Tugas Praktikum Pertemuan 1-PEMULUSAN

Adinda Shabrina Putri Salsabila

Library/Packages

```
library("forecast")

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

library("TSA")
```

Data

Data yang digunakan adalah data time series harga cabai rawit di Provinsi DKI Jakarta dari tanggal 14 Maret 2024 hingga 31 Juli 2024. Data diambil dari website resmi Badan Pangan Nasional.

Impor Data

```
DataMPDW <- read.csv("https://raw.githubusercontent.com/adindashabrina/dataMPDW/main/Data%20MPDW.csv", header = TRUE, sep=",")
DataMPDW$waktu <- as.Date(DataMPDW$waktu, format = "%Y-%m-%d")
head(DataMPDW)
```

```
##        waktu harga
## 1 2024-03-14 81090
## 2 2024-03-15 80000
## 3 2024-03-16 90670
## 4 2024-03-17 85730
## 5 2024-03-18 84340
## 6 2024-03-19 75040
```

Eksplorasi Data

Melihat data

Melihat menggunakan fungsi `View()`, struktur data menggunakan fungsi `str()`, dan dimensi data menggunakan fungsi `dim()`. Dapat dilihat bahwa data terdiri atas dua variabel yaitu waktu dan harga dengan total 140 baris data.

```
View(DataMPDW)
str(DataMPDW)
```

```
## 'data.frame':    140 obs. of  2 variables:
##  $ waktu: Date, format: "2024-03-14" "2024-03-15" ...
##  $ harga: int  81090 80000 90670 85730 84340 75040 71220 68390 67810 62700
...
```

```
dim(DataMPDW)

## [1] 140   2
```

Mengubah data

Mengubah data agar terbaca sebagai data deret waktu dengan fungsi `ts()` .

```
DataMPDW.ts <- ts(DataMPDW$waktu)
DataMPDW.ts <- ts(DataMPDW$harga)
```

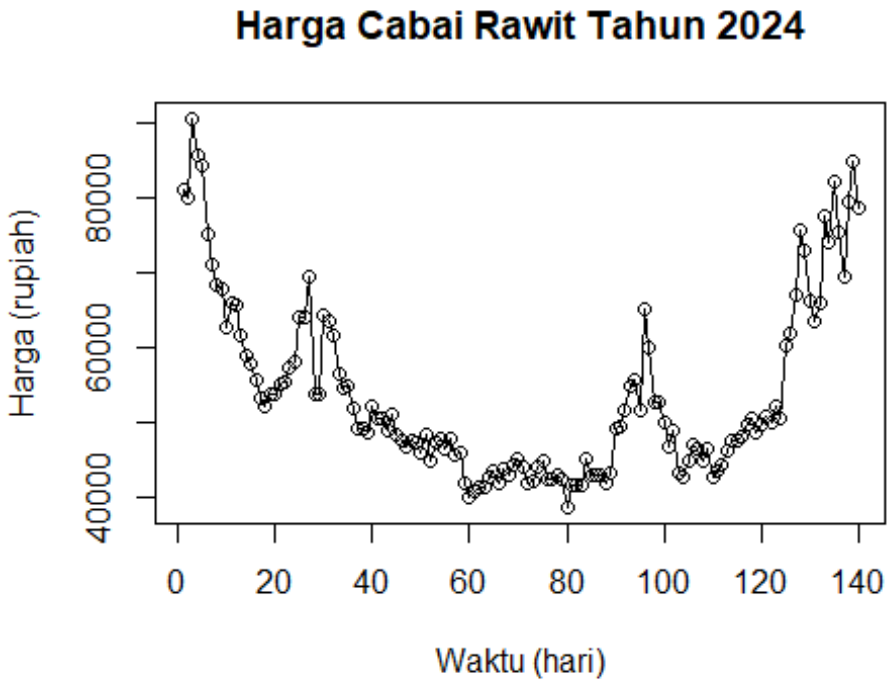
Menampilkan ringkasan data

Ringkasan data yang diperoleh adalah nilai minimun, quartil 1, median, quartil 3, dan nilai maksimum sebagai berikut

```
summary(DataMPDW.ts)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   38540   44705   49780   53811   60573   90670
```

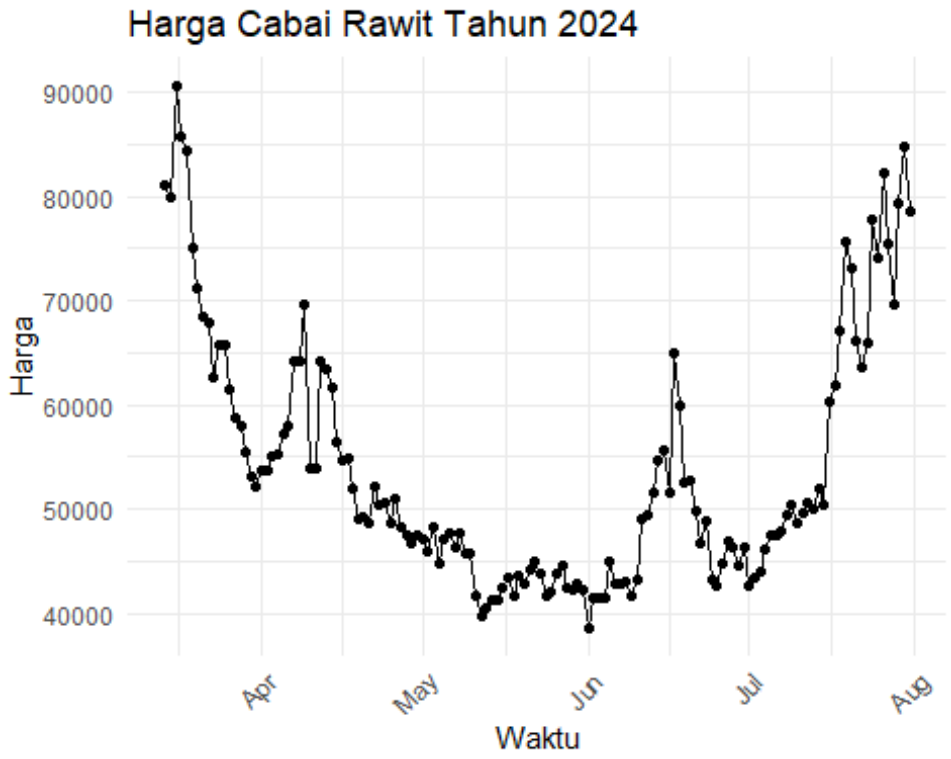
Membuat plot data deret waktu

```
ts.plot(DataMPDW.ts, xlab="Waktu (hari)", ylab="Harga (rupiah)",
        main = "Harga Cabai Rawit Tahun 2024")
points(DataMPDW.ts)
```




```
ggplot(DataMPDW, aes(x = waktu, y = harga)) +
  geom_line() +
```

```
  geom_point() +
  theme_minimal() +
  labs(title = "Harga Cabai Rawit Tahun 2024",
       x = "Waktu",
       y = "Harga") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Single Moving Average & Double Moving Average

Pembagian Data

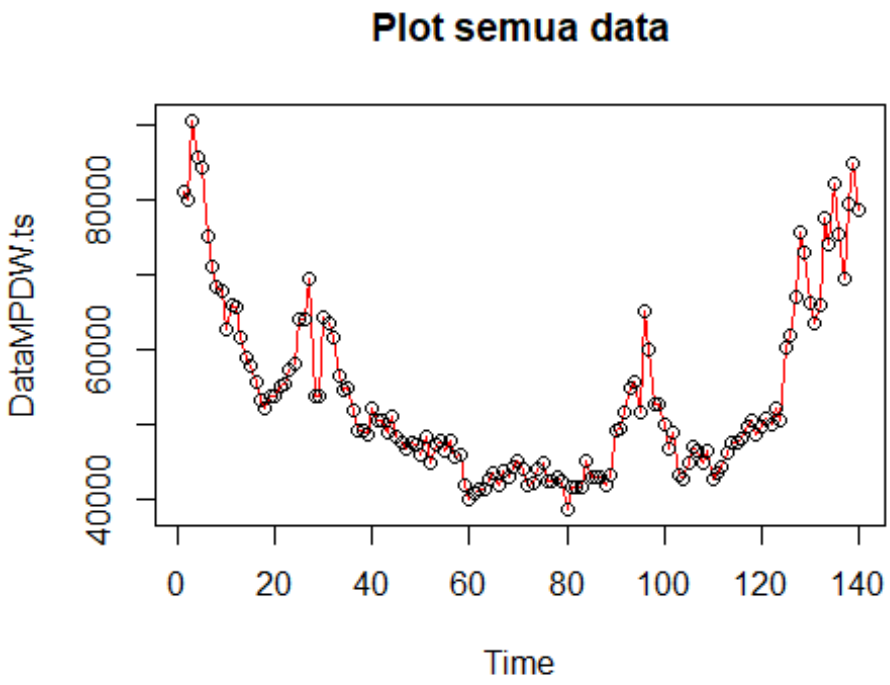
Pembagian data latih dan data uji dilakukan dengan perbandingan 87% data latih dan 13% data uji.Karna data yang saya gunakan berjumlah 140, maka 87% data latih adalah amatan ke 1 sampai 123 dan 17% data uji adalah amatan ke 124 sampai 140. Adapun pembagian persentase untuk masing-masing uji ini didasarkan pada garis amatan yang sekiranya baik berdasarkan grafik di atas.

```
#membagi data latih dan data uji
training_ma <- DataMPDW[1:123,]
testing_ma <- DataMPDW[124:140,]
train_ma.ts <- ts(training_ma$harga)
test_ma.ts <- ts(testing_ma$harga)
```

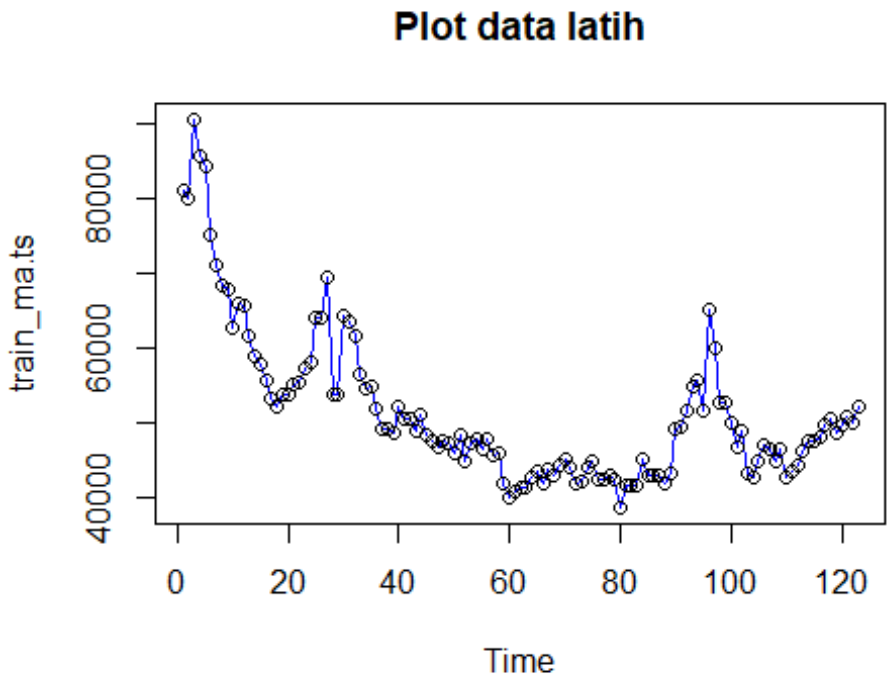
Eksplorasi Data

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

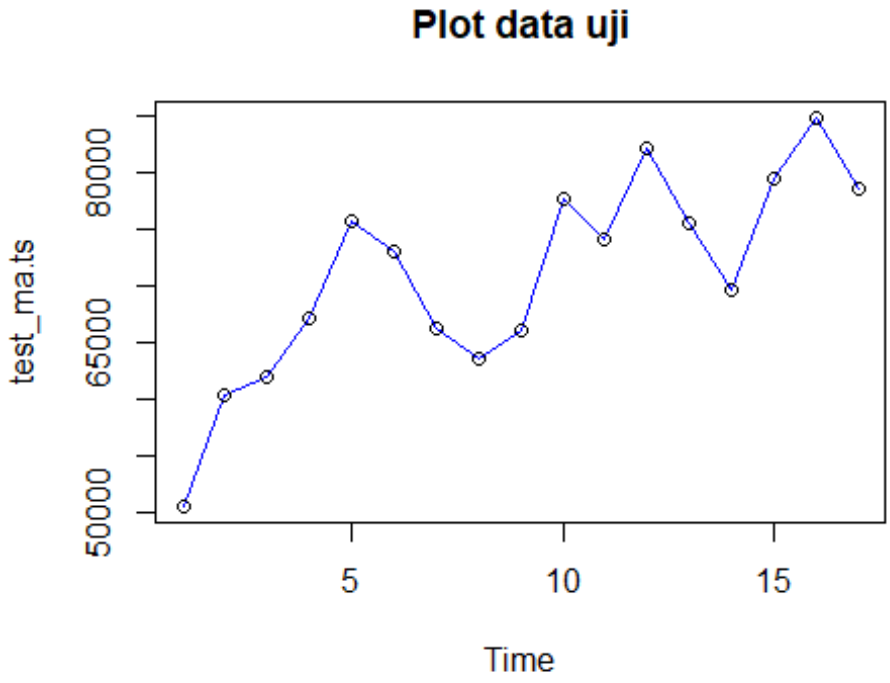
```r
#eksplorasi keseluruhan data
plot(DataMPDW.ts, col="red",main="Plot semua data")
points(DataMPDW.ts)
```




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

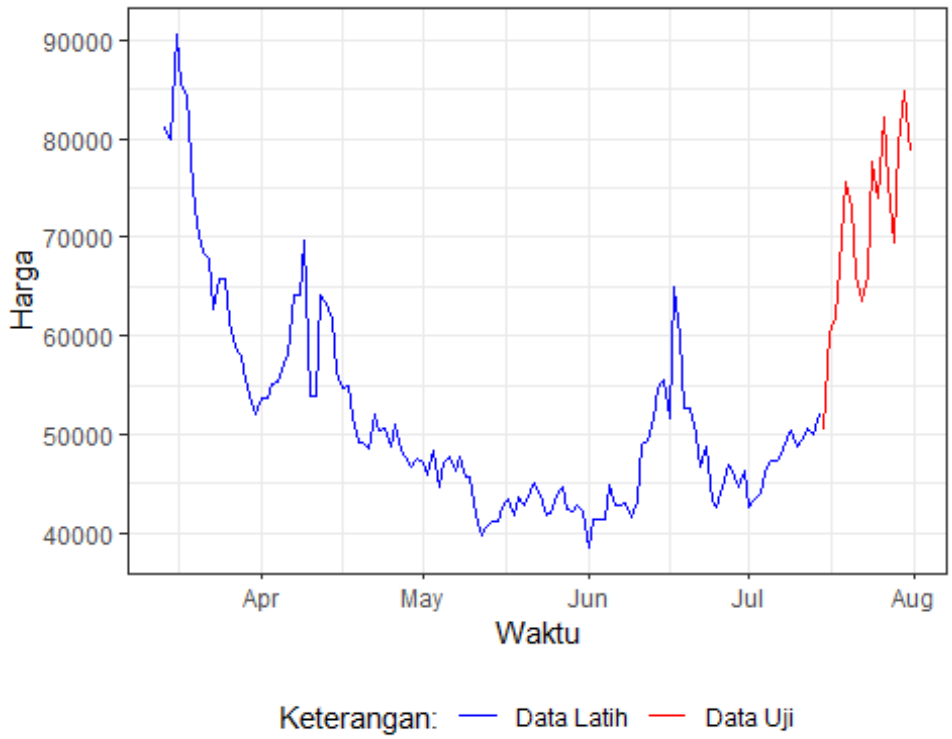



```
#eksplorasi data uji
plot(test_ma.ts, col="blue",main="Plot data uji")
points(test_ma.ts)
```



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

```
library(ggplot2)
ggplot() +
  geom_line(data = training_ma, aes(x = waktu, y = harga, col = "Data Latih")
) +
  geom_line(data = testing_ma, aes(x = waktu, y = harga, col = "Data Uji")) +
  labs(x = "Waktu", y = "Harga", 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=4`.

```
data.sma <- SMA(train_ma.ts, n=4)
data.sma
```

```
## Time Series:
## Start = 1
## End = 123
## Frequency = 1
##   [1]       NA       NA       NA 84372.50 85185.00 83945.00 79082.50 74747
.50
##   [9] 70615.00 67530.00 66177.50 65525.00 63957.50 62982.50 61010.00 58452
.50
##  [17] 56365.00 54685.00 53610.00 53127.50 53590.00 54390.00 55287.50 56400
.00
##  [25] 58665.00 60880.00 63972.50 62907.75 60335.50 60348.00 58798.00 60767
.75
##  [33] 61422.50 59040.00 56905.00 54467.50 52642.50 51272.50 49715.00 49760
.00
##  [41] 50097.50 50442.50 50475.00 50197.50 49647.50 48875.00 48362.50 47482
.50
##  [49] 47207.50 46830.00 47247.50 46555.00 46555.00 47005.00 46480.00 47240
.00
##  [57] 46867.50 46372.50 45242.50 43240.00 41965.00 40820.00 40685.00 41382
.50
##  [65] 42105.00 42237.50 42857.50 42930.00 43115.00 43950.00 43987.50 43712
.50
##  [73] 43165.00 42855.00 43055.00 43225.00 43290.00 43065.00 42457.50 41490
.00
##  [81] 41290.00 40930.00 40752.50 42352.50 42687.50 43045.00 43410.00 42602
.50
##  [89] 42702.50 44232.50 45842.50 48327.50 51202.50 52832.50 53387.50 56725
.00
##  [97] 58042.50 57302.50 57567.50 53797.50 50457.50 49510.00 47157.50 45320
.00
## [105] 44850.00 44375.00 45165.00 45687.50 46102.50 45032.50 44285.00 44142
.50
## [113] 44090.00 45290.00 46290.00 47252.50 48077.50 48810.00 49122.50 49547
.50
## [121] 49822.50 49727.50 50560.00
```

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       NA       NA       NA 84372.50 85185.00 83945.00 79082
.50
##   [9] 74747.50 70615.00 67530.00 66177.50 65525.00 63957.50 62982.50 61010
.00
##  [17] 58452.50 56365.00 54685.00 53610.00 53127.50 53590.00 54390.00 55287
.50
```

```
##  [25] 56400.00 58665.00 60880.00 63972.50 62907.75 60335.50 60348.00 58798
.00
##  [33] 60767.75 61422.50 59040.00 56905.00 54467.50 52642.50 51272.50 49715
.00
##  [41] 49760.00 50097.50 50442.50 50475.00 50197.50 49647.50 48875.00 48362
.50
##  [49] 47482.50 47207.50 46830.00 47247.50 46555.00 46555.00 47005.00 46480
.00
##  [57] 47240.00 46867.50 46372.50 45242.50 43240.00 41965.00 40820.00 40685
.00
##  [65] 41382.50 42105.00 42237.50 42857.50 42930.00 43115.00 43950.00 43987
.50
##  [73] 43712.50 43165.00 42855.00 43055.00 43225.00 43290.00 43065.00 42457
.50
##  [81] 41490.00 41290.00 40930.00 40752.50 42352.50 42687.50 43045.00 43410
.00
##  [89] 42602.50 42702.50 44232.50 45842.50 48327.50 51202.50 52832.50 53387
.50
##  [97] 56725.00 58042.50 57302.50 57567.50 53797.50 50457.50 49510.00 47157
.50
## [105] 45320.00 44850.00 44375.00 45165.00 45687.50 46102.50 45032.50 44285
.00
## [113] 44142.50 44090.00 45290.00 46290.00 47252.50 48077.50 48810.00 49122
.50
## [121] 49547.50 49822.50 49727.50 50560.00
```

Selanjutnya akan dilakukan peramalan sejumlah data uji yaitu 16 hari. Pada metode SMA, hasil peramalan 16 hari ke depan akan bernilai sama dengan hasil peramalan 1 hari kedepan. Dalam hal ini akan dilakukan pengguabungan data aktual train, data hasil pemulusan dan data hasil ramalan 16 hari kedepan.

