Tugas MPDW Pemulusan

Rakesha Putra Antique Yusuf - G1401221056

Library / Packages

```
library("forecast")
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

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

```
library("graphics")
library("TTR")
```

```
## Warning: package 'TTR' was built under R version 4.3.3
```

```
library("TSA")
```

```
## Warning: package 'TSA' was built under R version 4.3.3
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

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

Impor Data

```
data1 <- readxl::read_xlsx("C:/Users/RAKESHA/Downloads/DATA CURAH HUJAN (MPDW).xlsx")
data1
```

```
## # A tibble: 121 x 2
##    Tanggal `Akumulasi Hujan`
##      <dbl>             <dbl>
##  1       1             0.102
##  2       2             0.41
##  3       3             0.45
##  4       4             0.02
##  5       5             0.31
##  6       6             0.179
##  7       7             0.290
##  8       8             0.0544
##  9       9             0.061
## 10      10             0.0171
## # i 111 more rows
```

Eksplorasi Data

```
str(data1)
```

```
## tibble [121 x 2] (S3: tbl_df/tbl/data.frame)
##  $ Tanggal        : num [1:121] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Akumulasi Hujan: num [1:121] 0.102 0.41 0.45 0.02 0.31 ...
```

```
dim(data1)
```

```
## [1] 121   2
```

Mengubah data

```
data1.ts <- ts(data1$`Akumulasi Hujan`)
```

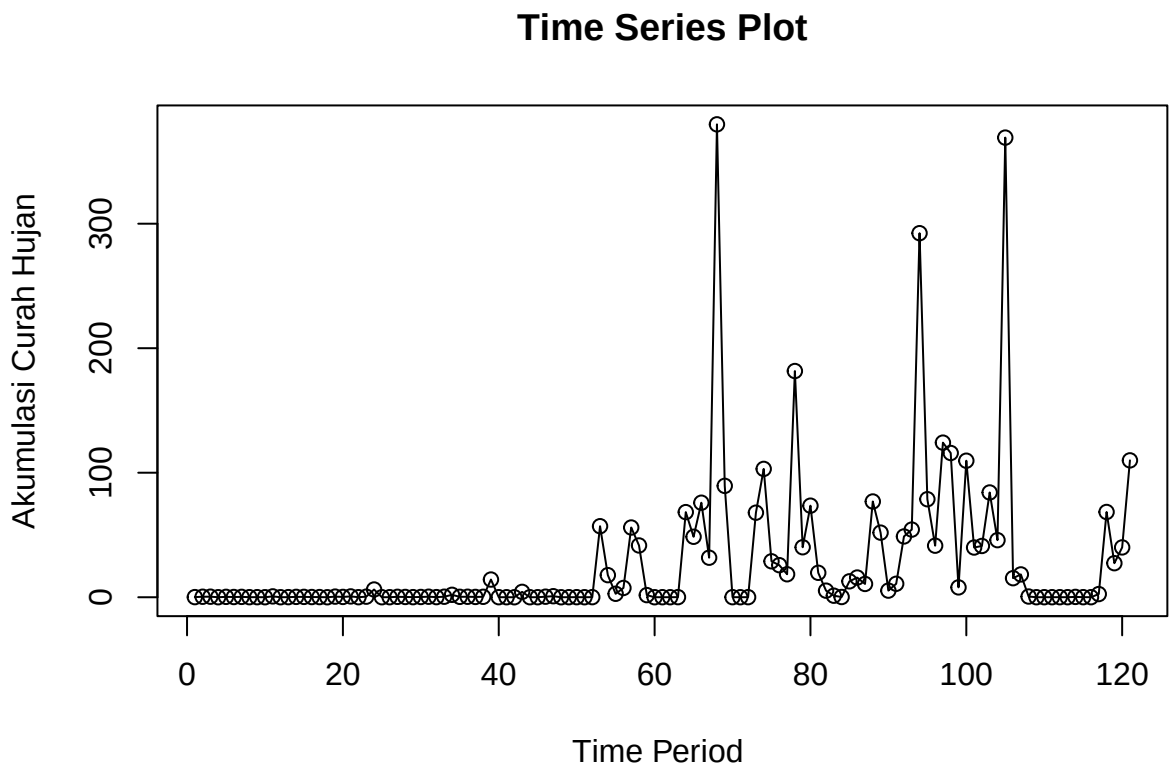
Menampilkan ringkasan data

```
summary(data1.ts)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0018   0.0620   0.5029  28.1145  39.9343 379.7904
```

Membuat plot data deret waktu

```
ts.plot(data1.ts, xlab="Time Period ", ylab="Akumulasi Curah Hujan",
        main = "Time Series Plot")
points(data1.ts)
```



Single Moving Average & Double Moving Average

Pembagian Data

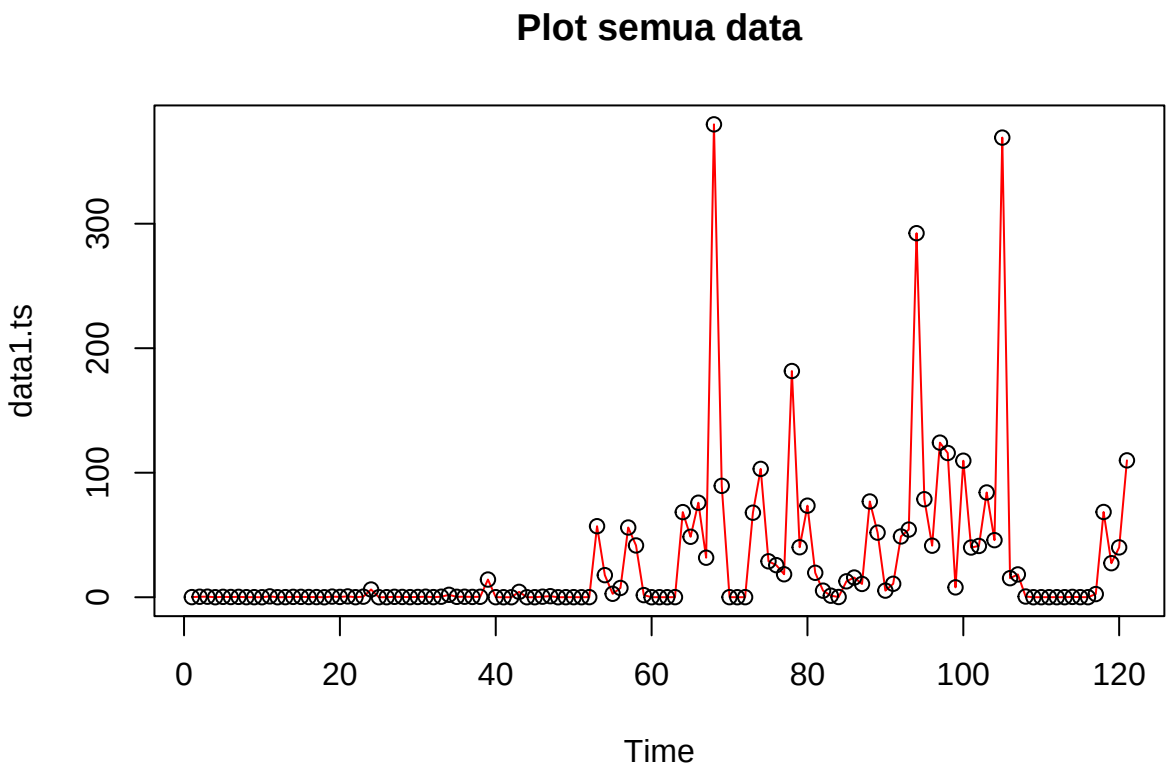
Pembagian data latih dan data uji dilakukan dengan perbandingan 90% data latih dan 10% data uji.

```
#membagi data latih dan data uji
training_ma <- data1[1:109,]
testing_ma <- data1[110:121,]
train_ma.ts <- ts(training_ma$`Akumulasi Hujan`)
test_ma.ts <- ts(testing_ma$`Akumulasi Hujan`)
```

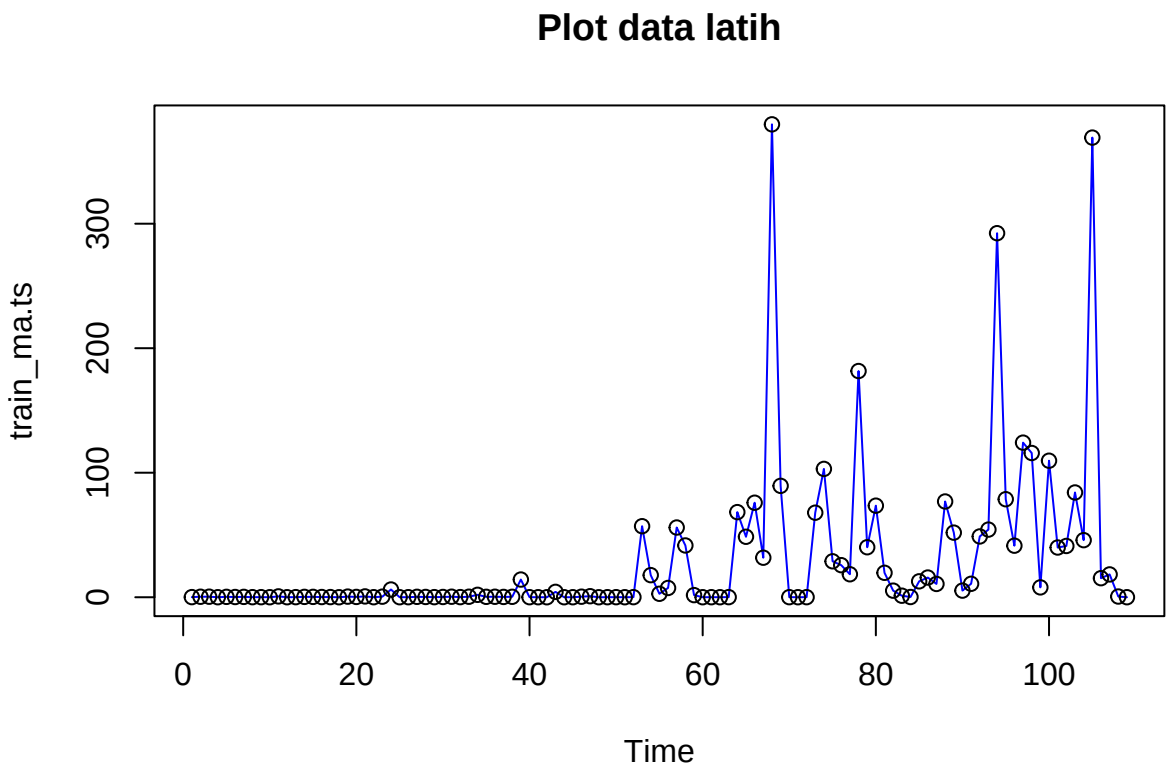
Eksplorasi Data

Eksplorasi data dilakukan pada keseluruhan data, data latih serta data uji menggunakan plot data deret waktu.

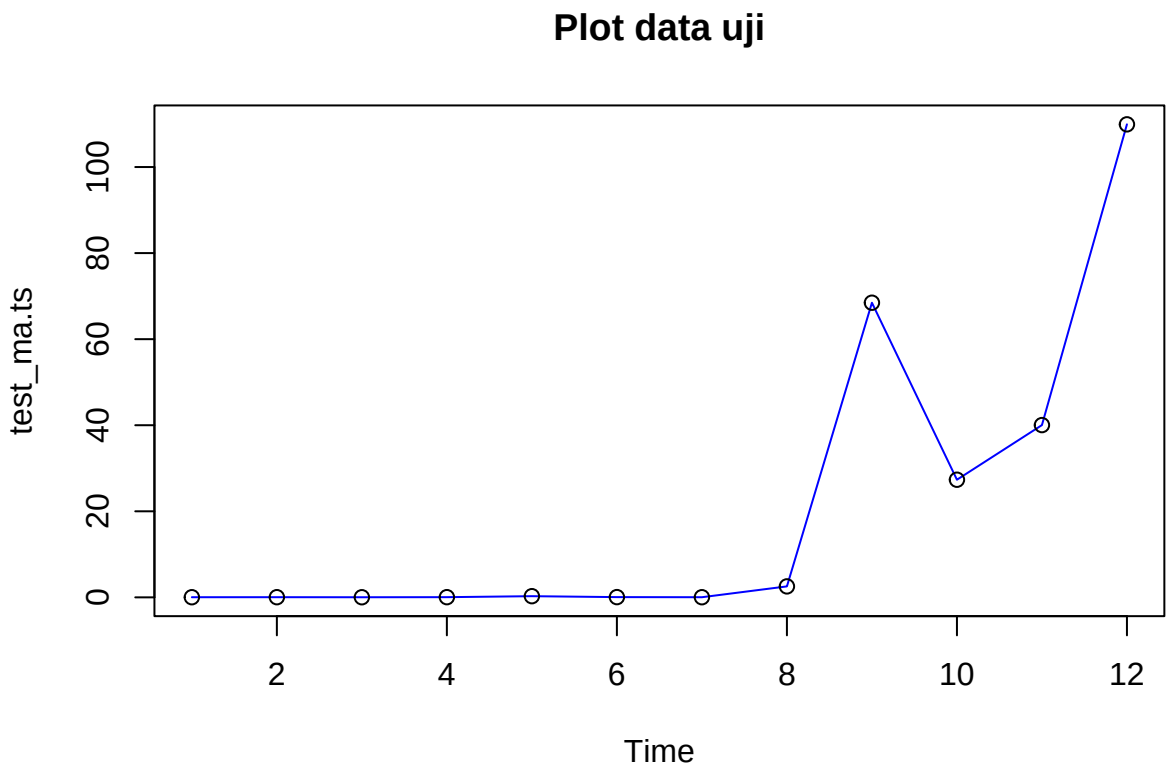
```
#eksplorasi keseluruhan data
plot(data1.ts, col="red",main="Plot semua data")
points(data1.ts)
```




```
#eksplorasi data latih
plot(train_ma.ts, col="blue",main="Plot data latih")
points(train_ma.ts)
```




```
#eksplorasi data uji
plot(test_ma.ts, col="blue",main="Plot data uji")
points(test_ma.ts)
```

