Simulation of Network Traffic and Application of ARIMA Time Series Method

STAT1005 Project **Jeevanpreet Singh (20153780)**

Background:

As the internet becomes more complex in its infrastructure to be able to sustain tremendous amounts of network pressure, network traffic increases proportionately (Jung et al., 2006). IP workstations and servers are accessed constantly by multiple unique users at any given time, and thus it is important for any network provider or network management system to ensure that their resources are optimised and their workstation remains functional for use at any given time (Benson et al., 2010) (Cortez et al., 2012).

Problem Statement:

The objective of this project is to, using data from major networks and IP workstations, simulate network traffic and gather an estimation for initial parameters and variation within simulated and fitted data. This will be achieved by applying the ARIMA Time series modelling, as well as built in R packages, which will synthetically generate data for simulation based on a fitted model. The fitted model will be created by real-life data on network traffic and will serve as means of comparison to analyse the accuracy of the simulation.

Literature Review:

The very basis of analyzing network traffic is by collecting the amount of packets that are being transferred at a specific time (Jung et al., 2006). This is important to distinguish whether internet traffic has certain trends, positions of maximum and minimum transfers and types of traffic, all of which dictate how network traffic is derived. Time series involves a sequence of observations ordered in time (Jung et al., 2006)(Cortez et al., 2012). This is the best way to analyse network traffic as the systems are maintained at millisecond intervals as millions of packets of data are sent every day. ARIMA (Auto Regressive Integrated Moving Average) time series modelling combines multiple models to demonstrate the best model for prediction that is viable to simulate and predict network traffic (Prabhakaran,2019).

ARIMA models involve the AR (AutoRegressive) order p , MA (Moving Average) order q, and differencing order d, which all combine to display a time series. Network traffic fitted to this time series must be stationary (Labarr, 2019), either by differencing or transformation. The ARIMA models which have been derived for this project are ARIMA(0,0,1), ARIMA(1,0,0) and ARIMA(1,0,1). Each with its own theoretical equation which will determine the simulation models. The general ARIMA model follows (Prabhakaran, 2019):

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

The values are both AR and MA values, each with their specific coefficients. These have been derived as the equations used for simulation.
ARIMA(0,0,1): macoeff x ma[i-1] + ma[i]

ARIMA(1,0,0): arcoeff x ar[i-1] + ma[i]

ARIMA(1,0,1): arcoeff x ar[i-1] + macoeff x ma[i-1]

Methodology:

The methodology involves satisfying both aspects of the problem statement; Parameter Estimation and Variation. These two measurements ensure that the simulation is carried out effectively and does not deviate too far from the original values of IP addresses. The following objectives chronologically dictate the methods of analysis that are being undertaken in the project

1. **Exploratory Data Analysis:** Before the simulation occurs, there must be an understanding of the data as well as the best fitted values which will create the appropriate time series analysis Exploratory data analysis will occur to have a better understanding of which variables in the original fitted network traffic data are important in analysing what occurs in the in the time series analysis. This will involve basic exploratory analysis and some advanced select and visualisation functions which will demonstrate the significant points of interest in the dataset. Certain functions will include:
    - Plotting and interactive visualisation of original variables
    - Sub-setting and log-transformation of data for better ARIMA modelling fit
    - Select functions to identify interesting points in the data.
    - Introducing and solving stationarity assumptions by differencing

2. **Fitted ARIMA Application:** The application of time series method is utilised to simulate the data *over time*, whether it be seconds, minutes, hours, or days. It comprises of multiple statistical methods used to analyse data on the variable time. Different time series methods will be used (ARIMA comprises of multiple smaller methods) on a daily time scale to accurately measure and forecast TCP/IP network connections. Depending on the correlation of values, there will be multiple ARIMA models constructed and modelled which will provide the necessary fixed parameters that are required for the simulation algorithms. Certain functions will include:
    - Seasonality significance in data modelling
    - Autocorrelation functions and visualisations to dictate which ARIMA models to conduct simulation on.
    - Application of ARIMA functions to apply ARIMA models
    - Comparison of error estimates, box-test p-values and coefficients to dictate accuracy of each forecasting model.
    - Residual analysis for further accuracy prediction

3. **Simulation:** The linear time series model, namely ARIMA, can be used to generate a program that can be used to simulate simple network traffic to synthesise data, which will then be applied to backwards estimation to measure the accuracy of the simulation in regards to the deviation between synthetic data and original data. Parameter estimation will occur by generating a program which takes fixed parameters of variance, Auto-Regressive coefficient, Moving-Average coefficient, Auto-Regressive order (p), Moving-Average order (q) and differencing order (d). Certain functions will include:
    - Applying theoretical ARIMA equations into each algorithm, dependent on the values of each model.
    - Iterating and generating synthetic data for simulation.
    - Applying loops and specific knowledge of the simulation to distinguish multiple unique ARIMA models for backwards estimation.
    - Comparison of fitted and synthetic data to ensure estimation is accurate

    3.1 **Backwards Parameter/Coefficient Estimation + Variation:** Once synthetic data is generated, back estimation will occur to find the value of these parameters. This involved taking every fitted ARIMA model and each synthetic sets of data and measuring their accuracy in predicting network traffic. Each ARIMA model will generate different sets of data with different variations which indicate the deviation between simulated data and synthetic data. Variation will be measured through two methods. A non-linearity test will conducted, as well as the calculated difference (residual) between synthetic data and original fitted data. Certain functions will include:
    - Residual difference between synthetic and fitted data
    - Nonlinearity testing to dictate whether the ARIMA models are adequately simulated in accordance to the fitted parameters.
    - Plot and visualisation of AR/MA roots

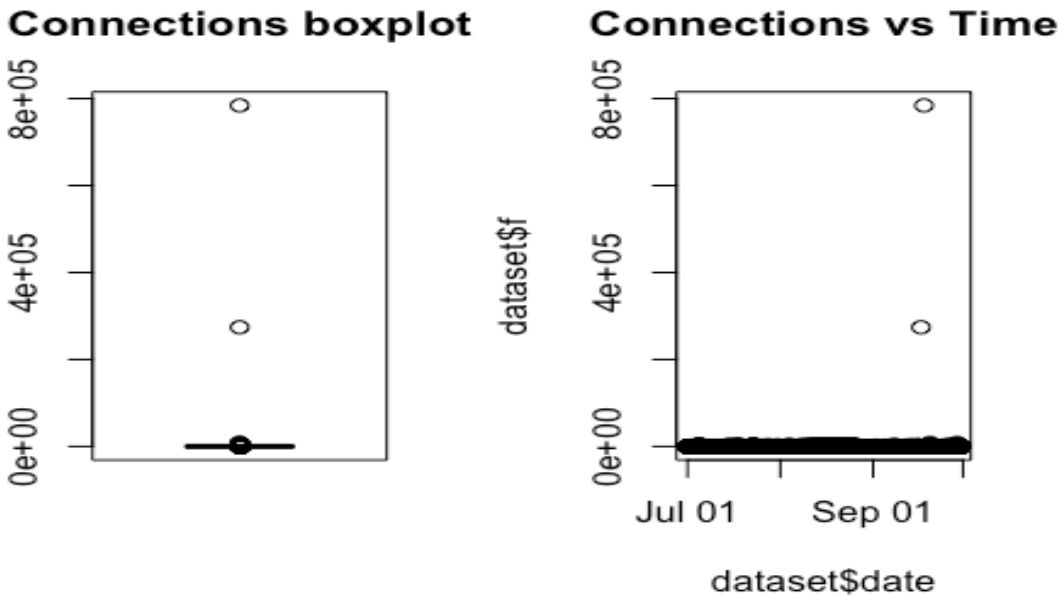
Further aspects of the methodology will be included in each objective.

1. EXPLORATORY ANALYSIS

To begin exploratory analysis, it's important to distinguish the specific variables in the dataset and see how they relate to the objectives of the project (SEE 1.3).

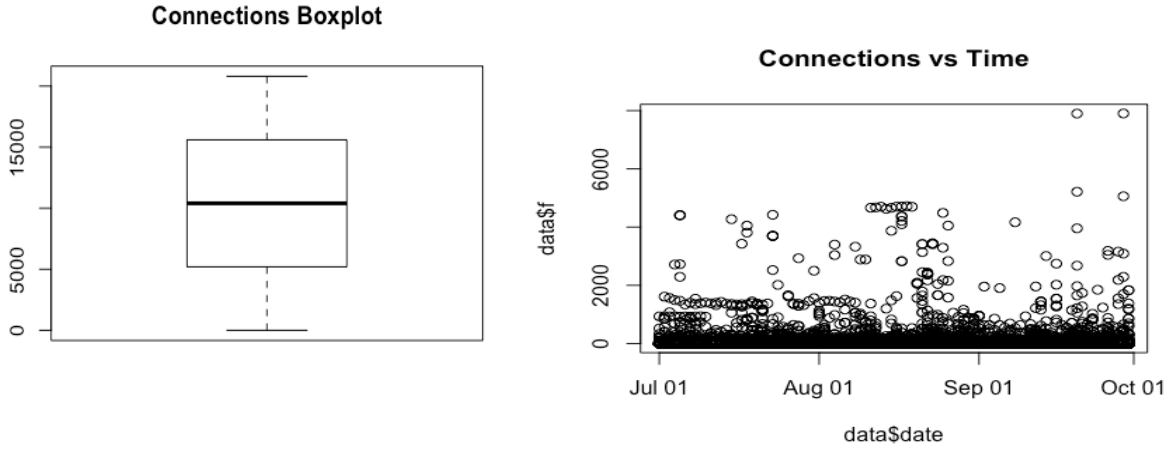
There appears to be 4 variables. Looking at the data structures, there are three numerical variables and one date time variable. This is important to note considering time series analysis involved a specific variable being fitted according to time. Looking at the (KAGGLE SOURCE), l_ipn is the ip station that the network traffic is engaged, while the asn is the unique ip address of the machines involved in the network traffic. As such, both are considered numerical categorical and do not hold any distinguishable value when fitted with time. The 'f' variable is significant as it is a quantitative measure of connections made

per machine at a specific time. The f variable will be the variable fitted against time (SEE 1.4).

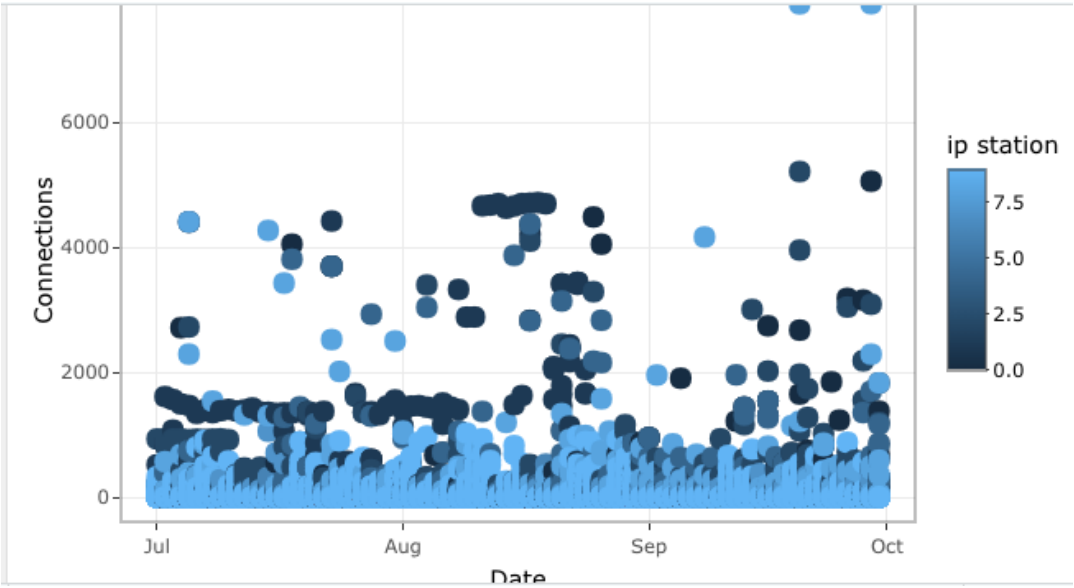


The boxplot and regular plot seem to show two specific variables which skew the values drastically. These values can be identified by applying advanced tidyverse functions (SEE 1.5) to arrange the data by descending 'f' values and forging an assumption about how far the outliers are in range with all the other variables.

By analysing the highest ten values in the dataset, there is an even further indication that there are 2 peculiar values that reach far further than any other. These two are over 10 000 while every other value is between 0 and 10000. As such, we will subset the data to exclude these two variables and examine the output of the boxplot to see if the mean is more consistent, as well as the ranges (SEE 1.6).



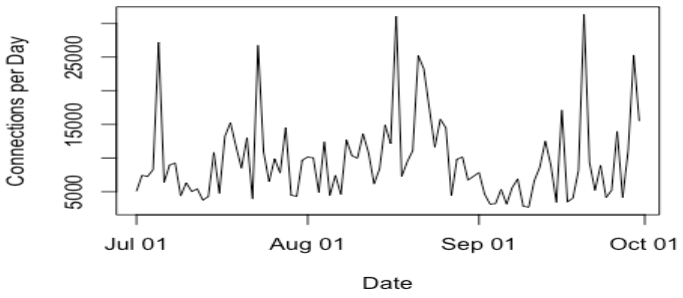
The data looks far more consistent, and although there are a couple of values that are still quite high, it would be a safer measure to leave them in. I opted to maintain every 1, 2, 3 and 4 digit number in the dataset. We can further enhance the visualisation (SEE 1.7) by categorically separating each data point into its separate workstations. The interactive functions have been omitted for now for the sake of slow operating speed. The interactive visualisation simply displays that there is no consistency between which ip stations create the most connections per day. It is fairly independent, with many ip stations creating many different connections at many different times. This is important to note considering that the values should not display much colinearity for time series analysis to occur.



A tidyverse function (SEE 1.8) clearly indicates that there can be a total sum of connections for each date recorded. However, considering time series involves a number of days with each datapoint of the variable plotted against it's specific time, it would be inefficient to repeat this function as firstdate, seconddate, thirddate,etc. There are 92 dates recorded (SEE 1.9), so it would be time consuming to make 92 variables and plot them one by one. To fix this, we will use the aggregate function to aggregate the number of connections each day to a specific date (SEE 1.10).

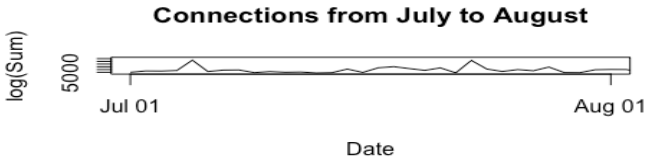
```
#1.10
fdata <-subset(data, select= -c(l_ipn,r_asn))
```

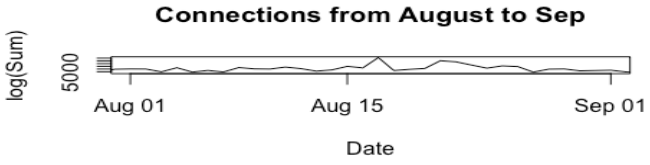
This subset excludes the ip workstation and unique ip indicator, as we're only focused on the time variable and connections variable.



This plot (SEE 1.11) shows the total sum of connections per day, ignoring which ip stations and unique ip addresses in which the connections are made from (they are already considered irrelevant, both through the objectives of the project and the interactive visualisation showing independence). This is the first piece of data from the variables which can attest to a time series model being made, as a numerical variable is fitted against a time variable. No changes will be done to the time variable as the objective states that the analysis will be carried on a daily frequency. The plot does show a few inconsistencies.

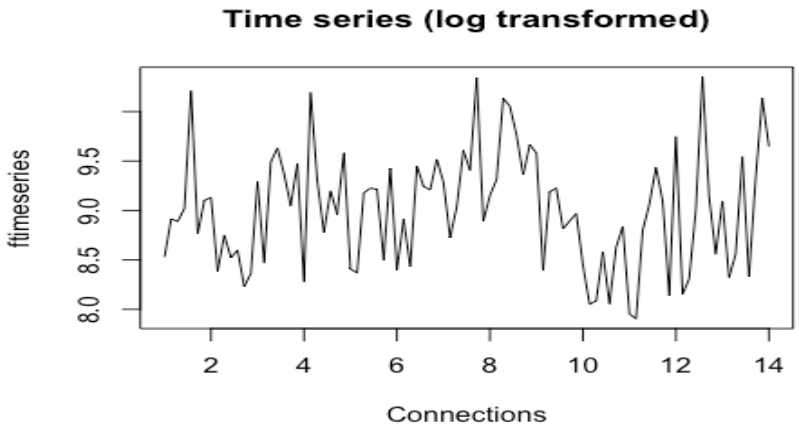
Stationarity is required for a successful time series model. We've selected certain periods in time between the two variables, one from July to August and the other from August to September (SEE 1.12). Stationarity involves the notion that the mean and variance of any time series must be consistent in which any previous data point will have the same period of values as the current one. In other words, every section of the data must appear slightly similar.





Stationarity involves of two distinct principles, that the mean and variance are both consistent (Labarr, 2019). This allows us to understand that the ARIMA model will not simply follow inconsistencies in data, thus a more accurate model will be produced. Stationarity is evident and can be identified in any set of data, and dependent on the type of model being used, it may significantly afflict the accuracy of the forecasting prediction model beign fitted. This plot demonstrates that the mean and variance from each period are not consistent with the other. Thus, stationarity is violated and a time series cannot be

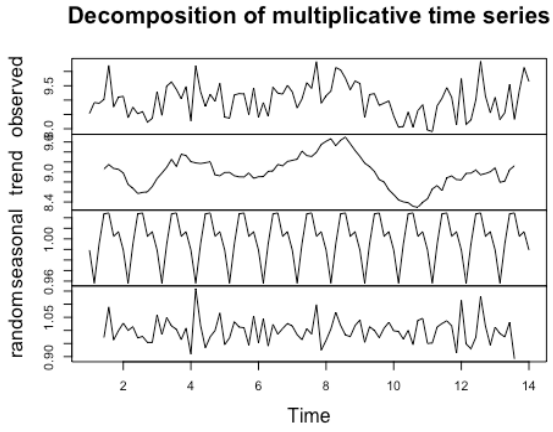
fitted. The simplest way to make the model stationary is by differencing the order of the stationarity, or applying transformations to make the variation between each value more consistent. To do this we convert it into a time series variable using the ts() function (SEE 1.13).

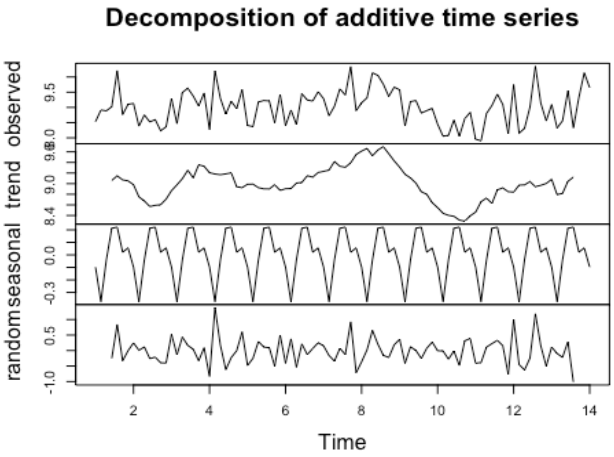


It's important to know how the log transformation is important for achieving stationarity (SEE 1.13). Before the transformation was applied, values would have a consistent mean, but variance was still fairly inconsistent as some were bigger than others. Applying a log transformation has made the variance far more consistent, in which the range between the highest and lowest values of specific periods can be attributed to being more stationary.

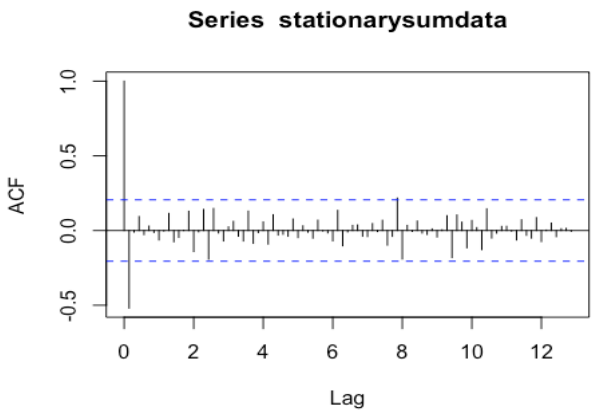
2. TIME SERIES ANALYSIS

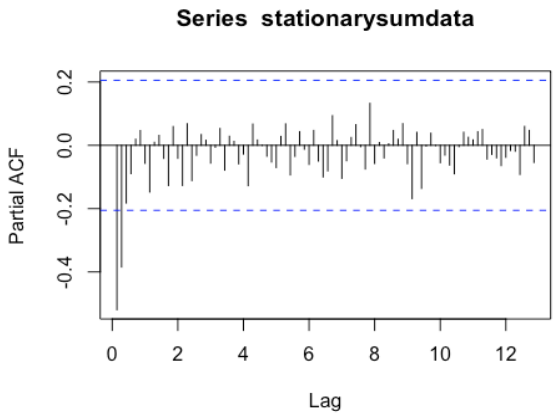
Before we being applying an ARIMA model, it's important to note it's seasonality. Seasonality is similar to stationarity but for the underlying trends in the data, in which it displays some sort of correlation between the values at certain times (Otexts, 2019) . This is what makes an ARIMA(p,d,q) model into a ARIMA(p,d,q) (P,D,Q)m model, which simply adds parameters to explain the seasonal trends. Non-seasonal data is easier to analyse and forge interpretations from, thus seasonality must be removed. To make the modelling less difficult, decomposition required to analyse seasonality and omit it from data (SEE 2.1).





There is two models two measure components, additive and multiplicative. The simple difference being that one adds the variables, when the other multiplies them. For now, we will use just the additive model. Both models display that there is some seasonal trend to the data being displayed, indicating that there could be some correlation and non-stationarity when ARIMA time series modelling is conducted.