```
data.gab<-cbind(aktual=c(train_ma.ts,rep(NA,17)),pemulusan=c(data.sma,rep(NA,
17)),ramalan=c(data.ramal,rep(data.ramal[length(data.ramal)],16)))
data.gab #forecast 17 hari ke depan
```

```
##        aktual pemulusan  ramalan
##   [1,]  81090        NA       NA
##   [2,]  80000        NA       NA
##   [3,]  90670        NA       NA
##   [4,]  85730  84372.50       NA
##   [5,]  84340  85185.00 84372.50
##   [6,]  75040  83945.00 85185.00
##   [7,]  71220  79082.50 83945.00
##   [8,]  68390  74747.50 79082.50
##   [9,]  67810  70615.00 74747.50
##  [10,]  62700  67530.00 70615.00
##  [11,]  65810  66177.50 67530.00
##  [12,]  65780  65525.00 66177.50
##  [13,]  61540  63957.50 65525.00
##  [14,]  58800  62982.50 63957.50
```

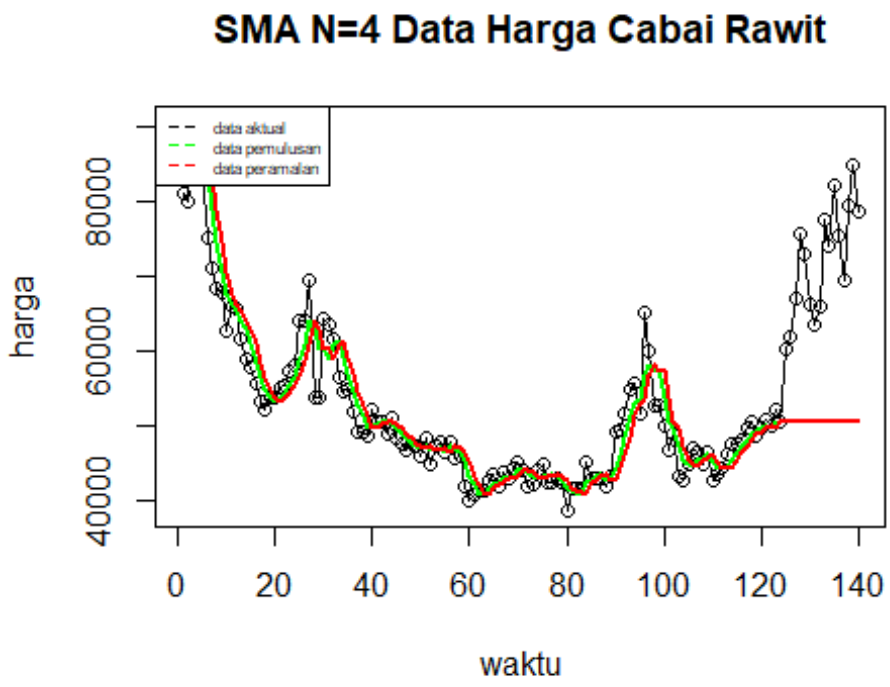
```
##  [15,]  57920  61010.00 62982.50
##  [16,]  55550  58452.50 61010.00
##  [17,]  53190  56365.00 58452.50
##  [18,]  52080  54685.00 56365.00
##  [19,]  53620  53610.00 54685.00
##  [20,]  53620  53127.50 53610.00
##  [21,]  55040  53590.00 53127.50
##  [22,]  55280  54390.00 53590.00
##  [23,]  57210  55287.50 54390.00
##  [24,]  58070  56400.00 55287.50
##  [25,]  64100  58665.00 56400.00
##  [26,]  64140  60880.00 58665.00
##  [27,]  69580  63972.50 60880.00
##  [28,]  53811  62907.75 63972.50
##  [29,]  53811  60335.50 62907.75
##  [30,]  64190  60348.00 60335.50
##  [31,]  63380  58798.00 60348.00
##  [32,]  61690  60767.75 58798.00
##  [33,]  56430  61422.50 60767.75
##  [34,]  54660  59040.00 61422.50
##  [35,]  54840  56905.00 59040.00
##  [36,]  51940  54467.50 56905.00
##  [37,]  49130  52642.50 54467.50
##  [38,]  49180  51272.50 52642.50
##  [39,]  48610  49715.00 51272.50
##  [40,]  52120  49760.00 49715.00
##  [41,]  50480  50097.50 49760.00
##  [42,]  50560  50442.50 50097.50
##  [43,]  48740  50475.00 50442.50
##  [44,]  51010  50197.50 50475.00
##  [45,]  48280  49647.50 50197.50
##  [46,]  47470  48875.00 49647.50
##  [47,]  46690  48362.50 48875.00
##  [48,]  47490  47482.50 48362.50
##  [49,]  47180  47207.50 47482.50
##  [50,]  45960  46830.00 47207.50
##  [51,]  48360  47247.50 46830.00
##  [52,]  44720  46555.00 47247.50
##  [53,]  47180  46555.00 46555.00
##  [54,]  47760  47005.00 46555.00
##  [55,]  46260  46480.00 47005.00
##  [56,]  47760  47240.00 46480.00
##  [57,]  45690  46867.50 47240.00
##  [58,]  45780  46372.50 46867.50
##  [59,]  41740  45242.50 46372.50
##  [60,]  39750  43240.00 45242.50
##  [61,]  40590  41965.00 43240.00
##  [62,]  41200  40820.00 41965.00
##  [63,]  41200  40685.00 40820.00
##  [64,]  42540  41382.50 40685.00
```

```
##  [65,]  43480  42105.00 41382.50
##  [66,]  41730  42237.50 42105.00
##  [67,]  43680  42857.50 42237.50
##  [68,]  42830  42930.00 42857.50
##  [69,]  44220  43115.00 42930.00
##  [70,]  45070  43950.00 43115.00
##  [71,]  43830  43987.50 43950.00
##  [72,]  41730  43712.50 43987.50
##  [73,]  42030  43165.00 43712.50
##  [74,]  43830  42855.00 43165.00
##  [75,]  44630  43055.00 42855.00
##  [76,]  42410  43225.00 43055.00
##  [77,]  42290  43290.00 43225.00
##  [78,]  42930  43065.00 43290.00
##  [79,]  42200  42457.50 43065.00
##  [80,]  38540  41490.00 42457.50
##  [81,]  41490  41290.00 41490.00
##  [82,]  41490  40930.00 41290.00
##  [83,]  41490  40752.50 40930.00
##  [84,]  44940  42352.50 40752.50
##  [85,]  42830  42687.50 42352.50
##  [86,]  42920  43045.00 42687.50
##  [87,]  42950  43410.00 43045.00
##  [88,]  41710  42602.50 43410.00
##  [89,]  43230  42702.50 42602.50
##  [90,]  49040  44232.50 42702.50
##  [91,]  49390  45842.50 44232.50
##  [92,]  51650  48327.50 45842.50
##  [93,]  54730  51202.50 48327.50
##  [94,]  55560  52832.50 51202.50
##  [95,]  51610  53387.50 52832.50
##  [96,]  65000  56725.00 53387.50
##  [97,]  60000  58042.50 56725.00
##  [98,]  52600  57302.50 58042.50
##  [99,]  52670  57567.50 57302.50
## [100,]  49920  53797.50 57567.50
## [101,]  46640  50457.50 53797.50
## [102,]  48810  49510.00 50457.50
## [103,]  43260  47157.50 49510.00
## [104,]  42570  45320.00 47157.50
## [105,]  44760  44850.00 45320.00
## [106,]  46910  44375.00 44850.00
## [107,]  46420  45165.00 44375.00
## [108,]  44660  45687.50 45165.00
## [109,]  46420  46102.50 45687.50
## [110,]  42630  45032.50 46102.50
## [111,]  43430  44285.00 45032.50
## [112,]  44090  44142.50 44285.00
## [113,]  46210  44090.00 44142.50
## [114,]  47430  45290.00 44090.00
```

```
## [115,]  47430  46290.00 45290.00
## [116,]  47940  47252.50 46290.00
## [117,]  49510  48077.50 47252.50
## [118,]  50360  48810.00 48077.50
## [119,]  48680  49122.50 48810.00
## [120,]  49640  49547.50 49122.50
## [121,]  50610  49822.50 49547.50
## [122,]  49980  49727.50 49822.50
## [123,]  52010  50560.00 49727.50
## [124,]     NA        NA 50560.00
## [125,]     NA        NA 50560.00
## [126,]     NA        NA 50560.00
## [127,]     NA        NA 50560.00
## [128,]     NA        NA 50560.00
## [129,]     NA        NA 50560.00
## [130,]     NA        NA 50560.00
## [131,]     NA        NA 50560.00
## [132,]     NA        NA 50560.00
## [133,]     NA        NA 50560.00
## [134,]     NA        NA 50560.00
## [135,]     NA        NA 50560.00
## [136,]     NA        NA 50560.00
## [137,]     NA        NA 50560.00
## [138,]     NA        NA 50560.00
## [139,]     NA        NA 50560.00
## [140,]     NA        NA 50560.00
```

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

```
ts.plot(DataMPDW.ts, xlab="waktu", ylab="harga", main= "SMA N=4 Data Harga Ca
bai Rawit")
points(DataMPDW.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, c
ol=c("black","green","red"), cex=0.5)
```



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.

```
#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)])*100))

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 = 4")
akurasi_train.sma
```

```
##      Akurasi m = 4
## SSE   1.960304e+09
## MSE   1.647314e+07
## MAPE  5.599684e+00
```

Dalam hal ini nilai MAPE data latih pada metode pemulusan SMA kurang dari 10%, nilai ini dapat dikategorikan sebagai nilai akurasi yang sangat baik. Selanjutnya dilakukan perhitungan nilai MAPE data uji pada metde pemulusan SMA.

```
#Menghitung nilai keakuratan data uji
error_test.sma = test_ma.ts-data.gab[124:140,3]
```

```
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 = 4")
akurasi_test.sma
```

```
##      Akurasi m = 4
## SSE   8.349331e+09
## MSE   4.911371e+08
## MAPE  2.755337e+01
```

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

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

t = 1:17
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,17)), pemulusan1 = c(data.sm
a,rep(NA,17)),pemulusan2 = c(data.dma, rep(NA,17)),At = c(At, rep(NA,17)), Bt
= c(Bt,rep(NA,17)),ramalan = c(data.ramal2, f[-1]))
data.gab2
```

```
##        aktual pemulusan1 pemulusan2       At          Bt  ramalan
##   [1,]  81090         NA         NA       NA          NA       NA
##   [2,]  80000         NA         NA       NA          NA       NA
##   [3,]  90670         NA         NA       NA          NA       NA
##   [4,]  85730   84372.50         NA       NA          NA       NA
##   [5,]  84340   85185.00         NA       NA          NA       NA
##   [6,]  75040   83945.00         NA       NA          NA       NA
##   [7,]  71220   79082.50   72309.58 75018.75 -2709.16667       NA
##   [8,]  68390   74747.50   64760.00 68755.00 -3995.00000 72309.58
```

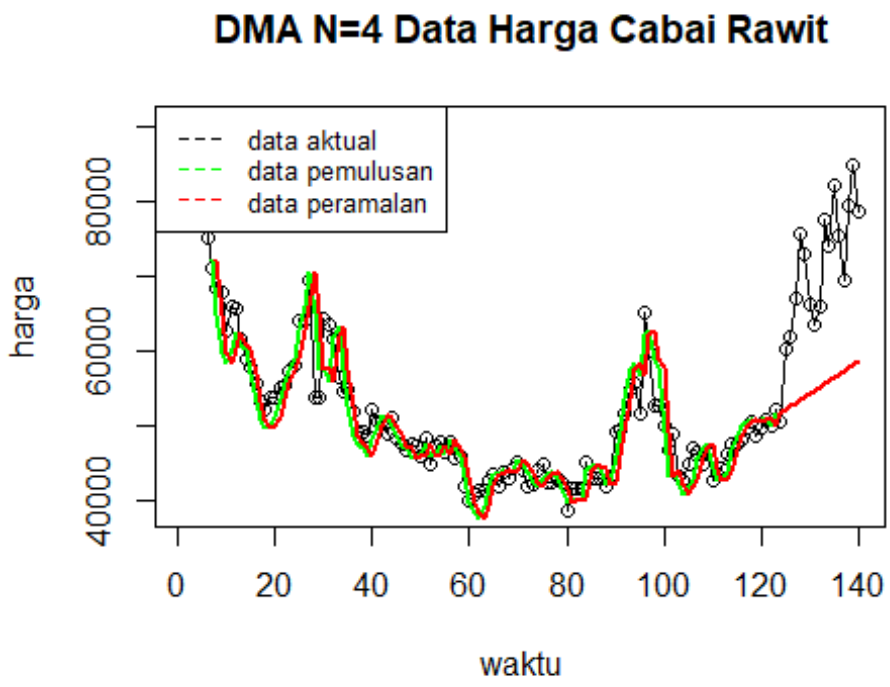
```
##   [9,]  67810   70615.00   59810.83 64132.50 -4321.66667 64760.00
##  [10,]  62700   67530.00   58423.75 62066.25 -3642.50000 59810.83
##  [11,]  65810   66177.50   60194.17 62587.50 -2393.33333 58423.75
##  [12,]  65780   65525.00   62296.88 63588.12 -1291.25000 60194.17
##  [13,]  61540   63957.50   60890.83 62117.50 -1226.66667 62296.88
##  [14,]  58800   62982.50   60185.62 61304.38 -1118.75000 60890.83
##  [15,]  57920   61010.00   57078.75 58651.25 -1572.50000 60185.62
##  [16,]  55550   58452.50   53205.62 55304.38 -2098.75000 57078.75
##  [17,]  53190   56365.00   50802.50 53027.50 -2225.00000 53205.62
##  [18,]  52080   54685.00   49779.79 51741.88 -1962.08333 50802.50
##  [19,]  53620   53610.00   49996.46 51441.88 -1445.41667 49779.79
##  [20,]  53620   53127.50   50928.54 51808.12  -879.58333 49996.46
##  [21,]  55040   53590.00   53318.12 53426.88  -108.75000 50928.54
##  [22,]  55280   54390.00   55574.38 55100.62   473.75000 53318.12
##  [23,]  57210   55287.50   57268.75 56476.25   792.50000 55574.38
##  [24,]  58070   56400.00   58871.88 57883.12   988.75000 57268.75
##  [25,]  64100   58665.00   62797.29 61144.38  1652.91667 58871.88
##  [26,]  64140   60880.00   65999.79 63951.88  2047.91667 62797.29
##  [27,]  69580   63972.50   70627.71 67965.62  2662.08333 65999.79
##  [28,]  53811   62907.75   65076.81 64209.19   867.62500 70627.71
##  [29,]  53811   60335.50   57521.44 58647.06 -1125.62500 65076.81
##  [30,]  64190   60348.00   57776.44 58805.06 -1028.62500 57521.44
##  [31,]  63380   58798.00   55799.15 56998.69 -1199.54167 57776.44
##  [32,]  61690   60767.75   61943.48 61473.19   470.29167 55799.15
##  [33,]  56430   61422.50   63236.56 62510.94   725.62500 61943.48
##  [34,]  54660   59040.00   57428.23 58072.94  -644.70833 63236.56
##  [35,]  54840   56905.00   52523.65 54276.19 -1752.54167 57428.23
##  [36,]  51940   54467.50   48648.75 50976.25 -2327.50000 52523.65
##  [37,]  49130   52642.50   47440.42 49521.25 -2080.83333 48648.75
##  [38,]  49180   51272.50   47023.54 48723.12 -1699.58333 47440.42
##  [39,]  48610   49715.00   45866.04 47405.62 -1539.58333 47023.54
##  [40,]  52120   49760.00   47947.50 48672.50  -725.00000 45866.04
##  [41,]  50480   50097.50   49907.92 49983.75   -75.83333 47947.50
##  [42,]  50560   50442.50   51173.75 50881.25   292.50000 49907.92
##  [43,]  48740   50475.00   50943.75 50756.25   187.50000 51173.75
##  [44,]  51010   50197.50   50021.46 50091.88   -70.41667 50943.75
##  [45,]  48280   49647.50   48742.29 49104.38  -362.08333 50021.46
##  [46,]  47470   48875.00   47335.42 47951.25  -615.83333 48742.29
##  [47,]  46690   48362.50   46848.96 47454.38  -605.41667 47335.42
##  [48,]  47490   47482.50   45633.54 46373.12  -739.58333 46848.96
##  [49,]  47180   47207.50   45916.88 46433.12  -516.25000 45633.54
##  [50,]  45960   46830.00   45762.29 46189.38  -427.08333 45916.88
##  [51,]  48360   47247.50   47340.21 47303.12    37.08333 45762.29
##  [52,]  44720   46555.00   45880.00 46150.00  -270.00000 47340.21
##  [53,]  47180   46555.00   46151.88 46313.12  -161.25000 45880.00
##  [54,]  47760   47005.00   47278.96 47169.38   109.58333 46151.88
##  [55,]  46260   46480.00   46198.75 46311.25  -112.50000 47278.96
##  [56,]  47760   47240.00   47940.00 47660.00   280.00000 46198.75
##  [57,]  45690   46867.50   46816.46 46836.88   -20.41667 47940.00
##  [58,]  45780   46372.50   45760.00 46005.00  -245.00000 46816.46
```

```
##  [59,]  41740   45242.50   43262.29 44054.38  -792.08333 45760.00
##  [60,]  39750   43240.00   39588.96 41049.38 -1460.41667 43262.29
##  [61,]  40590   41965.00   38231.67 39725.00 -1493.33333 39588.96
##  [62,]  41200   40820.00   37491.88 38823.12 -1331.25000 38231.67
##  [63,]  41200   40685.00   39030.83 39692.50  -661.66667 37491.88
##  [64,]  42540   41382.50   41664.79 41551.88   112.91667 39030.83
##  [65,]  43480   42105.00   43533.12 42961.88   571.25000 41664.79
##  [66,]  41730   42237.50   43295.83 42872.50   423.33333 43533.12
##  [67,]  43680   42857.50   44043.96 43569.38   474.58333 43295.83
##  [68,]  42830   42930.00   43592.50 43327.50   265.00000 44043.96
##  [69,]  44220   43115.00   43665.00 43445.00   220.00000 43592.50
##  [70,]  45070   43950.00   45178.12 44686.88   491.25000 43665.00
##  [71,]  43830   43987.50   44807.29 44479.38   327.91667 45178.12
##  [72,]  41730   43712.50   43747.92 43733.75    14.16667 44807.29
##  [73,]  42030   43165.00   42267.08 42626.25  -359.16667 43747.92
##  [74,]  43830   42855.00   41896.67 42280.00  -383.33333 42267.08
##  [75,]  44630   43055.00   42818.54 42913.12   -94.58333 41896.67
##  [76,]  42410   43225.00   43475.00 43375.00   100.00000 42818.54
##  [77,]  42290   43290.00   43596.25 43473.75   122.50000 43475.00
##  [78,]  42930   43065.00   42908.75 42971.25   -62.50000 43596.25
##  [79,]  42200   42457.50   41537.71 41905.62  -367.91667 42908.75
##  [80,]  38540   41490.00   39680.62 40404.38  -723.75000 41537.71
##  [81,]  41490   41290.00   39980.62 40504.38  -523.75000 39680.62
##  [82,]  41490   40930.00   39910.21 40318.12  -407.91667 39980.62
##  [83,]  41490   40752.50   40147.29 40389.38  -242.08333 39910.21
##  [84,]  44940   42352.50   44054.58 43373.75   680.83333 40147.29
##  [85,]  42830   42687.50   44365.62 43694.38   671.25000 44054.58
##  [86,]  42920   43045.00   44437.71 43880.62   557.08333 44365.62
##  [87,]  42950   43410.00   44303.75 43946.25   357.50000 44437.71
##  [88,]  41710   42602.50   42046.25 42268.75  -222.50000 44303.75
##  [89,]  43230   42702.50   42306.67 42465.00  -158.33333 42046.25
##  [90,]  49040   44232.50   45891.88 45228.12   663.75000 42306.67
##  [91,]  49390   45842.50   49171.67 47840.00  1331.66667 45891.88
##  [92,]  51650   48327.50   53412.92 51378.75  2034.16667 49171.67
##  [93,]  54730   51202.50   57537.92 55003.75  2534.16667 53412.92
##  [94,]  55560   52832.50   58301.25 56113.75  2187.50000 57537.92
##  [95,]  51610   53387.50   56637.50 55337.50  1300.00000 58301.25
##  [96,]  65000   56725.00   62038.54 59913.12  2125.41667 56637.50
##  [97,]  60000   58042.50   62701.88 60838.12  1863.75000 62038.54
##  [98,]  52600   57302.50   58866.04 58240.62   625.41667 62701.88
##  [99,]  52670   57567.50   57831.04 57725.62   105.41667 58866.04
## [100,]  49920   53797.50   48997.50 50917.50 -1920.00000 57831.04
## [101,]  46640   50457.50   43251.25 46133.75 -2882.50000 48997.50
## [102,]  48810   49510.00   43971.46 46186.88 -2215.41667 43251.25
## [103,]  43260   47157.50   42035.62 44084.38 -2048.75000 43971.46
## [104,]  42570   45320.00   40667.92 42528.75 -1860.83333 42035.62
## [105,]  44760   44850.00   41751.04 42990.62 -1239.58333 40667.92
## [106,]  46910   44375.00   42623.96 43324.38  -700.41667 41751.04
## [107,]  46420   45165.00   45560.83 45402.50   158.33333 42623.96
## [108,]  44660   45687.50   46801.04 46355.62   445.41667 45560.83
```

```
## [109,]  46420   46102.50   47385.83 46872.50   513.33333 46801.04
## [110,]  42630   45032.50   44258.54 44568.12  -309.58333 47385.83
## [111,]  43430   44285.00   42631.88 43293.12  -661.25000 44258.54
## [112,]  44090   44142.50   42895.62 43394.38  -498.75000 42631.88
## [113,]  46210   44090.00   43594.17 43792.50  -198.33333 42895.62
## [114,]  47430   45290.00   46686.88 46128.12   558.75000 43594.17
## [115,]  47430   46290.00   48518.12 47626.88   891.25000 46686.88
## [116,]  47940   47252.50   49788.96 48774.38  1014.58333 48518.12
## [117,]  49510   48077.50   50327.50 49427.50   900.00000 49788.96
## [118,]  50360   48810.00   50814.17 50012.50   801.66667 50327.50
## [119,]  48680   49122.50   50467.29 49929.38   537.91667 50814.17
## [120,]  49640   49547.50   50644.38 50205.62   438.75000 50467.29
## [121,]  50610   49822.50   50650.62 50319.38   331.25000 50644.38
## [122,]  49980   49727.50   50015.00 49900.00   115.00000 50650.62
## [123,]  52010   50560.00   51636.04 51205.62   430.41667 50015.00
## [124,]     NA         NA         NA       NA          NA 51636.04
## [125,]     NA         NA         NA       NA          NA 52066.46
## [126,]     NA         NA         NA       NA          NA 52496.88
## [127,]     NA         NA         NA       NA          NA 52927.29
## [128,]     NA         NA         NA       NA          NA 53357.71
## [129,]     NA         NA         NA       NA          NA 53788.12
## [130,]     NA         NA         NA       NA          NA 54218.54
## [131,]     NA         NA         NA       NA          NA 54648.96
## [132,]     NA         NA         NA       NA          NA 55079.38
## [133,]     NA         NA         NA       NA          NA 55509.79
## [134,]     NA         NA         NA       NA          NA 55940.21
## [135,]     NA         NA         NA       NA          NA 56370.62
## [136,]     NA         NA         NA       NA          NA 56801.04
## [137,]     NA         NA         NA       NA          NA 57231.46
## [138,]     NA         NA         NA       NA          NA 57661.88
## [139,]     NA         NA         NA       NA          NA 58092.29
## [140,]     NA         NA         NA       NA          NA 58522.71
```

