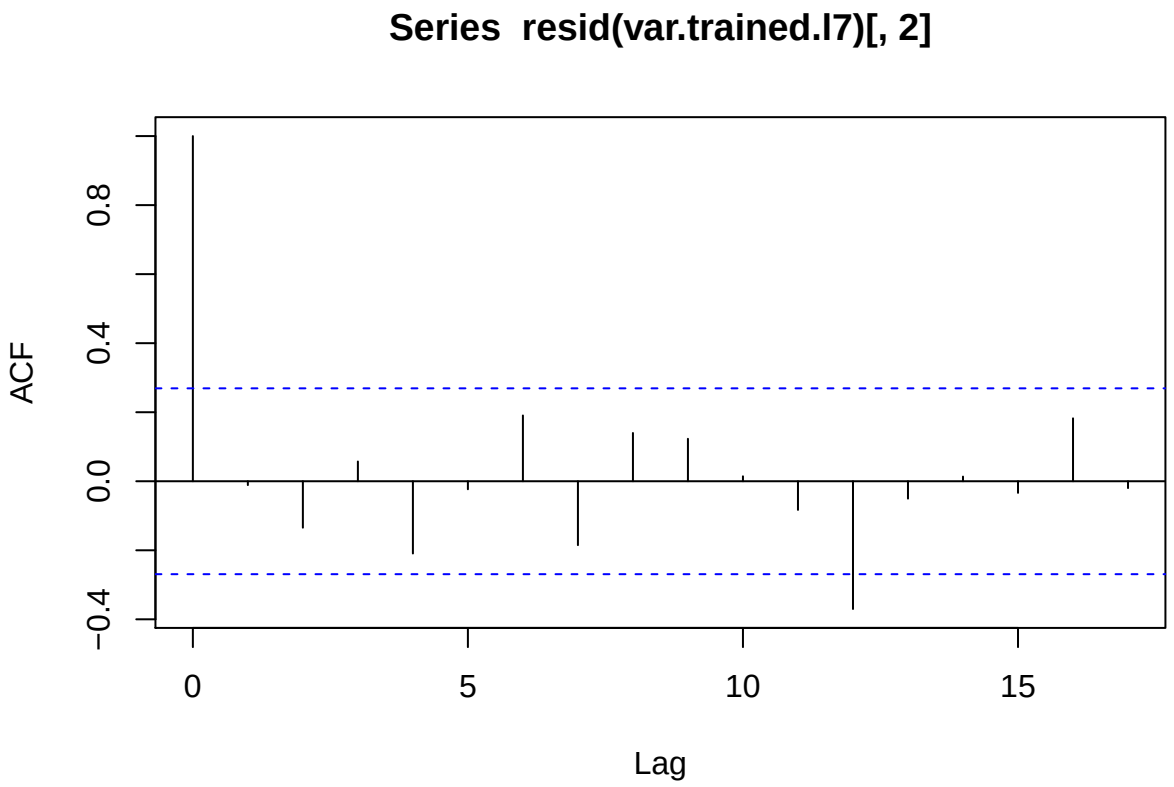
```
serial.test(var.trained.l7, lags.bg = 16, type = "BG")
```

```
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object var.trained.l7
## Chi-squared = 106, df = 64, p-value = 0.0007567
```

```
acf(resid(var.trained.l7)[, 1])
```



```
acf(resid(var.trained.l7)[, 2])
```



Model Selection

After examining the results of the VAR models, var.mod.lag1 was selected for use in forecasting. The model was selected because it has the lowest SC, essentially no serial correlation in the residuals and not locked into a temporary trend that we happen to have in the dataset but that we do not expect to continue for long (cannot go below zero). Other models could also be used, but are overly complex and provide little to no benefit and risk of overfitting.

