Pre-Covid Forecasting

Library

```r
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 4.0.5

## -- Attaching packages -------------------------------------------- fpp3 0.4.0 --

## v tibble      3.1.4      v tsibble     1.0.1
## v dplyr       1.0.7      v tsibbledata 0.3.0
## v tidyr       1.1.4      v feasts      0.2.2
## v lubridate   1.7.10     v fable       0.3.1
## v ggplot2     3.3.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'lubridate' was built under R version 4.0.5

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tsibble' was built under R version 4.0.5

## Warning: package 'tsibbledata' was built under R version 4.0.5

## Warning: package 'feasts' was built under R version 4.0.5

## Warning: package 'fabletools' was built under R version 4.0.5

## Warning: package 'fable' was built under R version 4.0.5

## -- Conflicts ------------------------------------------------- fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```r
library(TTR)
```

```
## Warning: package 'TTR' was built under R version 4.0.5
```

```r
library(ggplot2)
library(tsibble)
library(tsibbledata)
library(dplyr)
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.0.5
```

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

```r
library(fpp)
```

```
## Warning: package 'fpp' was built under R version 4.0.5
```

```
## Loading required package: fma
```

```
## Warning: package 'fma' was built under R version 4.0.5
```

```
## Loading required package: expsmooth
```

```
## Warning: package 'expsmooth' was built under R version 4.0.5
```

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

```
## Warning: package 'lmtest' was built under R version 4.0.5
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.0.5
```

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

```
## The following object is masked from 'package:tsibble':
##
##     index
```

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

```
## Loading required package: tseries
```

```
## Warning: package 'tseries' was built under R version 4.0.5

##
## Attaching package: 'fpp'

## The following object is masked from 'package:fpp3':
##
##     insurance
```

```r
library(fpp2)
```

```
## Warning: package 'fpp2' was built under R version 4.0.5

##
## Attaching package: 'fpp2'

## The following objects are masked from 'package:fpp':
##
##     ausair, ausbeer, austa, austourists, debitcards, departures,
##     elecequip, euretail, guinearice, oil, sunspotarea, usmelec

## The following object is masked from 'package:fpp3':
##
##     insurance
```

```r
library(bsts)
```

```
## Warning: package 'bsts' was built under R version 4.0.5

## Loading required package: BoomSpikeSlab

## Warning: package 'BoomSpikeSlab' was built under R version 4.0.5

## Loading required package: Boom

## Warning: package 'Boom' was built under R version 4.0.5

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following objects are masked from 'package:fma':
##
##     cement, housing, petrol

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

```
##
## Attaching package: 'Boom'

## The following object is masked from 'package:stats':
##
##     rWishart

##
## Attaching package: 'BoomSpikeSlab'

## The following object is masked from 'package:stats':
##
##     knots

## Loading required package: xts

## Warning: package 'xts' was built under R version 4.0.5

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

##
## Attaching package: 'bsts'

## The following object is masked from 'package:BoomSpikeSlab':
##
##     SuggestBurn
```

```r
library(prophet)
```

```
## Warning: package 'prophet' was built under R version 4.0.5

## Loading required package: Rcpp

## Warning: package 'Rcpp' was built under R version 4.0.5

## Loading required package: rlang

## Warning: package 'rlang' was built under R version 4.0.5
```

```r
library(repr)
```

```
## Warning: package 'repr' was built under R version 4.0.5
```

```r
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.0.5
```

```r
Tng_Ctr_Hour <- read_excel("C:/Users/prach/Desktop/Rutgers/BF/Project/Tng_Ctr_Hour.xlsx")
View(Tng_Ctr_Hour)
summary(Tng_Ctr_Hour)
```

```
##      Year              Quarter             Month             Device_Hrs
##  Length:81          Length:81          Length:81          Min.   : 222.8
##  Class :character   Class :character   Class :character   1st Qu.: 899.0
##  Mode  :character   Mode  :character   Mode  :character   Median :1008.0
##                                                           Mean   : 990.1
##                                                           3rd Qu.:1101.7
##                                                           Max.   :1519.9
##  DH_Prev_Year       DH_YoY_Change      DH_YoY_Ch_Per      Total_Inst_Hrs
##  Length:81          Length:81          Length:81          Min.   : 504.6
##  Class :character   Class :character   Class :character   1st Qu.:1937.3
##  Mode  :character   Mode  :character   Mode  :character   Median :2203.2
##                                                           Mean   :2165.7
##                                                           3rd Qu.:2446.8
##                                                           Max.   :3084.1
##  Total_Inst_Hrs_Prev_Year Inst_Hrs_YoY_Change Total_Inst_Hrs_YoY_Change_Per2
##  Length:81                Length:81           Length:81
##  Class :character         Class :character    Class :character
##  Mode  :character         Mode  :character    Mode  :character
##
##
##
```

Converting Data Frame to Time Series

Converting the Dataset into training and testing set.

```r
df_Tng = Tng_Ctr_Hour[,c(4)]
df_Tng
```

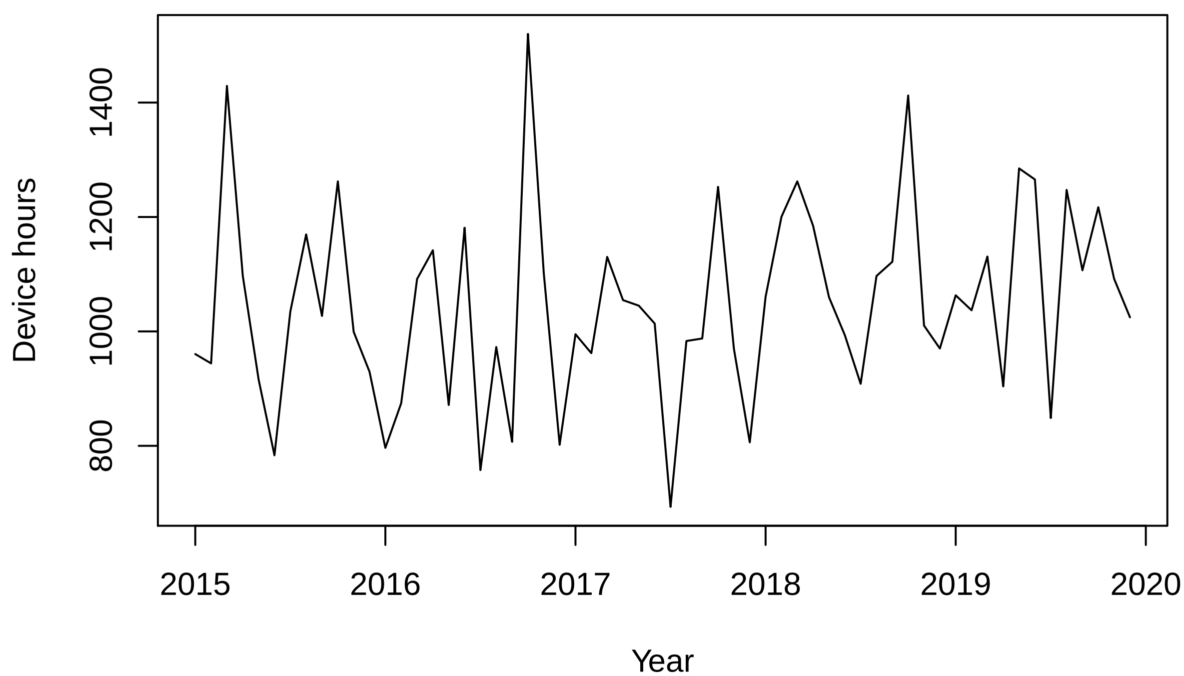
```
## # A tibble: 81 x 1
##    Device_Hrs
##         <dbl>
##  1       960.
##  2       944.
##  3      1429.
##  4      1097
##  5       916.
##  6       783.
##  7      1035.
##  8      1170.
##  9      1027.
## 10      1262.
## # ... with 71 more rows
```

```r
train_tng = ts(data = df_Tng,frequency = 12,start = c(2015, 1),end = c(2019,12))
train_tng
```

```
##          Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep
## 2015  960.42  944.08 1429.12 1097.00  915.85  783.45 1034.52 1169.50 1027.08
## 2016  796.42  874.55 1091.55 1141.84  871.36 1181.21  757.59  972.73  807.02
## 2017  995.09  962.00 1130.24 1054.71 1044.95 1013.73  693.33  983.25  987.64
## 2018 1060.57 1200.25 1262.25 1184.45 1059.92  993.55  908.37 1096.93 1121.75
## 2019 1063.13 1036.95 1130.87  903.97 1284.95 1265.56  848.64 1247.40 1106.84
##          Oct     Nov     Dec
## 2015 1262.32  999.25  929.42
## 2016 1519.92 1101.67  801.83
## 2017 1252.69  969.31  806.10
## 2018 1412.47 1010.25  970.12
## 2019 1217.08 1091.84 1024.67
```

Plotting the time series

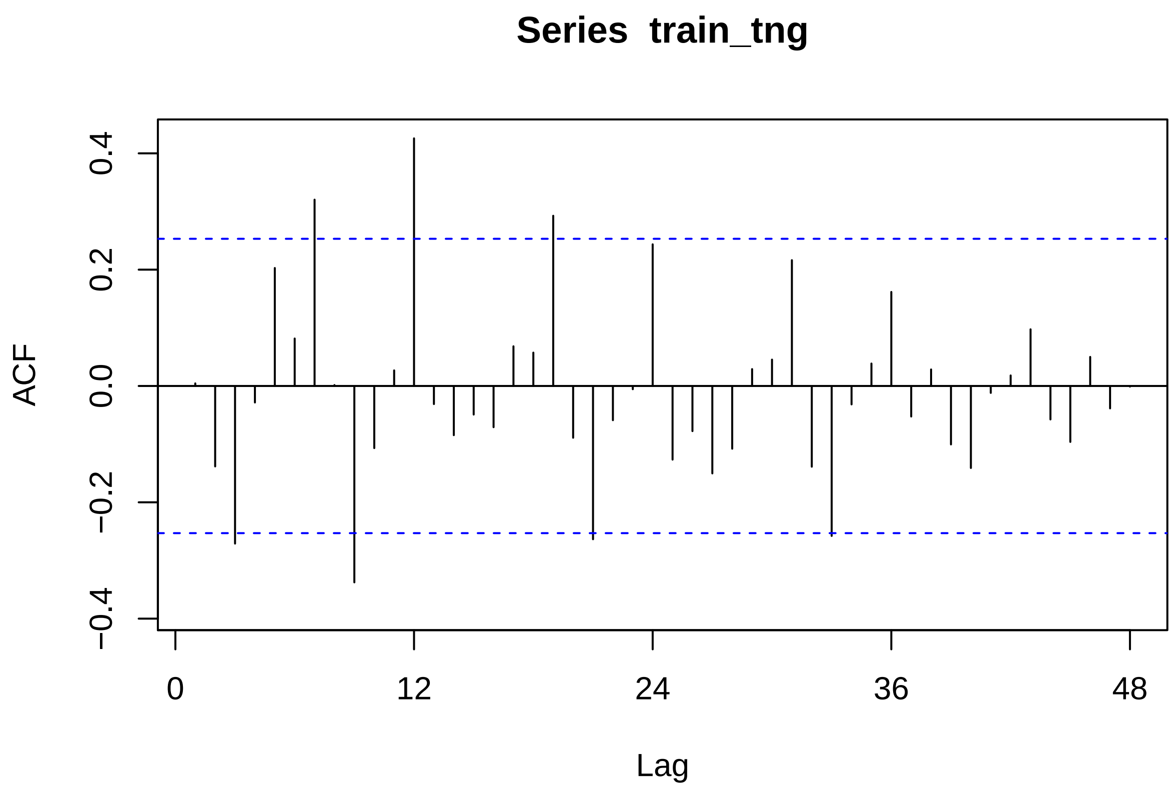
```r
plot(train_tng,xlab = "Year", ylab = "Device hours")
```



##We can notice in the plot that there is seasonilty and device hours are its peak mostly in the third quarter of every year.

Acf

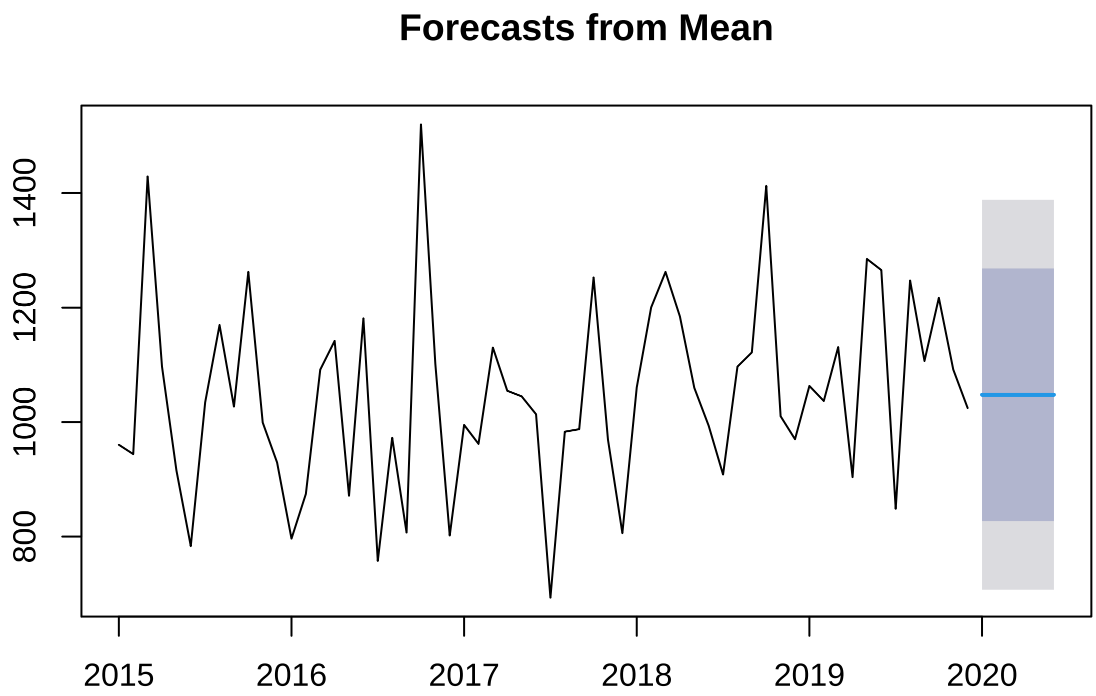
```r
Acf(train_tng, lag = 48)
```



Forecasting Methods

Mean Forecast

