ForcastStockMarket

Saeid Rezaei

January 8, 2018

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
#################################################################################################
##############################
# Script name: ForcastStockMarketARIMA.R
# Porpouse   : This script is developed to forcast the S&P 500 Index over the 10 years
#              I have used the ARIMA model (Time Series) in R
# Data source: Data source could be off-line (marketPriceHistory.csv) or online SP&500
# R Package usagae:
# ggplot2
# forecast
# plotly
# ggfortify
# tseries
# gridExtra
# docstring
# here
#################################################################################################
##############################
#Developer         Date              Version                    Reason
#Saeid Rezaei     2017-12-20           0                      Initial Version
#Saeid Rezaei     2018-01-05           1                   Added ARIMA method to script
#################################################################################################
##############################

#By Milind Paradkar

#Prediction is very difficult, especially about the future. Many of you must have come ac
ross this famous quote by Neils Bohr,
#a Danish physicist. Prediction is the theme of this blog post. In this post, we will cov
er the popular ARIMA forecasting model
#to predict returns on a stock and demonstrate a step-by-step process of ARIMA modeling u
sing R programming.

#What is a forecasting model in Time Series?
#Forecasting involves predicting values for a variable using its historical data points o
r it can also
#involve predicting the change in one variable given the change in the value of another v
ariable.
#Forecasting approaches are primarily categorized into qualitative forecasting and quanti
tative forecasting.
#Time series forecasting falls under the category of quantitative forecasting wherein sta
tistical principals and concepts are
#applied to a given historical data of a variable to forecast the future values of the sa
me variable.
#Some time series forecasting techniques used include:

#Autoregressive Models (AR)
#Moving Average Models (MA)
#Seasonal Regression Models
#Distributed Lags Models
```

```
#What is Autoregressive Integrated Moving Average (ARIMA)?
#ARIMA stands for Autoregressive Integrated Moving Average. ARIMA is also known as Box-Je
nkins approach.
#Box and Jenkins claimed that non-stationary data can be made stationary by differencing
the series,
#Yt. The general model for Yt is written as,

#identically distributed error terms with zero mean. Here, Yt is expressed in terms of it
s past values and the
#current and past values of error terms.

print ("STEP 1.0: Installing the packages...")
```

```
## [1] "STEP 1.0: Installing the packages..."
```

```
#install.packages("ggplot2")
#install.packages("forecast")
#install.packages("plotly")
#install.packages("ggfortify")
#install.packages("tseries")
#install.packages("gridExtra")
#install.packages("docstring")
#install.packages("here")

library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.3
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.3.3
```

```
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 3.3.3
```

```
##
## 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(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 3.3.3
```

```
##
## Attaching package: 'ggfortify'
```

```
## The following object is masked from 'package:forecast':
##
##     gglagplot
```

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 3.3.3
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.3.3
```

```
library(docstring)
```

```
## Warning: package 'docstring' was built under R version 3.3.3
```

```
##
## Attaching package: 'docstring'
```

```
## The following object is masked from 'package:utils':
##
##     ?
```

```
library(here)
```

```
## Warning: package 'here' was built under R version 3.3.3
```

```
## here() starts at C:/CHM136
```

```
# Find the local / Working directory and copy all the project there
print ("STEP 1.1: Find the working directory.")
```

```
## [1] "STEP 1.1: Find the working directory."
```

```
here()
```

```
## [1] "C:/CHM136"
```

```
source(here("src",'main_functions.R'))
# NOTE: For more information on helper functions use ?function_name

print ("STEP 1.2: Loading Data.")
```

```
## [1] "STEP 1.2: Loading Data."
```

```
# LOAD DATA
dataMaster <- read.csv(here("data", "SP500Data.csv"))
attach(dataMaster)
print ("STEP 2.1: Start Analysing.")
```

```
## [1] "STEP 2.1: Start Analysing."
```

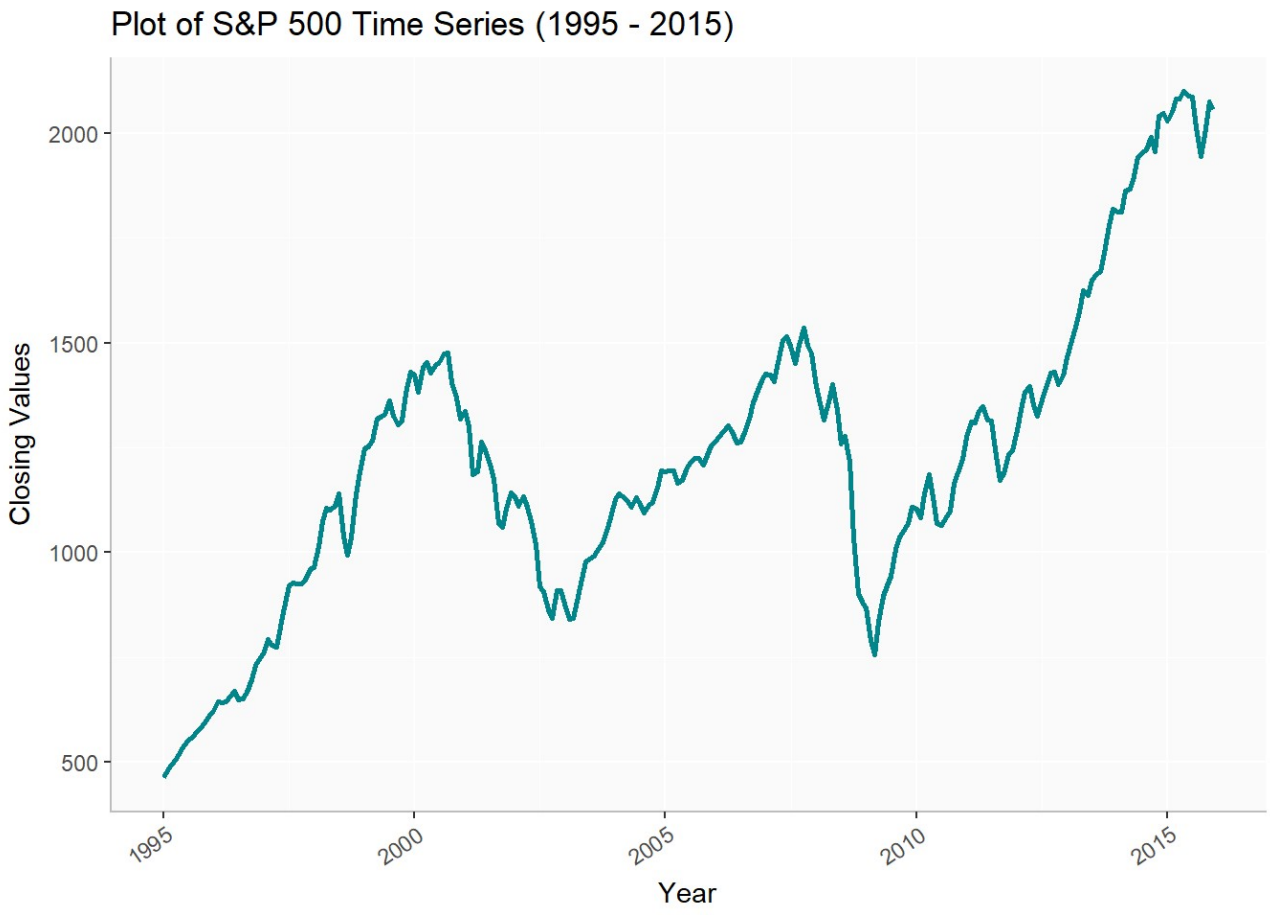
```
# EXPLORATORY ANALYSIS
# I'm going to get valumn from 1995 to present with frq 12
sp_500 <- ts(dataMaster$sp_500, start=c(1995, 1), freq=12)

# TESTS FOR STATIONARITY
Box.test(sp_500, lag = 20, type = 'Ljung-Box')
```

```
##
##  Box-Ljung test
##
## data:  sp_500
## X-squared = 2532.4, df = 20, p-value < 2.2e-16
```

```
adf.test(sp_500)
```

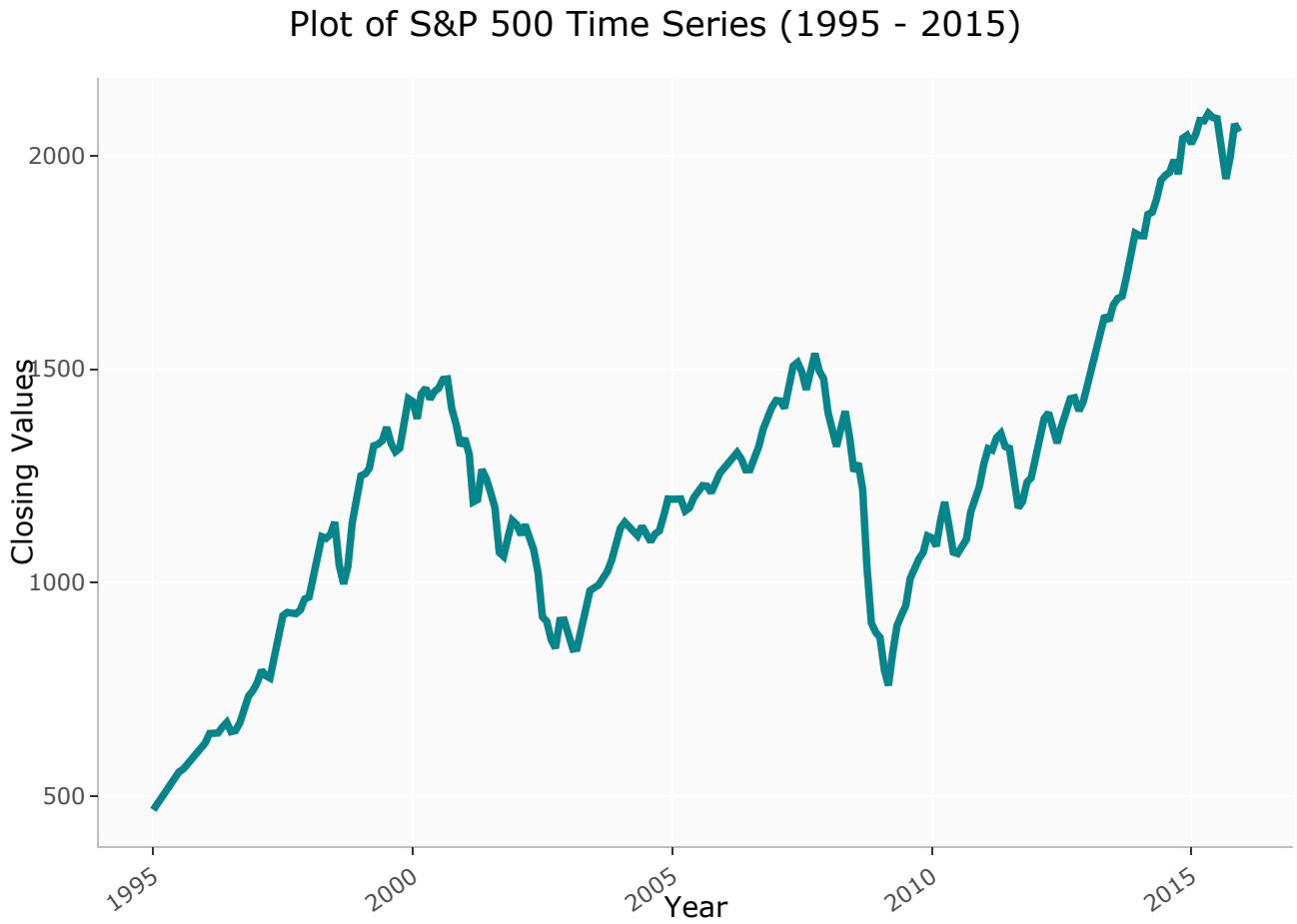
```
##
##  Augmented Dickey-Fuller Test
##
## data:  sp_500
## Dickey-Fuller = -1.816, Lag order = 6, p-value = 0.6533
## alternative hypothesis: stationary
```

```
# p-values are relatively high so we should so visual inspection and
# look at ACF and PACF plots to make appropriate transformation
# for stationarity.



# TIME SERIES PLOT OF S&P
tsSp <- plot_time_series(sp_500, 'S&P 500')

tsSp
```

