Flight Training

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
```

```
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
```

```
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
```

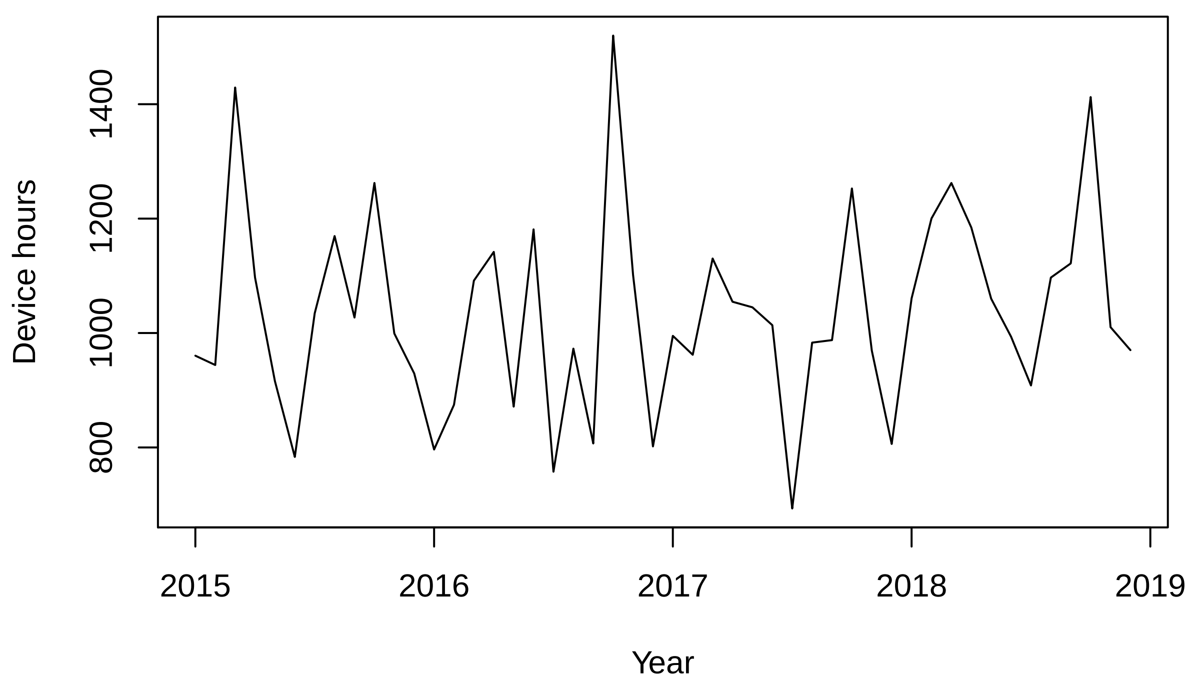
```
## # A tibble: 81 x 1
##    Device_Hrs
##         <dbl>
##  1       960.
##  2       944.
##  3      1429.
##  4      1097
##  5       916.
##  6       783.
##  7      1035.
##  8      1170.
##  9      1027.
## 10      1262.
## # ... with 71 more rows
```

```r
train_tng = ts(data = df_Tng,frequency = 12,start = c(2015, 1),end = c(2018,12))
test_tng = ts(data = df_Tng,frequency = 12,start = c(2019, 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
##          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
```

Plotting the time series

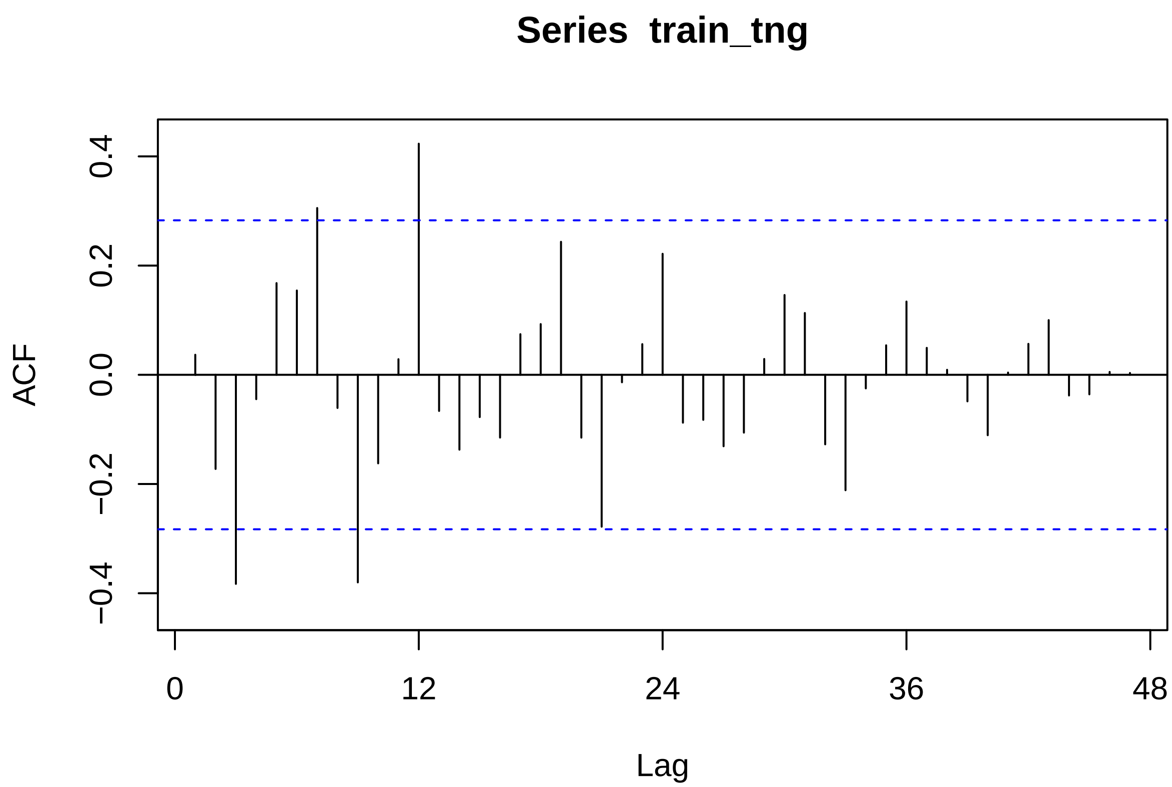
```r
plot(train_tng,xlab = "Year", ylab = "Device hours")
```



##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

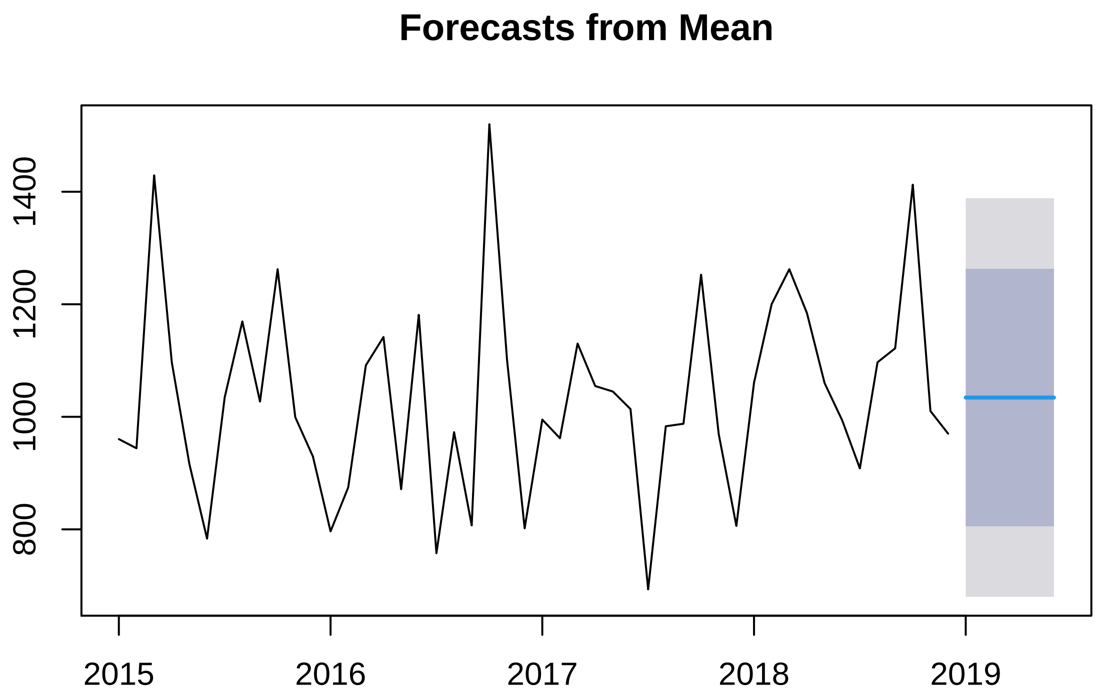
```
Acf(train_tng, lag = 48)
```



Forecasting Methods

Mean Forecast

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



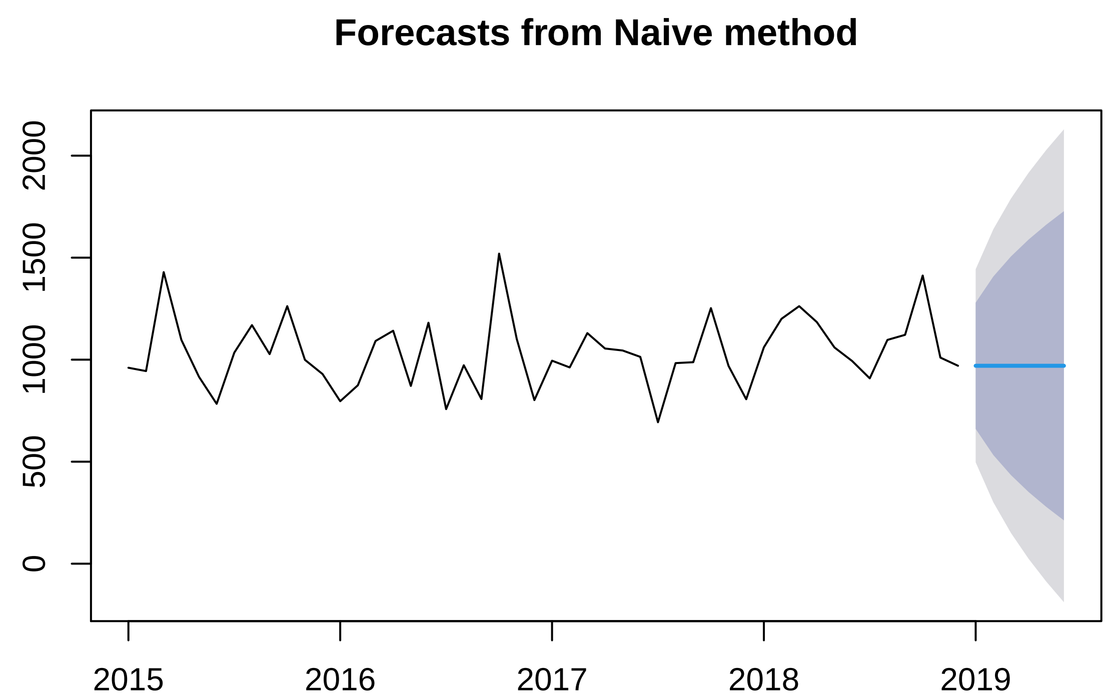