```r
mean_forecast = meanf(train_tng, h=6)
plot(mean_forecast)
```



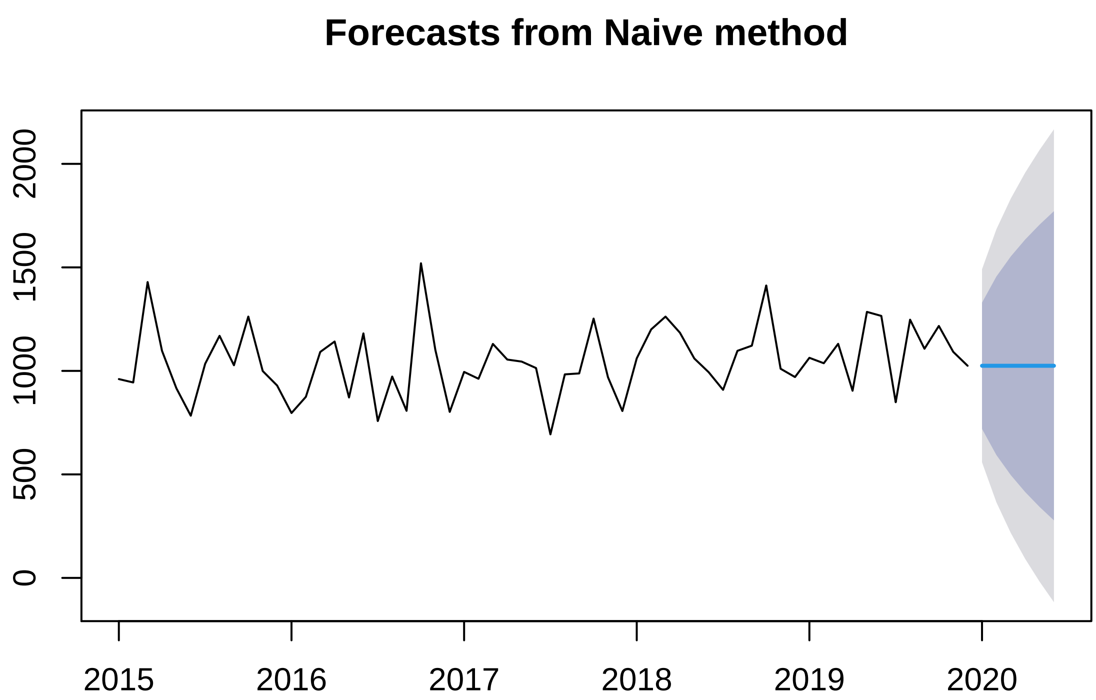

```r
summary(mean_forecast)
```

```
##
## Forecast method: Mean
##
## Model Information:
## $mu
## [1] 1047.759
##
## $mu.se
## [1] 21.79203
##
## $sd
## [1] 168.8003
##
## $bootstrap
## [1] FALSE
##
## $call
## meanf(y = train_tng, h = 6)
##
## attr(,"class")
## [1] "meanf"
##
## Error measures:
```

```
##                         ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 5.494447e-14 167.3878 130.0262 -2.590342 12.81213 0.9299448
##                      ACF1
## Training set 0.004503158
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95   Hi 95
## Jan 2020       1047.759 827.1667 1268.351 707.1869 1388.33
## Feb 2020       1047.759 827.1667 1268.351 707.1869 1388.33
## Mar 2020       1047.759 827.1667 1268.351 707.1869 1388.33
## Apr 2020       1047.759 827.1667 1268.351 707.1869 1388.33
## May 2020       1047.759 827.1667 1268.351 707.1869 1388.33
## Jun 2020       1047.759 827.1667 1268.351 707.1869 1388.33
```

Naive Forecast

```r
naive_forecast <- naive(train_tng,6)
plot(naive_forecast)
```



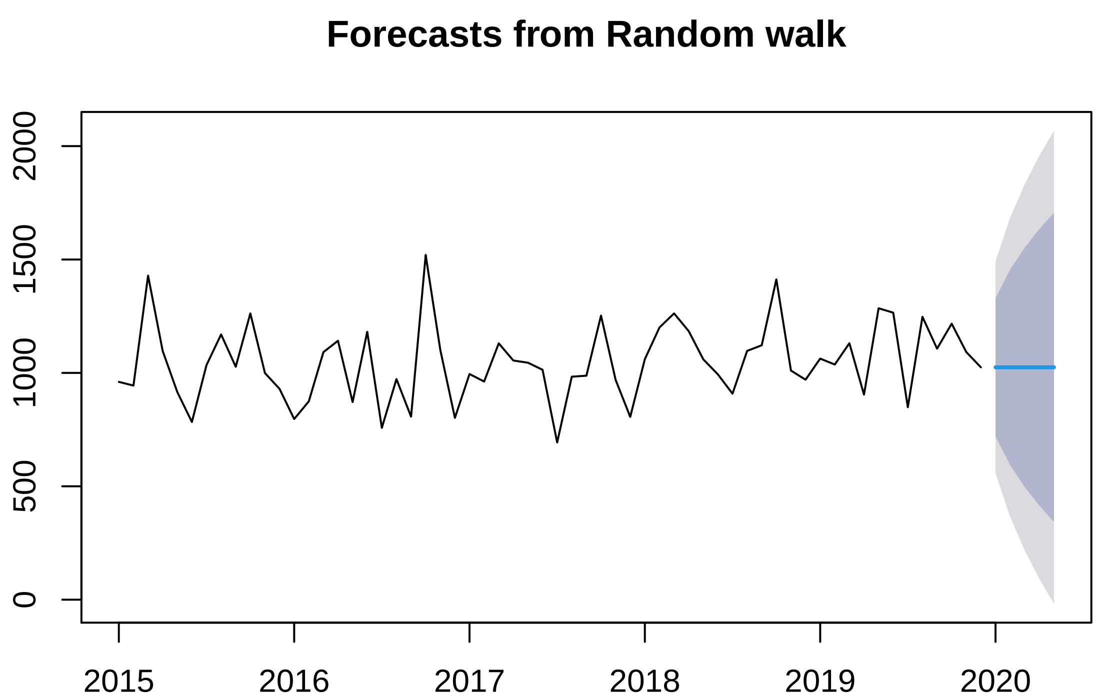

```r
summary(naive_forecast)
```

```
##
## Forecast method: Naive method
```

```
##
## Model Information:
## Call: naive(y = train_tng, h = 6)
##
## Residual sd: 237.8911
##
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE    MASE       ACF1
## Training set 1.088983 237.8911 190.3571 -2.425052 18.37449 1.36143 -0.4293306
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80      Lo 95    Hi 95
## Jan 2020        1024.67 719.8003 1329.540  558.41209 1490.928
## Feb 2020        1024.67 593.5192 1455.821  365.28174 1684.058
## Mar 2020        1024.67 496.6203 1552.720  217.08761 1832.252
## Apr 2020        1024.67 414.9307 1634.409   92.15418 1957.186
## May 2020        1024.67 342.9607 1706.379  -17.91439 2067.254
## Jun 2020        1024.67 277.8949 1771.445 -117.42397 2166.764
```

Random Walk Forecast

```
rwf_forecast = rwf(train_tng,5)
plot(rwf_forecast)
```

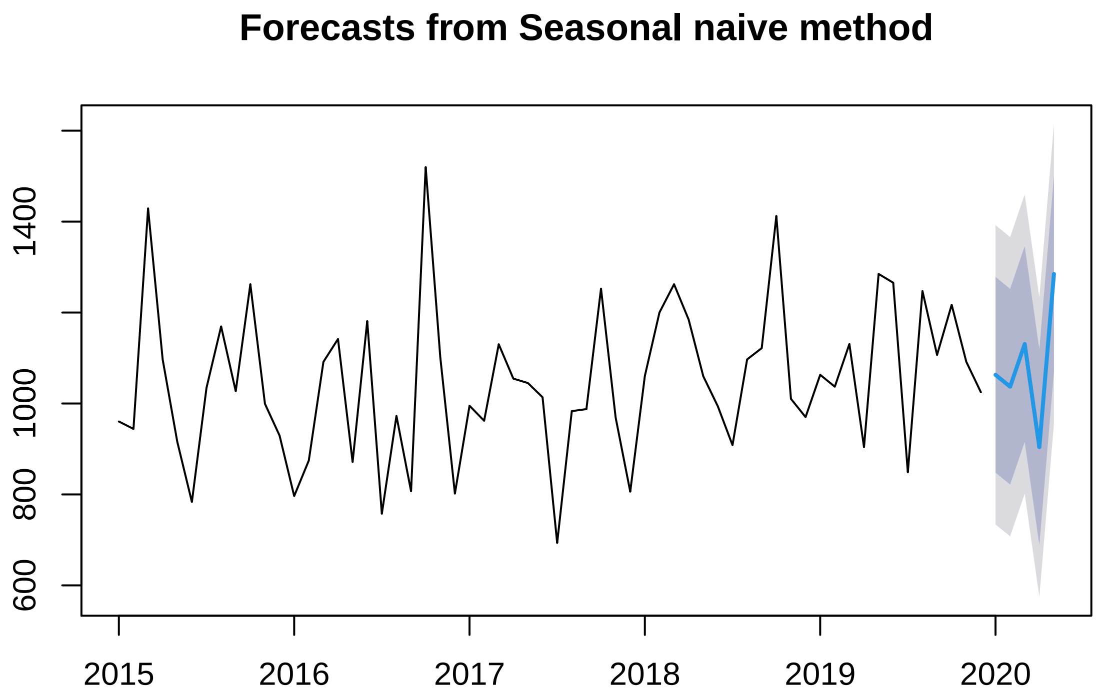


```r
summary(rwf_forecast)
```

```
##
## Forecast method: Random walk
##
## Model Information:
## Call: rwf(y = train_tng, h = 5)
##
## Residual sd: 237.8911
##
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE    MASE       ACF1
## Training set 1.088983 237.8911 190.3571 -2.425052 18.37449 1.36143 -0.4293306
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020        1024.67 719.8003 1329.540 558.41209 1490.928
## Feb 2020        1024.67 593.5192 1455.821 365.28174 1684.058
## Mar 2020        1024.67 496.6203 1552.720 217.08761 1832.252
## Apr 2020        1024.67 414.9307 1634.409  92.15418 1957.186
## May 2020        1024.67 342.9607 1706.379 -17.91439 2067.254
```

Seasonal Naive Forecast