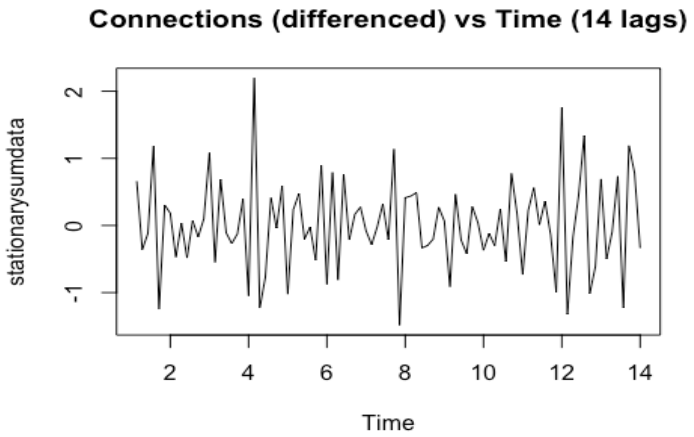




ACF and PACF correlogram (SEE 2.2) indicate the amount of correlation that each variable may have in a period before it. This explains the lag variable, which is the distance between each variable in the time series. ACF and PACF visualisations can indicate significant k values, which in turn may allow us to generate an assumption on which model is the most statistically significant. It follows that the values that are outside the threshold (dotted blue line) are statistically significant, in which the PACF values dictate the Auto Regression order and ACF values dictate the Moving Average order. The first value takes the default value of 1.0, as it sets the standard for the other lags that follow.

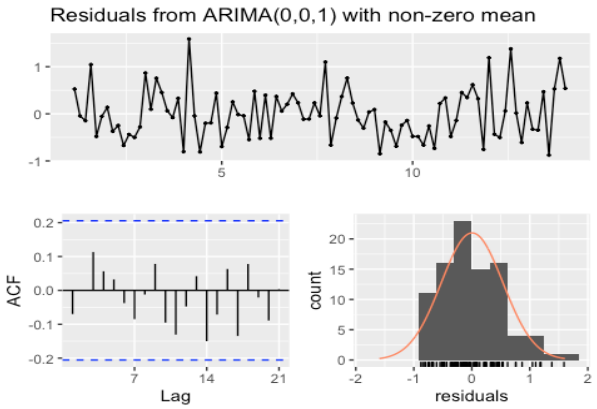
It follows that until there is no significant autocorrelation as the lags of data (range) increases. There is minimal linear association between the datapoints. Interpreting this data from the ACF, this displays the notion that k=1 is the only statistically significant point,

allowing us to predict that the best arima model is likely to have a q-value of 1. The PACF plot displays another cutoff at k=1, thus a likely p for ARIMA(p,d,q) could be p=1. In this case, we have a primary choice of models to apply to the data. AR(1) or MA(1). When subjecting this into the context of ARIMA, remembering that no differencing is required due to the values being stationary, the ARIMA model would display ARIMA(1,0,0), ARIMA(0,0,1) or ARIMA(1,0,1)

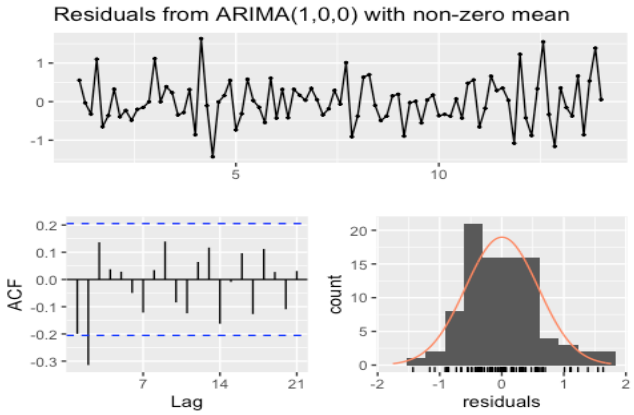


The data is far more stationary than previously (SEE 2.3). By log tranforming, decomposing and removing the seasonality, as well as the application of differencing, the data appears to be more consistent in mean and variance which centres around 0. The data appears to be stationary.

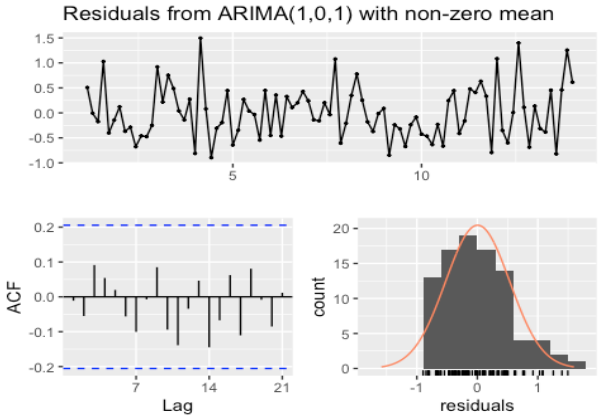
It's important to see what order fits the best for the data presented. auto.arima() (SEE 2.4) function simulates each possible arima model to determine the lowest AIC (least amount of error, used to compare models) and output the best ARIMA model to fit. The best model chosen by this function is ARIMA(0,0,1). This means that we have no auto regressive order and differencing order, but we have a single moving average order. In other words, p= 0, d = 0 and q =1. However, we will continue to model the data using the values we got from the ACF and PACF plots. These include ARIMA(1,0,1) and ARIMA(1,0,0). This allows us to generate discussion about the accuracy of simulated data in relation to how their parameters are back-estimated to the fitted network traffic values.



We've used a first order method to measure the ARMA model, with values p=0, d=0 and q=1 (SEE 2.5). This quantitatively gives the best fit for the ARIMA model in the ARIMA(0,0,1) format. This can also be referred to as MA(1). Looking at the plot displayed, the values for the mean centres around zero, are close to constant variance and is nearly normally distributed. This indicated that when simulation occurs, the synthetic data should also fit an approximately normal distribution, with similar statistical significance in autocoreelation. Furthermore, the p-value exceeding 0.05 indicates that it is statistically a good fit, rejecting the null hypothesis that it's not.This ARIMA model provides an adequate fit. Now it's important to repeat the process for the next two ARIMA models (SEE 2.6)



(SEE 2.6) The Ljung-box test, which uses the null hypothesis that the model not a good fit, can be used to dictate how accurate the data is. It is well below the 5% confidence level, thus the null hypothesis is accepted, and the model is not a good fit. Furthermore, The ARIMA(1,0,0) appears to give fairly similar residuals. They are not nearly as normally distributed as the previous model, and there appears to be a statisticall significant ACF value at k=1. This indicates that, for a better fit, a moving average of 1 should be applied. This is how both previous models are adapted to fit the ARIMA(1,0,1) model. In other words, an AR(1) and MA(1) are combined, with no differencing order, to create an ARMA(1,1) model. This is most appropriately known as ARIMA(1,0,1).

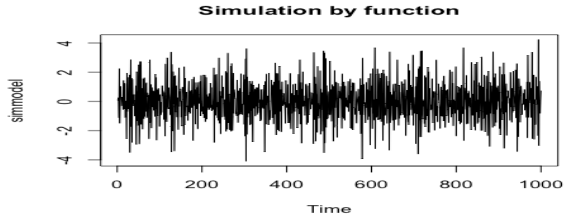


(SEE 2.7) The p-value is well above the 5% significance level, thus the model is an adequate fit. Furthermore, there are no statistically significant k-values when analysing the correlation of lags, thus no more auto regressive order, differencing order, or moving

average order values need to be used. The distribution also appears to be more normally distributed. As such, we can conclude that ARIMA(0,0,1) and ARIMA(1,0,1) are good ARIMA models for the fitted data, while the ARIMA(1,0,0) is not. We now know the order of the ARIMA model we are going to use. In other words, we know the values for p,d and q, as well as the AR and MA coefficients. We can finally start to simulate synthetic data through simulation.

3. SIMULATION:

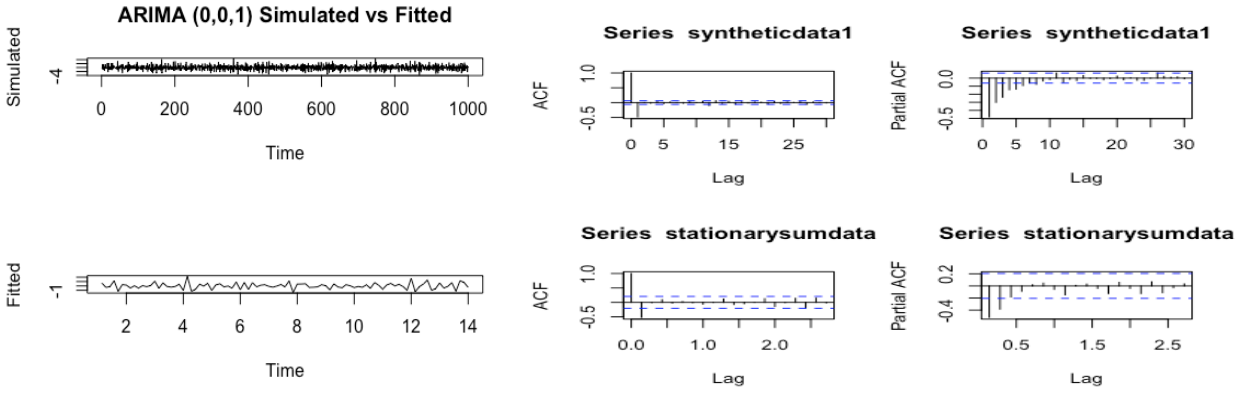
Part of the objectives states that we aim to simulate the values for each specific ARIMA model, then utilise that specific model to back estimate the parameters and conduct diagnostics checking.



We could very well use the built in function arima.sim() to allow simulation of an arima model and generate synthetic data (SEE 2.8), however one of the objectives of this project is to demonstrate knowledge in simulation by hand. As such, iterations and loops will be utilised to generate the data. There will be three sets of synthetic data measured for each ARIMA model (ARIMA(1,0,0), ARIMA(0,0,1) and ARIMA(1,0,1)). It's important to note that we already know the values for the auto-regressive order (p), differencing order (d), moving average order (q), Moving Average coefficients (MA) and Auto-regressive coefficients (AR), as mentioned in the methodology section. We can now begin to simulate each model and generate synthetic data from each.

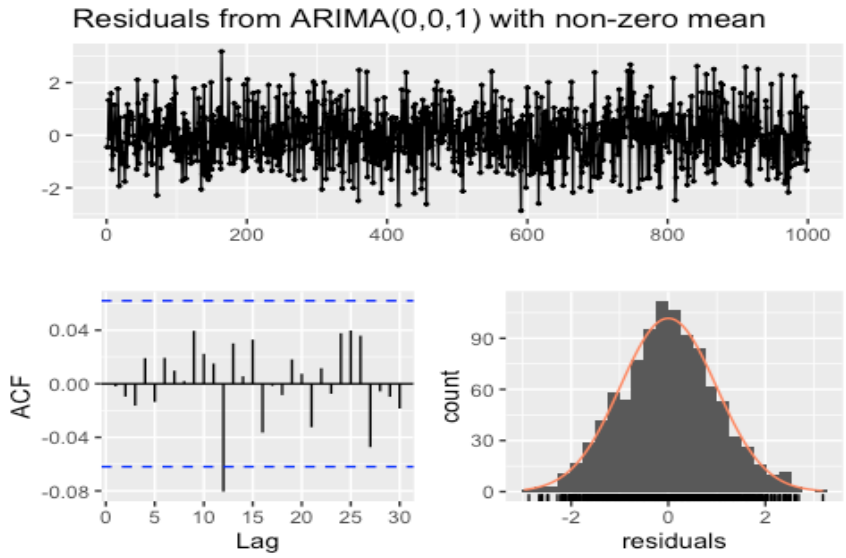
Theoretically, ARIMA(0,0,1), ARIMA(1,0,0) and ARIMA (1,0,1) have three distinct equations that follow.

We begin by simulating an ARIMA(0,0,1) model.



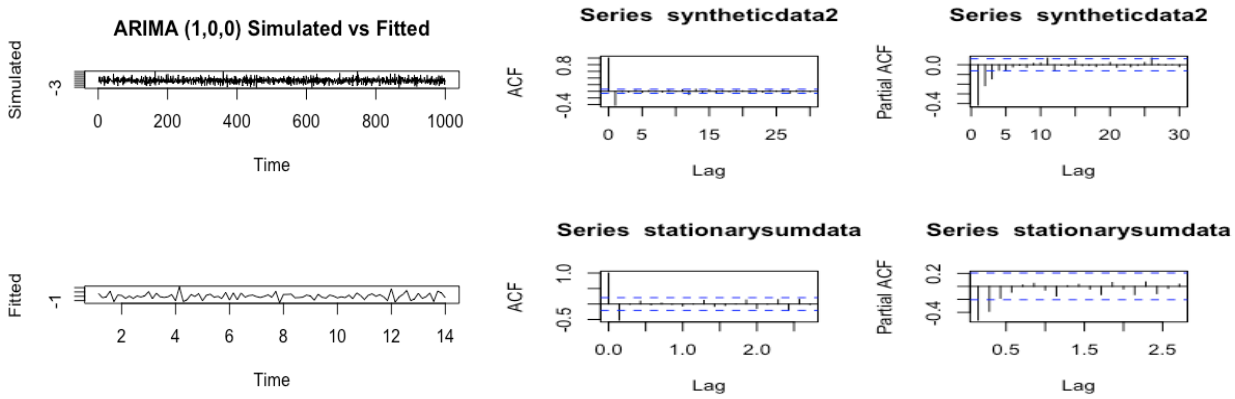
The autocorrelation plots (SEE 2.9) show significant similarities between the simulated and fitted data. Both ACF k-values cut off at k=1, thus the synthetic data was successful in back-

estimating a q value of 1 in ARIMA(p,d,q). The PACF has a few statistically significant values, although it follows the trend that as the lag distance gets larger, the statistical significance decreases. As such, the synthetic data becomes much less correlated at further lags, much like the fitted data.



(SEE 2.10) By measuring the residuals in both the synthetic and fitted ARIMA(0,0,1) models, we can see that the synthetic data achieves a far more normal distribution of residuals. This indicates that the model is very accurate. There is a lag at k=12, possibly indicating at some monthly correlation (12 months).

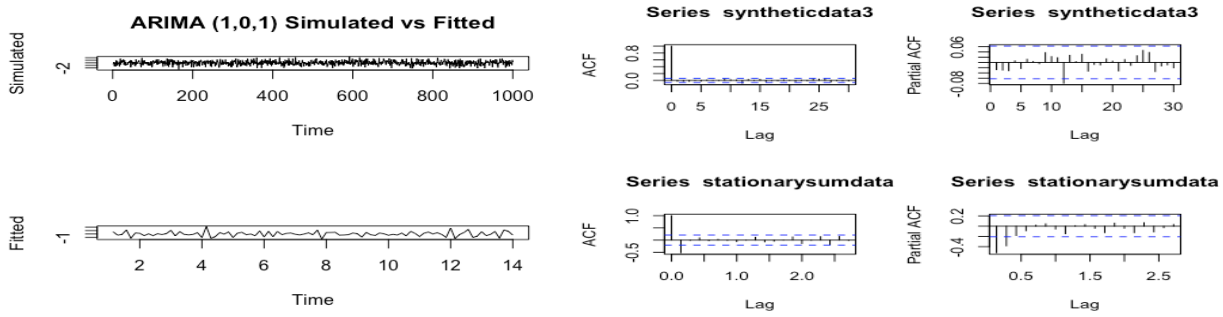
We have successfully simulated an ARIMA model by hand using the ARIMA(0,0,1) model that was auto-generated by the auto.arima() function. However, as previously mentioned when inspecting the ACF plots for the initial fitted values (from the network traffic data), there could be a few more possible models. We will repeat the same process to simulate an ARIMA(1,0,0) model, often considered the most common ARIMA model, and compare its accuracy with the previous ARIMA(0,0,1) model.



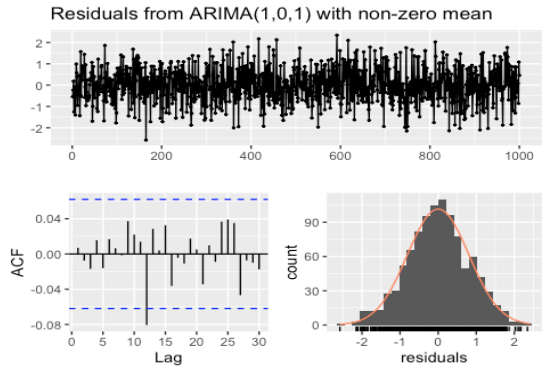
(SEE 2.11) The ARIMA(1,0,0) simulated data shows similar autocorrelation with the fitted data. There is a significant k value at k=1, indicating a successful backestimation of the

moving average value (p=1). There are more statistically significant partial correlations, however the same trend as (SEE 2.9) where a larger lag distance indicates less autocorrelation between the specific values.

(SEE 2.12) All the trends in residual data are similar. Box-test, which null hypothesis says there is no lack of fit for the model, is rejected for the fitted variables but not simulated variables. In other words, the ARIMA(1,0,0) model is an adequate fit for the simulated data (as expected, as it is the order that we simulate the data from) but not the original network traffic data. A normal distribution is evident for its simulated residuals, but the statistical k-value is at k=1. It's important to note that the p-value from the box-test is far below 5% significance level. Therefore, we accept the null hypothesis that this model is not a good fit.



(SEE 2.13) The autocorrelation between the synthetic and fitted ARIMA(1,0,1) show fairly different values. The PACF and ACF plots both show statistical significance at k=12 for the simulated data while it is k=1 for the fitted data. As such, this model is not as appropriate as the ARIMA(0,0,1) model for back estimation of parameters, although it does have a very good fit of data. This can be reinforced by checkign its residuals (SEE 2.14)



The residuals show a box-test that rejects the null hypothesis of the model not being an appropriate fit (therefore, it is). Again, the distribution is normal and the autocorrelation is statistically significant as the same points of the previous ARIMA models that were simulated.

Through the simulation of three distinct fitted ARIMA models, we were able to back-estimate certain variables and distinguish which model allowed for the most accurate simulation. It is indubitably decided that ARIMA(0,0,1) displays the best simulation, as predicted by the auto.arima() function, and thus the estimation of parameters is most

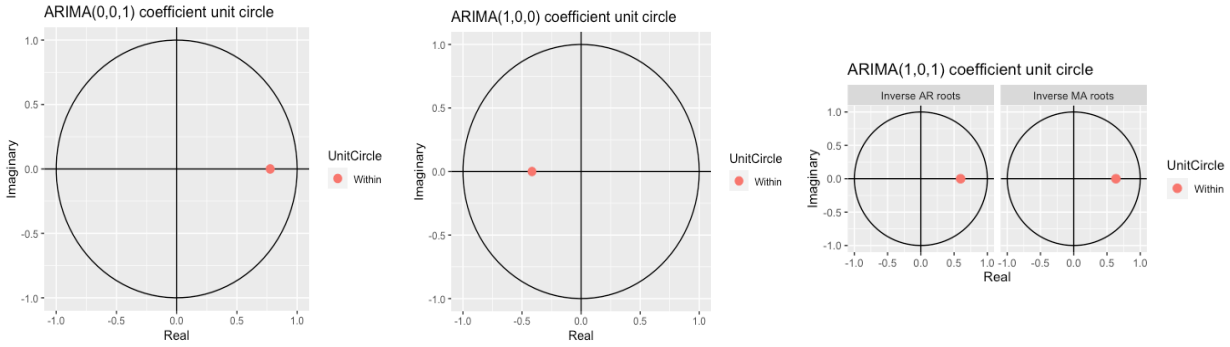
prominent in that model. Further estimation is conducted in the diagnostics checking, in which the comparison of ARIMA coefficients (p,d,q) is significant for both parameter estimation and variation analysis.

3.1 SIMULATION PART 2 (FURTHER PARAMETER ESTIMATION + DIAGNOSTICS):

(SEE 3.1) We can see that the residual analysis of each ARIMA model further indicates that ARIMA(0,0,1) provides the best estimation for the AR,MA coefficients. The difference between the residual coefficient and fitted coefficient (fitted-residual) indicated a 0.028 (3dp) difference, while the ARIMA(1,0,0) residuals showed a -0.105 difference and the ARIMA(1,0,1) residuals showed a -0.738 difference.

(SEE 3.2) Nonlinearity testing combines a variety of tests which all measure p-values against the null hypothesis that the model follows an ARIMA process. For the ARIMA(0,0,1), the comparison of nonlinearityTest indicates a single differentiation in the variation of both data sets. The fitted data from the actual values outputs p-values over 0.05 in every test, thus accepting the null hypothesis that the test does adequately follow an ARIMA process. The simulated data however, has p-values over 0.05 for every test except for the McLeod-li test, which is under 0.05 (rejecting the null hypothesis that it is a ARIMA process). Considering every other value for both the fitted data and simulated data achieve the same non-linearity results, we decide to ignore the rejection of the null hypothesis in a single test.

To summarize the other two models, ARIMA(1,0,0) failed zero tests while ARIMA(1,0,1) failed two.



(SEE 3.3) The unit circles above show the estimation of each coefficient as part of a unit circle difference. The MA(1) model only displays a moving average root, the AR(1) model only displays the auto-regressive root, while ARIMA(1,0,1) displays both. The ARIMA(0,0,1) and ARIMA(1,0,1) simulated models appear to display similar coefficients, indicating some similarity in the order of the values being simulated. This is due to the

separate equations used in each simulation, which primarily dictates what values each model will synthetically generate.

Summary of Findings/Interpretations:

Most of the results are shown and interpreted in the above sections. Below is a summary of findings and overall interpretation of results.

Through the simulation of multiple ARIMA models, we were successfully able to generate synthetic data which was then utilized to back-estimate values of the ARIMA order (p,d,q) as well as the MA and AR coefficients through residual analysis. Exploratory analysis was conducted to find, transform and aggregate data into a useable time series, one which achieved stationarity and non-seasonality. Fitted values were then applied against ARIMA models to find the best fixed parameters to apply simulation on. Simulation was then utilized to generate synthetic data which back-estimated the fixed parameters, with each of its variances in results being accounted for through diagnostics. Many indicators of the data showed that the connections for each day were almost in a stationary nature, however further transformation needed to be done to allow modelling to occur. The comparison of three separate ARIMA models, (0,0,1),(1,0,0) and (1,0,1) indicated that each had their own advantages and disadvantages in simulating synthetic data. ARIMA(1,0,0) was highly accurate in terms of non-linearity by not failing a single test, but it's coefficient estimation was wildly off. ARIMA(1,0,1) failed two non-linearity tests, but was far more successful in coefficient estimation. ARIMA(0,0,1), which was auto-selected initially, was the most successful in every aspect in terms of simulation, backwards-estimation and variation.

There were certain limitations that should be acknowledged in this project. Many seasonal variables (in which new parameters are involved) are far more difficult to simulate by hand, thus the auto.sim() function is a better option. Furthermore, the scope of the data does not allow much more value in a monthly or yearly scale as it does in a daily or weekly frequency, as the data only spans 3 months across. Many advanced functions were untouched and often used in insignificant analysis due to the fact that many of the simulation processes were tedious to construct and took far too long to visualize with standard visualisation tools. Finally, ARIMA modelling provides far more insight in terms of forecasting future values, however, when the simulated data is utilized in the same timeframe as fitted data to back-estimate specific values, it's purpose is far less significant and the capabilities and potential for analysis is hindered.