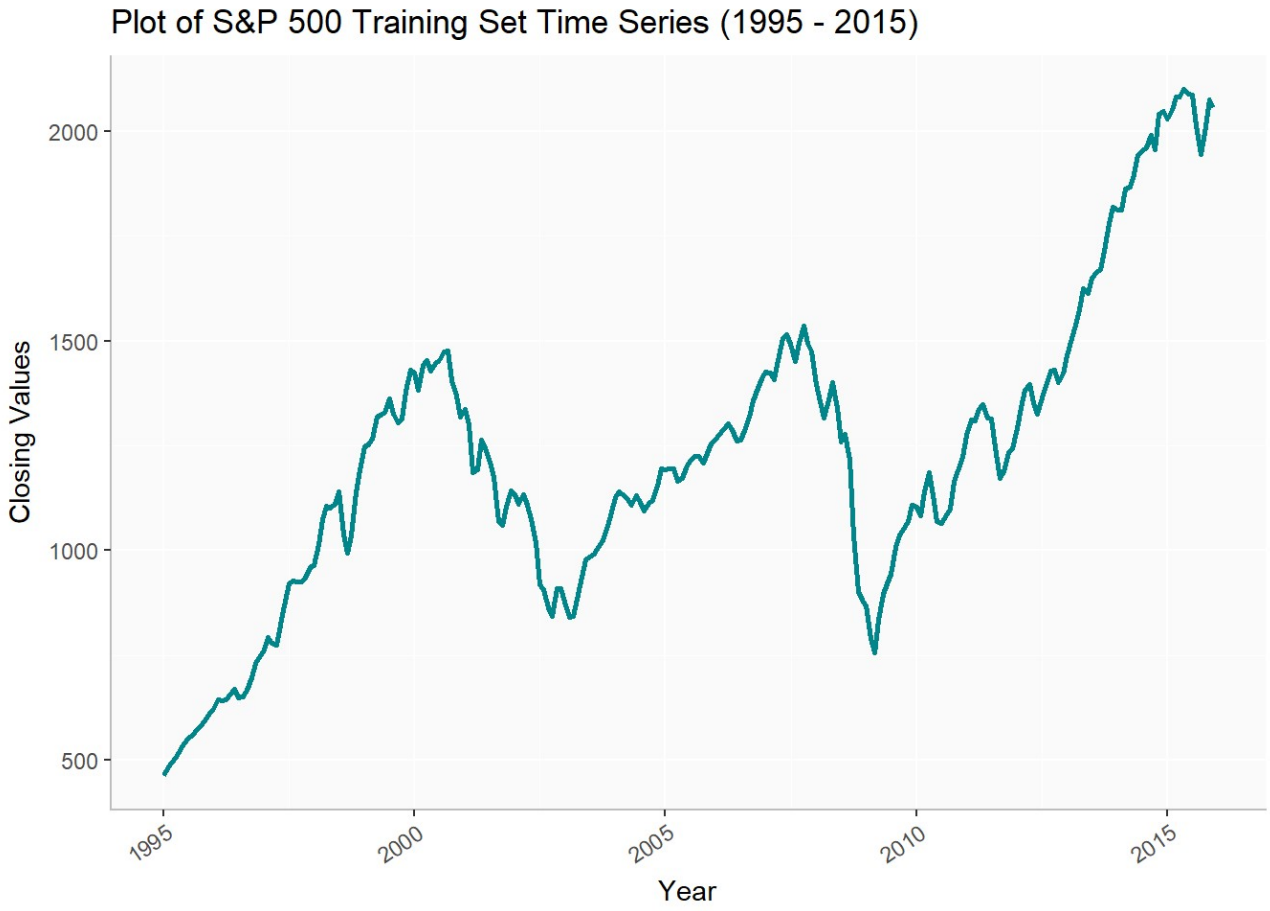



```
ggplotly(tsSp)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```



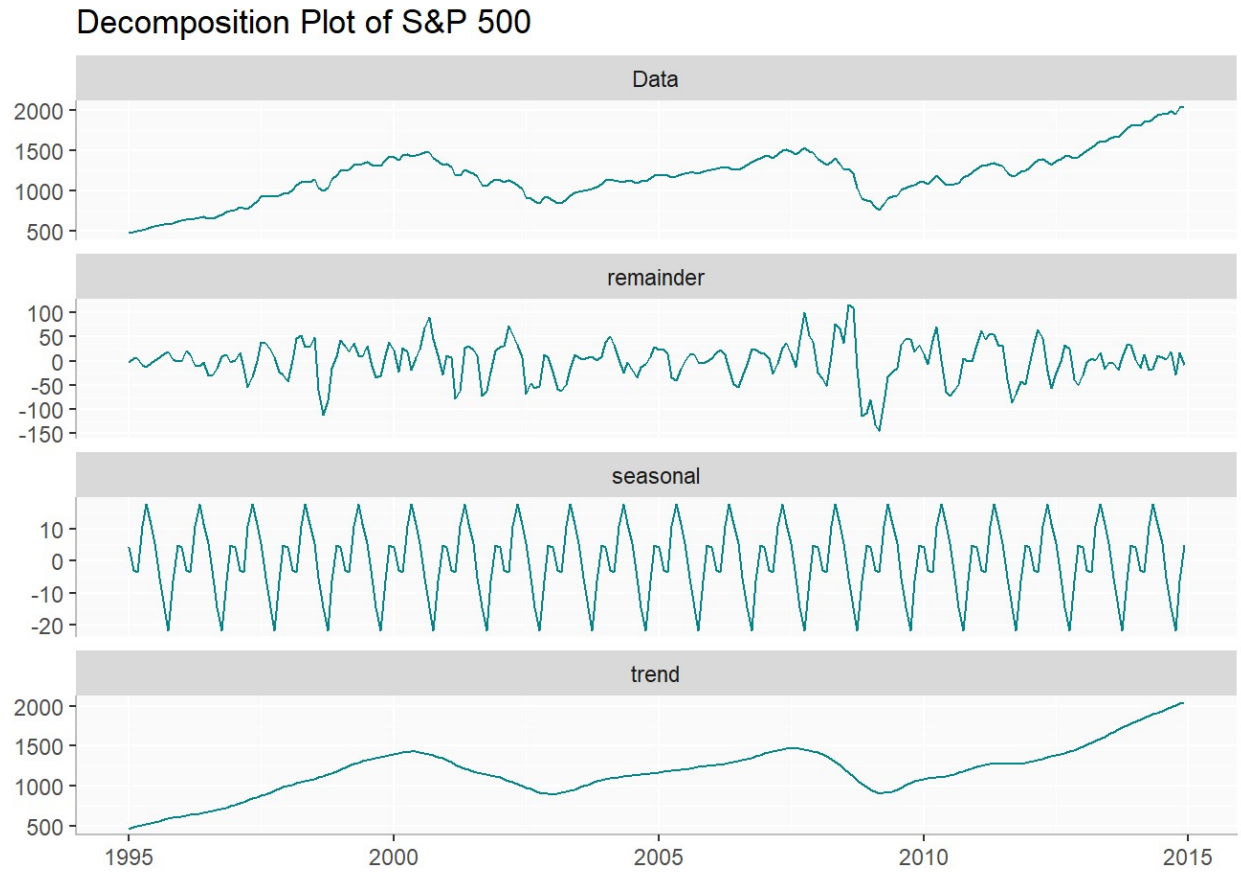

```
# Here we create the training set where we will compare the values for 2015
sp500_TR <- ts(sp_500, start=c(1995, 1), end=c(2014, 12), freq=12)
plot_time_series(sp_500, 'S&P 500 Training Set')
```




```
print ("STEP 2.2: plotting the SP500")
```

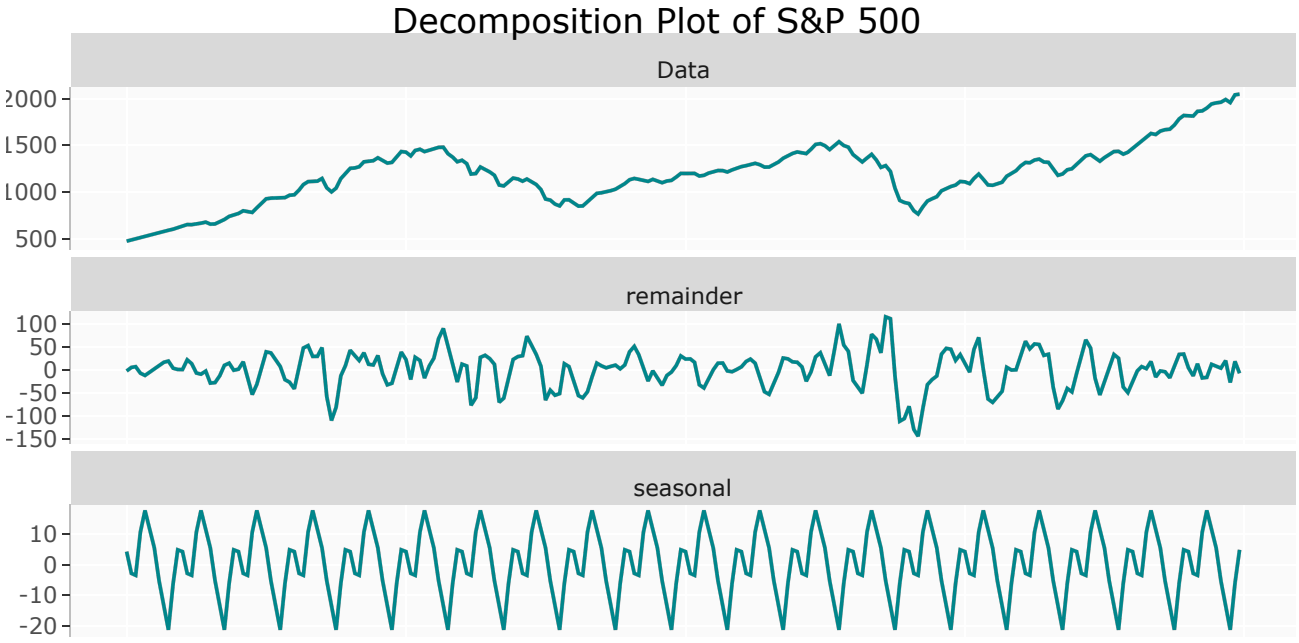
```
## [1] "STEP 2.2: plotting the SP500"
```

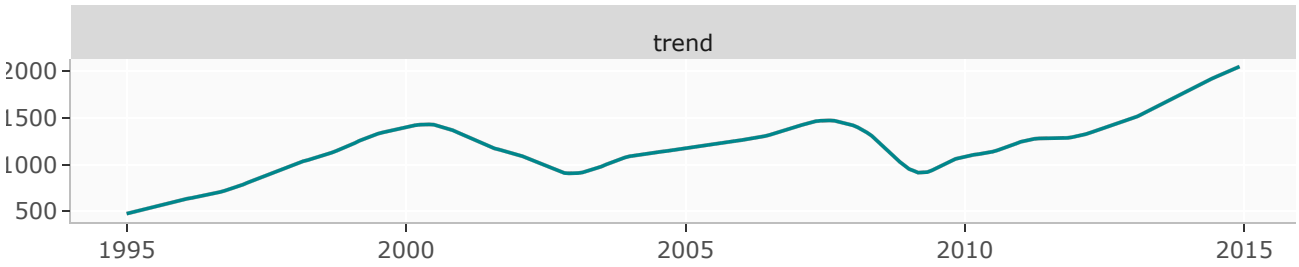
```
# DECOMPOSING TIME SERIES
sp500_stl <- plot_decomp(sp500_TR, 'S&P 500')
sp500_stl
```



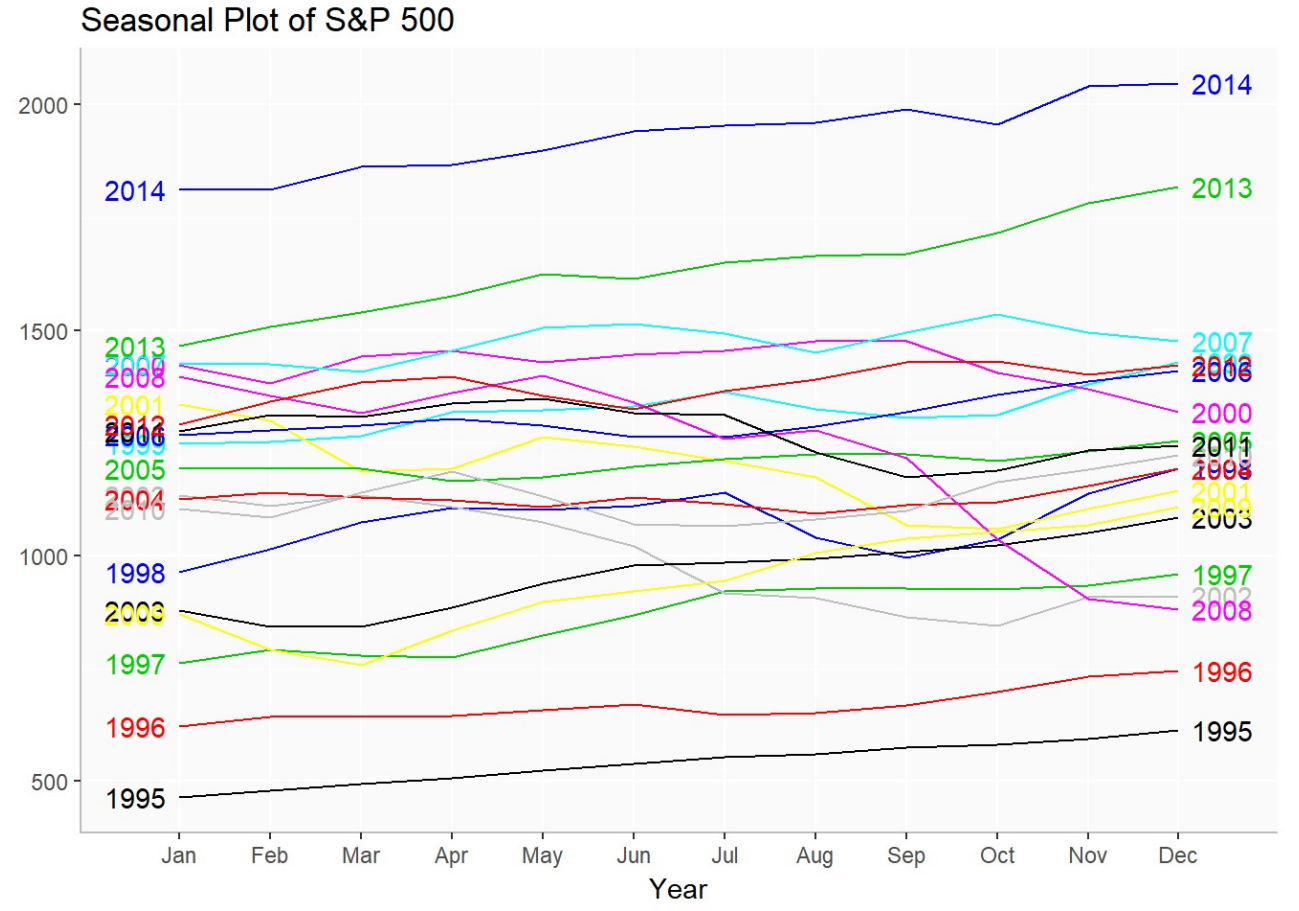

```
ggplotly(sp500_stl)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```



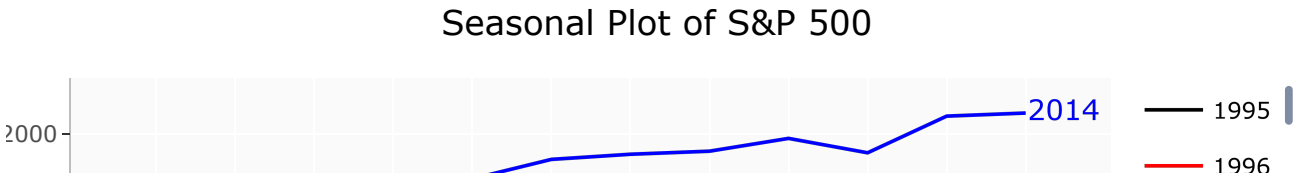


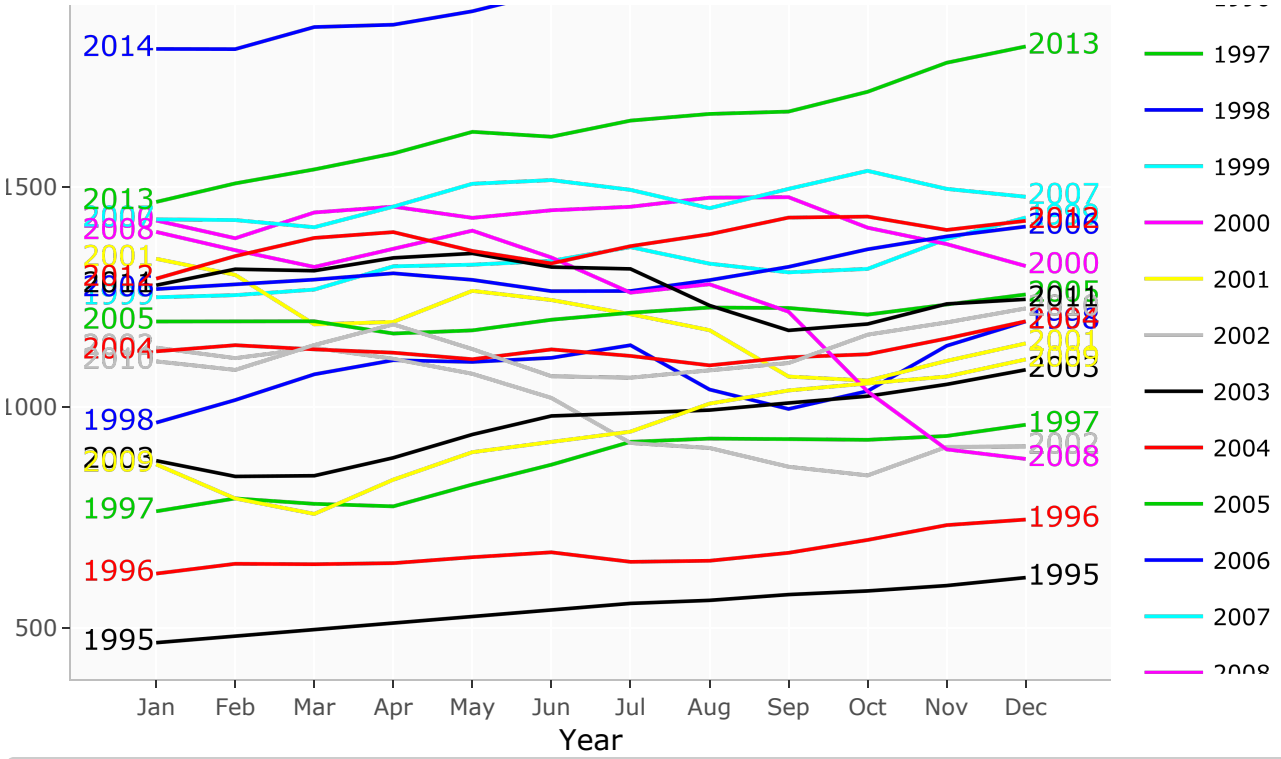

```
# SEASONAL PLOT
seasonal_Plot <- plot_seasonal(sp500_TR, 'S&P 500')

seasonal_Plot
```




```
ggplotly(seasonal_Plot)
```