```r
snaive_forecast = snaive(train_tng,5)
plot(snaive_forecast)
```

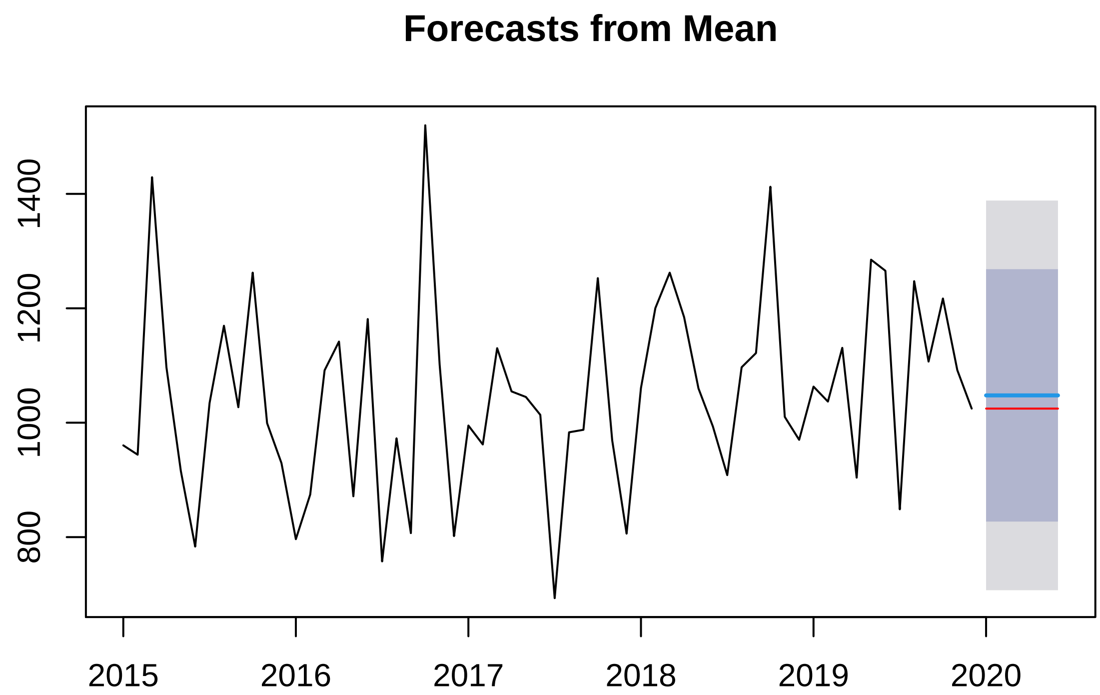



```r
summary(snaive_forecast)
```

```
##
## Forecast method: Seasonal naive method
##
## Model Information:
## Call: snaive(y = train_tng, h = 5)
##
## Residual sd: 167.9362
##
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE MASE       ACF1
## Training set 13.95604 167.9362 139.8215 0.1167369 13.38504    1 0.02684331
##
## Forecasts:
##          Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2020        1063.13  847.9111 1278.349 733.9811 1392.279
## Feb 2020        1036.95  821.7311 1252.169 707.8011 1366.099
## Mar 2020        1130.87  915.6511 1346.089 801.7211 1460.019
## Apr 2020         903.97  688.7511 1119.189 574.8211 1233.119
## May 2020        1284.95 1069.7311 1500.169 955.8011 1614.099
```

Plotting mean and naive forecasting together

```r
plot(mean_forecast)
lines(naive_forecast$mean,col="red")
```

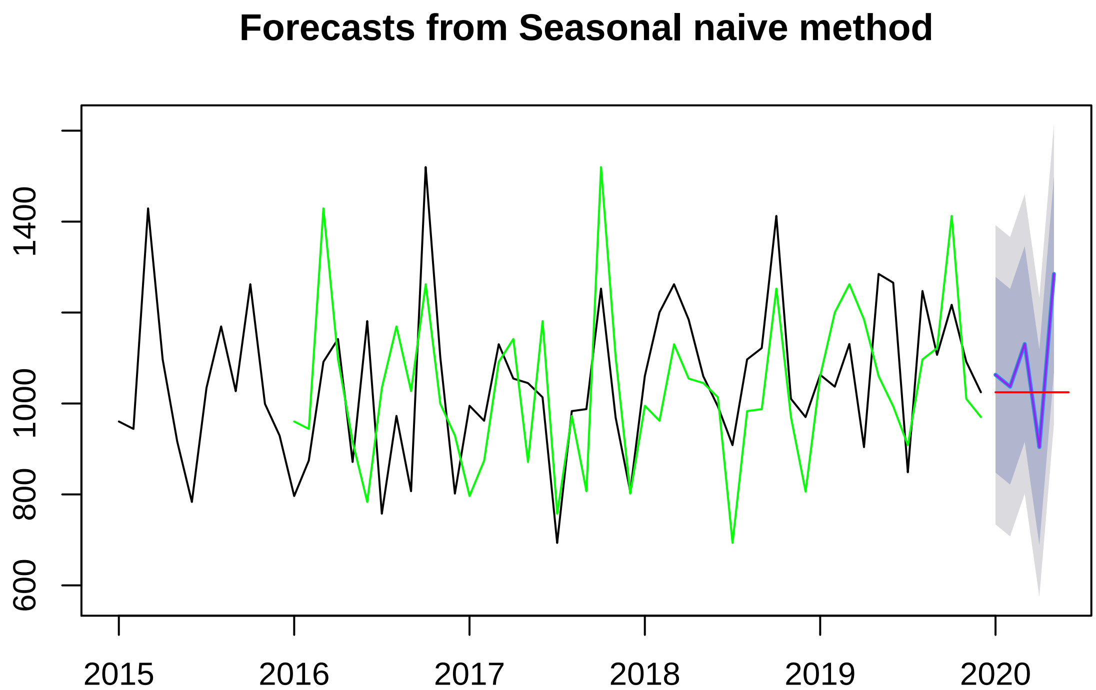



```r
attributes(naive_forecast)
```

```
## $names
##  [1] "method"    "model"     "lambda"    "x"         "fitted"    "residuals"
##  [7] "series"    "mean"      "level"     "lower"     "upper"
##
## $class
## [1] "forecast"
```

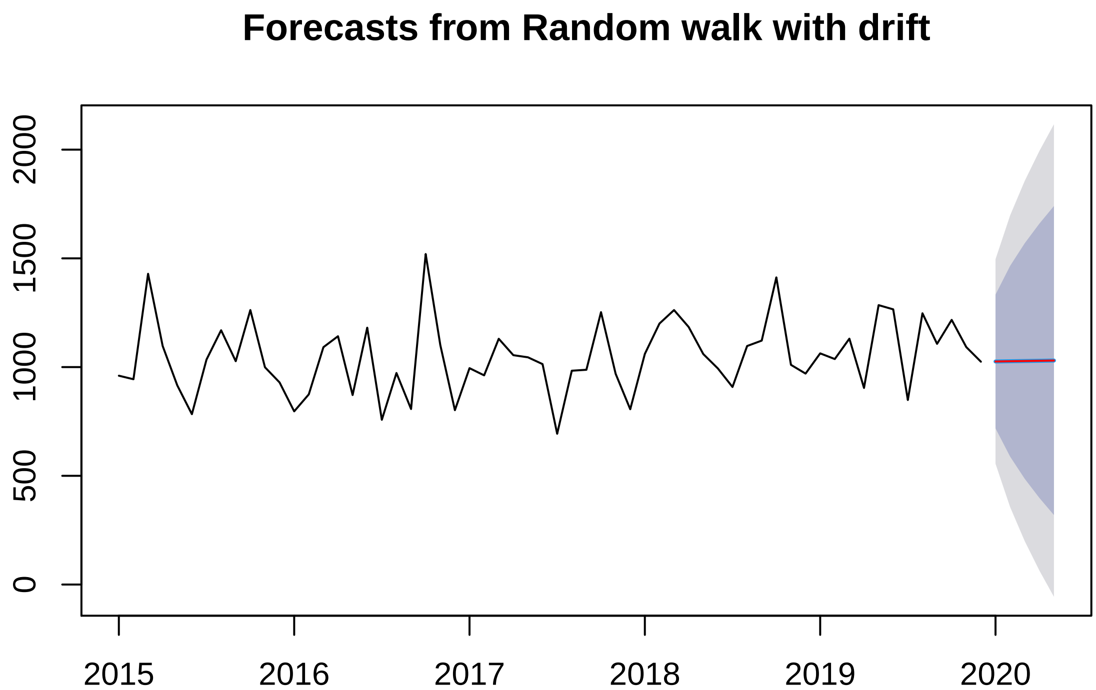
Plotting other attributes

```r
plot(snaive_forecast)
lines(rwf_forecast$mean,col="yellow")
lines(snaive_forecast$mean,col="purple")
lines(snaive_forecast$fitted, col = "green")
lines(naive_forecast$mean,col="red")
```



Drift with RWF

```
rwf_drift = rwf(train_tng,5,drift = TRUE)
plot(rwf_drift)
lines(rwf_drift$mean, col = "red")
```

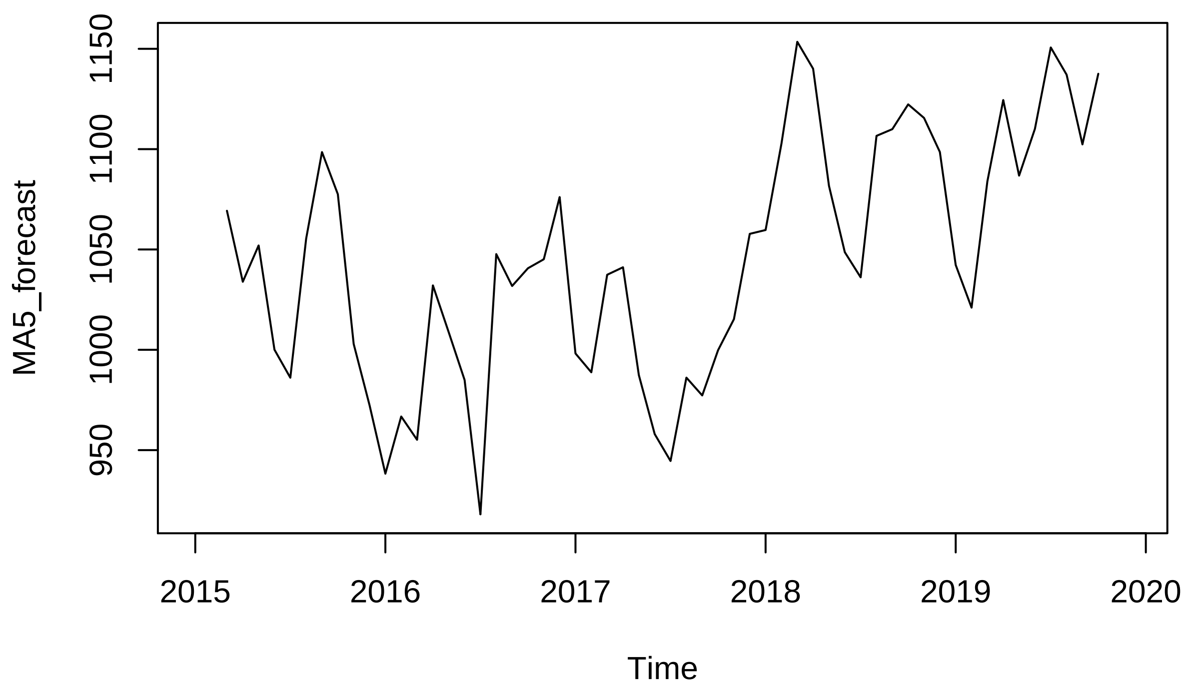