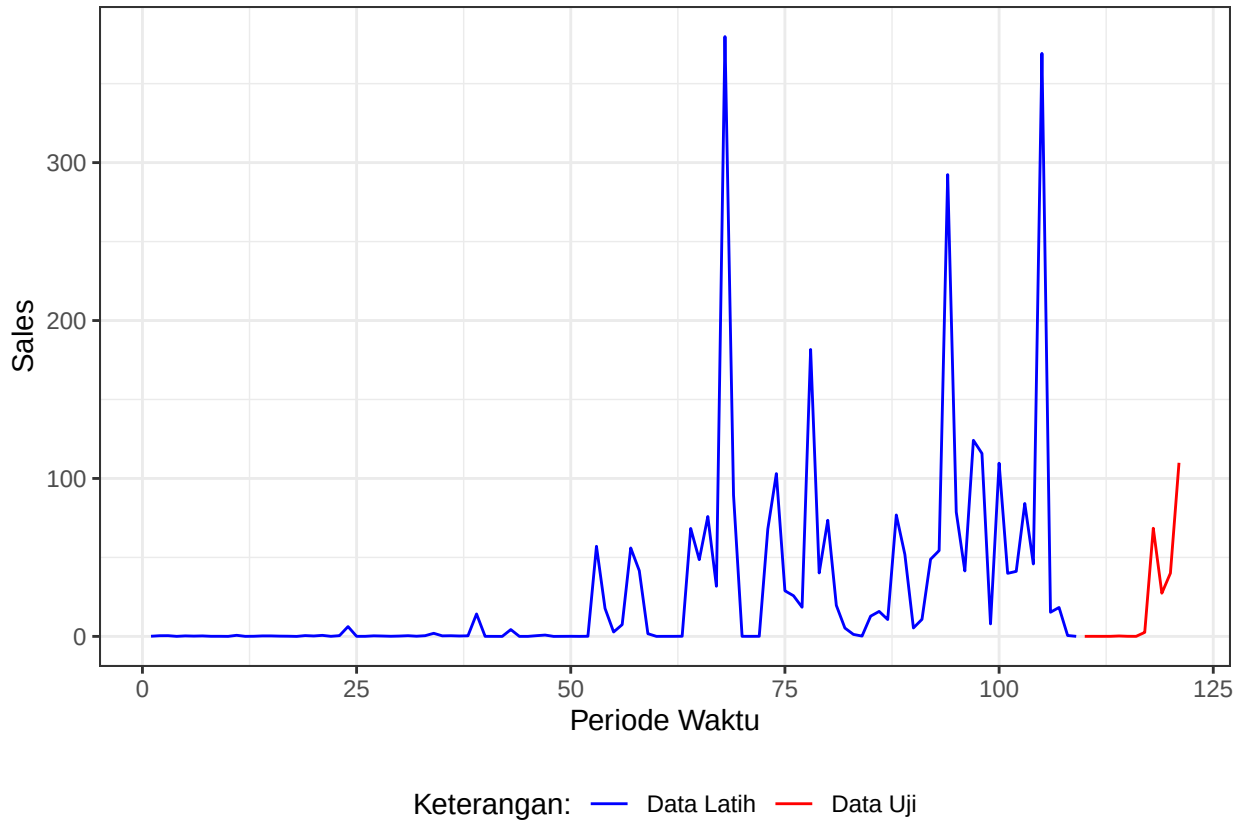


Eksplorasi data juga dapat dilakukan menggunakan package `ggplot2` dengan terlebih dahulu memanggil library *package* `ggplot2`.

```
#Eksplorasi dengan GGPLOT
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
ggplot() +
  geom_line(data = training_ma, aes(x = Tanggal, y = `Akumulasi Hujan`, col = "Data Latih")) +
  geom_line(data = testing_ma, aes(x = Tanggal, y = `Akumulasi Hujan`, col = "Data Uji")) +
  labs(x = "Periode Waktu", y = "Sales", color = "Legend") +
  scale_colour_manual(name="Keterangan:", breaks = c("Data Latih", "Data Uji"),
                      values = c("blue", "red")) +
  theme_bw() + theme(legend.position = "bottom",
                     plot.caption = element_text(hjust=0.5, size=12))
```



Single Moving Average (SMA)

Ide dasar dari Single Moving Average (SMA) adalah data suatu periode dipengaruhi oleh data periode sebelumnya. Metode pemulusan ini cocok digunakan untuk pola data stasioner atau konstan. Prinsip dasar metode pemulusan ini adalah data pemulusan pada periode ke-t merupakan rata rata dari m buah data pada periode ke-t hingga periode ke (t-m+1). Data pemulusan pada periode ke-t selanjutnya digunakan sebagai nilai peramalan pada periode ke t+1

Pemulusan menggunakan metode SMA dilakukan dengan fungsi `SMA()`. Dalam hal ini akan dilakukan pemulusan dengan parameter `m=1`.

```
data.sma<-SMA(train_ma.ts, n=1)
data.sma
```

```
## Time Series:
## Start = 1
## End = 109
## Frequency = 1
##   [1]   0.1019   0.4100   0.4500   0.0200   0.3100   0.1790   0.2901   0.0544
##   [9]   0.0610   0.0171   0.6863   0.0210   0.0831   0.3000   0.3000   0.1642
##  [17]   0.1190   0.0018   0.5029   0.2604   0.6587   0.0430   0.4627   6.2075
##  [25]   0.0210   0.0430   0.3448   0.2300   0.0786   0.2120   0.4300   0.1100
##  [33]   0.4265   1.9465   0.3400   0.4120   0.2410   0.3677  14.2121   0.0310
##  [41]   0.0122   0.0021   4.3156   0.0065   0.0130   0.4534   0.8348   0.0021
##  [49]   0.0302   0.0620   0.0180   0.0785  57.0441  17.7483   2.7758   7.4453
```

```
##  [57]  55.9917  41.5794   1.7026   0.0060   0.0330   0.0260   0.0730  68.3487
##  [65]  48.5761  75.9108  31.7568 379.7904  89.4546   0.0510   0.0210   0.0555
##  [73]  67.9166 103.0704  28.8643  25.7484  18.4597 181.6665  40.1587  73.5338
##  [81]  19.6265   5.2749   1.2463   0.1907  12.7869  15.8216  10.6741  76.8945
##  [89]  51.8810   5.3328  10.8060  48.8230  54.3312 292.3764  78.7232  41.4251
##  [97] 124.1762 115.8449   7.9111 109.6561  39.9343  41.1686  84.1471  45.8479
## [105] 369.1453  15.2997  18.3233   0.5688   0.0220
```

Data pemulusan pada periode ke-t selanjutnya digunakan sebagai nilai peramalan pada periode ke t+1 sehingga hasil peramalan 1 periode kedepan adalah sebagai berikut.

```
data.ramal<-c(NA,data.sma)
data.ramal #forecast 1 periode ke depan
```

```
##   [1]       NA   0.1019   0.4100   0.4500   0.0200   0.3100   0.1790   0.2901
##   [9]   0.0544   0.0610   0.0171   0.6863   0.0210   0.0831   0.3000   0.3000
##  [17]   0.1642   0.1190   0.0018   0.5029   0.2604   0.6587   0.0430   0.4627
##  [25]   6.2075   0.0210   0.0430   0.3448   0.2300   0.0786   0.2120   0.4300
##  [33]   0.1100   0.4265   1.9465   0.3400   0.4120   0.2410   0.3677  14.2121
##  [41]   0.0310   0.0122   0.0021   4.3156   0.0065   0.0130   0.4534   0.8348
##  [49]   0.0021   0.0302   0.0620   0.0180   0.0785  57.0441  17.7483   2.7758
##  [57]   7.4453  55.9917  41.5794   1.7026   0.0060   0.0330   0.0260   0.0730
##  [65]  68.3487  48.5761  75.9108  31.7568 379.7904  89.4546   0.0510   0.0210
##  [73]   0.0555  67.9166 103.0704  28.8643  25.7484  18.4597 181.6665  40.1587
##  [81]  73.5338  19.6265   5.2749   1.2463   0.1907  12.7869  15.8216  10.6741
##  [89]  76.8945  51.8810   5.3328  10.8060  48.8230  54.3312 292.3764  78.7232
##  [97]  41.4251 124.1762 115.8449   7.9111 109.6561  39.9343  41.1686  84.1471
## [105]  45.8479 369.1453  15.2997  18.3233   0.5688   0.0220
```

Selanjutnya akan dilakukan peramalan sejumlah data uji yaitu 24 periode. Pada metode SMA, hasil peramalan 24 periode ke depan akan bernilai sama dengan hasil peramalan 1 periode kedepan. Dalam hal ini akan dilakukan pengguabungan data aktual train, data hasil pemulusan dan data hasil ramalan 24 periode kedepan.

```
data.gab<-cbind(aktual=c(train_ma.ts,rep(NA,24)),pemulusan=c(data.sma,rep(NA,24)),ramalan=c(data.ramal,
data.gab
```

```
##          aktual pemulusan  ramalan
##   [1,]   0.1019    0.1019       NA
##   [2,]   0.4100    0.4100   0.1019
##   [3,]   0.4500    0.4500   0.4100
##   [4,]   0.0200    0.0200   0.4500
##   [5,]   0.3100    0.3100   0.0200
##   [6,]   0.1790    0.1790   0.3100
##   [7,]   0.2901    0.2901   0.1790
##   [8,]   0.0544    0.0544   0.2901
##   [9,]   0.0610    0.0610   0.0544
##  [10,]   0.0171    0.0171   0.0610
##  [11,]   0.6863    0.6863   0.0171
##  [12,]   0.0210    0.0210   0.6863
##  [13,]   0.0831    0.0831   0.0210
##  [14,]   0.3000    0.3000   0.0831
```

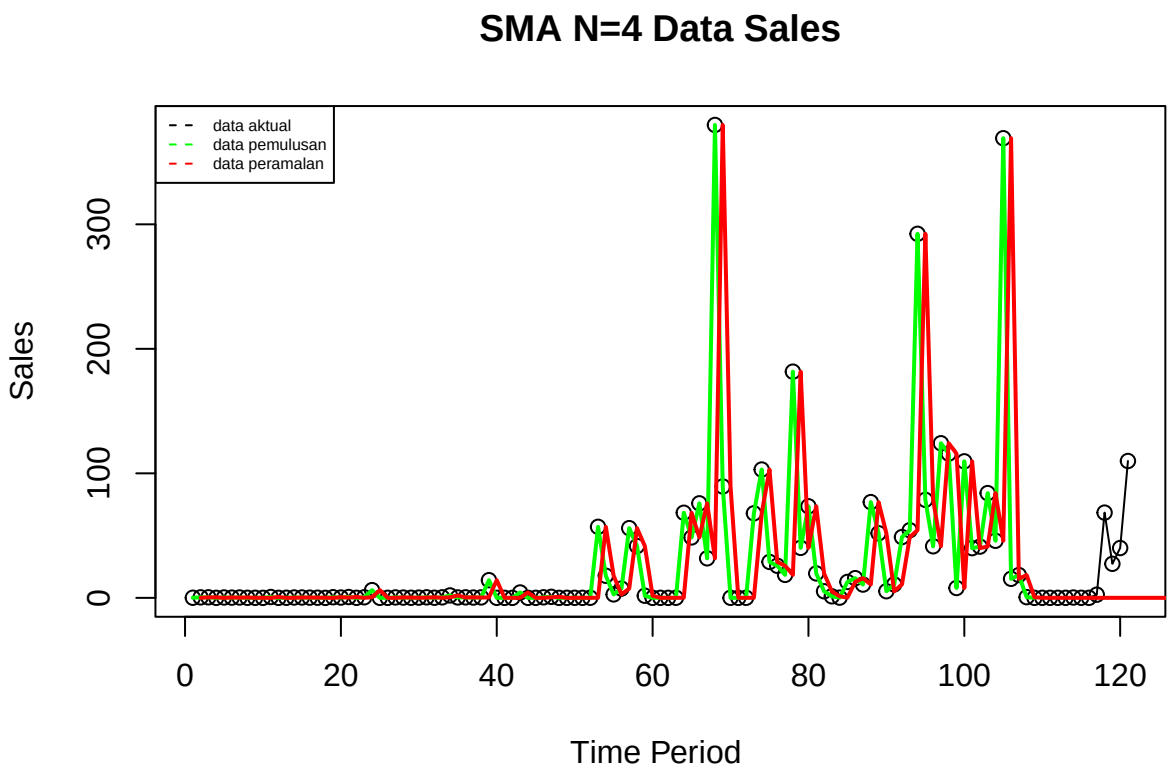
```
##  [15,]   0.3000    0.3000   0.3000
##  [16,]   0.1642    0.1642   0.3000
##  [17,]   0.1190    0.1190   0.1642
##  [18,]   0.0018    0.0018   0.1190
##  [19,]   0.5029    0.5029   0.0018
##  [20,]   0.2604    0.2604   0.5029
##  [21,]   0.6587    0.6587   0.2604
##  [22,]   0.0430    0.0430   0.6587
##  [23,]   0.4627    0.4627   0.0430
##  [24,]   6.2075    6.2075   0.4627
##  [25,]   0.0210    0.0210   6.2075
##  [26,]   0.0430    0.0430   0.0210
##  [27,]   0.3448    0.3448   0.0430
##  [28,]   0.2300    0.2300   0.3448
##  [29,]   0.0786    0.0786   0.2300
##  [30,]   0.2120    0.2120   0.0786
##  [31,]   0.4300    0.4300   0.2120
##  [32,]   0.1100    0.1100   0.4300
##  [33,]   0.4265    0.4265   0.1100
##  [34,]   1.9465    1.9465   0.4265
##  [35,]   0.3400    0.3400   1.9465
##  [36,]   0.4120    0.4120   0.3400
##  [37,]   0.2410    0.2410   0.4120
##  [38,]   0.3677    0.3677   0.2410
##  [39,]  14.2121   14.2121   0.3677
##  [40,]   0.0310    0.0310  14.2121
##  [41,]   0.0122    0.0122   0.0310
##  [42,]   0.0021    0.0021   0.0122
##  [43,]   4.3156    4.3156   0.0021
##  [44,]   0.0065    0.0065   4.3156
##  [45,]   0.0130    0.0130   0.0065
##  [46,]   0.4534    0.4534   0.0130
##  [47,]   0.8348    0.8348   0.4534
##  [48,]   0.0021    0.0021   0.8348
##  [49,]   0.0302    0.0302   0.0021
##  [50,]   0.0620    0.0620   0.0302
##  [51,]   0.0180    0.0180   0.0620
##  [52,]   0.0785    0.0785   0.0180
##  [53,]  57.0441   57.0441   0.0785
##  [54,]  17.7483   17.7483  57.0441
##  [55,]   2.7758    2.7758  17.7483
##  [56,]   7.4453    7.4453   2.7758
##  [57,]  55.9917   55.9917   7.4453
##  [58,]  41.5794   41.5794  55.9917
##  [59,]   1.7026    1.7026  41.5794
##  [60,]   0.0060    0.0060   1.7026
##  [61,]   0.0330    0.0330   0.0060
##  [62,]   0.0260    0.0260   0.0330
##  [63,]   0.0730    0.0730   0.0260
##  [64,]  68.3487   68.3487   0.0730
##  [65,]  48.5761   48.5761  68.3487
##  [66,]  75.9108   75.9108  48.5761
##  [67,]  31.7568   31.7568  75.9108
##  [68,] 379.7904  379.7904  31.7568
```

```
##  [69,]  89.4546   89.4546 379.7904
##  [70,]   0.0510    0.0510  89.4546
##  [71,]   0.0210    0.0210   0.0510
##  [72,]   0.0555    0.0555   0.0210
##  [73,]  67.9166   67.9166   0.0555
##  [74,] 103.0704  103.0704  67.9166
##  [75,]  28.8643   28.8643 103.0704
##  [76,]  25.7484   25.7484  28.8643
##  [77,]  18.4597   18.4597  25.7484
##  [78,] 181.6665  181.6665  18.4597
##  [79,]  40.1587   40.1587 181.6665
##  [80,]  73.5338   73.5338  40.1587
##  [81,]  19.6265   19.6265  73.5338
##  [82,]   5.2749    5.2749  19.6265
##  [83,]   1.2463    1.2463   5.2749
##  [84,]   0.1907    0.1907   1.2463
##  [85,]  12.7869   12.7869   0.1907
##  [86,]  15.8216   15.8216  12.7869
##  [87,]  10.6741   10.6741  15.8216
##  [88,]  76.8945   76.8945  10.6741
##  [89,]  51.8810   51.8810  76.8945
##  [90,]   5.3328    5.3328  51.8810
##  [91,]  10.8060   10.8060   5.3328
##  [92,]  48.8230   48.8230  10.8060
##  [93,]  54.3312   54.3312  48.8230
##  [94,] 292.3764  292.3764  54.3312
##  [95,]  78.7232   78.7232 292.3764
##  [96,]  41.4251   41.4251  78.7232
##  [97,] 124.1762  124.1762  41.4251
##  [98,] 115.8449  115.8449 124.1762
##  [99,]   7.9111    7.9111 115.8449
## [100,] 109.6561  109.6561   7.9111
## [101,]  39.9343   39.9343 109.6561
## [102,]  41.1686   41.1686  39.9343
## [103,]  84.1471   84.1471  41.1686
## [104,]  45.8479   45.8479  84.1471
## [105,] 369.1453  369.1453  45.8479
## [106,]  15.2997   15.2997 369.1453
## [107,]  18.3233   18.3233  15.2997
## [108,]   0.5688    0.5688  18.3233
## [109,]   0.0220    0.0220   0.5688
## [110,]       NA        NA   0.0220
## [111,]       NA        NA   0.0220
## [112,]       NA        NA   0.0220
## [113,]       NA        NA   0.0220
## [114,]       NA        NA   0.0220
## [115,]       NA        NA   0.0220
## [116,]       NA        NA   0.0220
## [117,]       NA        NA   0.0220
## [118,]       NA        NA   0.0220
## [119,]       NA        NA   0.0220
## [120,]       NA        NA   0.0220
## [121,]       NA        NA   0.0220
## [122,]       NA        NA   0.0220
```

```
## [123,]       NA        NA   0.0220
## [124,]       NA        NA   0.0220
## [125,]       NA        NA   0.0220
## [126,]       NA        NA   0.0220
## [127,]       NA        NA   0.0220
## [128,]       NA        NA   0.0220
## [129,]       NA        NA   0.0220
## [130,]       NA        NA   0.0220
## [131,]       NA        NA   0.0220
## [132,]       NA        NA   0.0220
## [133,]       NA        NA   0.0220
```