```r
summary(mean_forecast)
```

```
##
## Forecast method: Mean
##
## Model Information:
## $mu
## [1] 1034.242
##
## $mu.se
## [1] 25.14652
##
## $sd
## [1] 174.2202
##
## $bootstrap
## [1] FALSE
##
## $call
## meanf(y = train_tng, h = 6)
##
## attr(,"class")
## [1] "meanf"
##
## Error measures:
```

```
##                          ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 2.368331e-14 172.3958 131.521 -2.751125 13.03757 0.9320331
##                      ACF1
## Training set 0.03671515
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95   Hi 95
## Jan 2019       1034.242 805.4396 1263.045 680.1242 1388.36
## Feb 2019       1034.242 805.4396 1263.045 680.1242 1388.36
## Mar 2019       1034.242 805.4396 1263.045 680.1242 1388.36
## Apr 2019       1034.242 805.4396 1263.045 680.1242 1388.36
## May 2019       1034.242 805.4396 1263.045 680.1242 1388.36
## Jun 2019       1034.242 805.4396 1263.045 680.1242 1388.36
```

Naive Forecast

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



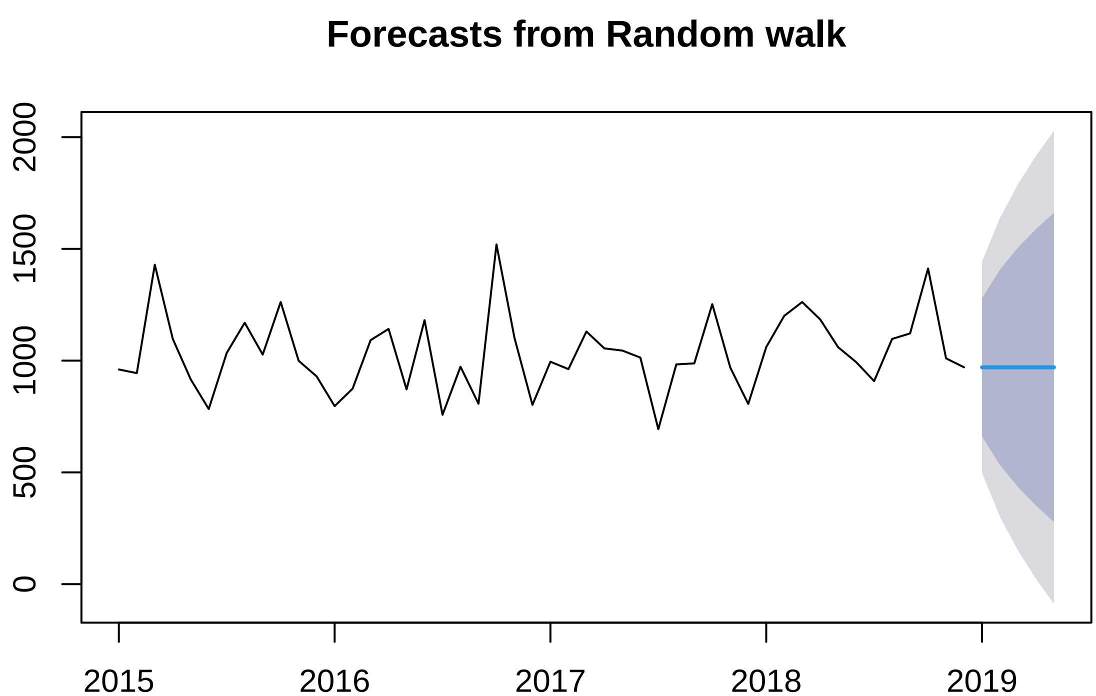

```
summary(naive_forecast)
```

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

```
##
## Model Information:
## Call: naive(y = train_tng, h = 6)
##
## Residual sd: 241.3982
##
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
## Training set 0.206383 241.3982 194.2936 -2.591405 18.87992 1.376876 -0.3923039
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80      Lo 95    Hi 95
## Jan 2019         970.12 660.7557 1279.484  496.98816 1443.252
## Feb 2019         970.12 532.6128 1407.627  301.01053 1639.229
## Mar 2019         970.12 434.2853 1505.955  150.63161 1789.608
## Apr 2019         970.12 351.3914 1588.849   23.85631 1916.384
## May 2019         970.12 278.3604 1661.880  -87.83497 2028.075
## Jun 2019         970.12 212.3354 1727.905 -188.81160 2129.052
```

Random Walk Forecast

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

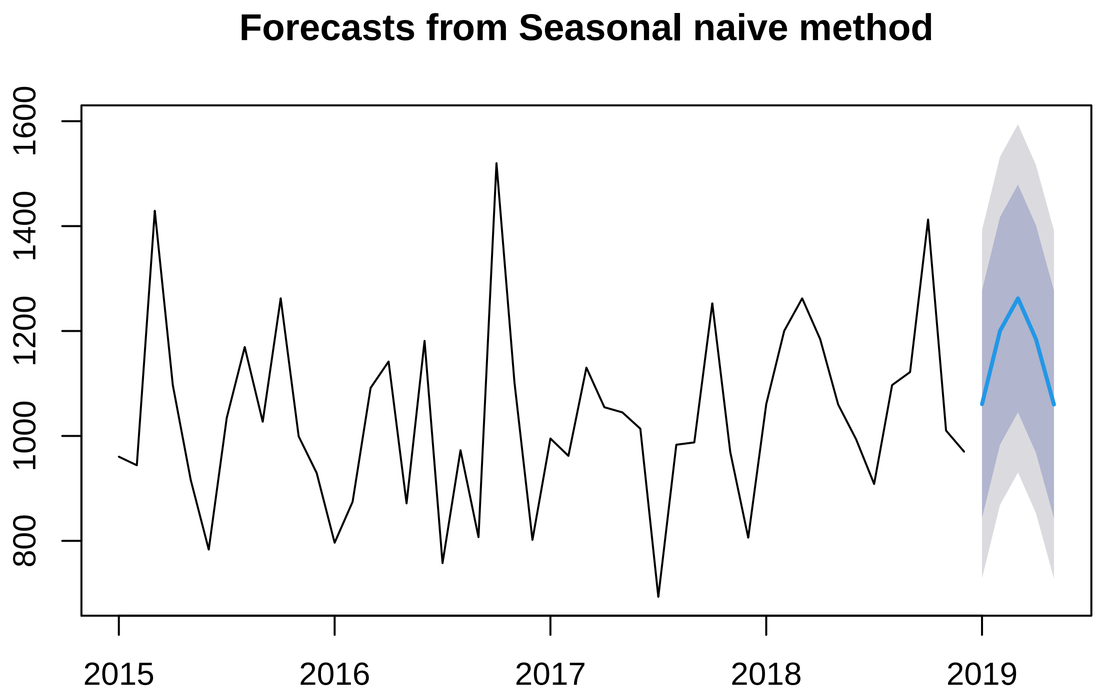