References:

Benson, T., Akella, A., Maltz. D. (2010). Network traffic characteristics of data centers in the wild. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. Association for Computing Machinery, New York, 267–280. https://doi.org/10.1145/1879141.1879175

Cortez, P., Rio, M., Rocha, M., Sousa, P., (2012). Multi-scale Internet traffic forecastin using neural networks and time series methods. Expert Systems. https://www.slideshare.net/ajayohri/exsy568

Crawford, C. 2017. Computer Network Traffic. Kaggle. https://www.kaggle.com/crawford/computer-network-traffic

Jung S., Kim C., Chung Y. (2006) A Prediction Method of Network Traffic Using Time Series Models. In: Gavrilova M. et al. (eds) Computational Science and Its Applications - ICCSA 2006. ICCSA 2006. Lecture Notes in Computer Science, vol 3982. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11751595_26

Labarr, A. [Aric Labarr]. (2019 Aug 5). What is Stationarity? [Video]. Youtube. https://www.youtube.com/watch?v=aIdTGKjQWjA&ab_channel=AricLaBarr

Otexts.. 2019 ARIMA modelling in R. https://www.youtube.com/watch?v=aIdTGKjQWjA&ab_channel=AricLaBarr

Prabhakaran, S. 2019. ARIMA Model – Complete Guide to Time Series Forecasting in Python. MachineLearningplus.com. https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

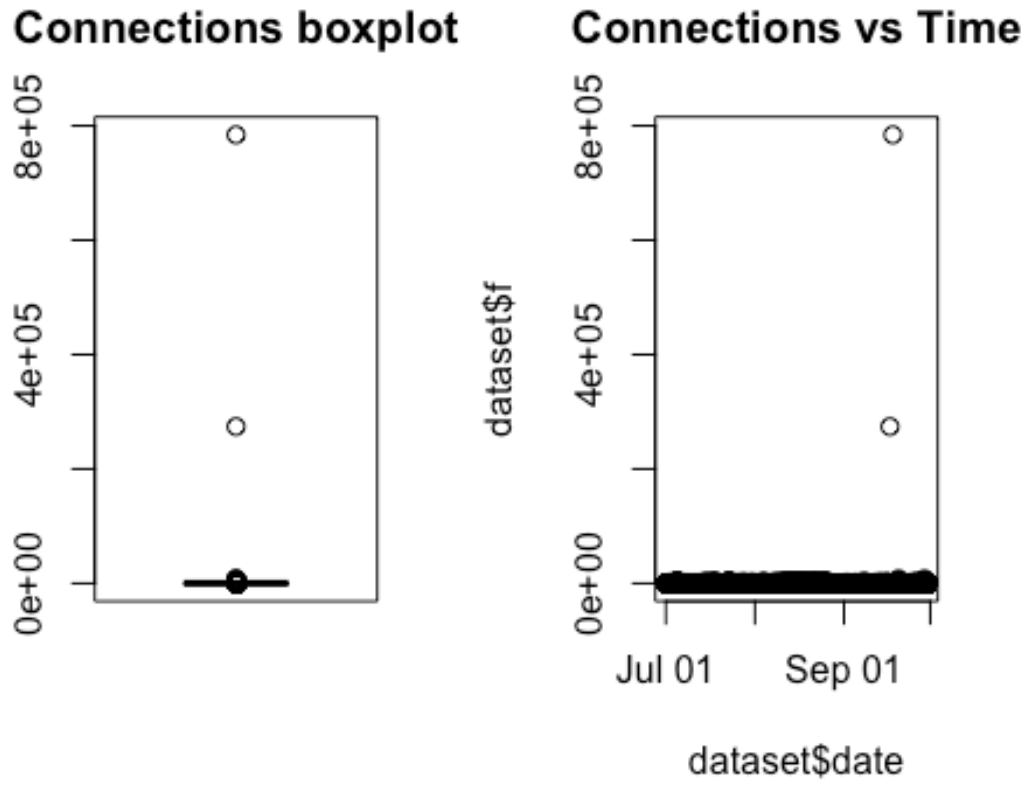
Appendix:

```
#1.1
library(readr)
dataset <- read_csv("cs448b_ipasn.csv") #reading in the file

## Parsed with column specification:
## cols(
##   date = col_date(format = ""),
##   l_ipn = col_double(),
##   r_asn = col_double(),
##   f = col_double()
## )

#View(dataset)

#1.2
#install.packages('package') to install these packages
library(dslabs)  #full list of packages utilised.
library(ggplot2)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(forecast)

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

library(plotly)

##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

library(flexdashboard)
library(nonlinearTseries)

##
## Attaching package: 'nonlinearTseries'

## The following object is masked from 'package:grDevices':
##
##     contourLines

#1.3
summary(dataset)

##       date                l_ipn           r_asn             f
##  Min.   :2006-07-01   Min.   :0.000   Min.   :    3   Min.   :     1.0
##  1st Qu.:2006-07-23   1st Qu.:1.000   1st Qu.: 4323   1st Qu.:     1.0
##  Median :2006-08-11   Median :4.000   Median : 8764   Median :     2.0
##  Mean   :2006-08-13   Mean   :4.228   Mean   :12138   Mean   :    93.9
##  3rd Qu.:2006-09-04   3rd Qu.:7.000   3rd Qu.:17676   3rd Qu.:     8.0
##  Max.   :2006-09-30   Max.   :9.000   Max.   :40092   Max.   :784234.0

str(dataset) #getting a feel for the data. One date, one numerical
quantitative, two numerical categorical.

## tibble [20,803 × 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ date : Date[1:20803], format: "2006-07-01" "2006-07-01" ...
##  $ l_ipn: num [1:20803] 0 0 0 0 0 0 0 0 0 0 ...
##  $ r_asn: num [1:20803] 701 714 1239 1680 2514 ...
##  $ f    : num [1:20803] 1 1 1 1 1 1 13 3 2 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   date = col_date(format = ""),
##   ..   l_ipn = col_double(),
##   ..   r_asn = col_double(),
##   ..   f = col_double()
##   .. )

#1.4
max(dataset$f)

## [1] 784234

min(dataset$f)  #getting a feel for the connections values range
```

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

```
par(mfrow=c(1,2))
boxplot(dataset$f, main = 'Connections boxplot')
plot(dataset$date, dataset$f, main = 'Connections vs Time')  #seeing how the
variables compare to each other
```

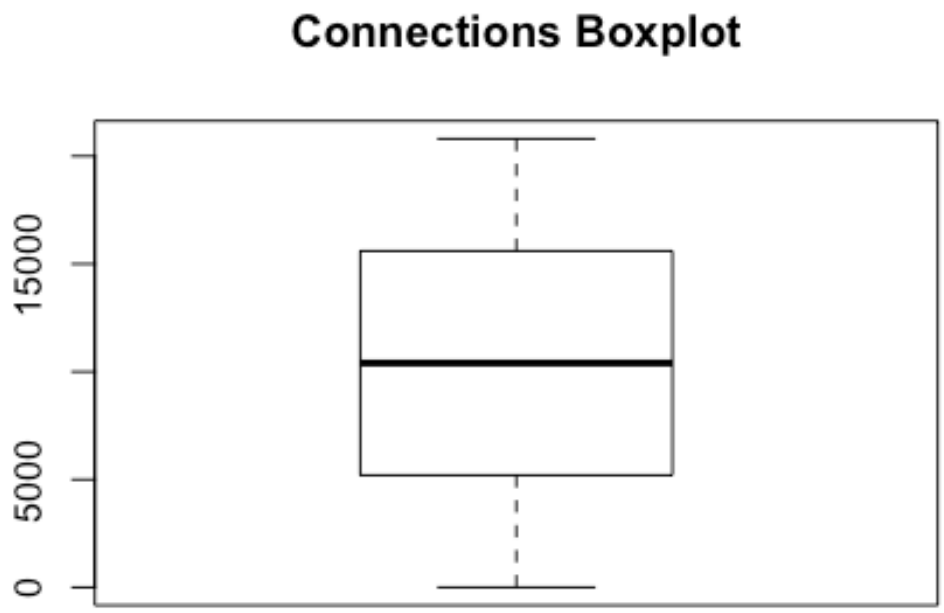



```
#1.5
largestvalues <- dataset %>%
  dplyr::arrange(desc(dataset$f)) %>%  #arranging according to highest f
value
  dplyr::select(date, f) #select date and f value
head(largestvalues,10)  #find out the highest 10 f values and their date
```

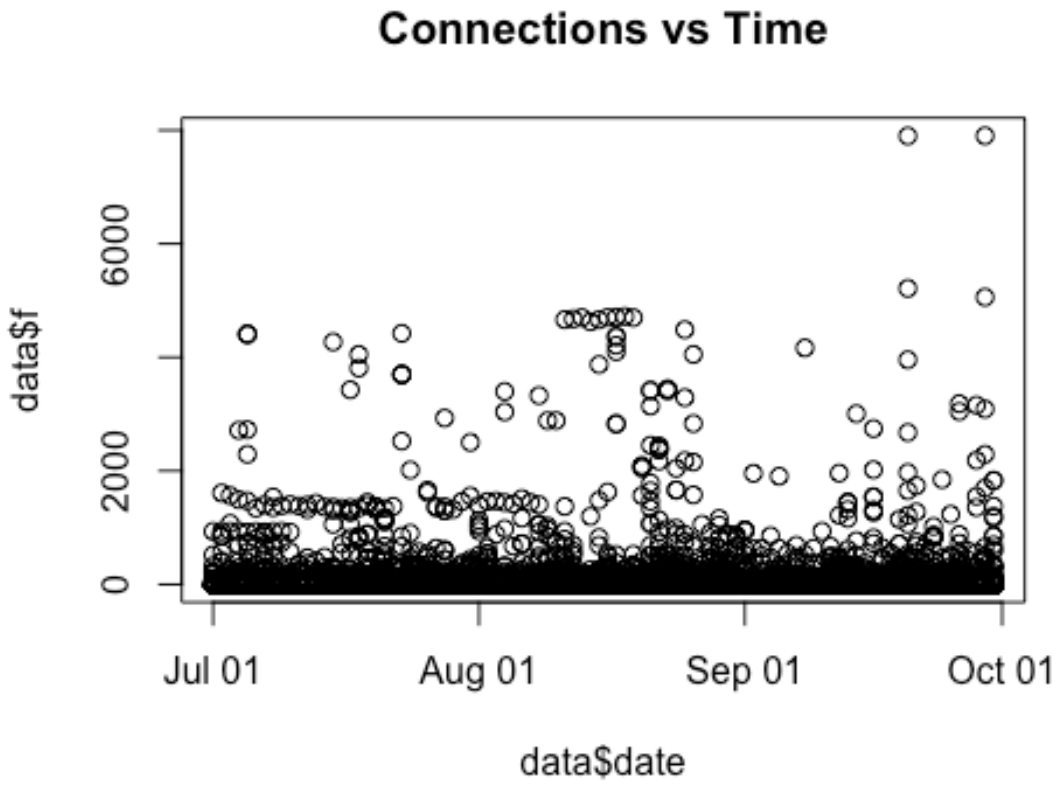
```
## # A tibble: 10 x 2
##    date            f
##    <date>      <dbl>
##  1 2006-09-18 784234
##  2 2006-09-17 274011
##  3 2006-09-29   7902
##  4 2006-09-20   7899
##  5 2006-09-20   5214
##  6 2006-09-29   5059
##  7 2006-08-18   4718
```

```
##  8 2006-08-17   4708
##  9 2006-08-16   4702
## 10 2006-08-13   4701
```

```r
#1.6
boxplot(which(dataset$f<10000), main = 'Connections Boxplot') #boxplot
without the two outliers
```

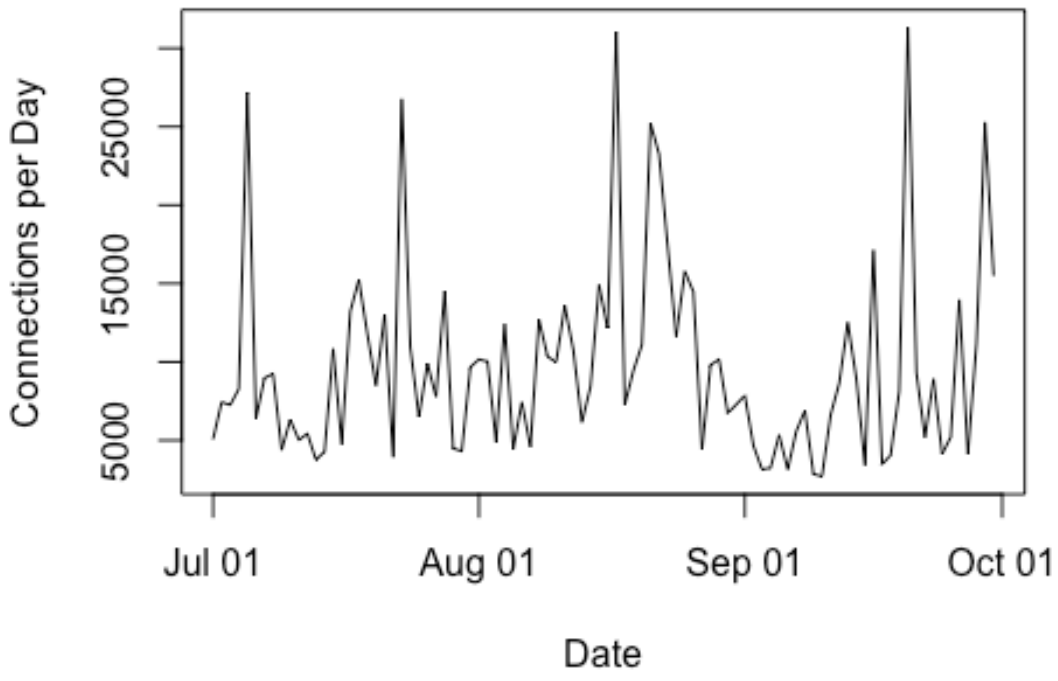



```r
#remove outliars
data <- subset(dataset, dataset$f < 10000) #removing outliers
#View(data)
plot(data$date,data$f, main = 'Connections vs Time')
```



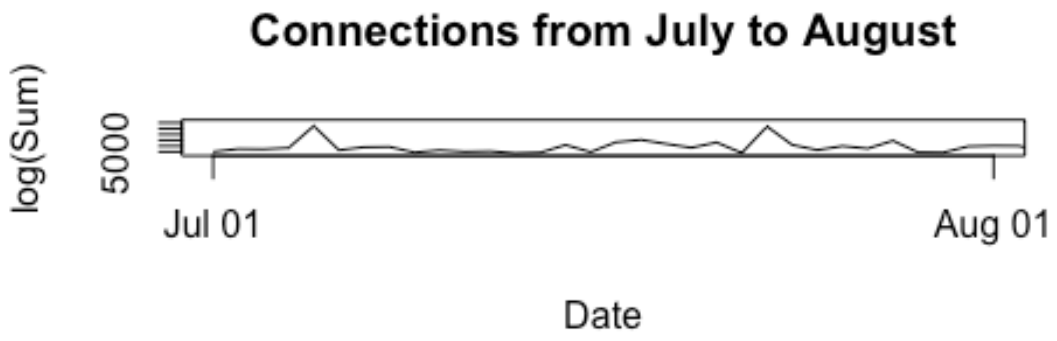

```
#1.7

#plot1 <- ggplot(data, aes(x=data$date,
#                      y=data$f,
#                      color= l_ipn )) +
#  geom_point(size=3) +
#  labs(x = "Date",
#       y = "Connections",
#       color = "ip station") +
#  theme_bw()

#ggplotly(plot1)

#1.8
#summary(data)
firstdate <- data %>%
  dplyr::filter(date == '2006-07-01') %>%
  dplyr::select(f)
sum(firstdate)  #can the f values be separated as sums of each day? How?

## [1] 5058

#1.9
length(unique(data$date))
```

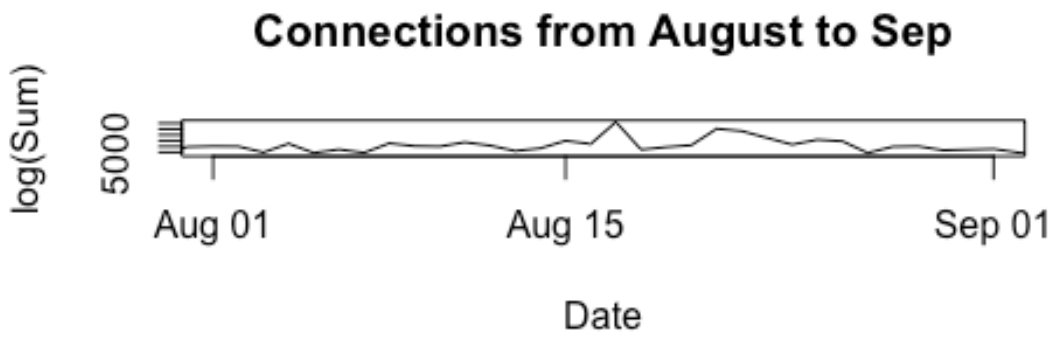
```
## [1] 92

#1.10 (USE THIS IN REPORT)
fdata <-subset(data, select= -c(l_ipn,r_asn))

#1.11

fsumdata <- aggregate(x = fdata[c('f')], by = list(fdata$date), FUN = sum)
#aggregate function applies the FUN function to every variable in the data,
thus it's important to specify that we only want f to be summed, hence
c('f'). The data is ordered by date

plot(fsumdata$Group.1, fsumdata$f, type = 'l', ylab= 'Connections per Day',
xlab = 'Date')
```




```
#1.12

par(mfrow = c(2,1))
plot(fsumdata$Group.1, fsumdata$f, type = 'l', xlim = as.Date(c('2006-07-01',
'2006-08-01')),xlab = 'Date', ylab = 'log(Sum)', main = 'Connections from
July to August')

plot(fsumdata$Group.1, fsumdata$f, type = 'l', xlim = as.Date(c('2006-08-01',
```

```
'2006-09-01')),xlab = 'Date', ylab = 'log(Sum)', main = 'Connections from
August to Sep')
```

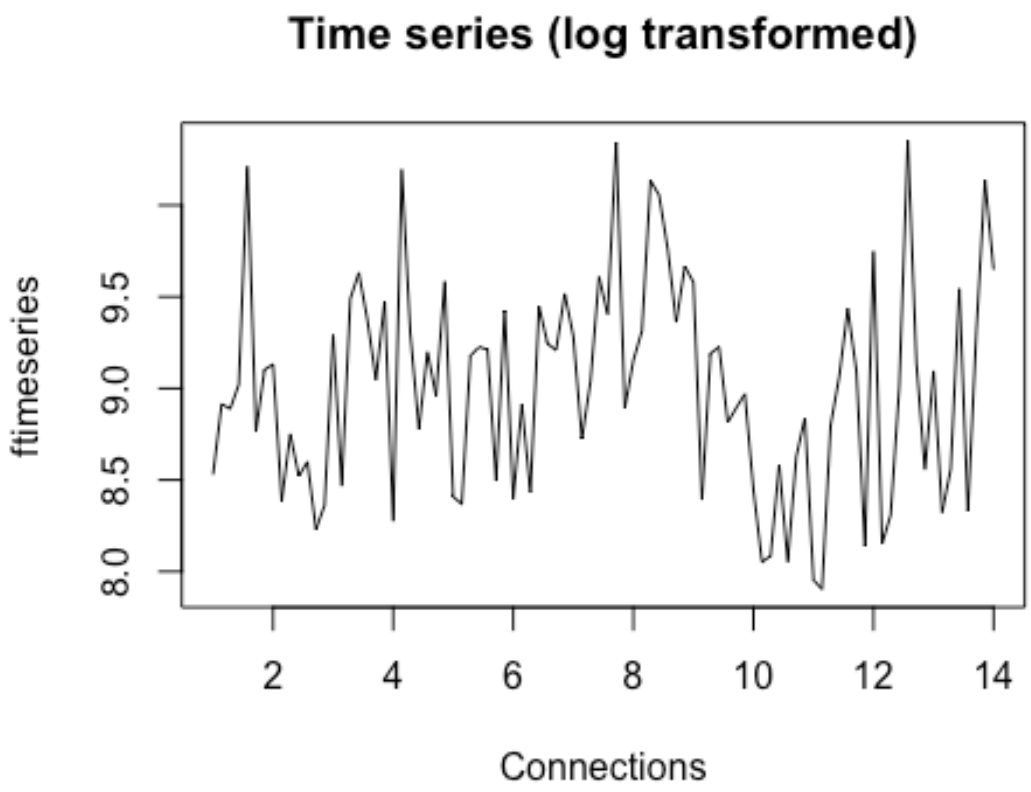




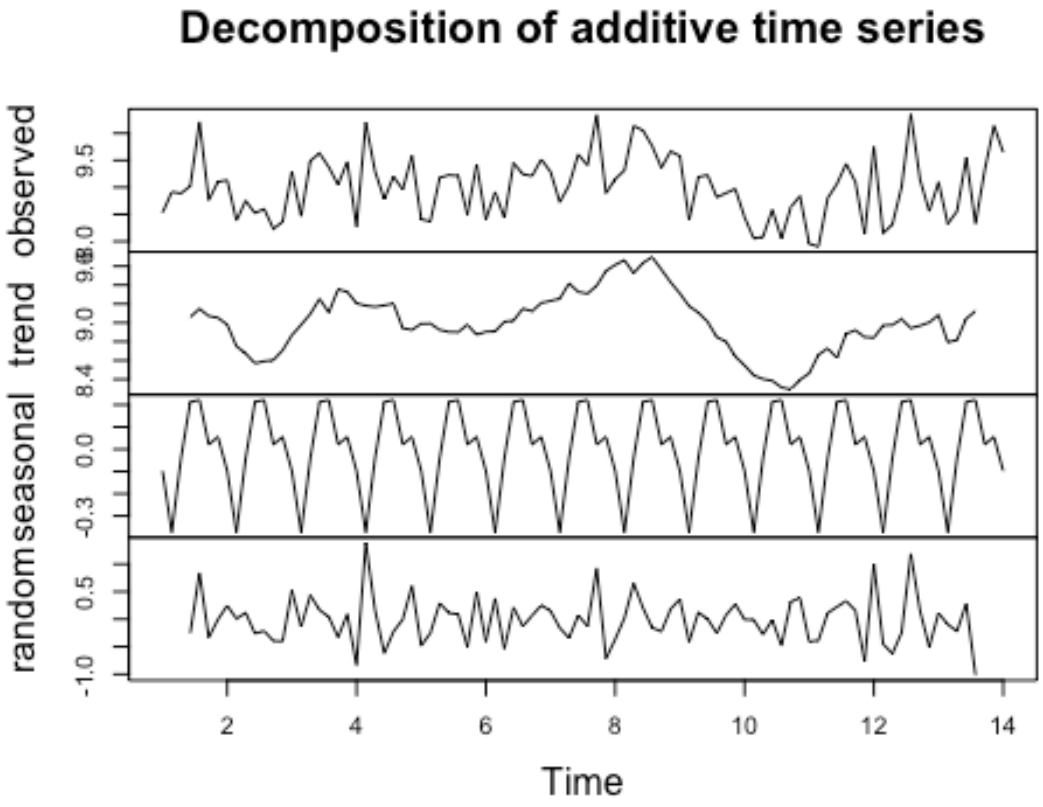

```
#1.13
ftimeseries <- ts(log(fsumdata$f), frequency = 7) #14 lags for the model. I
assume that daily data will have weekly seasonality with the greatest
significance
ftimeseries

## Time Series:
## Start = c(1, 1)
## End = c(14, 1)
## Frequency = 7
##  [1]  8.528726  8.914223  8.888481  9.021598 10.210568  8.762646  9.101752
##  [8]  9.131297  8.384804  8.751474  8.521185  8.599510  8.230311  8.362409
## [15]  9.290906  8.467583  9.489032  9.631941  9.371949  9.047704  9.474472
## [22]  8.275886 10.195000  9.298901  8.779096  9.199077  8.959954  9.582869
## [29]  8.408494  8.368461  9.176473  9.226509  9.212338  8.498010  9.425048
## [36]  8.398635  8.912877  8.434246  9.448333  9.243969  9.208639  9.516942
## [43]  9.285262  8.724533  9.036820  9.609116  9.405825 10.343161  8.891512
## [50]  9.151969  9.313889 10.135313 10.053888  9.763651  9.361257  9.666245
## [57]  9.579625  8.392083  9.190546  9.226115  8.814033  8.893710  8.969669
## [64]  8.447629  8.051978  8.085487  8.583168  8.054205  8.628377  8.838987
```

```
## [71]  7.955074  7.903227  8.798455  9.063347  9.435003  9.097843  8.138857
## [78]  9.748061  8.150756  8.309677  9.004423 10.351533  9.138844  8.556798
## [85]  9.095378  8.323123  8.559294  9.544166  8.329175  9.319105 10.137808
## [92]  9.647627
```