Adapun plot data deret waktu dari hasil peramalan yang dilakukan adalah sebagai berikut.

```
ts.plot(data1.ts, xlab="Time Period ", ylab="Sales", main= "SMA N=4 Data Sales")
points(data1.ts)
lines(data.gab[,2],col="green",lwd=2)
lines(data.gab[,3],col="red",lwd=2)
legend("topleft",c("data aktual","data pemulusan","data peramalan"), lty=8, col=c("black","green","red")
```



Selanjutnya perhitungan akurasi dilakukan dengan ukuran akurasi *Sum Squares Error* (SSE), *Mean Square Error* (MSE) dan *Mean Absolute Percentage Error* (MAPE). Perhitungan akurasi dilakukan baik pada data latih maupun pada data uji.

```r
#Menghitung nilai keakuratan data latih
error_train.sma = train_ma.ts-data.ramal[1:length(train_ma.ts)]
SSE_train.sma = sum(error_train.sma[5:length(train_ma.ts)]^2)
MSE_train.sma = mean(error_train.sma[5:length(train_ma.ts)]^2)
MAPE_train.sma = mean(abs((error_train.sma[5:length(train_ma.ts)]/train_ma.ts[5:length(train_ma.ts)])*1

akurasi_train.sma <- matrix(c(SSE_train.sma, MSE_train.sma, MAPE_train.sma))
row.names(akurasi_train.sma)<- c("SSE", "MSE", "MAPE")
colnames(akurasi_train.sma) <- c("Akurasi m = 1")
akurasi_train.sma
```

```
##      Akurasi m = 1
## SSE     672911.461
## MSE       6408.681
## MAPE      3990.899
```

Dalam hal ini nilai MAPE data latih pada metode pemulusan SMA kurang dari 2%, nilai ini dapat dikategorikan sebagai nilai akurasi yang sangat baik. Selanjutnya dilakukan perhitungan nilai MAPE data uji pada metde pemulusan SMA.

```r
#Menghitung nilai keakuratan data uji
error_test.sma = test_ma.ts-data.gab[110:121,3]
SSE_test.sma = sum(error_test.sma^2)
MSE_test.sma = mean(error_test.sma^2)
MAPE_test.sma = mean(abs((error_test.sma/test_ma.ts*100)))

akurasi_test.sma <- matrix(c(SSE_test.sma, MSE_test.sma, MAPE_test.sma))
row.names(akurasi_test.sma)<- c("SSE", "MSE", "MAPE")
colnames(akurasi_test.sma) <- c("Akurasi m = 1")
akurasi_test.sma
```

```
##      Akurasi m = 1
## SSE    19116.21341
## MSE     1593.01778
## MAPE      69.07574
```

Perhitungan akurasi menggunakan data latih menghasilkan nilai MAPE yang kurang dari 10% sehingga nilai akurasi ini dapat dikategorikan sebagai sangat baik.

Double Moving Average (DMA)

Metode pemulusan Double Moving Average (DMA) pada dasarnya mirip dengan SMA. Namun demikian, metode ini lebih cocok digunakan untuk pola data trend. Proses pemulusan dengan rata rata dalam metode ini dilakukan sebanyak 2 kali.

```r
dma <- SMA(data.sma, n = 1)
At <- 2*data.sma - dma
Bt <- 2/(4-1)*(data.sma - dma)
data.dma<- At+Bt
data.ramal2<- c(NA, data.dma)
```

```r
t = 1:24
f = c()

for (i in t) {
  f[i] = At[length(At)] + Bt[length(Bt)]*(i)
}

data.gab2 <- cbind(aktual = c(train_ma.ts,rep(NA,24)), pemulusan1 = c(data.sma,rep(NA,24)),pemulusan2 =
data.gab2
```

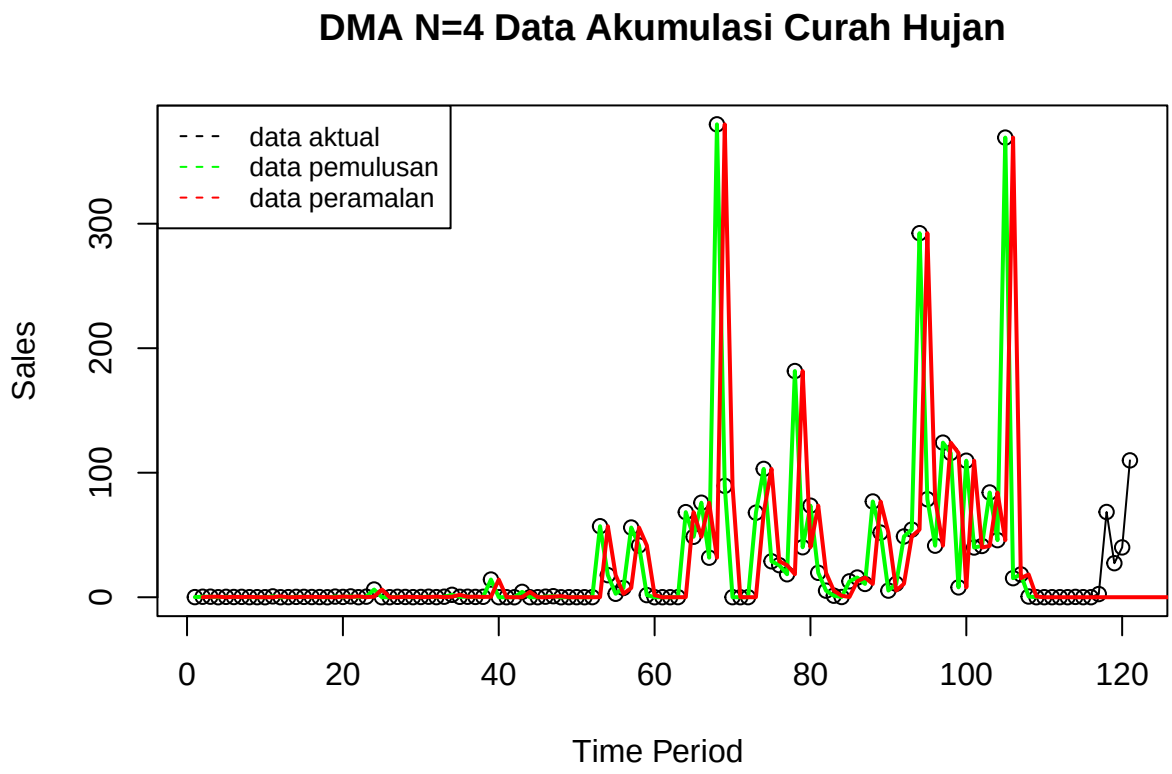
```
##          aktual pemulusan1 pemulusan2       At            Bt  ramalan
##   [1,]   0.1019     0.1019     0.1019   0.1019  0.000000e+00       NA
##   [2,]   0.4100     0.4100     0.4100   0.4100  0.000000e+00   0.1019
##   [3,]   0.4500     0.4500     0.4500   0.4500  3.700743e-17   0.4100
##   [4,]   0.0200     0.0200     0.0200   0.0200  3.700743e-17   0.4500
##   [5,]   0.3100     0.3100     0.3100   0.3100  3.700743e-17   0.0200
##   [6,]   0.1790     0.1790     0.1790   0.1790  3.700743e-17   0.3100
##   [7,]   0.2901     0.2901     0.2901   0.2901  3.700743e-17   0.1790
##   [8,]   0.0544     0.0544     0.0544   0.0544  3.700743e-17   0.2901
##   [9,]   0.0610     0.0610     0.0610   0.0610  4.163336e-17   0.0544
##  [10,]   0.0171     0.0171     0.0171   0.0171  4.163336e-17   0.0610
##  [11,]   0.6863     0.6863     0.6863   0.6863  7.401487e-17   0.0171
##  [12,]   0.0210     0.0210     0.0210   0.0210  7.401487e-17   0.6863
##  [13,]   0.0831     0.0831     0.0831   0.0831  7.401487e-17   0.0210
##  [14,]   0.3000     0.3000     0.3000   0.3000  7.401487e-17   0.0831
##  [15,]   0.3000     0.3000     0.3000   0.3000  7.401487e-17   0.3000
##  [16,]   0.1642     0.1642     0.1642   0.1642  7.401487e-17   0.3000
##  [17,]   0.1190     0.1190     0.1190   0.1190  7.401487e-17   0.1642
##  [18,]   0.0018     0.0018     0.0018   0.0018  7.401487e-17   0.1190
##  [19,]   0.5029     0.5029     0.5029   0.5029  7.401487e-17   0.0018
##  [20,]   0.2604     0.2604     0.2604   0.2604  7.401487e-17   0.5029
##  [21,]   0.6587     0.6587     0.6587   0.6587  7.401487e-17   0.2604
##  [22,]   0.0430     0.0430     0.0430   0.0430  7.401487e-17   0.6587
##  [23,]   0.4627     0.4627     0.4627   0.4627  3.700743e-17   0.0430
##  [24,]   6.2075     6.2075     6.2075   6.2075  0.000000e+00   0.4627
##  [25,]   0.0210     0.0210     0.0210   0.0210  0.000000e+00   6.2075
##  [26,]   0.0430     0.0430     0.0430   0.0430  0.000000e+00   0.0210
##  [27,]   0.3448     0.3448     0.3448   0.3448  3.700743e-17   0.0430
##  [28,]   0.2300     0.2300     0.2300   0.2300  3.700743e-17   0.3448
##  [29,]   0.0786     0.0786     0.0786   0.0786  5.551115e-17   0.2300
##  [30,]   0.2120     0.2120     0.2120   0.2120  5.551115e-17   0.0786
##  [31,]   0.4300     0.4300     0.4300   0.4300  7.401487e-17   0.2120
##  [32,]   0.1100     0.1100     0.1100   0.1100  3.700743e-17   0.4300
##  [33,]   0.4265     0.4265     0.4265   0.4265  3.700743e-17   0.1100
##  [34,]   1.9465     1.9465     1.9465   1.9465  0.000000e+00   0.4265
##  [35,]   0.3400     0.3400     0.3400   0.3400  0.000000e+00   1.9465
##  [36,]   0.4120     0.4120     0.4120   0.4120  3.700743e-17   0.3400
##  [37,]   0.2410     0.2410     0.2410   0.2410  0.000000e+00   0.4120
##  [38,]   0.3677     0.3677     0.3677   0.3677  3.700743e-17   0.2410
##  [39,]  14.2121    14.2121    14.2121  14.2121  0.000000e+00   0.3677
##  [40,]   0.0310     0.0310     0.0310   0.0310  0.000000e+00  14.2121
##  [41,]   0.0122     0.0122     0.0122   0.0122  0.000000e+00   0.0310
##  [42,]   0.0021     0.0021     0.0021   0.0021  0.000000e+00   0.0122
```

```
##  [43,]   4.3156     4.3156     4.3156   4.3156  0.000000e+00   0.0021
##  [44,]   0.0065     0.0065     0.0065   0.0065  0.000000e+00   4.3156
##  [45,]   0.0130     0.0130     0.0130   0.0130  0.000000e+00   0.0065
##  [46,]   0.4534     0.4534     0.4534   0.4534  0.000000e+00   0.0130
##  [47,]   0.8348     0.8348     0.8348   0.8348 -7.401487e-17   0.4534
##  [48,]   0.0021     0.0021     0.0021   0.0021 -7.401487e-17   0.8348
##  [49,]   0.0302     0.0302     0.0302   0.0302 -7.170190e-17   0.0021
##  [50,]   0.0620     0.0620     0.0620   0.0620 -7.401487e-17   0.0302
##  [51,]   0.0180     0.0180     0.0180   0.0180 -7.401487e-17   0.0620
##  [52,]   0.0785     0.0785     0.0785   0.0785 -7.401487e-17   0.0180
##  [53,]  57.0441    57.0441    57.0441  57.0441  0.000000e+00   0.0785
##  [54,]  17.7483    17.7483    17.7483  17.7483  0.000000e+00  57.0441
##  [55,]   2.7758     2.7758     2.7758   2.7758  0.000000e+00  17.7483
##  [56,]   7.4453     7.4453     7.4453   7.4453  0.000000e+00   2.7758
##  [57,]  55.9917    55.9917    55.9917  55.9917  0.000000e+00   7.4453
##  [58,]  41.5794    41.5794    41.5794  41.5794  0.000000e+00  55.9917
##  [59,]   1.7026     1.7026     1.7026   1.7026  0.000000e+00  41.5794
##  [60,]   0.0060     0.0060     0.0060   0.0060  0.000000e+00   1.7026
##  [61,]   0.0330     0.0330     0.0330   0.0330  0.000000e+00   0.0060
##  [62,]   0.0260     0.0260     0.0260   0.0260  0.000000e+00   0.0330
##  [63,]   0.0730     0.0730     0.0730   0.0730  0.000000e+00   0.0260
##  [64,]  68.3487    68.3487    68.3487  68.3487  0.000000e+00   0.0730
##  [65,]  48.5761    48.5761    48.5761  48.5761  0.000000e+00  68.3487
##  [66,]  75.9108    75.9108    75.9108  75.9108  0.000000e+00  48.5761
##  [67,]  31.7568    31.7568    31.7568  31.7568  0.000000e+00  75.9108
##  [68,] 379.7904   379.7904   379.7904 379.7904  0.000000e+00  31.7568
##  [69,]  89.4546    89.4546    89.4546  89.4546  0.000000e+00 379.7904
##  [70,]   0.0510     0.0510     0.0510   0.0510  0.000000e+00  89.4546
##  [71,]   0.0210     0.0210     0.0210   0.0210  4.625929e-18   0.0510
##  [72,]   0.0555     0.0555     0.0555   0.0555  4.625929e-18   0.0210
##  [73,]  67.9166    67.9166    67.9166  67.9166  0.000000e+00   0.0555
##  [74,] 103.0704   103.0704   103.0704 103.0704 -9.473903e-15  67.9166
##  [75,]  28.8643    28.8643    28.8643  28.8643 -1.894781e-14 103.0704
##  [76,]  25.7484    25.7484    25.7484  25.7484 -1.657933e-14  28.8643
##  [77,]  18.4597    18.4597    18.4597  18.4597 -1.421085e-14  25.7484
##  [78,] 181.6665   181.6665   181.6665 181.6665  0.000000e+00  18.4597
##  [79,]  40.1587    40.1587    40.1587  40.1587  0.000000e+00 181.6665
##  [80,]  73.5338    73.5338    73.5338  73.5338  0.000000e+00  40.1587
##  [81,]  19.6265    19.6265    19.6265  19.6265  0.000000e+00  73.5338
##  [82,]   5.2749     5.2749     5.2749   5.2749  0.000000e+00  19.6265
##  [83,]   1.2463     1.2463     1.2463   1.2463  0.000000e+00   5.2749
##  [84,]   0.1907     0.1907     0.1907   0.1907  0.000000e+00   1.2463
##  [85,]  12.7869    12.7869    12.7869  12.7869  0.000000e+00   0.1907
##  [86,]  15.8216    15.8216    15.8216  15.8216 -1.184238e-15  12.7869
##  [87,]  10.6741    10.6741    10.6741  10.6741 -1.184238e-15  15.8216
##  [88,]  76.8945    76.8945    76.8945  76.8945  0.000000e+00  10.6741
##  [89,]  51.8810    51.8810    51.8810  51.8810  0.000000e+00  76.8945
##  [90,]   5.3328     5.3328     5.3328   5.3328  0.000000e+00  51.8810
##  [91,]  10.8060    10.8060    10.8060  10.8060  0.000000e+00   5.3328
##  [92,]  48.8230    48.8230    48.8230  48.8230  0.000000e+00  10.8060
##  [93,]  54.3312    54.3312    54.3312  54.3312 -4.736952e-15  48.8230
##  [94,] 292.3764   292.3764   292.3764 292.3764  0.000000e+00  54.3312
##  [95,]  78.7232    78.7232    78.7232  78.7232  0.000000e+00 292.3764
##  [96,]  41.4251    41.4251    41.4251  41.4251  0.000000e+00  78.7232
```

```
##  [97,] 124.1762   124.1762   124.1762 124.1762 -9.473903e-15  41.4251
##  [98,] 115.8449   115.8449   115.8449 115.8449  0.000000e+00 124.1762
##  [99,]   7.9111     7.9111     7.9111   7.9111  0.000000e+00 115.8449
## [100,] 109.6561   109.6561   109.6561 109.6561  0.000000e+00   7.9111
## [101,]  39.9343    39.9343    39.9343  39.9343  0.000000e+00 109.6561
## [102,]  41.1686    41.1686    41.1686  41.1686  4.736952e-15  39.9343
## [103,]  84.1471    84.1471    84.1471  84.1471  0.000000e+00  41.1686
## [104,]  45.8479    45.8479    45.8479  45.8479  9.473903e-15  84.1471
## [105,] 369.1453   369.1453   369.1453 369.1453  0.000000e+00  45.8479
## [106,]  15.2997    15.2997    15.2997  15.2997  0.000000e+00 369.1453
## [107,]  18.3233    18.3233    18.3233  18.3233  0.000000e+00  15.2997
## [108,]   0.5688     0.5688     0.5688   0.5688  0.000000e+00  18.3233
## [109,]   0.0220     0.0220     0.0220   0.0220  0.000000e+00   0.5688
## [110,]       NA         NA         NA       NA            NA   0.0220
## [111,]       NA         NA         NA       NA            NA   0.0220
## [112,]       NA         NA         NA       NA            NA   0.0220
## [113,]       NA         NA         NA       NA            NA   0.0220
## [114,]       NA         NA         NA       NA            NA   0.0220
## [115,]       NA         NA         NA       NA            NA   0.0220
## [116,]       NA         NA         NA       NA            NA   0.0220
## [117,]       NA         NA         NA       NA            NA   0.0220
## [118,]       NA         NA         NA       NA            NA   0.0220
## [119,]       NA         NA         NA       NA            NA   0.0220
## [120,]       NA         NA         NA       NA            NA   0.0220
## [121,]       NA         NA         NA       NA            NA   0.0220
## [122,]       NA         NA         NA       NA            NA   0.0220
## [123,]       NA         NA         NA       NA            NA   0.0220
## [124,]       NA         NA         NA       NA            NA   0.0220
## [125,]       NA         NA         NA       NA            NA   0.0220
## [126,]       NA         NA         NA       NA            NA   0.0220
## [127,]       NA         NA         NA       NA            NA   0.0220
## [128,]       NA         NA         NA       NA            NA   0.0220
## [129,]       NA         NA         NA       NA            NA   0.0220
## [130,]       NA         NA         NA       NA            NA   0.0220
## [131,]       NA         NA         NA       NA            NA   0.0220
## [132,]       NA         NA         NA       NA            NA   0.0220
## [133,]       NA         NA         NA       NA            NA   0.0220
```