```r
summary(rwf_forecast)
```

```
##
## Forecast method: Random walk
##
## Model Information:
## Call: rwf(y = train_tng, h = 5)
##
## Residual sd: 241.3982
##
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
## Training set 0.206383 241.3982 194.2936 -2.591405 18.87992 1.376876 -0.3923039
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2019         970.12 660.7557 1279.484 496.98816 1443.252
## Feb 2019         970.12 532.6128 1407.627 301.01053 1639.229
## Mar 2019         970.12 434.2853 1505.955 150.63161 1789.608
## Apr 2019         970.12 351.3914 1588.849  23.85631 1916.384
## May 2019         970.12 278.3604 1661.880 -87.83497 2028.075
```

Seasonal Naive Forecast

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

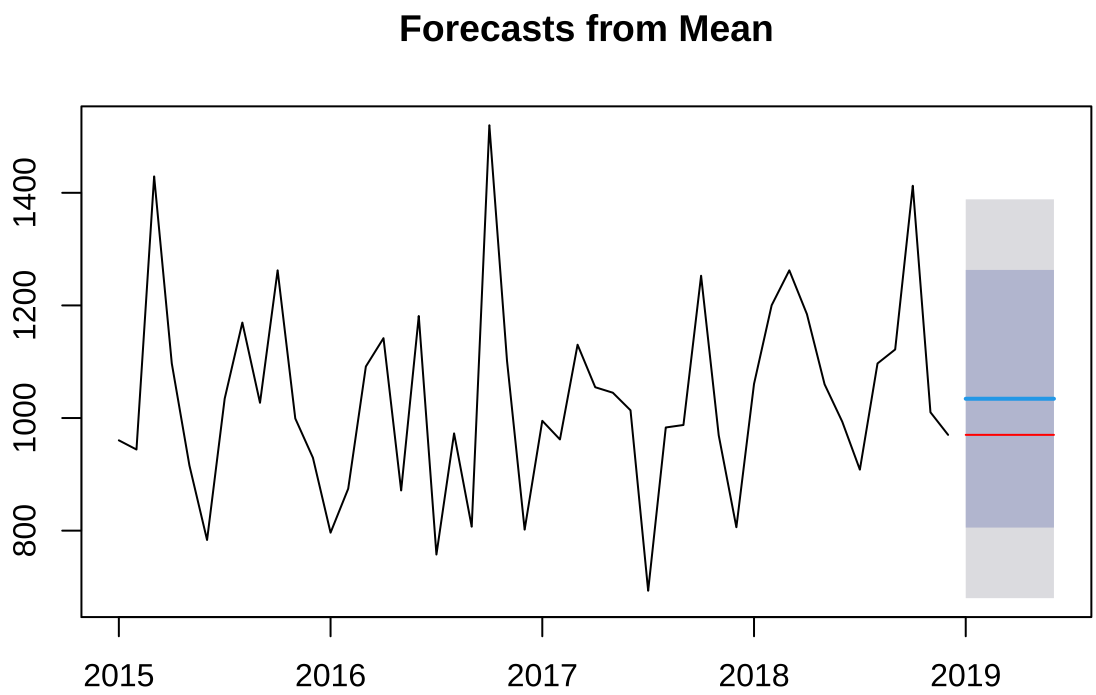



```r
summary(snaive_forecast)
```

```
##
## Forecast method: Seasonal naive method
##
## Model Information:
## Call: snaive(y = train_tng, h = 5)
##
## Residual sd: 169.3329
##
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE MASE      ACF1
## Training set 20.24639 169.3329 141.1119 0.6758389 13.76509    1 0.0114484
##
## Forecasts:
##          Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019        1060.57  843.5611 1277.579 728.6836 1392.456
## Feb 2019        1200.25  983.2411 1417.259 868.3636 1532.136
## Mar 2019        1262.25 1045.2411 1479.259 930.3636 1594.136
## Apr 2019        1184.45  967.4411 1401.459 852.5636 1516.336
## May 2019        1059.92  842.9111 1276.929 728.0336 1391.806
```

Plotting mean and naive forecasting together

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

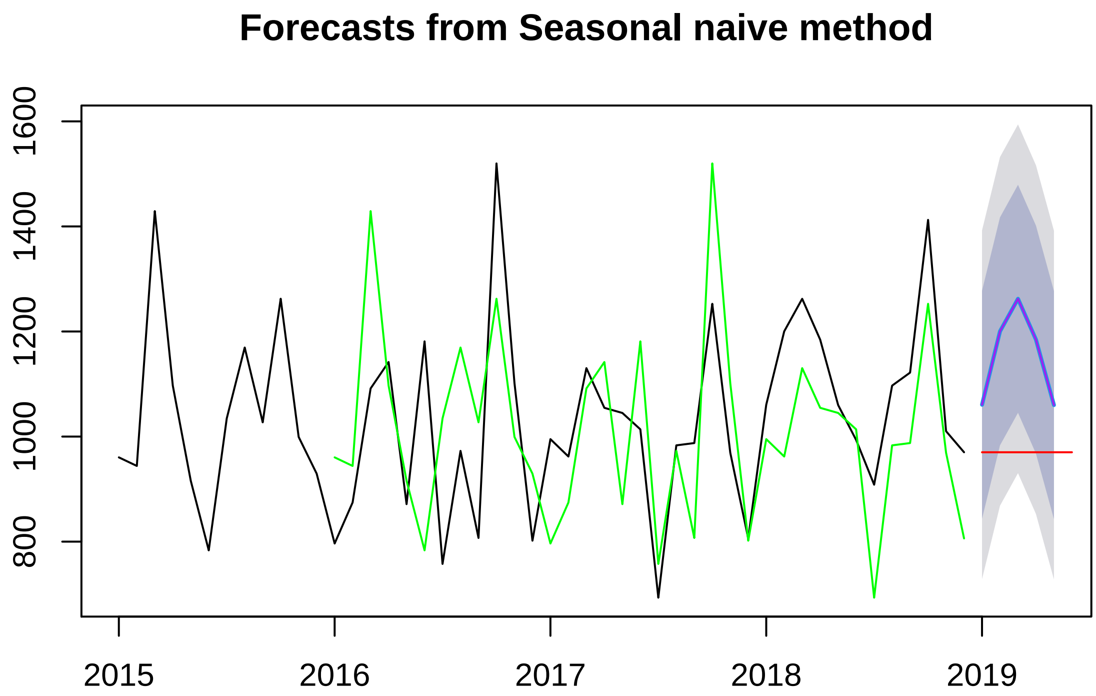



```
attributes(naive_forecast)
```

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

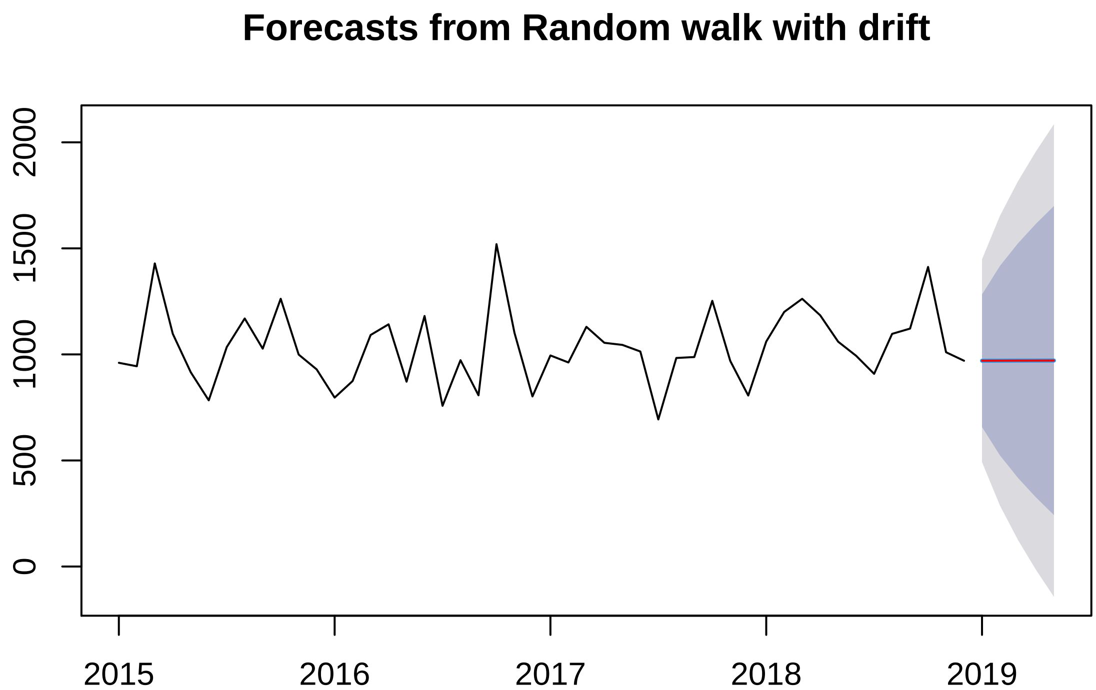
Plotting other attributes