```r
summary(rwf_drift)
```

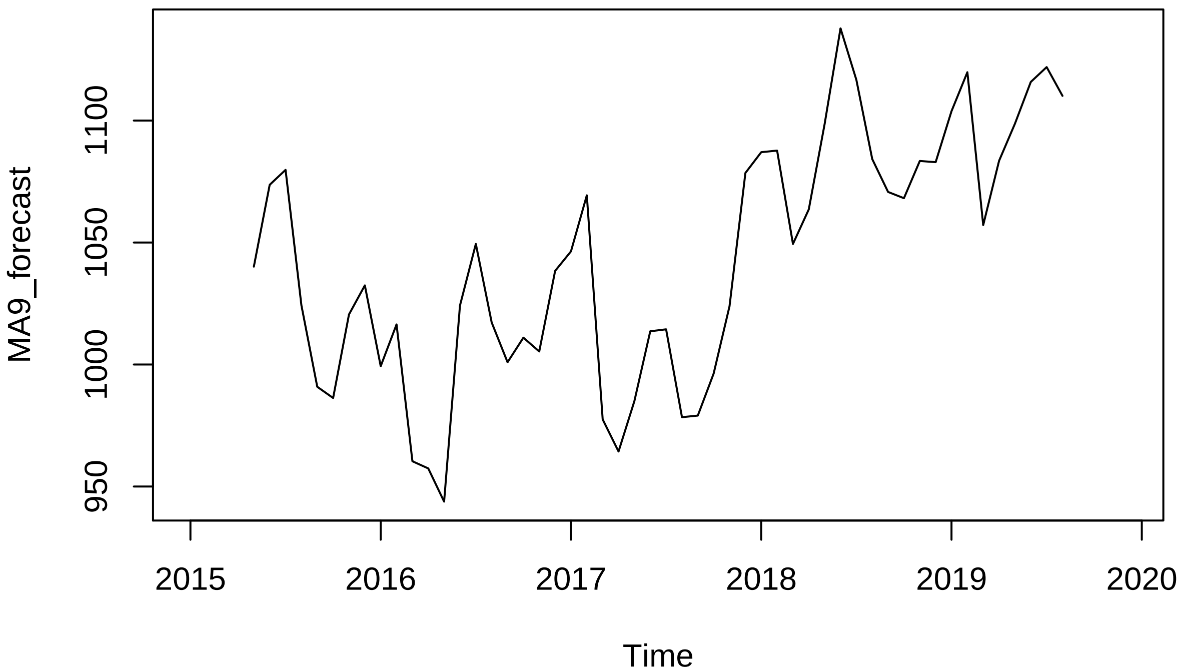
```
##
## Forecast method: Random walk with drift
##
## Model Information:
## Call: rwf(y = train_tng, h = 5, drift = TRUE)
##
## Drift: 1.089  (se 31.2363)
## Residual sd: 239.9306
##
## Error measures:
##                        ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 1.348733e-14 237.8886 190.4863 -2.531564 18.39682 1.362354
##                    ACF1
## Training set -0.4293306
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020       1025.759 718.2756 1333.242 555.50371 1496.014
## Feb 2020       1026.848 588.3311 1465.365 356.19432 1697.502
## Mar 2020       1027.937 486.4086 1569.465 199.74079 1856.133
## Apr 2020       1029.026 398.6182 1659.434  64.90057 1993.151
## May 2020       1030.115 319.6364 1740.593 -56.46818 2116.698
```

Moving Average Forecast

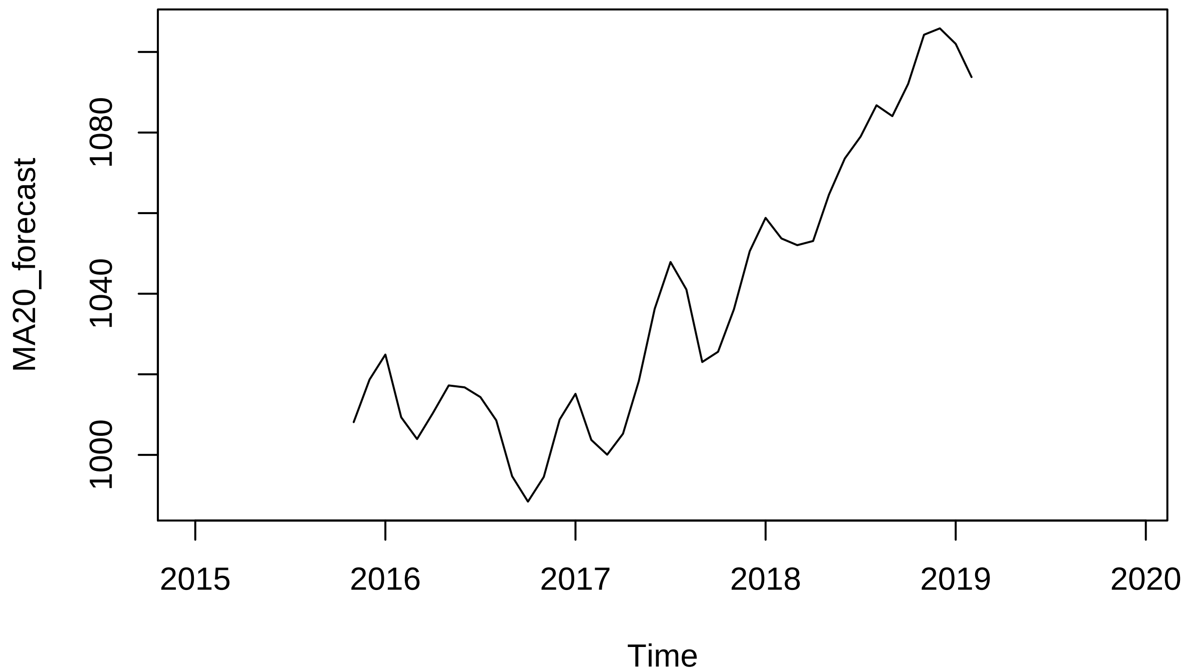
```r
MA5_forecast <- ma(train_tng,order=5)
MA9_forecast <- ma(train_tng,order=9)
MA20_forecast <- ma(train_tng,order=20)
plot(MA5_forecast)
```



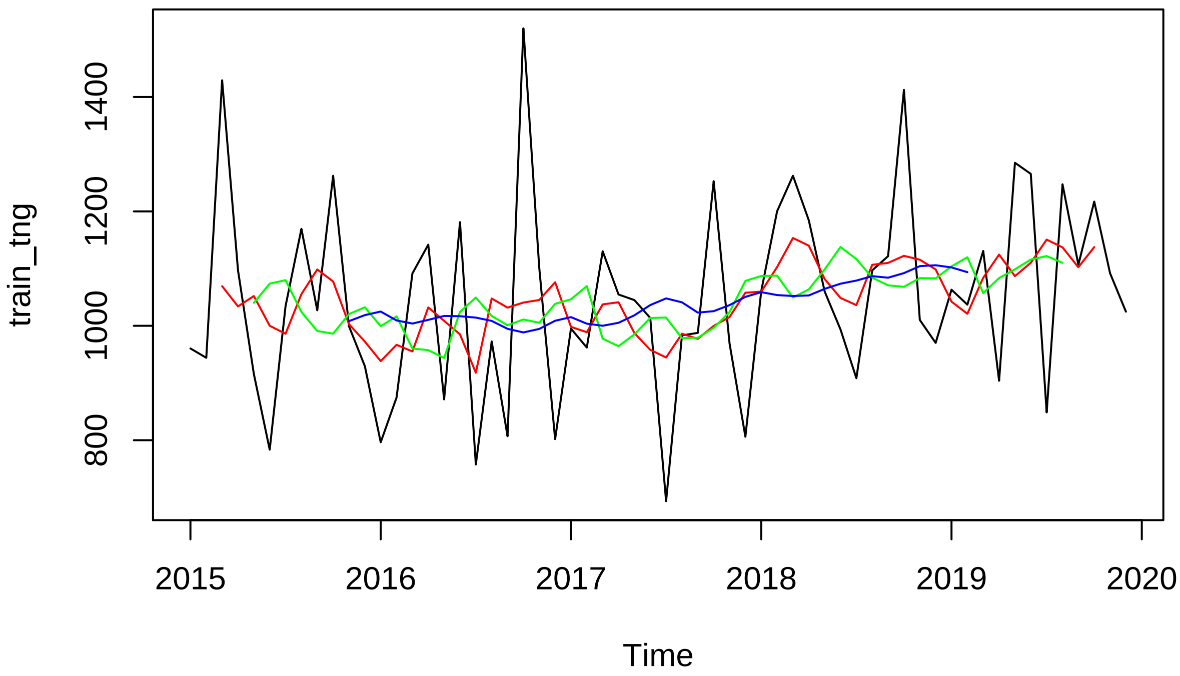

```r
plot(MA9_forecast)
```




```r
plot(MA20_forecast)
```




```
plot(train_tng)
lines(MA5_forecast, col = "Red")
lines(MA9_forecast, col = "Green")
lines(MA20_forecast, col = "Blue")
```




```
summary(MA5_forecast)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   918.0   999.4  1043.7  1045.7  1098.5  1153.5       4
```

As we increase the order, the graph becomes smoother and randomness in the data is decreased.

ETS

```
ets(train_tng)
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = train_tng)
##
##   Smoothing parameters:
##     alpha = 0.0576
##     gamma = 1e-04
##
##   Initial states:
```

```
##     l = 1047.4318
##     s = 0.8577 0.9689 1.2739 0.9471 1.0336 0.8229
##            1.0137 1.0033 1.0204 1.1643 0.9529 0.9413
##
##   sigma:  0.1274
##
##      AIC     AICc      BIC
## 845.0223 855.9314 876.4374
```

Holt Winters

```r
HoltWinters(train_tng)
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train_tng)
##
## Smoothing parameters:
##  alpha: 0.03242971
##  beta : 0.2550264
##  gamma: 0.5409953
##
## Coefficients:
##            [,1]
## a   1098.083982
## b      4.106445
## s1     5.752170
## s2    26.587256
## s3   131.000958
## s4   -13.303193
## s5   126.573215
## s6   121.179458
## s7  -195.078411
## s8   112.683879
## s9    24.150535
## s10  233.999789
## s11   -2.426057
## s12  -91.113979
```

SSE without trend and without seasonality

```r
HoltWinters(train_tng,beta=FALSE,gamma=FALSE)
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = train_tng, beta = FALSE, gamma = FALSE)
##
```

```
## Smoothing parameters:
##  alpha: 0.05955483
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 1078.308
```

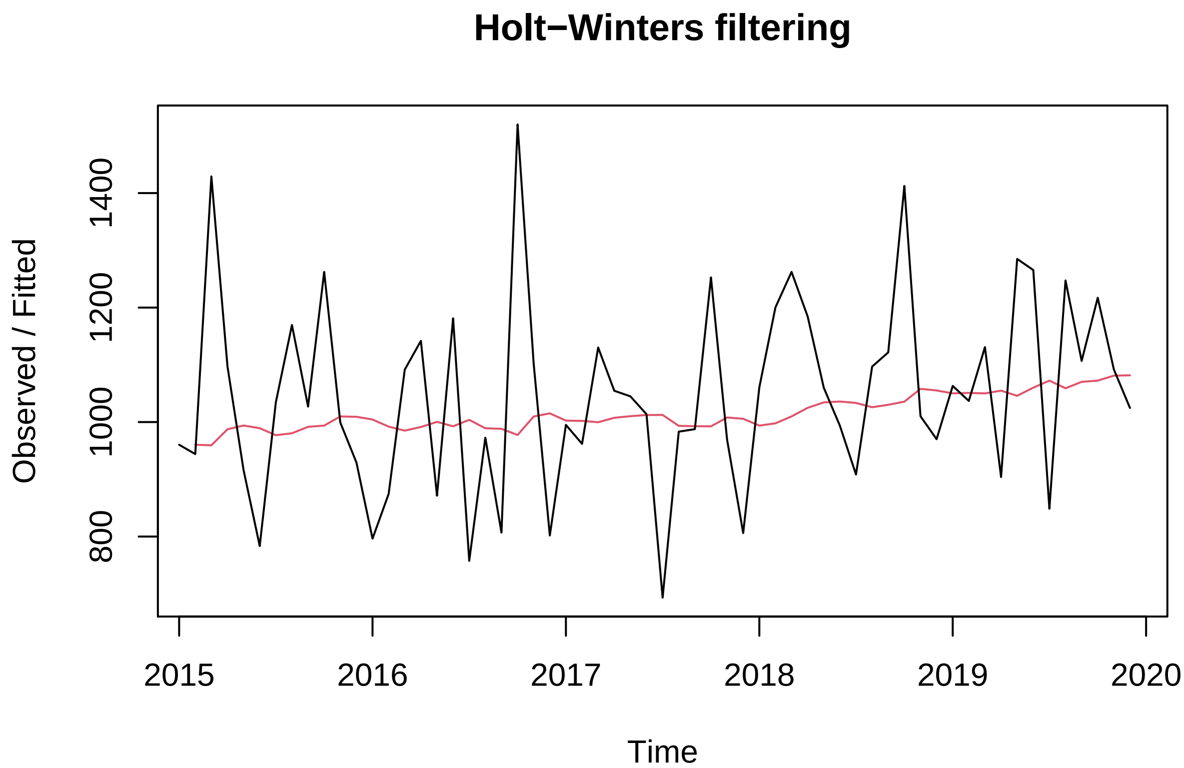
```r
hw_forecast_level = HoltWinters(train_tng,beta=FALSE,gamma=FALSE)
hw_forecast_level
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = train_tng, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.05955483
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 1078.308
```