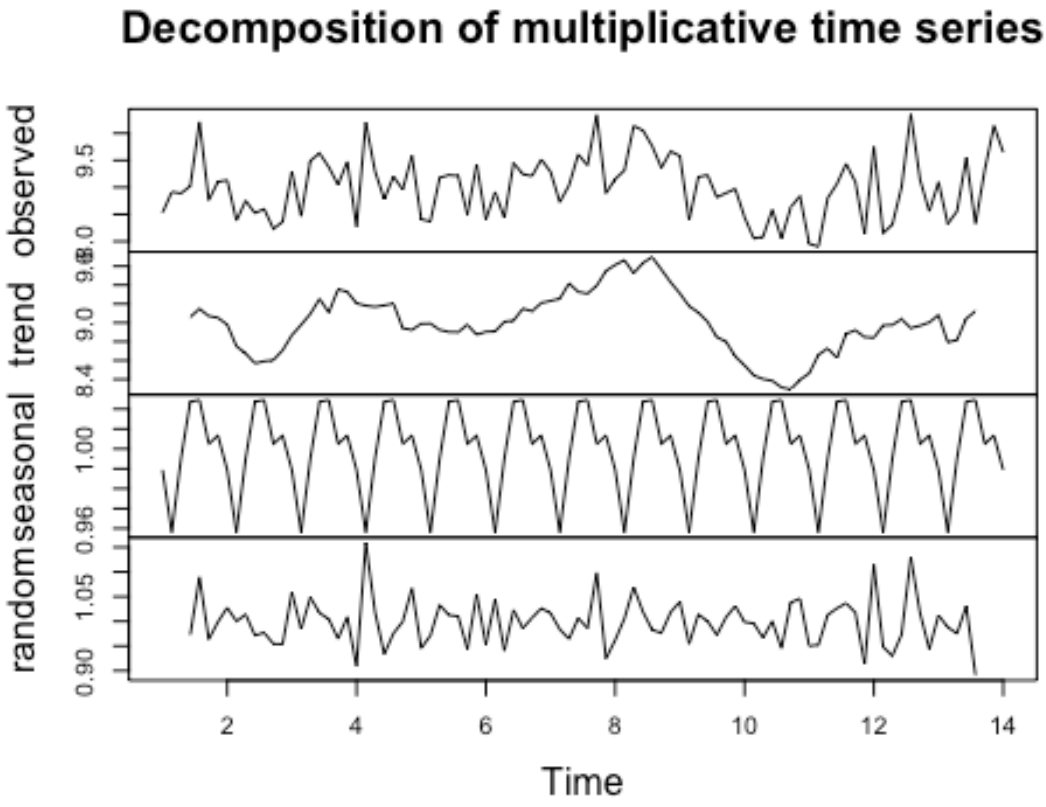
```
plot.ts(ftimeseries, main = 'Time series (log transformed)', xlab =
'Connections')
```



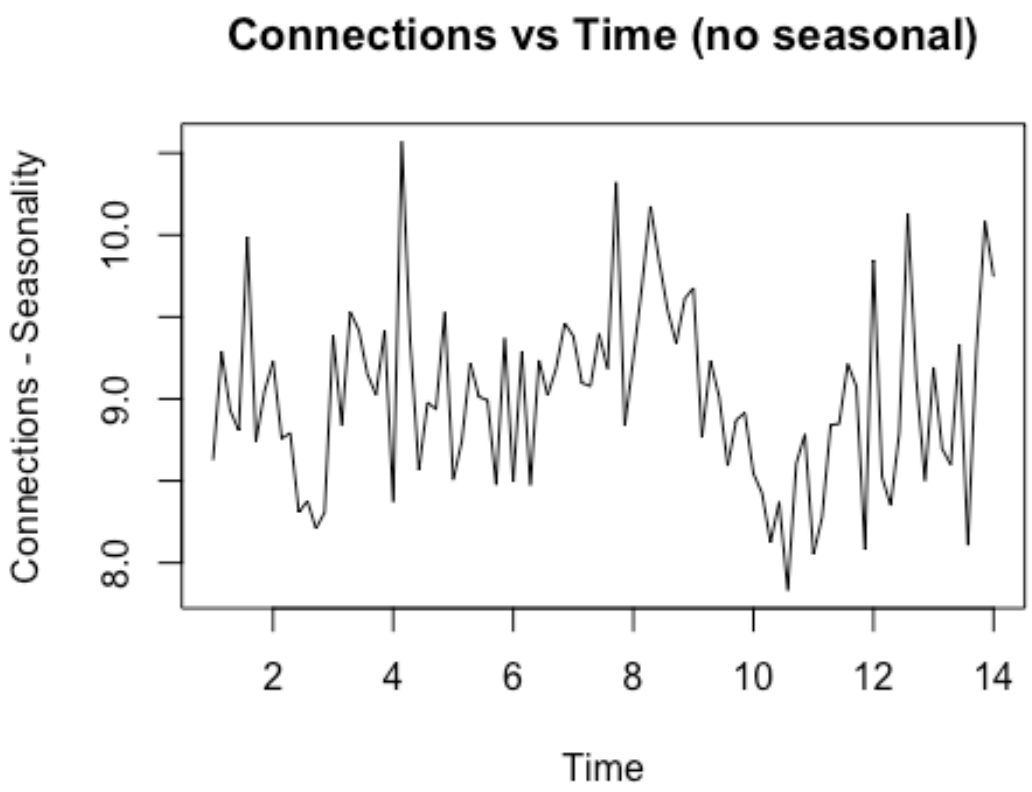

```
#2.1
components <- decompose(ftimeseries, type = 'additive')
componentsmulti <- decompose(ftimeseries, type = 'multiplicative') #two types
p * d * q vs p + d + q
par(mfrow=c(1,2))
plot(components)
```



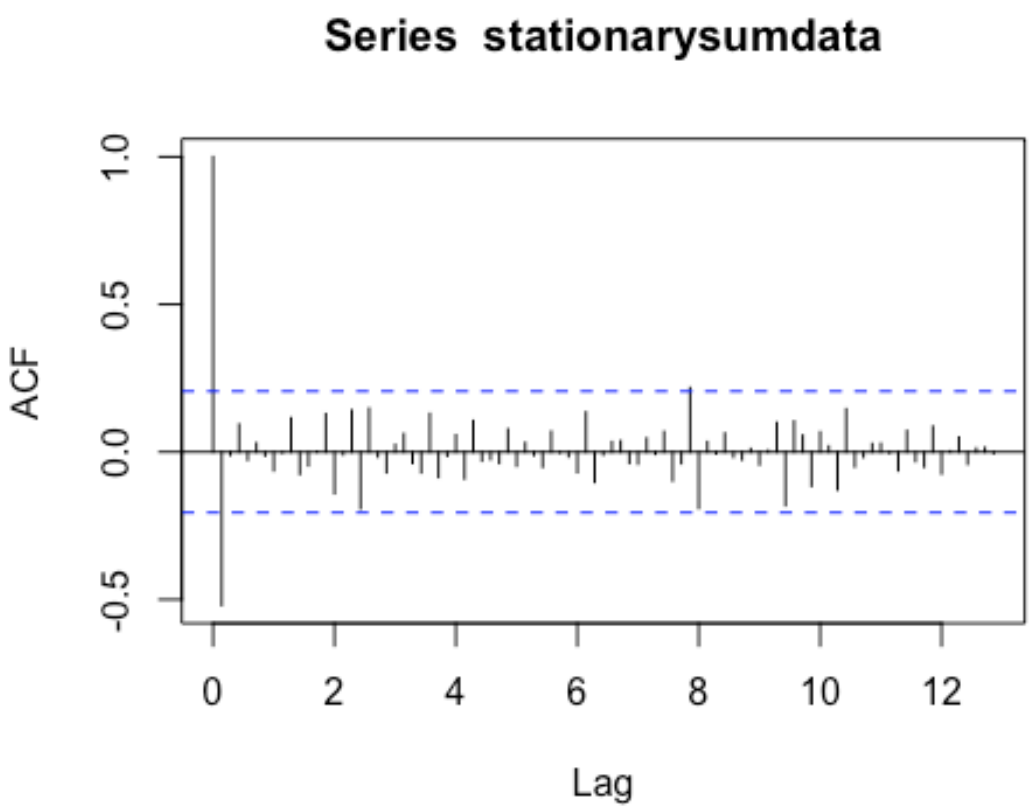

```
plot(componentsmulti)
```



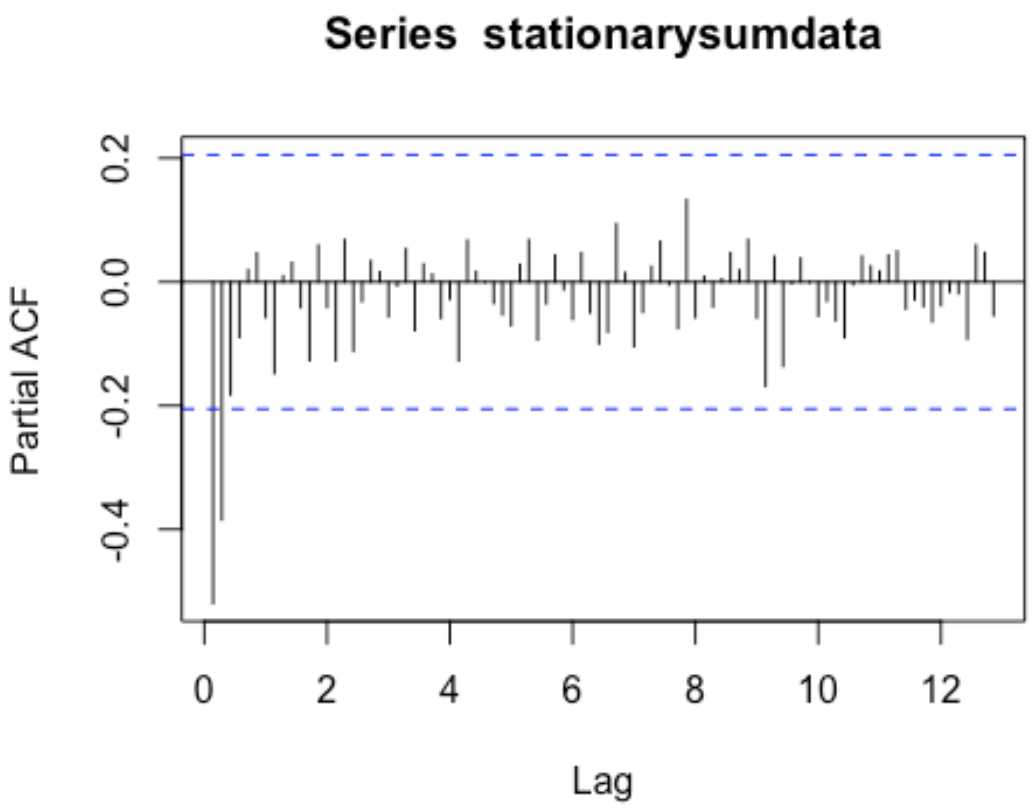

```r
#2.2
tsnoseasonal <- (ftimeseries - components$seasonal) #remove seasonal values

plot(tsnoseasonal, main = 'Connections vs Time (no seasonal)', ylab =
'Connections - Seasonality') #plot no-seasonality values
```



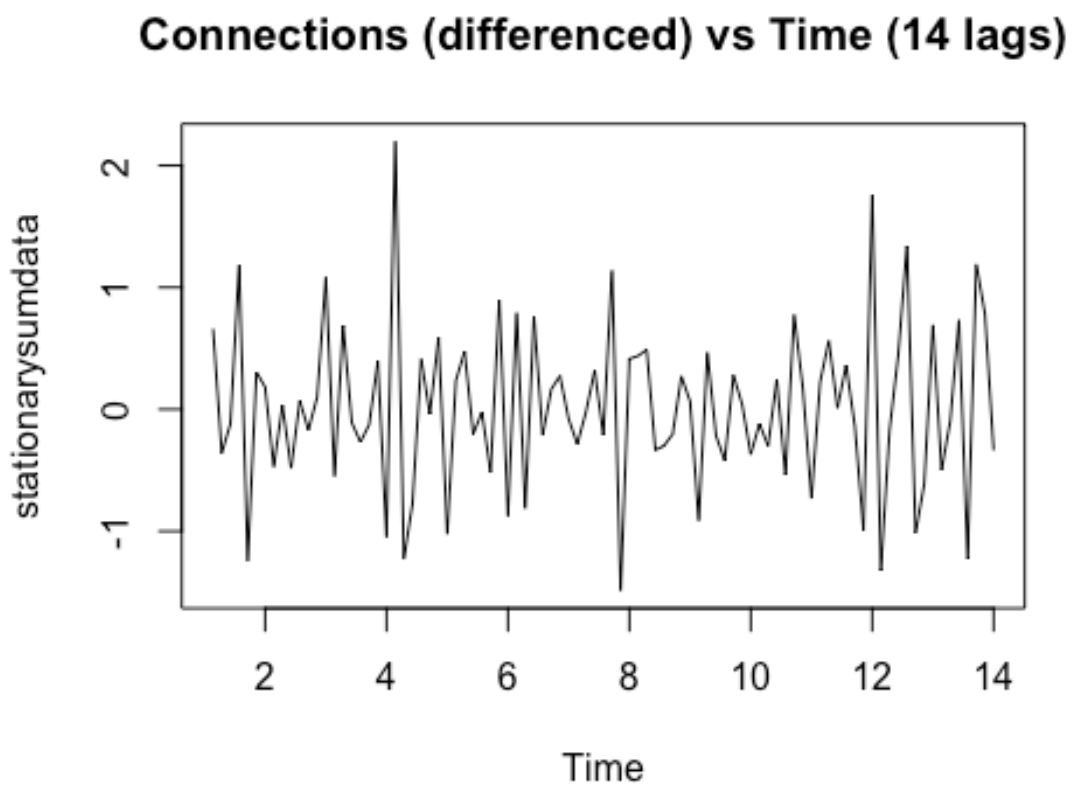

```r
stationarysumdata <- diff(tsnoseasonal) #differencing the data, which as explained before, makes data more stationary. It was important to remove seasonality before differencing the data.

acf(stationarysumdata, lag.max = 92)
```




```
pacf(stationarysumdata, lag.max = 92)
```

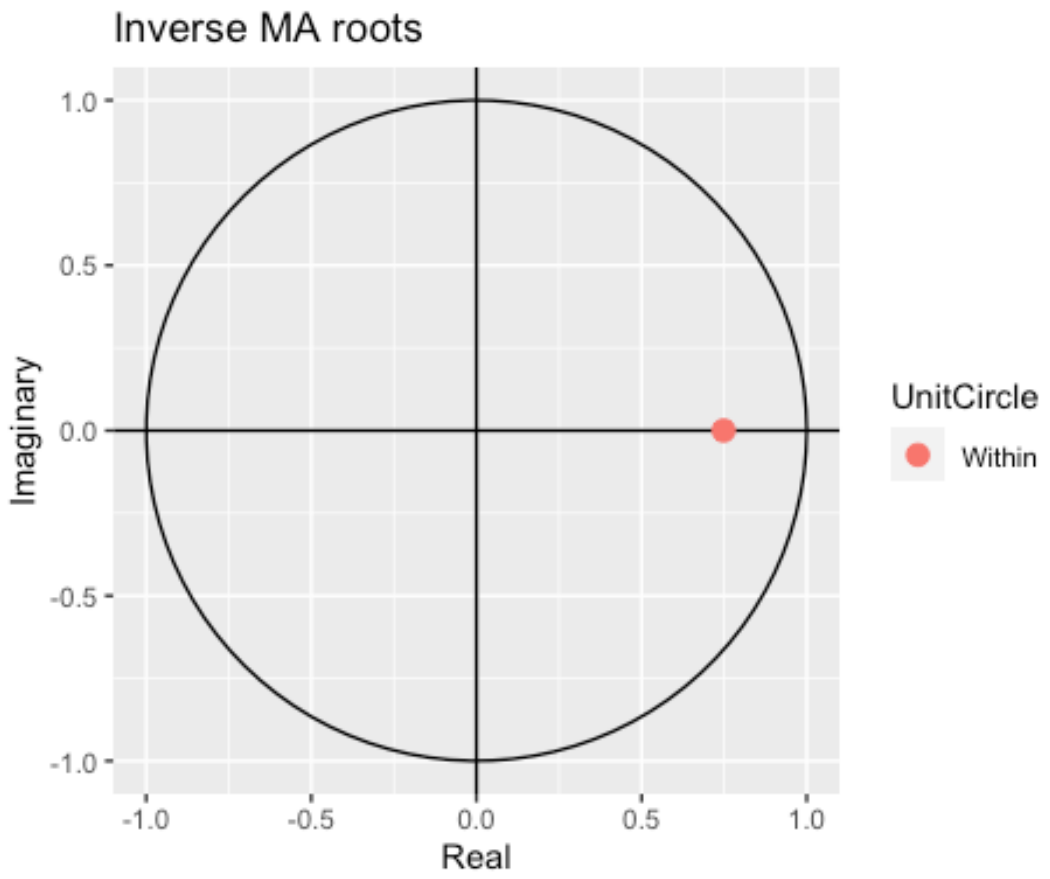



```
#acf(stationarysumdata, lag.max = 92, plot = FALSE)
#pacf(stationarysumdata, lag.max = 92, plot = FALSE)

#2.3
plot(stationarysumdata, main = 'Connections (differenced) vs Time (14 lags)')
```

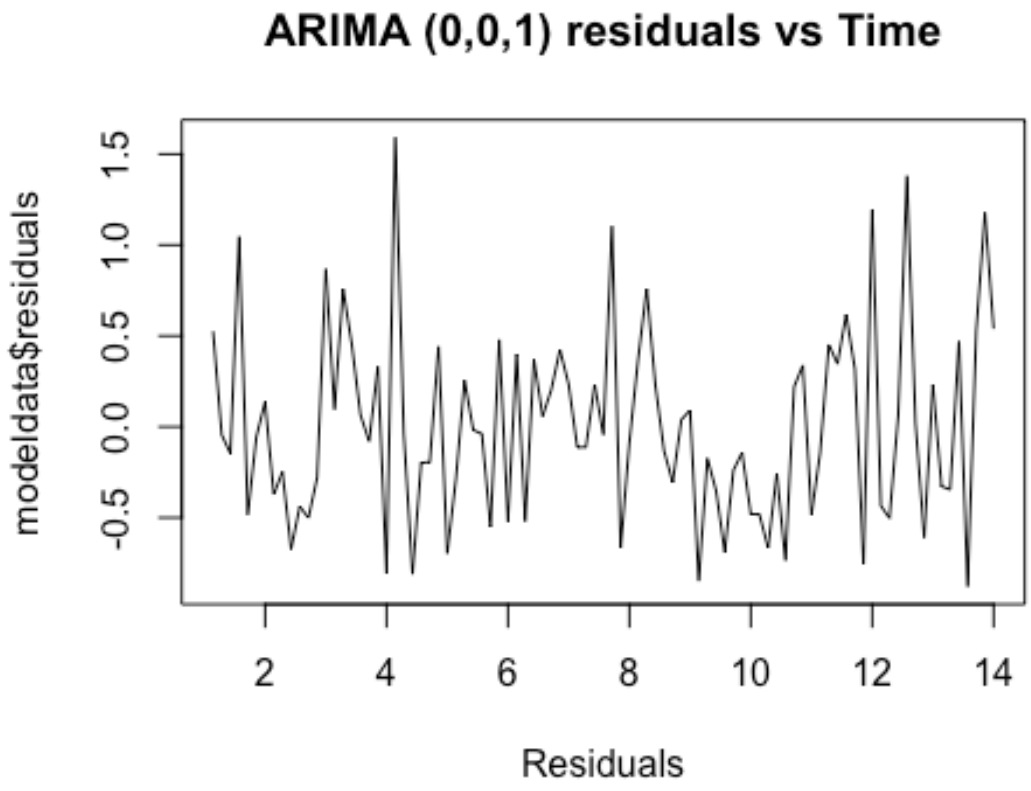



```
#2.4
auto.arima(stationarysumdata, trace=TRUE)

##
##  ARIMA(2,0,2)(1,0,1)[7] with non-zero mean : Inf
##  ARIMA(0,0,0)           with non-zero mean : 193.1646
##  ARIMA(1,0,0)(1,0,0)[7] with non-zero mean : 167.1863
##  ARIMA(0,0,1)(0,0,1)[7] with non-zero mean : 150.8988
##  ARIMA(0,0,0)           with zero mean     : 191.1027
##  ARIMA(0,0,1)           with non-zero mean : 149.8472
##  ARIMA(0,0,1)(1,0,0)[7] with non-zero mean : 151.3056
##  ARIMA(0,0,1)(1,0,1)[7] with non-zero mean : Inf
##  ARIMA(1,0,1)           with non-zero mean : 151.1443
##  ARIMA(0,0,2)           with non-zero mean : 150.8513
##  ARIMA(1,0,0)           with non-zero mean : 166.5528
##  ARIMA(1,0,2)           with non-zero mean : 152.9016
##  ARIMA(0,0,1)           with zero mean     : 147.7927
##  ARIMA(0,0,1)(1,0,0)[7] with zero mean     : 149.2125
##  ARIMA(0,0,1)(0,0,1)[7] with zero mean     : 148.8083
##  ARIMA(0,0,1)(1,0,1)[7] with zero mean     : Inf
##  ARIMA(1,0,1)           with zero mean     : 149.0515
##  ARIMA(0,0,2)           with zero mean     : 148.765
##  ARIMA(1,0,0)           with zero mean     : 164.4899
```