```
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")
```



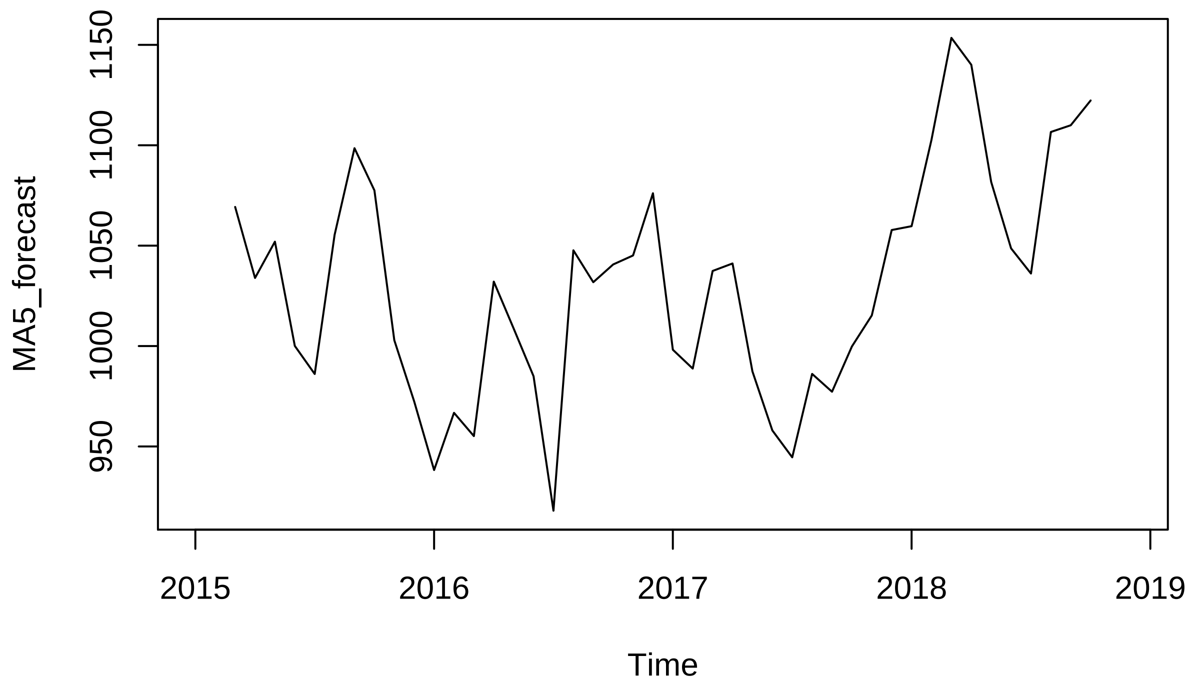
Drift with RWF

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

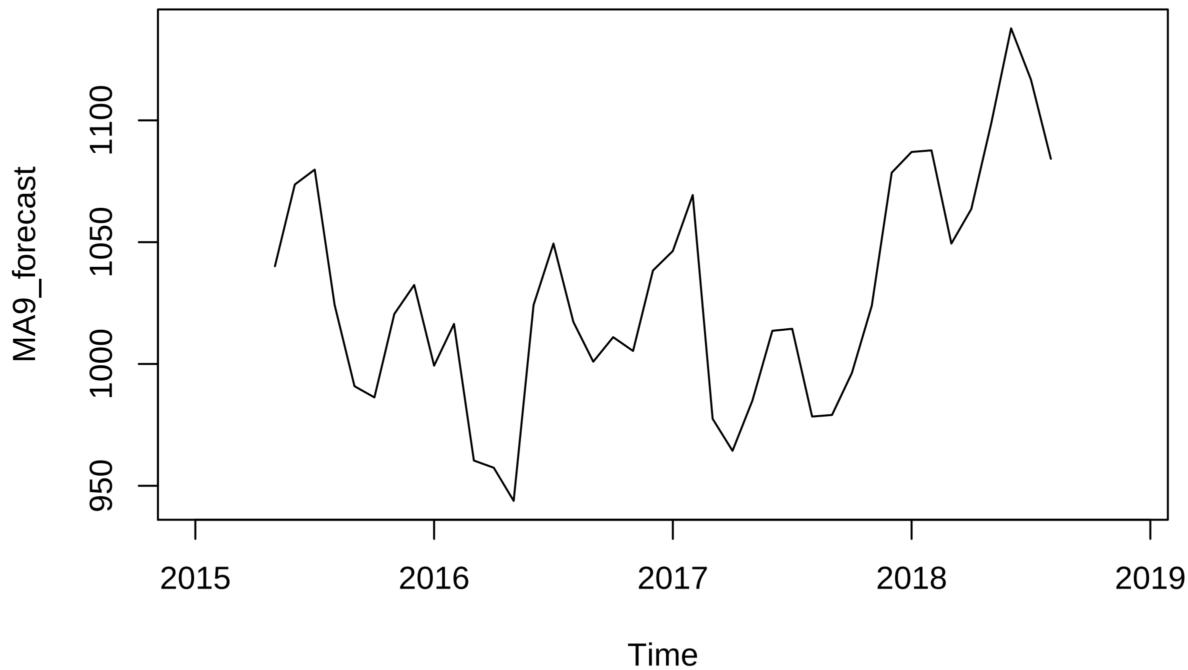


Moving Average Forecast

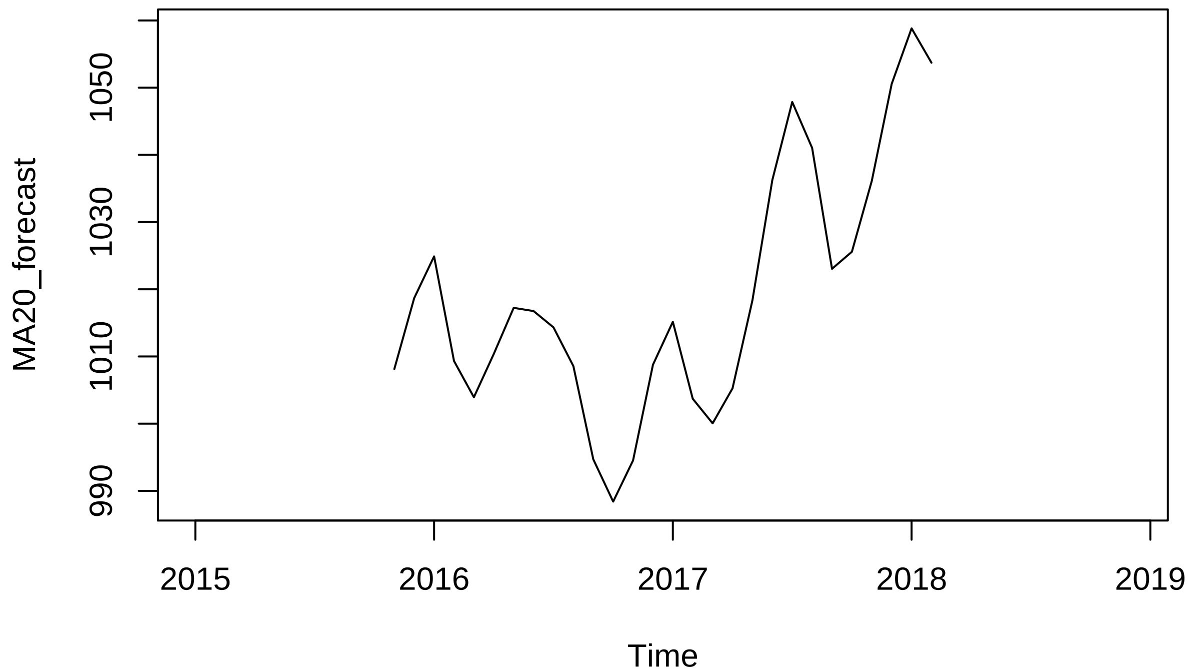
```
MA5_forecast <- ma(train_tng,order=5)
MA9_forecast <- ma(train_tng,order=9)
MA20_forecast <- ma(train_tng,order=20)
plot(MA5_forecast)
```



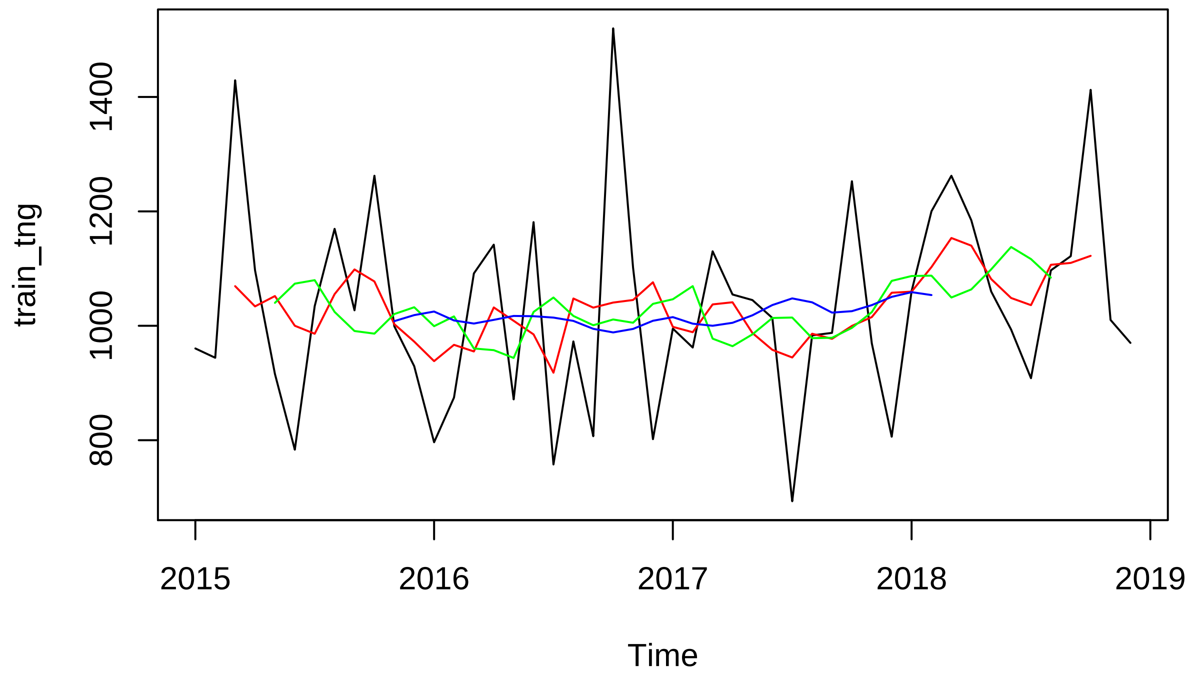

