Prediction of Oil Sales

Statistical Learning II | Project 2

Mounika Pasupuleti, Saahithi Chippa, Sahithya Arveti Nagaraju, Sushmitha Manjunatha

1. Load the dataset

```r
# Loading the data
set.seed(90)
data <- read.csv("oil.csv", na.strings = c("", "NA"))
head(data)
```

```
##         date dcoilwtico
## 1 2013-01-01         NA
## 2 2013-01-02      93.14
## 3 2013-01-03      92.97
## 4 2013-01-04      93.12
## 5 2013-01-07      93.20
## 6 2013-01-08      93.21
```

Given dataset consists of only 2 columns date and dcoilwtico.

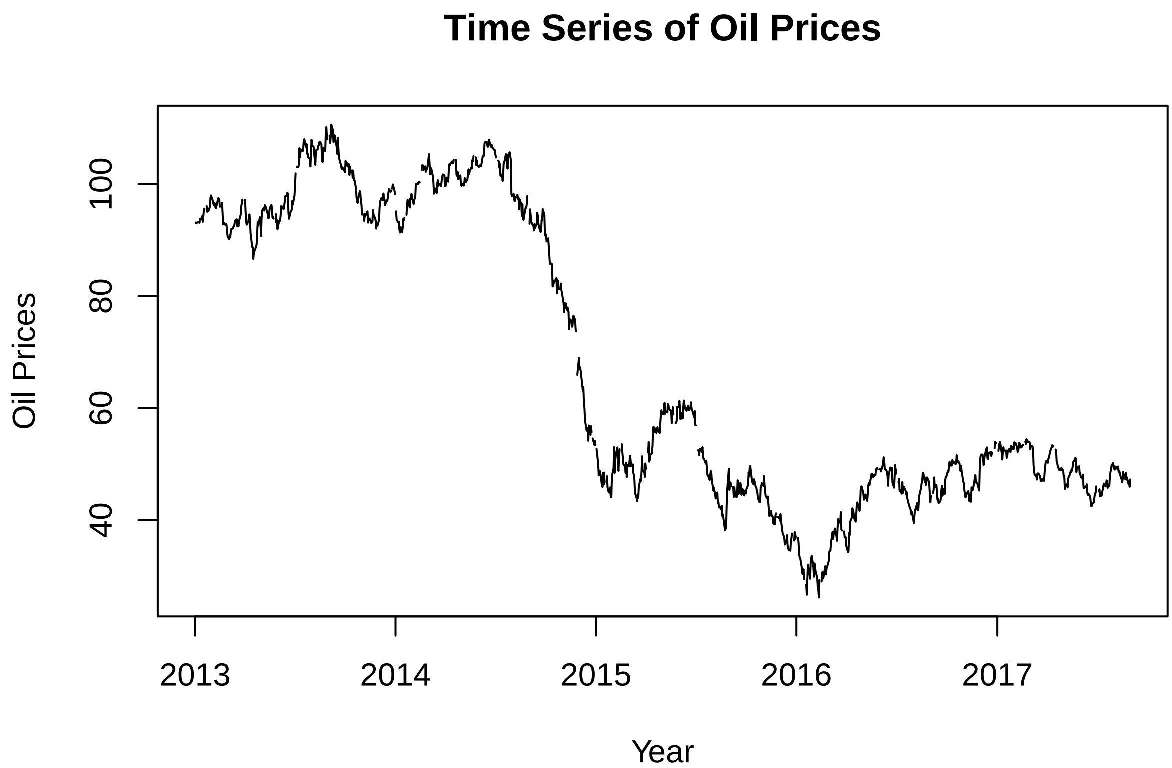
2. Plot the time series as is (without imputation)

```r
# Convert date column to Date type
str(data)
```

```
## 'data.frame':    1218 obs. of  2 variables:
##  $ date      : chr  "2013-01-01" "2013-01-02" "2013-01-03" "2013-01-04" ...
##  $ dcoilwtico: num  NA 93.1 93 93.1 93.2 ...
```

```r
data$date <- as.Date(data$date, format = "%Y-%m-%d")

plot(data, ylab = 'Oil Prices', xlab = 'Year', type = 'l', main = "Time Series of Oil Prices")
```



The graph clearly shows that there are some gaps in the data, indicating that some values are missing in the dataset.

3 Filling the missing data

```
sum(is.na(data))
```

```
## [1] 43
```

The number of missing values in this dataset are "43".

Linear interpolation is the method used in time series data imputation where an estimate of the missing values from the nearest available values both before and after the gap in the series is calculated.

This method, which is a straight-line approach to fill in missing data, has been applied with the na.approx() function in the zoo package in R. The argument rule = 2 handles missing values at the start or end of the series by using available data from the opposite end. It works quite well when data follows a relatively smooth, linear trend and gaps are small.

Linear interpolation is simple, computationally efficient, and widely used in time series preprocessing, especially when the missing values are scattered and do not represent major structural breaks in the data.

```
#Linear Interpolation
library(zoo)
```

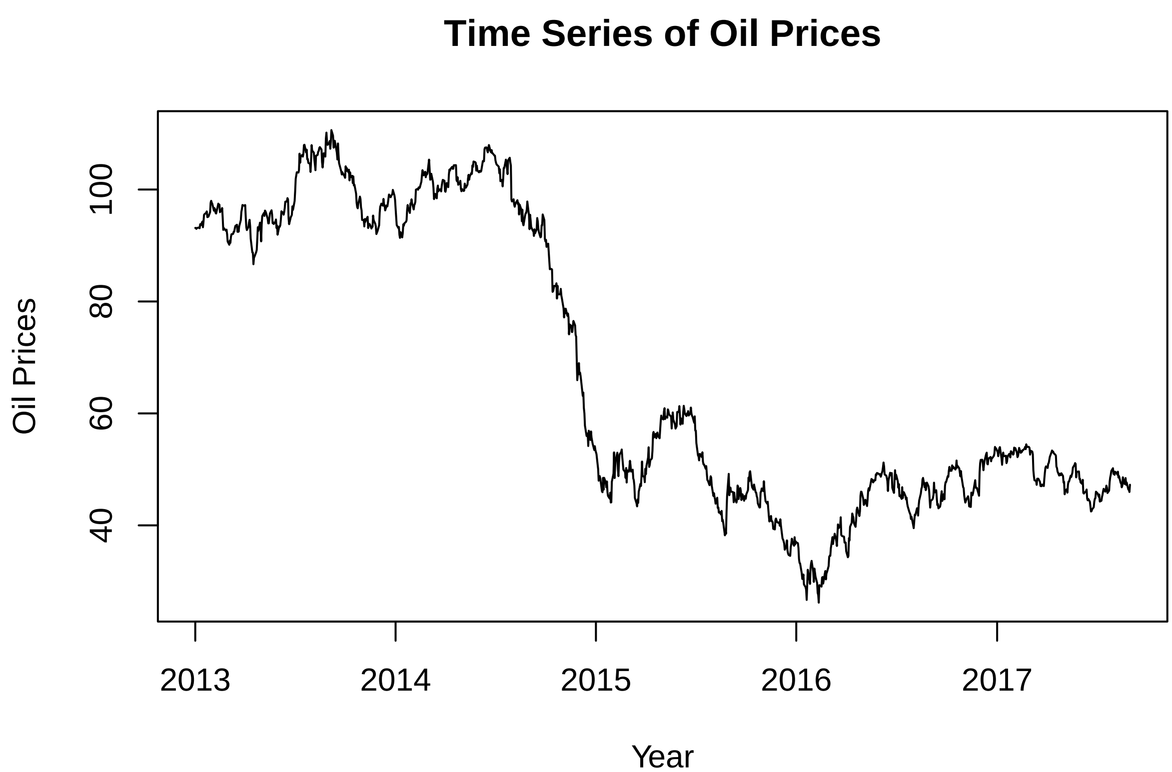
```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
data$dcoilwtico <- na.approx(data$dcoilwtico, rule = 2)
```

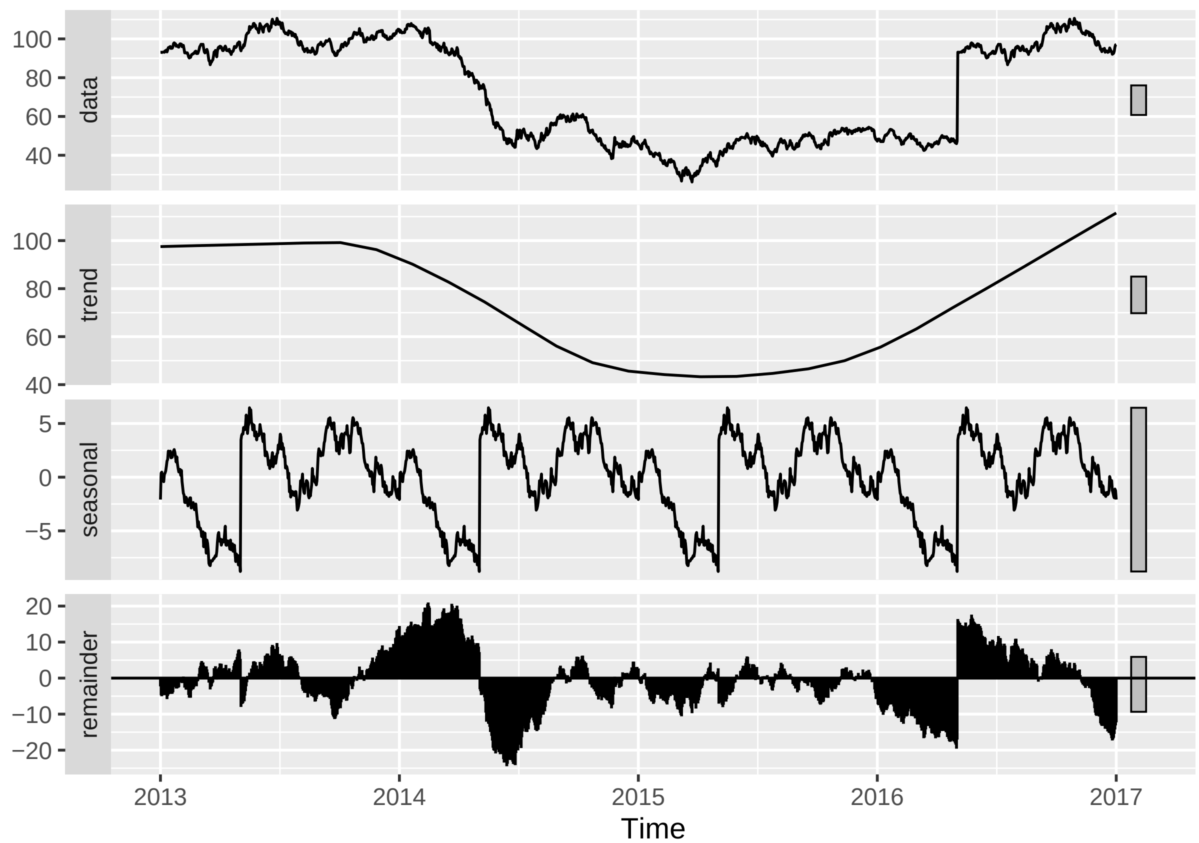
4. Time Series with imputed data.

```
plot(data, ylab = 'Oil Prices', xlab = 'Year', type = 'l', main = "Time Series of Oil Prices")
```




```
# Decompose the time series to find trend ans seasonality
library(forecast)
oil_ts <- ts(data$dcoilwtico, start = c(2013), end=c(2017), frequency = 365)

decomposed <- stl(oil_ts, s.window = "periodic")
autoplot(decomposed)
```



Trend and/or seasonality in the data?