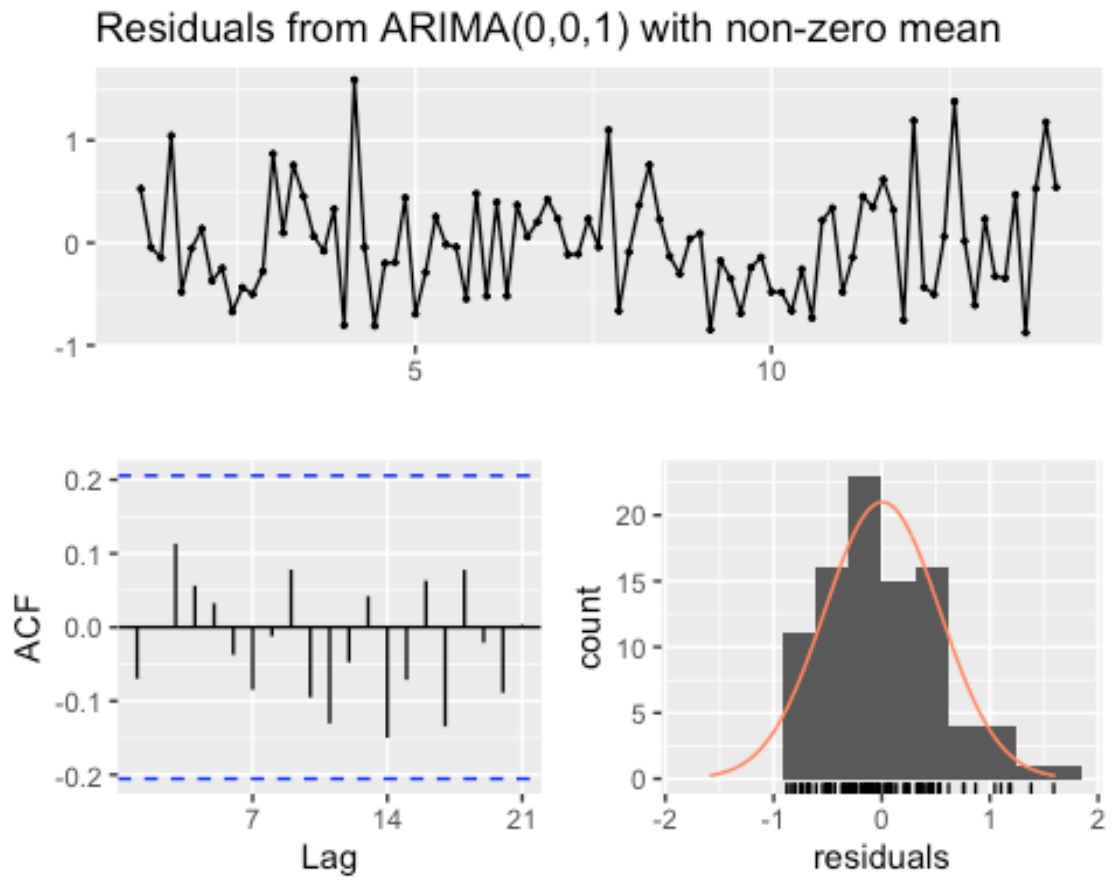
```
##  ARIMA(1,0,2)           with zero mean     : 150.766
##
##  Best model: ARIMA(0,0,1)           with zero mean

## Series: stationarysumdata
## ARIMA(0,0,1) with zero mean
##
## Coefficients:
##           ma1
##       -0.7485
## s.e.   0.0746
##
## sigma^2 estimated as 0.2844:  log likelihood=-71.83
## AIC=147.66   AICc=147.79   BIC=152.68

#2.5
#ARIMA(0,0,1) model
modeldata <- arima(stationarysumdata, order = c(0,0,1)) #p=0,d=0,q=1
summary(modeldata)

##
## Call:
## arima(x = stationarysumdata, order = c(0, 0, 1))
##
## Coefficients:
##           ma1  intercept
##       -0.7485     0.0042
## s.e.   0.0746     0.0145
##
## sigma^2 estimated as 0.2811:  log likelihood = -71.79,  aic = 149.57
##
## Training set error measures:
##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.004855522 0.5301485 0.4222177 75.76532 157.7597 0.4607509
##                      ACF1
## Training set -0.06997323

autoplot(modeldata)
```




```
plot.ts(modeldata$residuals, xlab = 'Residuals', main = 'ARIMA (0,0,1)
residuals vs Time')
```



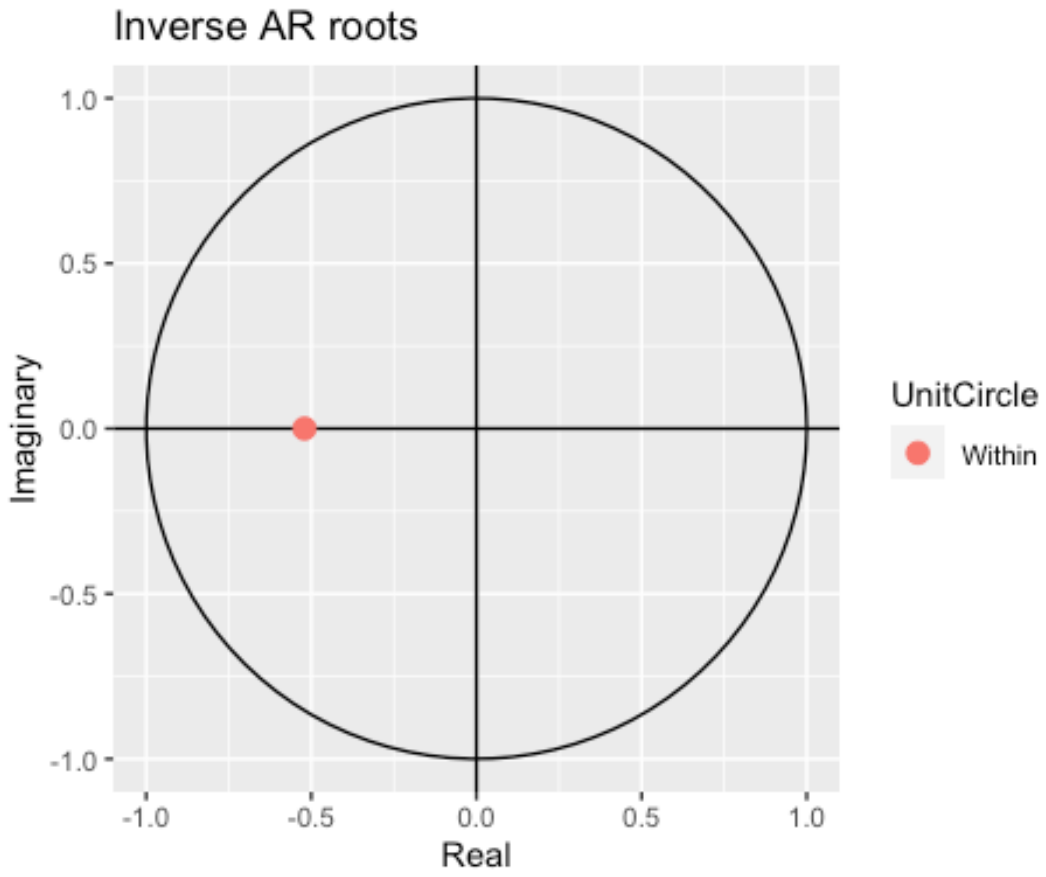

```
checkresiduals(modeldata)
```



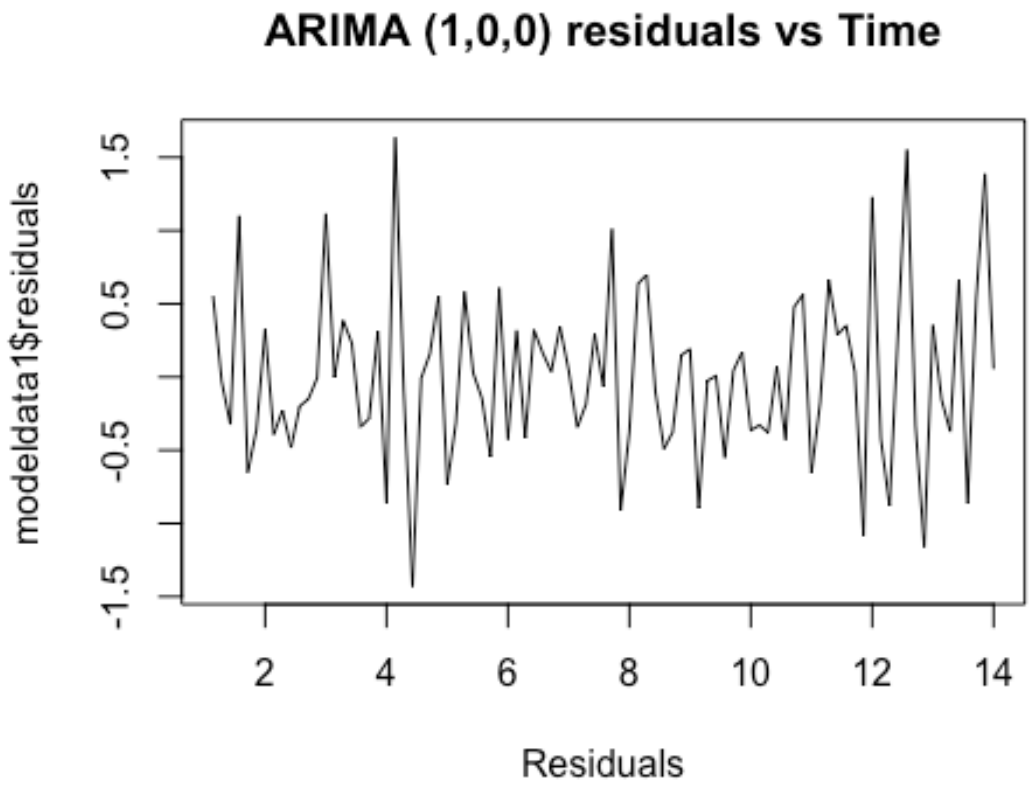

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1) with non-zero mean
## Q* = 9.2821, df = 12, p-value = 0.6787
##
## Model df: 2.   Total lags used: 14

coef(modeldata)

##         ma1   intercept
## -0.74845043  0.00421757

#modeldataforecast <- forecast(modeldata, h= 8)
#autoplot(modeldataforecast)
#plot.ts(modeldataforecast$residuals)

#2.6
#ARIMA(1,0,0) model
modeldata1 <- arima(stationarysumdata, order = c(1,0,0)) #p=1, d=0, q=0
summary(modeldata1)

##
## Call:
```

```
## arima(x = stationarysumdata, order = c(1, 0, 0))
##
## Coefficients:
##           ar1  intercept
##       -0.5206     0.0112
## s.e.   0.0891     0.0403
##
## sigma^2 estimated as 0.3396:  log likelihood = -80.14,  aic = 166.28
##
## Training set error measures:
##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.002671992 0.5827294 0.4495185 106.5913 155.2158 0.4905432
##                     ACF1
## Training set -0.1996828
```

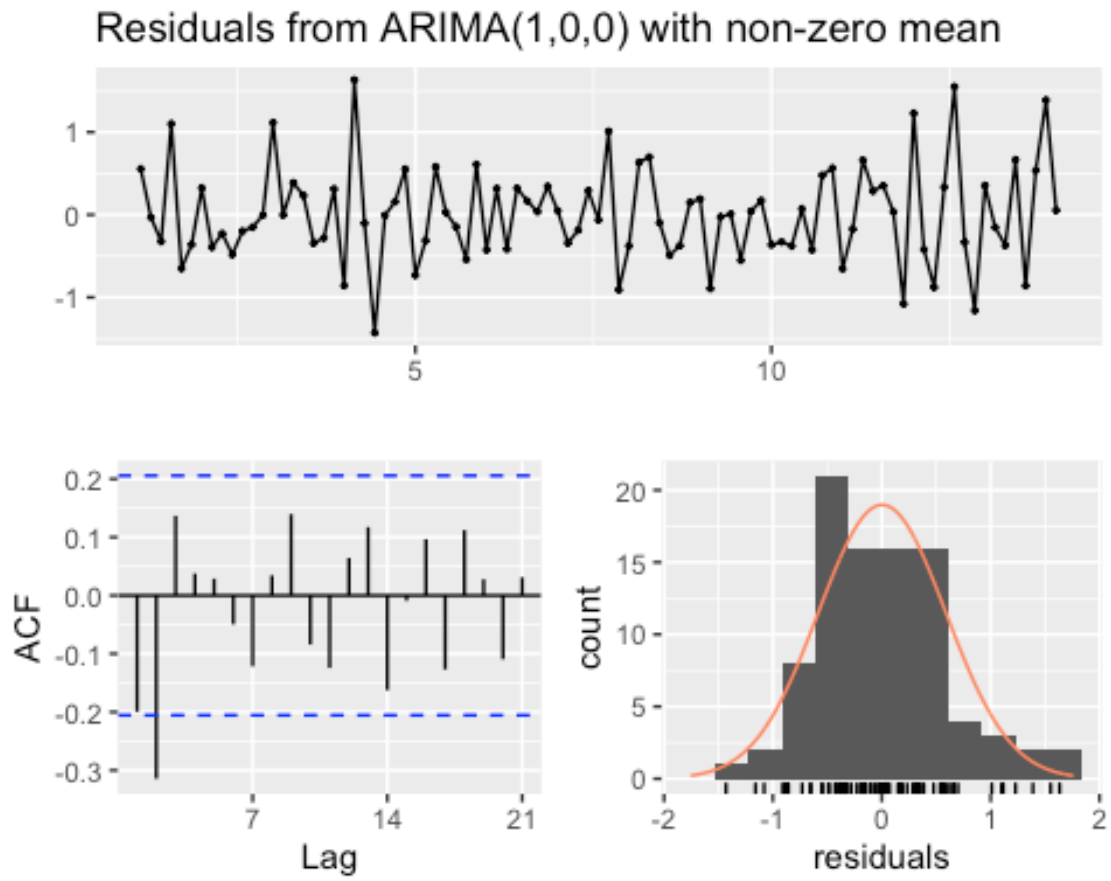
```
autoplot(modeldata1)
```




```
plot.ts(modeldata1$residuals, xlab = 'Residuals', main = 'ARIMA (1,0,0)
residuals vs Time')
```



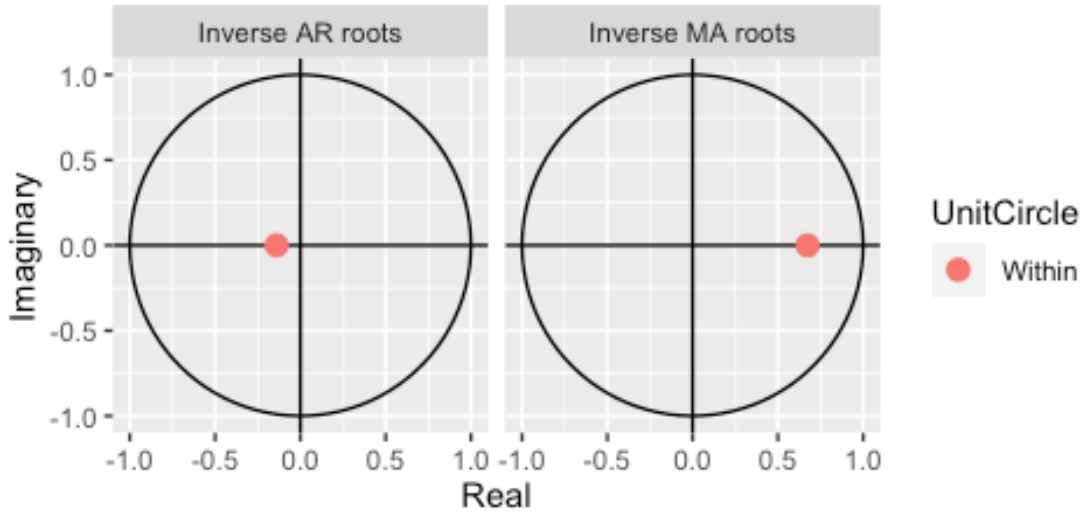

```
checkresiduals(modeldata1)
```



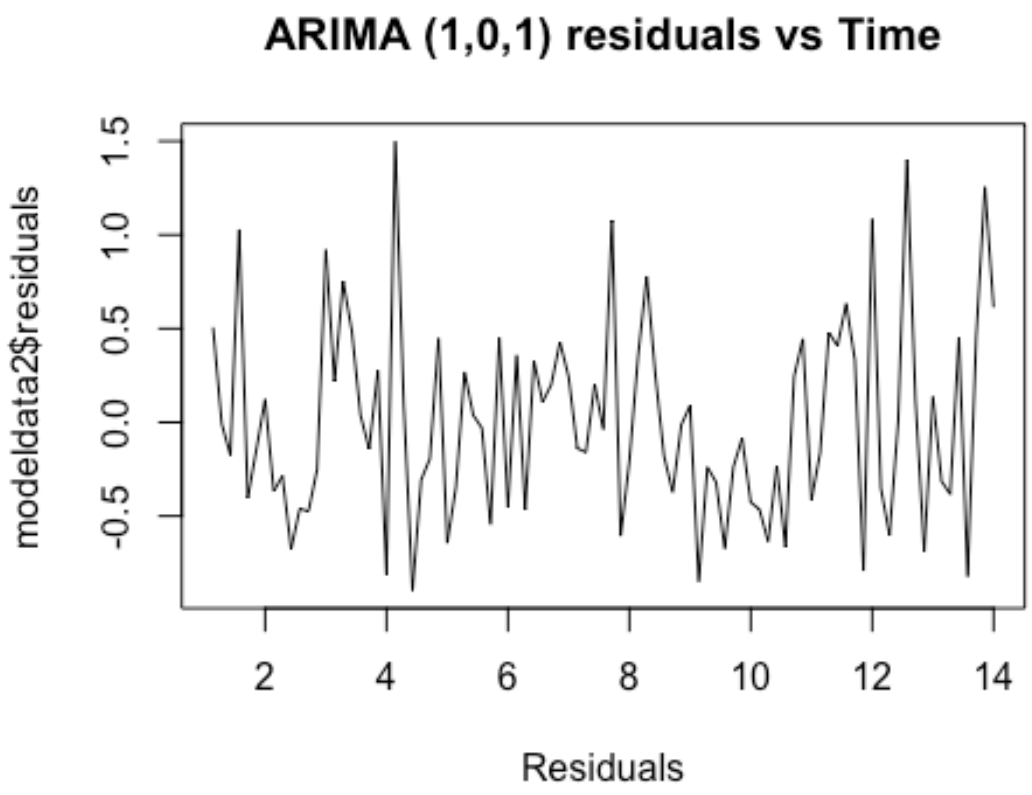

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with non-zero mean
## Q* = 26.267, df = 12, p-value = 0.009838
##
## Model df: 2.   Total lags used: 14

coef(modeldata1)

##         ar1   intercept
## -0.52055452  0.01116318

#2.7
#ARIMA(1,0,1)
modeldata2 <- arima(stationarysumdata, order = c(1,0,1)) #p=1, d=0, q=1
summary(modeldata2)

##
## Call:
## arima(x = stationarysumdata, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
```

```
##       -0.1397  -0.6741     0.0050
## s.e.   0.1432   0.1132     0.0163
##
## sigma^2 estimated as 0.2784:  log likelihood = -71.34,  aic = 150.68
##
## Training set error measures:
##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.005119117 0.5276018 0.4258615 81.44281 169.2123 0.4647272
##                      ACF1
## Training set -0.01103013
```