Hasil pemulusan menggunakan metode DMA divisualisasikan sebagai berikut

```
ts.plot(data1.ts, xlab="Time Period ", ylab="Sales", main= "DMA N=4 Data Akumulasi Curah Hujan")
points(data1.ts)
lines(data.gab2[,3],col="green",lwd=2)
lines(data.gab2[,6],col="red",lwd=2)
legend("topleft",c("data aktual","data pemulusan","data peramalan"), lty=8, col=c("black","green","red"
```



Selanjutnya perhitungan akurasi dilakukan baik pada data latih maupun data uji. Perhitungan akurasi dilakukan dengan ukuran akurasi SSE, MSE dan MAPE.

```
#Menghitung nilai keakuratan data latih
error_train.dma = train_ma.ts-data.ramal2[1:length(train_ma.ts)]
SSE_train.dma = sum(error_train.dma[8:length(train_ma.ts)]^2)
MSE_train.dma = mean(error_train.dma[8:length(train_ma.ts)]^2)
MAPE_train.dma = mean(abs((error_train.dma[8:length(train_ma.ts)]/train_ma.ts[8:length(train_ma.ts)])*1

akurasi_train.dma <- matrix(c(SSE_train.dma, MSE_train.dma, MAPE_train.dma))
row.names(akurasi_train.dma)<- c("SSE", "MSE", "MAPE")
colnames(akurasi_train.dma) <- c("Akurasi m = 1")
akurasi_train.dma
```

```
##      Akurasi m = 1
## SSE     672911.348
## MSE       6597.170
## MAPE      4106.268
```

Perhitungan akurasi pada data latih menggunakan nilai MAPE menghasilkan nilai MAPE yang lebih dari 10% sehingga dikategorikan diragukan. Selanjutnya, perhitungan nilai akurasi dilakukan pada data uji.

```
#Menghitung nilai keakuratan data uji
error_test.dma = test_ma.ts-data.gab2[110:120,6]
```

```
## Warning in `-.default`(test_ma.ts, data.gab2[110:120, 6]): longer object length
```

```
## is not a multiple of shorter object length
```

```
SSE_test.dma = sum(error_test.dma^2)
MSE_test.dma = mean(error_test.dma^2)
MAPE_test.dma = mean(abs((error_test.dma/test_ma.ts*100)))

akurasi_test.dma <- matrix(c(SSE_test.dma, MSE_test.dma, MAPE_test.dma))
row.names(akurasi_test.dma)<- c("SSE", "MSE", "MAPE")
colnames(akurasi_test.dma) <- c("Akurasi m = 1")
akurasi_test.dma
```

```
##      Akurasi m = 1
## SSE    19116.21341
## MSE     1593.01778
## MAPE      69.07574
```

Perhitungan akurasi menggunakan data latih menghasilkan nilai MAPE yang lebih dari 10% sehingga nilai akurasi ini dapat dikategorikan sebagai diragukan.

Pada data latih, metode SMA lebih baik dibandingkan dengan metode DMA, sedangkan pada data uji, metode DMA lebih baik dibandingkan SMA

Single Exponential Smoothing & Double Exponential Smoothing

Metode *Exponential Smoothing* adalah metode pemulusan dengan melakukan pembobotan menurun secara eksponensial. Nilai yang lebih baru diberi bobot yang lebih besar dari nilai terdahulu. Terdapat satu atau lebih parameter pemulusan yang ditentukan secara eksplisit, dan hasil pemilihan parameter tersebut akan menentukan bobot yang akan diberikan pada nilai pengamatan. Ada dua macam model, yaitu model tunggal dan ganda.

Pembagian Data

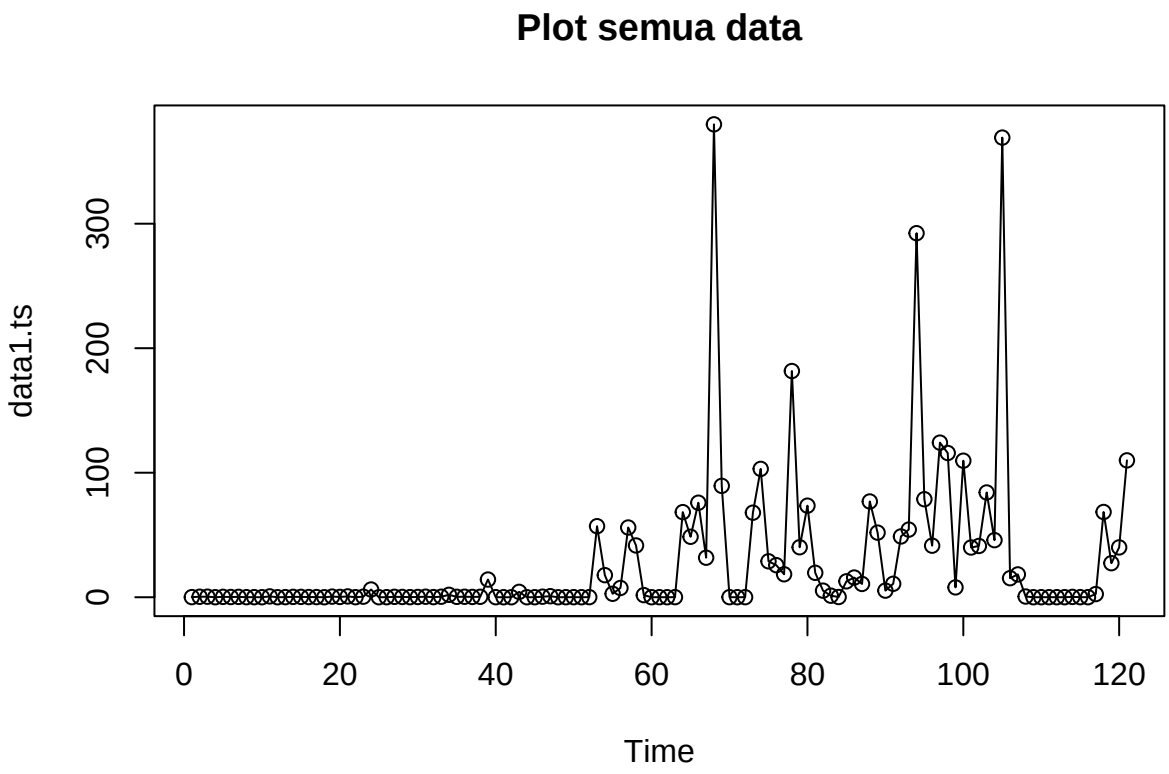
Pembagian data latih dan data uji dilakukan dengan perbandingan 80% data latih dan 20% data uji.

```
#membagi training dan testing
training<-data1[1:109,]
testing<-data1[110:121,]
train.ts <- ts(training$`Akumulasi Hujan`)
test.ts <- ts(testing$`Akumulasi Hujan`)
```

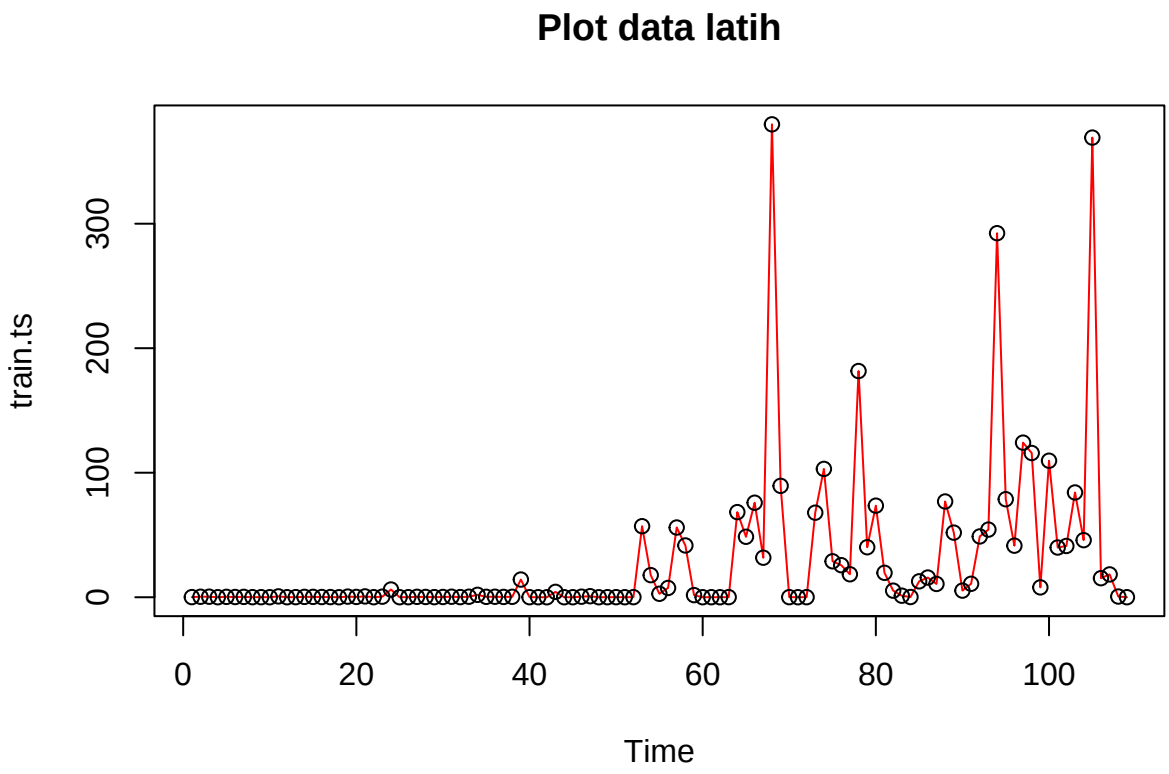
Eksplorasi

Eksplorasi dilakukan dengan membuat plot data deret waktu untuk keseluruhan data, data latih, dan data uji.