Hasil pemulusan menggunakan metode DMA divisualisasikan sebagai berikut

```
ts.plot(DataMPDW.ts, xlab="waktu", ylab="harga", main= "DMA N=4 Data Harga Ca
bai Rawit")
points(DataMPDW.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, c
ol=c("black","green","red"), cex=0.8)
```



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)])*100))

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 = 4")
akurasi_train.dma

##       Akurasi m = 4
## SSE    1.655536e+09
## MSE    1.427187e+07
## MAPE   5.393489e+00
```

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

```
#Menghitung nilai keakuratan data uji
error_test.dma = test_ma.ts-data.gab2[124:140,6]
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 = 4")
akurasi_test.dma
```

```
##       Akurasi m = 4
## SSE    5.148463e+09
## MSE    3.028508e+08
## MAPE   2.164339e+01
```

Perhitungan akurasi menggunakan data uji menghasilkan nilai MAPE yang lebih dari 10% sehingga nilai akurasi ini dapat dikategorikan cukup baik.

KESIMPULAN : Pada data latih maupun data uji, metode DMA lebih baik dibandingkan dengan metode SMA karena nilai MAPE data latih dan data uji pada metode DMA lebih kecil dibandingkan nilai MAPE pada metode 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

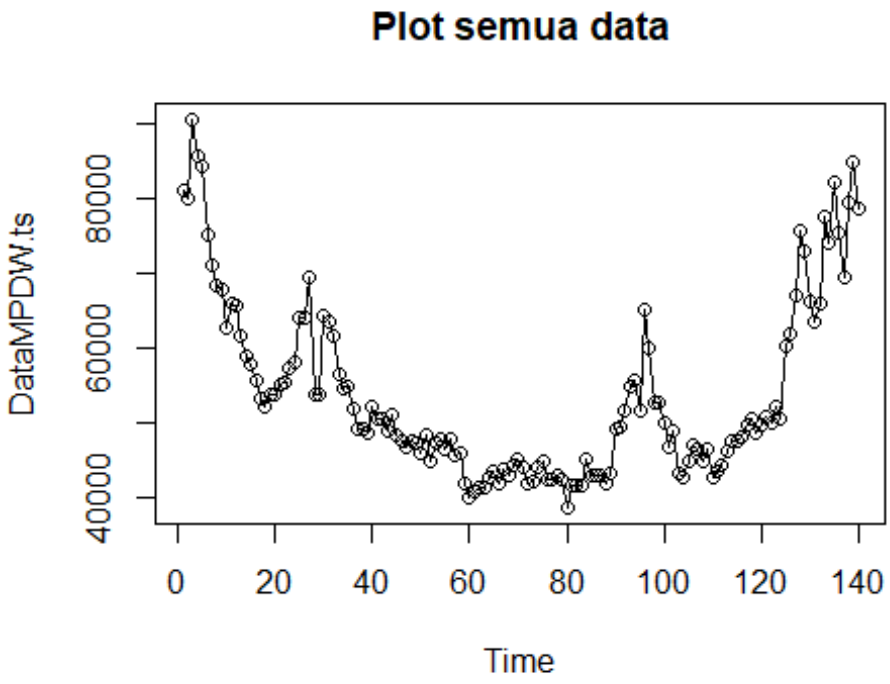
Pembagian data latih dan data uji dilakukan dengan perbandingan 87% data latih dan 13% data uji.Karna data yang saya gunakan berjumlah 140, maka 87% data latih adalah amatan ke 1 sampai 123 dan 17% data uji adalah amatan ke 124 sampai 140.

```
#membagi training dan testing
training<-DataMPDW[1:123,]
testing<-DataMPDW[124:140,]
train.ts <- ts(training$harga)
test.ts <- ts(testing$harga)
```

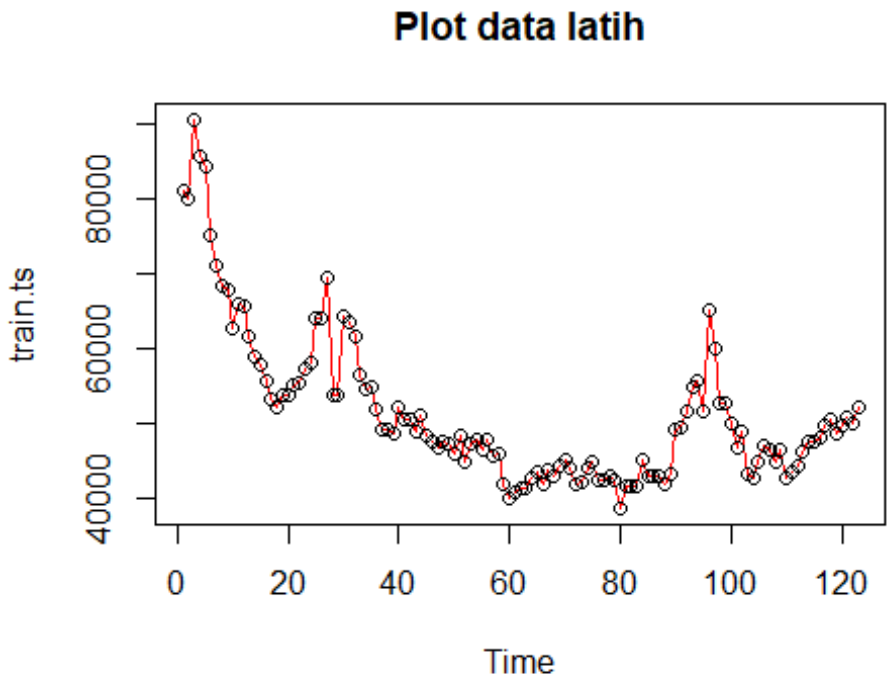
Eksplorasi

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

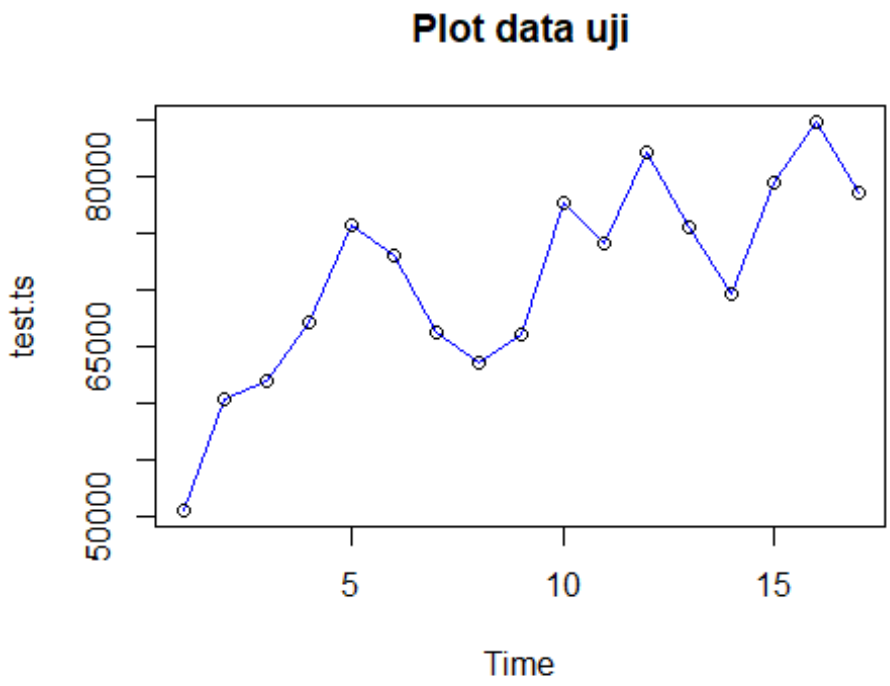
```
#eksplorasi data
plot(DataMPDW.ts, col="black",main="Plot semua data")
points(DataMPDW.ts)
```




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

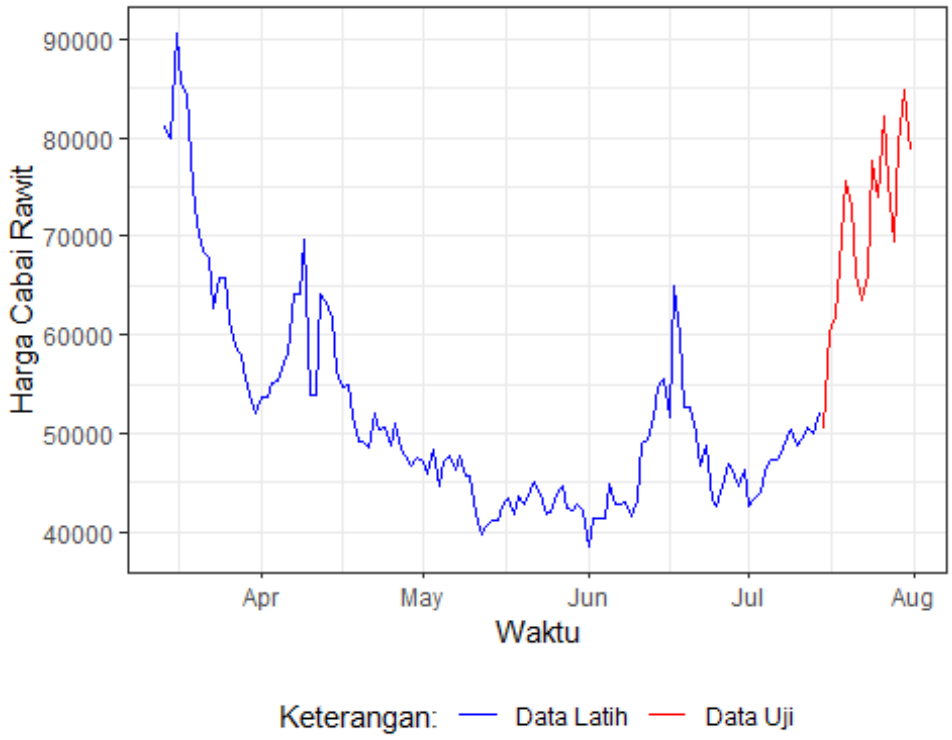


```r
plot(test.ts, col="blue",main="Plot data uji")
points(test.ts)
```



Eksplorasi data juga dapat dilakukan menggunakan package `ggplot2` .

```r
#Eksplorasi dengan GGPLOT
library(ggplot2)
ggplot() +
  geom_line(data = training, aes(x = waktu, y = harga, col = "Data Latih")) +
  geom_line(data = testing, aes(x = waktu, y = harga, col = "Data Uji")) +
  labs(x = "Waktu", y = "Harga Cabai Rawit", 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:

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

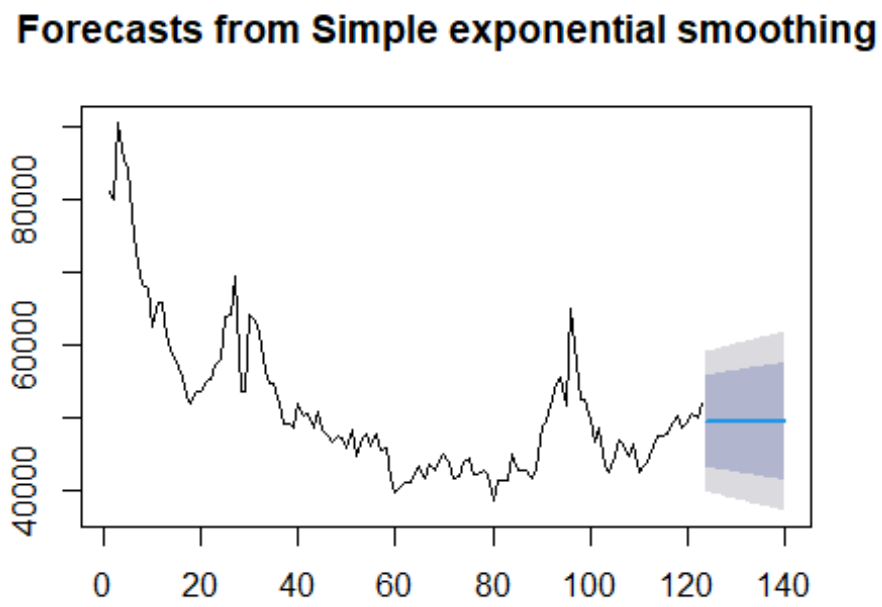
Nilai parameter λ adalah nilai antara 0 dan 1.

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` .