Trend: The above graph clearly shows a downward trend in the data, which means that overall oil prices decreased from 2013 to 2015 and then stabilizes in the mid-2015 and started a slow upward trends post-mid-2015.

Seasonality: From the seasonal graph plotted above clearly shows a repetitive cycle with some small fluctuations in the oil prices. By observing the seasonality graph there are recurring yearly patterns, with price peaks at the start of the year and declines towards end which indicates that there is annual seasonality.

Summary: The overall graph shows a trend shifts from decline to gradual rise in oil prices and seasonality shows repeating annual cycles of price peaks and dips.

5. ETS models and about Holt-Winters models

Theoretical Overview

1. ETS (Error, Trend, Seasonality) Models The **ETS** (Error, Trend, Seasonality), is one of the more common model frameworks used in time series forecasting. It breaks down a given univariate time series into three key components:

- **Error (E)**: This represents the residual or random variation in the data once the trend and seasonality are accounted for. The error component may be:

  - **Additive**: The error is the same over time.
  - **Multiplicative**: The error changes proportionally with the size of the data.

- **Trend (T)**: Models the underlying long-term direction or movement, which can be an increase, a decrease, or no change. The trend component can be:
  - **None**: No trend is modeled.
  - **Additive**: The trend increases or decreases by a fixed amount over time.
  - **Multiplicative**: The trend grows or shrinks at a rate proportional to the current level of the data.

- **Seasonality (S)**: This represents periodic variations in the data that recur at fixed periods, for instance, yearly or monthly. The seasonality component can be one of the following:
  - **None**: No seasonality is assumed.
  - **Additive**: The effect of the seasonality does not change over time.
  - **Multiplicative**: The effect of seasonality changes proportionally with the level of the data.

This gives several possible ETS models, such as:

- **ETS(AAA)**: Additive error, additive trend, and additive seasonality
- **ETS(AAM)**: Additive error, additive trend, and multiplicative seasonality
- **ETS(AMM)**: Additive error, multiplicative trend, and multiplicative seasonality
- **ETS(MMM)**: Multiplicative error, multiplicative trend, and multiplicative seasonality

In **R**, the `ets()` function automatically selects the best-fitting model based on criteria such as the **Akaike Information Criterion (AIC)**, which is used to select the model that minimizes the information loss.

---

2. Holt-Winters Models The Holt-Winters method is the peculiar case of the ETS model targeted at time series data possessing both trend and seasonality. The most important aspects of the Holt-Winters model are:

- **Level (L)**: it gives the smoothed value of the time series at a certain time representing the base value.
- **Trend (T)**: to capture slope-the direction in which the values are moving over time, indicating either an increase or a decrease.
- **Seasonality (S)**: Repeated patterns, cycles appearing at fixed intervals within the series.

There in exist two key forms depending on the kind of application used with the seasonality component:

- **Additive Seasonality**: The magnitude of the seasonal fluctuations remains constant over time.
- **Multiplicative Seasonality**: The magnitude of the seasonality wobbles in the series increases as well as decreases in the relation to the level in a series.

The Holt-Winters method contains three smoothing parameters that regulate the speed of response of the model to new data:

- **Alpha ( )**: The smoothing parameter for the level component. It controls how quickly the model adapts to new observations for the level.
- **Beta ( )**: The smoothing parameter for the trend component. It governs how quickly the model adjusts to changes in the trend.
- **Gamma ( )**: The smoothing parameter for the seasonal component. It regulates how quickly the model adjusts to changes in seasonality.

Additionally, a **damped** version of Holt-Winters model is also used where the trend damps out over time. Such models are useful when a trend is expected to diminish but not continue indefinitely.

6. Suitable model(s) for the data.

From the above identified trend and seasonality, we can clearly exhibits a clear annual seasonal pattern with non-linear trend. Based on this, we can explore the following models:

1. ETS Models: As discussed above we can explore this model as it automatically choose best combination.
2. Holt-Winters Models: As discussed above it is special case of ETS which explicitly focusing on trend and seasonality. So, we can explore this model.
3. STL + ETS Models: As our model contain both seasonal pattern and non-linear trend. This model helps us to handle complex seasonal patterns and non-linear trends.
4. SARIMA (Seasonal ARIMA): This model is an extension of ARIMA model which will be useful for time series with seasonal patterns and autocorrelated residuals.

```
n <- nrow(data)
train <- data[1:(n - 12), ]  # Training set (all except the last 12 observations)
test <- data[(n - 11):n, ]   # Test set (last 12 observations)

# Converting training data to a time series object
train_ts <- ts(train$dcoilwtico, start = c(2013), frequency = 365)
```

7. Run the models and check their adequacy.

ETS Model

```
# Forecasting for next 30 days
ets_forecast <- forecast(ets(train_ts), h = 30)
```

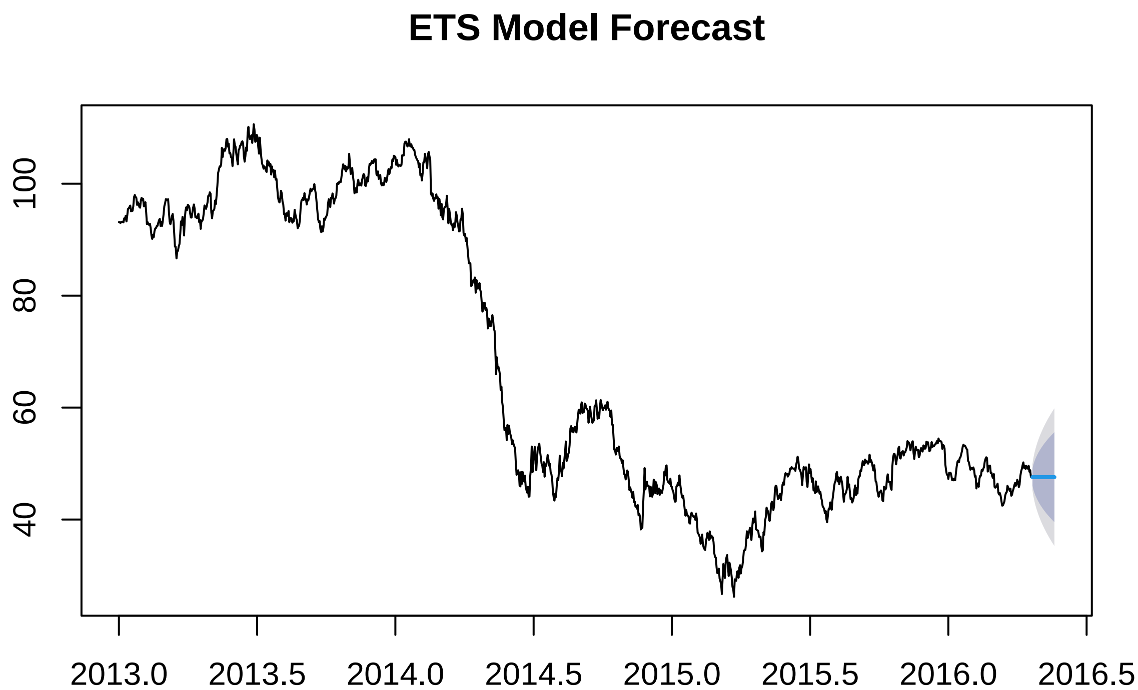
```
## Warning in ets(train_ts): I can't handle data with frequency greater than 24.
## Seasonality will be ignored. Try stlf() if you need seasonal forecasts.
```

```
summary(ets_forecast)
```

```
##
## Forecast method: ETS(A,N,N)
##
## Model Information:
## ETS(A,N,N)
##
## Call:
## ets(y = train_ts)
##
##   Smoothing parameters:
##     alpha = 0.9705
##
##   Initial states:
##     l = 93.131
##
##   sigma:  1.1795
##
```

```
##      AIC     AICc      BIC
## 8958.799 8958.819 8974.084
##
## Error measures:
##                     ME     RMSE       MAE         MPE     MAPE       MASE
## Training set -0.038925 1.178499 0.8962004 -0.08202926 1.547217 0.03231939
##                       ACF1
## Training set -0.0004452751
##
## Forecasts:
##           Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2016.3041       47.57164 46.06008 49.08321 45.25991 49.88338
## 2016.3068       47.57164 45.46525 49.67804 44.35020 50.79309
## 2016.3096       47.57164 45.00475 50.13854 43.64592 51.49737
## 2016.3123       47.57164 44.61512 50.52817 43.05004 52.09325
## 2016.3151       47.57164 44.27118 50.87211 42.52401 52.61927
## 2016.3178       47.57164 43.95984 51.18345 42.04786 53.09543
## 2016.3205       47.57164 43.67328 51.47001 41.60962 53.53367
## 2016.3233       47.57164 43.40640 51.73689 41.20145 53.94184
## 2016.3260       47.57164 43.15561 51.98768 40.81790 54.32538
## 2016.3288       47.57164 42.91832 52.22497 40.45500 54.68829
## 2016.3315       47.57164 42.69256 52.45073 40.10973 55.03356
## 2016.3342       47.57164 42.47679 52.66650 39.77974 55.36355
## 2016.3370       47.57164 42.26979 52.87349 39.46316 55.68013
## 2016.3397       47.57164 42.07058 53.07271 39.15850 55.98479
## 2016.3425       47.57164 41.87834 53.26495 38.86448 56.27881
## 2016.3452       47.57164 41.69237 53.45091 38.58008 56.56321
## 2016.3479       47.57164 41.51212 53.63117 38.30439 56.83889
## 2016.3507       47.57164 41.33707 53.80622 38.03668 57.10661
## 2016.3534       47.57164 41.16680 53.97649 37.77628 57.36701
## 2016.3562       47.57164 41.00095 54.14234 37.52263 57.62066
## 2016.3589       47.57164 40.83917 54.30411 37.27522 57.86807
## 2016.3616       47.57164 40.68120 54.46209 37.03362 58.10967
## 2016.3644       47.57164 40.52677 54.61652 36.79743 58.34585
## 2016.3671       47.57164 40.37565 54.76764 36.56632 58.57697
## 2016.3699       47.57164 40.22764 54.91565 36.33996 58.80333
## 2016.3726       47.57164 40.08255 55.06074 36.11807 59.02522
## 2016.3753       47.57164 39.94023 55.20306 35.90040 59.24289
## 2016.3781       47.57164 39.80050 55.34278 35.68671 59.45658
## 2016.3808       47.57164 39.66325 55.48004 35.47680 59.66649
## 2016.3836       47.57164 39.52834 55.61495 35.27047 59.87282
```

```r
plot(ets_forecast, main = "ETS Model Forecast")
```