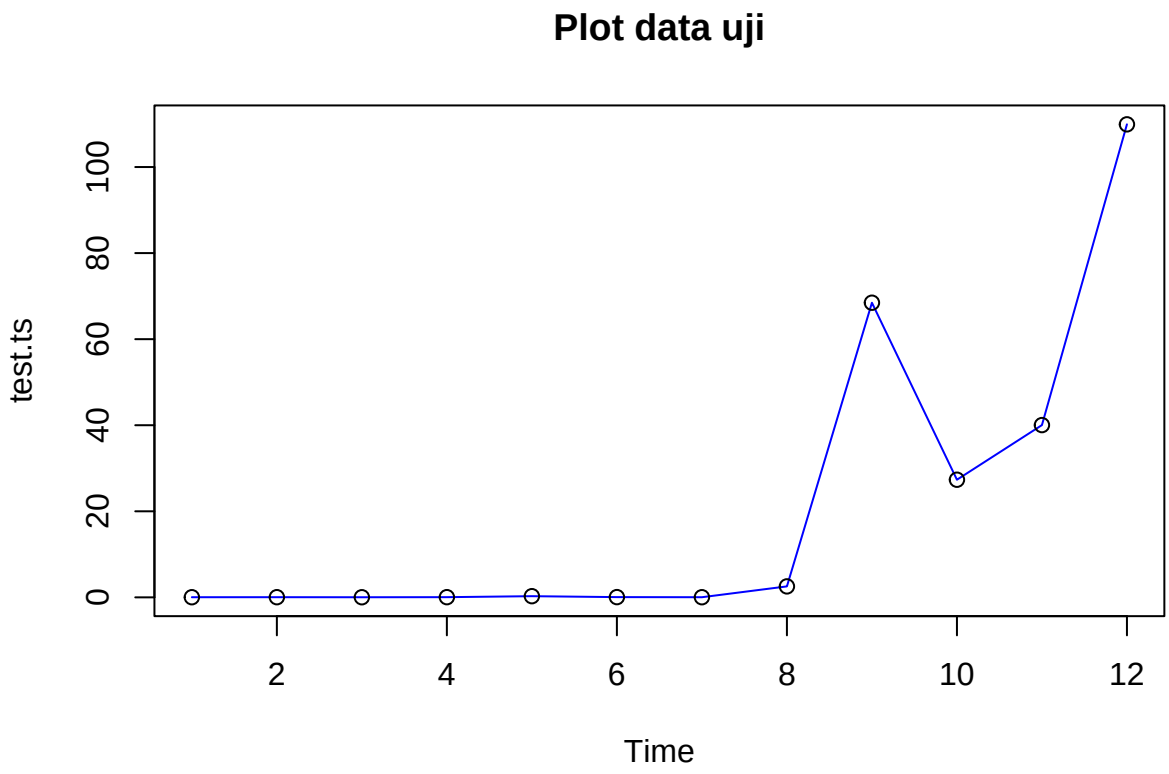
```
#eksplorasi data
plot(data1.ts, col="black",main="Plot semua data")
points(data1.ts)
```




```
plot(train.ts, col="red",main="Plot data latih")
points(train.ts)
```

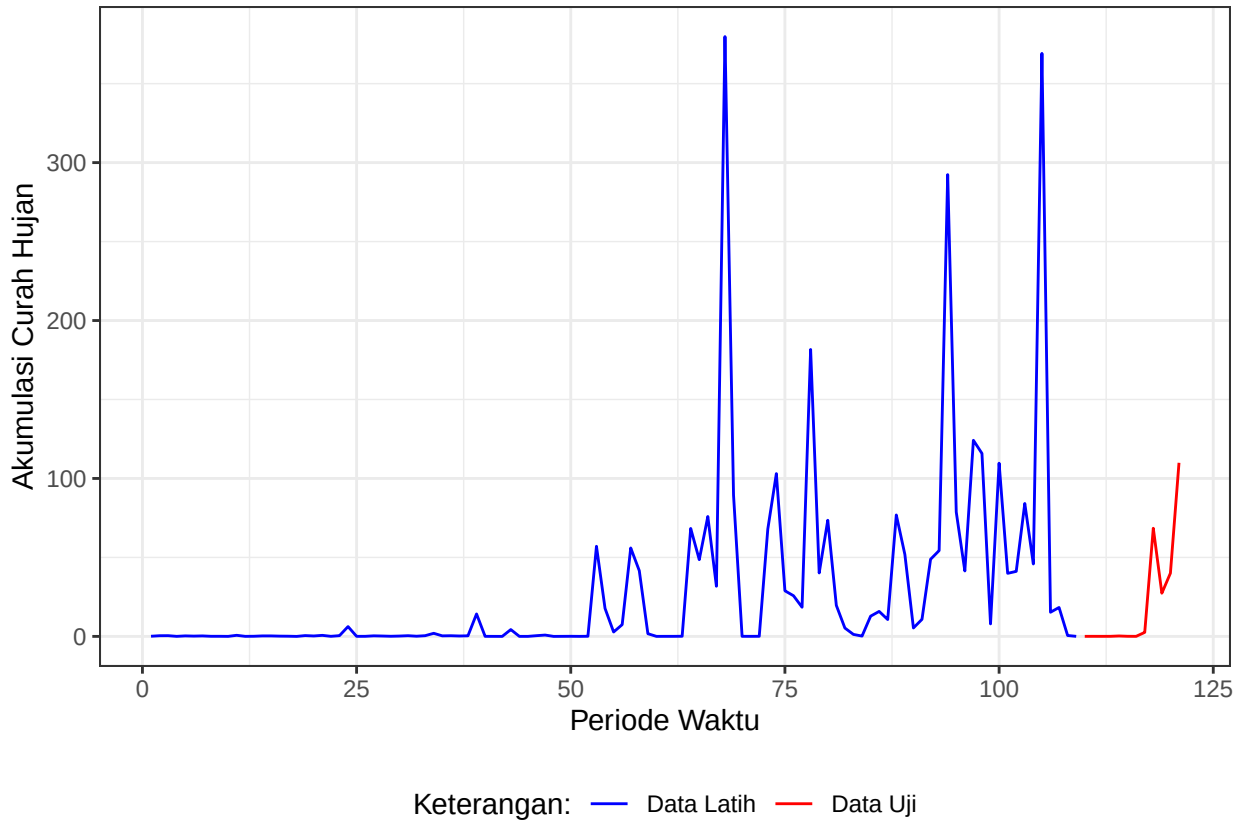



```
plot(test.ts, col="blue",main="Plot data uji")
points(test.ts)
```



Eksplorasi data juga dapat dilakukan menggunakan package `ggplot2` .

```
#Eksplorasi dengan GGPLOT
library(ggplot2)
ggplot() +
  geom_line(data = training_ma, aes(x = Tanggal, y = `Akumulasi Hujan`, col = "Data Latih")) +
  geom_line(data = testing_ma, aes(x = Tanggal, y = `Akumulasi Hujan`, col = "Data Uji")) +
  labs(x = "Periode Waktu", y = "Akumulasi Curah Hujan", color = "Legend") +
  scale_colour_manual(name="Keterangan:", breaks = c("Data Latih", "Data Uji"),
                      values = c("blue", "red")) +
  theme_bw() + theme(legend.position = "bottom",
                     plot.caption = element_text(hjust=0.5, size=12))
```



SES

Single Exponential Smoothing merupakan metode pemulusan yang tepat digunakan untuk data dengan pola stasioner atau konstan.

Nilai pemulusan pada periode ke-t didapat dari persamaan:

$$\tilde{y}_T = \lambda y_t + (1 - \lambda)\tilde{y}_{T-1}$$

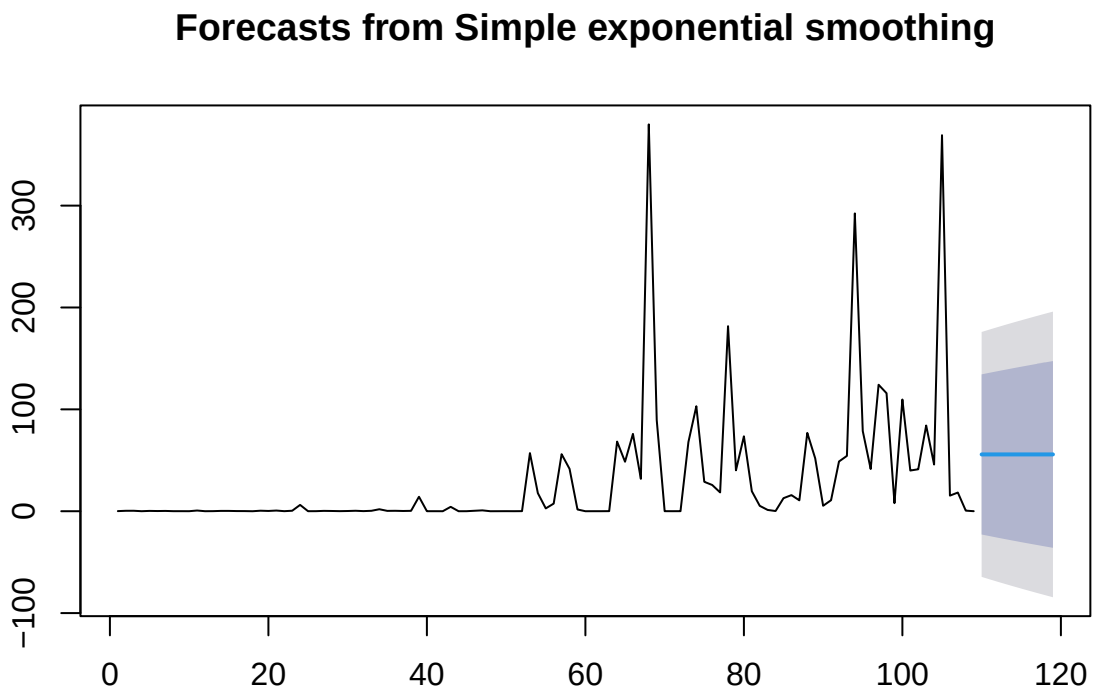
Nilai parameter λ adalah nilai antara 0 dan 1.

Nilai pemulusan periode ke-t bertindak sebagai nilai ramalan pada periode ke-$(T + \tau)$.

$$\tilde{y}_{T+\tau}(T) = \tilde{y}_T$$

Pemulusan dengan metode SES dapat dilakukan dengan dua fungsi dari *packages* berbeda, yaitu (1) fungsi `ses()` dari *packages* `forecast` dan (2) fungsi `HoltWinters` dari *packages* `stats` .

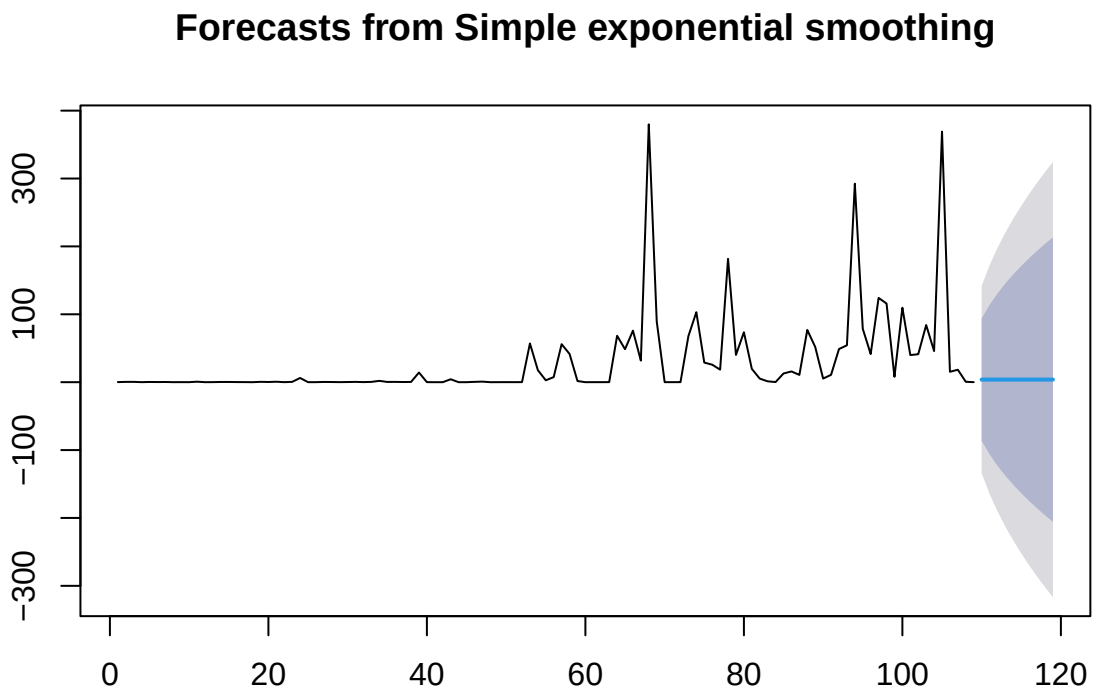
```
#Cara 1 (fungsi ses)
ses.1 <- ses(train.ts, h = 10, alpha = 0.2)
plot(ses.1)
```




```
ses.1
```

```
##     Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 110       55.74972 -22.85620 134.3556 -64.46766 175.9671
## 111       55.74972 -24.41290 135.9123 -66.84843 178.3479
## 112       55.74972 -25.93995 137.4394 -69.18384 180.6833
## 113       55.74972 -27.43896 138.9384 -71.47639 182.9758
## 114       55.74972 -28.91144 140.4109 -73.72836 185.2278
## 115       55.74972 -30.35875 141.8582 -75.94182 187.4413
## 116       55.74972 -31.78213 143.2816 -78.11869 189.6181
## 117       55.74972 -33.18272 144.6822 -80.26072 191.7602
## 118       55.74972 -34.56160 146.0610 -82.36953 193.8690
## 119       55.74972 -35.91975 147.4192 -84.44663 195.9461
```

```
ses.2<- ses(train.ts, h = 10, alpha = 0.7)
plot(ses.2)
```

