Tugas Praktikum Pertemuan 2 - REGRESI

Adinda Shabrina Putri Salsabila G1401221081

Pemanggilan *Packages*

```
library(dplyr)

library(TTR)

library(forecast)

library(lmtest) #digunakan untuk uji formal pendeteksian autokorelasi

library(orcutt) #untuk membuat model regresi Cochrane-Orcutt

library(HoRM) #untuk membuat model regresi Hildreth-Lu
```

Input Data

Data yang digunakan dalam kesempatan kali ini adalah data GDP Eropa Union dari tahun 1960-2023

```
DataEropa <-
read.csv("https://raw.githubusercontent.com/adindashabrina/dataMPDW/main/Data
%20GDP%20Europa%20Union.csv", header = TRUE, sep=",")
head(DataEropa)
```

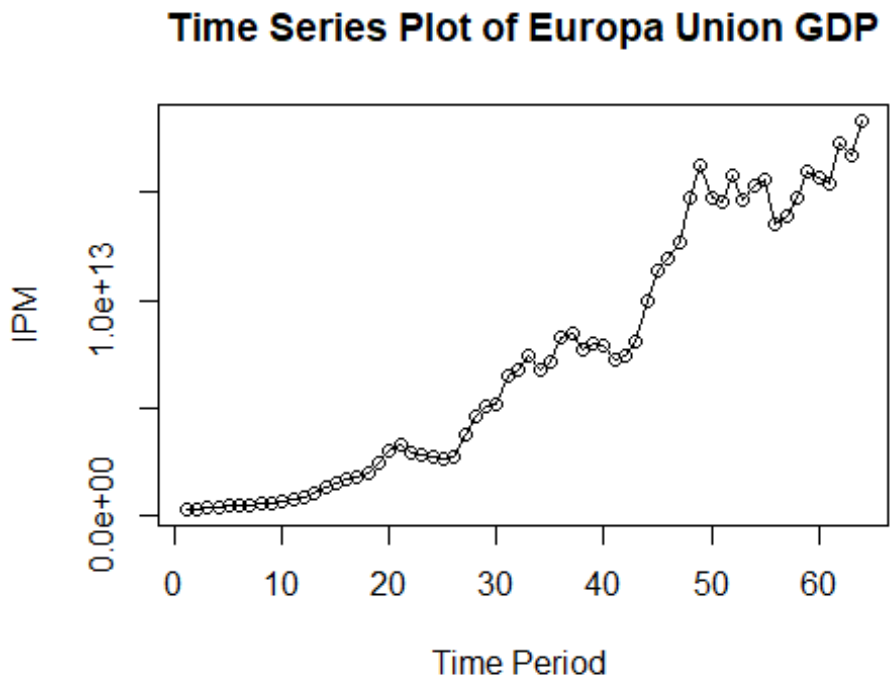
```
##   Tahun          GDP
## 1  1960 285073053086
## 2  1961 316870410351
## 3  1962 350221814291
## 4  1963 384821620915
## 5  1964 426857747596
## 6  1965 467082505496
```

Eksplorasi Data

Sebelum melakukan regresi, akan diperlihatkan *plot time-series* dari GDP Eropa Union periode tahun 1960-2023

```
#Membentuk objek time series
data.ts1<-ts(DataEropa$GDP)
data.ts1
```

```
## Time Series:
## Start = 1
## End = 64
## Frequency = 1
##  [1] 2.850731e+11 3.168704e+11 3.502218e+11 3.848216e+11 4.268577e+11
##  [6] 4.670825e+11 5.057039e+11 5.384998e+11 5.779985e+11 6.450426e+11
## [11] 7.292302e+11 8.242884e+11 9.944014e+11 1.286732e+12 1.456742e+12
## [16] 1.696432e+12 1.777203e+12 2.012781e+12 2.451546e+12 2.962559e+12
## [21] 3.314876e+12 2.893851e+12 2.792073e+12 2.714737e+12 2.615738e+12
## [26] 2.690006e+12 3.755825e+12 4.642293e+12 5.096307e+12 5.205659e+12
## [31] 6.509871e+12 6.733633e+12 7.392186e+12 6.751223e+12 7.155503e+12
## [36] 8.296091e+12 8.431591e+12 7.733844e+12 7.969881e+12 7.925845e+12
## [41] 7.276391e+12 7.393612e+12 8.083520e+12 9.932135e+12 1.141890e+13
## [46] 1.191007e+13 1.271264e+13 1.472752e+13 1.629539e+13 1.476284e+13
## [51] 1.455612e+13 1.576516e+13 1.464163e+13 1.529485e+13 1.565137e+13
## [56] 1.355390e+13 1.388883e+13 1.476588e+13 1.598145e+13 1.569405e+13
## [61] 1.538142e+13 1.731513e+13 1.676150e+13 1.834939e+13

#Membuat plot time series
ts.plot(data.ts1, xlab="Time Period ", ylab="IPM", main= "Time Series Plot of
Europa Union GDP")
points(data.ts1)
```