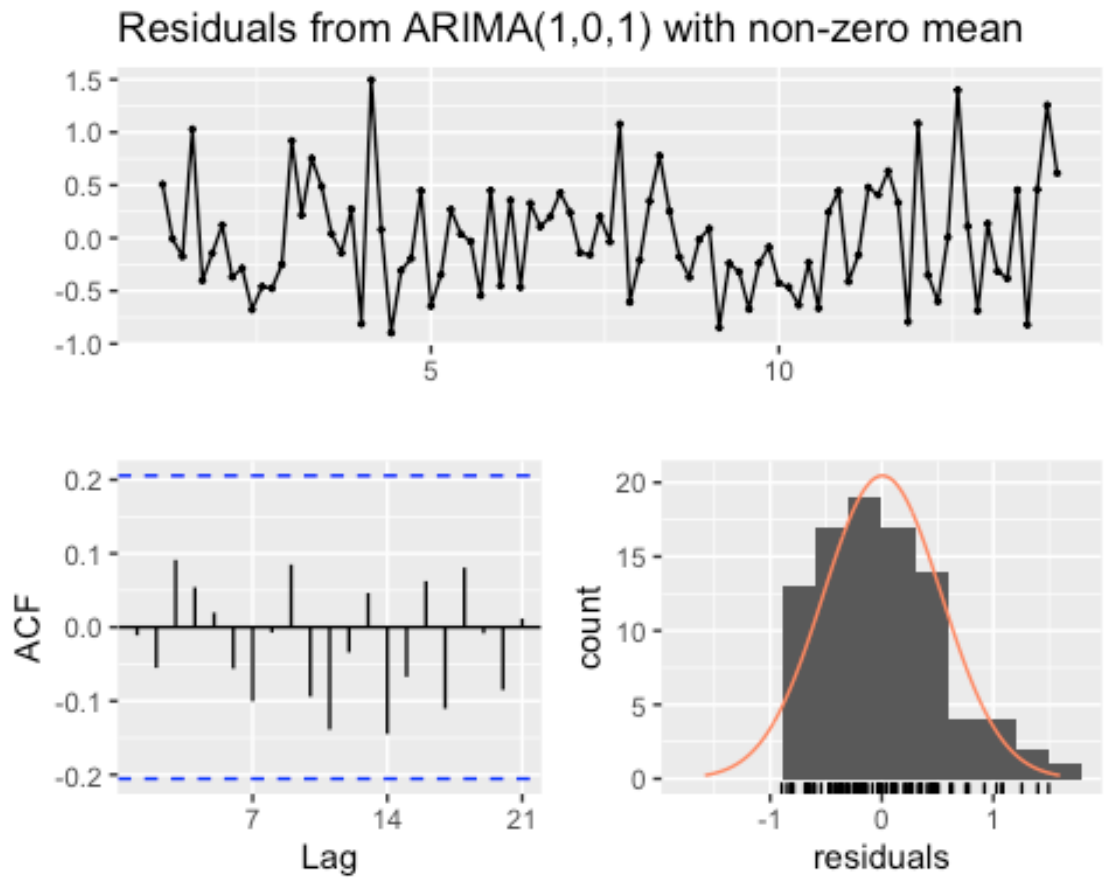
```
autoplot(modeldata2)
```



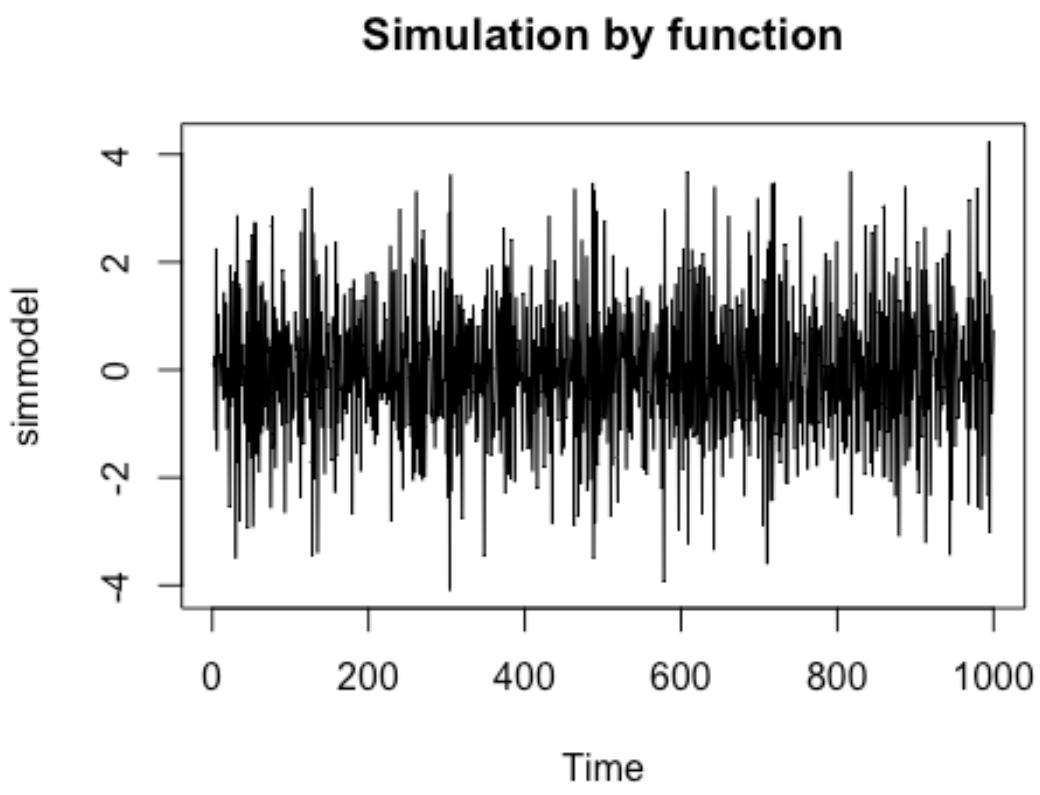

```
plot.ts(modeldata2$residuals, xlab = 'Residuals', main = 'ARIMA (1,0,1)
residuals vs Time')
```




```
checkresiduals(modeldata2)
```

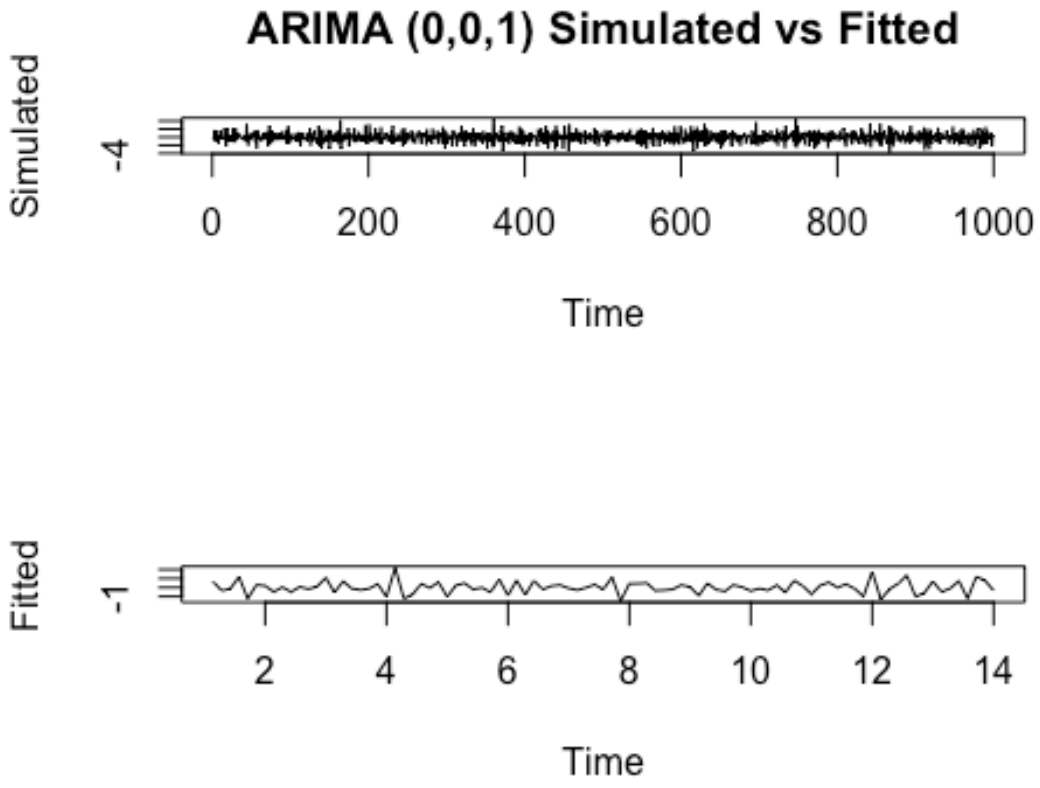



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1) with non-zero mean
## Q* = 9.1452, df = 11, p-value = 0.6085
##
## Model df: 3.   Total lags used: 14

coef(modeldata2)

##         ar1         ma1   intercept
## -0.13971145 -0.67408147  0.00503714

#2.8
simmodel <- arima.sim(model = list(ma = -0.7485), n=1000)
plot.ts(simmodel, main = 'Simulation by function')
```

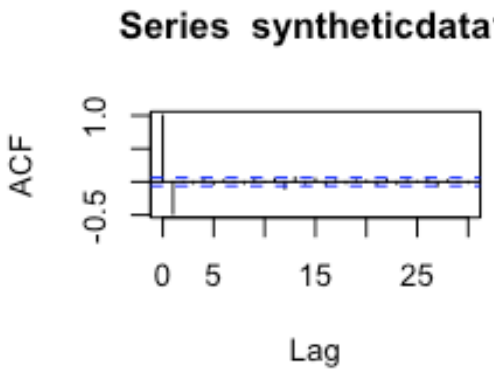


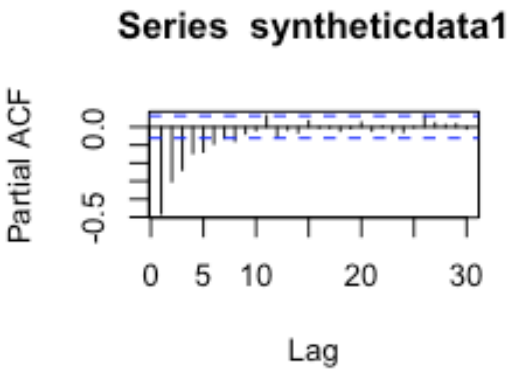

```r
#2.9
#ARIMA(0,0,1), where only one moving average is applied to the function for simulation

set.seed(123) #begin simulation
n <- 1000 #changed from 92, Immediately looking at the data when n=92, the coefficients are wildly different. Althought the ACF and PACF distributions look identical, it's important that simulation extends previous knowledge, thus the number of iterations will be 1000 as it will ensure a more accurate ARIMA model.

syntheticts1 <- ts(rnorm(n), frequency = 7) #adding variables into simulation

syntheticdata1 <- syntheticts1[1:2]


for(i in 3:n){ #loops for simulation using 1000 iterations
   syntheticdata1[i]      <- syntheticts1[i - 1]  * -0.7485 + syntheticts1[i]

   #ARIMA(0,0,1) equation + coefficient is derived from modeldata

}
```

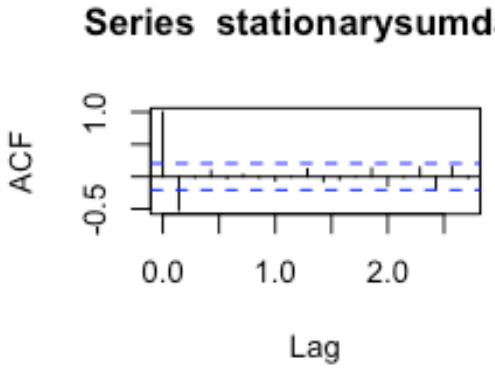
```r
syntheticdata1 <- ts(syntheticdata1)

syntheticdiff1 <- diff(syntheticdata1)

par(mfrow=c(2,1))
plot(syntheticdata1, main = 'ARIMA (0,0,1) Simulated vs Fitted', ylab =
'Simulated')
plot(stationarysumdata, ylab = 'Fitted')
```

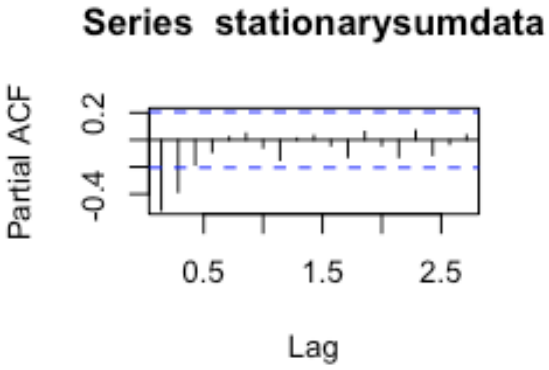



```r
par(mfrow=c(2,2))
acf(syntheticdata1)
pacf(syntheticdata1)
acf(stationarysumdata)
pacf(stationarysumdata)
```

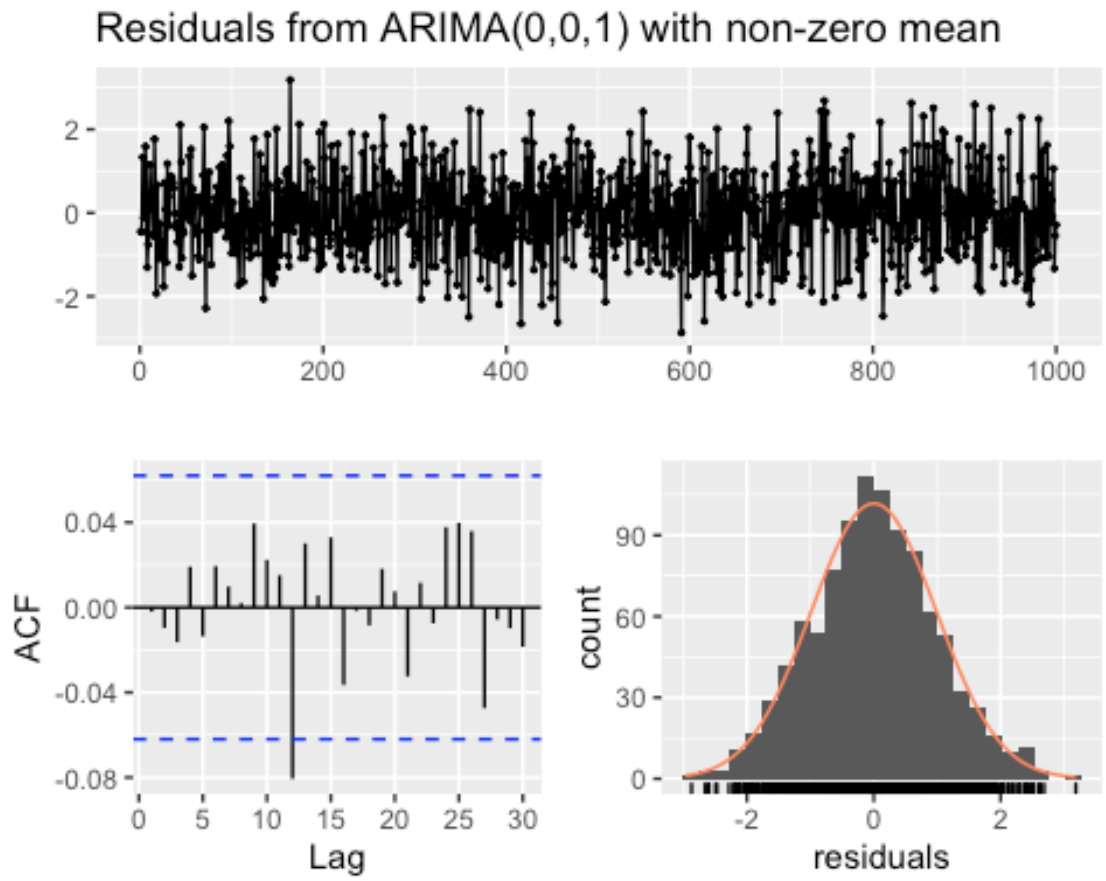






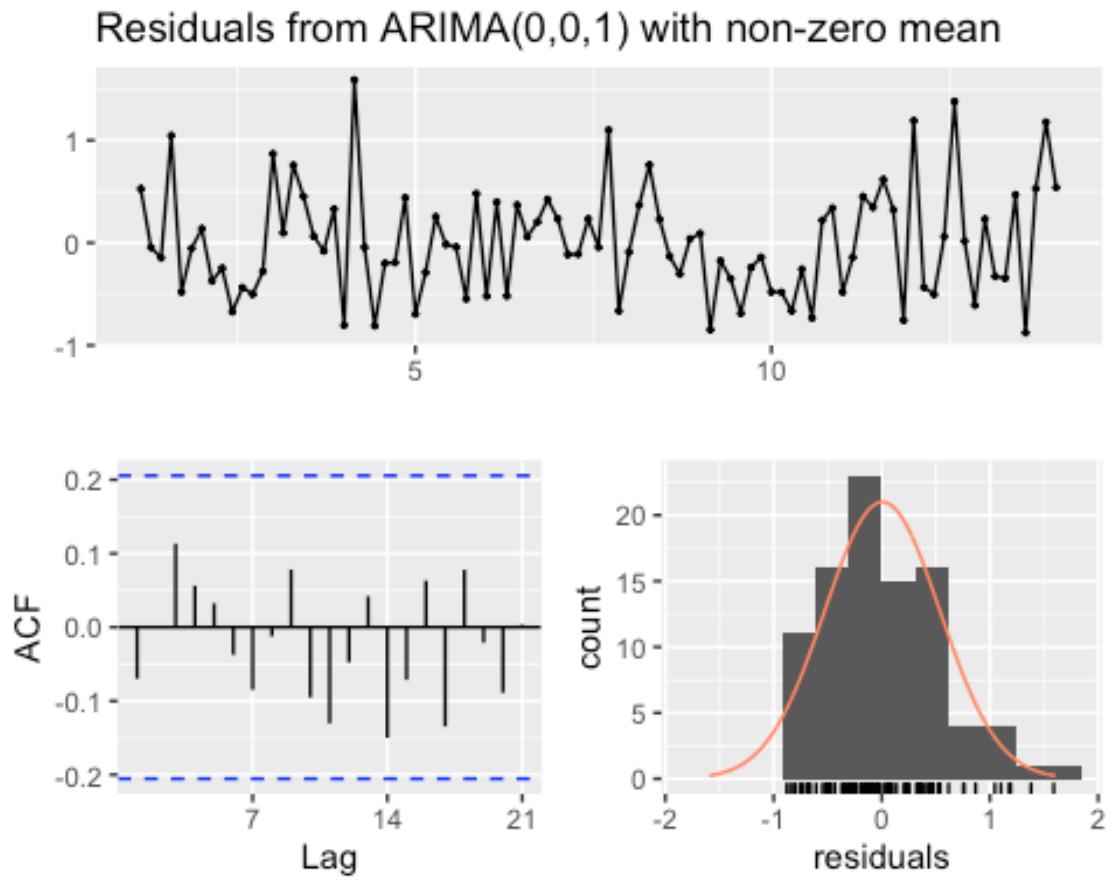



```r
#2.10
modelsynthetic <- arima(syntheticdata1, order = c(0,0,1))
#modelsynthetic

#coef(modelsynthetic)
#coef(modeldata)
#coef(modeldata) - coef(modelsynthetic)

checkresiduals(modelsynthetic)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1) with non-zero mean
## Q* = 3.5002, df = 8, p-value = 0.8992
##
## Model df: 2.   Total lags used: 10
```

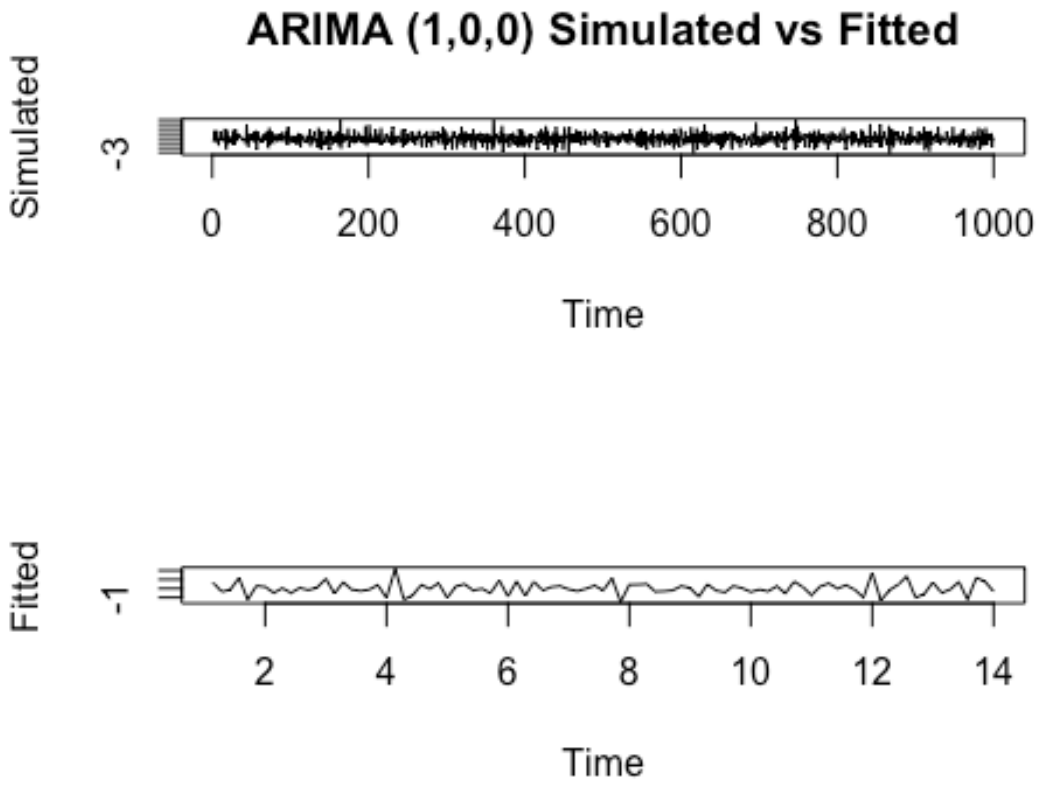
```
checkresiduals(modeldata)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1) with non-zero mean
## Q* = 9.2821, df = 12, p-value = 0.6787
##
## Model df: 2.   Total lags used: 14
```

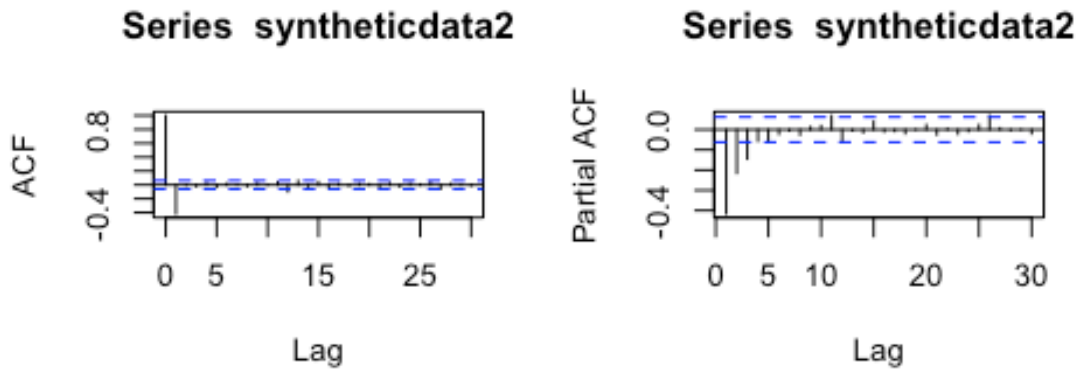
```r
#autoplot(modelsynthetic)