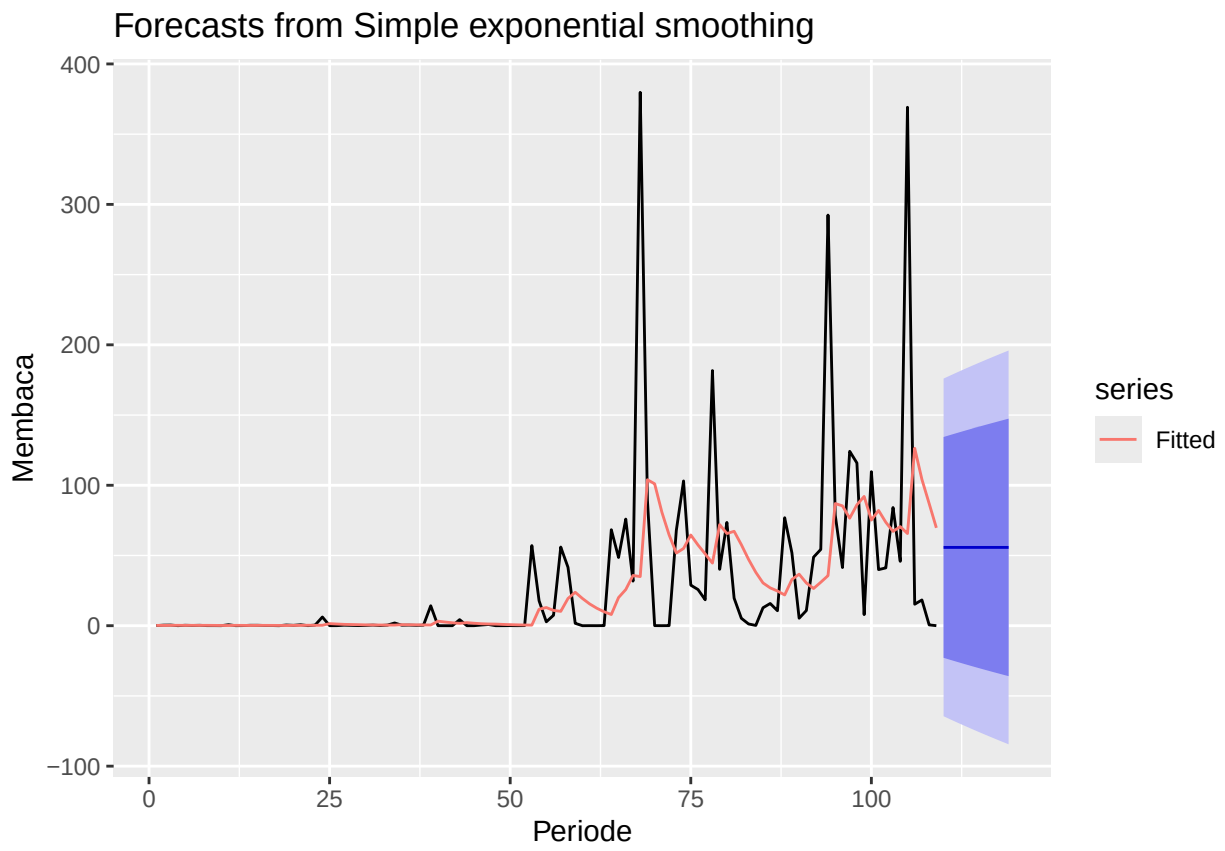



```
ses.2
```

```
##     Point Forecast      Lo 80     Hi 80     Lo 95    Hi 95
## 110       3.802242  -86.32273  93.92722 -134.0320 141.6365
## 111       3.802242 -106.20931 113.81379 -164.4459 172.0504
## 112       3.802242 -123.01484 130.61932 -190.1477 197.7522
## 113       3.802242 -137.84027 145.44476 -212.8213 220.4258
## 114       3.802242 -151.25463 158.85911 -233.3368 240.9413
## 115       3.802242 -163.59747 171.20195 -252.2135 259.8180
## 116       3.802242 -175.09073 182.69521 -269.7909 277.3954
## 117       3.802242 -185.88889 193.49337 -286.3053 293.9098
## 118       3.802242 -196.10462 203.70911 -301.9289 309.5334
## 119       3.802242 -205.82310 213.42759 -316.7921 324.3965
```

Untuk mendapatkan gambar hasil pemulusan pada data latih dengan fungsi `ses()` , perlu digunakan fungsi `autoplot()` dan `autolayer()` dari *library packages* `ggplot2` .

```
autoplot(ses.1) +
  autolayer(fitted(ses.1), series="Fitted") +
  ylab("Membaca") + xlab("Periode")
```

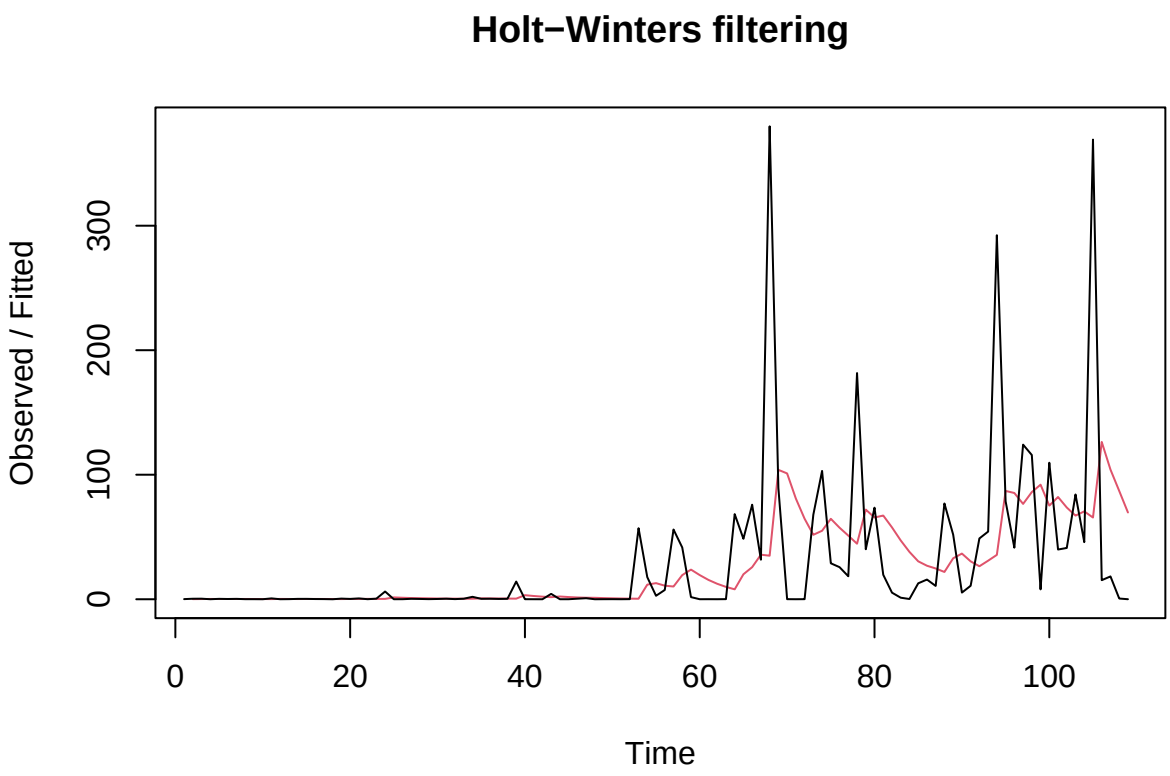


Pada fungsi `ses()` , terdapat beberapa argumen yang umum digunakan, yaitu nilia `y` , `gamma` , `beta` , `alpha` , dan `h` .

Nilai `y` adalah nilai data deret waktu, `gamma` adalah parameter pemulusan untuk komponen musiman, `beta` adalah parameter pemulusan untuk tren, dan `alpha` adalah parameter pemulusan untuk stasioner, serta `h` adalah banyaknya periode yang akan diramalkan.

Kasus di atas merupakan contoh inisialisasi nilai parameter λ dengan nilai `alpha` 0,2 dan 0,7 dan banyak periode data yang akan diramalkan adalah sebanyak 10 periode. Selanjutnya akan digunakan fungsi `HoltWinters()` dengan nilai inisialisasi parameter dan panjang periode peramalan yang sama dengan fungsi `ses()` .

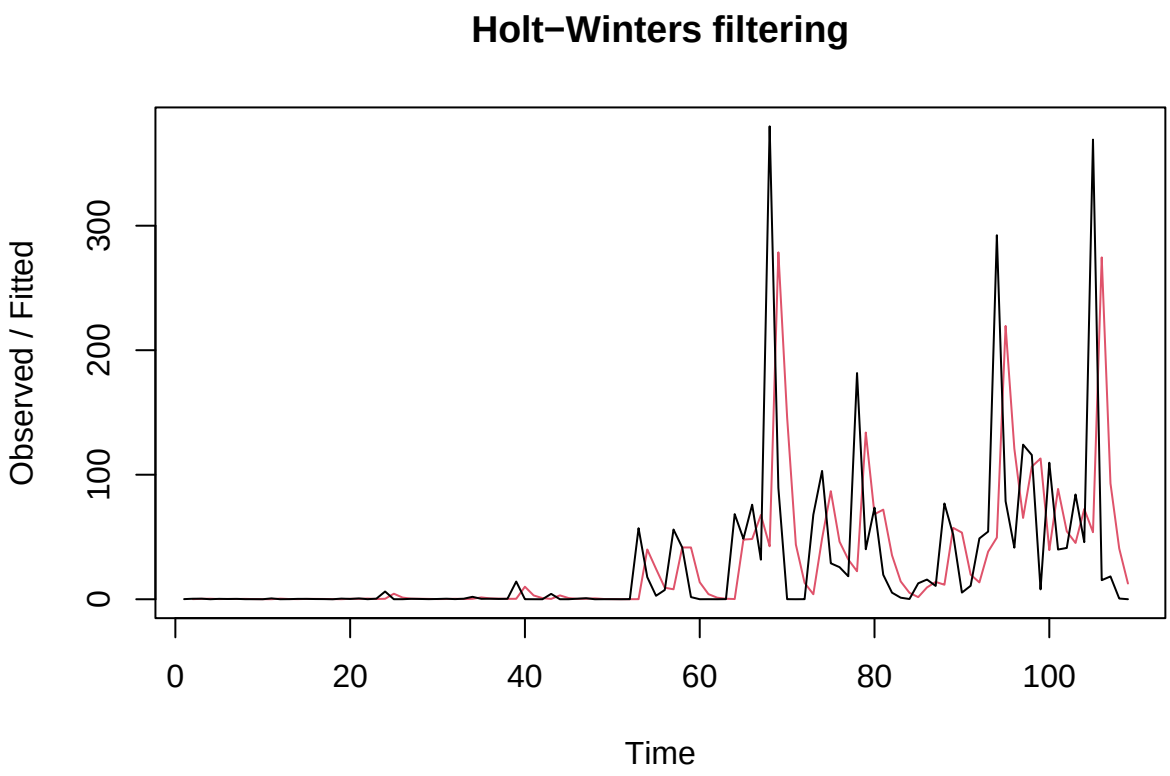
```
#Cara 2 (fungsi Holtwinter)
ses1<- HoltWinters(train.ts, gamma = FALSE, beta = FALSE, alpha = 0.2)
plot(ses1)
```




```
#ramalan
ramalan1<- forecast(ses1, h=10)
ramalan1
```

```
##     Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 110       55.74972 -22.78619 134.2856 -64.36058 175.8600
## 111       55.74972 -24.34150 135.8409 -66.73924 178.2387
## 112       55.74972 -25.86719 137.3666 -69.07257 180.5720
## 113       55.74972 -27.36487 138.8643 -71.36308 182.8625
## 114       55.74972 -28.83604 140.3355 -73.61303 185.1125
## 115       55.74972 -30.28205 141.7815 -75.82452 187.3240
## 116       55.74972 -31.70416 143.2036 -77.99945 189.4989
## 117       55.74972 -33.10351 144.6030 -80.13957 191.6390
## 118       55.74972 -34.48117 145.9806 -82.24651 193.7460
## 119       55.74972 -35.83810 147.3375 -84.32176 195.8212
```

```
ses2<- HoltWinters(train.ts, gamma = FALSE, beta = FALSE, alpha = 0.7)
plot(ses2)
```



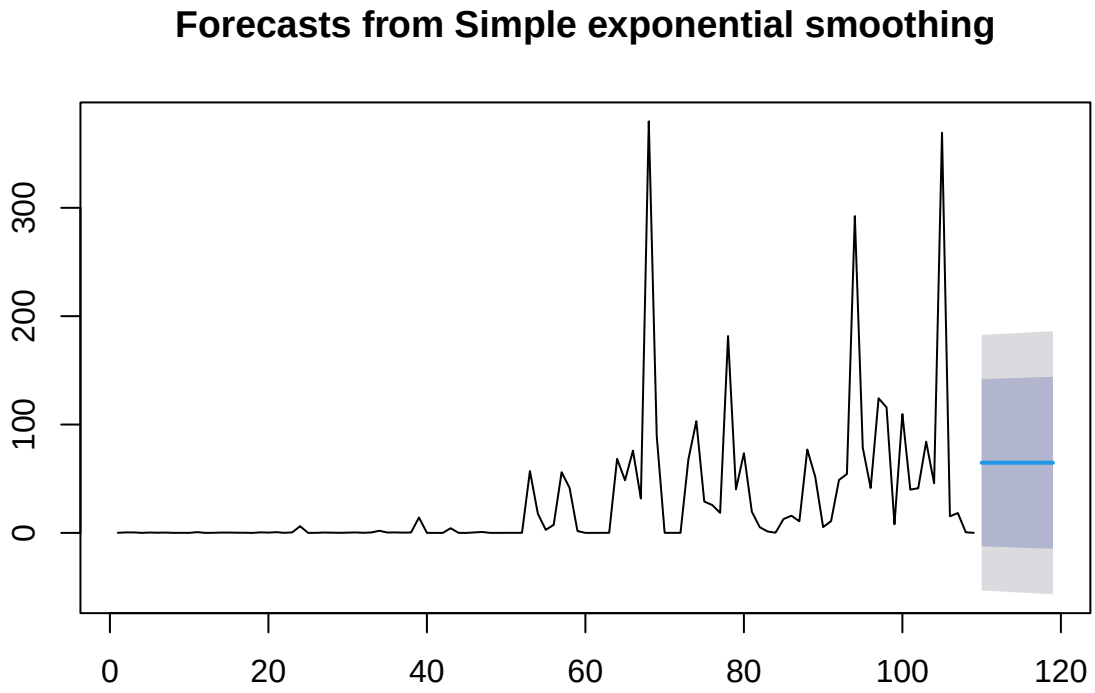

```
#ramalan
ramalan2<- forecast(ses2, h=10)
ramalan2
```

```
##     Point Forecast      Lo 80    Hi 80     Lo 95    Hi 95
## 110       3.802242  -86.32271  93.9272 -134.0320 141.6365
## 111       3.802242 -106.20928 113.8138 -164.4459 172.0504
## 112       3.802242 -123.01481 130.6193 -190.1477 197.7522
## 113       3.802242 -137.84024 145.4447 -212.8212 220.4257
## 114       3.802242 -151.25459 158.8591 -233.3367 240.9412
## 115       3.802242 -163.59743 171.2019 -252.2135 259.8179
## 116       3.802242 -175.09068 182.6952 -269.7909 277.3954
## 117       3.802242 -185.88884 193.4933 -286.3052 293.9097
## 118       3.802242 -196.10458 203.7091 -301.9289 309.5333
## 119       3.802242 -205.82305 213.4275 -316.7920 324.3965
```

Fungsi `HoltWinters` memiliki argumen yang sama dengan fungsi `ses()` . Argumen-argumen kedua fungsi dapat dilihat lebih lanjut dengan `?ses()` atau `?HoltWinters` .