Selanjutnya akan dilakukan ramalan dan pemulusan dengan metode DMA dan DES karena terlihat pada plot di atas menunjukkan adanya *trend*.

```r
dt.sma1 <- SMA(data.ts1, n=3)
dma1 <- SMA(dt.sma1, n = 3)
At1 <- 2*dt.sma1 - dma1
Bt1 <- 2/(3-1)*(dt.sma1 - dma1)
dt.dma1<- At1+Bt1
dt.ramal1<- c(NA, dt.dma1)

t = 1:5
f = c()

for (i in t) {
  f[i] = At1[length(At1)] + Bt1[length(Bt1)]*(i)
}

dt.gab1 <- cbind(aktual = c(data.ts1,rep(NA,5)),
                 pemulusann1 = c(dt.sma1,rep(NA,5)),
                 pemulusann2 = c(dt.dma1, rep(NA,5)),
                 At = c(At1, rep(NA,5)),
                 Bt = c(Bt1,rep(NA,5)),
                 ramalan1 = c(dt.ramal1, f[-1]))
dt.gab1
```

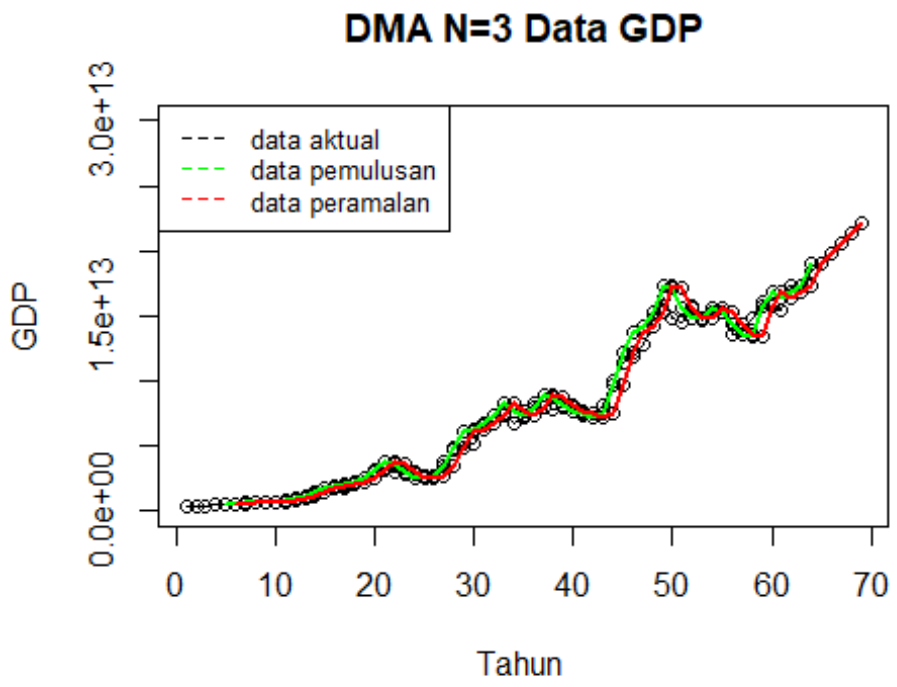
```
##             aktual  pemulusann1  pemulusann2           At            Bt
##  [1,] 2.850731e+11           NA           NA           NA            NA
##  [2,] 3.168704e+11           NA           NA           NA            NA
##  [3,] 3.502218e+11 3.173884e+11           NA           NA            NA
##  [4,] 3.848216e+11 3.506379e+11           NA           NA            NA
##  [5,] 4.268577e+11 3.873004e+11 4.583500e+11 4.228252e+11   35524804702
##  [6,] 4.670825e+11 4.262540e+11 5.026337e+11 4.644438e+11   38189857740
##  [7,] 5.057039e+11 4.665480e+11 5.462425e+11 5.063953e+11   39847241584
##  [8,] 5.384998e+11 5.037620e+11 5.802434e+11 5.420027e+11   38240696441
##  [9,] 5.779985e+11 5.407340e+11 6.148393e+11 5.777867e+11   37052658907
## [10,] 6.450426e+11 5.871803e+11 6.737566e+11 6.304684e+11   43288165543
## [11,] 7.292302e+11 6.507571e+11 7.664904e+11 7.086237e+11   57866629388
## [12,] 8.242884e+11 7.328538e+11 8.847005e+11 8.087771e+11   75923376317
## [13,] 9.944014e+11 8.493067e+11 1.059308e+12 9.543075e+11  105000823738
## [14,] 1.286732e+12 1.035141e+12 1.360555e+12 1.197848e+12  162707016756
## [15,] 1.456742e+12 1.245959e+12 1.650938e+12 1.448449e+12  202489925512
## [16,] 1.696432e+12 1.479969e+12 1.932528e+12 1.706248e+12  226279400128
## [17,] 1.777203e+12 1.643459e+12 2.017453e+12 1.830456e+12  