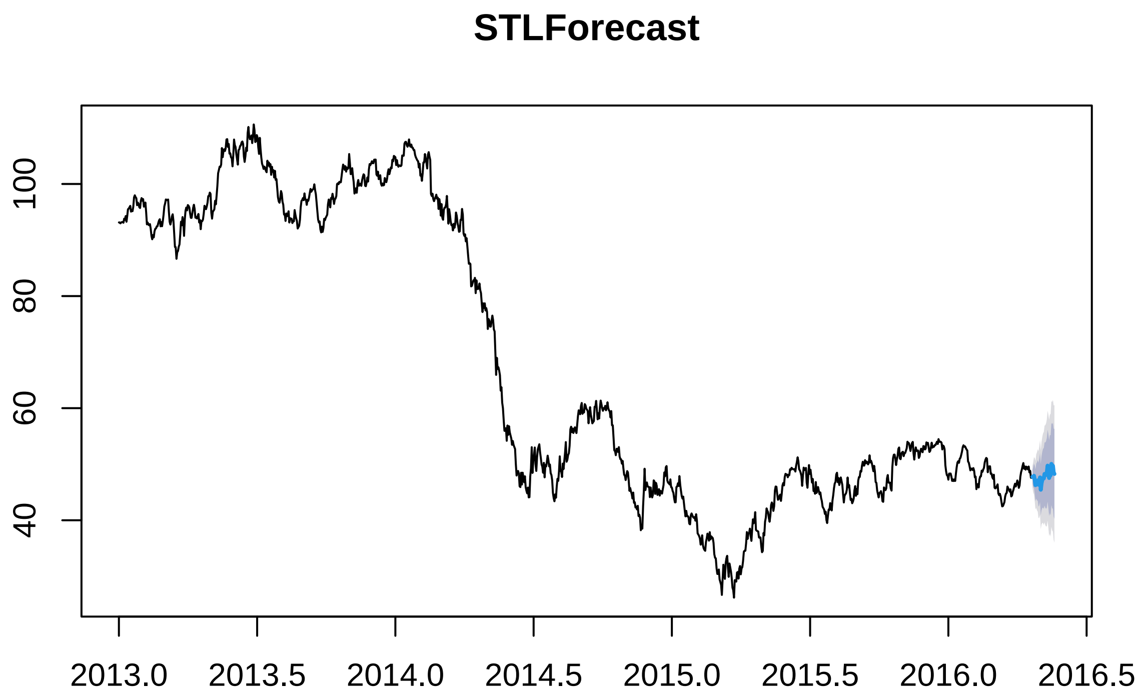
STL Model

```
stl_forecast <- forecast(stlf(train_ts), h = 30)
summary(stl_forecast)
```

```
##
## Forecast method: STL +  ETS(A,Ad,N)
##
## Model Information:
## ETS(A,Ad,N)
##
## Call:
## ets(y = na.interp(x), model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 0.97
##     beta  = 0.0164
##     phi   = 0.9774
##
##   Initial states:
##     l = 90.6927
##     b = 0.0968
##
```

```
##   sigma:  0.9796
##
##      AIC     AICc      BIC
## 8514.036 8514.106 8544.606
##
## Error measures:
##                       ME      RMSE      MAE         MPE     MAPE       MASE
## Training set -0.02278024 0.9776128 0.754535 -0.03848586 1.298884 0.02721055
##                       ACF1
## Training set -0.0003192384
##
## Forecasts:
##           Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2016.3041       47.75777 46.50231 49.01324 45.83770 49.67784
## 2016.3068       47.66613 45.90297 49.42928 44.96961 50.36264
## 2016.3096       47.93181 45.76608 50.09755 44.61960 51.24402
## 2016.3123       47.25218 44.73803 49.76632 43.40712 51.09723
## 2016.3151       46.26226 43.43386 49.09067 41.93659 50.58794
## 2016.3178       46.98494 43.86615 50.10374 42.21516 51.75472
## 2016.3205       46.95814 43.56676 50.34952 41.77148 52.14480
## 2016.3233       47.09968 43.44959 50.74977 41.51735 52.68201
## 2016.3260       46.34223 42.44459 50.23986 40.38131 52.30314
## 2016.3288       46.78855 42.65260 50.92450 40.46315 53.11395
## 2016.3315       47.64245 43.27594 52.00896 40.96445 54.32045
## 2016.3342       45.45298 40.86257 50.04339 38.43255 52.47340
## 2016.3370       46.61550 41.80697 51.42403 39.26149 53.96952
## 2016.3397       47.20430 42.18273 52.22588 39.52447 54.88414
## 2016.3425       47.50386 42.27375 52.73397 39.50510 55.50262
## 2016.3452       47.46685 42.03225 52.90145 39.15534 55.77835
## 2016.3479       48.26520 42.62976 53.90064 39.64653 56.88387
## 2016.3507       48.04769 42.21473 53.88065 39.12694 56.96844
## 2016.3534       48.08419 42.05675 54.11163 38.86602 57.30236
## 2016.3562       48.47117 42.25206 54.69029 38.95986 57.98249
## 2016.3589       49.74089 43.33270 56.14908 39.94041 59.54138
## 2016.3616       48.79626 42.20140 55.39111 38.71030 58.88221
## 2016.3644       47.54606 40.76681 54.32531 37.17809 57.91403
## 2016.3671       48.23347 41.27194 55.19500 37.58673 58.88021
## 2016.3699       48.02385 40.88204 55.16565 37.10140 58.94629
## 2016.3726       50.05056 42.73037 57.37075 38.85530 61.24582
## 2016.3753       49.64728 42.15050 57.14405 38.18195 61.11261
## 2016.3781       49.72080 42.04915 57.39246 37.98802 61.45359
## 2016.3808       48.50051 40.65561 56.34541 36.50277 60.49825
## 2016.3836       48.27195 40.25536 56.28854 36.01164 60.53226
```

```r
plot(stl_forecast, main = "STLForecast")
```



Holt-Winters Model with Model "Additive and Multiplicative Seasonality"

```r
# Forecast using Holt-Winters model
hw_additive <- HoltWinters(train_ts, seasonal = "additive")
forecast_additive <- forecast(hw_additive, h = 30)
summary(forecast_additive)
```

```
##
## Forecast method: HoltWinters
##
## Model Information:
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train_ts, seasonal = "additive")
##
## Smoothing parameters:
##  alpha: 0.9191512
##  beta : 0
##  gamma: 1
##
## Coefficients:
##                [,1]
```

```
## a      46.37619676
## b      -0.09657653
## s1      1.80177330
## s2      1.12365534
## s3      0.71148291
## s4     -0.80897788
## s5     -2.37057191
## s6     -0.84544113
## s7     -1.35845786
## s8     -0.62777580
## s9     -1.56844078
## s10    -1.01776027
## s11    -1.22674085
## s12    -4.17875899
## s13    -2.40784829
## s14    -2.51790321
## s15    -3.31849781
## s16    -3.41159411
## s17    -2.27144767
## s18    -1.28824335
## s19    -1.75998427
## s20    -3.07959326
## s21    -3.30310167
## s22    -6.86399486
## s23   -10.64655900
## s24    -8.06771172
## s25    -9.38690506
## s26    -8.90856766
## s27    -9.41169872
## s28   -10.14439194
## s29   -12.68569712
## s30   -12.16836683
## s31   -14.40147478
## s32   -15.28812335
## s33   -17.25401385
## s34   -19.07116197
## s35   -18.87181584
## s36   -18.26749112
## s37   -20.24566650
## s38   -17.82215504
## s39   -18.85271425
## s40   -17.46222000
## s41   -18.25731302
## s42   -18.66858949
## s43   -19.10926371
## s44   -19.95786124
## s45   -19.29683597
## s46   -19.63558904
## s47   -19.94061059
## s48   -20.11912262
## s49   -22.48982633
## s50   -24.44509343
## s51   -23.80130734
## s52   -23.50259730
```

```
## s53  -23.79478495
## s54  -25.66459826
## s55  -25.83552708
## s56  -23.33355869
## s57  -25.06761078
## s58  -23.17985385
## s59  -23.56832466
## s60  -24.25671485
## s61  -23.21366257
## s62  -24.85435926
## s63  -25.17364067
## s64  -25.52177483
## s65  -24.61143403
## s66  -25.96965478
## s67  -25.90511505
## s68  -22.61343535
## s69  -20.71974833
## s70  -16.69293618
## s71  -20.45794310
## s72   -7.73902653
## s73    5.40909424
## s74    5.77563993
## s75    4.42174089
## s76    6.86973288
## s77    5.61975639
## s78    4.19903519
## s79    2.88527717
## s80    2.54841279
## s81    1.53963812
## s82    1.98603911
## s83    1.76635684
## s84    1.26165276
## s85    0.89035758
## s86    2.83113288
## s87    1.95639209
## s88    2.63430473
## s89    1.85930454
## s90    2.24574217
## s91    0.68269534
## s92    1.44719598
## s93    0.86184580
## s94    0.93563124
## s95   -0.22635874
## s96    0.80016602
## s97   -0.48745995
## s98   -0.79061747
## s99   -1.98425221
## s100  -3.69015104
## s101  -4.51369925
## s102  -4.77055029
## s103  -4.07874753
## s104  -2.86120360
## s105  -3.22940236
## s106  -4.45144986
```

```
## s107  -4.93166673
## s108  -6.95851300
## s109  -7.25443863
## s110  -8.16028204
## s111  -7.12603028
## s112  -7.47888139
## s113  -7.13872411
## s114  -6.73352031
## s115  -8.35345641
## s116  -8.46420109
## s117  -8.33041106
## s118  -8.05045396
## s119  -8.82510164
## s120  -8.61433695
## s121  -8.44675169
## s122  -6.80179825
## s123  -7.67272671
## s124  -8.07941663
## s125  -8.79454670
## s126  -9.94545200
## s127  -9.81607027
## s128  -9.75836469
## s129  -8.75378255
## s130  -6.57164870
## s131  -5.19346323
## s132  -4.94132695
## s133  -4.93497034
## s134  -4.97372637
## s135  -4.01642781
## s136  -4.53796817
## s137  -5.00856076
## s138  -5.73764297
## s139  -5.03852787
## s140  -5.06938197
## s141  -4.60517971
## s142  -3.91871876
## s143  -3.28277492
## s144  -3.54623727
## s145  -3.44856991
## s146  -3.15333838
## s147  -2.93335121
## s148  -2.49648750
## s149  -3.29818181
## s150  -4.00001173
## s151  -5.24936845
## s152  -6.95313299
## s153  -8.16261312
## s154  -9.07113486
## s155  -8.98912131
## s156 -10.35619492
## s157 -10.89686865
## s158 -10.06322643
## s159 -10.93950692
## s160 -10.13009955
```

```
## s161  -8.78164191
## s162  -8.84207865
## s163  -8.50699523
## s164  -8.06861686
## s165  -7.75640314
## s166  -6.16715705
## s167  -5.17829056
## s168  -5.64314530
## s169  -6.23572797
## s170  -4.63980461
## s171  -4.88585901
## s172  -4.06910446
## s173  -4.43645615
## s174  -5.36144401
## s175  -4.74784754
## s176  -4.90432839
## s177  -4.22549244
## s178  -2.45378720
## s179  -2.10997886
## s180  -1.95932330
## s181  -1.24156091
## s182  -1.27975941
## s183  -1.38488063
## s184  -0.51425479
## s185   0.64563218
## s186   1.43276598
## s187   1.62885823
## s188   1.12798896
## s189   1.61802127
## s190   0.93452830
## s191   1.17414526
## s192   1.23109006
## s193   1.61188980
## s194   3.67509982
## s195   2.73422666
## s196   0.94367030
## s197   0.79952617
## s198   1.54312470
## s199   0.50556170
## s200  -0.37400384
## s201  -2.17296458
## s202  -2.27510635
## s203  -1.66914384
## s204  -2.09378914
## s205  -0.90806560
## s206   0.17779515
## s207  -0.56647322
## s208  -0.30223210
## s209  -0.32768667
## s210  -0.40061761
## s211   0.49479027
## s212   1.37420151
## s213   1.83890602
## s214   1.81916315
```