Nilai parameter α dari kedua fungsi dapat dioptimalkan menyesuaikan dari *error*-nya paling minimumnya. Caranya adalah dengan membuat parameter $\alpha =$ `NULL` .

```
#SES
ses.opt <- ses(train.ts, h = 10, alpha = NULL)
plot(ses.opt)
```




```
ses.opt
```

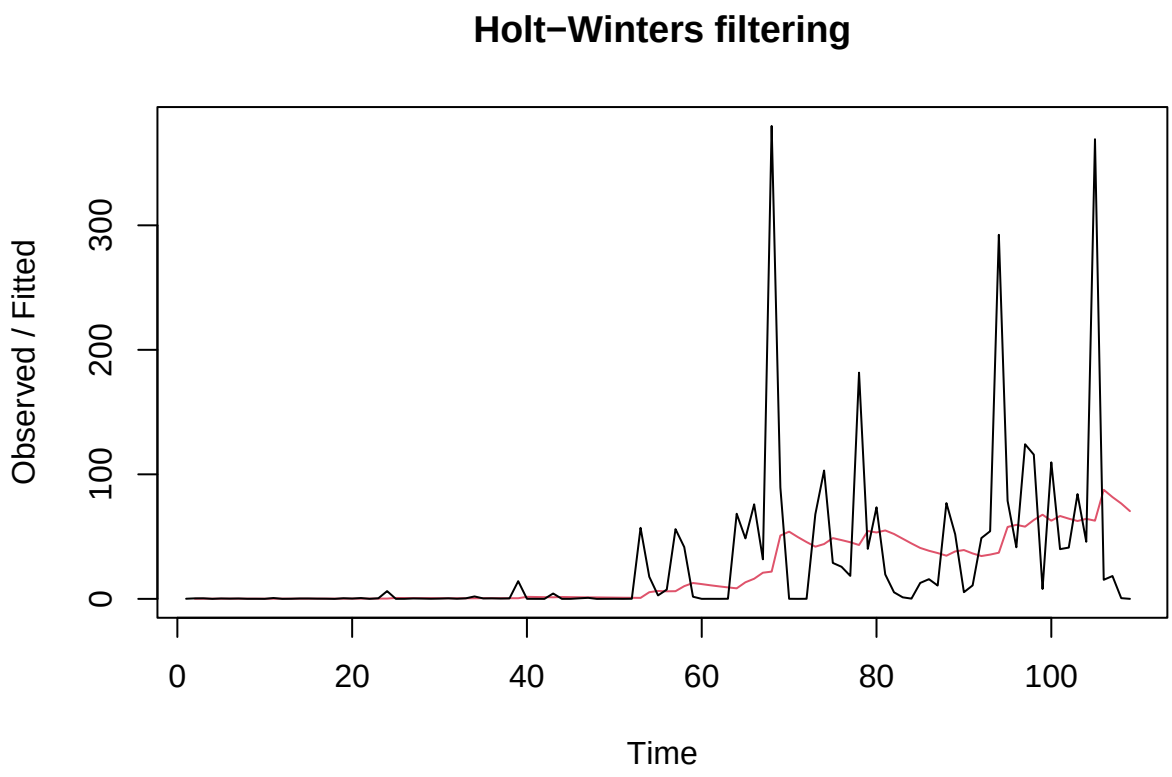
```
##     Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 110       64.73991 -12.36985 141.8497 -53.18929 182.6691
## 111       64.73991 -12.62202 142.1018 -53.57495 183.0548
## 112       64.73991 -12.87337 142.3532 -53.95936 183.4392
## 113       64.73991 -13.12391 142.6037 -54.34252 183.8224
## 114       64.73991 -13.37364 142.8535 -54.72446 184.2043
## 115       64.73991 -13.62258 143.1024 -55.10518 184.5850
## 116       64.73991 -13.87073 143.3506 -55.48470 184.9645
## 117       64.73991 -14.11811 143.5979 -55.86302 185.3428
## 118       64.73991 -14.36470 143.8445 -56.24016 185.7200
## 119       64.73991 -14.61053 144.0904 -56.61612 186.0959
```

```
#Lamda Optimum Holt Winter
sesopt<- HoltWinters(train.ts, gamma = FALSE, beta = FALSE,alpha = NULL)
sesopt
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = train.ts, alpha = NULL, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.08098375
```

```
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 64.74224
```

```
plot(sesopt)
```




```
#ramalan
ramalanopt<- forecast(sesopt, h=10)
ramalanopt
```

```
##     Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 110       64.74224 -11.77849 141.2630 -52.28612 181.7706
## 111       64.74224 -12.02901 141.5135 -52.66925 182.1537
## 112       64.74224 -12.27871 141.7632 -53.05113 182.5356
## 113       64.74224 -12.52760 142.0121 -53.43178 182.9163
## 114       64.74224 -12.77570 142.2602 -53.81121 183.2957
## 115       64.74224 -13.02300 142.5075 -54.18943 183.6739
## 116       64.74224 -13.26952 142.7540 -54.56645 184.0509
## 117       64.74224 -13.51526 142.9997 -54.94228 184.4268
## 118       64.74224 -13.76024 143.2447 -55.31693 184.8014
## 119       64.74224 -14.00445 143.4889 -55.69042 185.1749
```

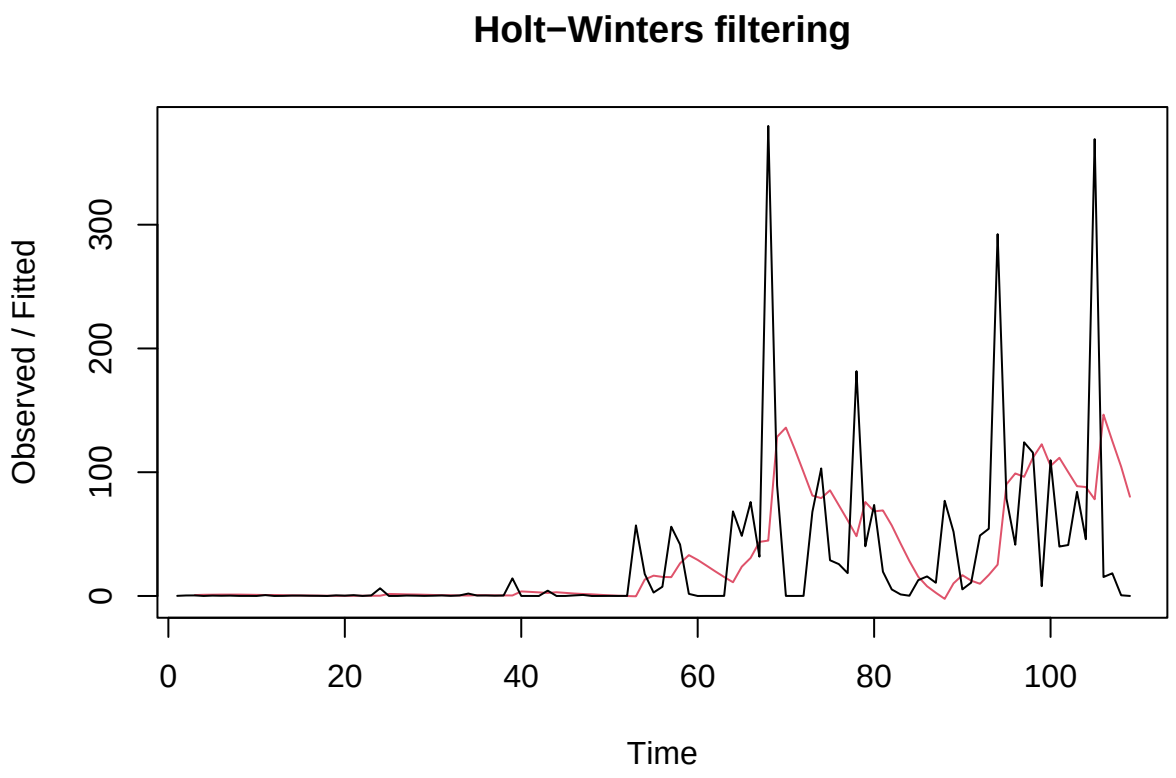
Setelah dilakukan peramalan, akan dilakukan perhitungan keakuratan hasil peramalan. Perhitungan akurasi ini dilakukan baik pada data latih dan data uji.

DES

Metode pemulusan *Double Exponential Smoothing* (DES) digunakan untuk data yang memiliki pola tren. Metode DES adalah metode semacam SES, hanya saja dilakukan dua kali, yaitu pertama untuk tahapan 'level' dan kedua untuk tahapan 'tren'. Pemulusan menggunakan metode ini akan menghasilkan peramalan tidak konstan untuk periode berikutnya.

Pemulusan dengan metode DES kali ini akan menggunakan fungsi `HoltWinters()` . Jika sebelumnya nilai argumen `beta` dibuat `FALSE` , kali ini argumen tersebut akan diinisialisasi bersamaan dengan nilai `alpha` .

```
#Lamda=0.2 dan gamma=0.2
des.1<- HoltWinters(train.ts, gamma = FALSE, beta = 0.2, alpha = 0.2)
plot(des.1)
```

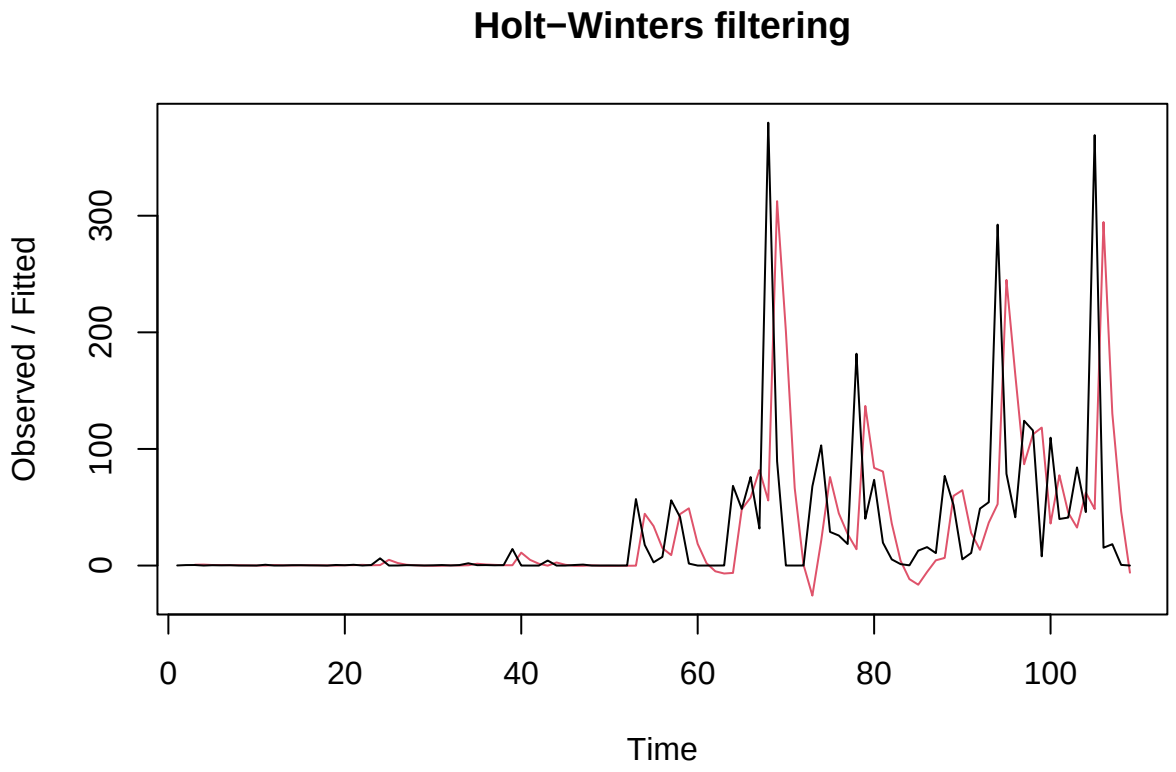



```
#ramalan
ramalandes1<- forecast(des.1, h=12)
ramalandes1
```

```
##     Point Forecast      Lo 80    Hi 80      Lo 95    Hi 95
## 110      57.468035  -25.71732 140.6534  -69.75299 184.6891
## 111      50.762781  -34.78477 136.3103  -80.07092 181.5965
## 112      44.057527  -44.60417 132.7192  -91.53884 179.6539
```

```
## 113      37.352273  -55.21924 129.9238 -104.22364 178.9282
## 114      30.647019  -66.64785 127.9419 -118.15264 179.4467
## 115      23.941765  -78.88557 126.7691 -133.31907 181.2026
## 116      17.236510  -91.91079 126.3838 -149.68989 184.1629
## 117      10.531256 -105.69033 126.7528 -167.21432 188.2768
## 118       3.826002 -120.18437 127.8364 -185.83149 193.4835
## 119      -2.879252 -135.35046 129.5920 -205.47649 199.7180
## 120      -9.584506 -151.14633 131.9773 -226.08464 206.9156
## 121     -16.289760 -167.53163 134.9521 -247.59424 215.0147
```

```r
#Lamda=0.6 dan gamma=0.3
des.2<- HoltWinters(train.ts, gamma = FALSE, beta = 0.3, alpha = 0.6)
plot(des.2)
```



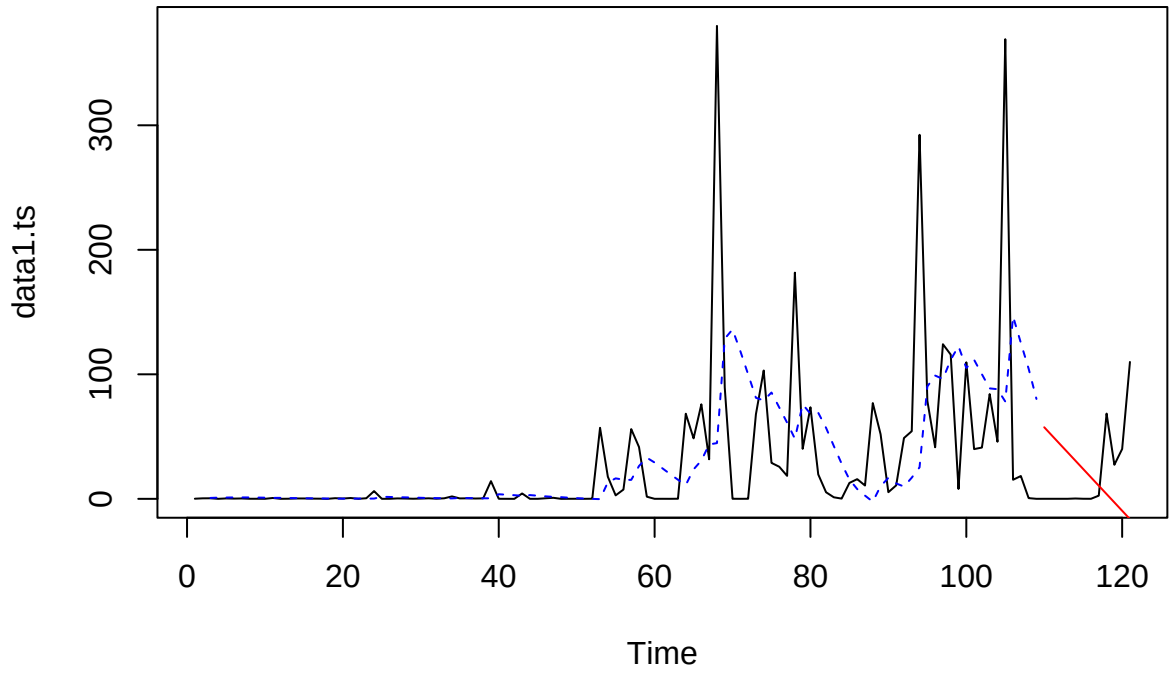

```r
#ramalan
ramalandes2<- forecast(des.2, h=12)
ramalandes2
```

```
##     Point Forecast     Lo 80     Hi 80      Lo 95    Hi 95
## 110      -26.32639 -123.4801  70.82736  -174.9102 122.2575
## 111      -50.22922 -173.4422  72.98381  -238.6673 138.2088
## 112      -74.13204 -228.6645  80.40046  -310.4691 162.2050
## 113      -98.03487 -288.1586  92.08885  -388.8040 192.7343
## 114     -121.93769 -351.2699 107.39456  -472.6711 228.7957
## 115     -145.84052 -417.5586 125.87753  -561.3974 269.7164
```

```
## 116     -169.74334 -486.7148 147.22813  -654.5094 315.0227
## 117     -193.64617 -558.5099 171.21759  -751.6571 364.3648
## 118     -217.54899 -632.7674 197.66937  -852.5707 417.4727
## 119     -241.45182 -709.3457 226.44210  -957.0338 474.1302
## 120     -265.35464 -788.1285 257.41922 -1064.8683 534.1590
## 121     -289.25747 -869.0171 290.50220 -1175.9234 597.4084
```