Forecasting with In-Sample VAR Model

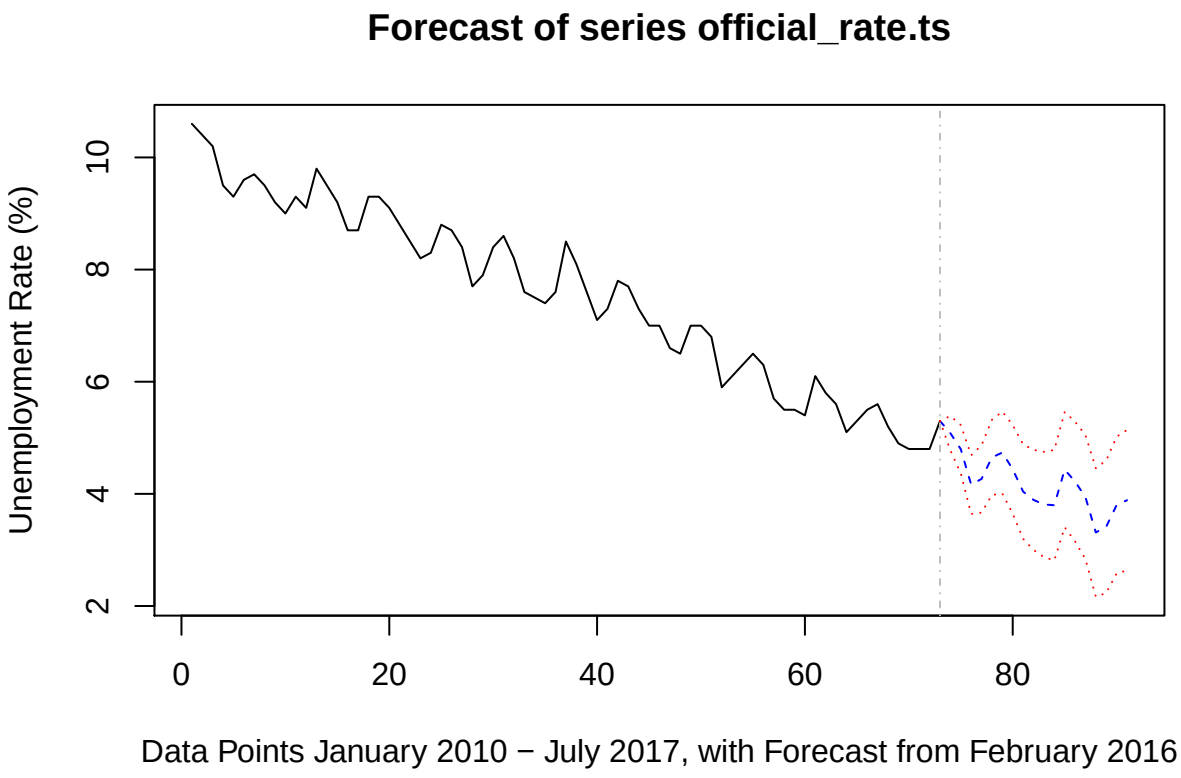
Make a Forecast for 6 months ahead using the in-sample model 1. As our model was based on the combines time series ending January 2016, we need to add another 12 months to get to a 6 month forecast for 2017. The results are shown in the table below and in graph.

```
var.mod.lag1.forecast <- predict(var.mod.lag1, n.ahead = 18)
var.mod.lag1.forecast
```

```
## $google_search.ts
##             fcst     lower       upper        CI
##  [1,] -0.8221089 -1.341921 -0.30229687 0.5198120
##  [2,] -0.9614211 -1.576418 -0.34642386 0.6149972
##  [3,] -0.9631610 -1.617749 -0.30857342 0.6545876
##  [4,] -1.1595613 -1.836579 -0.48254314 0.6770181
```

```
##  [5,] -1.0164600 -1.710088 -0.32283212 0.6936279
##  [6,] -0.7988592 -1.507004 -0.09071466 0.7081446
##  [7,] -1.2714932 -1.993315 -0.54967172 0.7218214
##  [8,] -1.3451686 -2.080246 -0.61009078 0.7350778
##  [9,] -1.2991695 -2.047222 -0.55111695 0.7480525
## [10,] -1.3033660 -2.064158 -0.54257391 0.7607921
## [11,] -1.0110137 -1.784328 -0.23769943 0.7733143
## [12,] -0.5795657 -1.365194  0.20606265 0.7856284
## [13,] -1.1577760 -1.955518 -0.36003420 0.7977418
## [14,] -1.3234615 -2.133123 -0.51379986 0.8096616
## [15,] -1.3409295 -2.162325 -0.51953448 0.8213950
## [16,] -1.5466656 -2.379615 -0.71371658 0.8329490
## [17,] -1.4090618 -2.253392 -0.56473136 0.8443304
## [18,] -1.1946537 -2.050200 -0.33910789 0.8555458
##
## $official_rate.ts
##           fcst    lower    upper        CI
##  [1,] 5.081423 4.773497 5.389349 0.3079257
##  [2,] 4.797801 4.367310 5.228291 0.4304908
##  [3,] 4.165151 3.641726 4.688576 0.5234249
##  [4,] 4.266419 3.664963 4.867874 0.6014552
##  [5,] 4.651369 3.981256 5.321483 0.6701133
##  [6,] 4.736530 4.004367 5.468693 0.7321629
##  [7,] 4.438482 3.649265 5.227699 0.7892170
##  [8,] 4.040509 3.198192 4.882825 0.8423164
##  [9,] 3.892580 3.000400 4.784759 0.8921796
## [10,] 3.811343 2.872015 4.750672 0.9393287
## [11,] 3.796789 2.812631 4.780946 0.9841575
## [12,] 4.432243 3.405271 5.459214 1.0269718
## [13,] 4.220485 3.152470 5.288500 1.0680150
## [14,] 3.941187 2.833702 5.048672 1.1074848
## [15,] 3.311365 2.165820 4.456909 1.1455443
## [16,] 3.414560 2.232230 4.596889 1.1823299
## [17,] 3.800897 2.582940 5.018855 1.2179573
## [18,] 3.887120 2.634595 5.139645 1.2525254
```

```
plot(var.mod.lag1.forecast, names = "official_rate.ts", ylab = "Unemployment Rate (%)",
    xlab = "Data Points January 2010 - July 2017, with Forecast from February 2016")
```


Forecast of series official_rate.ts
Unemployment Rate (%)
2
4
6
8
10
0
20
40
60
80
Data Points January 2010 − July 2017, with Forecast from February 2016