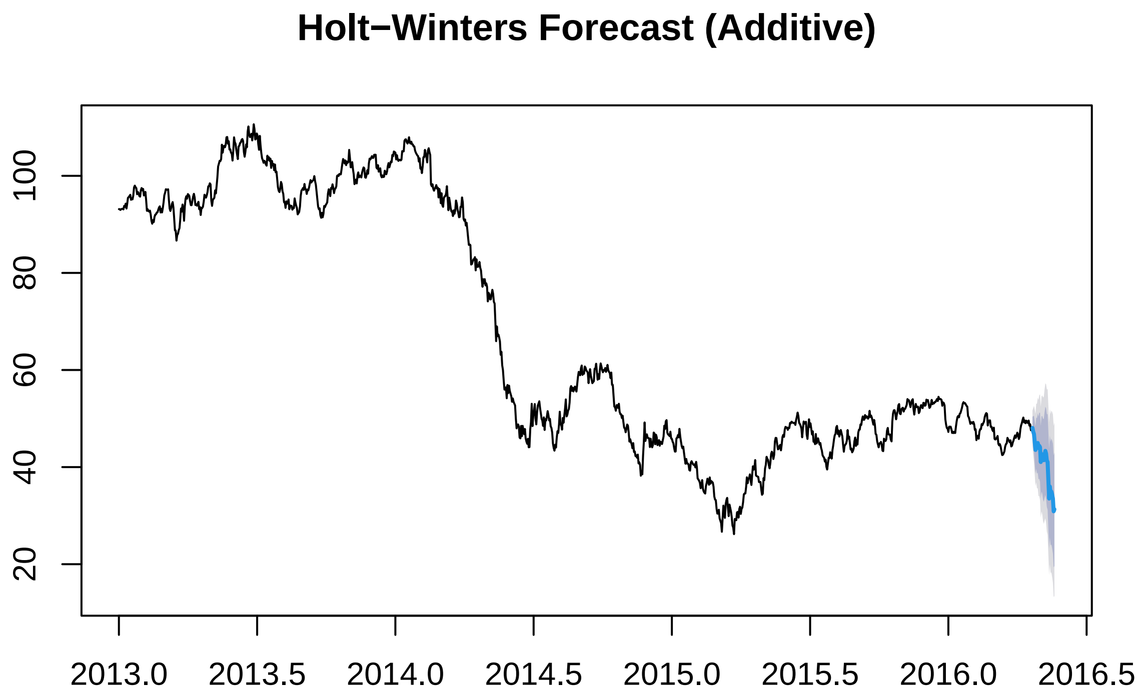
```
## s215   0.56585978
## s216   0.17488804
## s217   1.30170419
## s218   2.29814059
## s219   1.98473089
## s220   3.21594300
## s221   4.72556532
## s222   4.99622876
## s223   5.20910096
## s224   5.68718449
## s225   5.64695349
## s226   5.71653383
## s227   6.50770059
## s228   6.61164852
## s229   6.69121114
## s230   4.66644452
## s231   4.50790941
## s232   5.06834978
## s233   4.02044061
## s234   4.38785260
## s235   4.87295844
## s236   3.70688323
## s237   3.52246686
## s238   3.96271480
## s239   3.76356682
## s240   4.06905842
## s241   5.17642604
## s242   5.13057806
## s243   5.05462840
## s244   5.71728542
## s245   7.01740629
## s246   7.66455248
## s247   7.24377278
## s248   7.85460197
## s249   8.35552840
## s250   8.59314457
## s251   9.88643946
## s252  10.14220955
## s253  11.23516203
## s254  11.32445145
## s255  10.50986485
## s256  10.25417468
## s257  11.07976765
## s258  10.58562980
## s259  10.33057611
## s260  10.69002792
## s261  11.09792990
## s262  10.94678811
## s263  11.24931342
## s264  12.78191712
## s265  13.38987921
## s266  13.34143332
## s267  15.29346003
## s268  15.93991454
```

```
## s269  16.17988509
## s270  16.00829030
## s271  15.66601848
## s272  16.14882719
## s273  17.09723586
## s274  16.58949188
## s275  16.27598839
## s276  16.71652566
## s277  16.46081419
## s278  16.41105282
## s279  16.28014059
## s280  16.51442623
## s281  15.69584803
## s282  15.43915451
## s283  15.24507470
## s284  15.10255798
## s285  15.20077021
## s286  14.28986337
## s287  14.95011616
## s288  13.26729059
## s289  13.14770569
## s290  12.36626325
## s291  13.56399444
## s292  15.31642097
## s293  15.88815373
## s294  17.42439808
## s295  17.10002658
## s296  16.44602808
## s297  15.55728572
## s298  17.58376277
## s299  18.48897699
## s300  18.09270909
## s301  17.58586316
## s302  12.48790263
## s303  11.38874959
## s304  11.62651597
## s305  11.13242150
## s306  10.55117910
## s307  10.84755134
## s308  11.45515966
## s309  11.93834780
## s310  11.43439654
## s311  11.41509586
## s312  10.00580520
## s313  11.33529096
## s314  10.82979558
## s315   9.19793978
## s316  10.57829848
## s317   9.10573226
## s318   8.45791362
## s319  10.05571506
## s320  10.73744195
## s321  10.70372982
## s322  11.76929115
```

```
## s323  13.23811796
## s324  11.08514236
## s325   8.77484183
## s326  10.95571683
## s327  10.51063996
## s328   9.49882404
## s329   8.67379057
## s330   8.78791140
## s331   8.30395628
## s332   9.06480324
## s333   8.76073130
## s334   9.22054386
## s335  11.08067515
## s336  11.06539232
## s337  10.05109443
## s338   9.61318527
## s339   8.68724323
## s340   8.72537195
## s341  10.65496194
## s342  10.93541111
## s343  12.59402598
## s344  12.21508366
## s345   9.36116506
## s346   8.49175845
## s347   8.78671897
## s348   7.75613194
## s349   8.33093019
## s350   7.42312321
## s351   5.82925646
## s352   4.33126990
## s353   4.39610205
## s354   4.41602960
## s355   0.76005369
## s356   0.62908559
## s357   1.00284577
## s358   1.68866946
## s359   1.63705488
## s360   2.16147033
## s361   0.33330496
## s362   1.91892002
## s363   0.93207505
## s364   0.82198282
## s365   1.19380324
##
## Error measures:
##                      ME    RMSE      MAE        MPE     MAPE       MASE
## Training set 0.02712318 1.80837 1.082395 0.08973982 2.370685 0.03903405
##                   ACF1
## Training set 0.1489383
##
## Forecasts:
##           Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2016.3041       48.08139 45.76276 50.40003 44.53534 51.62744
## 2016.3068       47.30670 44.15741 50.45599 42.49028 52.12312
```

```
## 2016.3096       46.79795 42.99533 50.60057 40.98234 52.61356
## 2016.3123       45.18091 40.82180 49.54002 38.51423 51.84760
## 2016.3151       43.52274 38.67055 48.37494 36.10195 50.94353
## 2016.3178       44.95130 39.65170 50.25089 36.84627 53.05633
## 2016.3205       44.34170 38.62964 50.05376 35.60586 53.07754
## 2016.3233       44.97581 38.87913 51.07249 35.65174 54.29988
## 2016.3260       43.93857 37.48013 50.39701 34.06123 53.81590
## 2016.3288       44.39267 37.59169 51.19366 33.99146 54.79388
## 2016.3315       44.08711 36.96003 51.21420 33.18718 54.98705
## 2016.3342       41.03852 33.59962 48.47742 29.66170 52.41534
## 2016.3370       42.71285 34.97469 50.45102 30.87835 54.54735
## 2016.3397       42.50622 34.47995 50.53250 30.23109 54.78135
## 2016.3425       41.60905 33.30465 49.91345 28.90857 54.30953
## 2016.3452       41.41938 32.84588 49.99288 28.30734 54.53142
## 2016.3479       42.46295 33.62853 51.29736 28.95188 55.97402
## 2016.3507       43.34958 34.26174 52.43741 29.45093 57.24822
## 2016.3534       42.78126 33.44688 52.11564 28.50555 57.05697
## 2016.3562       41.36507 31.79049 50.93966 26.72201 56.00813
## 2016.3589       41.04499 31.23609 50.85389 26.04357 56.04641
## 2016.3616       37.38752 27.34977 47.42527 22.03610 52.73894
## 2016.3644       33.50838 23.24688 43.76988 17.81477 49.20199
## 2016.3671       35.99065 25.51017 46.47112 19.96215 52.01915
## 2016.3699       34.57488 23.87991 45.26984 18.21834 50.93141
## 2016.3726       34.95664 24.05140 45.86188 18.27852 51.63476
## 2016.3753       34.35693 23.24540 45.46846 17.36331 51.35055
## 2016.3781       33.52766 22.21360 44.84172 16.22430 50.83103
## 2016.3808       30.88978 19.37675 42.40281 13.28211 48.49745
## 2016.3836       31.31053 19.60191 43.01916 13.40374 49.21733
```

```r
plot(forecast_additive, main = "Holt-Winters Forecast (Additive)")
```




```r
# Multiplicative Seasonality
hw_multiplicative <- HoltWinters(train_ts, seasonal = "multiplicative")
forecast_multiplicative <- forecast(hw_multiplicative, h = 30)
summary(forecast_multiplicative)
```

```
##
## Forecast method: HoltWinters
##
## Model Information:
## Holt-Winters exponential smoothing with trend and multiplicative seasonal component.
##
## Call:
## HoltWinters(x = train_ts, seasonal = "multiplicative")
##
## Smoothing parameters:
##  alpha: 0.9595718
##  beta : 0
##  gamma: 1
##
## Coefficients:
##             [,1]
## a    46.47788287
## b    -0.09657653
## s1    1.03115754
## s2    1.02212645
```

```
## s3    1.01712257
## s4    0.99733126
## s5    0.97754399
## s6    0.99860644
## s7    0.99105844
## s8    1.00091755
## s9    0.98808271
## s10   0.99605275
## s11   0.99290017
## s12   0.95349326
## s13   0.97868274
## s14   0.97638761
## s15   0.96538735
## s16   0.96489929
## s17   0.98035147
## s18   0.99313749
## s19   0.98616206
## s20   0.96815950
## s21   0.96565355
## s22   0.91717447
## s23   0.86744728
## s24   0.90494716
## s25   0.88487343
## s26   0.89132384
## s27   0.88452877
## s28   0.87474317
## s29   0.84007708
## s30   0.84836216
## s31   0.81621715
## s32   0.80453581
## s33   0.77721433
## s34   0.75300952
## s35   0.75537623
## s36   0.76315003
## s37   0.73484908
## s38   0.77017759
## s39   0.75284312
## s40   0.77314193
## s41   0.76097170
## s42   0.75494973
## s43   0.74867236
## s44   0.73572793
## s45   0.74546481
## s46   0.73905106
## s47   0.73489707
## s48   0.73170982
## s49   0.69709350
## s50   0.66948089
## s51   0.67937303
## s52   0.68270285
## s53   0.67791221
## s54   0.64896922
## s55   0.64712609
## s56   0.68325800
```

```
## s57   0.65657232
## s58   0.68519547
## s59   0.67701427
## s60   0.66640077
## s61   0.68175853
## s62   0.65679055
## s63   0.64994281
## s64   0.64441882
## s65   0.65886716
## s66   0.63677405
## s67   0.63782854
## s68   0.68831197
## s69   0.71350756
## s70   0.77101239
## s71   0.71031354
## s72   0.89032230
## s73   1.06224020
## s74   1.05999225
## s75   1.04506720
## s76   1.07267352
## s77   1.05698223
## s78   1.04261963
## s79   1.02974275
## s80   1.02728976
## s81   1.01679944
## s82   1.02242282
## s83   1.01971873
## s84   1.01438858
## s85   1.01076416
## s86   1.03208371
## s87   1.02084583
## s88   1.02890160
## s89   1.01991634
## s90   1.02477726
## s91   1.00753552
## s92   1.01712763
## s93   1.01032168
## s94   1.01136633
## s95   0.99887246
## s96   1.01120495
## s97   0.99642791
## s98   0.99405916
## s99   0.98156467
## s100  0.96426783
## s101  0.95689991
## s102  0.95488375
## s103  0.96269727
## s104  0.97513208
## s105  0.97007724
## s106  0.95714179
## s107  0.95279501
## s108  0.93194438
## s109  0.93053756
## s110  0.92089876
```

```
## s111  0.93279094
## s112  0.92819755
## s113  0.93203947
## s114  0.93616531
## s115  0.91824919
## s116  0.91908324
## s117  0.92030422
## s118  0.92309896
## s119  0.91473180
## s120  0.91751665
## s121  0.91880454
## s122  0.93654964
## s123  0.92635107
## s124  0.92224726
## s125  0.91533735
## s126  0.90337284
## s127  0.90574218
## s128  0.90652093
## s129  0.91714247
## s130  0.93934344
## s131  0.95214484
## s132  0.95368842
## s133  0.95377925
## s134  0.95321667
## s135  0.96329747
## s136  0.95683565
## s137  0.95287732
## s138  0.94511467
## s139  0.95320238
## s140  0.95210194
## s141  0.95723699
## s142  0.96432194
## s143  0.97054749
## s144  0.96709881
## s145  0.96853910
## s146  0.97125125
## s147  0.97323389
## s148  0.97816041
## s149  0.96918887
## s150  0.96229932
## s151  0.94929447
## s152  0.93254247
## s153  0.92034424
## s154  0.91237528
## s155  0.91388245
## s156  0.89977715
## s157  0.89499109
## s158  0.90411595
## s159  0.89445155
## s160  0.90305371
## s161  0.91695497
## s162  0.91548230
## s163  0.91914357
## s164  0.92343171
```

```
## s165  0.92626288
## s166  0.94300953
## s167  0.95210609
## s168  0.94644727
## s169  0.94028147
## s170  0.95772876
## s171  0.95422024
## s172  0.96294303
## s173  0.95825204
## s174  0.94869227
## s175  0.95583431
## s176  0.95422556
## s177  0.96116875
## s178  0.97974236
## s179  0.98188475
## s180  0.98299813
## s181  0.99037548
## s182  0.98945059
## s183  0.98843243
## s184  0.99791801
## s185  1.00958339
## s186  1.01708678
## s187  1.01837973
## s188  1.01288348
## s189  1.01855139
## s190  1.01086538
## s191  1.01418639
## s192  1.01442172
## s193  1.01847674
## s194  1.04044762
## s195  1.02858704
## s196  1.01009369
## s197  1.01001906
## s198  1.01825875
## s199  1.00631412
## s200  0.99781365
## s201  0.97893605
## s202  0.97967995
## s203  0.98627364
## s204  0.98131975
## s205  0.99429877
## s206  1.00510852
## s207  0.99604166
## s208  0.99971877
## s209  0.99940326
## s210  0.99858165
## s211  1.00790599
## s212  1.01720170
## s213  1.02157543
## s214  1.02047037
## s215  1.00737185
## s216  1.00414709
## s217  1.01733653
## s218  1.02712819
```