```r
attributes(hw_forecast_level)
```

```
## $names
## [1] "fitted"       "x"            "alpha"        "beta"         "gamma"
## [6] "coefficients" "seasonal"     "SSE"          "call"
##
## $class
## [1] "HoltWinters"
```

```r
plot(hw_forecast_level)
```

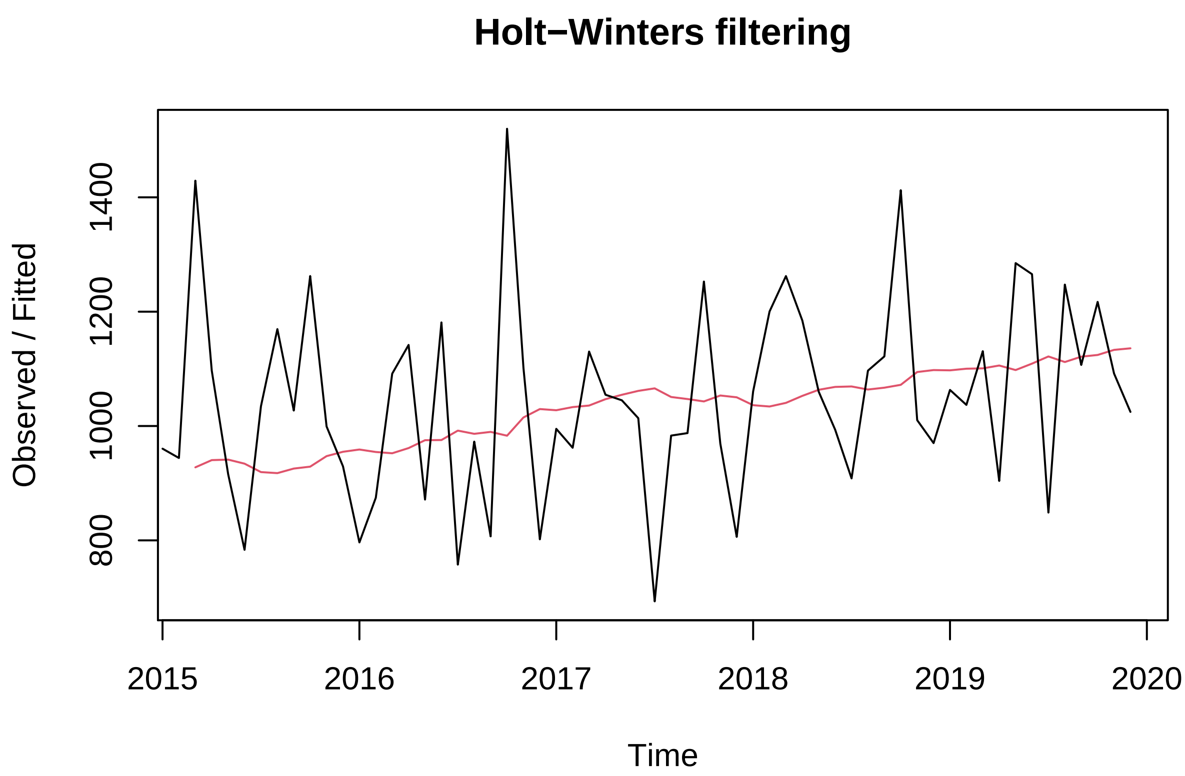



```r
hw_forecast_level$SSE
```

```
## [1] 1776425
```

SSE with Trend but no Seasonlaity

```r
hw_forecast_trend = HoltWinters(train_tng,gamma=FALSE)
plot(hw_forecast_trend)
```




```
hw_forecast_trend
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = train_tng, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.04146809
##  beta : 0.3933152
##  gamma: FALSE
##
## Coefficients:
##         [,1]
## a 1131.268702
## b    2.580617
```

```
hw_forecast_trend$SSE #Check the residual error magnitude
```

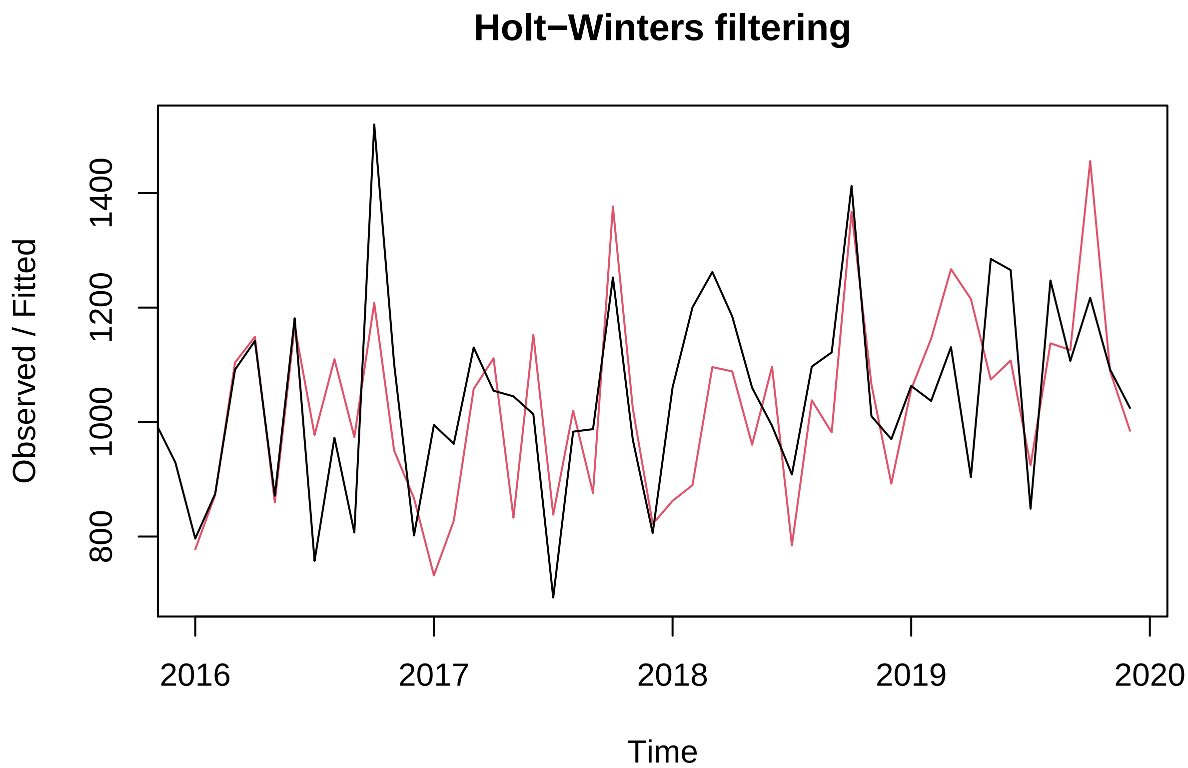
```
## [1] 1841394
```

SSE with trend and seasonality

```r
hw_forecast_season = HoltWinters(train_tng)
hw_forecast_season
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train_tng)
##
## Smoothing parameters:
##  alpha: 0.03242971
##  beta : 0.2550264
##  gamma: 0.5409953
##
## Coefficients:
##             [,1]
## a   1098.083982
## b      4.106445
## s1     5.752170
## s2    26.587256
## s3   131.000958
## s4   -13.303193
## s5   126.573215
## s6   121.179458
## s7  -195.078411
## s8   112.683879
## s9    24.150535
## s10  233.999789
## s11   -2.426057
## s12  -91.113979
```

```r
plot(hw_forecast_season)
```



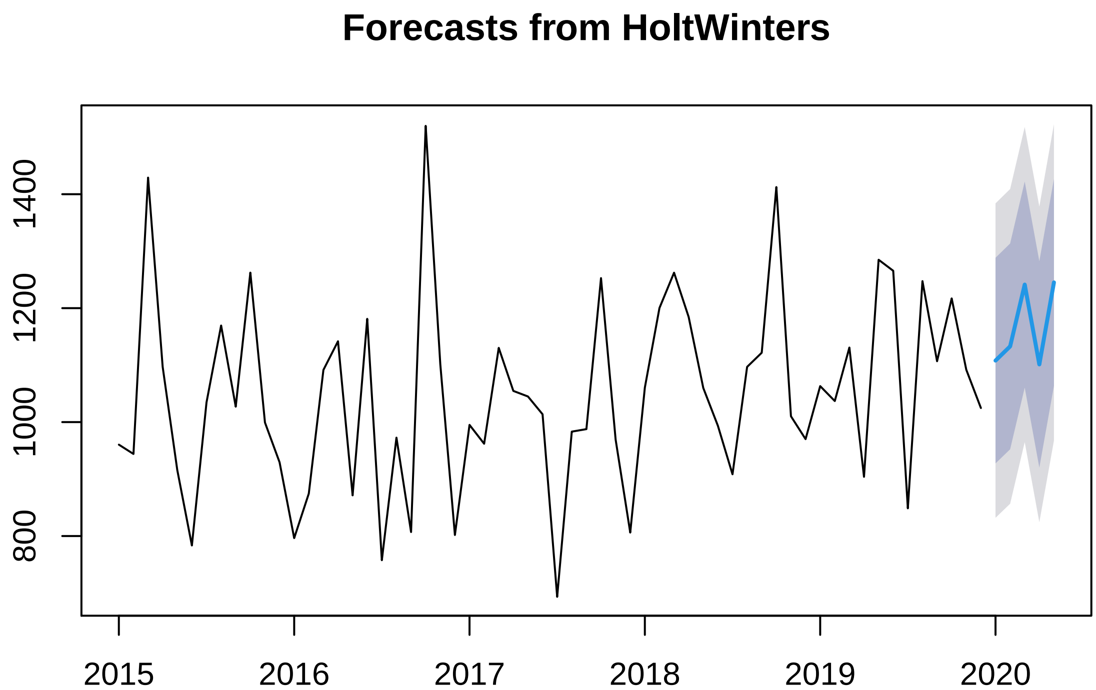

```
hw_forecast_season$SSE
```

```
## [1] 949245.4
```

```
hw_forecast_all = forecast(hw_forecast_season,h =5)
hw_forecast_all
```

```
##          Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2020       1107.943  927.5032 1288.382 831.9843 1383.901
## Feb 2020       1132.884  952.2953 1313.473 856.6974 1409.071
## Mar 2020       1241.404 1060.5994 1422.209 964.8871 1517.921
## Apr 2020       1101.207  920.1069 1282.306 824.2386 1378.175
## May 2020       1245.189 1063.7044 1426.674 967.6320 1522.747
```

```
plot(hw_forecast_all)
```




```r
accuracy(hw_forecast_all)
```

```
##                    ME    RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 19.10421 140.627 111.9195 0.8341924 10.74567 0.8004457 0.1613357
```

SSE of HoltWinters with Trend and Seasonality is smaller than the SSE of Holtwinter without trend, without seasonality and SSE of Holtwinters with Trend and without seasonality.

Ets

```r
ets(train_tng)
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = train_tng)
##
##   Smoothing parameters:
##     alpha = 0.0576
##     gamma = 1e-04
##
```

```
##   Initial states:
##     l = 1047.4318
##     s = 0.8577 0.9689 1.2739 0.9471 1.0336 0.8229
##            1.0137 1.0033 1.0204 1.1643 0.9529 0.9413
##
##   sigma:  0.1274
##
##      AIC     AICc      BIC
## 845.0223 855.9314 876.4374
```