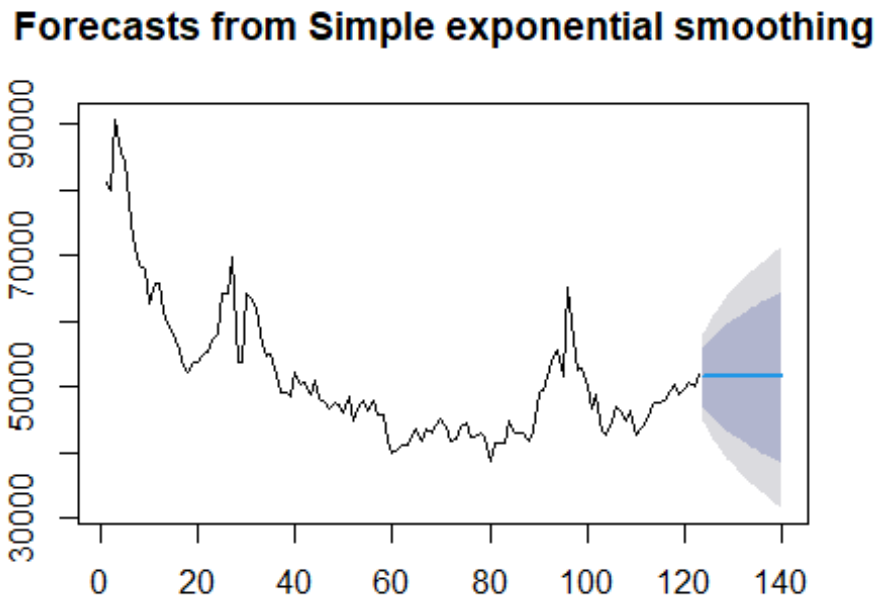
```
#Cara 1 (fungsi ses)
ses.1 <- ses(train.ts, h = 17, alpha = 0.2)
plot(ses.1)
```




```
ses.1
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 124       49585.34 43317.39 55853.30 39999.33 59171.36
## 125       49585.34 43193.26 55977.43 39809.49 59361.20
## 126       49585.34 43071.49 56099.20 39623.27 59547.42
## 127       49585.34 42951.96 56218.73 39440.46 59730.23
## 128       49585.34 42834.55 56336.14 39260.89 59909.80
## 129       49585.34 42719.14 56451.55 39084.39 60086.30
## 130       49585.34 42605.64 56565.05 38910.81 60259.88
## 131       49585.34 42493.96 56676.73 38740.01 60430.68
## 132       49585.34 42384.01 56786.68 38571.85 60598.84
## 133       49585.34 42275.71 56894.98 38406.23 60764.46
## 134       49585.34 42169.00 57001.69 38243.02 60927.67
## 135       49585.34 42063.79 57106.89 38082.13 61088.56
## 136       49585.34 41960.04 57210.65 37923.45 61247.24
## 137       49585.34 41857.69 57313.00 37766.91 61403.78
## 138       49585.34 41756.67 57414.02 37612.42 61558.27
## 139       49585.34 41656.93 57513.75 37459.89 61710.80
## 140       49585.34 41558.44 57612.25 37309.26 61861.43
```

```
ses.2<- ses(train.ts, h = 17, alpha = 0.7)
plot(ses.2)
```

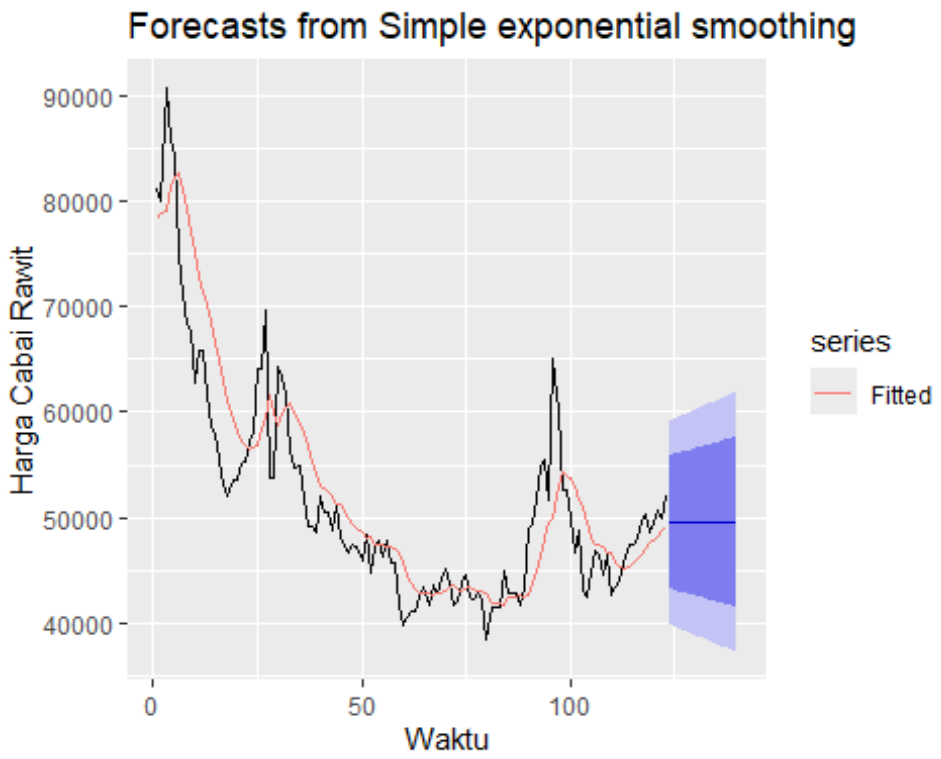



```
ses.2
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 124       51426.81 47067.19 55786.43 44759.34 58094.27
## 125       51426.81 46105.21 56748.40 43288.13 59565.49
## 126       51426.81 45292.28 57561.34 42044.85 60808.76
## 127       51426.81 44575.13 58278.49 40948.06 61905.55
## 128       51426.81 43926.23 58927.38 39955.67 62897.95
## 129       51426.81 43329.17 59524.45 39042.54 63811.08
## 130       51426.81 42773.21 60080.41 38192.27 64661.35
## 131       51426.81 42250.87 60602.75 37393.42 65460.20
## 132       51426.81 41756.70 61096.92 36637.66 66215.96
## 133       51426.81 41286.59 61567.03 35918.68 66934.94
## 134       51426.81 40837.33 62016.29 35231.59 67622.03
## 135       51426.81 40406.36 62447.25 34572.49 68281.13
## 136       51426.81 39991.63 62861.99 33938.21 68915.41
## 137       51426.81 39591.42 63262.20 33326.14 69527.47
## 138       51426.81 39204.31 63649.31 32734.11 70119.51
## 139       51426.81 38829.09 64024.53 32160.26 70693.36
## 140       51426.81 38464.72 64388.89 31603.01 71250.61
```

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("Harga Cabai Rawit") + xlab("Waktu")
```

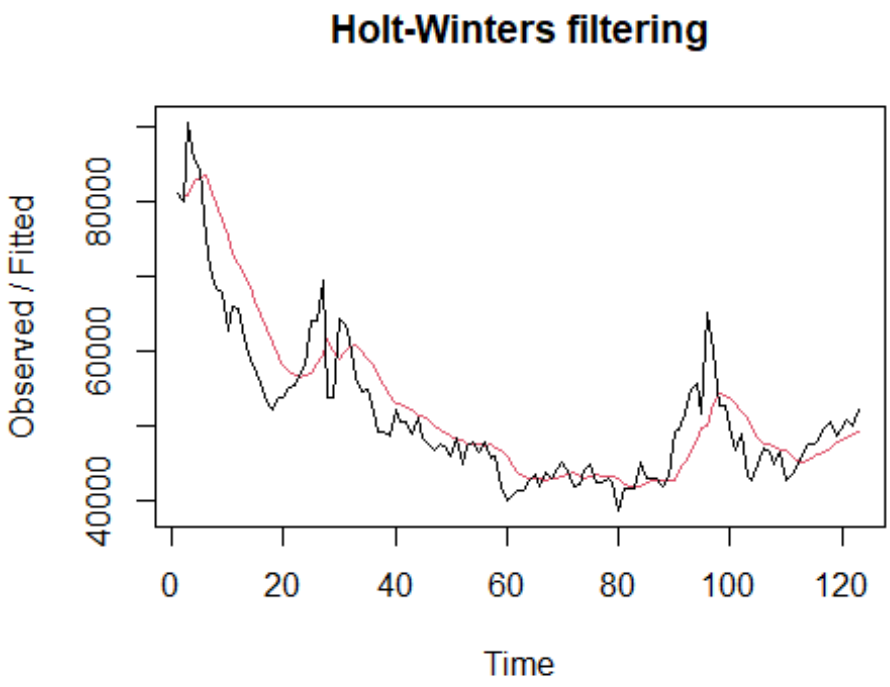


Pada fungsi `ses()` , terdapat beberapa argumen yang umum digunakan, yaitu nilia `y` , `gamma` , `beta` , `alpha` , dan `h` .

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 17 hari. Selanjutnya akan digunakan fungsi `HoltWinters()` dengan nilai inisialisasi parameter dan panjang periode peramalan yang sama dengan fungsi `ses()` .

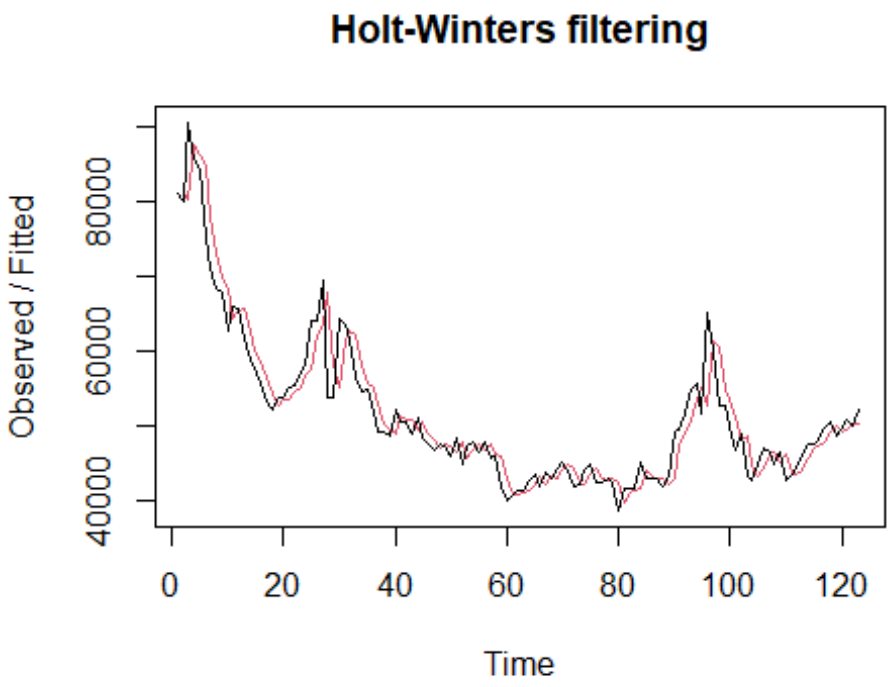
```
#Cara 2 (fungsi Holtwinter)
ses1<- HoltWinters(train.ts, gamma = FALSE, beta = FALSE, alpha = 0.2)
plot(ses1)
```




```
#ramalan
ramalan1<- forecast(ses1, h=17)
ramalan1
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 124       49585.34 43516.99 55653.70 40304.60 58866.09
## 125       49585.34 43396.81 55773.87 40120.80 59049.88
## 126       49585.34 43278.93 55891.76 39940.51 59230.18
## 127       49585.34 43163.20 56007.49 39763.53 59407.16
## 128       49585.34 43049.53 56121.16 39589.68 59581.01
## 129       49585.34 42937.80 56232.89 39418.80 59751.89
## 130       49585.34 42827.91 56342.78 39250.74 59919.95
## 131       49585.34 42719.79 56450.90 39085.38 60085.31
## 132       49585.34 42613.34 56557.35 38922.58 60248.11
## 133       49585.34 42508.49 56662.20 38762.23 60408.46
## 134       49585.34 42405.17 56765.52 38604.22 60566.47
## 135       49585.34 42303.32 56867.37 38448.45 60722.24
## 136       49585.34 42202.87 56967.81 38294.83 60875.86
## 137       49585.34 42103.78 57066.91 38143.27 61027.42
## 138       49585.34 42005.97 57164.72 37993.70 61176.99
## 139       49585.34 41909.42 57261.27 37846.03 61324.66
## 140       49585.34 41814.06 57356.63 37700.19 61470.50
```

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



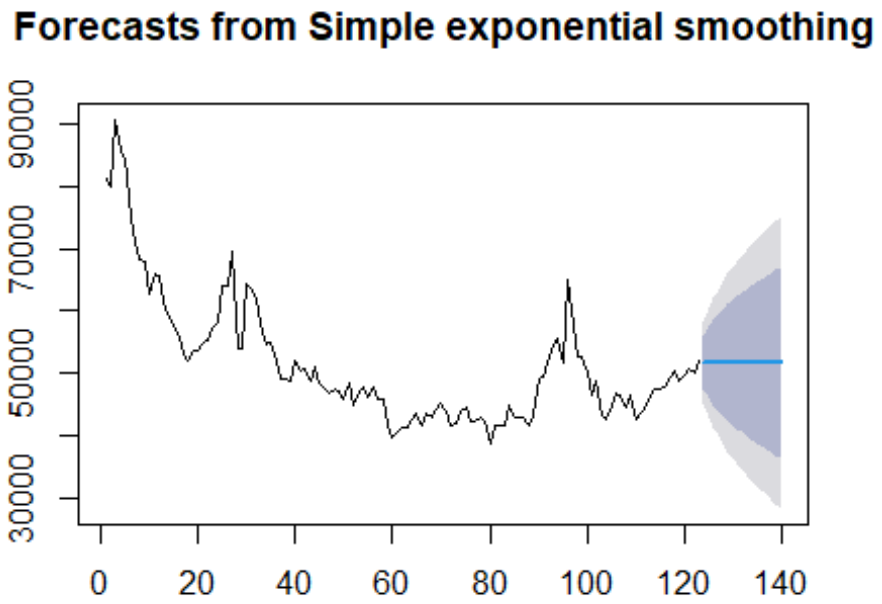

```
#ramalan
ramalan2<- forecast(ses2, h=17)
ramalan2
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 124       51426.81 47089.80 55763.82 44793.93 58059.69
## 125       51426.81 46132.81 56720.80 43330.34 59523.27
## 126       51426.81 45324.10 57529.52 42093.52 60760.10
## 127       51426.81 44610.66 58242.95 41002.41 61851.20
## 128       51426.81 43965.14 58888.48 40015.17 62838.45
## 129       51426.81 43371.17 59482.45 39106.78 63746.84
## 130       51426.81 42818.09 60035.53 38260.91 64592.71
## 131       51426.81 42298.46 60555.16 37466.21 65387.41
## 132       51426.81 41806.86 61046.76 36714.36 66139.25
## 133       51426.81 41339.18 61514.43 35999.12 66854.50
## 134       51426.81 40892.25 61961.37 35315.59 67538.03
## 135       51426.81 40463.52 62390.09 34659.91 68193.71
## 136       51426.81 40050.94 62802.68 34028.92 68824.70
## 137       51426.81 39652.81 63200.81 33420.03 69433.59
## 138       51426.81 39267.70 63585.91 32831.06 70022.55
## 139       51426.81 38894.43 63959.19 32260.19 70593.43
## 140       51426.81 38531.95 64321.66 31705.83 71147.79
```

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 = 17, alpha = NULL)
plot(ses.opt)
```



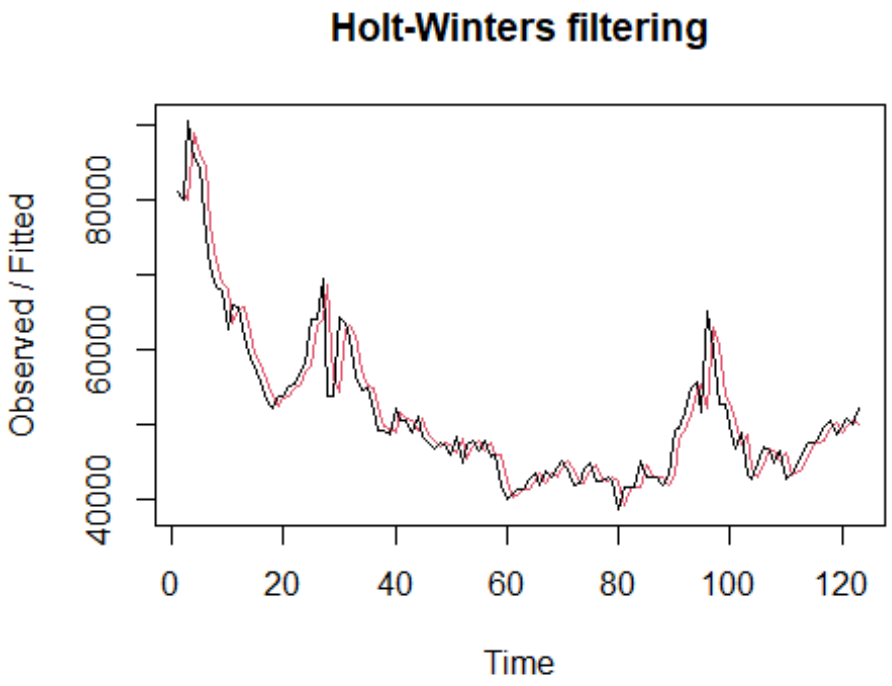

```
ses.opt
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 124       51716.41 47405.70 56027.12 45123.74 58309.07
## 125       51716.41 46058.31 57374.51 43063.09 60369.72
## 126       51716.41 44975.04 58457.77 41406.38 62026.43
## 127       51716.41 44043.22 59389.60 39981.28 63451.54
## 128       51716.41 43212.90 60219.92 38711.41 64721.40
## 129       51716.41 42456.73 60976.08 37554.96 65877.86
## 130       51716.41 41757.82 61674.99 36486.06 66946.75
## 131       51716.41 41104.84 62327.97 35487.42 67945.39
## 132       51716.41 40489.78 62943.03 34546.76 68886.05
## 133       51716.41 39906.71 63526.11 33655.03 69777.79
## 134       51716.41 39351.10 64081.72 32805.30 70627.52
## 135       51716.41 38819.40 64613.41 31992.14 71440.68
## 136       51716.41 38308.77 65124.04 31211.20 72221.61
## 137       51716.41 37816.89 65615.92 30458.93 72973.88
## 138       51716.41 37341.83 66090.98 29732.39 73700.42
## 139       51716.41 36881.98 66550.83 29029.11 74403.71
## 140       51716.41 36435.96 66996.85 28346.98 75085.84
```

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

```
## Holt-Winters exponential smoothing without trend and without seasonal comp
onent.
##
## Call:
## HoltWinters(x = train.ts, alpha = NULL, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.8502217
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 51716.46
```

```r
plot(sesopt)
```




```r
#ramalan
ramalanopt<- forecast(sesopt, h=17)
ramalanopt
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 124       51716.46 47421.17 56011.74 45147.38 58285.53
```

```
## 125       51716.46 46078.53 57354.38 43093.99 60338.92
## 126       51716.46 44999.10 58433.81 41443.14 61989.77
## 127       51716.46 44070.57 59362.34 40023.08 63409.83
## 128       51716.46 43243.18 60189.73 38757.71 64675.20
## 129       51716.46 42489.70 60943.21 37605.35 65827.56
## 130       51716.46 41793.26 61639.65 36540.24 66892.67
## 131       51716.46 41142.60 62290.31 35545.14 67887.77
## 132       51716.46 40529.72 62903.19 34607.81 68825.10
## 133       51716.46 39948.71 63484.20 33719.24 69713.67
## 134       51716.46 39395.07 64037.84 32872.52 70560.39
## 135       51716.46 38865.26 64567.65 32062.25 71370.66
## 136       51716.46 38356.44 65076.47 31284.08 72148.83
## 137       51716.46 37866.30 65566.61 30534.48 72898.43
## 138       51716.46 37392.93 66039.98 29810.51 73622.40
## 139       51716.46 36934.71 66498.20 29109.72 74323.19
## 140       51716.46 36490.27 66942.64 28430.01 75002.90
```

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

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       2.916422e+09
## MSE       2.371075e+07
## RMSE      4.869368e+03
```

```
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       1.400498e+09
## MSE       1.138616e+07
## RMSE      3.374338e+03
```

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

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]        NA -1090.000  9798.000  2898.400   928.720 -8557.024
```

```
resid1<-training$harga-ramalan1$fitted
head(resid1)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]        NA -1090.000  9798.000  2898.400   928.720 -8557.024
```

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

```
## [1] 2916421757
```

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

```
## [1] 23710746
```

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

```
## [1] 6.740305
```

```
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       2.916422e+09
## MSE       2.371075e+07
## MAPE      6.740305e+00
```

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

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]        NA -1090.000 10343.000 -1837.100 -1941.130 -9882.339
```

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

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]        NA -1090.000 10343.000 -1837.100 -1941.130 -9882.339
```

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

```
## [1] 1400497659
```

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

```
## [1] 11386160
```

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

```
## [1] 4.379033
```

```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       1.400498e+09
## MSE       1.138616e+07
## MAPE      4.379033e+00
```

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 sangat baik.