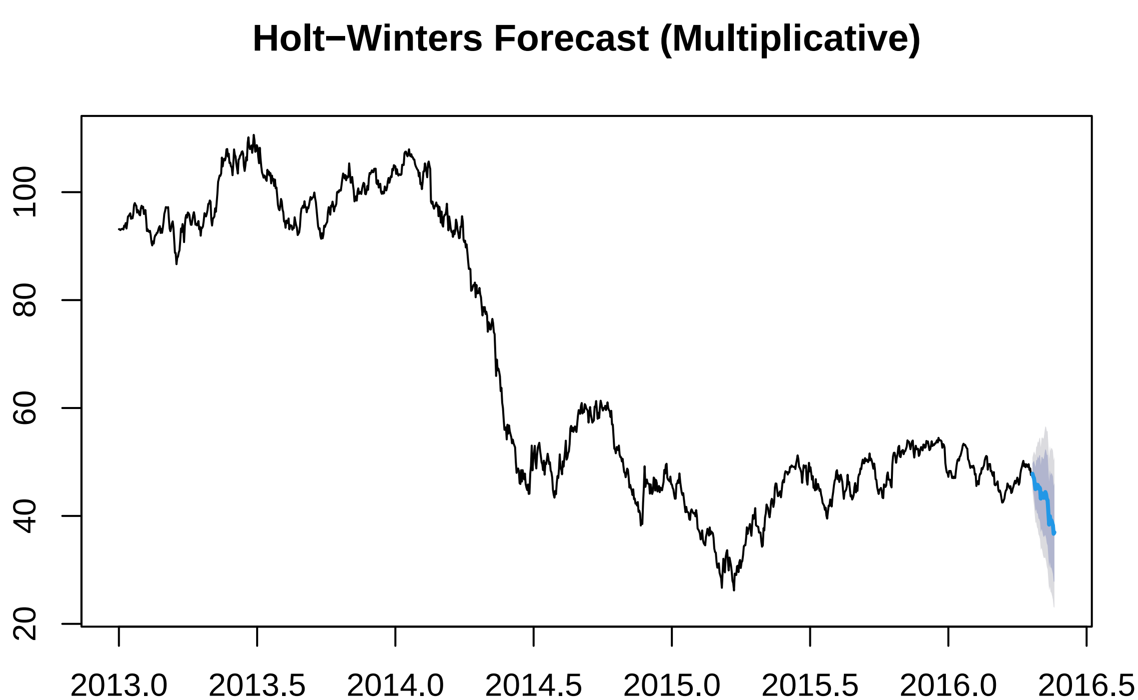
```
## s219  1.02258374
## s220  1.03634940
## s221  1.05237170
## s222  1.05398029
## s223  1.05625784
## s224  1.06153305
## s225  1.06070288
## s226  1.06135585
## s227  1.07075821
## s228  1.07101763
## s229  1.07205988
## s230  1.04943376
## s231  1.04956536
## s232  1.05608751
## s233  1.04371355
## s234  1.04933932
## s235  1.05437320
## s236  1.04081688
## s237  1.03994472
## s238  1.04531644
## s239  1.04260126
## s240  1.04648571
## s241  1.05901829
## s242  1.05740009
## s243  1.05697822
## s244  1.06469261
## s245  1.07922937
## s246  1.08562452
## s247  1.08041747
## s248  1.08784801
## s249  1.09334013
## s250  1.09594322
## s251  1.11101361
## s252  1.11288049
## s253  1.12570687
## s254  1.12584303
## s255  1.12103491
## s256  1.11546450
## s257  1.12491818
## s258  1.11833115
## s259  1.11603659
## s260  1.12082308
## s261  1.12572972
## s262  1.12362325
## s263  1.12756923
## s264  1.14566596
## s265  1.15182967
## s266  1.15075077
## s267  1.17435506
## s268  1.18056068
## s269  1.18317066
## s270  1.18127849
## s271  1.17746113
## s272  1.18385093
```

```
## s273  1.19517672
## s274  1.18884898
## s275  1.18554302
## s276  1.19176327
## s277  1.18849208
## s278  1.18824884
## s279  1.18730393
## s280  1.19104454
## s281  1.18098665
## s282  1.17868953
## s283  1.17697546
## s284  1.17576825
## s285  1.17748083
## s286  1.16633969
## s287  1.17606661
## s288  1.15442769
## s289  1.15451130
## s290  1.14575706
## s291  1.16160543
## s292  1.18236793
## s293  1.18783782
## s294  1.20744646
## s295  1.20223113
## s296  1.19368618
## s297  1.18391918
## s298  1.21062283
## s299  1.22036698
## s300  1.21543042
## s301  1.20936587
## s302  1.14537099
## s303  1.13679687
## s304  1.14128112
## s305  1.13548337
## s306  1.12825506
## s307  1.13337473
## s308  1.14048980
## s309  1.14640555
## s310  1.13944305
## s311  1.13951461
## s312  1.12198233
## s313  1.14096436
## s314  1.13261114
## s315  1.11250047
## s316  1.13275556
## s317  1.11277850
## s318  1.10504784
## s319  1.12700488
## s320  1.13441733
## s321  1.13255450
## s322  1.14919895
## s323  1.16715762
## s324  1.13587336
## s325  1.10855478
## s326  1.14106275
```

```
## s327  1.13253118
## s328  1.11975621
## s329  1.10932986
## s330  1.11175611
## s331  1.10715039
## s332  1.11685988
## s333  1.11201686
## s334  1.11845494
## s335  1.14198310
## s336  1.14083687
## s337  1.12697188
## s338  1.12496076
## s339  1.11176826
## s340  1.11333087
## s341  1.13992055
## s342  1.14107307
## s343  1.16218845
## s344  1.15700579
## s345  1.11879850
## s346  1.10972861
## s347  1.11595231
## s348  1.10180157
## s349  1.11083395
## s350  1.09806661
## s351  1.07758406
## s352  1.05929939
## s353  1.06209975
## s354  1.06245331
## s355  1.01325298
## s356  1.01528261
## s357  1.02087697
## s358  1.02952179
## s359  1.02824993
## s360  1.03546386
## s361  1.00985493
## s362  1.03401143
## s363  1.01859852
## s364  1.01861778
## s365  1.02349757
##
## Error measures:
##                      ME     RMSE       MAE        MPE     MAPE      MASE
## Training set 0.01388818 1.517334 0.8827826 0.05194969 1.912539 0.0318355
##                    ACF1
## Training set 0.09034836
##
## Forecasts:
##           Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2016.3041       47.82643 45.88082 49.77205 44.85087 50.80200
## 2016.3068       47.30885 44.62367 49.99402 43.20222 51.41547
## 2016.3096       46.97901 43.71892 50.23911 41.99313 51.96490
## 2016.3123       45.96857 42.26509 49.67205 40.30459 51.63255
## 2016.3151       44.96214 40.87873 49.04554 38.71710 51.20717
## 2016.3178       45.83446 41.26577 50.40316 38.84725 52.82168
```

```
## 2016.3205       45.39231 40.48836 50.29625 37.89237 52.89224
## 2016.3233       45.74721 40.45485 51.03957 37.65324 53.84117
## 2016.3260       45.06516 39.51642 50.61390 36.57909 53.55123
## 2016.3288       45.33247 39.43604 51.22890 36.31465 54.35029
## 2016.3315       45.09310 38.92580 51.26040 35.66102 54.52517
## 2016.3342       43.21133 36.99963 49.42303 33.71135 52.71130
## 2016.3370       44.25837 37.61606 50.90069 34.09983 54.41691
## 2016.3397       44.06028 37.17547 50.94509 33.53088 54.58969
## 2016.3425       43.47065 36.41156 50.52975 32.67470 54.26661
## 2016.3452       43.35549 36.05711 50.65387 32.19359 54.51739
## 2016.3479       43.95512 36.30908 51.60115 32.26152 55.64872
## 2016.3507       44.43248 36.46540 52.39956 32.24788 56.61708
## 2016.3534       44.02516 35.89647 52.15385 31.59340 56.45692
## 2016.3562       43.12797 34.93154 51.32441 30.59260 55.66334
## 2016.3589       42.92308 34.53730 51.30887 30.09813 55.74803
## 2016.3616       40.67962 32.49716 48.86209 28.16562 53.19362
## 2016.3644       38.39029 30.42746 46.35312 26.21219 50.56839
## 2016.3671       39.96251 31.44979 48.47523 26.94343 52.98159
## 2016.3699       38.99059 30.45913 47.52205 25.94285 52.03834
## 2016.3726       39.18874 30.39488 47.98260 25.73969 52.63779
## 2016.3753       38.80456 29.88001 47.72910 25.15564 52.45347
## 2016.3781       38.29078 29.26930 47.31226 24.49362 52.08794
## 2016.3808       36.69218 27.82804 45.55633 23.13565 50.24872
## 2016.3836       36.97212 27.82827 46.11597 22.98780 50.95644
```

```r
plot(forecast_multiplicative, main = "Holt-Winters Forecast (Multiplicative)")
```