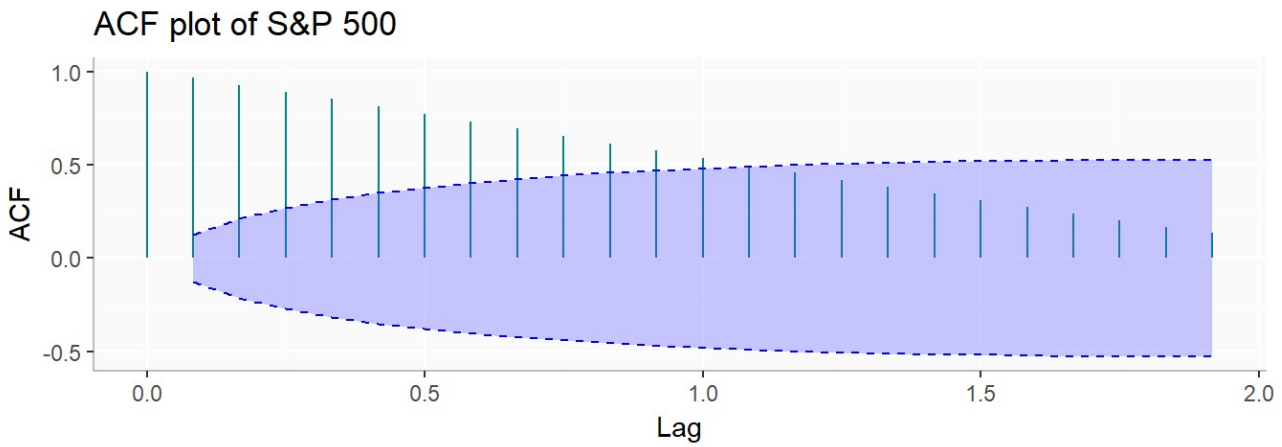
```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

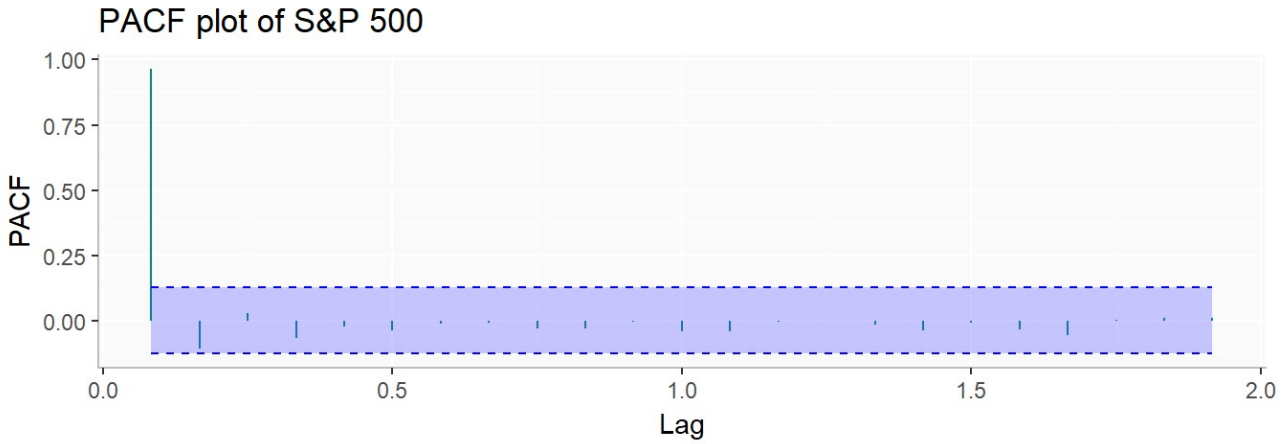




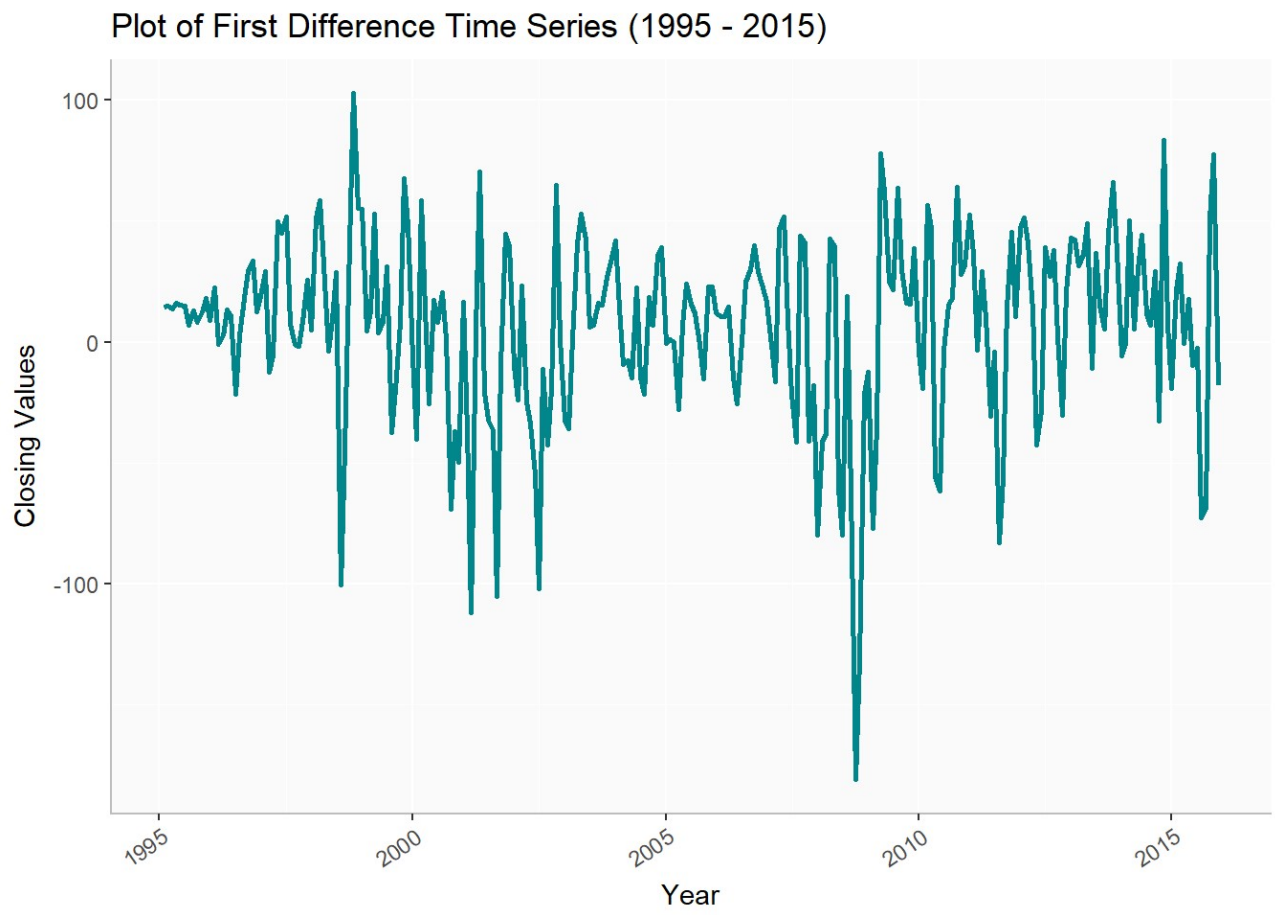

```
# DIAGNOSING ACF AND PACF PLOTS
plot_acf_pacf(sp500_TR, 'S&P 500')
```

```
## Warning: package 'bindrcpp' was built under R version 3.3.3
```



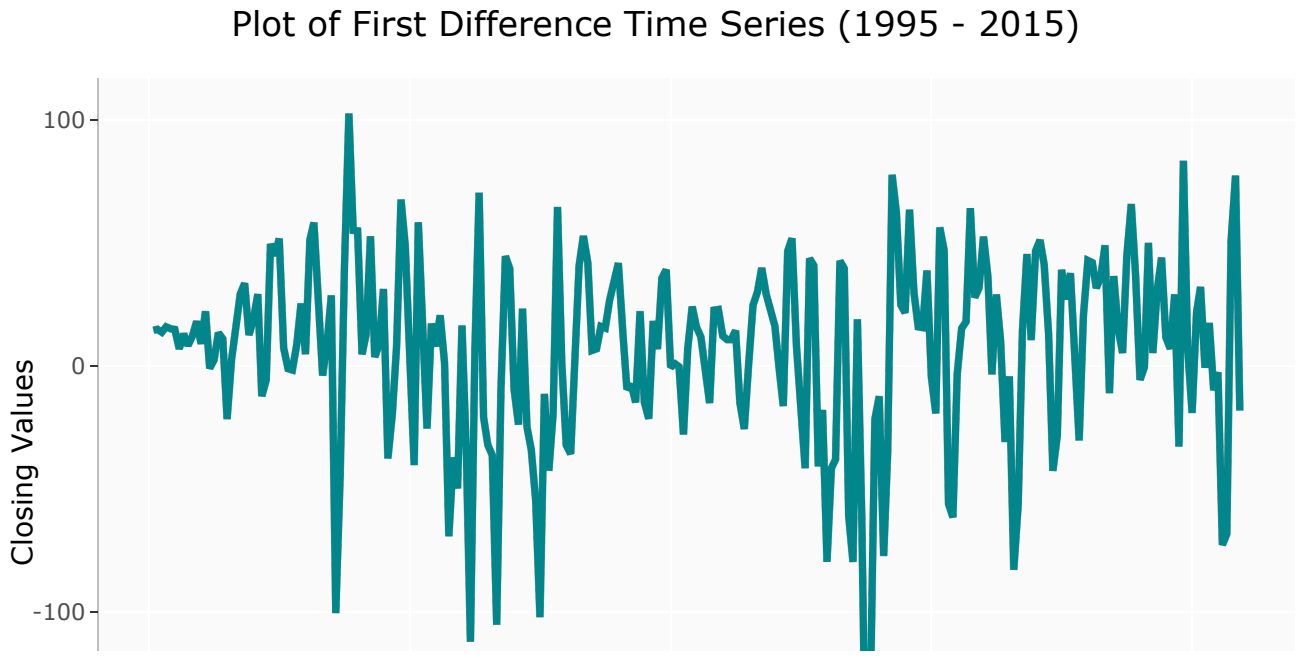


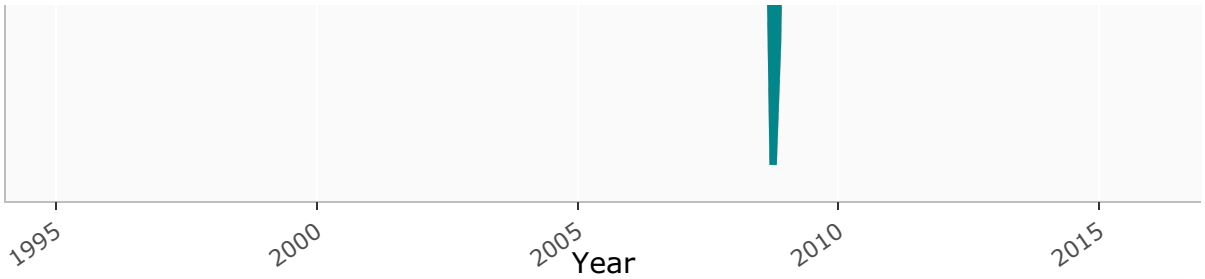

```
# TRANSFORMING OUR DATA TO ADJUST FOR NON STATIONARY
diff <- diff(sp_500)

tsDiff <- plot_time_series(diff, 'First Difference')
tsDiff
```




```
ggplotly(tsDiff)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```






```
# TESTS FOR STATIONARITY FOR DIFFERENCED TIME SERIES OBJECT
Box.test(diff, lag = 20, type = 'Ljung-Box')
```

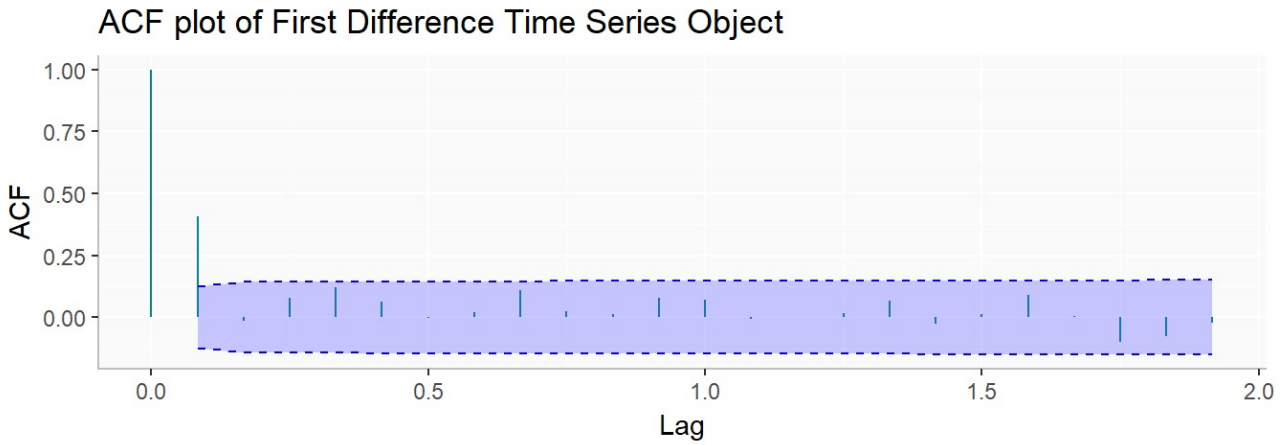
```
##
##  Box-Ljung test
##
## data:  diff
## X-squared = 58.2, df = 20, p-value = 1.347e-05
```