Akurasi Data Uji

Akurasi data uji dapat dihitung dengan cara yang hampir sama dengan perhitungan akurasi data latih.

```
selisih1<-ramalan1$mean-testing$harga
SSEtesting1<-sum(selisih1^2)
MSEtesting1<-SSEtesting1/length(testing)

selisih2<-ramalan2$mean-testing$harga
SSEtesting2<-sum(selisih2^2)
MSEtesting2<-SSEtesting2/length(testing)

selisihopt<-ramalanopt$mean-testing$harga
SSEtestingopt<-sum(selisihopt^2)
MSEtestingopt<-SSEtestingopt/length(testing)

akurasitesting1 <- matrix(c(SSEtesting1,SSEtesting2,SSEtestingopt))
row.names(akurasitesting1)<- c("SSE1", "SSE2", "SSEopt")
akurasitesting1
```

```
##              [,1]
## SSE1   9040857528
## SSE2   7761457384
## SSEopt 7570712345
```

```
akurasitesting2 <- matrix(c(MSEtesting1,MSEtesting2,MSEtestingopt))
row.names(akurasitesting2)<- c("MSE1", "MSE2", "MSEopt")
akurasitesting2
```

```
##              [,1]
## MSE1   4520428764
## MSE2   3880728692
## MSEopt 3785356172
```

Selain dengan cara di atas, perhitungan nilai akurasi dapat menggunakan fungsi `accuracy()` dari *package* `forecast` . Penggunaannya yaitu dengan menuliskan `accuracy(hasil ramalan, kondisi aktual)` . Contohnya adalah sebagai berikut.

```
#cara lain
accuracy(ramalanopt,testing$harga)
```

```
##                      ME      RMSE       MAE        MPE      MAPE     MASE
## Training set  -283.1812  3349.856  2272.388 -0.6297878  4.248449 1.023984
## Test set     19224.1332 21102.986 19368.422 25.8796183 26.165395 8.727801
##                      ACF1
## Training set 0.0001672648
## Test set               NA
```

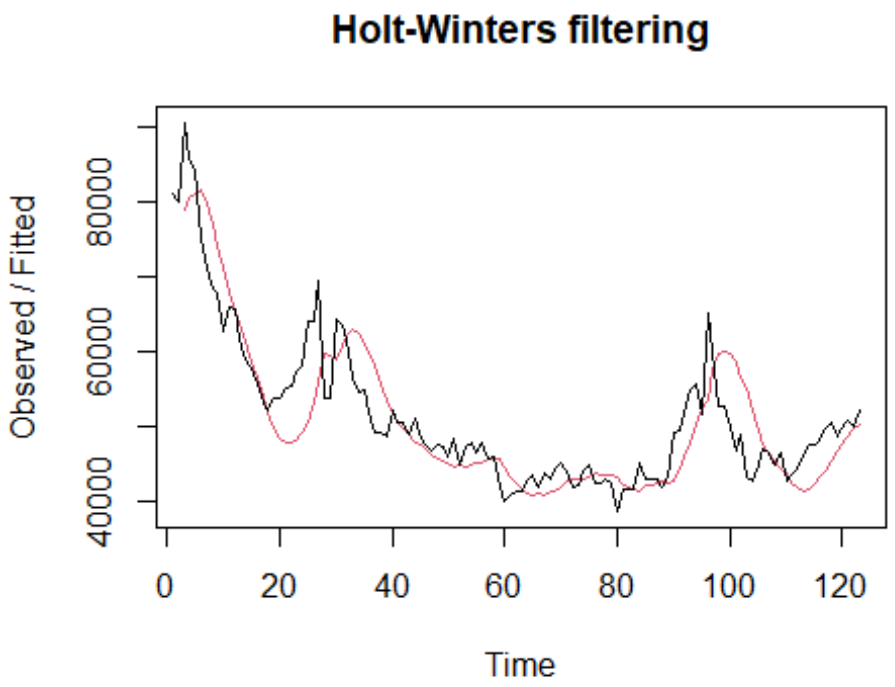
Perhitungan akurasi menggunakan data uji menghasilkan nilai MAPE yang lebih dari 10% sehingga nilai akurasi ini dapat dikategorikan sebagai cukup baik.

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)
```

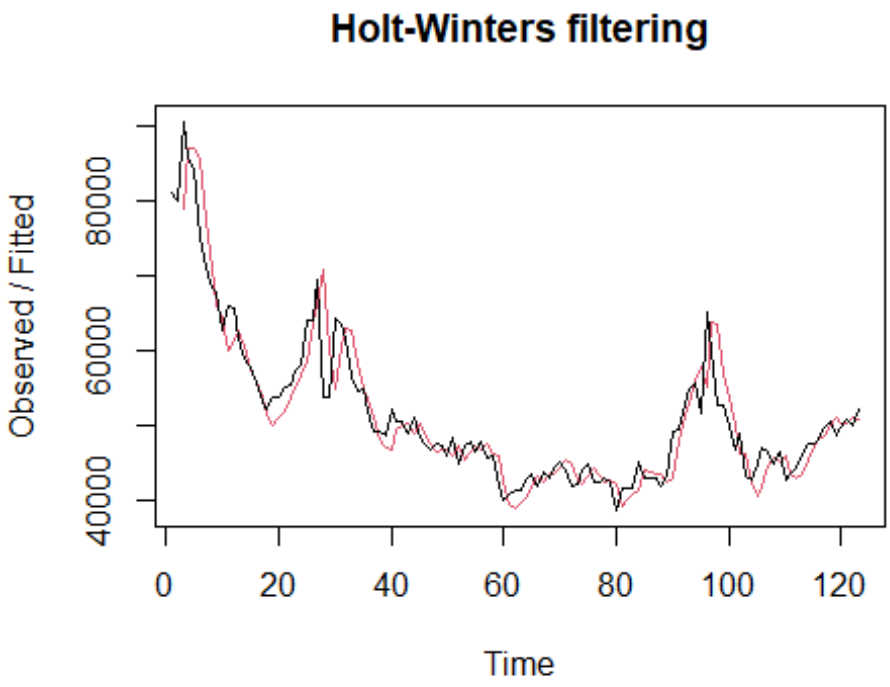



```
#ramalan
ramalandes1<- forecast(des.1, h=17)
ramalandes1
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 124       51343.15 44807.22 57879.07 41347.31 61338.98
## 125       52054.71 45333.19 58776.24 41775.03 62334.40
## 126       52766.28 45800.08 59732.48 42112.39 63420.17
## 127       53477.84 46204.44 60751.24 42354.14 64601.55
## 128       54189.41 46544.89 61833.93 42498.13 65880.69
## 129       54900.97 46821.77 62980.18 42544.90 67257.05
```

```
## 130       55612.54 47036.77 64188.31 42497.03 68728.05
## 131       56324.11 47192.50 65455.71 42358.53 70289.68
## 132       57035.67 47292.10 66779.24 42134.17 71937.17
## 133       57747.24 47338.89 68155.58 41829.05 73665.42
## 134       58458.80 47336.20 69581.40 41448.25 75469.35
## 135       59170.37 47287.20 71053.53 40996.63 77344.10
## 136       59881.93 47194.83 72569.03 40478.68 79285.18
## 137       60593.50 47061.77 74125.23 39898.50 81288.49
## 138       61305.06 46890.42 75719.70 39259.77 83350.35
## 139       62016.63 46682.95 77350.30 38565.80 85467.46
## 140       62728.19 46441.28 79015.11 37819.51 87636.88
```

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



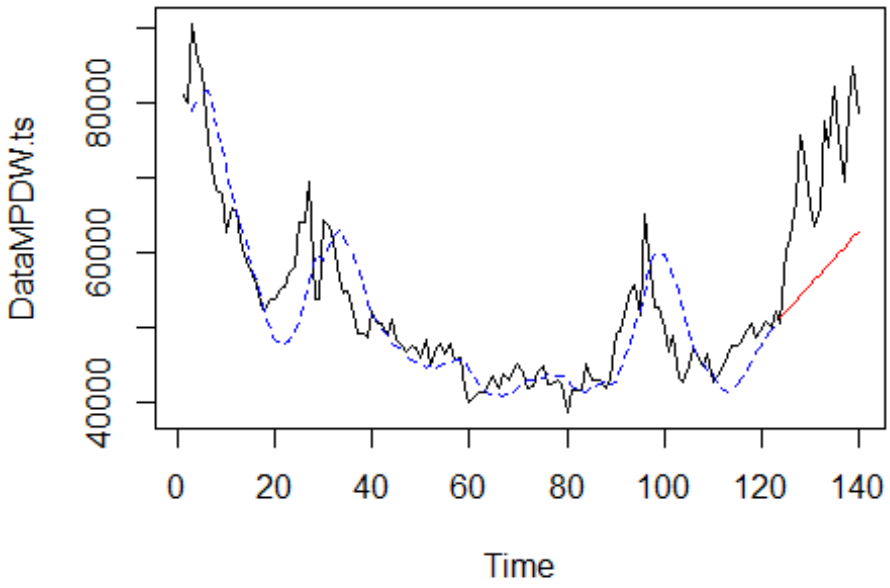

```
#ramalan
ramalandes2<- forecast(des.2, h=17)
ramalandes2
```

```
##     Point Forecast    Lo 80     Hi 80      Lo 95     Hi 95
## 124       51986.50 47318.10  56654.89 44846.8026  59126.19
## 125       52517.43 46596.85  58438.02 43462.6802  61572.19
## 126       53048.37 45622.83  60473.91 41691.9922  64404.75
## 127       53579.31 44443.56  62715.06 39607.3809  67551.23
## 128       54110.24 43090.46  65130.03 37256.9372  70963.55
## 129       54641.18 41584.69  67697.67 34673.0012  74609.36
```

```
## 130       55172.12 39941.13  70403.11 31878.3274  78465.91
## 131       55703.05 38170.76  73235.35 28889.7268  82516.38
## 132       56233.99 36282.08  76185.90 25720.1744  86747.81
## 133       56764.93 34281.87  79247.99 22380.0579  91149.80
## 134       57295.87 32175.74  82415.99 18877.9450  95713.79
## 135       57826.80 29968.41  85685.19 15221.0740 100432.53
## 136       58357.74 27663.98  89051.50 11415.6798 105299.80
## 137       58888.68 25265.99  92511.36  7467.2178 110310.14
## 138       59419.61 22777.62  96061.61  3380.5232 115458.70
## 139       59950.55 20201.70  99699.40  -840.0724 120741.17
## 140       60481.49 17540.78 103422.19 -5190.6542 126153.63
```

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

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

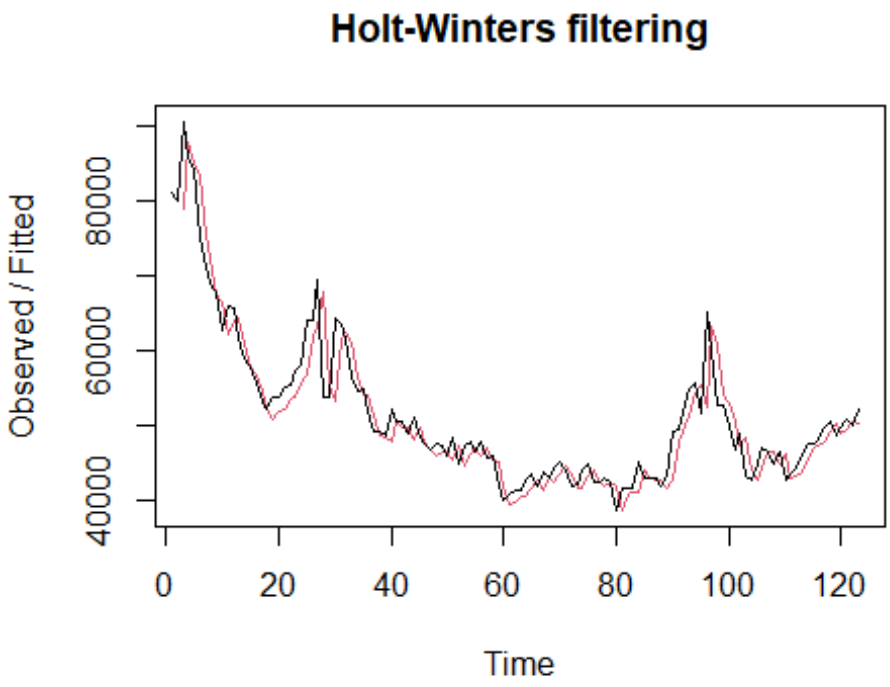


Untuk mendapatkan nilai parameter optimum dari DES, argumen `alpha` dan `beta` dapat dibuat `NULL` seperti berikut.

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

## Holt-Winters exponential smoothing with trend and without seasonal compone
nt.
```

```
##
## Call:
## HoltWinters(x = train.ts, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.8274969
##  beta : 0.02795107
##  gamma: FALSE
##
## Coefficients:
##          [,1]
## a 51672.97243
## b    44.27825
```

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




```
#ramalan
ramalandesopt<- forecast(des.opt, h=17)
ramalandesopt
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 124       51717.25 47400.17 56034.33 45114.84 58319.66
## 125       51761.53 46093.86 57429.20 43093.58 60429.48
## 126       51805.81 44997.65 58613.96 41393.63 62217.98
## 127       51850.09 44017.92 59682.25 39871.82 63828.35
## 128       51894.36 43112.69 60676.04 38463.95 65324.78
## 129       51938.64 42258.97 61618.31 37134.86 66742.42
```

```
## 130       51982.92 41442.65 62523.19 35862.97 68102.87
## 131       52027.20 40654.35 63400.04 34633.93 69420.46
## 132       52071.48 39887.52 64255.44 33437.72 70705.24
## 133       52115.75 39137.35 65094.16 32266.99 71964.52
## 134       52160.03 38400.23 65919.83 31116.24 73203.83
## 135       52204.31 37673.38 66735.24 29981.17 74427.45
## 136       52248.59 36954.59 67542.59 28858.43 75638.75
## 137       52292.87 36242.08 68343.65 27745.31 76840.42
## 138       52337.15 35534.42 69139.87 26639.60 78034.69
## 139       52381.42 34830.42 69932.43 25539.47 79223.38
## 140       52425.70 34129.07 70722.34 24443.42 80407.99
```

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

Akurasi Data Latih

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

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]        NA        NA 11760.000  5087.600  3096.176 -6530.810
```

```
mapedes.train1 <- sum(abs(sisaandes1[3:length(train.ts)]/train.ts[3:length(tr
ain.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                     3.137975e+09
## MSE                     2.551199e+07
## MAPE                    7.015581e+00
```

```
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  11760.000  -1262.800  -2694.616 -10692.312
```

```
mapedes.train2 <- sum(abs(sisaandes2[3:length(train.ts)]/train.ts[3:length(tr
ain.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                     1.593041e+09
## MSE                     1.295155e+07
## MAPE                    4.640332e+00
```

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 sangat baik berdasarkan nilai MAPE-nya.

Akurasi Data Uji

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

```
## Time Series:
## Start = 124
## End = 140
## Frequency = 1
##  [1]    853.1478  -8195.2869  -9073.7216 -13562.1563 -21470.5909 -18199.02
56
##  [7] -10587.4603  -7245.8950  -8944.3297 -19972.7643 -15591.1990 -23039.63
37
## [13] -15578.0684  -8946.5030 -18094.9377 -22833.3724 -15901.8071
```

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

```
selisihdes2<-ramalandes2$mean-testing$harga
selisihdes2
```

```
## Time Series:
## Start = 124
## End = 140
## Frequency = 1
##  [1]   1496.496  -7732.567  -8791.630 -13460.693 -21549.756 -18458.819
##  [7] -11027.882  -7866.945  -9746.008 -20955.071 -16754.134 -24383.197
## [13] -17102.260 -10651.323 -19980.387 -24899.450 -18148.513
```

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

selisihdesopt<-ramalandesopt$mean-testing$harga
selisihdesopt
```

```
## Time Series:
## Start = 124
## End = 140
## Frequency = 1
##  [1]   1227.251  -8488.471 -10034.193 -15189.915 -23765.636 -21161.358
##  [7] -14217.080 -11542.802 -13908.523 -25604.245 -21889.967 -30005.689
## [13] -23211.410 -17247.132 -27062.854 -32468.576 -26204.297
```

```
SSEtestingdesopt<-sum(selisihdesopt^2)
MSEtestingdesopt<-SSEtestingdesopt/length(testing$harga)
MAPEtestingdesopt<-sum(abs(selisihdesopt/testing$harga)*100)/length(testing$h
arga)

akurasitestingdes <-
  matrix(c(SSEtestingdes1,MSEtestingdes1,MAPEtestingdes1,SSEtestingdes2,MSEte
stingdes2,
           MAPEtestingdes2,SSEtestingdesopt,MSEtestingdesopt,MAPEtestingdesop
t),
         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  3.968139e+09 4.472984e+09 7.290990e+09
## MSE  2.334199e+08 2.631167e+08 4.288818e+08
## MAPE 1.897809e+01 2.014416e+01 2.568932e+01
```

Perhitungan akurasi menggunakan data uji menghasilkan nilai MAPE yang lebih dari 10% untuk semua skenario sehingga nilai akurasi ini dapat dikategorikan sebagai cukup baik.