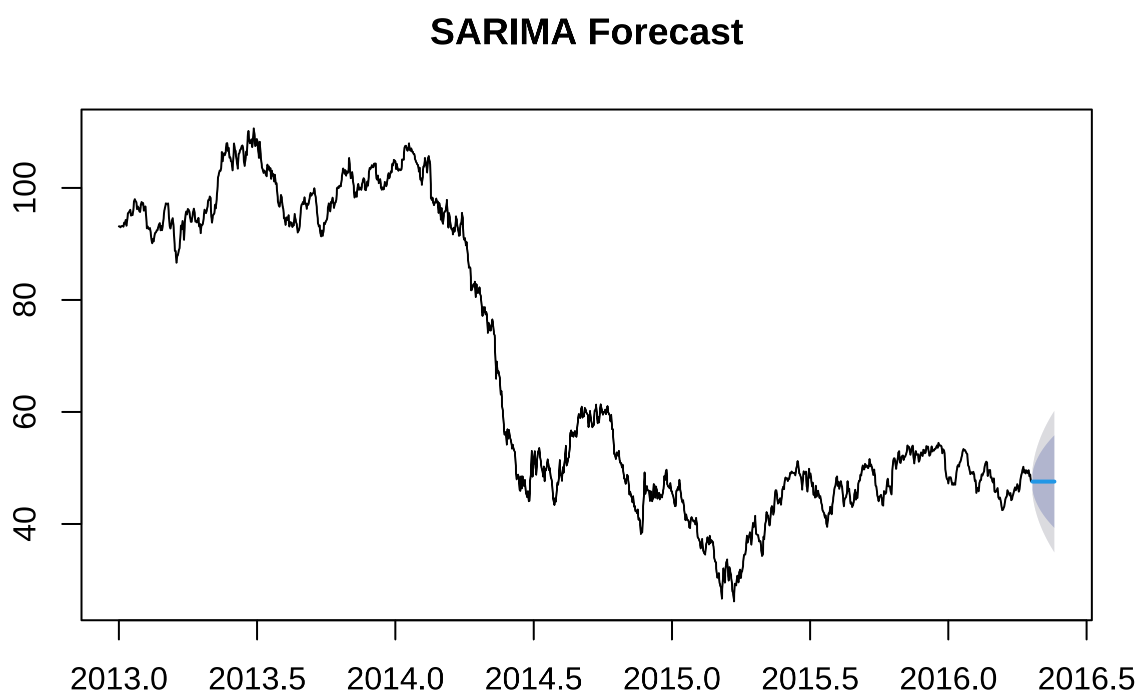


SARIMA Model

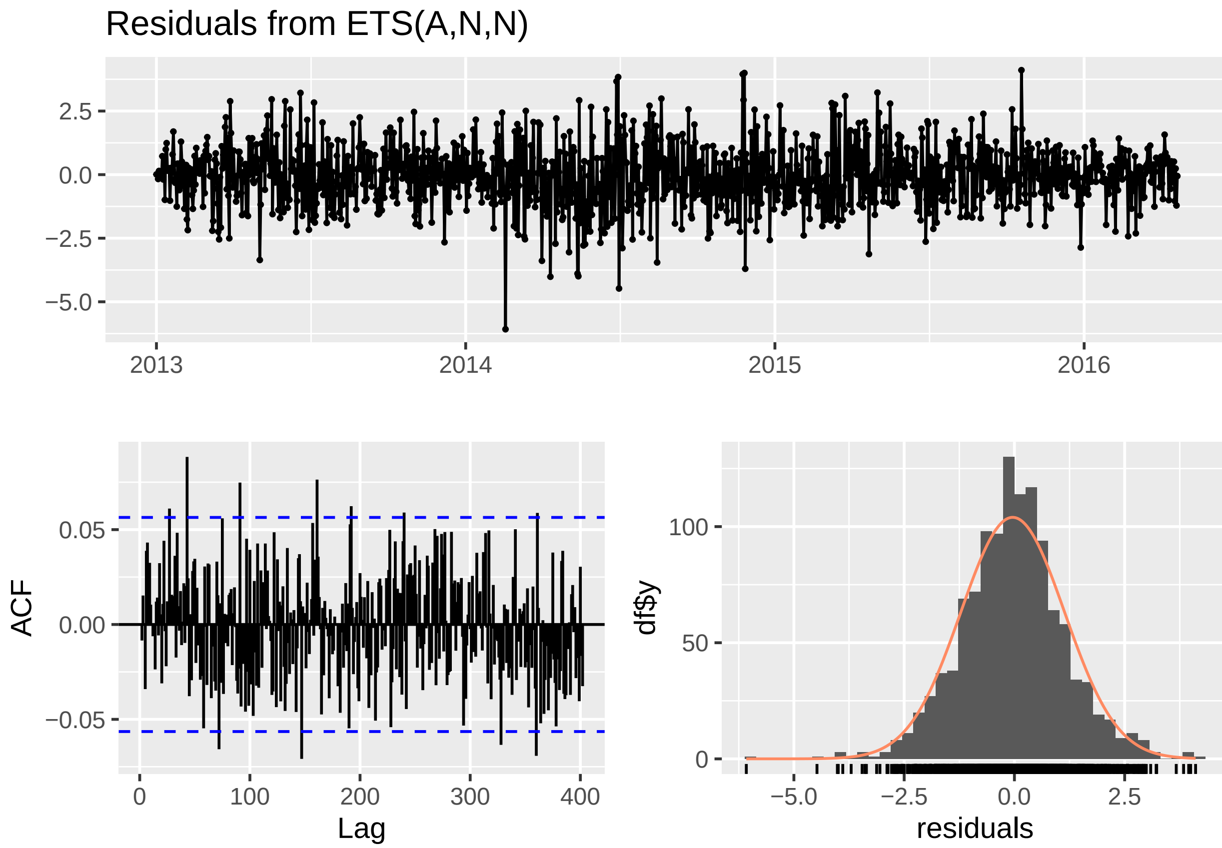
```
forecast_sarima <- forecast(auto.arima(train_ts, seasonal = TRUE), h = 30)
summary(forecast_sarima)
```

```
##
## Forecast method: ARIMA(0,1,0)
##
## Model Information:
## Series: train_ts
## ARIMA(0,1,0)
##
## sigma^2 = 1.391:  log likelihood = -1908.73
## AIC=3819.47   AICc=3819.47   BIC=3824.56
##
## Error measures:
##                        ME     RMSE       MAE         MPE     MAPE       MASE
## Training set -0.03770884 1.178992 0.8958898 -0.07959036 1.546013 0.03230818
##                      ACF1
## Training set -0.02965348
##
## Forecasts:
##           Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2016.3041          47.57 46.05843 49.08157 45.25826 49.88174
## 2016.3068          47.57 45.43232 49.70768 44.30071 50.83929
## 2016.3096          47.57 44.95189 50.18811 43.56595 51.57405
## 2016.3123          47.57 44.54687 50.59313 42.94652 52.19348
## 2016.3151          47.57 44.19004 50.94996 42.40079 52.73921
## 2016.3178          47.57 43.86744 51.27256 41.90742 53.23258
## 2016.3205          47.57 43.57077 51.56923 41.45371 53.68629
## 2016.3233          47.57 43.29465 51.84535 41.03141 54.10859
## 2016.3260          47.57 43.03530 52.10470 40.63478 54.50522
## 2016.3288          47.57 42.79001 52.34999 40.25964 54.88036
## 2016.3315          47.57 42.55670 52.58330 39.90282 55.23718
## 2016.3342          47.57 42.33378 52.80622 39.56190 55.57810
## 2016.3370          47.57 42.11997 53.02003 39.23490 55.90510
## 2016.3397          47.57 41.91424 53.22576 38.92026 56.21974
## 2016.3425          47.57 41.71573 53.42427 38.61667 56.52333
## 2016.3452          47.57 41.52374 53.61626 38.32304 56.81696
## 2016.3479          47.57 