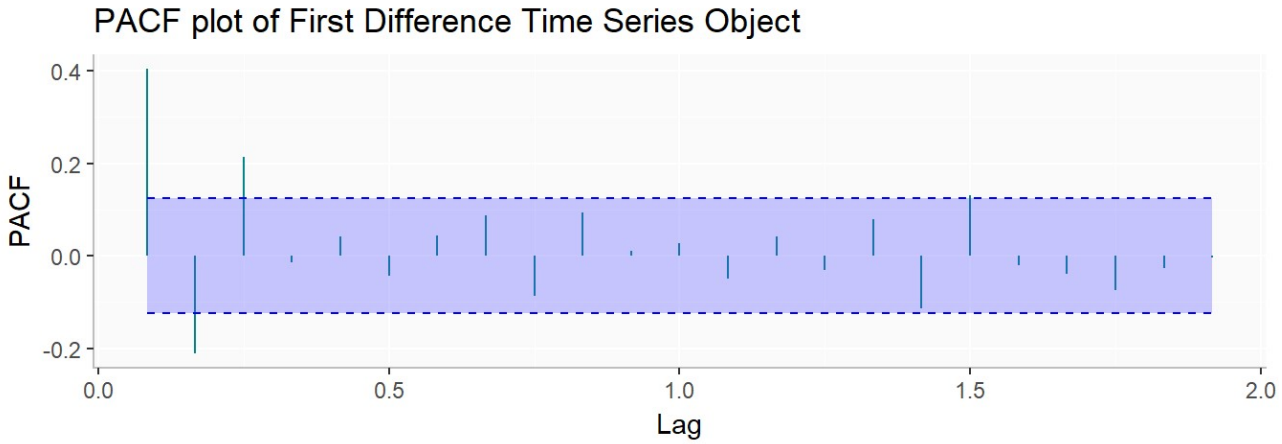
```
adf.test(diff)
```

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

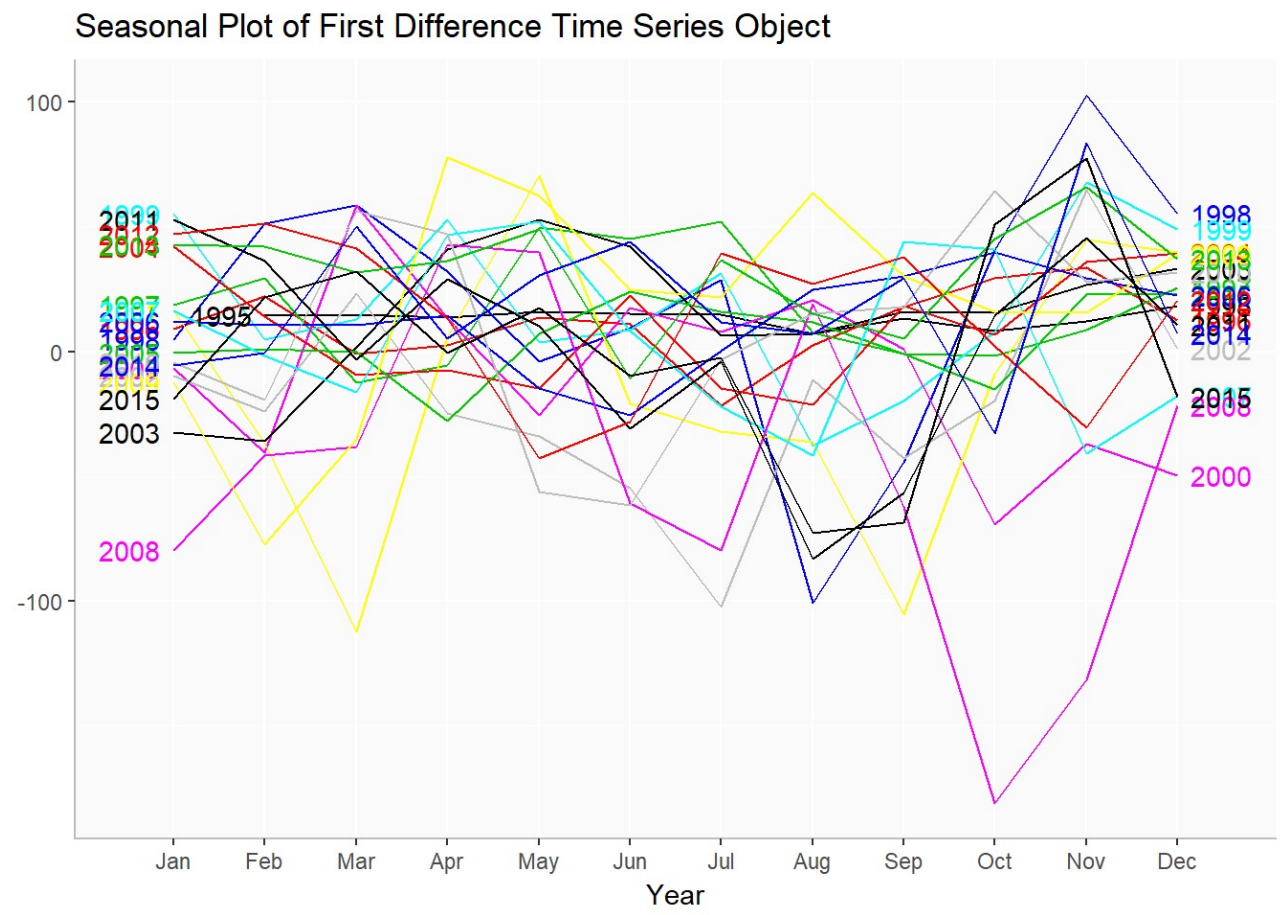
```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff
## Dickey-Fuller = -4.9552, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```
# p-values seems small enough to infer stationarity for the first difference
# Let's begin analysis with visually inspecting ACF and PACF plots

# DIAGNOSING ACF AND PACF PLOTS FOR DIFFERENCED TIME SERIES OBJECT
plot_acf_pacf(diff, 'First Difference Time Series Object')
```



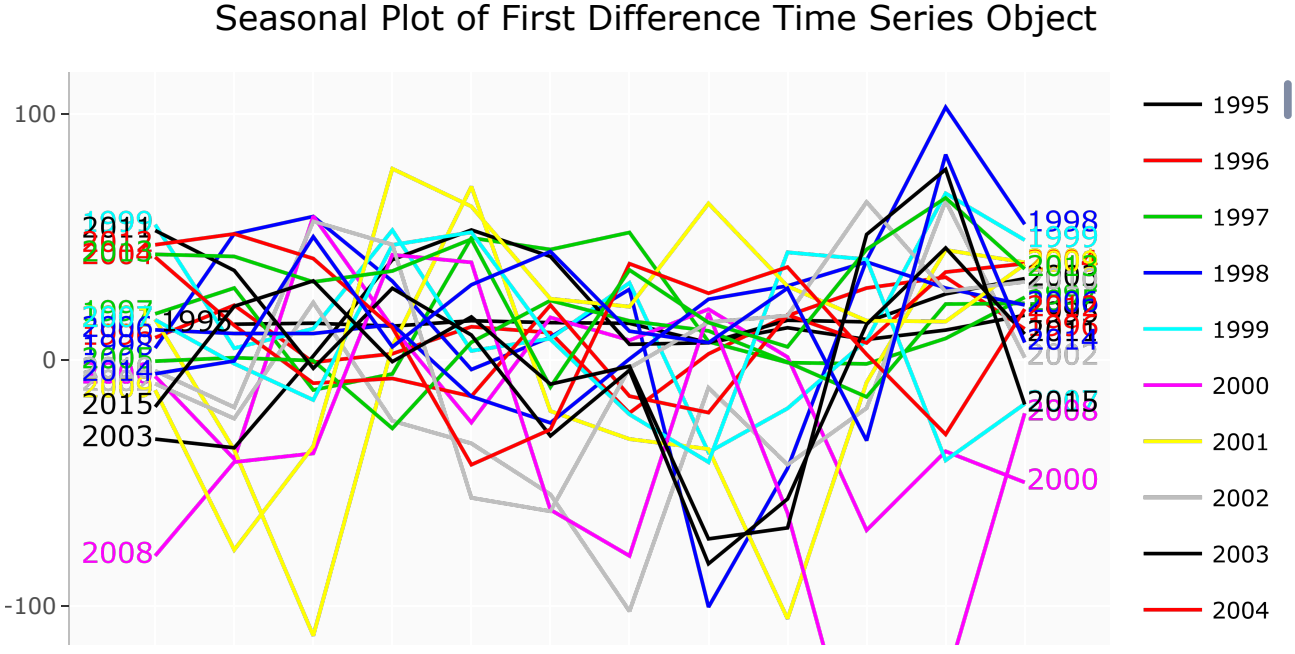


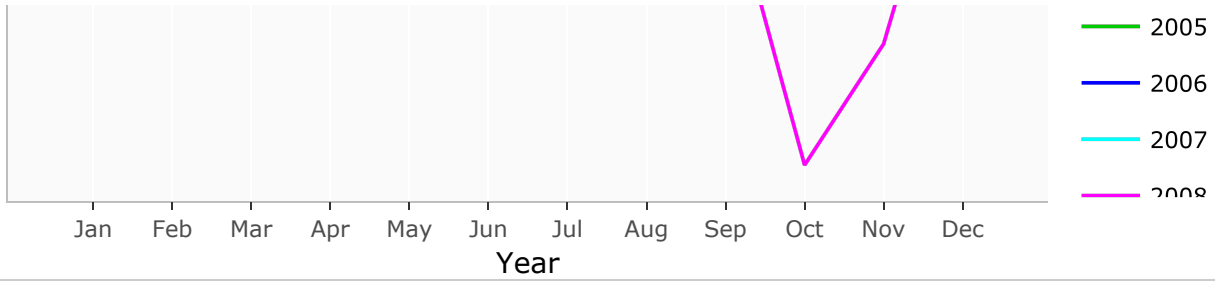

```
# SEASONAL PLOT FOR DIFFERENCED TIME SERIES OBJECT
spDiff <- plot_seasonal(diff, 'First Difference Time Series Object')
spDiff
```




```
ggplotly(spDiff)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```





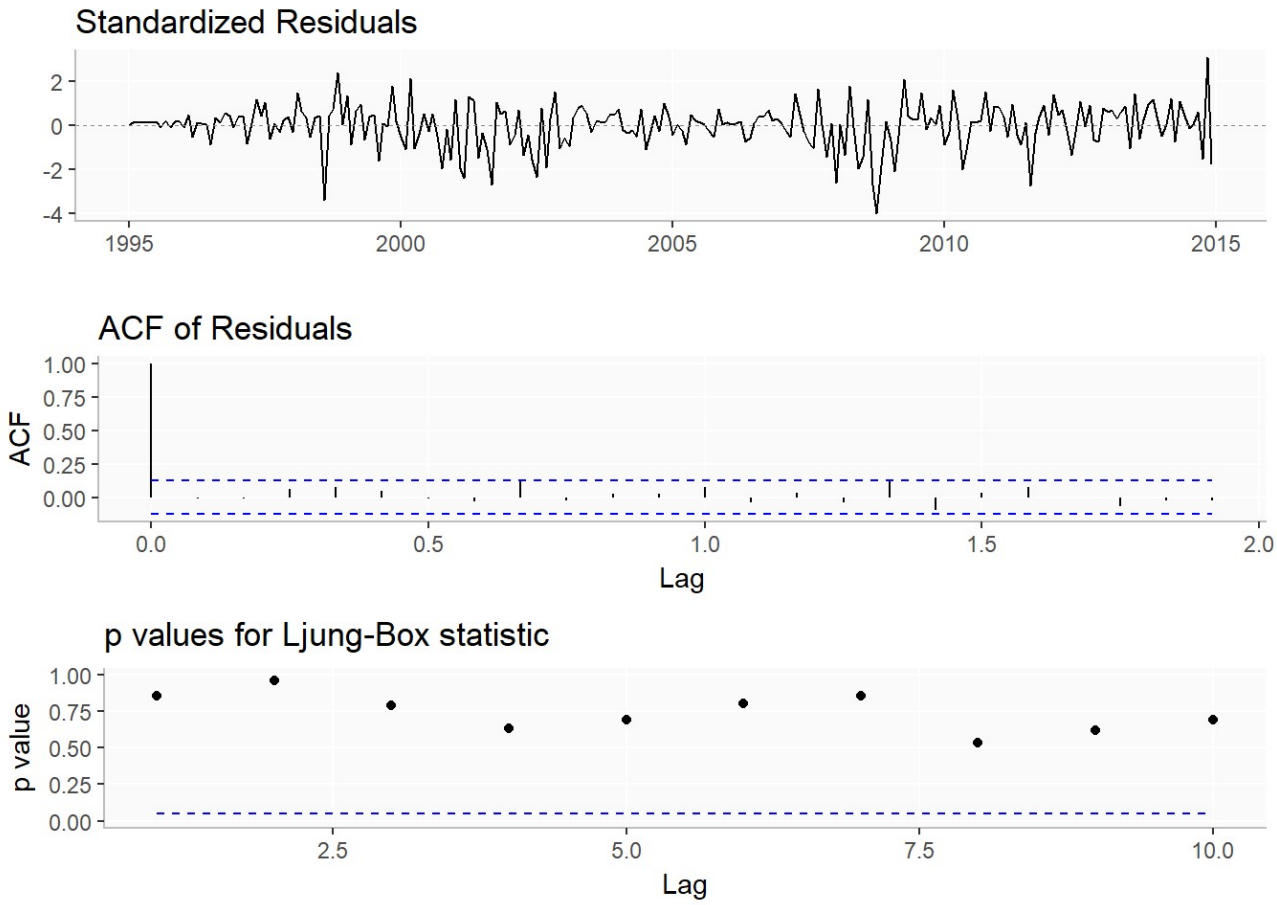

```
# AUTO.ARIMA ESTIMATION
auto.arima(sp500_TR)
```

```
## Series: sp500_TR
## ARIMA(0,1,1) with drift
##
## Coefficients:
##          ma1   drift
##       0.5666  6.4975
## s.e.  0.0551  3.4326
##
## sigma^2 estimated as 1161:  log likelihood=-1181.58
## AIC=2369.17   AICc=2369.27   BIC=2379.59
```

```
# From our visual inspection and auto.arima model we will choose an
# ARIMA(0, 1, 1) with drift