```r
plot(MA9_forecast)
```




```r
plot(MA20_forecast)
```




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




```r
summary(MA5_forecast)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   918.0   987.1  1035.0  1030.6  1062.1  1153.5       4
```

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

ETS

```r
ets(train_tng)
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = train_tng)
##
##   Smoothing parameters:
##     alpha = 0.0523
##     gamma = 0.0012
##
##   Initial states:
```

```
##     l = 1032.3831
##     s = 0.8324 0.9804 1.3176 0.9464 1.0033 0.8335
##            0.9825 0.9549 1.078 1.1775 0.9744 0.9192
##
##   sigma:  0.1249
##
##      AIC     AICc      BIC
## 664.3594 679.3594 692.4274
```

Holt Winters

```r
HoltWinters(train_tng)
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train_tng)
##
## Smoothing parameters:
##  alpha: 0.01670918
##  beta : 1
##  gamma: 0.7138949
##
## Coefficients:
##            [,1]
## a   1090.033935
## b     12.085411
## s1    38.103050
## s2   134.598709
## s3   226.239016
## s4   153.356954
## s5    32.759172
## s6    -1.165754
## s7  -163.195873
## s8    39.529001
## s9    36.894803
## s10  337.423704
## s11  -45.657127
## s12 -135.072455
```

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: 0.04889481
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 1042.894
```

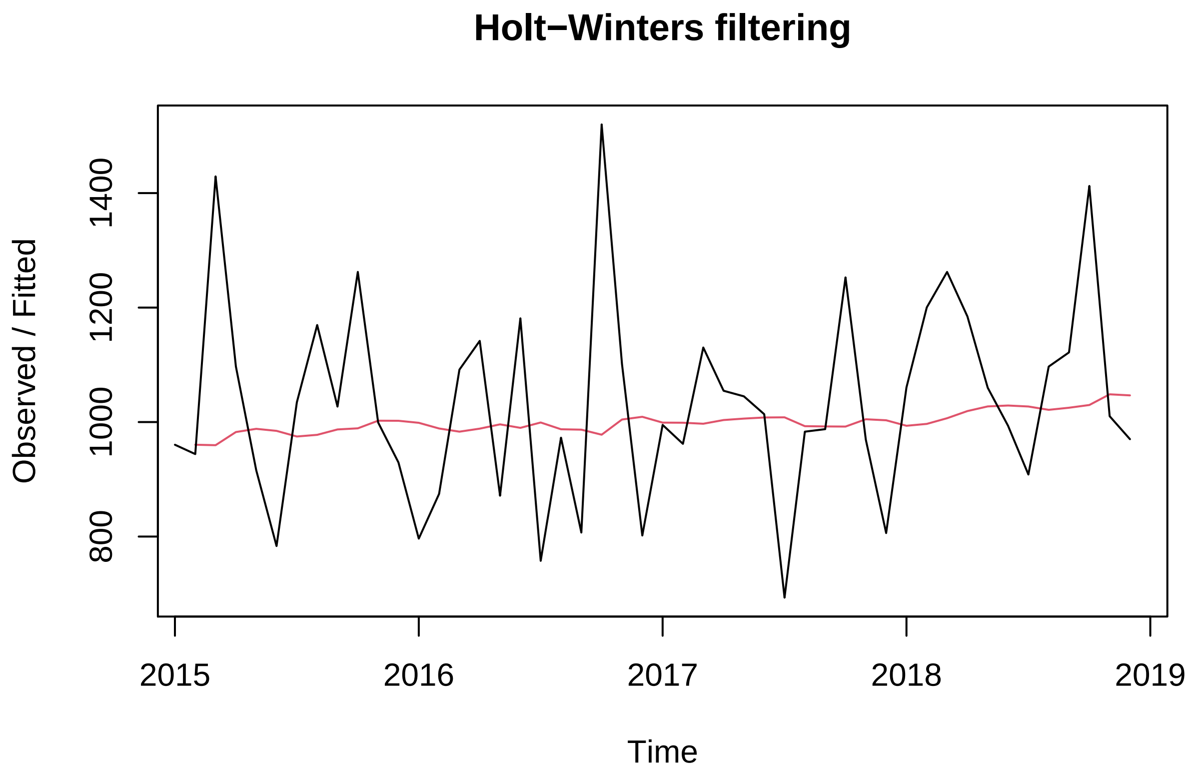
```r
hw_forecast_level = HoltWinters(train_tng,beta=FALSE,gamma=FALSE)
hw_forecast_level
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = train_tng, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.04889481
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 1042.894
```

```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)
```

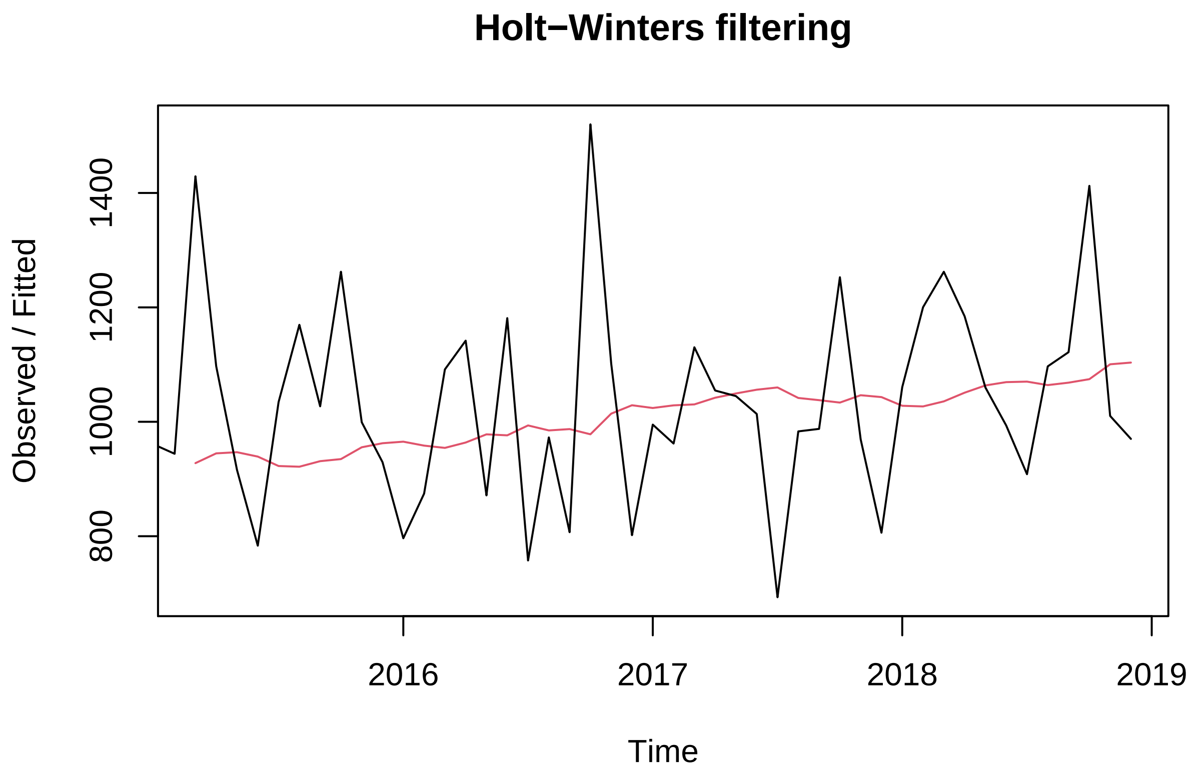



```r
hw_forecast_level$SSE
```

```
## [1] 1534136
```