Perbandingan SES dan DES

```
MSEfull <-
  matrix(c(MSEtesting1,MSEtesting2,MSEtestingopt,MSEtestingdes1,MSEtestingdes
2,
           MSEtestingdesopt),nrow=3,ncol=2)
row.names(MSEfull)<- c("ske 1", "ske 2", "ske opt")
colnames(MSEfull) <- c("ses","des")
MSEfull
```

```
##                ses       des
## ske 1   4520428764 233419943
```

```
## ske 2   3880728692 263116710
## ske opt 3785356172 428881773
```

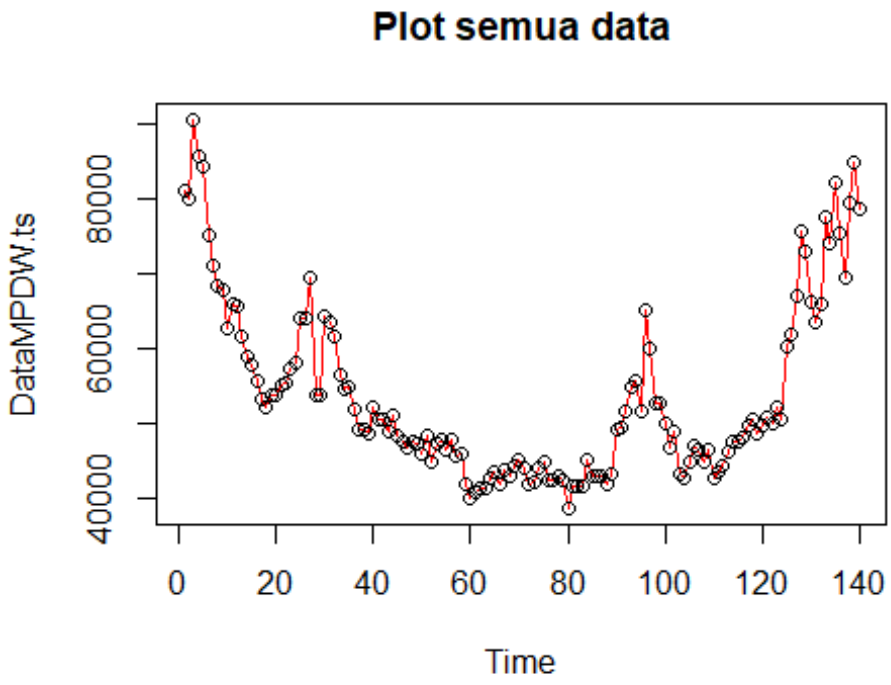
KESIMPULAN : Kedua metode dapat dibandingkan dengan menggunakan ukuran akurasi yang sama. Contoh di atas adalah perbandingan kedua metode dengan ukuran akurasi MSE. Hasilnya didapatkan metode DES lebih baik dibandingkan metode SES dilihat dari MSE yang lebih kecil nilainya.

Pemulusan Data Musiman

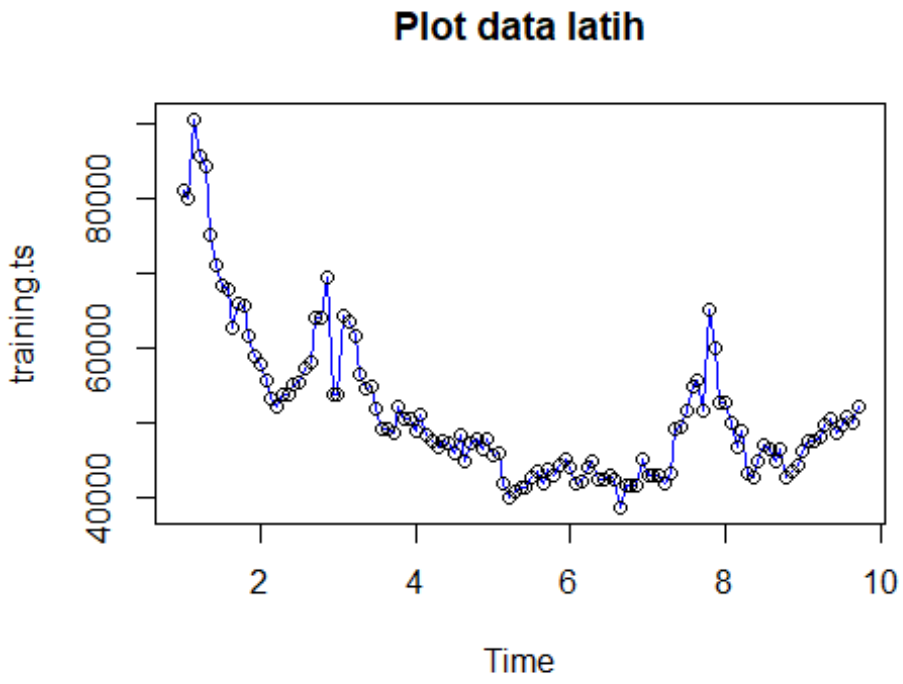
```
#membagi data menjadi training dan testing
training<-DataMPDW[1:123,2]
testing<-DataMPDW[124:140,2]
training.ts<-ts(training, frequency = 14)
testing.ts<-ts(testing, frequency = 14)
```

Kemudian akan dilakukan eskplorasi dengan plot data deret waktu sebagai berikut.

```
#Membuat plot time series
plot(DataMPDW.ts, col="red",main="Plot semua data")
points(DataMPDW.ts)
```




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




```
plot(testing.ts, col="green",main="Plot data uji")
points(testing.ts)
```

Metode Holt-Winter untuk peramalan data musiman menggunakan tiga persamaan pemulusan yang terdiri atas persamaan untuk level (L_t), trend (B_t), dan komponen seasonal / musiman (S_t) dengan parameter pemulusan berupa α, β, dan γ. Metode Holt-Winter musiman terbagi menjadi dua, yaitu metode aditif dan metode multiplikatif.

Pemulusan data musiman dengan metode Winter dilakukan menggunakan fungsi `HoltWinters()` dengan memasukkan argumen tambahan, yaitu `gamma()` dan `seasonal()`. Arguman `seasonal()` diinisialisasi menyesuaikan jenis musiman, aditif atau multiplikatif.

Winter Aditif

Perhitungan dengan model aditif dilakukan jika plot data asli menunjukkan fluktuasi musiman yang relatif stabil (konstan).

Pemulusan

```
#Pemulusan dengan winter aditif
winter1 <- HoltWinters(training.ts,alpha=0.2,beta=0.1,gamma=0.1,seasonal = "a
dditive")
winter1$fitted
```

```
## Time Series:
## Start = c(2, 1)
## End = c(9, 11)
## Frequency = 14
##              xhat    level       trend      season
## 2.000000 70495.26 70344.57 -1146.63242  1297.32398
## 2.071429 65058.86 66682.88 -1398.13759  -225.89031
## 2.142857 59752.70 63382.97 -1588.31469 -2041.96173
## 2.214286 55837.02 60482.12 -1719.56865 -2925.53316
## 2.285714 54950.55 58011.15 -1794.70902 -1265.89031
## 2.357143 52634.55 55950.33 -1821.31999 -1494.46173
## 2.428571 52341.06 54326.10 -1801.61093  -183.42602
## 2.500000 50081.11 53064.28 -1747.63219 -1235.53316
## 2.571429 50597.95 52356.42 -1643.65440  -114.81888
## 2.642857 47510.73 52035.18 -1511.41337 -3013.03316
## 2.714286 53972.71 52635.62 -1300.22800  2637.32398
## 2.785714 57169.42 53360.85 -1097.68228  4906.25255
## 2.857143 55227.41 53657.28  -958.27063  2528.39541
## 2.928571 56029.56 55569.53  -671.21875  1131.25255
## 3.000000 54030.31 54454.60  -715.59002   291.30329
## 3.071429 51988.57 53695.15  -719.97625  -986.59871
## 3.142857 52372.53 55415.46  -475.94767 -2566.97757
## 3.214286 53659.11 57141.00  -255.79827 -3226.09462
## 3.285714 57023.87 58491.38   -95.18045 -1372.33418
## 3.357143 56754.74 58277.43  -107.05779 -1415.62552
## 3.428571 57634.96 57751.42  -148.95268    32.48897
## 3.500000 56019.00 57043.48  -204.85183  -819.62203
## 3.571429 56150.54 56022.82  -286.43189   414.14524
## 3.642857 51737.15 54332.29  -426.84265 -2168.29166
## 3.714286 56363.53 53394.01  -477.98566  3447.50685
```

```
## 3.785714 56196.16 51365.32  -633.05633  5463.89915
## 3.857143 52879.05 49917.03  -714.57959  3676.60291
## 3.928571 48913.85 48722.64  -762.56067   953.76748
## 4.000000 47833.43 48289.31  -729.63762   273.75837
## 4.071429 47019.00 47740.99  -711.50624   -10.48438
## 4.142857 45509.61 47827.68  -631.68616 -1686.37998
## 4.214286 44590.17 47750.07  -576.27846 -2583.62333
## 4.285714 45811.23 47749.76  -518.68187 -1419.84355
## 4.357143 45322.52 47406.83  -501.10655 -1583.20508
## 4.428571 46690.36 47339.22  -457.75693  -191.10763
## 4.500000 45385.49 46979.39  -447.96405 -1145.94228
## 4.571429 46062.36 46646.33  -436.47377  -147.49779
## 4.642857 43906.00 46669.39  -390.52096 -2372.86372
## 4.714286 48894.65 46441.66  -374.24097  2827.22418
## 4.785714 50453.77 45724.49  -408.53392  5137.80613
## 4.857143 47799.48 44777.21  -462.40924  3484.67856
## 4.928571 44599.16 44006.90  -493.19876  1085.45969
## 5.000000 44062.17 44145.87  -429.98203   346.28388
## 5.071429 43952.82 44041.45  -397.42548   308.79594
## 5.142857 42183.83 44009.46  -360.88198 -1464.74918
## 5.214286 40836.82 43559.82  -369.75863 -2353.23695
## 5.285714 41231.66 42972.69  -391.49502 -1349.54227
## 5.357143 40638.73 42452.87  -404.32813 -1409.80662
## 5.428571 41615.75 42160.79  -393.10277  -151.93610
## 5.500000 40183.14 41684.54  -401.41784 -1099.98116
## 5.571429 41436.53 41754.49  -354.28063    36.31345
## 5.642857 39187.75 41808.91  -313.41115 -2307.74377
## 5.714286 44431.43 42003.95  -262.56619  2690.05238
## 5.785714 46235.80 41591.09  -277.59483  4922.30485
## 5.857143 43648.15 40632.34  -345.71089  3361.52048
## 5.928571 41405.05 40401.00  -334.27383  1338.32663
## 6.000000 41015.25 40799.71  -260.97483   476.51006
## 6.071429 41351.98 41101.69  -204.67980   454.96996
## 6.142857 39275.24 40972.61  -197.11938 -1500.25580
## 6.214286 38744.24 41326.45  -142.02415 -2440.18253
## 6.285714 40760.39 42201.57   -40.30894 -1400.87471
## 6.357143 41607.37 42935.19    37.08325 -1364.90517
## 6.428571 43000.74 43132.80    53.13595  -185.19636
## 6.500000 42171.27 43043.79    38.92121  -911.43233
## 6.571429 43488.34 43234.45    54.09571   199.79139
## 6.642857 40954.84 43030.88    28.32893 -2104.36395
## 6.714286 45186.21 42576.24   -19.96797  2629.93784
## 6.785714 46372.98 41817.03   -93.89217  4649.84060
## 6.857143 43962.26 40746.54  -191.55174  3407.26870
## 6.928571 41451.06 40060.54  -240.99692  1631.52263
## 7.000000 41047.80 40517.33  -171.21821   701.69017
## 7.071429 41052.19 40702.55  -135.57422   485.21164
## 7.142857 39562.45 40940.54   -98.21798 -1279.87484
## 7.214286 39456.04 41519.83   -30.46690 -2033.32169
## 7.285714 40863.46 41940.16    14.61225 -1091.30595
```

```
## 7.357143 41189.32 42428.08    61.94300 -1300.69439
## 7.428571 44037.06 44060.15   218.95651  -242.05531
## 7.500000 44824.98 45349.70   326.01541  -850.73432
## 7.571429 47599.96 47040.72   462.51580    96.72425
## 7.642857 47236.81 48929.24   605.11662 -2297.55155
## 7.714286 54304.82 51199.00   771.58047  2334.24105
## 7.785714 56408.50 51431.61   717.68408  4259.20233
## 7.857143 57966.60 53867.60   889.51406  3209.48797
## 7.928571 58004.61 55163.79   930.18206  1910.63748
## 8.000000 56679.41 55013.05   822.08982   844.26611
## 8.071429 56409.80 55033.26   741.90166   634.63661
## 8.142857 54080.44 54477.20   612.10569 -1008.87051
## 8.214286 52211.51 53601.22   463.29694 -1853.00513
## 8.285714 52877.50 53384.21   395.26670  -901.98294
## 8.357143 51386.26 51855.98   202.91672  -672.64034
## 8.428571 50508.42 50295.65    26.59156   186.18028
## 8.500000 48779.45 49172.55   -88.37682  -304.73276
## 8.571429 49251.65 48710.29  -125.76572   667.12755
## 8.642857 46204.10 48018.19  -182.39873 -1631.69618
## 8.714286 49432.35 47526.97  -213.28070  2118.65551
## 8.785714 51384.22 46711.22  -273.52769  4946.52226
## 8.857143 47610.40 44686.85  -448.61206  3372.15994
## 8.928571 44348.21 43402.16  -532.22007  1478.26854
## 9.000000 42804.43 42818.30  -537.38425   523.51347
## 9.071429 42608.21 42962.03  -469.27281   115.45273
## 9.142857 41480.17 43457.11  -372.83698 -1604.10550
## 9.214286 41895.28 44274.24  -253.84042 -2125.12610
## 9.285714 43425.02 45229.35  -132.94594 -1671.38283
## 9.357143 44924.21 46313.40   -11.24631 -1377.94100
## 9.428571 47213.09 47389.31    97.46948  -273.69322
## 9.500000 47452.68 47780.16   126.80777  -454.28837
## 9.571429 48955.58 48344.43   170.55415   440.59549
## 9.642857 47294.29 48845.87   203.64249 -1755.22403
## 9.714286 51721.68 49586.66   257.35671  1877.66755
```

```
xhat1 <- winter1$fitted[,2]
```

```
winter1.opt<- HoltWinters(training.ts, alpha= NULL,  beta = NULL, gamma = NUL
L, seasonal = "additive")
winter1.opt
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal compon
ent.
##
## Call:
## HoltWinters(x = training.ts, alpha = NULL, beta = NULL, gamma = NULL,
seasonal = "additive")
##
## Smoothing parameters:
##  alpha: 0.7633572
```

```
##  beta : 0.03265924
##  gamma: 0.3474856
##
## Coefficients:
##           [,1]
## a   50903.78413
## b      64.12201
## s1   4521.18184
## s2   3573.61693
## s3   1617.40199
## s4    697.60319
## s5    585.40494
## s6  -1206.47027
## s7  -1945.62054
## s8  -1814.80842
## s9  -1146.73723
## s10  -375.14136
## s11  -219.74497
## s12   281.58421
## s13 -1830.60179
## s14  1255.81889
```

```
winter1.opt$fitted
```

```
## Time Series:
## Start = c(2, 1)
## End = c(9, 11)
## Frequency = 14
##              xhat    level        trend      season
## 2.000000 70495.26 70344.57 -1146.632418  1297.32398
## 2.071429 57912.49 59598.52 -1460.141997  -225.89031
## 2.142857 52773.95 56334.96 -1519.040409 -2041.96173
## 2.214286 50699.31 55133.51 -1508.668107 -2925.53316
## 2.285714 51938.67 54678.80 -1474.246490 -1265.89031
## 2.357143 51561.22 54488.01 -1432.329690 -1494.46173
## 2.428571 53062.84 54627.27 -1381.003009  -183.42602
## 2.500000 52188.30 54755.54 -1331.711040 -1235.53316
## 2.571429 54414.45 55783.90 -1254.632913  -114.81888
## 2.642857 52465.30 56663.27 -1184.938061 -3013.03316
## 2.714286 61348.84 59756.72 -1045.209191  2637.32398
## 2.785714 64741.26 60811.63  -976.620846  4906.25255
## 2.857143 60912.82 59376.03  -991.610772  2528.39541
## 2.928571 65356.30 65000.58  -775.532125  1131.25255
## 3.000000 54611.76 55411.86 -1063.364107   263.26081
## 3.071429 52233.75 53737.23 -1083.327492  -420.15767
## 3.142857 58987.80 61780.80  -785.250142 -2007.75029
## 3.214286 60860.62 64348.37  -675.749564 -2811.99875
## 3.285714 62523.02 64305.73  -655.072528 -1127.63418
## 3.357143 56867.36 58999.50  -806.975697 -1325.16856
## 3.428571 55624.67 56507.52  -862.006667   -20.84404
```

```
## 3.500000 53183.66 55046.53  -881.569110  -981.30281
## 3.571429 52418.09 53215.61  -912.574371   115.05888
## 3.642857 46246.34 49793.04  -994.548657 -2552.15896
## 3.714286 52980.07 51037.93  -921.410471  2863.55210
## 3.785714 50607.05 46780.59 -1030.359233  4856.81058
## 3.857143 49153.62 46905.16  -992.640257  3241.09762
## 3.928571 46147.33 46925.02  -959.572640   181.88299
## 4.000000 48681.74 49333.89  -849.561928   197.41465
## 4.071429 48243.69 48528.80  -848.109565   563.00467
## 4.142857 47366.65 49792.37  -779.143726 -1646.57946
## 4.214286 46210.27 49710.44  -756.373198 -2743.79872
## 4.285714 47562.06 49915.69  -724.967260 -1628.66344
## 4.357143 46271.64 48525.03  -746.708339 -1506.67976
## 4.428571 47906.66 48708.36  -716.333874   -85.36775
## 4.500000 45619.31 47437.33  -734.450056 -1083.56890
## 4.571429 46081.67 46962.95  -725.956385  -155.32070
## 4.642857 44996.09 47976.17  -669.156088 -2310.92389
## 4.714286 48924.42 47096.26  -676.039200  2504.20135
## 4.785714 49350.30 45088.60  -719.528773  4981.22083
## 4.857143 45746.10 43155.11  -759.175897  3350.16607
## 4.928571 42586.60 42788.22  -746.364061   544.73642
## 5.000000 45575.83 45991.01  -617.387647   202.20505
## 5.071429 45636.71 45460.78  -614.541363   790.47789
## 5.142857 42773.17 44955.61  -610.969159 -1571.47437
## 5.214286 40279.03 43555.97  -636.726802 -2640.21106
## 5.285714 40165.11 42515.40  -649.915849 -1700.37305
## 5.357143 40144.01 42189.83  -639.323135 -1406.49425
## 5.428571 41598.48 42356.60  -612.996581  -145.12119
## 5.500000 39760.93 41439.42  -622.931049 -1055.55383
## 5.571429 42416.29 42937.91  -553.647098    32.02635
## 5.642857 40335.50 43196.25  -527.128029 -2333.62678
## 5.714286 45602.02 43733.63  -492.362185  2360.75767
## 5.785714 46084.25 41774.08  -540.279514  4850.45089
## 5.857143 41520.66 38749.64  -621.410055  3392.42395
## 5.928571 40604.83 40188.80  -554.113639   970.14530
## 6.000000 42812.00 43043.20  -442.793915   211.59308
## 6.071429 43762.35 43377.51  -417.414568   802.26025
## 6.142857 39284.17 41408.68  -468.082430 -1656.43200
## 6.214286 39953.31 43036.65  -399.626973 -2683.71311
## 6.285714 43627.91 45596.32  -302.978511 -1665.43461
## 6.357143 44460.64 46058.30  -277.995718 -1319.66016
## 6.428571 43707.92 44214.93  -329.119568  -177.88851
## 6.500000 41611.93 42803.43  -364.469298  -827.03136
## 6.571429 43233.01 43445.12  -331.608905   119.49542
## 6.642857 39748.64 42324.96  -357.362428 -2218.95699
## 6.714286 42860.19 41044.97  -387.494593  2202.70973
## 6.785714 43772.74 39611.54  -421.654288  4582.85426
## 6.857143 40583.17 37447.34  -478.564401  3614.39083
## 6.928571 38542.37 37661.01  -455.956403  1337.31630
## 7.000000 42087.57 42088.73  -296.459300   295.30296
```

```
## 7.071429 42716.20 42359.01  -277.950147   635.14010
## 7.142857 40533.12 42236.63  -272.869212 -1430.64220
## 7.214286 41231.16 43808.70  -212.614785 -2364.93301
## 7.285714 42177.91 43961.62  -200.676913 -1583.03271
## 7.357143 42901.33 44564.06  -174.447574 -1488.28432
## 7.428571 48759.72 49075.61   -21.406463  -294.48416
## 7.500000 48810.99 49535.33    -5.693206  -718.64641
## 7.571429 51796.45 51696.82    65.085113    34.55138
## 7.642857 51821.12 54001.25   138.220409 -2318.34331
## 7.714286 59315.04 56993.56   231.433111  2090.03923
## 7.785714 55777.79 51343.30    39.341399  4395.14500
## 7.857143 62380.70 58422.48   269.257268  3688.95984
## 7.928571 58947.71 56874.42   209.904779  1863.39298
## 8.000000 52646.75 52238.75    51.652039   356.35255
## 8.071429 53012.28 52308.15    52.231613   651.89878
## 8.142857 48743.10 49999.87   -24.860901 -1231.90223
## 8.214286 45966.74 48369.59   -77.292664 -2325.55777
## 8.285714 48959.79 50462.72    -6.408199 -1496.51927
## 8.357143 44973.32 46105.33  -148.507820  -983.50174
## 8.428571 43671.15 44122.23  -208.424305  -242.65639
## 8.500000 44078.51 44744.99  -181.278563  -485.19500
## 8.571429 46890.23 46725.14  -110.687741   275.77692
## 8.642857 44122.19 46255.50  -122.410974 -2010.89553
## 8.714286 47891.08 46543.63  -109.003110  1456.45424
## 8.785714 50319.48 45311.67  -145.678083  5153.48716
## 8.857143 42451.98 39296.17  -337.381821  3493.19478
## 8.928571 40733.79 39705.36  -312.999253  1341.42064
## 9.000000 42083.29 41954.36  -229.326610   358.26419
## 9.071429 45146.36 44875.18  -126.445055   397.62097
## 9.142857 45017.62 46491.97   -69.512292 -1404.84034
## 9.214286 46162.84 48263.97    -9.370012 -2091.75626
## 9.285714 47680.93 49611.20    34.935739 -1965.21303
## 9.357143 49941.78 51042.38    80.535745 -1181.12706
## 9.428571 51380.00 51442.16    90.962130  -153.12034
## 9.500000 49243.34 49472.06    23.649300  -252.36202
## 9.571429 50069.14 49798.50    33.538210   237.10964
## 9.642857 48325.25 50244.90    47.022112 -1966.67176
## 9.714286 52978.85 51555.09    88.276107  1335.48747

xhat1.opt <- winter1.opt$fitted[,2]
```