```
ets_forecast = ets(train_tng)
attributes(ets)
```

```
## NULL
```

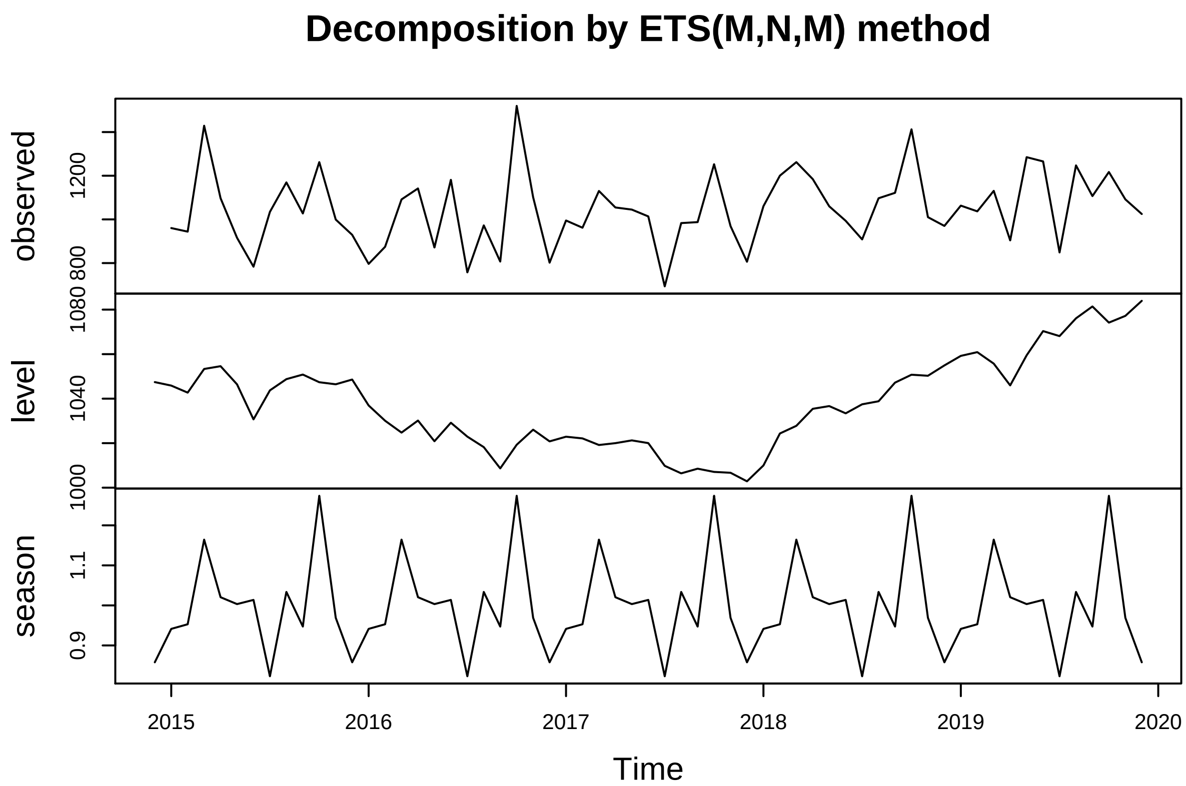
```
attributes(ets_forecast)
```

```
## $names
##  [1] "loglik"     "aic"        "bic"        "aicc"       "mse"
##  [6] "amse"       "fit"        "residuals"  "fitted"     "states"
## [11] "par"        "m"          "method"     "series"     "components"
## [16] "call"       "initstate"  "sigma2"     "x"
##
## $class
## [1] "ets"
```

```
ets_forecast$mse
```

```
## [1] 13415.89
```

```
plot(ets_forecast)
```

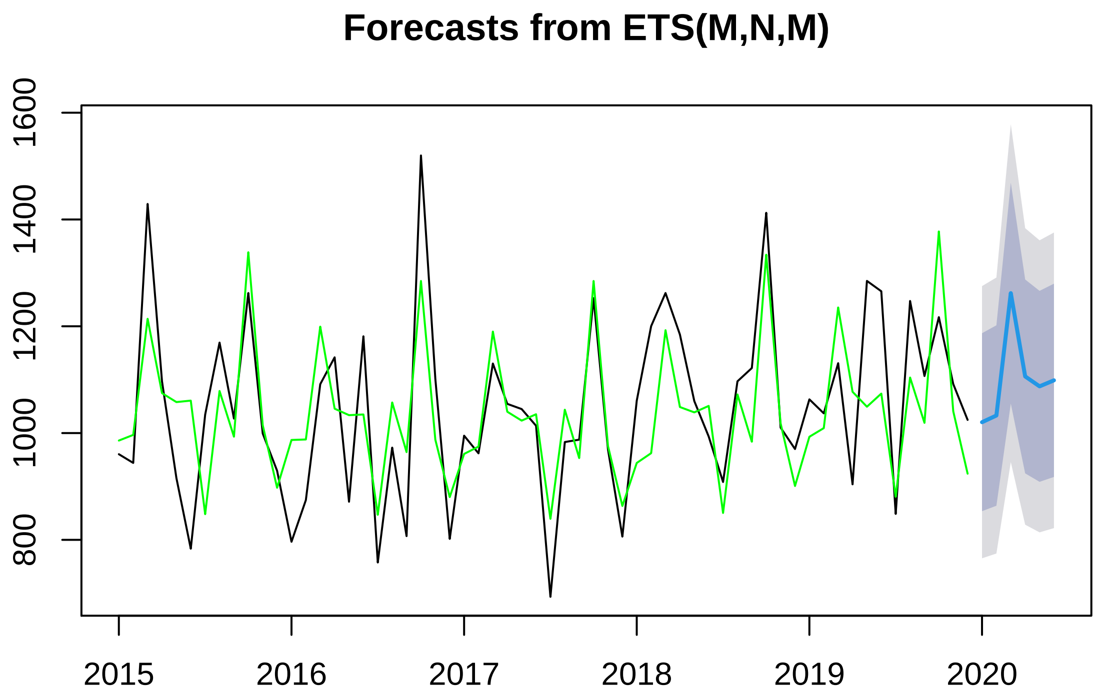


Forecast with Ets

```r
forecast.ets(ets_forecast, h=6)
```

```
##          Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2020       1020.365  853.7138 1187.016 765.4940 1275.236
## Feb 2020       1032.863  863.8867 1201.839 774.4362 1291.289
## Mar 2020       1262.057 1055.2391 1468.876 945.7561 1578.359
## Apr 2020       1106.075  924.5152 1287.634 828.4033 1383.746
## May 2020       1087.494  908.6867 1266.301 814.0321 1360.955
## Jun 2020       1098.813  917.8453 1279.781 822.0466 1375.580
```

```r
forecast_ets = forecast.ets(ets_forecast, h=6)
plot(forecast_ets)
lines(forecast_ets$fitted, col="green")
```




```
accuracy(forecast_ets)
```

```
##                    ME    RMSE      MAE        MPE     MAPE      MASE       ACF1
## Training set 10.47703 115.827 93.97734 -0.2245269 9.112492 0.6721239 0.04408278
```

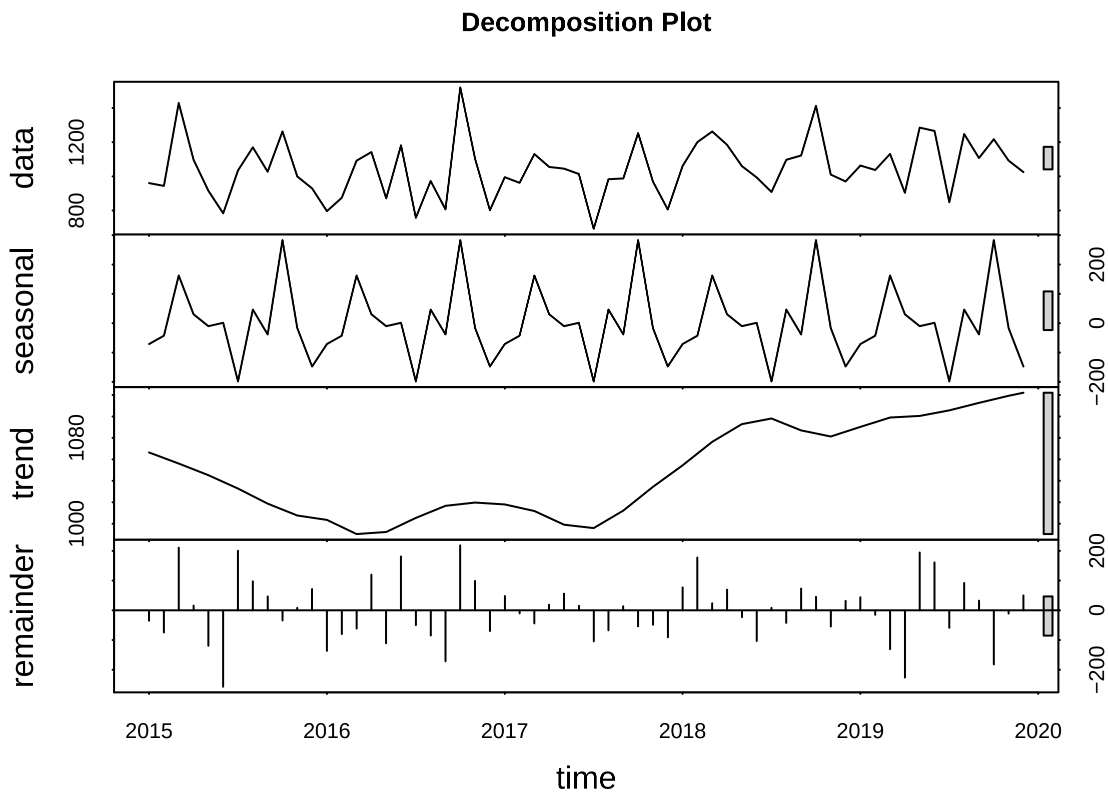
Decomposition

```
stl_decomp = stl(train_tng, s.window = "periodic")
stl_decomp
```

```
##  Call:
##  stl(x = train_tng, s.window = "periodic")
##
## Components
##              seasonal     trend   remainder
## Jan 2015  -70.915777 1066.4205  -35.084739
## Feb 2015  -42.611463 1061.2528  -74.561376
## Mar 2015  162.492749 1056.0852  210.542088
## Apr 2015   30.504883 1050.6743   15.820826
## May 2015  -10.059213 1045.2634 -119.354208
## Jun 2015    1.393934 1039.0208 -256.964697
## Jul 2015 -198.257090 1032.7781  199.998984
## Aug 2015   46.454532 1025.7504   97.295063
```

```
## Sep 2015  -38.201779 1018.7227   46.559073
## Oct 2015  283.244350 1013.1913  -34.115634
## Nov 2015  -16.571712 1007.6599    8.161849
## Dec 2015 -147.473465 1005.6070   71.286507
## Jan 2016  -70.915777 1003.5541 -136.218277
## Feb 2016  -42.611463  996.9635  -79.802043
## Mar 2016  162.492749  990.3730  -61.315707
## Apr 2016   30.504883  991.3423  119.992841
## May 2016  -10.059213  992.3116 -110.892381
## Jun 2016    1.393934  998.8536  180.962509
## Jul 2016 -198.257090 1005.3955  -49.548430
## Aug 2016   46.454532 1011.0417  -84.766246
## Sep 2016  -38.201779 1016.6879 -171.466130
## Oct 2016  283.244350 1018.2038  218.471859
## Nov 2016  -16.571712 1019.7197   98.522038
## Dec 2016 -147.473465 1018.8521  -69.548663
## Jan 2017  -70.915777 1017.9846   48.021194
## Feb 2017  -42.611463 1014.9185  -10.307084
## Mar 2017  162.492749 1011.8525  -44.105260
## Apr 2017   30.504883 1005.4660   18.739121
## May 2017  -10.059213  999.0795   55.929732
## Jun 2017    1.393934  997.5038   14.832273
## Jul 2017 -198.257090  995.9281 -104.341015
## Aug 2017   46.454532 1004.0896  -67.294128
## Sep 2017  -38.201779 1012.2511   13.590691
## Oct 2017  283.244350 1023.3028  -53.857162
## Nov 2017  -16.571712 1034.3545  -48.472825
## Dec 2017 -147.473465 1044.3906  -90.817098
## Jan 2018  -70.915777 1054.4266   77.059186
## Feb 2018  -42.611463 1065.3955  177.465993
## Mar 2018  162.492749 1076.3643   23.392902
## Apr 2018   30.504883 1084.6045   69.340655
## May 2018  -10.059213 1092.8446  -22.865363
## Jun 2018    1.393934 1095.4737 -103.317629
## Jul 2018 -198.257090 1098.1028    8.524276
## Aug 2018   46.454532 1092.5451  -42.069635
## Sep 2018  -38.201779 1086.9874   72.964385
## Oct 2018  283.244350 1084.1472   45.078464
## Nov 2018  -16.571712 1081.3070  -54.485268
## Dec 2018 -147.473465 1085.7953   31.798174
## Jan 2019  -70.915777 1090.2836   43.762173
## Feb 2019  -42.611463 1094.6359  -15.074463
## Mar 2019  162.492749 1098.9882 -130.610997
## Apr 2019   30.504883 1099.7370 -226.271912
## May 2019  -10.059213 1100.4858  194.523402
## Jun 2019    1.393934 1103.0818  161.084246
## Jul 2019 -198.257090 1105.6778  -58.780740
## Aug 2019   46.454532 1109.1747   91.770742
## Sep 2019  -38.201779 1112.6716   32.370157
## Oct 2019  283.244350 1115.9599 -182.124262
## Nov 2019  -16.571712 1119.2482  -10.836491
## Dec 2019 -147.473465 1122.1292   50.014278
```