SSE with Trend but no Seasonlaity

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




```r
hw_forecast_trend
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = train_tng, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.05023002
##  beta : 0.3269799
##  gamma: FALSE
##
## Coefficients:
##          [,1]
## a 1096.799914
## b    5.336964
```

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

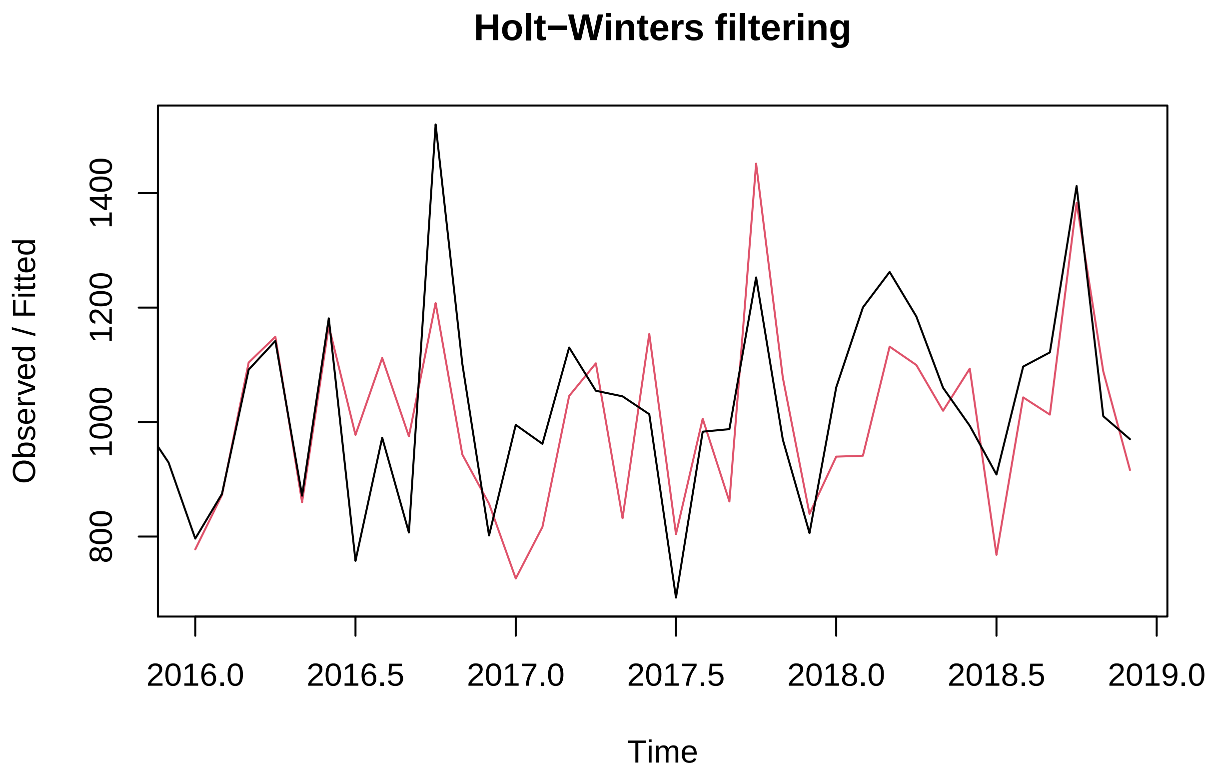
```
## [1] 1617916
```

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: 0.01670918
##  beta : 1
##  gamma: 0.7138949
##
## Coefficients:
##            [,1]
## a   1090.033935
## b     12.085411
## s1    38.103050
## s2   134.598709
## s3   226.239016
## s4   153.356954
## s5    32.759172
## s6    -1.165754
## s7  -163.195873
## s8    39.529001
## s9    36.894803
## s10  337.423704
## s11  -45.657127
## s12 -135.072455
```

```r
plot(hw_forecast_season)
```



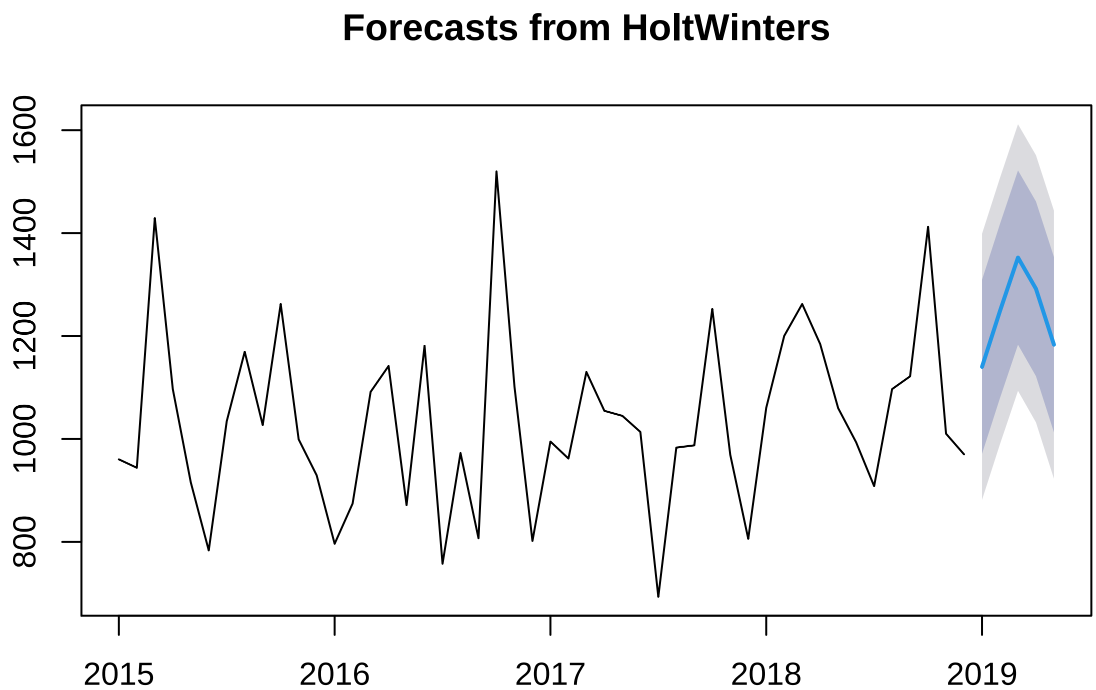

```
hw_forecast_season$SSE
```

```
## [1] 633072
```

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

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2019       1140.222  971.1777 1309.267  881.6908 1398.754
## Feb 2019       1248.803 1079.6644 1417.943  990.1276 1507.479
## Mar 2019       1352.529 1183.1780 1521.880 1093.5288 1611.530
## Apr 2019       1291.733 1122.0048 1461.460 1032.1564 1551.309
## May 2019       1183.220 1012.9059 1353.534  922.7470 1443.693
```

```
plot(hw_forecast_all)
```




```r
accuracy(hw_forecast_all)
```

```
##                    ME     RMSE      MAE      MPE     MAPE      MASE      ACF1
## Training set 25.87229 132.6097 106.0589 1.555031 10.36601 0.7515941 0.1503594
```

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 = 0.0523
##     gamma = 0.0012
##
```

```
##   Initial states:
##     l = 1032.3831
##     s = 0.8324 0.9804 1.3176 0.9464 1.0033 0.8335
##            0.9825 0.9549 1.078 1.1775 0.9744 0.9192
##
##   sigma:  0.1249
##
##      AIC     AICc      BIC
## 664.3594 679.3594 692.4274
```

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