41.33765 53.80235 38.03845 57.10155
## 2016.3507          47.57 41.15697 53.98303 37.76212 57.37788
## 2016.3534          47.57 40.98124 54.15876 37.49336 57.64664
## 2016.3562          47.57 40.81007 54.32993 37.23158 57.90842
## 2016.3589          47.57 40.64314 54.49686 36.97627 58.16373
## 2016.3616          47.57 40.48013 54.65987 36.72698 58.41302
## 2016.3644          47.57 40.32079 54.81921 36.48328 58.65672
## 2016.3671          47.57 40.16487 54.97513 36.24483 58.89517
## 2016.3699          47.57 40.01217 55.12783 36.01130 59.12870
## 2016.3726          47.57 39.86250 55.27750 35.78239 59.35761
## 2016.3753          47.57 39.71567 55.42433 35.55784 59.58216
## 2016.3781          47.57 39.57155 55.56845 35.33742 59.80258
## 2016.3808          47.57 39.42997 55.71003 35.12090 60.01910
## 2016.3836          47.57 39.29081 55.84919 34.90808 60.23192
```

```r
plot(forecast_sarima, main = "SARIMA Forecast")
```



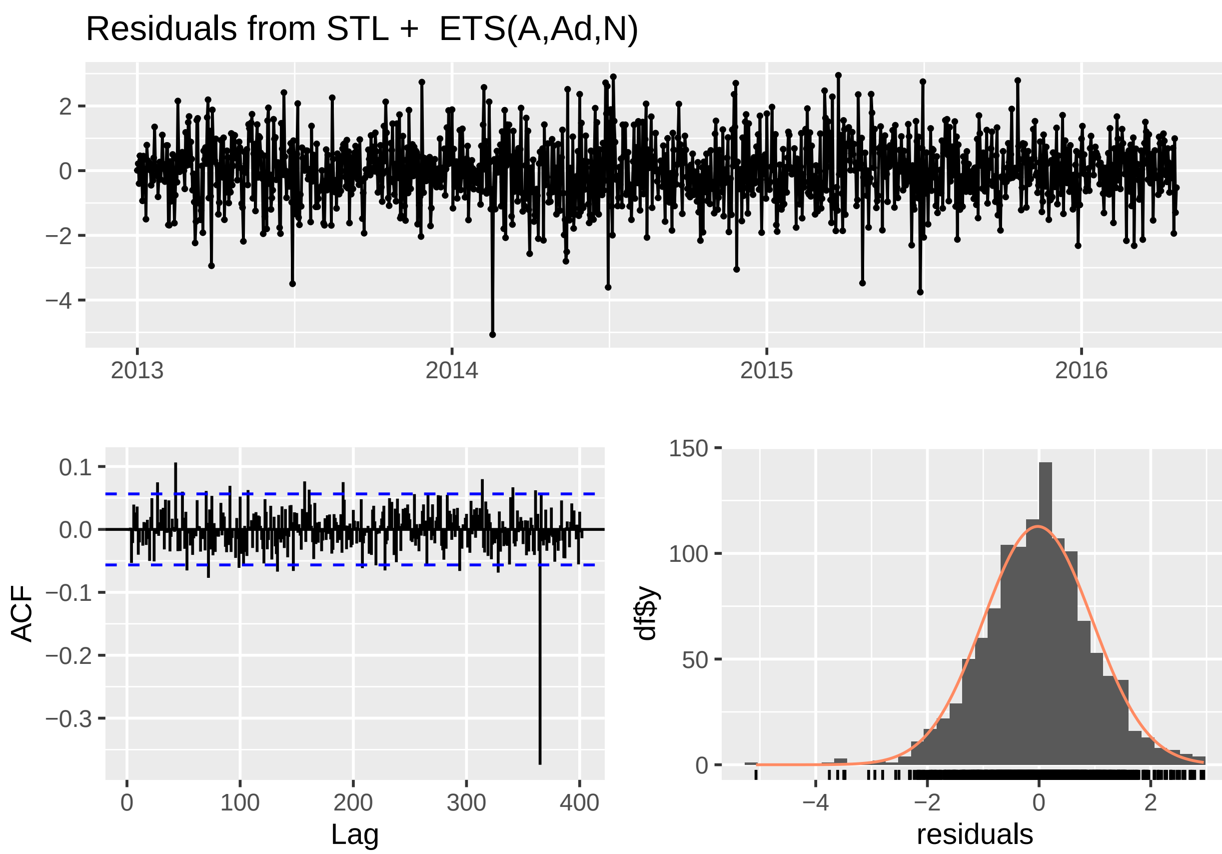

```r
checkresiduals(ets_forecast)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 255.38, df = 241, p-value = 0.2506
##
## Model df: 0.   Total lags used: 241
```

By performing Ljung-Box test for ETS model we can observe that the residuals are uncorrelated ,since the p-value(0.2506) > 0.05. The ETS(A, N, N) captures the time series structure well.

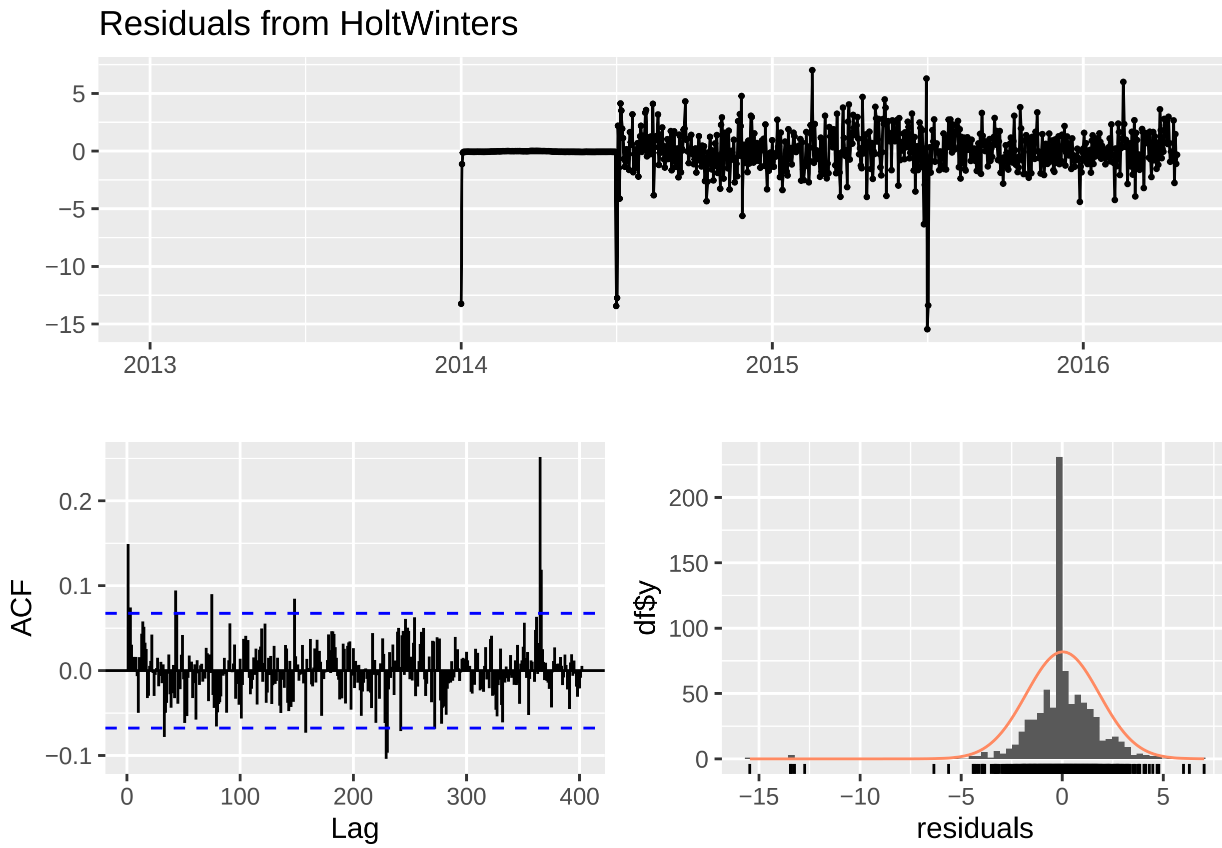
```r
checkresiduals(stl_forecast)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ETS(A,Ad,N)
## Q* = 321.39, df = 241, p-value = 0.0004058
##
## Model df: 0.   Total lags used: 241
```

By performing Ljung-Box test for stl model we can observe that the residuals are autocorrelated, since the p-value(0.0004) < 0.05. The STL + ETS(A, Ad, N) does not captures the time series structure well.