```r
plot(stl_decomp, main="Decomposition Plot")
```




```r
attributes(stl_decomp)
```

```
## $names
## [1] "time.series" "weights"     "call"        "win"         "deg"
## [6] "jump"        "inner"       "outer"
##
## $class
## [1] "stl"
```

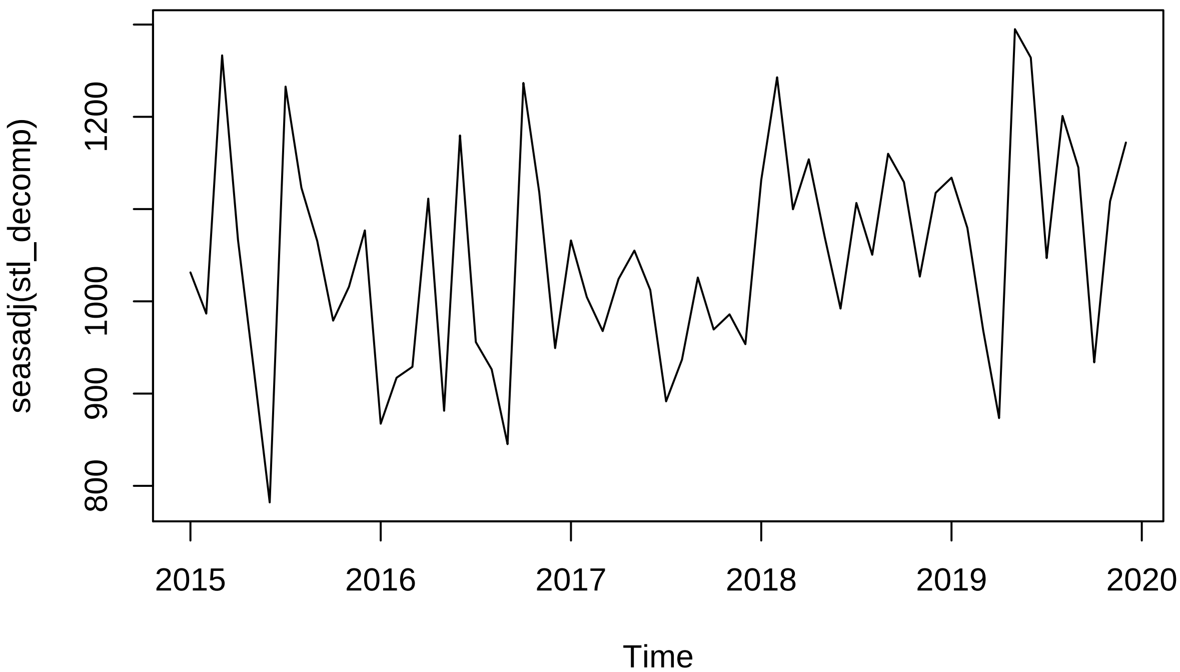
Seasonal Adjustment

```r
seasadj(stl_decomp)
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2015 1031.3358  986.6915 1266.6273 1066.4951  925.9092  782.0561 1232.7771
## 2016  867.3358  917.1615  929.0573 1111.3351  881.4192 1179.8161  955.8471
## 2017 1066.0058 1004.6115  967.7473 1024.2051 1055.0092 1012.3361  891.5871
## 2018 1131.4858 1242.8615 1099.7573 1153.9451 1069.9792  992.1561 1106.6271
## 2019 1134.0458 1079.5615  968.3773  873.4651 1295.0092 1264.1661 1046.8971
##            Aug       Sep       Oct       Nov       Dec
```

```
## 2015 1123.0455 1065.2818  979.0757 1015.8217 1076.8935
## 2016  926.2755  845.2218 1236.6757 1118.2417  949.3035
## 2017  936.7955 1025.8418  969.4457  985.8817  953.5735
## 2018 1050.4755 1159.9518 1129.2257 1026.8217 1117.5935
## 2019 1200.9455 1145.0418  933.8357 1108.4117 1172.1435
```

```
plot(seasadj(stl_decomp))
```



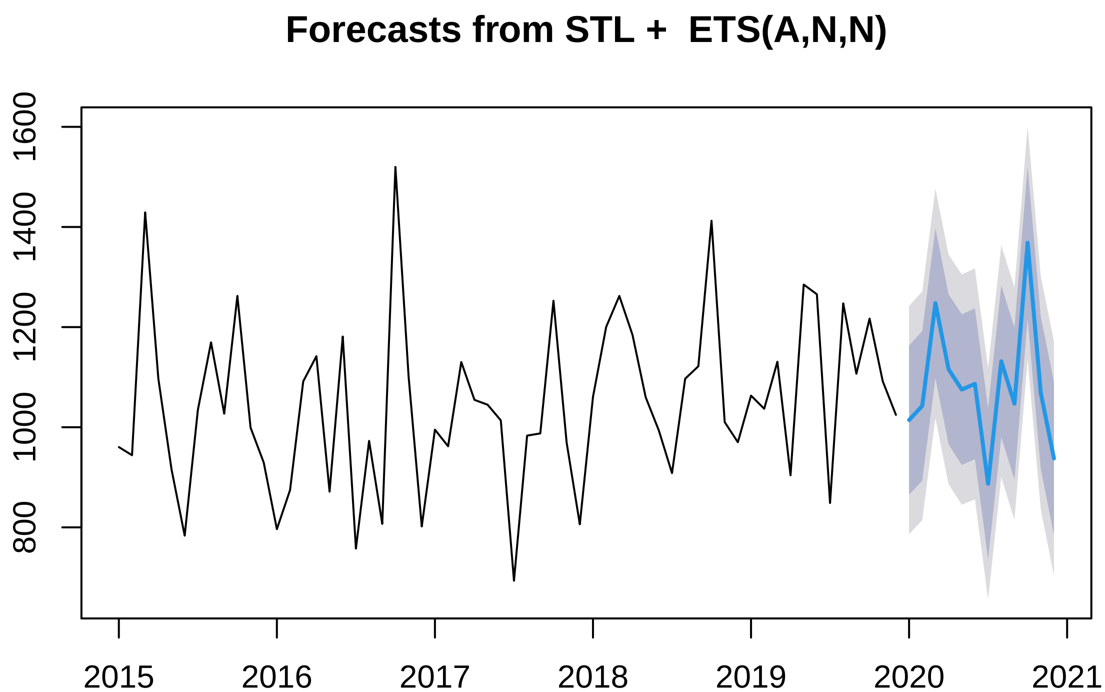
Default Period Forecast

```
f_stl = forecast(stl_decomp,h = 12)
f_stl
```

```
##          Point Forecast      Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2020      1014.3270  865.2967 1163.357  786.4049 1242.249
## Feb 2020      1042.6313  893.2525 1192.010  814.1762 1271.086
## Mar 2020      1247.7355 1098.0091 1397.462 1018.7487 1476.722
## Apr 2020      1115.7477  965.6743 1265.821  886.2303 1345.265
## May 2020      1075.1836  924.7641 1225.603  845.1369 1305.230
## Jun 2020      1086.6367  935.8720 1237.401  856.0619 1317.211
## Jul 2020       886.9857  735.8764 1038.095  655.8840 1118.087
## Aug 2020      1131.6973  980.2443 1283.150  900.0699 1363.325
## Sep 2020      1047.0410  895.2451 1198.837  814.8891 1279.193
```

```
## Oct 2020       1368.4871 1216.3490 1520.625 1135.8120 1601.162
## Nov 2020       1068.6711  916.1915 1221.151  835.4738 1301.868
## Dec 2020        937.7693  784.9491 1090.589  704.0510 1171.488
```

```
plot(f_stl)
```




```
accuracy(f_stl)
```

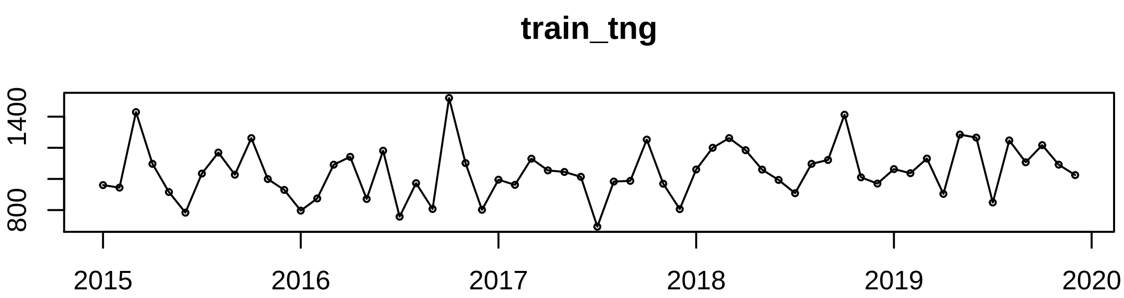
```
##                    ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
## Training set 13.38141 114.3344 91.29621 0.1050379 8.793197 0.6529485 0.02342607
```

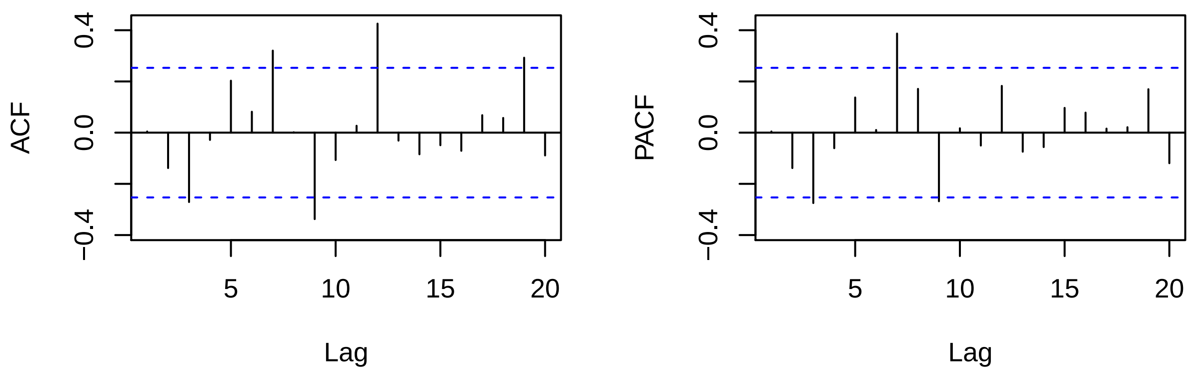
Accuracy is improved for stl decomp as MAPE is slightly lower compared to other forecasts.