```
## NULL
```

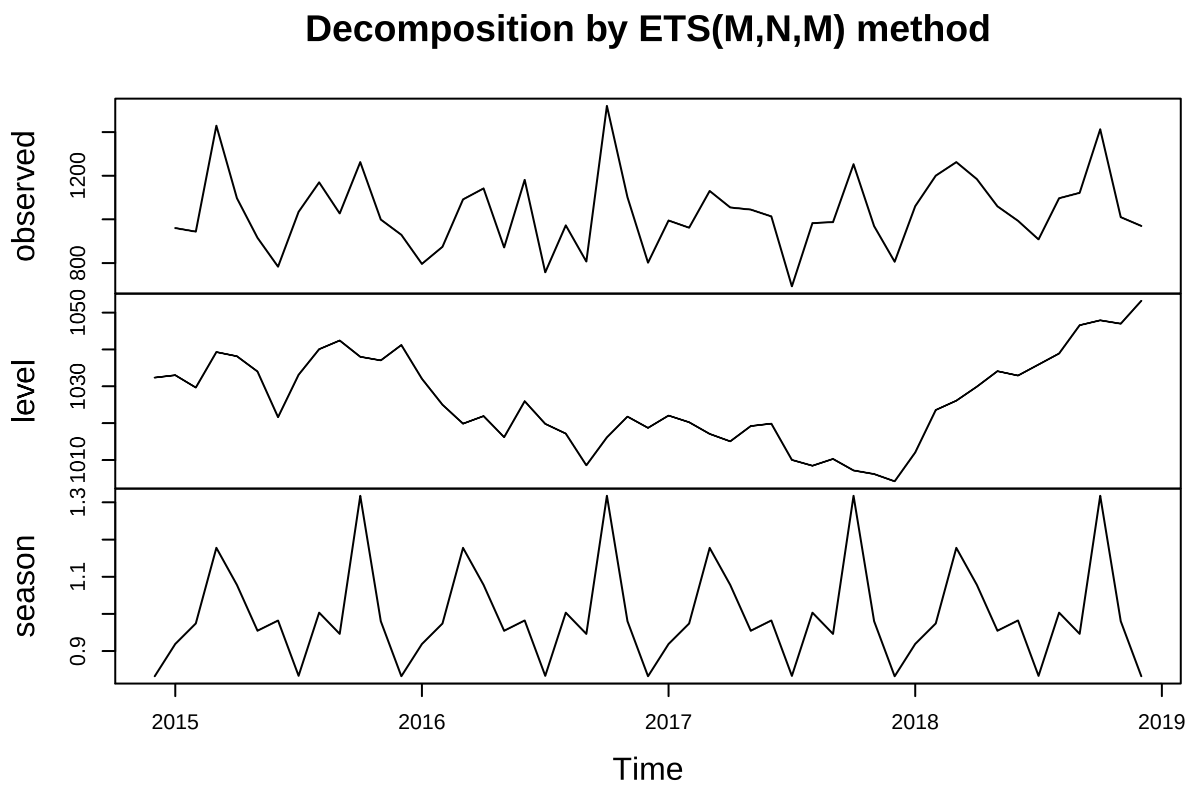
```
attributes(ets_forecast)
```

```
## $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] 11215.5
```

```
plot(ets_forecast)
```

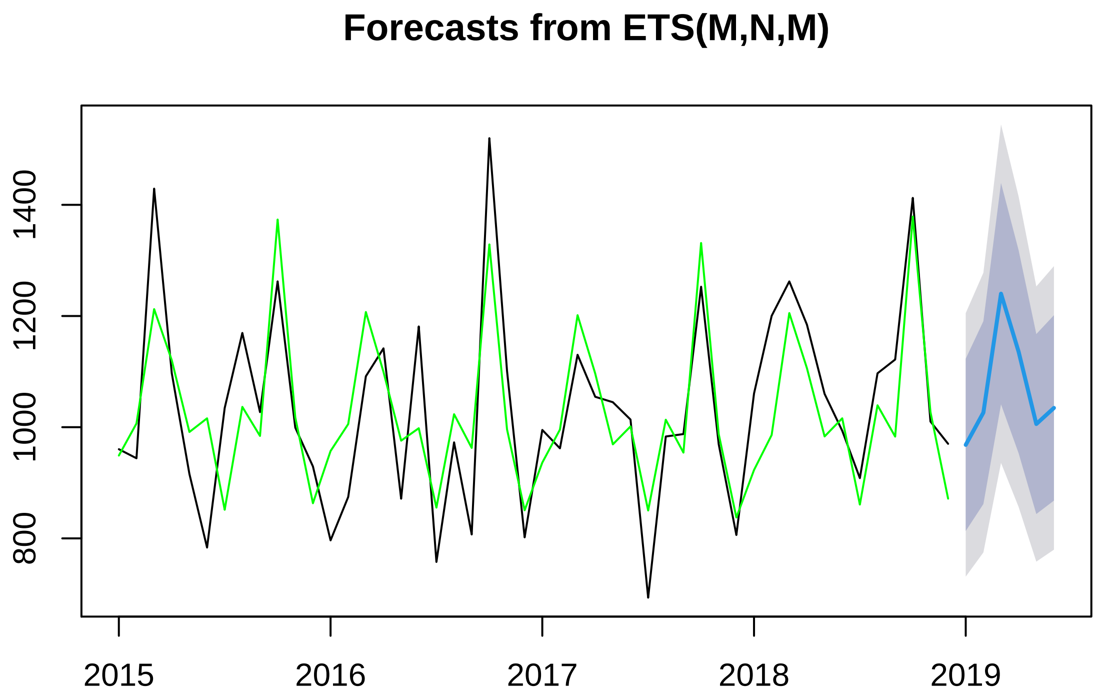


Forecast with Ets

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

```
##          Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019        968.184  813.2283 1123.140 731.1997 1205.168
## Feb 2019       1026.268  861.7881 1190.748 774.7176 1277.818
## Mar 2019       1240.202 1041.1600 1439.245 935.7933 1544.611
## Apr 2019       1135.404  952.9295 1317.879 856.3332 1414.475
## May 2019       1005.668  843.8214 1167.515 758.1447 1253.192
## Jun 2019       1034.692  867.9455 1201.439 779.6752 1289.709
```

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




```r
accuracy(forecast_ets)
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 8.442611 105.9033 87.19015 -0.2842957 8.633811 0.6178793
##                    ACF1
## Training set 0.02967374
```

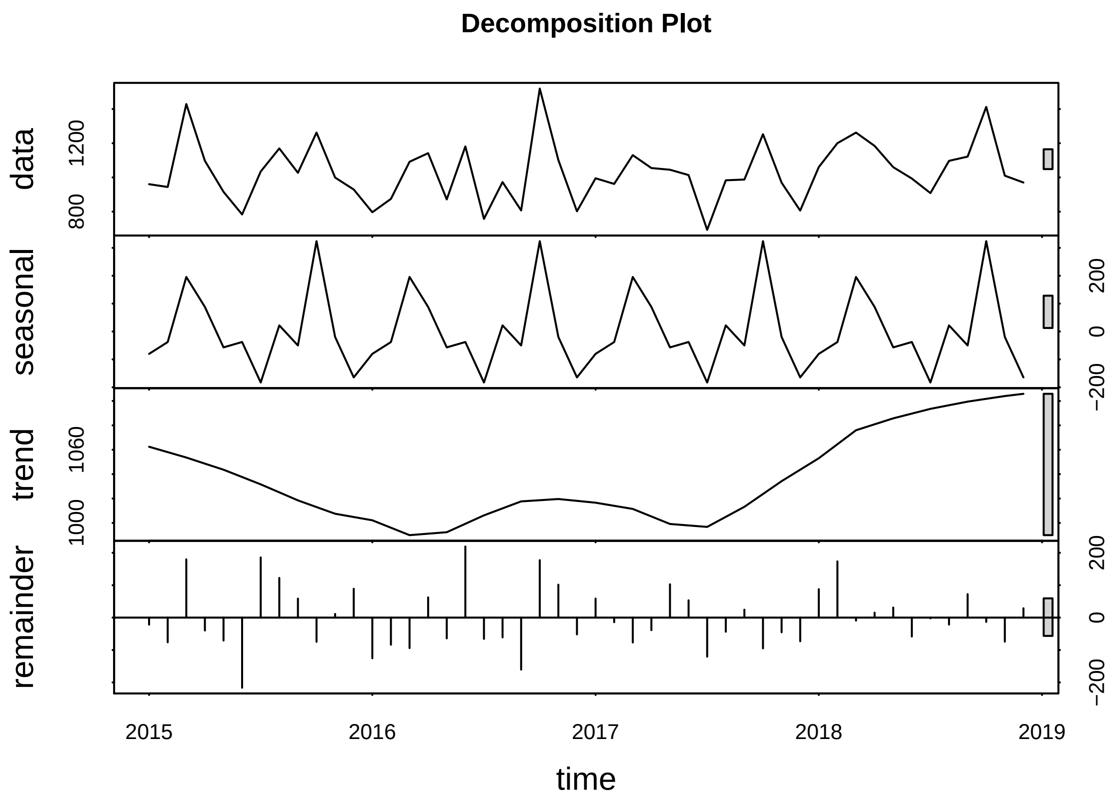
Decomposition

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

```
##  Call:
##  stl(x = train_tng, s.window = "periodic")
##
## Components
##            seasonal     trend   remainder
## Jan 2015  -80.24561 1062.4550  -21.789419
## Feb 2015  -37.81585 1058.0382  -76.142381
## Mar 2015  195.58890 1053.6214  179.909677
## Apr 2015   88.16603 1048.6299  -39.795893
## May 2015  -56.94693 1043.6383  -70.841376
## Jun 2015  -37.66431 1037.6384 -216.524096
```