# BUILD MODEL
fit <- Arima(sp500_TR, order = c(0,1,1), include.drift = TRUE)
summary(fit)
```

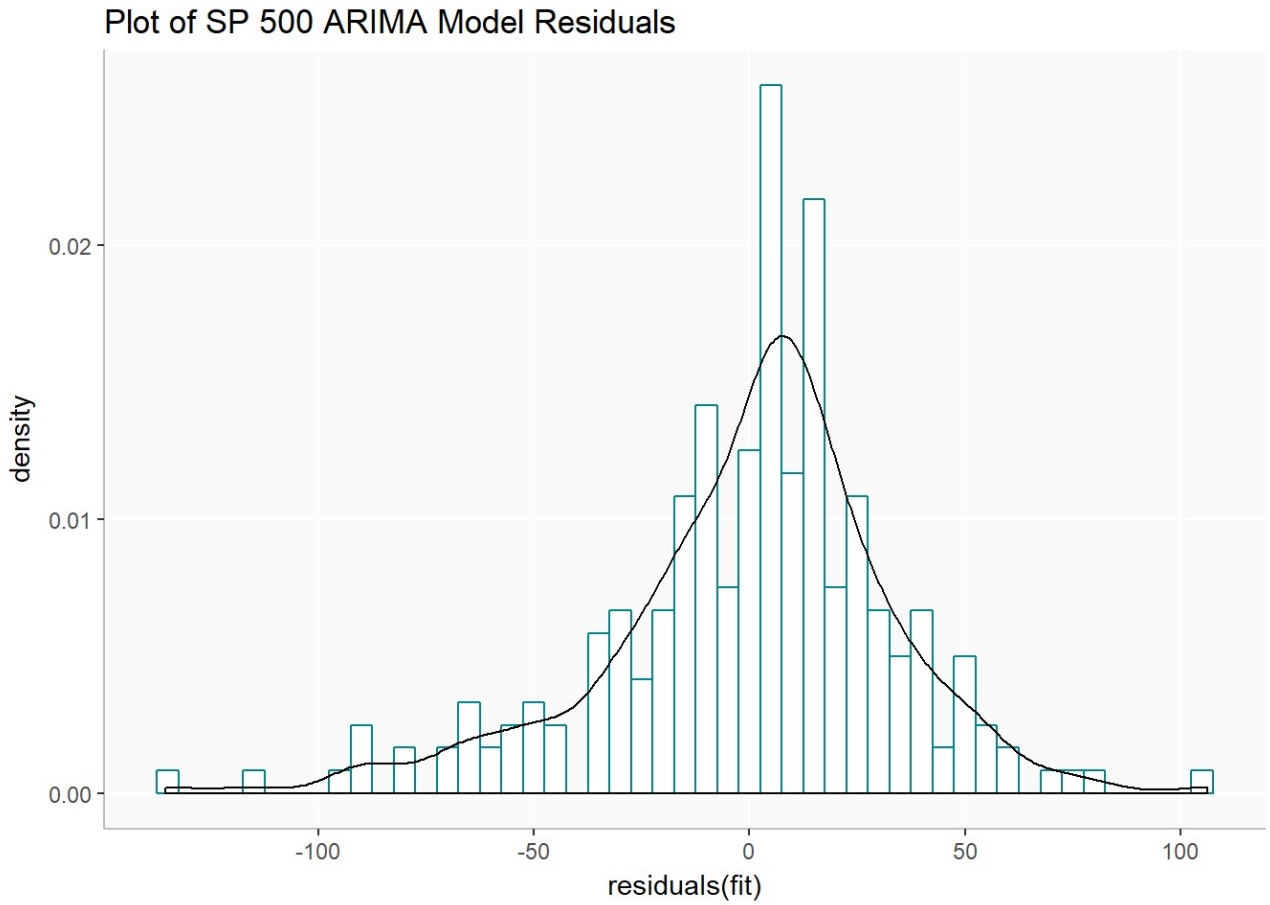
```
## Series: sp500_TR
## ARIMA(0,1,1) with drift
##
## Coefficients:
##          ma1   drift
##       0.5666  6.4975
## s.e.  0.0551  3.4326
##
## sigma^2 estimated as 1161:  log likelihood=-1181.58
## AIC=2369.17   AICc=2369.27   BIC=2379.59
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.00911296 33.85289 24.84955 -0.00840343 2.141218 0.1310854
##                      ACF1
## Training set -0.01137429
```

```
# RESIDUAL DIAGNOSTICS
ggtsdiag(fit) +
  theme(panel.background = element_rect(fill = "gray98"),
        panel.grid.minor = element_blank(),
        axis.line.y = element_line(colour="gray"),
        axis.line.x = element_line(colour="gray"))
```




```
residFit <- ggplot(data=fit, aes(residuals(fit))) +
  geom_histogram(aes(y =..density..),
                 binwidth = 5,
                 col="turquoise4", fill="white") +
  geom_density(col=1) +
  theme(panel.background = element_rect(fill = "gray98"),
        panel.grid.minor = element_blank(),
        axis.line   = element_line(colour="gray"),
        axis.line.x = element_line(colour="gray")) +
  ggtitle("Plot of SP 500 ARIMA Model Residuals")

residFit
```

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



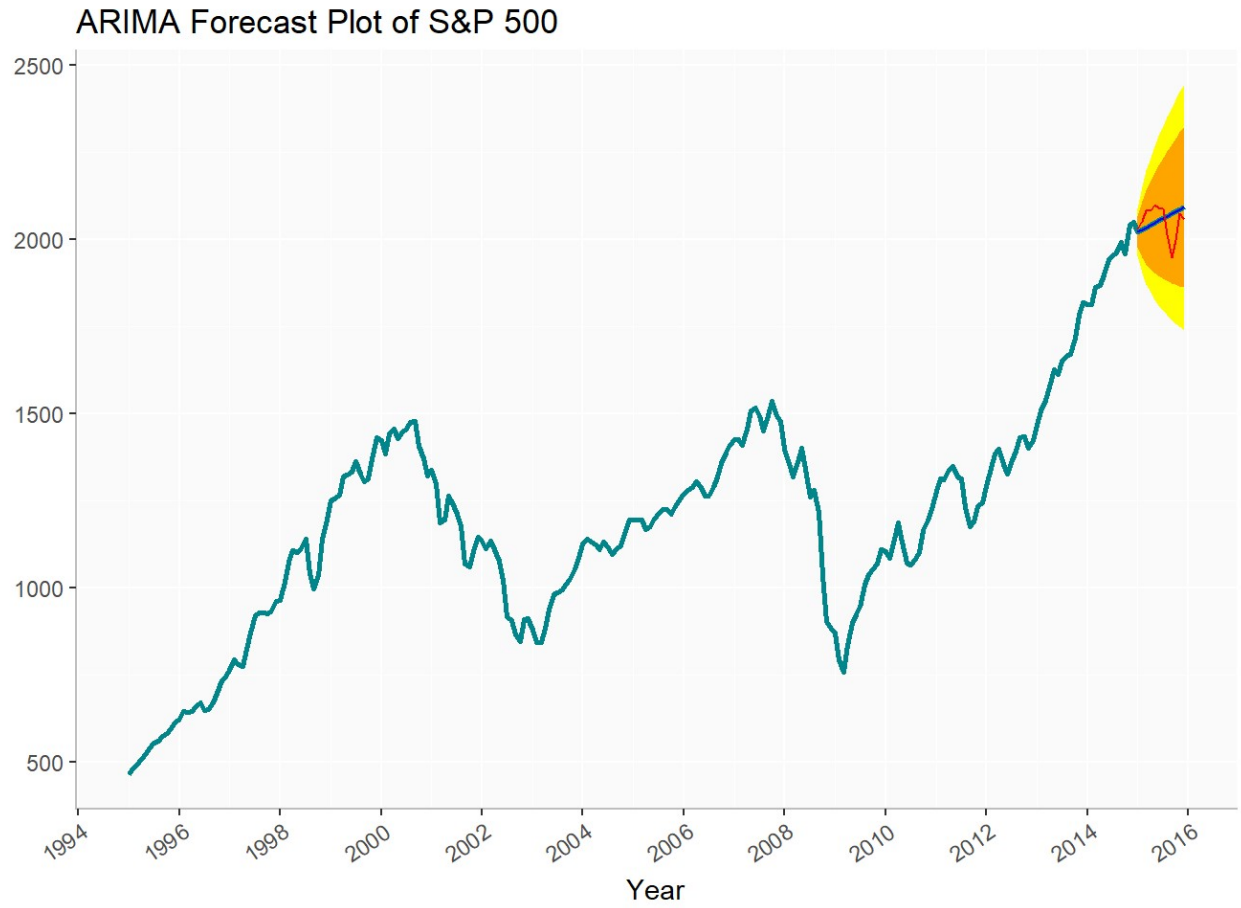

```
# TEST SET THAT WE WILL COMPARE OUR FORECAST AGAINST
dataMaster_TS <- dataMaster[-c(1:240), ]
act_sp500_2015_ts <- ts(dataMaster_TS$sp_500, start = c(2015, 1), freq = 12)
act_sp500_2015_ts
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2015 2028.592 2050.415 2082.582 2081.860 2099.355 2089.485 2086.920
##           Aug      Sep      Oct      Nov      Dec
## 2015 2014.085 1945.723 1996.758 2074.260 2056.100
```

```
print ("STEP 2.2: Forcasting.")
```

```
## [1] "STEP 2.2: Forcasting."
```

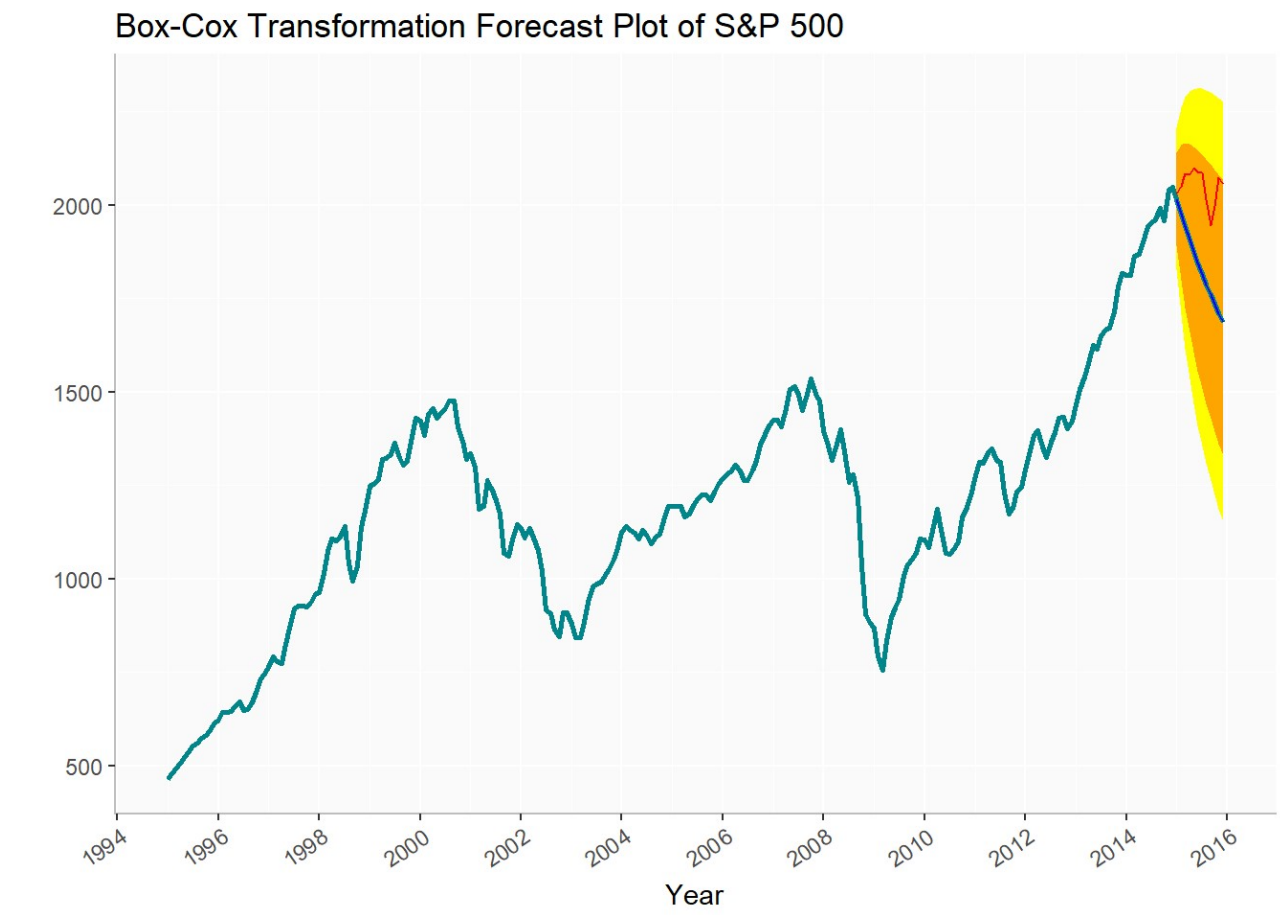
```
# FORECASTING
# METHOD CHOSEN THROUGH BOX JENKINS METHODOLOGY WAS ARIMA(0,1,1) WITH DRIFT
## ARIMA MODEL CHOSEN
fit_arima <- forecast(fit, h = 12)
forSp500 <- autoplot(fit_arima,
                     holdout = act_sp500_2015_ts,
                     ts_object_name = 'ARIMA')

forSp500
```




```
ggplotly(forSp500)
```

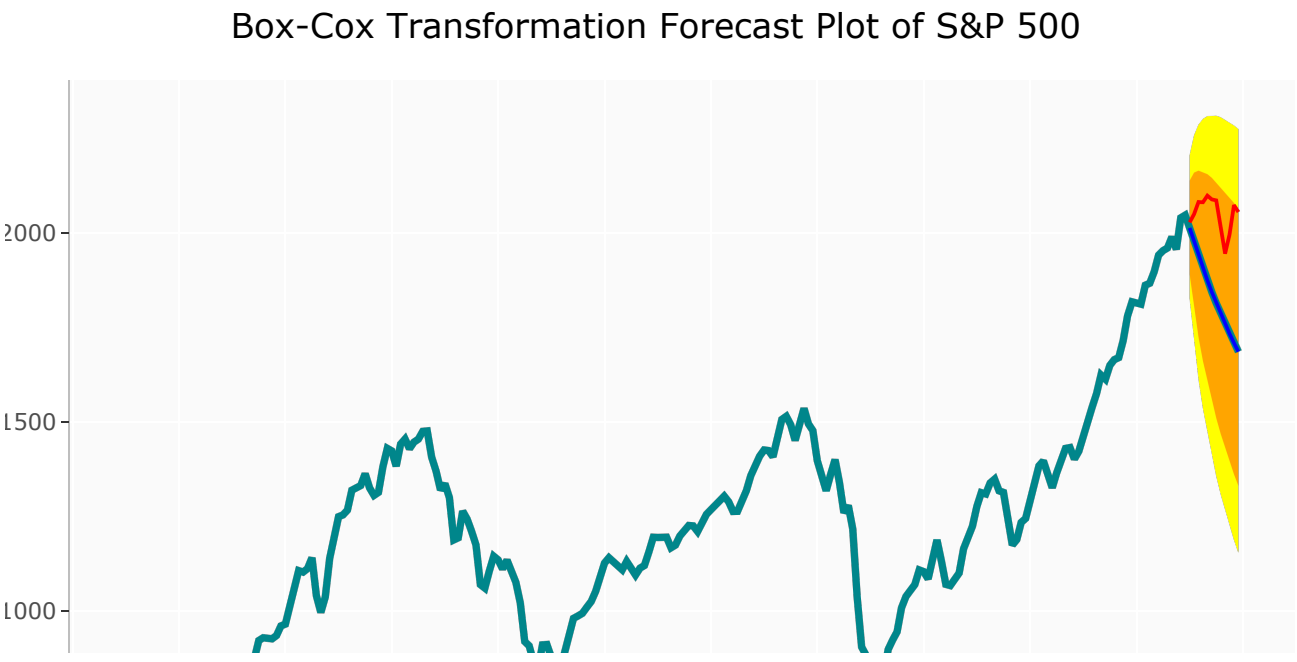
```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