```
ndiffs(train_tng)
```

```
## [1] 0
```

```
tsdisplay(train_tng)
```






```r
auto_fit = auto.arima(train_tng,trace = TRUE, stepwise = FALSE)
```

```
##
##  ARIMA(0,0,0)            with zero mean     : 1008.383
##  ARIMA(0,0,0)            with non-zero mean : 788.9207
##  ARIMA(0,0,0)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,0)(0,0,1)[12] with non-zero mean : 779.2159
##  ARIMA(0,0,0)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(0,0,0)(1,0,0)[12] with non-zero mean : 776.3417
##  ARIMA(0,0,0)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,0)(1,0,1)[12] with non-zero mean : 777.4751
##  ARIMA(0,0,1)            with zero mean     : 952.3381
##  ARIMA(0,0,1)            with non-zero mean : 791.1371
##  ARIMA(0,0,1)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,1)(0,0,1)[12] with non-zero mean : 781.4647
##  ARIMA(0,0,1)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(0,0,1)(1,0,0)[12] with non-zero mean : 778.6208
##  ARIMA(0,0,1)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,1)(1,0,1)[12] with non-zero mean : 779.7131
##  ARIMA(0,0,2)            with zero mean     : 907.2159
##  ARIMA(0,0,2)            with non-zero mean : 791.9724
##  ARIMA(0,0,2)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,2)(0,0,1)[12] with non-zero mean : 782.9171
##  ARIMA(0,0,2)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(0,0,2)(1,0,0)[12] with non-zero mean : 780.6271
```

```
##  ARIMA(0,0,2)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,2)(1,0,1)[12] with non-zero mean : 781.8547
##  ARIMA(0,0,3)            with zero mean     : 894.2347
##  ARIMA(0,0,3)            with non-zero mean : 790.644
##  ARIMA(0,0,3)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,3)(0,0,1)[12] with non-zero mean : 782.7132
##  ARIMA(0,0,3)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(0,0,3)(1,0,0)[12] with non-zero mean : 780.7328
##  ARIMA(0,0,3)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,3)(1,0,1)[12] with non-zero mean : 782.8134
##  ARIMA(0,0,4)            with zero mean     : Inf
##  ARIMA(0,0,4)            with non-zero mean : 790.0069
##  ARIMA(0,0,4)(0,0,1)[12] with zero mean     : 858.5537
##  ARIMA(0,0,4)(0,0,1)[12] with non-zero mean : 784.3248
##  ARIMA(0,0,4)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(0,0,4)(1,0,0)[12] with non-zero mean : 782.8422
##  ARIMA(0,0,5)            with zero mean     : Inf
##  ARIMA(0,0,5)            with non-zero mean : 787.4745
##  ARIMA(1,0,0)            with zero mean     : 833.2195
##  ARIMA(1,0,0)            with non-zero mean : 791.1376
##  ARIMA(1,0,0)(0,0,1)[12] with zero mean     : 824.8455
##  ARIMA(1,0,0)(0,0,1)[12] with non-zero mean : 781.4753
##  ARIMA(1,0,0)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(1,0,0)(1,0,0)[12] with non-zero mean : 778.6238
##  ARIMA(1,0,0)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(1,0,0)(1,0,1)[12] with non-zero mean : 779.7377
##  ARIMA(1,0,1)            with zero mean     : Inf
##  ARIMA(1,0,1)            with non-zero mean : 792.5145
##  ARIMA(1,0,1)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(1,0,1)(0,0,1)[12] with non-zero mean : 782.803
##  ARIMA(1,0,1)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(1,0,1)(1,0,0)[12] with non-zero mean : 780.7184
##  ARIMA(1,0,1)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(1,0,1)(1,0,1)[12] with non-zero mean : 782.291
##  ARIMA(1,0,2)            with zero mean     : Inf
##  ARIMA(1,0,2)            with non-zero mean : 793.6809
##  ARIMA(1,0,2)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(1,0,2)(0,0,1)[12] with non-zero mean : 784.6528
##  ARIMA(1,0,2)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(1,0,2)(1,0,0)[12] with non-zero mean : 782.5433
##  ARIMA(1,0,2)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(1,0,2)(1,0,1)[12] with non-zero mean : 784.137
##  ARIMA(1,0,3) with zero mean     : Inf
##  ARIMA(1,0,3)            with non-zero mean : 792.4993
##  ARIMA(1,0,3)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(1,0,3)(0,0,1)[12] with non-zero mean : 785.0136
##  ARIMA(1,0,3)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(1,0,3)(1,0,0)[12] with non-zero mean : 783.1365
##  ARIMA(1,0,4)            with zero mean     : Inf
##  ARIMA(1,0,4)            with non-zero mean : 788.8028
##  ARIMA(2,0,0)            with zero mean     : 824.0601
##  ARIMA(2,0,0)            with non-zero mean : 792.3004
##  ARIMA(2,0,0)(0,0,1)[12] with zero mean     : 814.1226
##  ARIMA(2,0,0)(0,0,1)[12] with non-zero mean : 783.2925
```

```
##  ARIMA(2,0,0)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(2,0,0)(1,0,0)[12] with non-zero mean : 780.6224
##  ARIMA(2,0,0)(1,0,1)[12] with zero mean     : 809.0495
##  ARIMA(2,0,0)(1,0,1)[12] with non-zero mean : 781.6862
##  ARIMA(2,0,1) with zero mean     : Inf
##  ARIMA(2,0,1)            with non-zero mean : 792.7482
##  ARIMA(2,0,1)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(2,0,1)(0,0,1)[12] with non-zero mean : Inf
##  ARIMA(2,0,1)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(2,0,1)(1,0,0)[12] with non-zero mean : 782.2036
##  ARIMA(2,0,1)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(2,0,1)(1,0,1)[12] with non-zero mean : 783.8318
##  ARIMA(2,0,2) with zero mean     : Inf
##  ARIMA(2,0,2)            with non-zero mean : Inf
##  ARIMA(2,0,2)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(2,0,2)(0,0,1)[12] with non-zero mean : 782.2635
##  ARIMA(2,0,2)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(2,0,2)(1,0,0)[12] with non-zero mean : Inf
##  ARIMA(2,0,3) with zero mean     : Inf
##  ARIMA(2,0,3)            with non-zero mean : 792.2954
##  ARIMA(3,0,0)            with zero mean     : Inf
##  ARIMA(3,0,0)            with non-zero mean : 789.6905
##  ARIMA(3,0,0)(0,0,1)[12] with zero mean     : 813.255
##  ARIMA(3,0,0)(0,0,1)[12] with non-zero mean : 782.134
##  ARIMA(3,0,0)(1,0,0)[12] with zero mean     : 808.3923
##  ARIMA(3,0,0)(1,0,0)[12] with non-zero mean : 780.3886
##  ARIMA(3,0,0)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(3,0,0)(1,0,1)[12] with non-zero mean : 782.5533
##  ARIMA(3,0,1)            with zero mean     : Inf
##  ARIMA(3,0,1)            with non-zero mean : 792.0619
##  ARIMA(3,0,1)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(3,0,1)(0,0,1)[12] with non-zero mean : 784.5655
##  ARIMA(3,0,1)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(3,0,1)(1,0,0)[12] with non-zero mean : 782.9557
##  ARIMA(3,0,2)            with zero mean     : Inf
##  ARIMA(3,0,2)            with non-zero mean : Inf
##  ARIMA(4,0,0)            with zero mean     : Inf
##  ARIMA(4,0,0)            with non-zero mean : 791.9508
##  ARIMA(4,0,0)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(4,0,0)(0,0,1)[12] with non-zero mean : 784.4907
##  ARIMA(4,0,0)(1,0,0)[12] with zero mean     : Inf
##  ARIMA(4,0,0)(1,0,0)[12] with non-zero mean : 782.9545
##  ARIMA(4,0,1)            with zero mean     : Inf
##  ARIMA(4,0,1)            with non-zero mean : 793.4599
##  ARIMA(5,0,0)            with zero mean     : Inf
##  ARIMA(5,0,0)            with non-zero mean : 793.1683
##
##
##
##  Best model: ARIMA(0,0,0)(1,0,0)[12] with non-zero mean
```

```
auto_fit
```

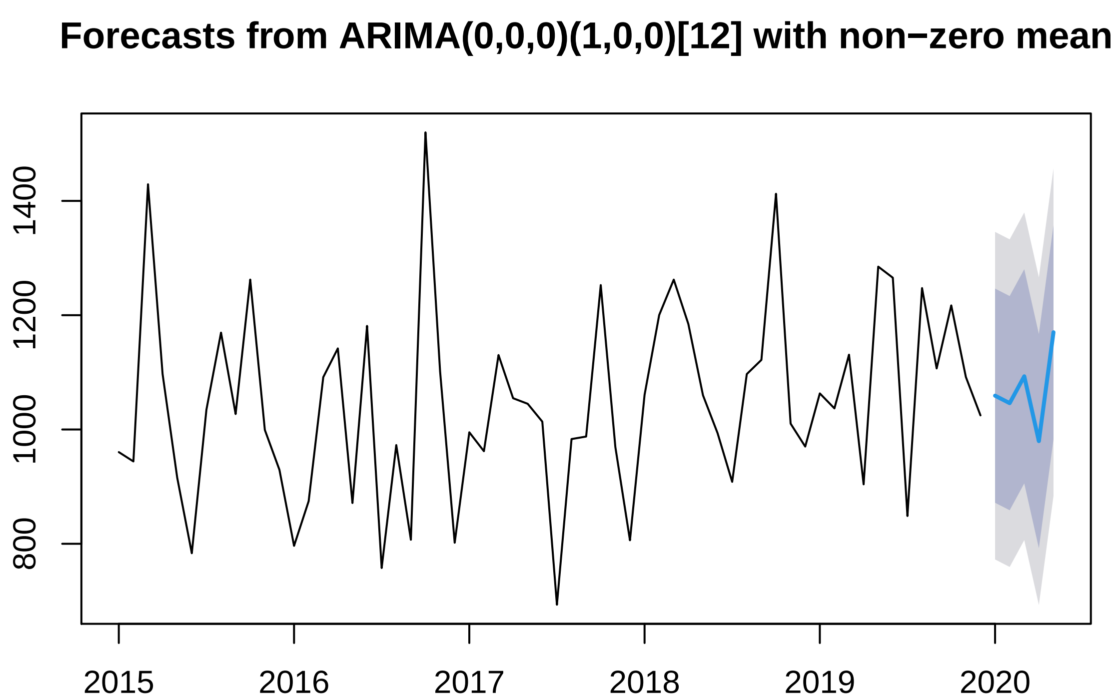
```
## Series: train_tng
```

```
## ARIMA(0,0,0)(1,0,0)[12] with non-zero mean
##
## Coefficients:
##          sar1      mean
##        0.5000  1055.231
## s.e.   0.1131    31.467
##
## sigma^2 estimated as 21383:  log likelihood=-384.96
## AIC=775.91   AICc=776.34   BIC=782.2
```

```r
forecast_ts = forecast(auto_fit, h=5)
forecast_ts
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2020      1059.1806 871.7783 1246.583 772.5735 1345.788
## Feb 2020      1046.0905 858.6882 1233.493 759.4834 1332.698
## Mar 2020      1093.0509 905.6486 1280.453 806.4438 1379.658
## Apr 2020       979.5999 792.1976 1167.002 692.9928 1266.207
## May 2020      1170.0915 982.6892 1357.494 883.4844 1456.699
```

```r
plot(forecast_ts)
```




```r
summary(forecast_ts)
```

```
##
## Forecast method: ARIMA(0,0,0)(1,0,0)[12] with non-zero mean
##
## Model Information:
## Series: train_tng
## ARIMA(0,0,0)(1,0,0)[12] with non-zero mean
##
## Coefficients:
##         sar1      mean
##       0.5000  1055.231
## s.e.  0.1131    31.467
##
## sigma^2 estimated as 21383:  log likelihood=-384.96
## AIC=775.91   AICc=776.34   BIC=782.2
##
## Error measures:
##                    ME    RMSE      MAE       MPE     MAPE      MASE       ACF1
## Training set 1.170595 143.773 113.7024 -1.853976 11.13907 0.8131972 0.01549854
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2020      1059.1806 871.7783 1246.583 772.5735 1345.788
## Feb 2020      1046.0905 858.6882 1233.493 759.4834 1332.698
## Mar 2020      1093.0509 905.6486 1280.453 806.4438 1379.658
## Apr 2020       979.5999 792.1976 1167.002 692.9928 1266.207
## May 2020      1170.0915 982.6892 1357.494 883.4844 1456.699
```

Here we can see that the stl_decomp forecast has the lowest value of MAPE. Therefore, we consider stl_decomp as our forecasting model.