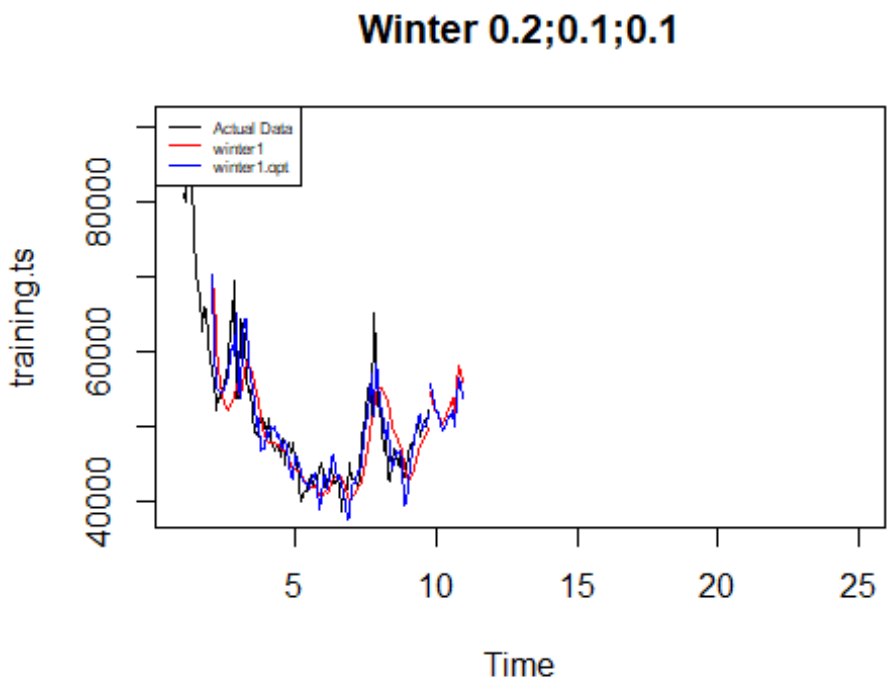
Peramalan

```
#Forecast
forecast1 <- predict(winter1, n.ahead = 17)
forecast1.opt <- predict(winter1.opt, n.ahead = 17)
```

Plot Deret Waktu

```
#Plot time series
plot(training.ts,main="Winter 0.2;0.1;0.1",type="l",col="black",
     xlim=c(1,25),pch=12)
```

```
lines(xhat1,type="l",col="red")
lines(xhat1.opt,type="l",col="blue")
lines(forecast1,type="l",col="red")
lines(forecast1.opt,type="l",col="blue")
legend("topleft",c("Actual Data",expression(paste(winter1)),
                   expression(paste(winter1.opt))),cex=0.5,
       col=c("black","red","blue"),lty=1)
```



Akurasi Data Latih

```
#Akurasi data training
SSE1<-winter1$SSE
MSE1<-winter1$SSE/length(training.ts)
RMSE1<-sqrt(MSE1)
fitted_values <- winter1$fitted[,1]
abs_error <- abs(training.ts - fitted_values)
MAPE1 <- mean(abs_error / training.ts * 100)
akurasi1 <- matrix(c(SSE1,MSE1,RMSE1, MAPE1))
row.names(akurasi1)<- c("SSE", "MSE", "RMSE", "MAPE")
colnames(akurasi1) <- c("Akurasi")
akurasi1
```

```
##           Akurasi
## SSE  2.640113e+09
## MSE  2.146434e+07
## RMSE 4.632962e+03
## MAPE 7.586912e+00
```

```
SSE1.opt<-winter1.opt$SSE
MSE1.opt<-winter1.opt$SSE/length(training.ts)
RMSE1.opt<-sqrt(MSE1.opt)
fitted_values <- winter1.opt$fitted[,1]
abs_error <- abs(training.ts - fitted_values)
MAPE1.opt <- mean(abs_error / training.ts * 100)

akurasi1.opt <- matrix(c(SSE1.opt,MSE1.opt,RMSE1.opt, MAPE1.opt))
row.names(akurasi1.opt)<- c("SSE1.opt", "MSE1.opt", "RMSE1.opt", "MAPE1.opt")
colnames(akurasi1.opt) <- c("Akurasi")
akurasi1.opt
```

```
##                Akurasi
## SSE1.opt  1.362301e+09
## MSE1.opt  1.107561e+07
## RMSE1.opt 3.328005e+03
## MAPE1.opt 4.992731e+00
```

```
akurasi1.train = data.frame(Model_Winter = c("Winter 1","Winter1 optimal"),
                            Nilai_SSE=c(SSE1,SSE1.opt),
                            Nilai_MSE=c(MSE1,MSE1.opt),Nilai_RMSE=c(RMSE1,RMS
E1.opt), Nilai_MAPE=c(MAPE1,MAPE1.opt))
akurasi1.train
```

```
##      Model_Winter  Nilai_SSE Nilai_MSE Nilai_RMSE Nilai_MAPE
## 1        Winter 1 2640113408  21464337   4632.962   7.586912
## 2 Winter1 optimal 1362300629  11075615   3328.005   4.992731
```

Perhitungan akurasi menggunakan data latih menghasilkan nilai MAPE yang kurang dari 10% sehingga nilai akurasi ini dapat dikategorikan sebagai sangat baik.Selain itu, hasil akurasi skenario 2 (winter1 optimal) lebih baik dari skenario 1 (winter1) karena nilai SSE, MSE dan RMSE nya lebih kecil.

Akurasi Data Uji

```
#Akurasi Data Testing

forecast11<-data.frame(forecast1)
testing.ts1<-data.frame(testing_ma$harga)
selisih1<-forecast11-testing.ts1
SSEtesting1<-sum(selisih1^2)
MSEtesting1<-SSEtesting1/length(testing.ts1)
MAPEtesting1<-sum(abs(selisih1/testing.ts1$testing_ma.harga)*100)/length(test
ing.ts1$testing_ma.harga)



forecast1.opt<-data.frame(forecast1.opt)
selisih1.opt<-forecast1.opt-testing.ts1
SSEtesting1.opt<-sum(selisih1.opt^2)
MSEtesting1.opt<-SSEtesting1.opt/length(testing.ts1)
MAPEtesting1.opt<-sum(abs(selisih1.opt/testing.ts1$testing_ma.harga)*100)/len
```

```
gth(testing.ts1$testing_ma.harga)

akurasi1.test = data.frame(Model_Winter = c("Winter 1","Winter1 optimal"),
                           Nilai_SSE=c(SSEtesting1,SSEtesting1.opt),
                           Nilai_MSE=c(MSEtesting1,MSEtesting1.opt), Nilai_M
APE=c(MAPEtesting1, MAPEtesting1.opt))
akurasi1.test
```

```
##      Model_Winter  Nilai_SSE  Nilai_MSE Nilai_MAPE
## 1        Winter 1 6537934749 6537934749   24.90338
## 2 Winter1 optimal 7266406110 7266406110   26.19163
```

Perhitungan akurasi menggunakan data uji menghasilkan nilai MAPE lebih dari 10% sehingga nilai akurasi ini dapat dikategorikan sebagai cukup baik.Selain itu, hasil akurasi skenario 1 (winter1) lebih baik dari skenario 1 (winter1 optimal) karena nilai SSE dan MSEnya lebih kecil.

Winter Multiplikatif

Model multiplikatif digunakan cocok digunakan jika plot data asli menunjukkan fluktuasi musiman yang bervariasi.

Pemulusan

```
#Pemulusan dengan winter multiplikatif
winter2 <- HoltWinters(training.ts,alpha=0.2,beta=0.1,gamma=0.3,seasonal = "m
ultiplicative")
winter2$fitted
```

```
## Time Series:
## Start = c(2, 1)
## End = c(9, 11)
## Frequency = 14
##              xhat    level        trend    season
## 2.000000 70694.45 70344.57 -1146.632418 1.0216266
## 2.071429 64915.34 66697.13 -1396.713036 0.9941030
## 2.142857 59411.31 63416.24 -1585.130931 0.9608644
## 2.214286 55558.60 60536.17 -1714.624925 0.9445282
## 2.285714 54860.63 58084.96 -1788.282905 0.9744914
## 2.357143 52634.74 56042.06 -1813.744924 0.9706136
## 2.428571 52344.78 54431.33 -1793.443140 0.9944317
## 2.500000 50839.18 53179.95 -1739.236956 0.9883062
## 2.571429 50874.91 52339.39 -1649.369597 1.0036475
## 2.642857 48320.45 51952.43 -1523.128191 0.9581821
## 2.714286 53431.61 52464.31 -1319.627304 1.0447149
## 2.785714 56353.13 53187.04 -1115.391896 1.0822228
## 2.857143 54809.04 53510.70  -971.486695 1.0432026
## 2.928571 55725.29 55371.06  -688.301825 1.0190651
## 3.000000 52277.00 54307.06  -725.871333 0.9756596
## 3.071429 51001.86 53895.65  -694.426003 0.9586597
## 3.142857 51990.25 55952.59  -419.289056 0.9361996
```

```
## 3.214286 53754.14 57966.49  -175.970246 0.9301550
## 3.285714 57657.91 59496.87    -5.335007 0.9691784
## 3.357143 57724.79 59238.14   -30.674182 0.9749579
## 3.428571 58870.94 58578.77   -93.544349 1.0065951
## 3.500000 58009.27 57684.32  -173.634955 1.0086694
## 3.571429 57856.86 56307.26  -293.976994 1.0329131
## 3.642857 54010.40 54323.53  -462.952618 1.0027818
## 3.714286 57197.70 52897.17  -559.292685 1.0928547
## 3.785714 55913.04 50766.27  -716.453541 1.1171476
## 3.857143 53796.12 49370.76  -784.359309 1.1072257
## 3.928571 47643.11 47987.41  -844.258857 1.0106052
## 4.000000 46112.08 47720.40  -786.533297 0.9824905
## 4.071429 47447.48 47468.82  -733.038321 1.0152282
## 4.142857 45996.26 47437.60  -662.856721 0.9833568
## 4.214286 44858.92 47239.22  -616.408908 0.9621669
## 4.285714 44935.19 47165.56  -562.134017 0.9642036
## 4.357143 44695.54 46967.42  -525.734850 0.9624013
## 4.428571 46081.01 47022.41  -467.662103 0.9898241
## 4.500000 45534.46 46776.81  -445.456371 0.9828002
## 4.571429 45721.73 46417.95  -436.796635 0.9943581
## 4.642857 45245.44 46511.80  -383.731818 0.9808657
## 4.714286 48010.74 46020.93  -394.445597 1.0522559
## 4.785714 49506.02 45468.59  -410.235216 1.0987090
## 4.857143 48313.75 44740.52  -442.018274 1.0906408
## 4.928571 44540.22 43921.89  -479.679701 1.0252751
## 5.000000 43469.08 44070.29  -416.871490 0.9957772
## 5.071429 45181.24 44099.49  -372.264672 1.0332519
## 5.142857 43263.27 43843.12  -360.674807 0.9949594
## 5.214286 41734.72 43176.25  -391.294534 0.9754532
## 5.285714 40820.64 42378.02  -431.987871 0.9731705
## 5.357143 40494.35 41898.63  -436.727908 0.9766641
## 5.428571 40997.27 41606.41  -422.277788 0.9954627
## 5.500000 40194.57 41224.86  -418.204609 0.9850004
## 5.571429 41238.44 41282.88  -370.581769 1.0079715
## 5.642857 40133.43 41357.07  -326.105033 0.9781255
## 5.714286 43029.73 41357.42  -293.459631 1.0478710
## 5.785714 44561.78 41188.07  -281.048387 1.0893429
## 5.857143 43474.91 40589.07  -312.843258 1.0794186
## 5.928571 41832.56 40414.28  -299.037910 1.0428096
## 6.000000 40817.69 40736.15  -236.947243 1.0078639
## 6.071429 42414.58 41096.97  -177.171117 1.0365296
## 6.142857 40048.94 40787.71  -190.380197 0.9864921
## 6.214286 39386.90 40998.96  -150.216525 0.9642131
## 6.285714 40538.06 41770.35   -58.056334 0.9718494
## 6.357143 41760.20 42554.38    26.152932 0.9807345
## 6.428571 42608.93 42713.05    39.404191 0.9966430
## 6.500000 42663.17 42688.45    33.004037 0.9986357
## 6.571429 43711.45 42774.89    38.347915 1.0209796
## 6.642857 41989.68 42517.16     8.740136 0.9873905
## 6.714286 43923.66 41827.16   -61.134463 1.0516600
```

```
## 6.785714 44454.50 41303.20  -107.416707 1.0791031
## 6.857143 43878.30 40646.35  -162.360452 1.0838433
## 6.928571 42302.08 40043.28  -206.431362 1.0618832
## 7.000000 41199.66 40333.69  -156.747496 1.0254554
## 7.071429 41682.04 40494.91  -124.950059 1.0325014
## 7.142857 40431.37 40609.76  -100.970328 0.9980889
## 7.214286 40542.78 41013.48   -50.501317 0.9897419
## 7.285714 40963.07 41198.84   -26.914867 0.9949272
## 7.357143 40996.00 41627.63    18.654841 0.9843857
## 7.428571 43238.85 43280.60   182.086715 0.9948499
## 7.500000 45011.01 44699.28   305.746600 1.0001328
## 7.571429 47353.36 46332.65   438.508814 1.0124478
## 7.642857 47230.89 48228.35   584.227733 0.9675966
## 7.714286 53214.94 50534.19   756.388583 1.0375188
## 7.785714 54891.72 50981.20   725.450595 1.0615989
## 7.857143 58318.08 53611.00   915.885608 1.0695290
## 7.928571 60116.81 54841.40   947.337194 1.0775798
## 8.000000 57139.99 54393.61   807.824337 1.0351179
## 8.071429 57251.55 54337.76   721.457588 1.0398176
## 8.142857 54925.12 53649.06   580.441520 1.0128272
## 8.214286 52826.96 52593.46   416.837791 0.9965414
## 8.285714 52960.51 52204.12   336.219708 1.0079970
## 8.357143 52230.96 50615.63   143.748777 1.0289913
## 8.428571 50199.03 48881.63   -44.026610 1.0278768
## 8.500000 49273.72 47779.30  -149.857085 1.0345223
## 8.571429 49286.13 47172.47  -195.553884 1.0491564
## 8.642857 46510.72 46430.55  -250.190760 1.0071538
## 8.714286 46890.01 45812.84  -286.942256 1.0299634
## 8.785714 49961.65 45434.63  -296.068984 1.1068506
## 8.857143 46721.11 43813.79  -428.546588 1.0768895
## 8.928571 44162.34 42774.02  -489.669111 1.0444135
## 9.000000 42421.79 42270.49  -491.054393 1.0153748
## 9.071429 42404.77 42525.61  -416.437375 1.0070198
## 9.142857 41721.66 43107.21  -316.633414 0.9750197
## 9.214286 42802.44 43961.50  -199.541574 0.9780741
## 9.285714 43018.77 44812.50   -94.486855 0.9620009
## 9.357143 45257.67 46067.54    40.465805 0.9815577
## 9.428571 47318.37 47147.65   144.429701 1.0005560
## 9.500000 48809.79 47564.25   171.647181 1.0224964
## 9.571429 49737.52 47898.29   187.886156 1.0343413
## 9.642857 48336.47 48254.88   204.756475 0.9974584
## 9.714286 50374.18 48789.18   237.710865 1.0274806
```

```
xhat2 <- winter2$fitted[,2]
```

```
winter2.opt<- HoltWinters(training.ts, alpha= NULL,  beta = NULL, gamma = NUL
L, seasonal = "multiplicative")
winter2.opt$fitted
```

```
## Time Series:
## Start = c(2, 1)
## End = c(9, 11)
## Frequency = 14
##              xhat    level         trend    season
## 2.000000 70694.45 70344.57 -1146.6324176 1.0216266
## 2.071429 57839.08 59647.75 -1465.5779698 0.9941030
## 2.142857 52750.66 56423.48 -1524.3127618 0.9608644
## 2.214286 50754.92 55248.39 -1512.6498157 0.9445282
## 2.285714 51969.98 54807.23 -1476.8654659 0.9744914
## 2.357143 51626.85 54623.59 -1433.6760399 0.9706136
## 2.428571 53079.78 54758.30 -1381.2967784 0.9944317
## 2.500000 52925.30 54882.54 -1331.0167713 0.9883062
## 2.571429 54298.33 55371.24 -1270.2438513 1.0036475
## 2.642857 52815.49 56316.76 -1196.2446253 0.9581821
## 2.714286 60857.28 59308.89 -1056.3661883 1.0447149
## 2.785714 64550.28 60623.21  -977.1929493 1.0822228
## 2.857143 60891.32 59356.46  -986.8630192 1.0432026
## 2.928571 65175.83 64730.91  -774.4154443 1.0190651
## 3.000000 54907.83 55438.79 -1058.8796874 1.0097080
## 3.071429 52035.82 53550.24 -1086.5880605 0.9918453
## 3.142857 58686.76 61822.95  -774.0174918 0.9613070
## 3.214286 60657.24 64777.76  -649.4865852 0.9458737
## 3.285714 62807.48 64962.20  -621.6361133 0.9761724
## 3.357143 56960.16 59350.75  -788.2798772 0.9726393
## 3.428571 55707.47 56756.26  -848.6014285 0.9964194
## 3.500000 53864.79 55242.73  -870.8079772 0.9906728
## 3.571429 52306.70 52887.99  -920.3665476 1.0065247
## 3.642857 46765.11 49557.09 -1000.8706723 0.9631126
## 3.714286 51896.74 50471.28  -936.9137512 1.0476917
## 3.785714 49895.89 47138.33 -1016.9337849 1.0818382
## 3.857143 49094.89 47691.60  -964.4939505 1.0506725
## 3.928571 47161.48 47733.99  -930.8672478 1.0076566
## 4.000000 48950.09 49379.08  -844.8383828 1.0085681
## 4.071429 47657.41 48375.15  -850.1517837 1.0027861
## 4.142857 47604.35 50078.48  -764.8733390 0.9653390
## 4.214286 46486.93 49848.18  -747.0204844 0.9467584
## 4.285714 47707.93 49894.22  -720.5348670 0.9701924
## 4.357143 46214.57 48372.33  -747.2974808 0.9703839
## 4.428571 47701.69 48628.90  -713.7715257 0.9955455
## 4.500000 46256.60 47514.90  -727.1379738 0.9886475
## 4.571429 45959.35 46558.62  -734.7903327 1.0029574
## 4.642857 45370.41 47651.97  -673.7365560 0.9657753
## 4.714286 47778.32 46463.86  -690.9148718 1.0438113
## 4.785714 48397.85 45335.15  -705.5358443 1.0844335
## 4.857143 45732.24 44180.38  -720.5389227 1.0522874
## 4.928571 43630.64 43842.89  -707.7460092 1.0114869
## 5.000000 46029.70 46253.21  -603.6127619 1.0083264
## 5.071429 45071.68 45392.29  -612.2059556 1.0065118
## 5.142857 43206.90 45317.57  -594.2555468 0.9660933
```

```
## 5.214286 40692.02 43563.63  -632.9855040 0.9478549
## 5.285714 40227.22 42171.58  -658.3358669 0.9690213
## 5.357143 39991.75 41799.18  -648.7863334 0.9718435
## 5.428571 41272.75 42099.96  -617.0739875 0.9949344
## 5.500000 40330.36 41427.04  -618.9390877 0.9882930
## 5.571429 42195.53 42515.75  -561.9091148 1.0057610
## 5.642857 40915.76 42929.25  -529.3333620 0.9649963
## 5.714286 44368.89 43044.36  -507.8109394 1.0430769
## 5.785714 44978.74 42032.13  -524.6571544 1.0836300
## 5.857143 41505.21 39992.99  -575.2360741 1.0529573
## 5.928571 41550.12 41386.94  -509.4714489 1.0164553
## 6.000000 43442.11 43522.32  -421.1418098 1.0079100
## 6.071429 43301.08 43395.11  -411.3253236 1.0073817
## 6.142857 39862.31 41792.64  -451.1058625 0.9642194
## 6.214286 40387.02 43058.59  -393.7618165 0.9466118
## 6.285714 43765.15 45442.77  -300.9873229 0.9695043
## 6.357143 44335.23 45823.10  -278.2334141 0.9734407
## 6.428571 43479.97 44034.32  -328.6808835 0.9948367
## 6.500000 42058.83 42792.06  -359.1915948 0.9911853
## 6.571429 43084.98 43104.15  -336.7727568 1.0074262
## 6.642857 40320.24 42096.45  -359.1798456 0.9660490
## 6.714286 41606.97 40329.79  -406.1851524 1.0421647
## 6.785714 42608.37 39837.88  -409.0480347 1.0806400
## 6.857143 40365.53 38638.39  -435.4460540 1.0566078
## 6.928571 39416.53 39015.77  -408.3004519 1.0209561
## 7.000000 42826.28 42739.53  -270.3028886 1.0084074
## 7.071429 42425.06 42472.04  -270.2088080 1.0052896
## 7.142857 40924.53 42577.86  -257.6506663 0.9670210
## 7.214286 41566.15 43919.96  -204.2242600 0.9508282
## 7.285714 42346.19 43831.28  -200.3652670 0.9705547
## 7.357143 42869.29 44326.42  -177.1375992 0.9710076
## 7.428571 48659.21 49003.00   -15.0390596 0.9932891
## 7.500000 49172.55 49549.89     3.7275320 0.9923101
## 7.571429 51850.26 51460.48    67.4106371 1.0062562
## 7.642857 51893.38 53713.67   140.4087918 0.9635924
## 7.714286 59391.79 56760.33   237.4684672 1.0420013
## 7.785714 55398.30 51293.89    46.9761233 1.0790292
## 7.857143 61811.40 58137.23   273.9529832 1.0582117
## 7.928571 58947.95 57103.80   230.2904860 1.0281483
## 8.000000 53134.53 52618.47    72.8037513 1.0084123
## 8.071429 52711.57 52339.44    61.0537028 1.0059365
## 8.142857 48742.36 50280.96    -9.7318434 0.9695875
## 8.214286 46171.68 48615.15   -65.0396511 0.9510108
## 8.285714 49238.80 50668.98     5.7236282 0.9716643
## 8.357143 44816.48 45975.12  -151.2273544 0.9780155
## 8.428571 43601.38 44069.54  -209.8172370 0.9941099
## 8.500000 44346.46 44749.88  -180.0887693 0.9949893
## 8.571429 46852.17 46537.60  -114.3702322 1.0092398
## 8.642857 44462.46 46096.17  -125.2928852 0.9671874
## 8.714286 47550.71 46126.88  -120.0830816 1.0335585
```