#2.11
#ARIMA(1,0,0), where only one moving average is applied to the function for simulation

set.seed(123) #begin simulation
n <- 1000 #changed from 92

syntheticts2 <- ts(rnorm(n), frequency = 7) #adding variables into simulation

syntheticdata2 <- syntheticts2[1:2]


for(i in 3:n){ #loops for simulation using 1000 iterations
```

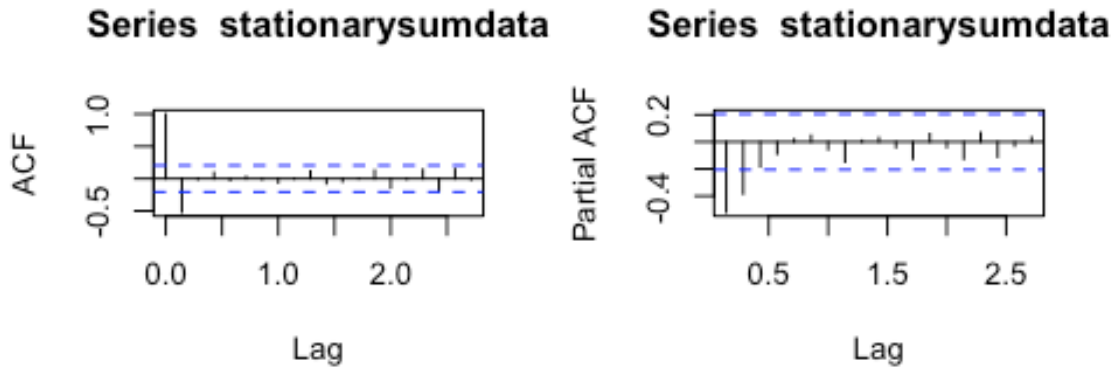
```
   syntheticdata2[i]      <- syntheticts2[i - 1]  * -0.52055452 +
syntheticts1[i] #MA(1) equation

      #ARIMA(1,0,0) equation + coefficient is derived from modeldata1

}

syntheticdata2 <- ts(syntheticdata2)

syntheticdiff2 <- diff(syntheticdata2)

par(mfrow=c(2,1))
plot(syntheticdata2, main = 'ARIMA (1,0,0) Simulated vs Fitted', ylab =
'Simulated')
plot(stationarysumdata, ylab = 'Fitted')
```



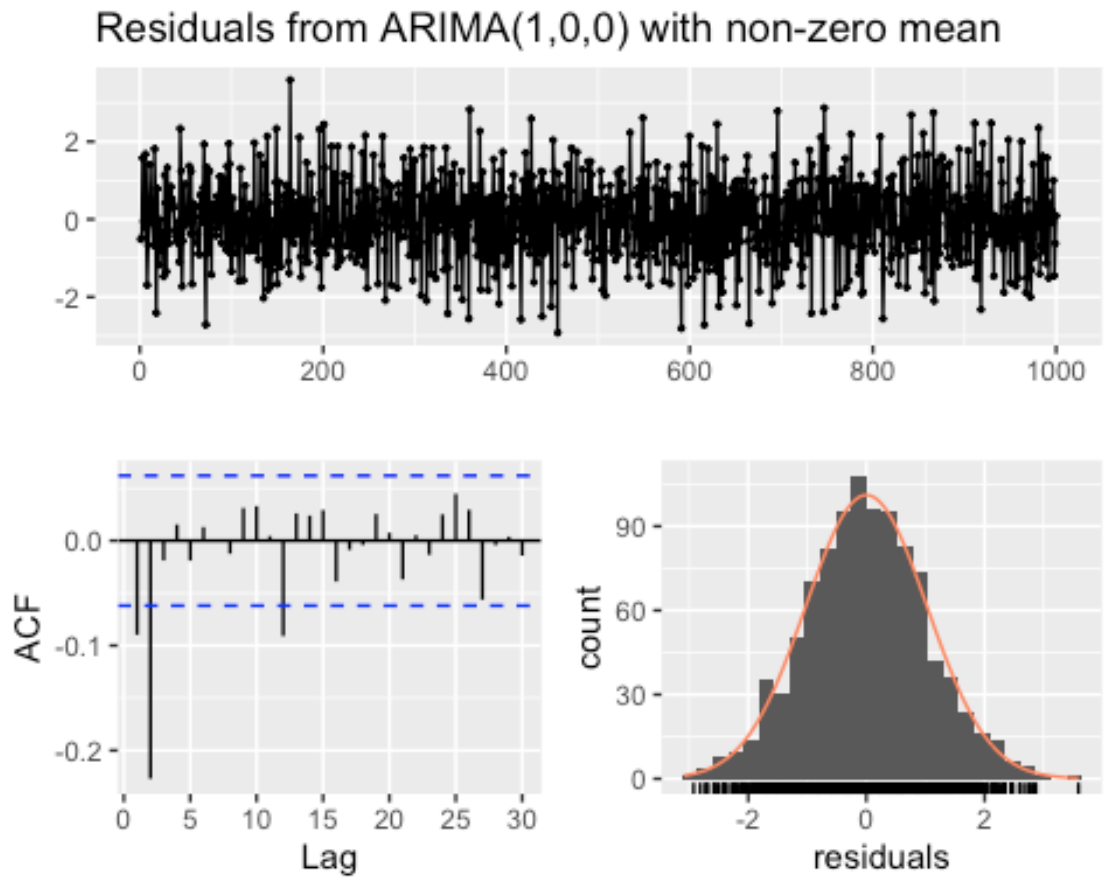

```
#summary(syntheticdata2)
#summary(stationarysumdata)

par(mfrow=c(2,2))
acf(syntheticdata2)
pacf(syntheticdata2)
```

```r
acf(stationarysumdata)
pacf(stationarysumdata)
```

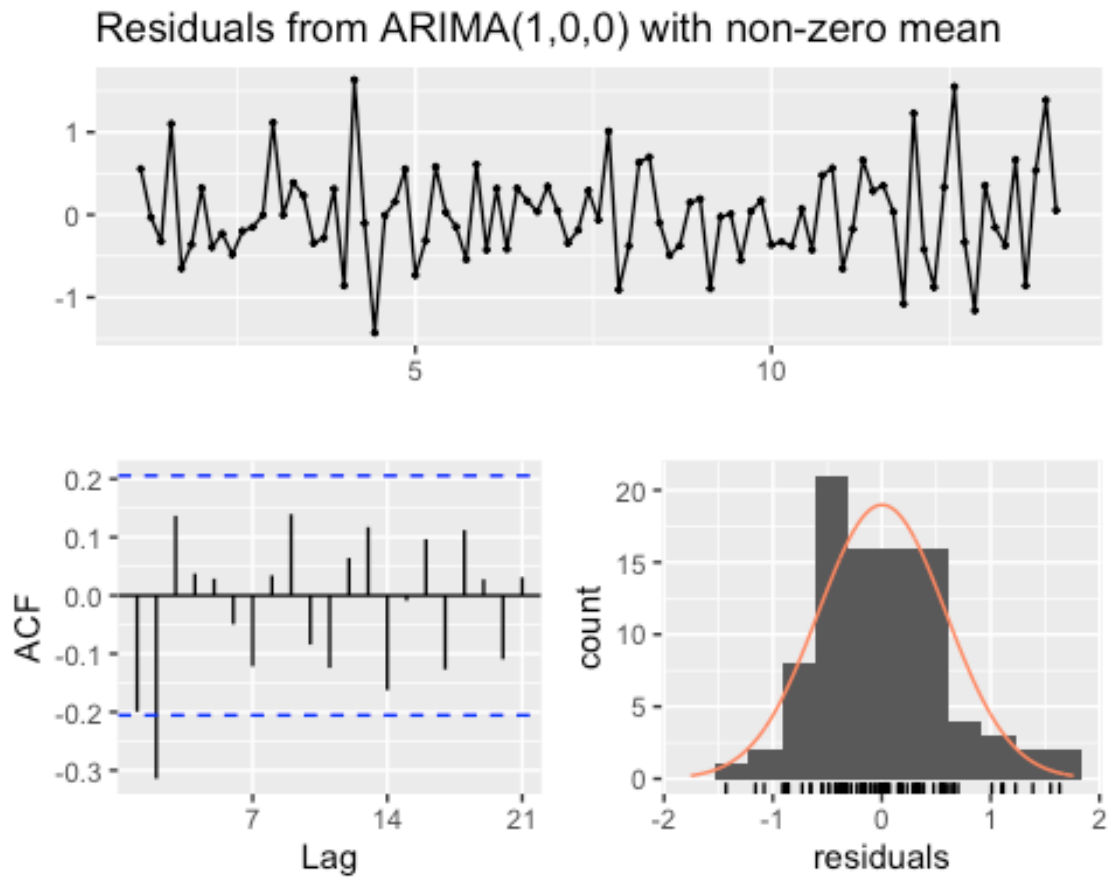





```r
#2.12

modelsynthetic2 <- arima(syntheticdata2, order = c(1,0,0))
#modelsynthetic2

#coef(modelsynthetic2)
#coef(modeldata1)
#coef(modeldata1) - coef(modelsynthetic2)

checkresiduals(modelsynthetic2)
```

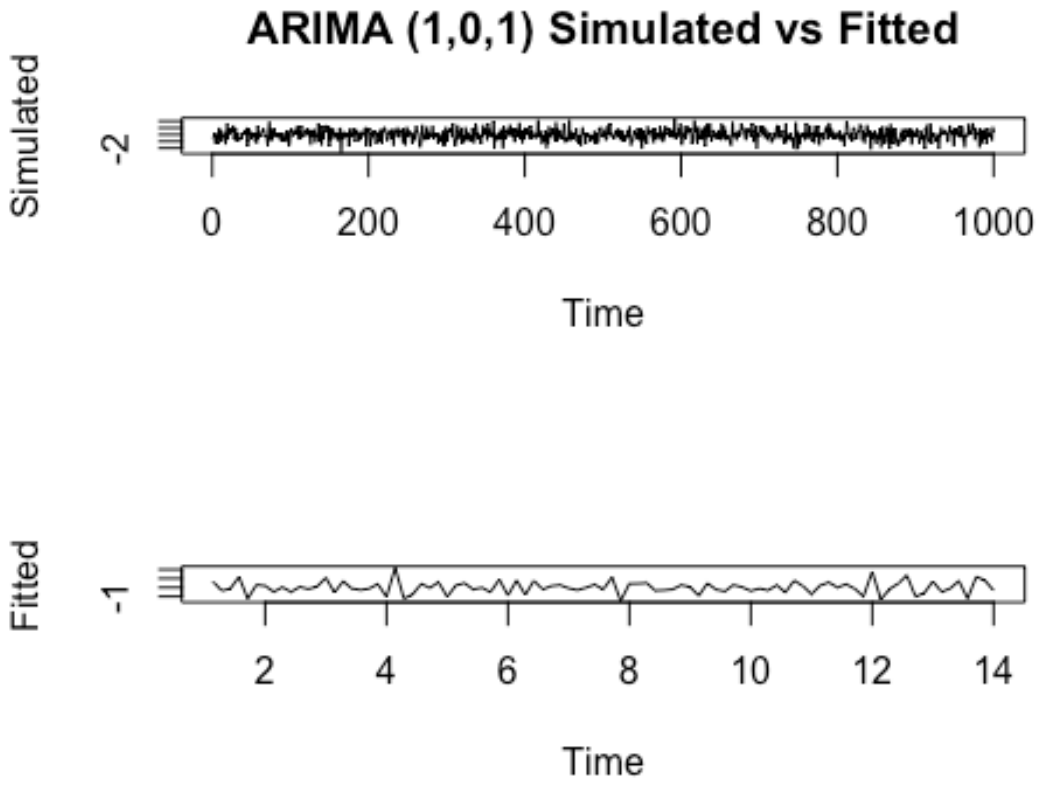



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with non-zero mean
## Q* = 63.219, df = 8, p-value = 1.085e-10
##
## Model df: 2.   Total lags used: 10

checkresiduals(modeldata1)
```



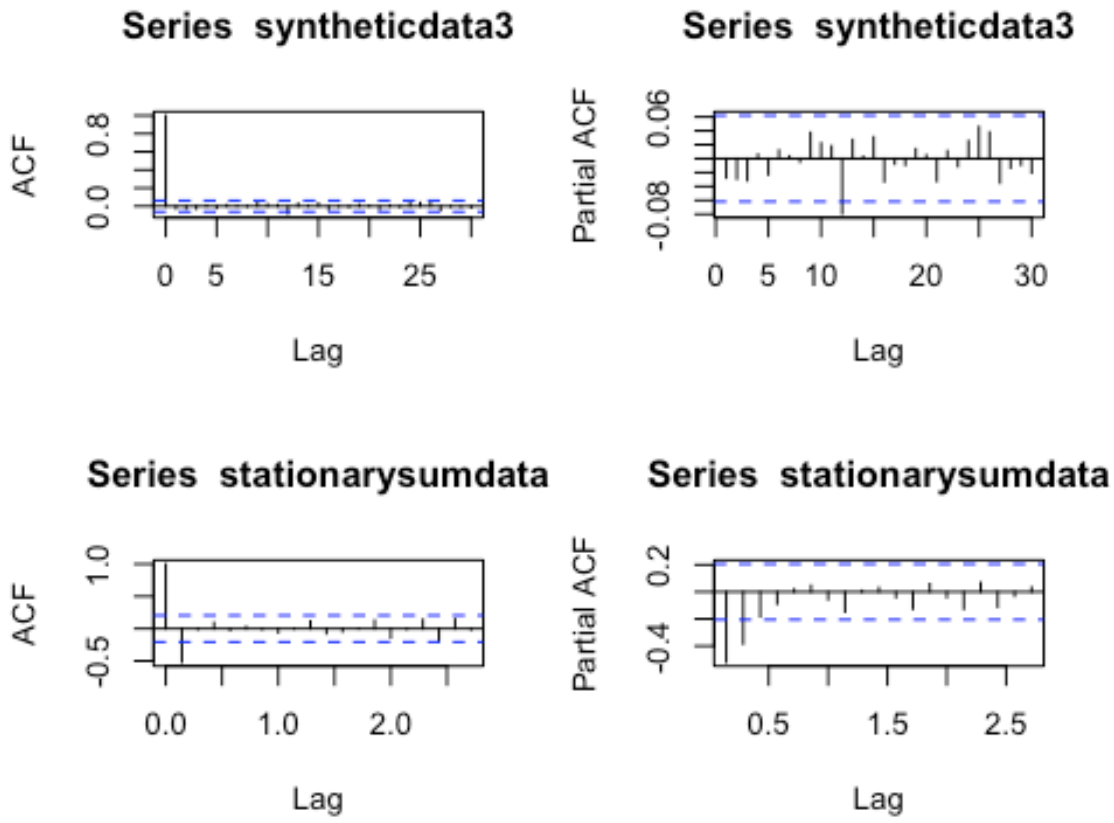

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with non-zero mean
## Q* = 26.267, df = 12, p-value = 0.009838
##
## Model df: 2.   Total lags used: 14

#autoplot(modelsynthetic2)

#2.13
#ARIMA(1,0,1), where only one moving average and one auto regression
coefficient is applied to the function for simulation

set.seed(123) #begin simulation
n <- 1000 #changed from 92

syntheticts3 <- ts(rnorm(n), frequency = 7) #adding variables into simulation

syntheticdata3 <- syntheticts3[1:2]


for(i in 3:n){ #loops for simulation using 1000 iterations
```

```
   syntheticdata3[i]      <- syntheticts2[i - 1]  *  -0.13971145 +
syntheticts1[i - 1] * -0.67408147  #coefficients derived from modeldata2

       #ARIMA(1,0,1) equation + coefficient is derived from modeldata2
}

syntheticdata3 <- ts(syntheticdata3)

syntheticdiff3 <- diff(syntheticdata3)

par(mfrow=c(2,1))
plot(syntheticdata3, main = 'ARIMA (1,0,1) Simulated vs Fitted', ylab =
'Simulated')
plot(stationarysumdata, ylab = 'Fitted')
```

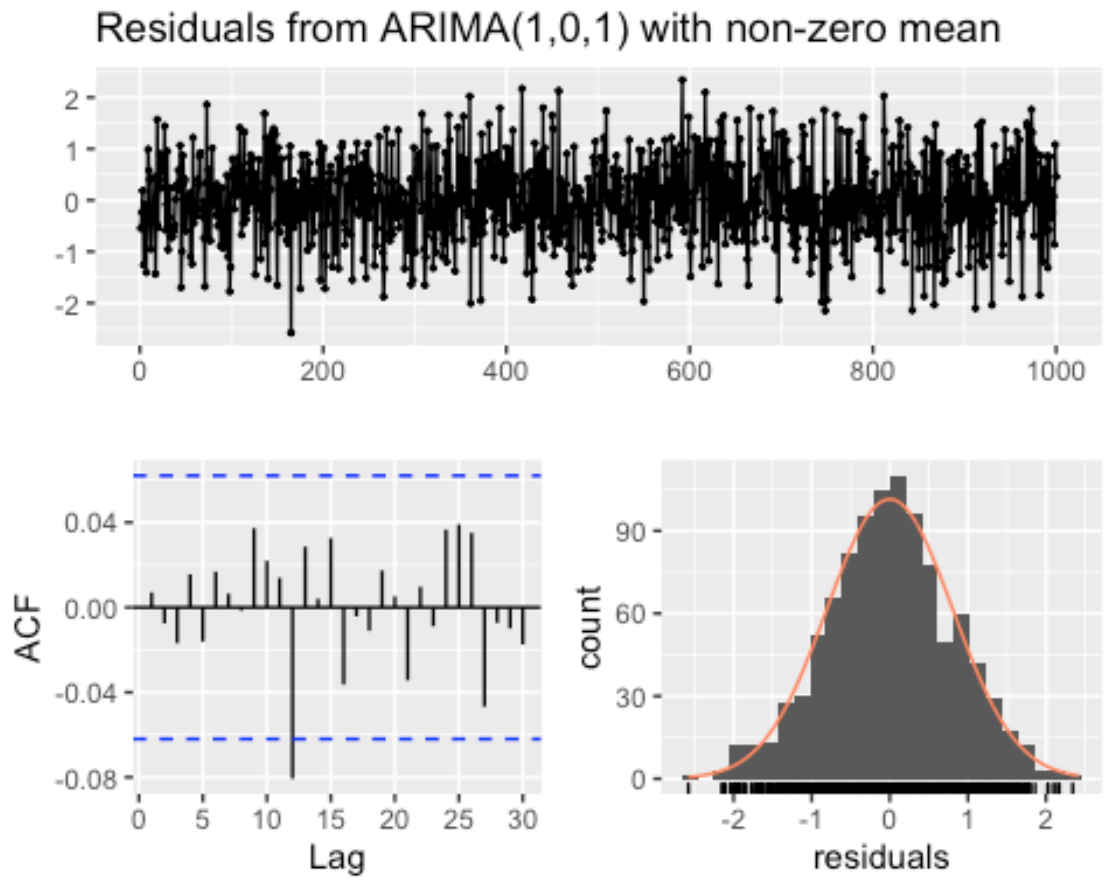



```
summary(syntheticdata3)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.63754 -0.54243 -0.01616 -0.01457  0.51133  2.28657

summary(stationarysumdata)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.48516 -0.36414 -0.02326  0.01230  0.42549  2.19504
```

```r
par(mfrow=c(2,2))
acf(syntheticdata3)
pacf(syntheticdata3)
acf(stationarysumdata)
pacf(stationarysumdata)
```

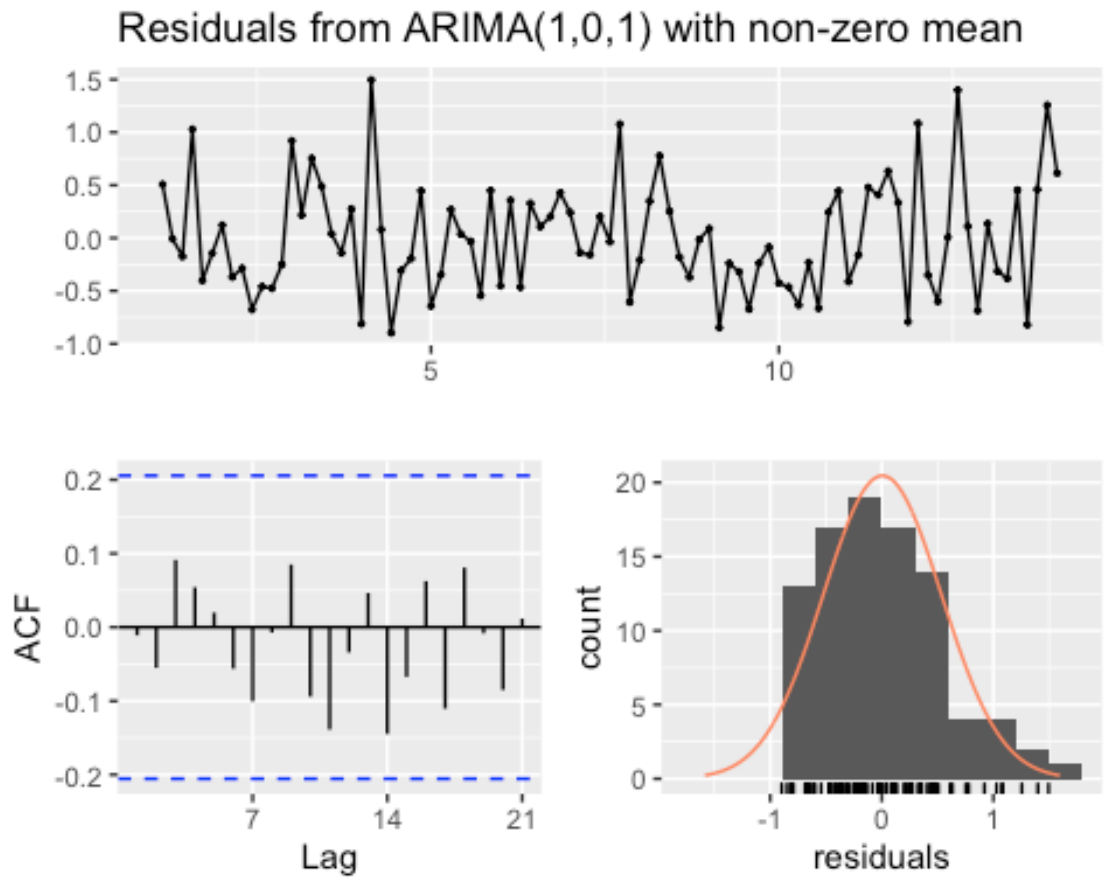



```r
#2.14

modelsynthetic3 <- arima(syntheticdata3, order = c(1,0,1))
#modelsynthetic3

#coef(modelsynthetic3)
#coef(modeldata2)
#coef(modeldata2) - coef(modelsynthetic3)

checkresiduals(modelsynthetic3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1) with non-zero mean
## Q* = 3.1233, df = 7, p-value = 0.8734
##
## Model df: 3.   Total lags used: 10
```

```
checkresiduals(modeldata2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1) with non-zero mean
## Q* = 9.1452, df = 11, p-value = 0.6085
##
## Model df: 3.   Total lags used: 14
```

```
#3.1

coef(modeldata) - coef(modelsynthetic)

##          ma1    intercept
## 0.0276586565 0.0004896633

coef(modeldata1) - coef(modelsynthetic2)