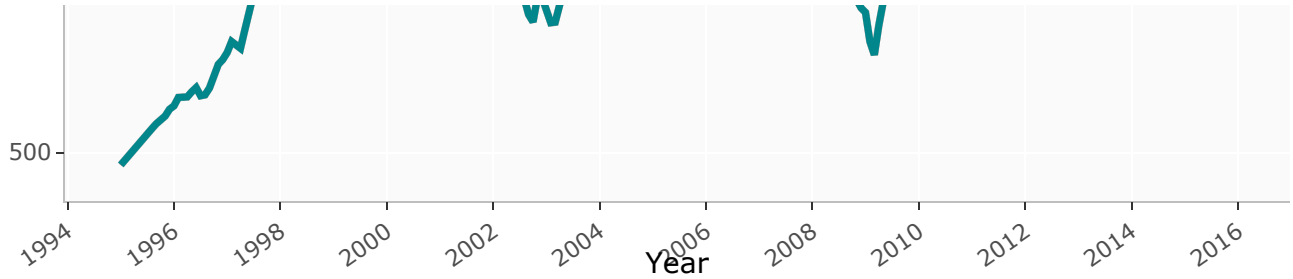

```
## BOX COX TRANSFORMATION
lambda <- BoxCox.lambda(sp500_TR)
fit_sp500_BC <- ar(BoxCox(sp500_TR,lambda))
fit_BC <- forecast(fit_sp500_BC,h=12,lambda=lambda)

s <- autoplot(fit_BC,
              holdout = act_sp500_2015_ts,
              ts_object_name = 'Box-Cox Transformation')
s
```



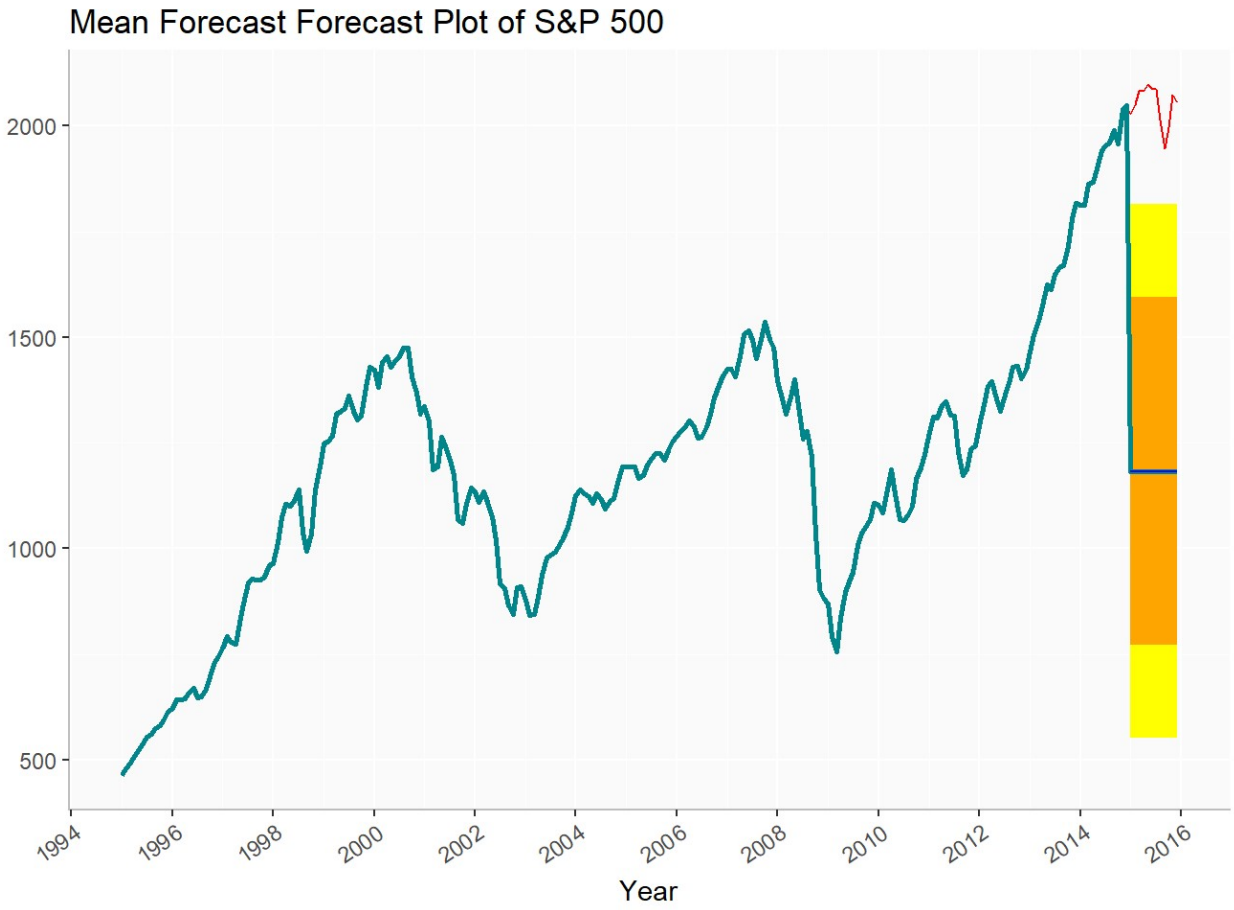

```
ggplotly(s)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```






```
# MEAN FORECAST METHOD
fit_meanf <- forecast(meanf(sp500_TR, h = 12))
e <- autoplot(fit_meanf,
              holdout = act_sp500_2015_ts,
              ts_object_name = 'Mean Forecast')
e
```

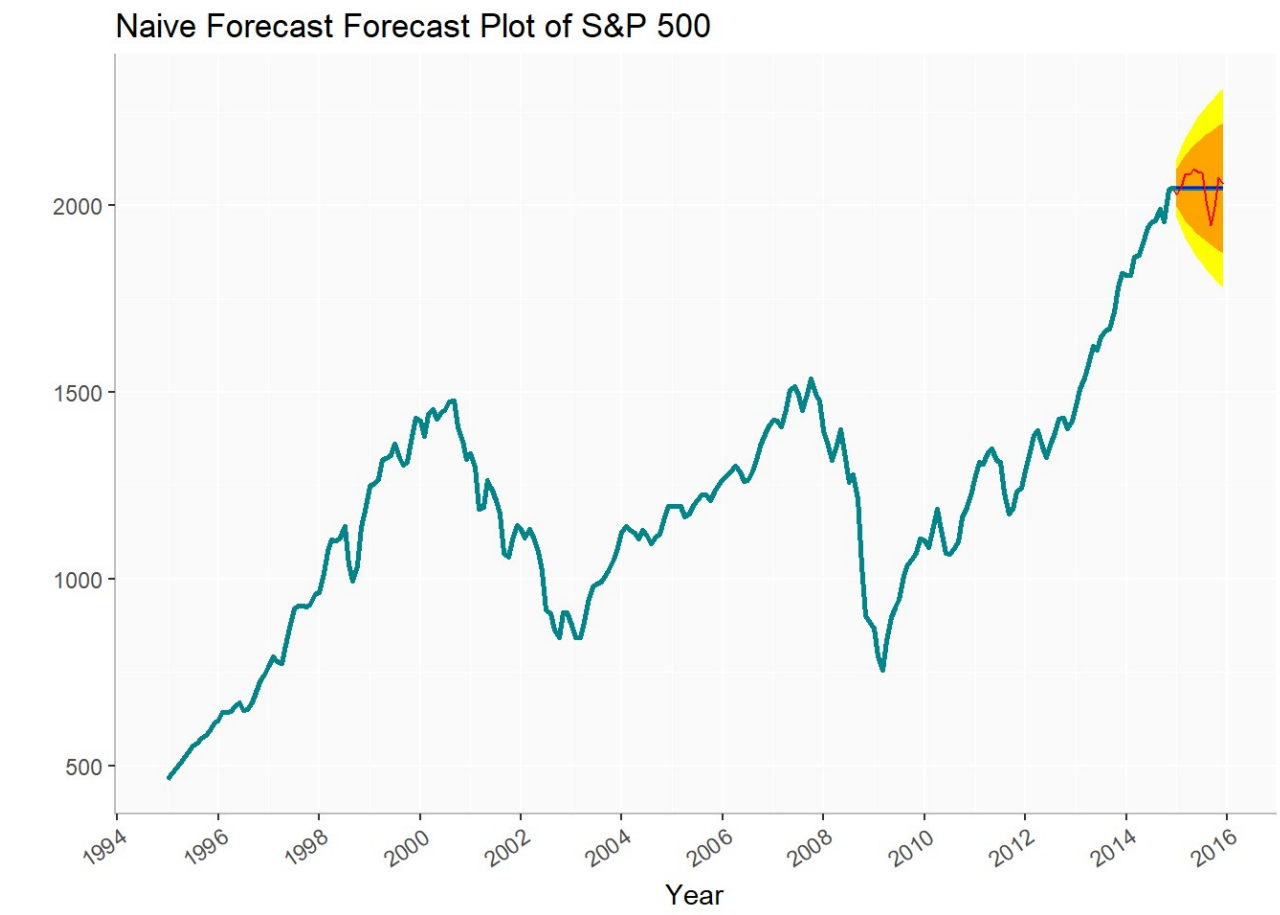



```
ggplotly(e)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

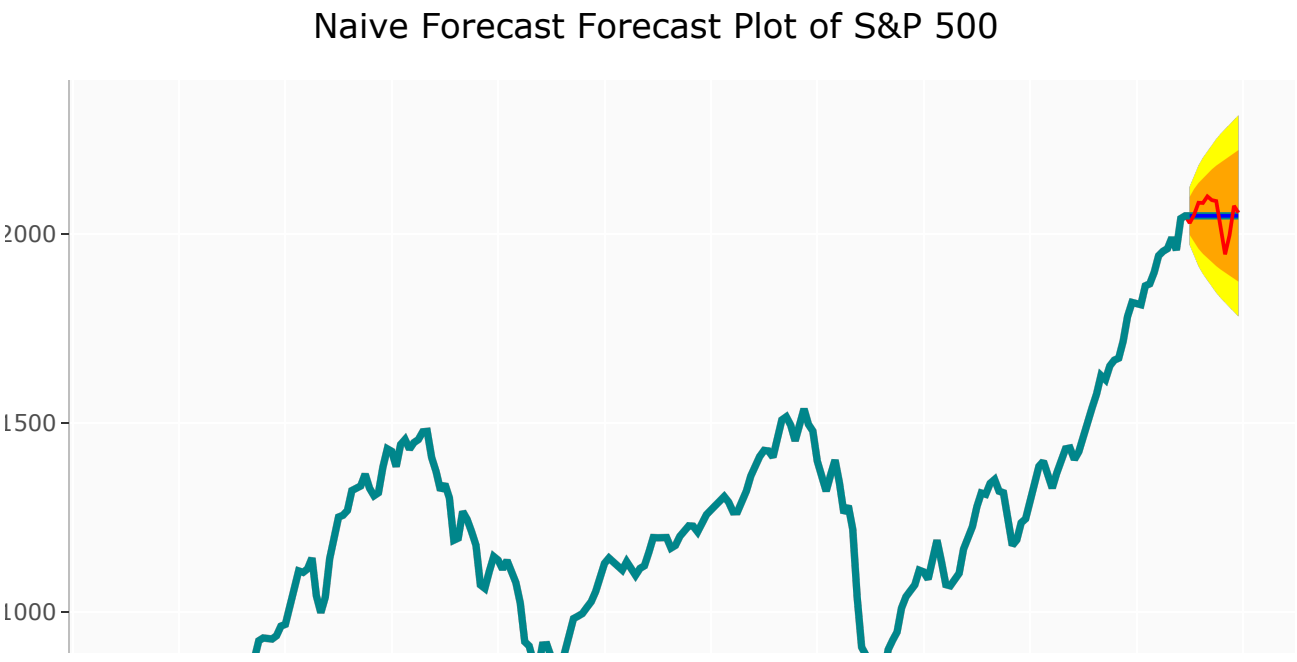
Mean Forecast Forecast Plot of S&P 500

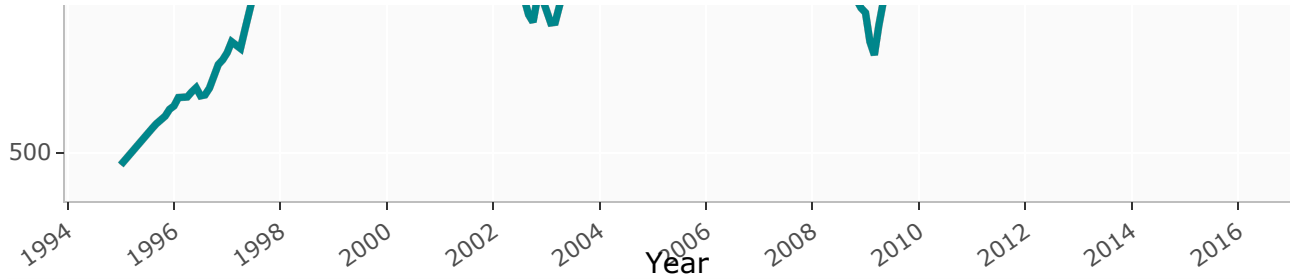

```
# NAIVE METHOD
fit_naive <- forecast(naive(sp500_TR, h = 12))
f <- autoplot(fit_naive,
              holdout = act_sp500_2015_ts,
              ts_object_name = "Naive Forecast")
f
```