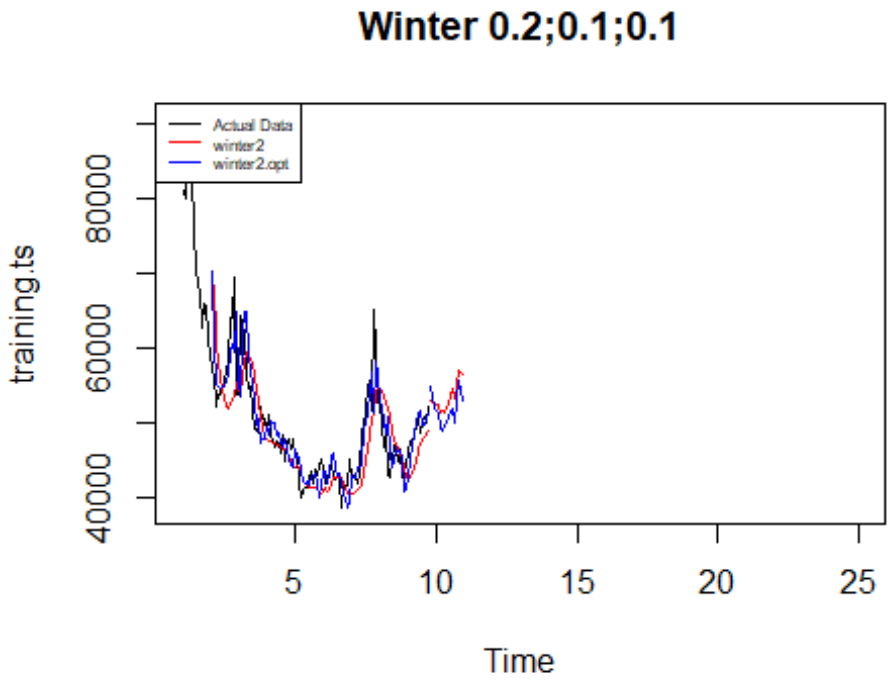
```
## 8.785714 48995.21 45171.23  -147.9881198 1.0882204
## 8.857143 42531.08 40555.82  -297.1859317 1.0564464
## 8.928571 41503.97 40908.51  -275.4820270 1.0214344
## 9.000000 42691.19 42566.71  -210.9033604 1.0079184
## 9.071429 45028.21 45022.24  -121.8527848 1.0028468
## 9.142857 45137.20 46729.59   -60.7632485 0.9671809
## 9.214286 46244.65 48479.42    -0.2952989 0.9539086
## 9.285714 48107.15 49836.54    45.0380860 0.9644272
## 9.357143 49806.95 50992.55    82.1410365 0.9751786
## 9.428571 51374.86 51507.85    96.6070009 0.9955508
## 9.500000 49468.16 49537.01    27.5609942 0.9980548
## 9.571429 50161.56 49696.07    31.9526906 1.0087181
## 9.642857 48478.54 50067.57    43.2924043 0.9674258
## 9.714286 53031.76 51296.24    82.8803348 1.0321654

xhat2.opt <- winter2.opt$fitted[,2]
```

Peramalan

```
#Forecast
forecast2 <- predict(winter2, n.ahead = 17)
forecast2.opt <- predict(winter2.opt, n.ahead = 17)
```

Plot Deret Waktu

```
#Plot time series
plot(training.ts,main="Winter 0.2;0.1;0.1",type="l",col="black",
     xlim=c(1,25),pch=12)
lines(xhat2,type="l",col="red")
lines(xhat2.opt,type="l",col="blue")
lines(forecast2,type="l",col="red")
lines(forecast2.opt,type="l",col="blue")
legend("topleft",c("Actual Data",expression(paste(winter2)),
                   expression(paste(winter2.opt))),cex=0.5,
       col=c("black","red","blue"),lty=1)
```



Akurasi Data Latih

```
#Akurasi data training
SSE2<-winter2$SSE
MSE2<-winter2$SSE/length(training.ts)
RMSE2<-sqrt(MSE2)
fitted_values <- winter2$fitted[,1]
abs_error <- abs(training.ts - fitted_values)
MAPE2 <- mean(abs_error / training.ts * 100)

akurasi2 <- matrix(c(SSE2,MSE2,RMSE2, MAPE2))
row.names(akurasi2)<- c("SSE2", "MSE2", "RMSE2", "MAPE2")
colnames(akurasi2) <- c("Akurasi lamda=0.2")
akurasi2

##       Akurasi lamda=0.2
## SSE2       2.788558e+09
## MSE2       2.267120e+07
## RMSE2      4.761429e+03
## MAPE2      7.626018e+00

SSE2.opt<-winter2.opt$SSE
MSE2.opt<-winter2.opt$SSE/length(training.ts)
RMSE2.opt<-sqrt(MSE2.opt)
fitted_values <- winter2.opt$fitted[,1]
abs_error <- abs(training.ts - fitted_values)
MAPE2.opt <- mean(abs_error / training.ts * 100)
```

```r
akurasi2.opt <- matrix(c(SSE2.opt,MSE2.opt,RMSE2.opt,MAPE2.opt))
row.names(akurasi2.opt)<- c("SSE2.opt", "MSE2.opt", "RMSE2.opt", "MAPE2.opt")
colnames(akurasi2.opt) <- c("Akurasi")
akurasi2.opt
```

```
##                Akurasi
## SSE2.opt  1.270486e+09
## MSE2.opt  1.032915e+07
## RMSE2.opt 3.213900e+03
## MAPE2.opt 4.619999e+00
```

```r
akurasi2.train = data.frame(Model_Winter = c("Winter 2","winter2 optimal"),
                            Nilai_SSE=c(SSE2,SSE2.opt),
                            Nilai_MSE=c(MSE2,MSE2.opt),Nilai_RMSE=c(RMSE2,RMS
E2.opt))
akurasi2.train
```

```
##      Model_Winter  Nilai_SSE Nilai_MSE Nilai_RMSE
## 1        Winter 2 2788557948  22671203   4761.429
## 2 winter2 optimal 1270486004  10329155   3213.900
```

```r
SSE2.opt<-winter2.opt$SSE
MSE2.opt<-winter2.opt$SSE/length(training.ts)
RMSE2.opt<-sqrt(MSE2.opt)
fitted_values <- winter2.opt$fitted[,1]
abs_error <- abs(training.ts - fitted_values)
MAPE2.opt <- mean(abs_error / training.ts * 100)

akurasi2.opt <- matrix(c(SSE2.opt,MSE2.opt,RMSE2.opt,MAPE2.opt))
row.names(akurasi2.opt)<- c("SSE2.opt", "MSE2.opt", "RMSE2.opt", "MAPE2.opt")
colnames(akurasi2.opt) <- c("Akurasi")
akurasi2.opt
```

```
##                Akurasi
## SSE2.opt  1.270486e+09
## MSE2.opt  1.032915e+07
## RMSE2.opt 3.213900e+03
## MAPE2.opt 4.619999e+00
```

```r
akurasi2.train = data.frame(Model_Winter = c("Winter 2","winter2 optimal"),
                            Nilai_SSE=c(SSE2,SSE2.opt),
                            Nilai_MSE=c(MSE2,MSE2.opt),Nilai_RMSE=c(RMSE2,RMS
E2.opt),Nilai_MAPE=c(MAPE2,MAPE2.opt))
akurasi2.train
```

```
##      Model_Winter  Nilai_SSE Nilai_MSE Nilai_RMSE Nilai_MAPE
## 1        Winter 2 2788557948  22671203   4761.429   7.626018
## 2 winter2 optimal 1270486004  10329155   3213.900   4.619999
```

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

akurasi skenario 2 (winter2 optimal) lebih baik dari skenario 1 (winter2) karena nilai SSE, MSE dan RMSE nya lebih kecil.

Akurasi Data Uji

```
#Akurasi Data Testing
forecast2<-data.frame(forecast2)
testing.ts<-data.frame(testing_ma$harga)
selisih2<-forecast2-testing.ts
SSEtesting2<-sum(selisih2^2)
MSEtesting2<-SSEtesting2/length(testing.ts)
RMSEtesting2 <- sqrt(MSEtesting2)
MAPEtesting2<-sum(abs(selisih2/testing.ts$testing_ma.harga)*100)/length(testi
ng.ts$testing_ma.harga)
akurasi2 <- matrix(c(SSEtesting2, MSEtesting2, RMSEtesting2, MAPEtesting2))
row.names(akurasi2) <- c("SSE2", "MSE2", "RMSE2","MAPE2" )
colnames(akurasi2) <- c("Akurasi lambda=0.2")
akurasi2
```

```
##       Akurasi lambda=0.2
## SSE2        6.255243e+09
## MSE2        6.255243e+09
## RMSE2       7.909010e+04
## MAPE2       2.419394e+01
```

```
forecast2.opt<-data.frame(forecast2.opt)
selisih2.opt<-forecast2.opt-testing.ts
SSEtesting2.opt<-sum(selisih2.opt^2)
MSEtesting2.opt<-SSEtesting2.opt/length(testing.ts)
RMSEtesting2.opt <- sqrt(MSEtesting2.opt)
MAPEtesting2.opt<-sum(abs(selisih2.opt/testing.ts$testing_ma.harga)*100)/leng
th(testing.ts$testing_ma.harga)
akurasi2.opt <- matrix(c(SSEtesting2.opt, MSEtesting2.opt, RMSEtesting2.opt,M
APEtesting2.opt))
row.names(akurasi2.opt) <- c("SSE2", "MSE2", "RMSE2", "MAPE2")
colnames(akurasi2.opt) <- c("Akurasi Optimal")
akurasi2.opt
```

```
##       Akurasi Optimal
## SSE2     7.577437e+09
## MSE2     7.577437e+09
## RMSE2    8.704847e+04
## MAPE2    2.675819e+01
```

```
akurasi2.test = data.frame(Model_Winter = c("Winter2","Winter2 optimal"),
                           Nilai_SSE=c(SSEtesting2,SSEtesting2.opt),
                           Nilai_MSE=c(MSEtesting2,MSEtesting2.opt), Nilai_R
MSE=c(RMSEtesting2,RMSEtesting2.opt), Nilai_MAPE=c(MAPEtesting2, MAPEtesting2
.opt))
akurasi2.test
```

```
##      Model_Winter  Nilai_SSE  Nilai_MSE Nilai_RMSE Nilai_MAPE
## 1         Winter2 6255243448 6255243448   79090.10   24.19394
## 2 Winter2 optimal 7577436717 7577436717   87048.47   26.75819
```

Perhitungan akurasi menggunakan data uji menghasilkan nilai MAPE lebih dari 10% sehingga nilai akurasi ini dapat dikategorikan sebagai cukup baik.Selain itu, hasil akurasi skenario 1 (winter2) lebih baik dari skenario 1 (winter2 optimal) karena nilai SSE dan MSEnya lebih kecil.

KESIMPULAN : Metode winter aditif lebih baik digunakan untuk skenario kedua (winter optimal) karena nilai SSE, MSE, RMSE, dan MAPEnya lebih kecil dibandingkan metode multiplikatif. Sedangkan sebaliknya metode multiplikatif lebih baik digunkan untuk skenario pertama (winter) karena nilai SSE, MSE, RMSE, dan MAPEnya lebih kecil dibandingkan metode aditif.

Kesimpulan Keseluruhan Metode

Berdasarkan keenam metode yaitu SMA, DMA, SES, DES, Winter Aditif dan Winter Multiplikatif, metode yang paling baik untuk digunakan adalah metode DES Skenario pertama karena secara keseluruhan nilai MAPEnya lebih rendah dibandingkan kelima metode lainnya. Adapun MAPE untuk seluruh metode berada di atas 10% yang berarti nilai akurasi data uji ini dapat dikategorikan sebagai cukup baik.