##          ar1    intercept
## -0.104948950  0.003691287

coef(modeldata2) - coef(modelsynthetic3)

##         ar1         ma1   intercept
## -0.73774293 -0.03967629  0.01954557
```

```r
#3.2

nonlinearityTest(syntheticdata1, verbose = TRUE)
```

```
##       ** Teraesvirta's neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  1.455488  df =  2  p-value =  0.4829974
##
##       ** White neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  1.075368  df =  2  p-value =  0.5840996
##
##       ** Keenan's one-degree test for nonlinearity  **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.122667  p-value =  0.7262355
##
##       ** McLeod-Li test  **
##       Null hypothesis: The time series follows some ARIMA process
##       Maximum p-value =  7.186474e-13
##
##       ** Tsay's Test for nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  1.036559  p-value =  0.3963836
##
##       ** Likelihood ratio test for threshold nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       Alternativce hypothesis: The time series follows some TAR process
##       X-squared =  14.16178  p-value =  0.1700933

## $Terasvirta
##
##  Teraesvirta Neural Network Test
##
## data:  ts(time.series)
## X-squared = 1.4555, df = 2, p-value = 0.483
##
##
## $White
##
##  White Neural Network Test
##
## data:  ts(time.series)
## X-squared = 1.0754, df = 2, p-value = 0.5841
##
##
## $Keenan
## $Keenan$test.stat
## [1] 0.122667
##
## $Keenan$df1
```

```
## [1] 1
##
## $Keenan$df2
## [1] 974
##
## $Keenan$p.value
## [1] 0.7262355
##
## $Keenan$order
## [1] 12
##
##
## $McLeodLi
## $McLeodLi$p.values
##  [1] 1.443290e-15 1.498801e-14 9.359180e-14 1.243450e-13 1.544320e-13
##  [6] 7.671641e-14 5.950795e-14 8.060219e-14 1.402212e-13 4.357625e-13
## [11] 7.186474e-13 8.970602e-14 2.609024e-14 6.439294e-15 1.632028e-14
## [16] 4.329870e-14 1.053602e-13 1.566525e-13 9.436896e-14 1.183498e-13
## [21] 2.162714e-13 1.760814e-13 3.081979e-13 4.773959e-15 3.774758e-15
## [26] 6.439294e-15 9.769963e-15 9.214851e-15 1.754152e-14 1.298961e-14
##
##
## $Tsay
## $Tsay$test.stat
## [1] 1.036559
##
## $Tsay$p.value
## [1] 0.3963836
##
## $Tsay$order
## [1] 12
##
##
## $TarTest
## $TarTest$percentiles
## [1] 25 75
##
## $TarTest$test.statistic
## [1] 14.16178
##
## $TarTest$p.value
## [1] 0.1700933
```

```
nonlinearityTest(stationarysumdata, verbose = TRUE)
```

```
##       ** Teraesvirta's neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  2.027456  df =  2  p-value =  0.3628636
##
##       ** White neural network test  **
```

```
##       Null hypothesis: Linearity in "mean"
##       X-squared =  1.694883  df =  2  p-value =  0.4285099
##
##       ** Keenan's one-degree test for nonlinearity  **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.799738  p-value =  0.3737576
##
##       ** McLeod-Li test  **
##       Null hypothesis: The time series follows some ARIMA process
##       Maximum p-value =  0.4046276
##
##       ** Tsay's Test for nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.7492603  p-value =  0.6118222
##
##       ** Likelihood ratio test for threshold nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       Alternativce hypothesis: The time series follows some TAR process
##       X-squared =  4.798722  p-value =  0.2020134

## $Terasvirta
##
##  Teraesvirta Neural Network Test
##
## data:  ts(time.series)
## X-squared = 2.0275, df = 2, p-value = 0.3629
##
##
## $White
##
##  White Neural Network Test
##
## data:  ts(time.series)
## X-squared = 1.6949, df = 2, p-value = 0.4285
##
##
## $Keenan
## $Keenan$test.stat
## [1] 0.799738
##
## $Keenan$df1
## [1] 1
##
## $Keenan$df2
## [1] 83
##
## $Keenan$p.value
## [1] 0.3737576
##
## $Keenan$order
```

```
## [1] 3
##
##
## $McLeodLi
## $McLeodLi$p.values
##  [1] 0.0002371136 0.0009117580 0.0014175476 0.0032968081 0.0068071287
##  [6] 0.0131643775 0.0238800753 0.0400735382 0.0566407323 0.0852897182
## [11] 0.1173419650 0.1402354503 0.1865495182 0.2368196433 0.2456424816
## [16] 0.2548110251 0.3130805073 0.3750653611 0.4046276316
##
##
## $Tsay
## $Tsay$test.stat
## [1] 0.7492603
##
## $Tsay$p.value
## [1] 0.6118222
##
## $Tsay$order
## [1] 3
##
##
## $TarTest
## $TarTest$percentiles
## [1] 25 75
##
## $TarTest$test.statistic
## [1] 4.798722
##
## $TarTest$p.value
## [1] 0.2020134
```

```
nonlinearityTest(syntheticdata2, verbose = TRUE)
```

```
##       ** Teraesvirta's neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  1.663517  df =  2  p-value =  0.4352831
##
##       ** White neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  1.87707  df =  2  p-value =  0.3912005
##
##       ** Keenan's one-degree test for nonlinearity  **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.09552841  p-value =  0.7573283
##
##       ** McLeod-Li test  **
##       Null hypothesis: The time series follows some ARIMA process
##       Maximum p-value =  5.403136e-07
##
```

```
##       ** Tsay's Test for nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.9545004  p-value =  0.5023176
##
##       ** Likelihood ratio test for threshold nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       Alternativce hypothesis: The time series follows some TAR process
##       X-squared =  6.175857  p-value =  0.04613262

## $Terasvirta
##
##  Teraesvirta Neural Network Test
##
## data:  ts(time.series)
## X-squared = 1.6635, df = 2, p-value = 0.4353
##
##
## $White
##
##  White Neural Network Test
##
## data:  ts(time.series)
## X-squared = 1.8771, df = 2, p-value = 0.3912
##
##
## $Keenan
## $Keenan$test.stat
## [1] 0.09552841
##
## $Keenan$df1
## [1] 1
##
## $Keenan$df2
## [1] 988
##
## $Keenan$p.value
## [1] 0.7573283
##
## $Keenan$order
## [1] 5
##
##
## $McLeodLi
## $McLeodLi$p.values
##  [1] 2.780942e-09 2.000407e-08 9.616448e-08 8.049551e-08 1.327192e-07
##  [6] 6.102310e-08 4.948186e-08 8.221706e-08 1.151418e-07 2.817803e-07
## [11] 5.403136e-07 7.396539e-08 4.708063e-08 1.097184e-08 2.423489e-08
## [16] 5.221833e-08 1.059423e-07 1.640145e-07 1.294142e-07 2.089150e-07
## [21] 2.532552e-07 2.269501e-07 3.502254e-07 2.495636e-08 1.895158e-08
## [26] 2.998223e-08 4.695746e-08 4.498599e-08 7.105069e-08 6.138993e-08
```

```
##
##
## $Tsay
## $Tsay$test.stat
## [1] 0.9545004
##
## $Tsay$p.value
## [1] 0.5023176
##
## $Tsay$order
## [1] 5
##
##
## $TarTest
## $TarTest$percentiles
## [1] 24.92462 75.07538
##
## $TarTest$test.statistic
## [1] 6.175857
##
## $TarTest$p.value
## [1] 0.04613262
```

```
nonlinearityTest(stationarysumdata, verbose = TRUE)
```

```
##       ** Teraesvirta's neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  2.027456  df =  2  p-value =  0.3628636
##
##       ** White neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  1.335458  df =  2  p-value =  0.5128719
##
##       ** Keenan's one-degree test for nonlinearity  **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.799738  p-value =  0.3737576
##
##       ** McLeod-Li test  **
##       Null hypothesis: The time series follows some ARIMA process
##       Maximum p-value =  0.4046276
##
##       ** Tsay's Test for nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.7492603  p-value =  0.6118222
##
##       ** Likelihood ratio test for threshold nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       Alternativce hypothesis: The time series follows some TAR process
##       X-squared =  4.798722  p-value =  0.2020134
```

```
## $Terasvirta
##
##  Teraesvirta Neural Network Test
##
## data:  ts(time.series)
## X-squared = 2.0275, df = 2, p-value = 0.3629
##
##
## $White
##
##  White Neural Network Test
##
## data:  ts(time.series)
## X-squared = 1.3355, df = 2, p-value = 0.5129
##
##
## $Keenan
## $Keenan$test.stat
## [1] 0.799738
##
## $Keenan$df1
## [1] 1
##
## $Keenan$df2
## [1] 83
##
## $Keenan$p.value
## [1] 0.3737576
##
## $Keenan$order
## [1] 3
##
##
## $McLeodLi
## $McLeodLi$p.values
##  [1] 0.0002371136 0.0009117580 0.0014175476 0.0032968081 0.0068071287
##  [6] 0.0131643775 0.0238800753 0.0400735382 0.0566407323 0.0852897182
## [11] 0.1173419650 0.1402354503 0.1865495182 0.2368196433 0.2456424816
## [16] 0.2548110251 0.3130805073 0.3750653611 0.4046276316
##
##
## $Tsay
## $Tsay$test.stat
## [1] 0.7492603
##
## $Tsay$p.value
## [1] 0.6118222
##
## $Tsay$order
## [1] 3
```

```
##
##
## $TarTest
## $TarTest$percentiles
## [1] 25 75
##
## $TarTest$test.statistic
## [1] 4.798722
##
## $TarTest$p.value
## [1] 0.2020134
```

```
nonlinearityTest(syntheticdata3, verbose = TRUE)
```

```
##       ** Teraesvirta's neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  1.168975  df =  2  p-value =  0.5573914
##
##       ** White neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  0.4995581  df =  2  p-value =  0.7789729
##
##       ** Keenan's one-degree test for nonlinearity  **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.6036085  p-value =  0.4373886
##
##       ** McLeod-Li test  **
##       Null hypothesis: The time series follows some ARIMA process
##       Maximum p-value =  0.7759115
##
##       ** Tsay's Test for nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.02485515  p-value =  0.8747605
##
##       ** Likelihood ratio test for threshold nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       Alternativce hypothesis: The time series follows some TAR process
##       X-squared =  1151.57  p-value =  2.598402e-249
```

```
## $Terasvirta
##
##  Teraesvirta Neural Network Test
##
## data:  ts(time.series)
## X-squared = 1.169, df = 2, p-value = 0.5574
##
##
## $White
##
##  White Neural Network Test
```

```
##
## data:  ts(time.series)
## X-squared = 0.49956, df = 2, p-value = 0.779
##
##
## $Keenan
## $Keenan$test.stat
## [1] 0.6036085
##
## $Keenan$df1
## [1] 1
##
## $Keenan$df2
## [1] 998
##
## $Keenan$p.value
## [1] 0.4373886
##
## $Keenan$order
## [1] 0
##
##
## $McLeodLi
## $McLeodLi$p.values
##  [1] 0.77591145 0.52518465 0.63563227 0.25740513 0.18952463 0.27753119
##  [7] 0.24358552 0.24640200 0.27879904 0.35148272 0.33472100 0.08018685
## [13] 0.11173787 0.14886027 0.17067614 0.21054286 0.23810454 0.28462482
## [19] 0.34250194 0.40083532 0.42031946 0.46720126 0.50210376 0.56123686
## [25] 0.41213568 0.44400714 0.41382261 0.46192384 0.51304570 0.53392093
##
##
## $Tsay
## $Tsay$test.stat
## [1] 0.02485515
##
## $Tsay$p.value
## [1] 0.8747605
##
## $Tsay$order
## [1] 1
##
##
## $TarTest
## $TarTest$percentiles
## [1] 24.92492 75.07508
##
## $TarTest$test.statistic
## [1] 1151.57
##
```

```
## $TarTest$p.value
## [1] 2.598402e-249
```

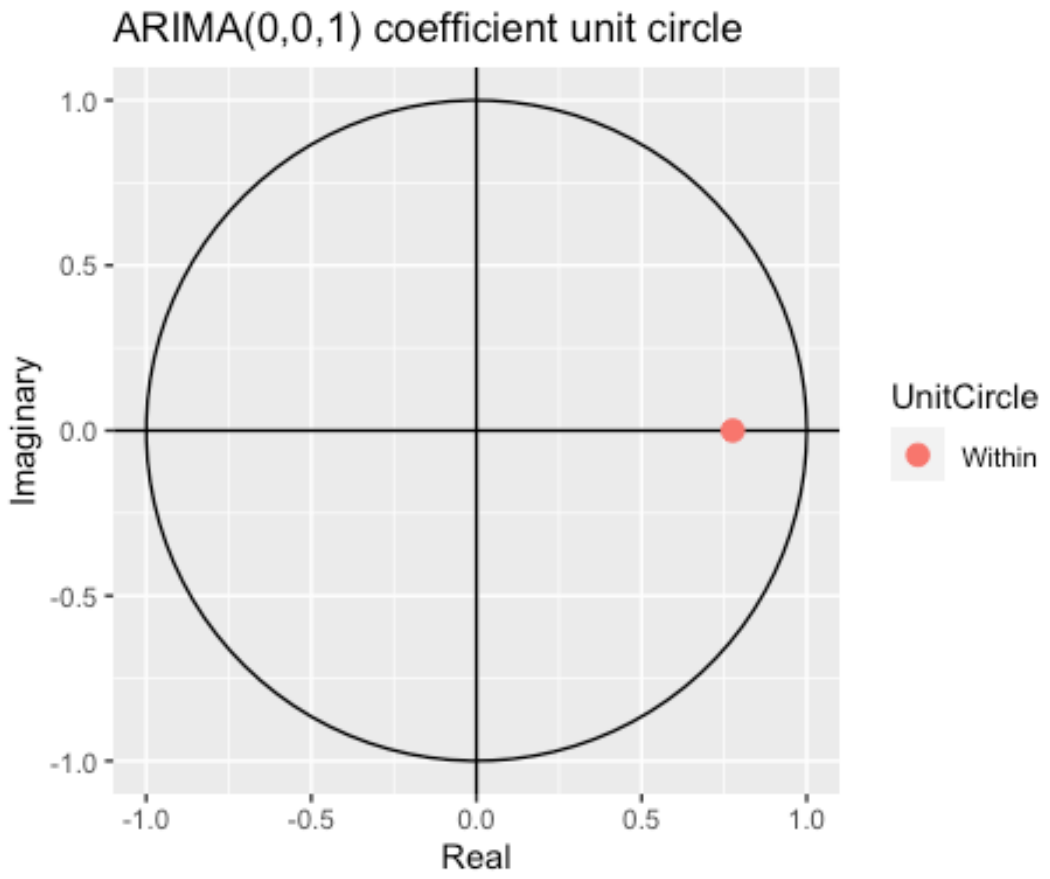
```
nonlinearityTest(stationarysumdata, verbose = TRUE)
```

```
##       ** Teraesvirta's neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  2.027456  df =  2  p-value =  0.3628636
##
##       ** White neural network test  **
##       Null hypothesis: Linearity in "mean"
##       X-squared =  1.300206  df =  2  p-value =  0.5219921
##
##       ** Keenan's one-degree test for nonlinearity  **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.799738  p-value =  0.3737576
##
##       ** McLeod-Li test  **
##       Null hypothesis: The time series follows some ARIMA process
##       Maximum p-value =  0.4046276
##
##       ** Tsay's Test for nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       F-stat =  0.7492603  p-value =  0.6118222
##
##       ** Likelihood ratio test for threshold nonlinearity **
##       Null hypothesis: The time series follows some AR process
##       Alternativce hypothesis: The time series follows some TAR process
##       X-squared =  4.798722  p-value =  0.2020134
```

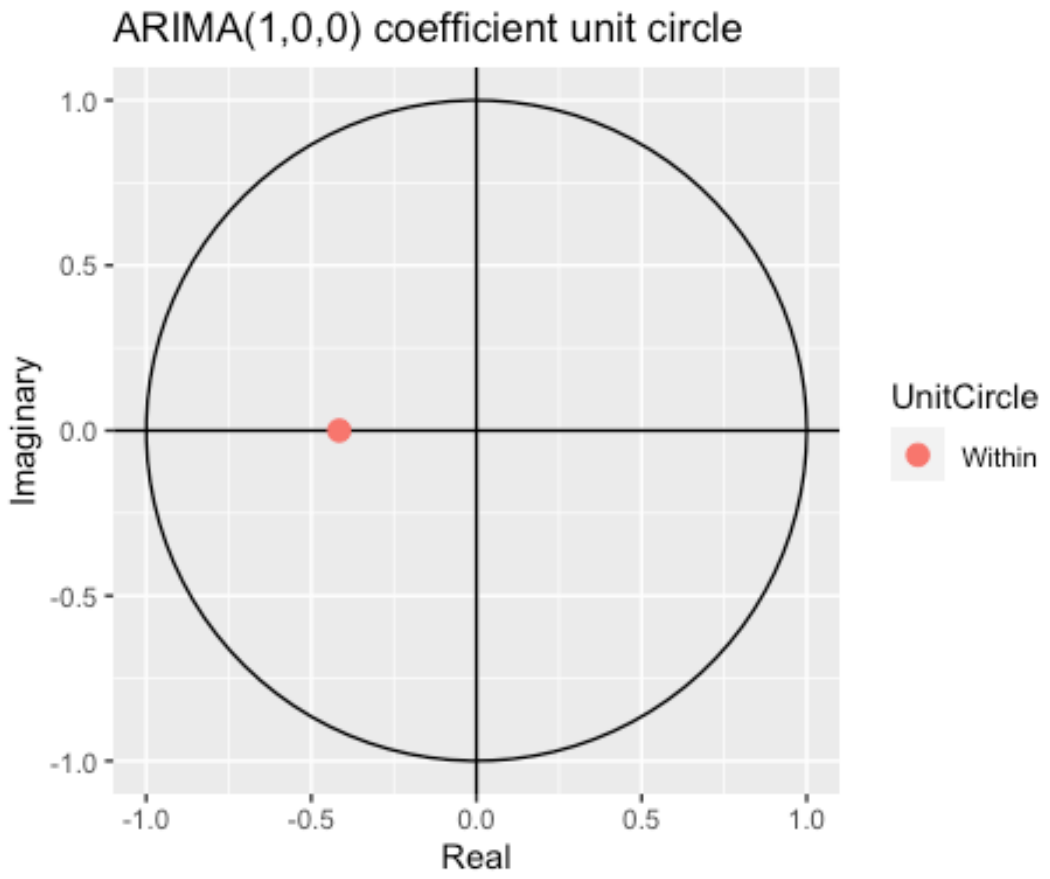
```
## $Terasvirta
##
##  Teraesvirta Neural Network Test
##
## data:  ts(time.series)
## X-squared = 2.0275, df = 2, p-value = 0.3629
##
##
## $White
##
##  White Neural Network Test
##
## data:  ts(time.series)
## X-squared = 1.3002, df = 2, p-value = 0.522
##
##
## $Keenan
## $Keenan$test.stat
## [1] 0.799738
##
```

```
## $Keenan$df1
## [1] 1
##
## $Keenan$df2
## [1] 83
##
## $Keenan$p.value
## [1] 0.3737576
##
## $Keenan$order
## [1] 3
##
##
## $McLeodLi
## $McLeodLi$p.values
##  [1] 0.0002371136 0.0009117580 0.0014175476 0.0032968081 0.0068071287
##  [6] 0.0131643775 0.0238800753 0.0400735382 0.0566407323 0.0852897182
## [11] 0.1173419650 0.1402354503 0.1865495182 0.2368196433 0.2456424816
## [16] 0.2548110251 0.3130805073 0.3750653611 0.4046276316
##
##
## $Tsay
## $Tsay$test.stat
## [1] 0.7492603
##
## $Tsay$p.value
## [1] 0.6118222
##
## $Tsay$order
## [1] 3
##
##
## $TarTest
## $TarTest$percentiles
## [1] 25 75
##
## $TarTest$test.statistic
## [1] 4.798722
##
## $TarTest$p.value
## [1] 0.2020134
```

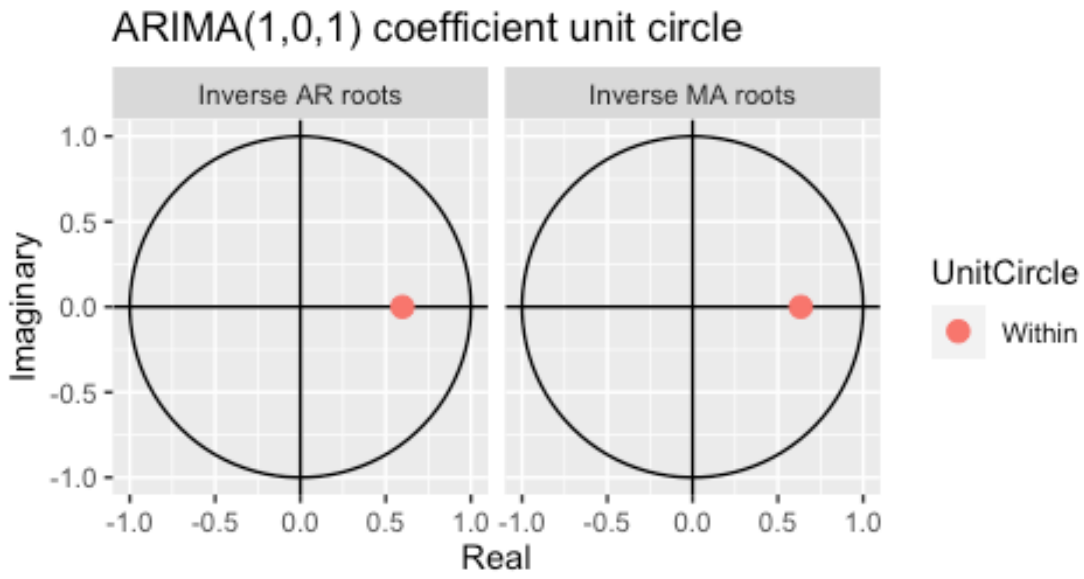
```r
#3.3

par(mfrow = c(3,1))
autoplot(modelsynthetic, main = 'ARIMA(0,0,1) coefficient unit circle')
```




```
autoplot(modelsynthetic2, main = 'ARIMA(1,0,0) coefficient unit circle')
```




```
autoplot(modelsynthetic3, main = 'ARIMA(1,0,1) coefficient unit circle')
```


ARIMA(1,0,1) coefficient unit circle
Inverse AR roots
Inverse MA roots
Imaginary
Real
1.0
0.5
0.0
-0.5
-1.0
UnitCircle
Within