Selanjutnya jika ingin membandingkan plot data latih dan data uji adalah sebagai berikut.

```
#Visually evaluate the prediction
plot(data1.ts)
lines(des.1$fitted[,1], lty=2, col="blue")
lines(ramalandes1$mean, col="red")
```



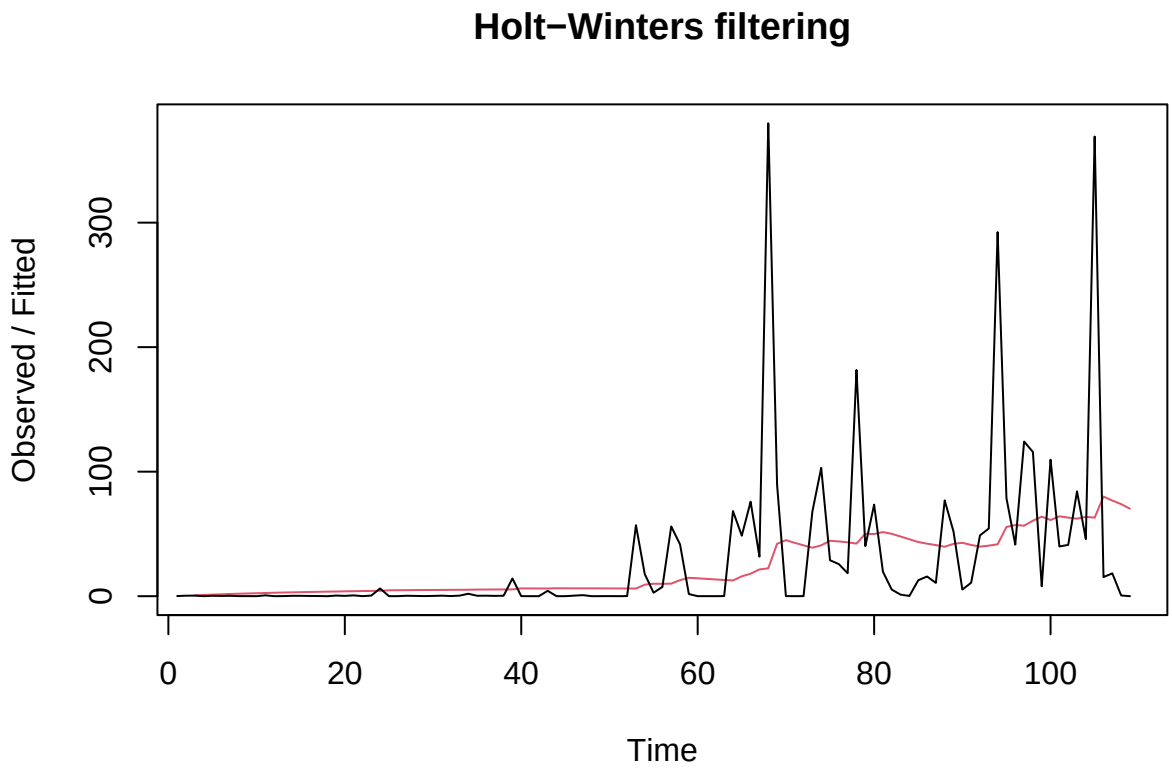
Untuk mendapatkan nilai parameter optimum dari DES, argumen `alpha` dan `beta` dapat dibuat `NULL` seperti berikut.

```
#Lamda dan gamma optimum
des.opt<- HoltWinters(train.ts, gamma = FALSE)
des.opt
```

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

```
##
## Smoothing parameters:
##  alpha: 0.05437989
##  beta : 0
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 66.43145
## b  0.30810
```

```
plot(des.opt)
```




```
#ramalan
ramalandesopt<- forecast(des.opt, h=12)
ramalandesopt
```

```
##     Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 110       66.73955 -9.729078 143.2082 -50.20912 183.6882
## 111       67.04765 -9.533960 143.6293 -50.07381 184.1691
## 112       67.35575 -9.338676 144.0502 -49.93825 184.6497
## 113       67.66385 -9.143226 144.4709 -49.80243 185.1301
## 114       67.97195 -8.947610 144.8915 -49.66636 185.6103
## 115       68.28005 -8.751831 145.3119 -49.53004 186.0901
## 116       68.58815 -8.555888 145.7322 -49.39347 186.5698
```

```
## 117       68.89625 -8.359782 146.1523 -49.25665 187.0491
## 118       69.20435 -8.163514 146.5722 -49.11959 187.5283
## 119       69.51245 -7.967085 146.9920 -48.98227 188.0072
## 120       69.82055 -7.770495 147.4116 -48.84471 188.4858
## 121       70.12865 -7.573745 147.8310 -48.70691 188.9642
```

Selanjutnya akan dilakukan perhitungan akurasi pada data latih maupun data uji dengan ukuran akurasi SSE, MSE dan MAPE.

```
#Akurasi Data Training
ssedes.train1<-des.1$SSE
msedes.train1<-ssedes.train1/length(train.ts)
sisaandes1<-ramalandes1$residuals
head(sisaandes1)
```

Akurasi Data Latih

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]         NA         NA -0.2681000 -0.9418560 -0.7231866 -0.9403235
```

```
mapedes.train1 <- sum(abs(sisaandes1[3:length(train.ts)]/train.ts[3:length(train.ts)])
                      *100)/length(train.ts)

akurasides.1 <- matrix(c(ssedes.train1,msedes.train1,mapedes.train1))
row.names(akurasides.1)<- c("SSE", "MSE", "MAPE")
colnames(akurasides.1) <- c("Akurasi lamda=0.2 dan gamma=0.2")
akurasides.1
```

```
##      Akurasi lamda=0.2 dan gamma=0.2
## SSE                        446896.303
## MSE                          4099.966
## MAPE                        22354.875
```

```
ssedes.train2<-des.2$SSE
msedes.train2<-ssedes.train2/length(train.ts)
sisaandes2<-ramalandes2$residuals
head(sisaandes2)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]         NA         NA -0.2681000 -0.7970820 -0.1452000 -0.2793112
```

```
mapedes.train2 <- sum(abs(sisaandes2[3:length(train.ts)]/train.ts[3:length(train.ts)])
                      *100)/length(train.ts)

akurasides.2 <- matrix(c(ssedes.train2,msedes.train2,mapedes.train2))
row.names(akurasides.2)<- c("SSE", "MSE", "MAPE")
colnames(akurasides.2) <- c("Akurasi lamda=0.6 dan gamma=0.3")
akurasides.2
```

```
##      Akurasi lamda=0.6 dan gamma=0.3
## SSE                        609359.195
## MSE                          5590.451
## MAPE                        12536.088
```

Hasil akurasi dari data latih didapatkan skenario 2 dengan lamda=0.6 dan gamma=0.3 memiliki hasil yang lebih baik. Namun untuk kedua skenario dapat dikategorikan peramalan diragukan berdasarkan nilai MAPE-nya.

```
#Akurasi Data Testing
selisihdes1<-ramalandes1$mean-testing$`Akumulasi Hujan`
selisihdes1
```

Akurasi Data Uji

```
## Time Series:
## Start = 110
## End = 121
## Frequency = 1
##  [1]   57.436035   50.724781   44.030527   37.306273   30.376619   23.885765
##  [7]   17.203510    7.981356  -64.646998  -30.213652  -49.616406 -126.209260
```

```
SSEtestingdes1<-sum(selisihdes1^2)
MSEtestingdes1<-SSEtestingdes1/length(testing$`Akumulasi Hujan`)
MAPEtestingdes1<-sum(abs(selisihdes1/testing$`Akumulasi Hujan`)*100)/length(testing$`Akumulasi Hujan`)

selisihdes2<-ramalandes2$mean-testing$`Akumulasi Hujan`
selisihdes2
```

```
## Time Series:
## Start = 110
## End = 121
## Frequency = 1
##  [1]  -26.35839  -50.26722  -74.15904  -98.08087 -122.20809 -145.89652
##  [7] -169.77634 -196.19607 -286.02199 -268.78622 -305.38654 -399.17697
```

```
SSEtestingdes2<-sum(selisihdes2^2)
MSEtestingdes2<-SSEtestingdes2/length(testing$`Akumulasi Hujan`)
MAPEtestingdes2<-sum(abs(selisihdes2/testing$`Akumulasi Hujan`)*100)/length(testing$`Akumulasi Hujan`)

selisihdesopt<-ramalandesopt$mean-testing$`Akumulasi Hujan`
selisihdesopt
```

```
## Time Series:
## Start = 110
## End = 121
## Frequency = 1
##  [1]  66.707546  67.009646  67.328746  67.617846  67.701546  68.224046
##  [7]  68.555146  66.346346   0.731346  42.178046  29.788646 -39.790854
```

```
SSEtestingdesopt<-sum(selisihdesopt^2)
MSEtestingdesopt<-SSEtestingdesopt/length(testing$`Akumulasi Hujan`)
MAPEtestingdesopt<-sum(abs(selisihdesopt/testing$`Akumulasi Hujan`)*100)/length(testing$`Akumulasi Hujan`)

akurasitestingdes <-
  matrix(c(SSEtestingdes1,MSEtestingdes1,MAPEtestingdes1,SSEtestingdes2,MSEtestingdes2,
           MAPEtestingdes2,SSEtestingdesopt,MSEtestingdesopt,MAPEtestingdesopt),
         nrow=3,ncol=3)
row.names(akurasitestingdes)<- c("SSE", "MSE", "MAPE")
colnames(akurasitestingdes) <- c("des ske1","des ske2","des opt")
akurasitestingdes
```

```
##       des ske1  des ske2   des opt
## SSE  34537.943 528536.29 40635.386
## MSE   2878.162  44044.69  3386.282
## MAPE 55327.181 127746.20 94886.690
```

Akurasi Data Latih Perhitungan akurasi data dapat dilakukan dengan cara langsung maupun manual. Secara langsung, nilai akurasi dapat diambil dari objek yang tersimpan pada hasil SES, yaitu *sum of squared errors* (SSE). Nilai akurasi lain dapat dihitung pula dari nilai SSE tersebut.

```
#Keakuratan Metode
#Pada data training
SSE1<-ses1$SSE
MSE1<-ses1$SSE/length(train.ts)
RMSE1<-sqrt(MSE1)

akurasi1 <- matrix(c(SSE1,MSE1,RMSE1))
row.names(akurasi1)<- c("SSE", "MSE", "RMSE")
colnames(akurasi1) <- c("Akurasi lamda=0.2")
akurasi1
```

```
##      Akurasi lamda=0.2
## SSE         402552.1518
## MSE           3693.1390
## RMSE            60.7712
```

```
SSE2<-ses2$SSE
MSE2<-ses2$SSE/length(train.ts)
RMSE2<-sqrt(MSE2)

akurasi2 <- matrix(c(SSE2,MSE2,RMSE2))
row.names(akurasi2)<- c("SSE", "MSE", "RMSE")
colnames(akurasi2) <- c("Akurasi lamda=0.7")
akurasi2
```

```
##      Akurasi lamda=0.7
## SSE       529178.18227
## MSE         4854.84571
## RMSE          69.67672
```

```
#Cara Manual
fitted1<-ramalan1$fitted
sisaan1<-ramalan1$residuals
head(sisaan1)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]          NA  0.30810000  0.28648000 -0.20081600  0.12934720 -0.02752224
```

```
resid1<-training$`Akumulasi Hujan`-ramalan1$fitted
head(resid1)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]          NA  0.30810000  0.28648000 -0.20081600  0.12934720 -0.02752224
```

```
#Cara Manual
SSE.1=sum(sisaan1[2:length(train.ts)]^2)
SSE.1
```

```
## [1] 402552.2
```

```
MSE.1 = SSE.1/length(train.ts)
MSE.1
```

```
## [1] 3693.139
```

```
MAPE.1 = sum(abs(sisaan1[2:length(train.ts)]/train.ts[2:length(train.ts)])*
               100)/length(train.ts)
MAPE.1
```

```
## [1] 16161.69
```

```
akurasi.1 <- matrix(c(SSE.1,MSE.1,MAPE.1))
row.names(akurasi.1)<- c("SSE", "MSE", "MAPE")
colnames(akurasi.1) <- c("Akurasi lamda=0.2")
akurasi.1
```

```
##      Akurasi lamda=0.2
## SSE         402552.152
## MSE           3693.139
## MAPE         16161.692
```

```r
fitted2<-ramalan2$fitted
sisaan2<-ramalan2$residuals
head(sisaan2)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]          NA  0.30810000  0.13243000 -0.39027100  0.17291870 -0.07912439
```

```r
resid2<-training$`Akumulasi Hujan` -ramalan2$fitted
head(resid2)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]          NA  0.30810000  0.13243000 -0.39027100  0.17291870 -0.07912439
```

```r
SSE.2=sum(sisaan2[2:length(train.ts)]^2)
SSE.2
```

```
## [1] 529178.2
```

```r
MSE.2 = SSE.2/length(train.ts)
MSE.2
```

```
## [1] 4854.846
```

```r
MAPE.2 = sum(abs(sisaan2[2:length(train.ts)]/train.ts[2:length(train.ts)])*
               100)/length(train.ts)
MAPE.2
```

```
## [1] 9858.519
```

```r
akurasi.2 <- matrix(c(SSE.2,MSE.2,MAPE.2))
row.names(akurasi.2)<- c("SSE", "MSE", "MAPE")
colnames(akurasi.2) <- c("Akurasi lamda=0.7")
akurasi.2
```

```
##      Akurasi lamda=0.7
## SSE         529178.182
## MSE           4854.846
## MAPE          9858.519
```

Berdasarkan nilai SSE, MSE, RMSE, dan MAPE di antara kedua parameter, nilai parameter $\lambda = 0,7$ menghasilkan akurasi yang lebih baik dibanding $\lambda = 0,2$. Hal ini dilihat dari nilai masing-masing ukuran akurasi yang lebih kecil. Berdasarkan nilai MAPE-nya, hasil ini dapat dikategorikan sebagai peramalan diragukan.