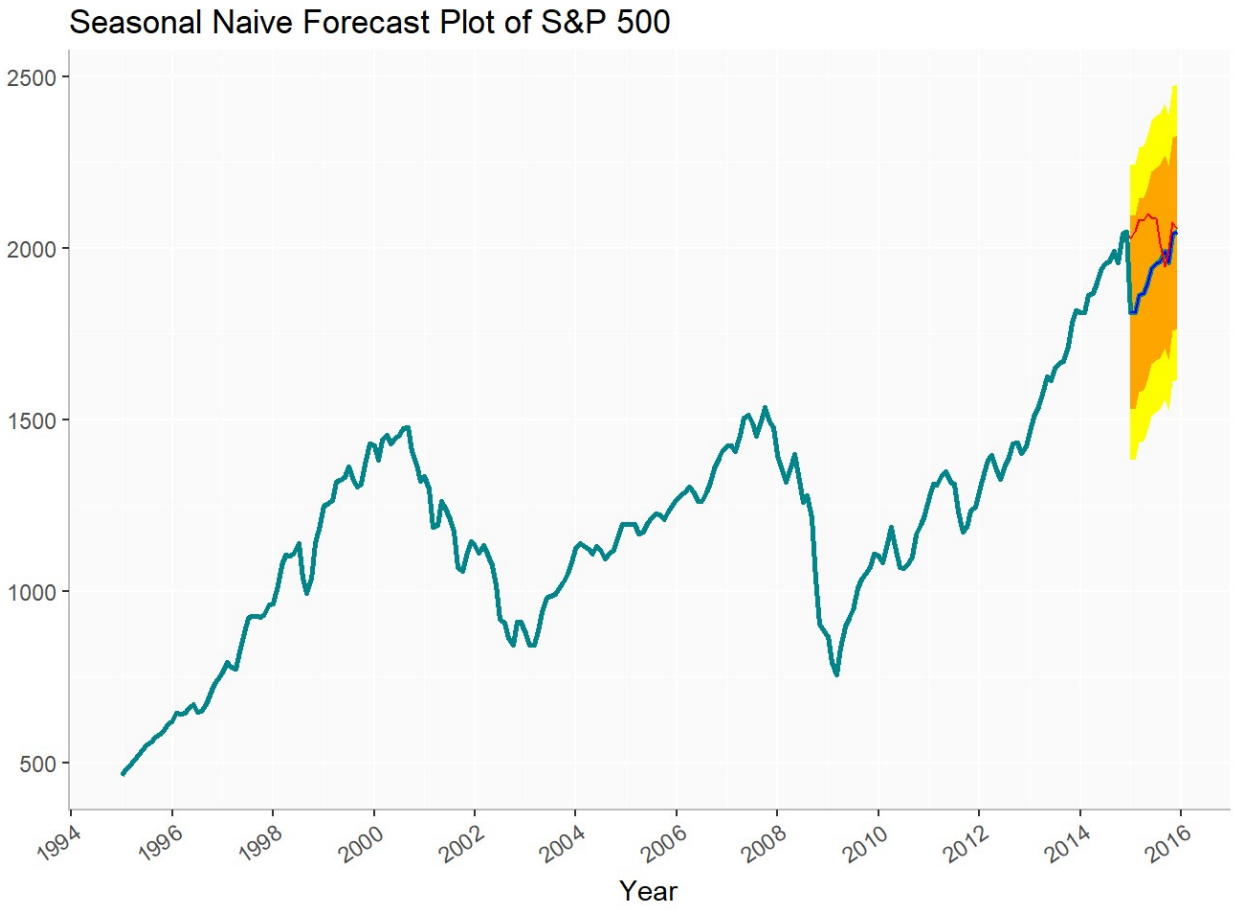

```
ggplotly(f)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```





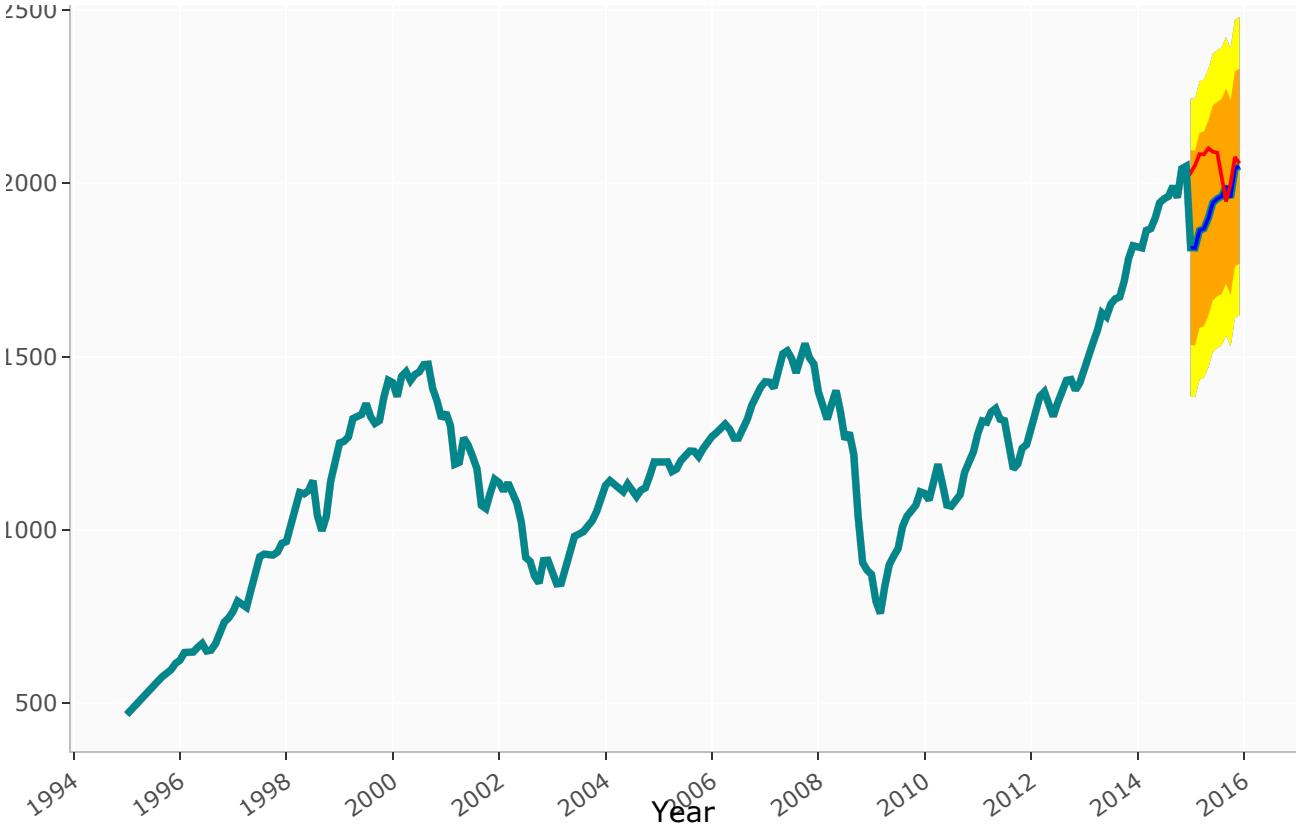

```
# SEASONAL NAIVE METHOD
fit_snaive <- forecast(snaive(sp500_TR, h = 12))
g <- autoplot(fit_snaive,
              holdout = act_sp500_2015_ts,
              ts_object_name = "Seasonal Naive")
g
```



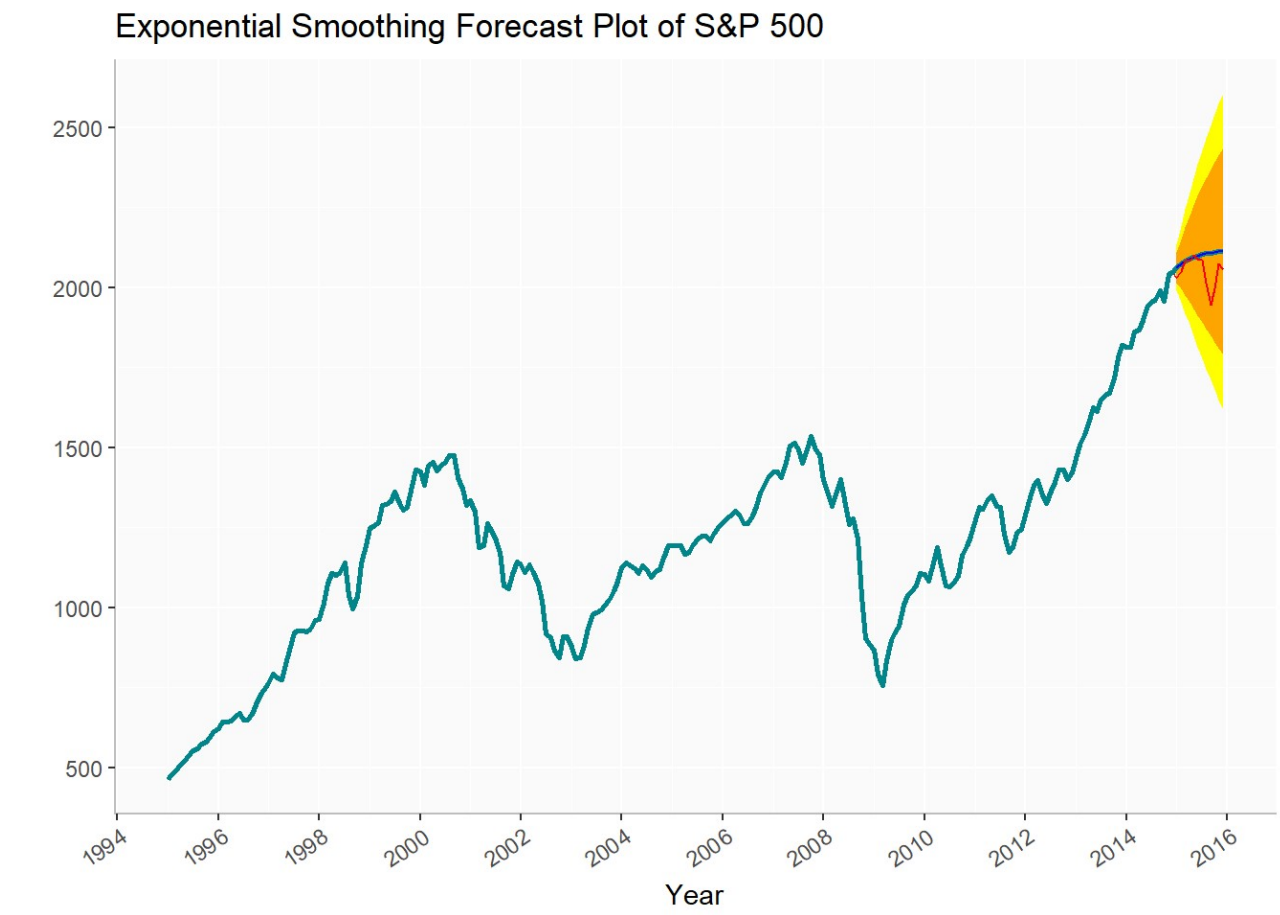

```
ggplotly(g)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

Seasonal Naive Forecast Plot of S&P 500

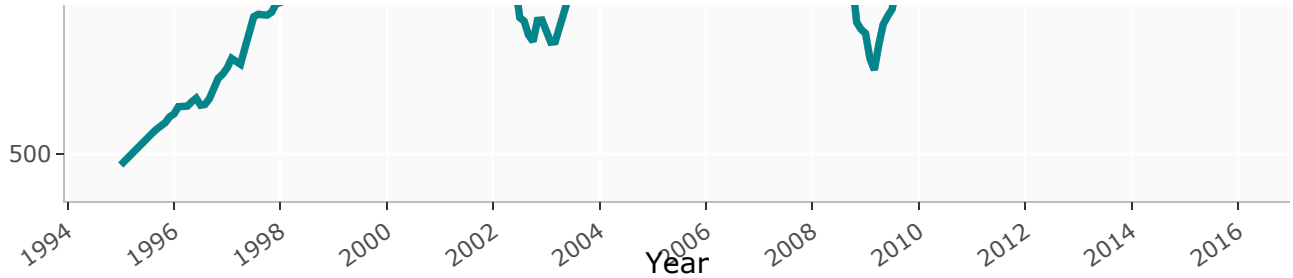



```
# EXPONENTIAL SMOOTHING METHOD
fit_ets <- forecast(ets(sp500_TR), h = 12)
h <- autoplot(fit_ets,
              holdout=act_sp500_2015_ts,
              ts_object_name = "Exponential Smoothing")

h
```




```
ggplotly(h)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```




```
# COMPARE FORECAST ACCURACIES ACROSS DIFFERENT METHODS USED
accuracy(fit_arima)
```

```
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.00911296 33.85289 24.84955 -0.00840343 2.141218 0.1310854
##                      ACF1
## Training set -0.01137429
```

```
accuracy(fit_BC)
```

```
##                    ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 6.808873 39.28348 30.16598 0.282006 2.567669 0.1591304
##                    ACF1
## Training set 0.4091459
```

```
accuracy(fit_meanf)
```

```
##                        ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -9.668655e-15 319.3598 244.9373 -9.110565 24.74398 1.292084
##                    ACF1
## Training set 0.9666459
```

```
accuracy(fit_naive)
```

```
##                    ME     RMSE      MAE       MPE    MAPE      MASE
## Training set 6.624059 39.30052 30.14866 0.5581768 2.66548 0.1590391
##                    ACF1
## Training set 0.4170651
```

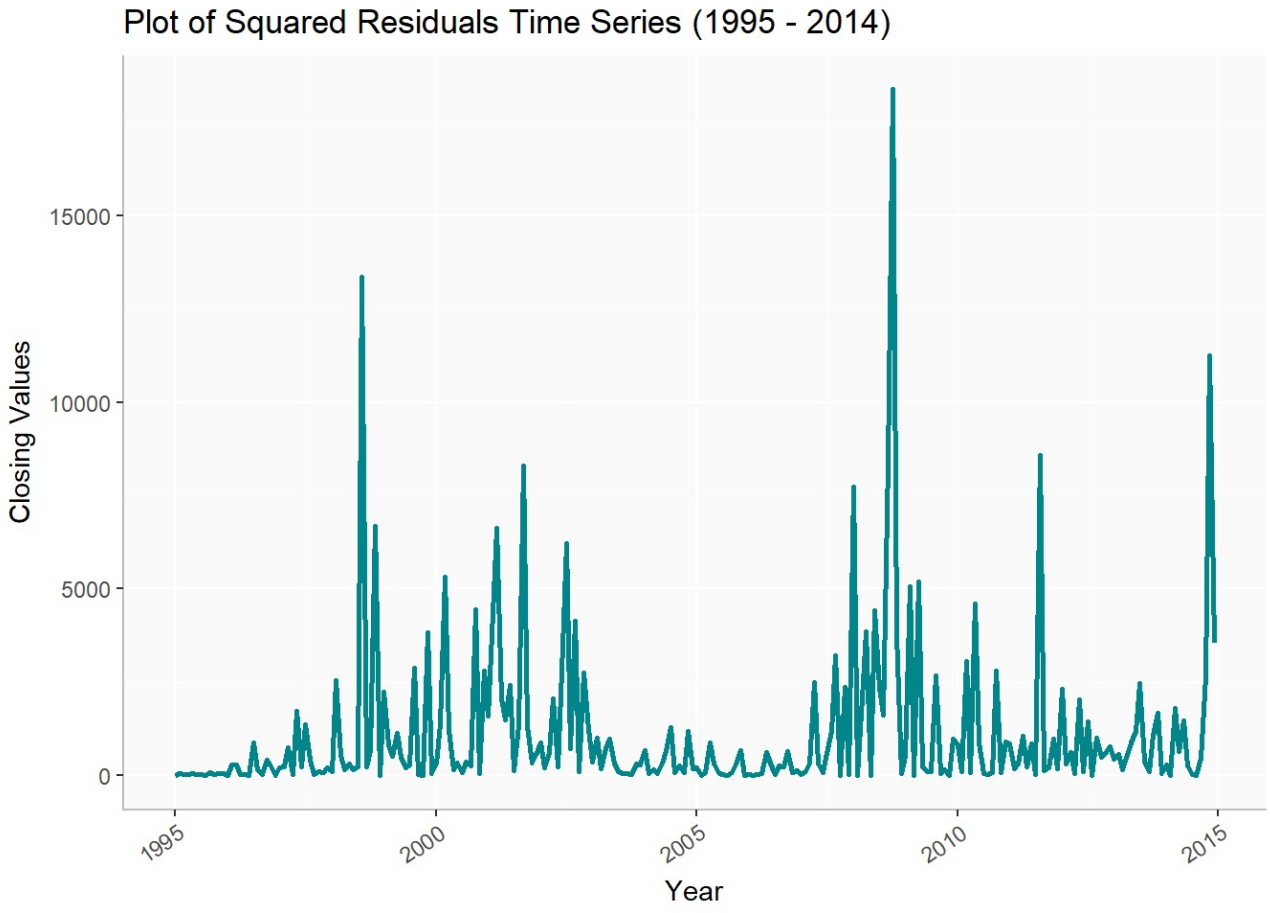
```
accuracy(fit_snaive)
```