```
## Jul 2015 -182.87953 1031.6385  185.761019
## Aug 2015   21.95281 1025.0648  122.482424
## Sep 2015  -50.09491 1018.4910   58.683883
## Oct 2015  324.02207 1013.0154  -74.717425
## Nov 2015  -19.56858 1007.5397   11.278884
## Dec 2015 -164.51421 1004.8528   89.081460
## Jan 2016  -80.24561 1002.1658 -125.500201
## Feb 2016  -37.81585  996.0810  -83.715141
## Mar 2016  195.58890  989.9962  -94.035062
## Apr 2016   88.16603  991.1921   62.481865
## May 2016  -56.94693  992.3881  -64.081122
## Jun 2016  -37.66431  999.3013  219.573013
## Jul 2016 -182.87953 1006.2145  -65.745018
## Aug 2016   21.95281 1011.9359  -61.158694
## Sep 2016  -50.09491 1017.6572 -160.542316
## Oct 2016  324.02207 1018.6284  177.269570
## Nov 2016  -19.56858 1019.5995  101.639074
## Dec 2016 -164.51421 1018.0979  -51.753707
## Jan 2017  -80.24561 1016.5963   58.739275
## Feb 2017  -37.81585 1014.0360  -14.220179
## Mar 2017  195.58890 1011.4757  -76.824613
## Apr 2017   88.16603 1005.3158  -38.771854
## May 2017  -56.94693  999.1559  102.740991
## Jun 2017  -37.66431  997.9515   53.442775
## Jul 2017 -182.87953  996.7471 -120.537606
## Aug 2017   21.95281 1004.9838  -43.686579
## Sep 2017  -50.09491 1013.2204   24.514502
## Oct 2017  324.02207 1023.7274  -95.059453
## Nov 2017  -19.56858 1034.2344  -45.355789
## Dec 2017 -164.51421 1043.6364  -73.022142
## Jan 2018  -80.24561 1053.0383   87.777269
## Feb 2018  -37.81585 1064.5129  173.552900
## Mar 2018  195.58890 1075.9876   -9.326450
## Apr 2018   88.16603 1080.8642   15.419734
## May 2018  -56.94693 1085.7409   31.126005
## Jun 2018  -37.66431 1089.6332  -58.418844
## Jul 2018 -182.87953 1093.5254   -2.275859
## Aug 2018   21.95281 1096.4897  -21.512523
## Sep 2018  -50.09491 1099.4540   72.390867
## Oct 2018  324.02207 1101.7365  -13.288588
## Nov 2018  -19.56858 1104.0190  -74.200425
## Dec 2018 -164.51421 1105.8608   28.773428
```

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




```r
attributes(stl_decomp)
```

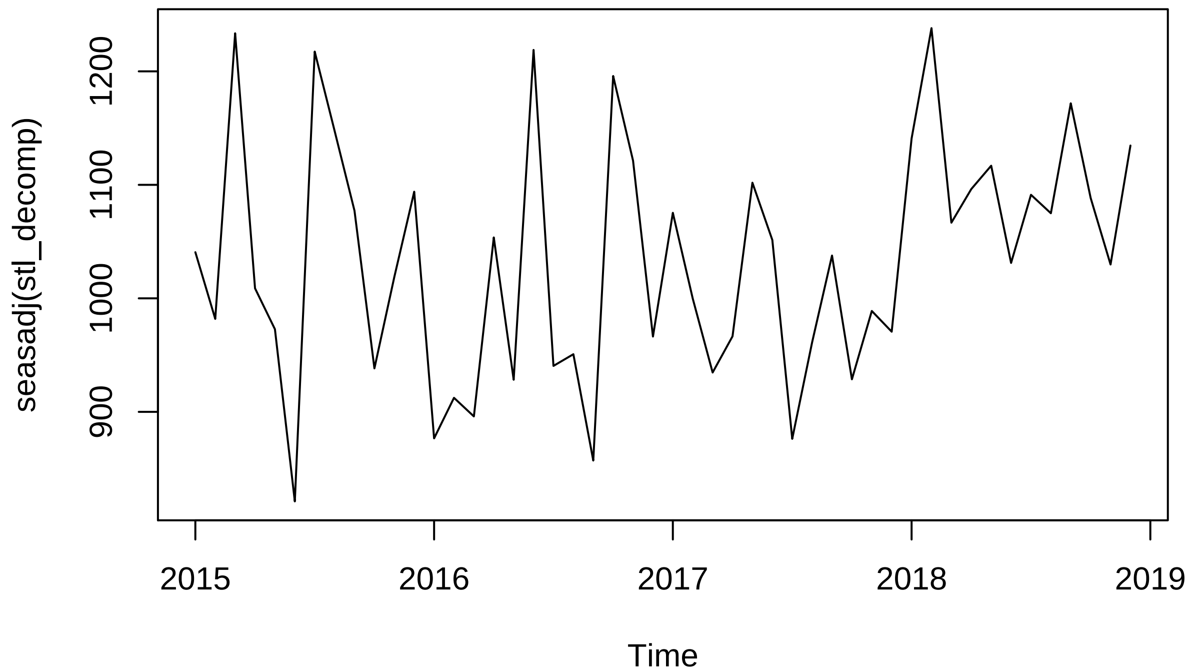
```
## $names
## [1] "time.series" "weights"     "call"        "win"         "deg"
## [6] "jump"        "inner"       "outer"
##
## $class
## [1] "stl"
```

Seasonal Adjustment

```r
seasadj(stl_decomp)
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2015 1040.6656  981.8958 1233.5311 1008.8340  972.7969  821.1143 1217.3995
## 2016  876.6656  912.3658  895.9611 1053.6740  928.3069 1218.8743  940.4695
## 2017 1075.3356  999.8158  934.6511  966.5440 1101.8969 1051.3943  876.2095
## 2018 1140.8156 1238.0658 1066.6611 1096.2840 1116.8669 1031.2143 1091.2495
##            Aug       Sep       Oct       Nov       Dec
## 2015 1147.5472 1077.1749  938.2979 1018.8186 1093.9342
## 2016  950.7772  857.1149 1195.8979 1121.2386  966.3442
## 2017  961.2972 1037.7349  928.6679  988.8786  970.6142
## 2018 1074.9772 1171.8449 1088.4479 1029.8186 1134.6342
```

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

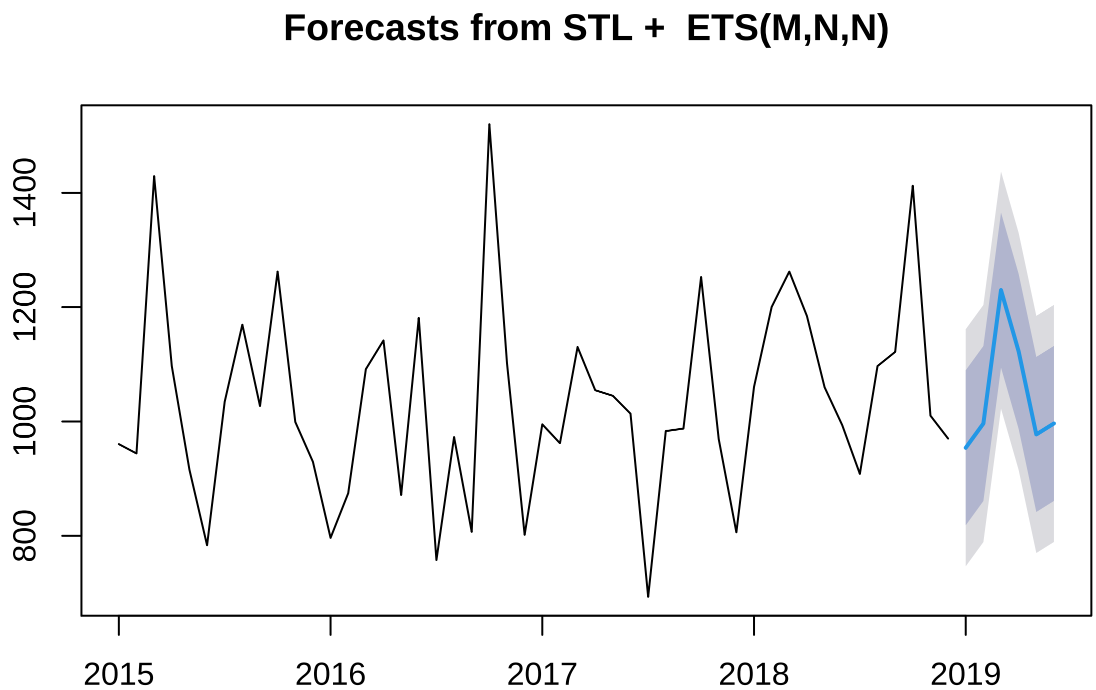


Default Period Forecast

```r
f_stl = forecast(stl_decomp,h = 6)
f_stl
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2019       954.0297  818.4401 1089.619  746.6632 1161.396
## Feb 2019       996.4595  860.8698 1132.049  789.0930 1203.826
## Mar 2019      1229.8642 1094.2746 1365.454 1022.4977 1437.231
## Apr 2019      1122.4414  986.8517 1258.031  915.0749 1329.808
## May 2019       977.3284  841.7387 1112.918  769.9619 1184.695
## Jun 2019       996.6110  861.0214 1132.201  789.2445 1203.978
```

```r
plot(f_stl)
```




```r
accuracy(f_stl)
```

```
##                         ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.00529333 103.5708 86.01888 -1.071521 8.579715 0.609579
##                    ACF1
## Training set 0.05904374
```

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