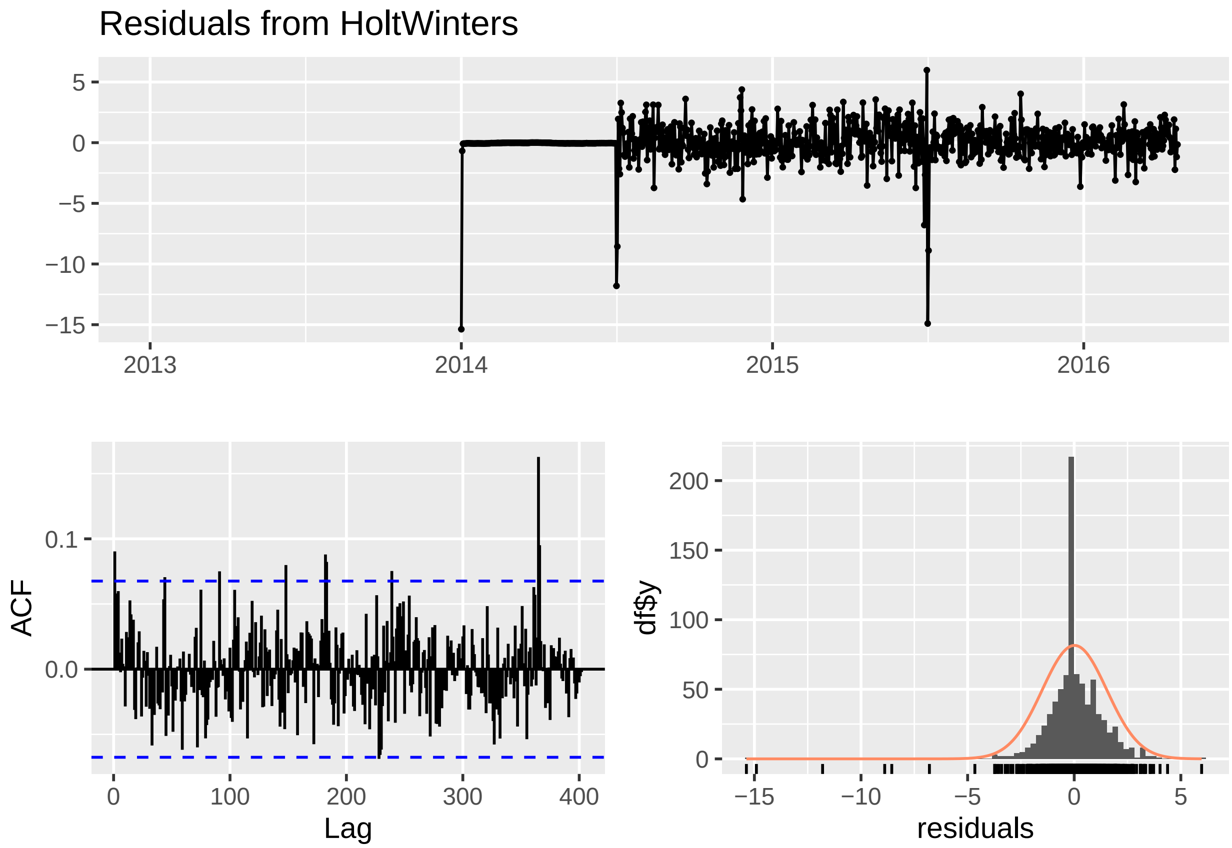
```
checkresiduals(forecast_additive)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from HoltWinters
## Q* = 258.43, df = 241, p-value = 0.2102
##
## Model df: 0.   Total lags used: 241
```

The residuals are uncorrelated, indicating this Holt-Winters model (second variant) is a good fit for the data.

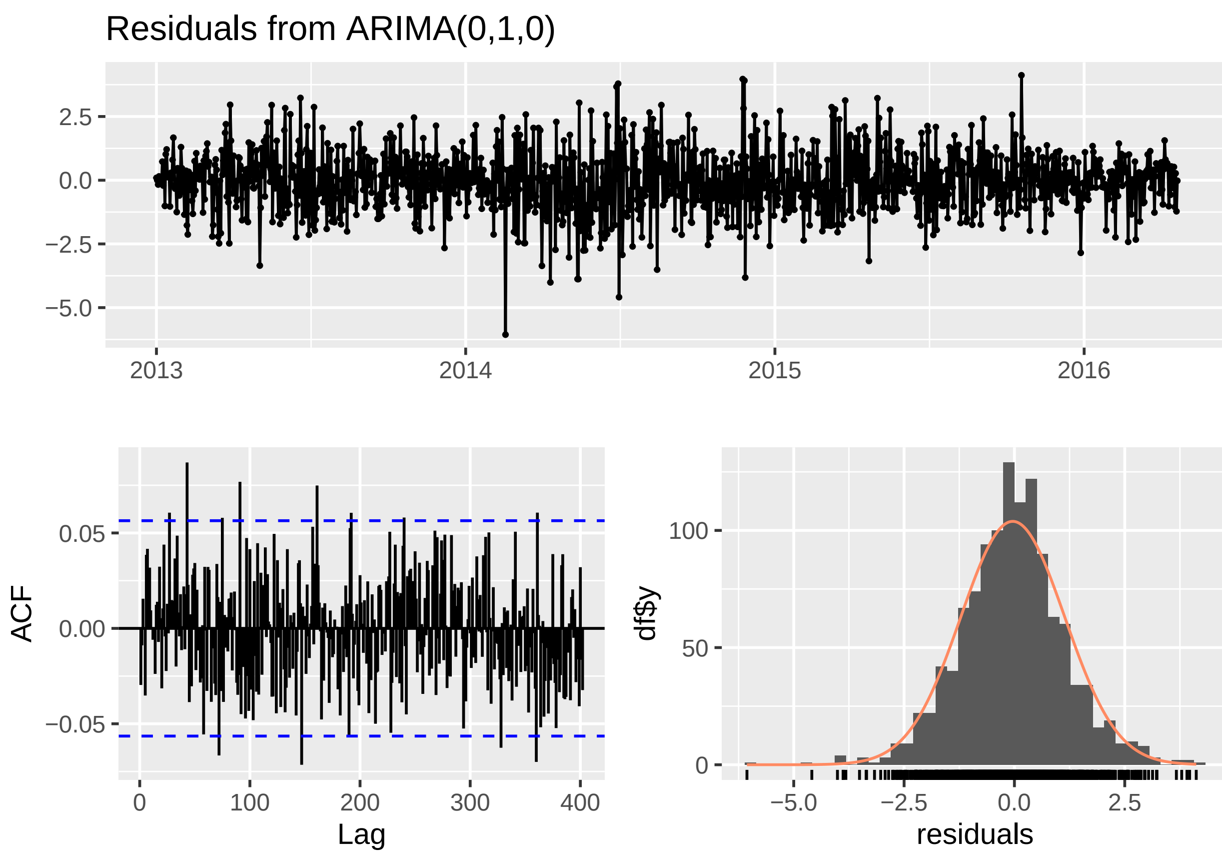
```
checkresiduals(forecast_multiplicative)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from HoltWinters
## Q* = 215.6, df = 241, p-value = 0.8789
##
## Model df: 0.   Total lags used: 241
```

By performing Ljung-Box test for Holt-Winters models both additive and multiplicative we can observe that the residuals are uncorrelated, since the p-value > 0.05. The model captures the time series structures well.

```r
checkresiduals(forecast_sarima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)
## Q* = 258, df = 241, p-value = 0.2157
##
## Model df: 0.   Total lags used: 241
```

By performing Ljung-Box test for SARIMA model we can observe that the residuals are uncorrelated, since the p-value > 0.05. The model captures the time series structures well.

8. Comparison of models' performance

```
ets_accuracy <- accuracy(ets_forecast, test$dcoilwtico)
stl_accuracy <- accuracy(stl_forecast, test$dcoilwtico)
hw_additive_accuracy <- accuracy(forecast_additive, test$dcoilwtico)
hw_multiplicative_accuracy <- accuracy(forecast_multiplicative, test$dcoilwtico)
sarima_accuracy <- accuracy(forecast_sarima, test$dcoilwtico)

cat("Model Metrics Comparison:\n")
```

```
## Model Metrics Comparison:
```

```r
cat("\nETS Model Metrics: ",
    "RMSE:", ets_accuracy["Test set", "RMSE"],
    "| MAE:", ets_accuracy["Test set", "MAE"],
    "| MAPE:", ets_accuracy["Test set", "MAPE"], "\n")
```

```
##
## ETS Model Metrics:  RMSE: 0.8222896 | MAE: 0.6705481 | MAPE: 1.426778
```

```r
cat("STL Model Metrics: ",
    "RMSE:", stl_accuracy["Test set", "RMSE"],
    "| MAE:", stl_accuracy["Test set", "MAE"],
    "| MAPE:", stl_accuracy["Test set", "MAPE"], "\n")
```

```
## STL Model Metrics:  RMSE: 1.015983 | MAE: 0.82589 | MAPE: 1.746843
```

```r
cat("Holt-Winters Additive Metrics: ",
    "RMSE:", hw_additive_accuracy["Test set", "RMSE"],
    "| MAE:", hw_additive_accuracy["Test set", "MAE"],
    "| MAPE:", hw_additive_accuracy["Test set", "MAPE"], "\n")
```

```
## Holt-Winters Additive Metrics:  RMSE: 2.988354 | MAE: 2.611734 | MAPE: 5.517892
```

```r
cat("Holt-Winters Multiplicative Metrics: ",
    "RMSE:", hw_multiplicative_accuracy["Test set", "RMSE"],
    "| MAE:", hw_multiplicative_accuracy["Test set", "MAE"],
    "| MAPE:", hw_multiplicative_accuracy["Test set", "MAPE"], "\n")
```

```
## Holt-Winters Multiplicative Metrics:  RMSE: 1.979541 | MAE: 1.727461 | MAPE: 3.643008
```

```r
cat("SARIMA Model Metrics: ",
    "RMSE:", sarima_accuracy["Test set", "RMSE"],
    "| MAE:", sarima_accuracy["Test set", "MAE"],
    "| MAPE:", sarima_accuracy["Test set", "MAPE"], "\n")
```

```
## SARIMA Model Metrics:  RMSE: 0.8216345 | MAE: 0.67 | MAPE: 1.425576
```

Model with a low RMSE

Observations: ETS and SARIMA models perform best, with the lowest RMSE values of 0.8222896 and 0.8216345 respectively which is approximately ~0.822. But SARIMA performs slightly better compared to ETS in terms of RMSE.

The RMSE values are high for both Holt-Winters Additive and Multiplicative models indicating that it is not suitable for this dataset. STL performs better than Holt-winters.