```
##                    ME     RMSE      MAE      MPE    MAPE MASE      ACF1
## Training set 73.06769 219.6302 189.5676 4.738094 16.6731    1 0.9647997
```

```
accuracy(fit_ets)
```

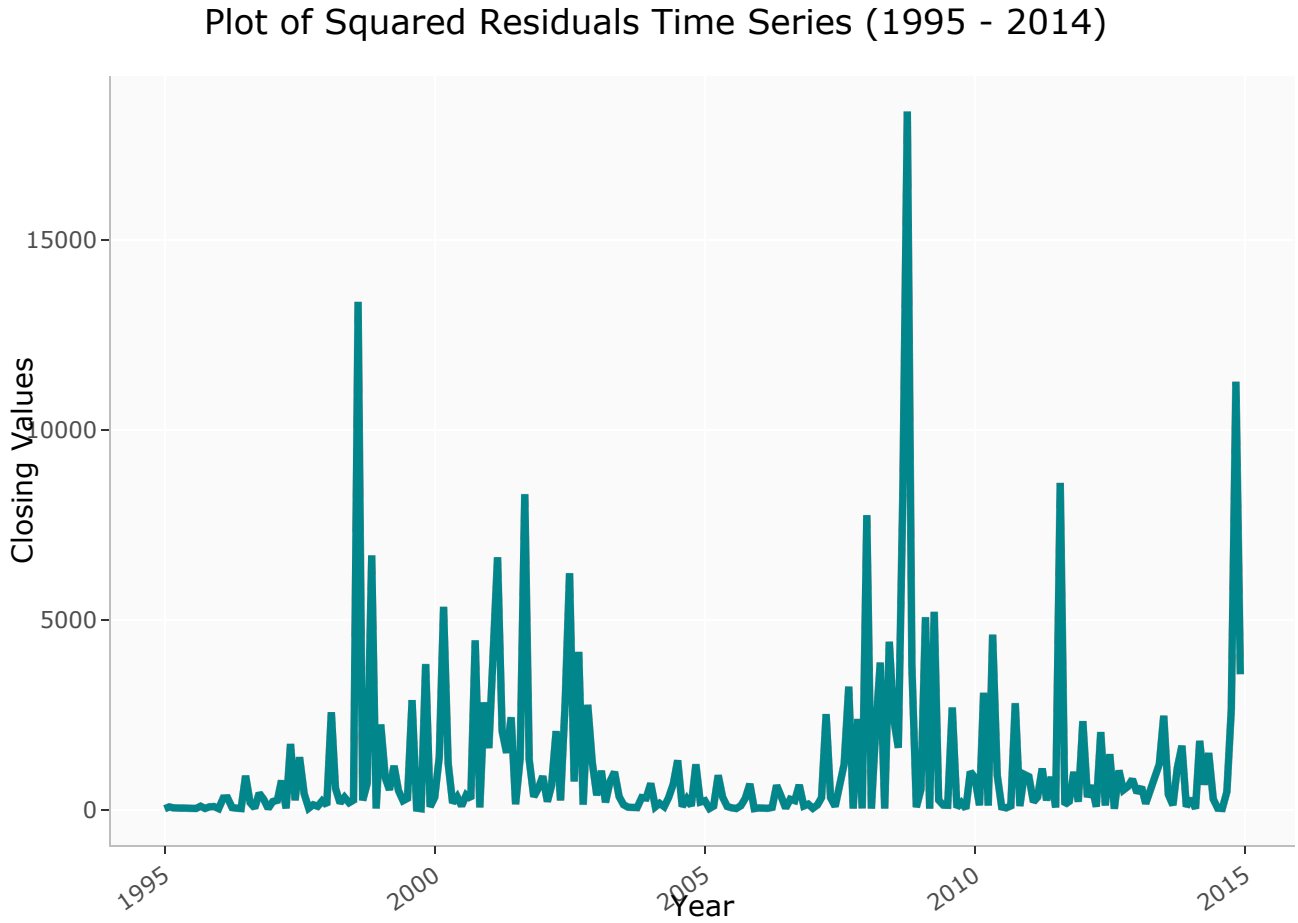
```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 2.648054 36.65711 27.74495 0.2519063 2.409392 0.1463591
##                    ACF1
## Training set 0.1592457
```

```
# CONCLUSIONS
# The model with the best diagnostics is our ARIMA Model

# ARCH Modeling
# Here we first square the residuals and plot the time series/ACF/PACF
# to see if there is correlation within the residuals.
# If there is we can continue adding on to our ARIMA model with a gARCH
# aspect that helps in the volatity of our data.
squared_res_fit <- fit$residuals^2

sq_res <- plot_time_series(squared_res_fit, "Squared Residuals")

sq_res
```

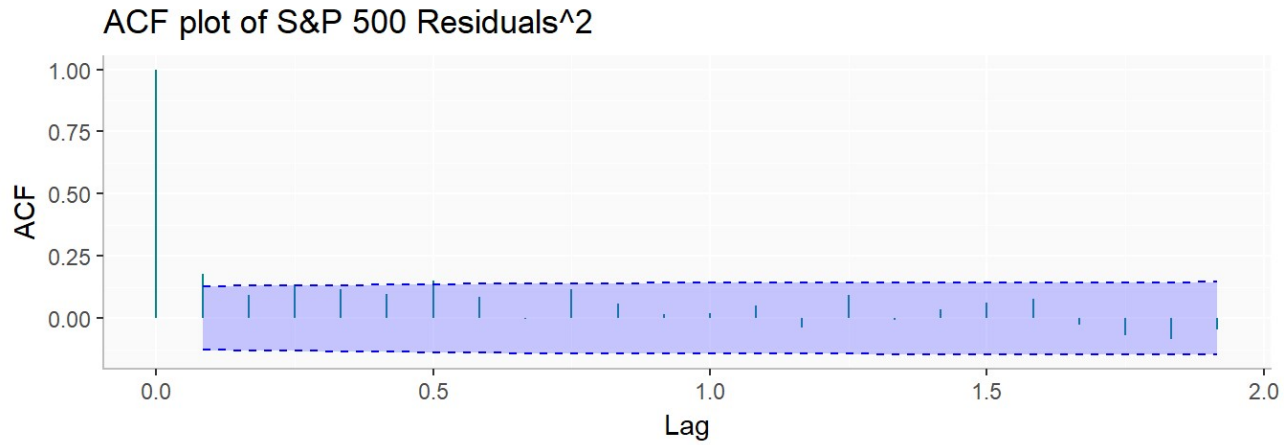


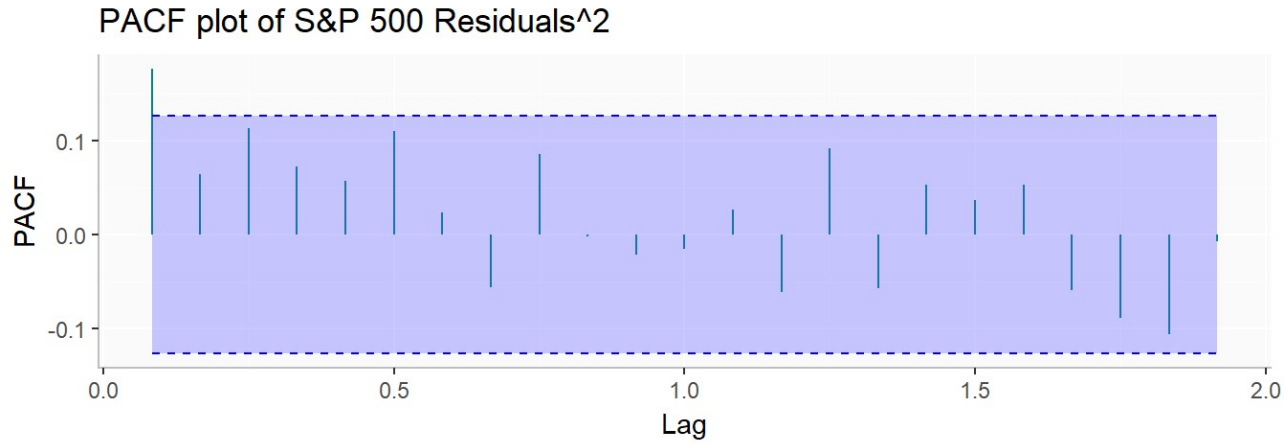

```
ggplotly(sq_res)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```




```
# ACF AND PACF PLOT FOR SQUARED RESIDUALS
plot_acf_pacf(squared_res_fit, 'S&P 500 Residuals^2')
```






```
# The acf plot shows one significant lag, as does the pacf,
# but that isn't enough to suggest we need GARCH modeling
gfit <- garch(fit$residuals, order = c(1,1), trace = TRUE)
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     1.035732e+03     1.000e+00
##      2     5.000000e-02     1.000e+00
##      3     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1  9.591e+02
##      1    3  9.578e+02  1.28e-03  3.40e-03  4.8e-05  3.3e+02  1.0e-01  5.55e-01
##      2    5  9.577e+02  8.70e-05  9.35e-05  4.8e-06  2.7e+01  1.0e-02  4.23e-02
##      3    7  9.576e+02  1.26e-04  1.26e-04  9.6e-06  2.0e+00  2.0e-02  6.79e-04
##      4    9  9.576e+02  1.83e-05  1.83e-05  1.9e-06  1.0e+01  4.0e-03  3.90e-04
##      5   11  9.576e+02  3.00e-05  3.01e-05  3.8e-06  2.0e+00  8.0e-03  1.40e-04
##      6   13  9.576e+02  4.99e-06  4.99e-06  7.4e-07  1.7e+01  1.6e-03  8.28e-05
##      7   16  9.576e+02  9.64e-08  9.64e-08  1.5e-08  7.2e+02  3.2e-05  4.99e-05
##      8   18  9.576e+02  1.92e-07  1.92e-07  3.0e-08  9.1e+01  6.4e-05  4.68e-05
##      9   20  9.576e+02  3.83e-07  3.83e-07  5.9e-08  4.6e+01  1.3e-04  4.65e-05
##     10   23  9.576e+02  7.64e-09  7.64e-09  1.2e-09  8.9e+03  2.6e-06  4.60e-05
##     11   25  9.576e+02  1.53e-08  1.53e-08  2.4e-09  1.1e+03  5.1e-06  4.58e-05
##     12   27  9.576e+02  3.06e-08  3.06e-08  4.7e-09  5.6e+02  1.0e-05  4.57e-05
##     13   29  9.576e+02  6.11e-09  6.11e-09  9.4e-10  1.1e+04  2.0e-06  4.57e-05
##     14   31  9.576e+02  1.22e-09  1.22e-09  1.9e-10  5.6e+04  4.1e-07  4.57e-05
##     15   33  9.576e+02  2.44e-09  2.44e-09  3.8e-10  7.0e+03  8.2e-07  4.57e-05
##     16   35  9.576e+02  4.89e-10  4.89e-10  7.6e-11  1.4e+05  1.6e-07  4.57e-05
##     17   37  9.576e+02  9.77e-10  9.77e-10  1.5e-10  1.7e+04  3.3e-07  4.57e-05
##     18   39  9.576e+02  1.95e-10  1.95e-10  3.0e-11  3.5e+05  6.6e-08  4.57e-05
##     19   41  9.576e+02  3.91e-10  3.91e-10  6.0e-11  4.4e+04  1.3e-07  4.57e-05
##     20   43  9.576e+02  7.82e-10  7.82e-10  1.2e-10  2.2e+04  2.6e-07  4.57e-05
##     21   45  9.576e+02  1.56e-10  1.56e-10  2.4e-11  4.4e+05  5.2e-08  4.57e-05
##     22   47  9.576e+02  3.13e-11  3.13e-11  4.8e-12  2.2e+06  1.0e-08  4.57e-05
##     23   49  9.576e+02  6.25e-12  6.25e-12  9.7e-13  1.1e+07  2.1e-09  4.57e-05
##     24   51  9.576e+02  1.25e-12  1.25e-12  1.9e-13  5.4e+07  4.2e-10  4.57e-05
##     25   53  9.576e+02  2.50e-12  2.50e-12  3.9e-13  6.8e+06  8.4e-10  4.57e-05
##     26   55  9.576e+02  5.00e-13  5.00e-13  7.7e-14  1.4e+08  1.7e-10  4.57e-05
##     27   57  9.576e+02  1.00e-12  1.00e-12  1.5e-13  1.7e+07  3.4e-10  4.57e-05
##     28   59  9.576e+02  2.00e-13  2.00e-13  3.1e-14  3.4e+08  6.7e-11  4.57e-05
##     29   62  9.576e+02  1.60e-12  1.60e-12  2.5e-13  1.1e+07  5.4e-10  4.57e-05
##     30   65  9.576e+02  3.30e-14  3.20e-14  5.0e-15  2.1e+09  1.1e-11  4.57e-05
##     31   67  9.576e+02  6.36e-14  6.40e-14  9.9e-15  2.7e+08  2.1e-11  4.57e-05
##     32   68  9.576e+02 -1.04e+07  1.28e-13  2.0e-14  5.3e+08  4.3e-11  4.57e-05
##
##  ***** FALSE CONVERGENCE *****
##
##  FUNCTION     9.575700e+02   RELDX        1.980e-14
##  FUNC. EVALS      68         GRAD. EVALS      32
##  PRELDF       1.281e-13      NPRELDF      4.565e-05
```

```
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    1.035732e+03     1.000e+00     6.818e-03
##      2    1.505576e-01     1.000e+00    -8.455e-01
##      3    7.106407e-12     1.000e+00     2.728e+00
```

```
print ("end of script.")
```

```
## [1] "end of script."
```