186996874830
## [18,] 2.012781e+12 1.828805e+12 2.184928e+12 2.006867e+12  178061076139
## [19,] 2.451546e+12 2.080510e+12 2.539681e+12 2.310095e+12  229585227041
## [20,] 2.962559e+12 2.475629e+12 3.170257e+12 2.822943e+12  347313942868
## [21,] 3.314876e+12 2.909660e+12 3.751782e+12 3.330721e+12  421060647749
## [22,] 2.893851e+12 3.057095e+12 3.543030e+12 3.300063e+12  242967269058
## [23,] 2.792073e+12 3.000267e+12 3.022785e+12 3.011526e+12   11259220809
## [24,] 2.714737e+12 2.800220e+12 2.495606e+12 2.647913e+12 -152307159603
## [25,] 2.615738e+12 2.707516e+12 2.450546e+12 2.579031e+12 -128485062533
## [26,] 2.690006e+12 2.673494e+12 2.566327e+12 2.619911e+12  -53583047183
## [27,] 3.755825e+12 3.020523e+12 3.460547e+12 3.240535e+12  220012090749
```

```
## [28,] 4.642293e+12 3.696041e+12 4.828086e+12 4.262064e+12  566022126475
## [29,] 5.096307e+12 4.498142e+12 6.017955e+12 5.258048e+12  759906527151
## [30,] 5.205659e+12 4.981420e+12 6.160524e+12 5.570972e+12  589552129454
## [31,] 6.509871e+12 5.603946e+12 6.756166e+12 6.180056e+12  576109894596
## [32,] 6.733633e+12 6.149721e+12 7.292438e+12 6.721080e+12  571358683901
## [33,] 7.392186e+12 6.878563e+12 8.214203e+12 7.546383e+12  667819980336
## [34,] 6.751223e+12 6.959014e+12 7.552176e+12 7.255595e+12  296581257169
## [35,] 7.155503e+12 7.099638e+12 7.340769e+12 7.220203e+12  120565949168
## [36,] 8.296091e+12 7.400939e+12 7.896424e+12 7.648681e+12  247742283114
## [37,] 8.431591e+12 7.961062e+12 8.908760e+12 8.434911e+12  473849043737
## [38,] 7.733844e+12 8.153842e+12 8.784297e+12 8.469070e+12  315227667937
## [39,] 7.969881e+12 8.045105e+12 8.028643e+12 8.036874e+12   -8231184951
## [40,] 7.925845e+12 7.876523e+12 7.579256e+12 7.727889e+12 -148633742858
## [41,] 7.276391e+12 7.724039e+12 7.408338e+12 7.566188e+12 -157850293342
## [42,] 7.393612e+12 7.531949e+12 7.174173e+12 7.353061e+12 -178887836846
## [43,] 8.083520e+12 7.584507e+12 7.526526e+12 7.555517e+12  -28990932904
## [44,] 9.932135e+12 8.469755e+12 9.685125e+12 9.077440e+12  607684804775
## [45,] 1.141890e+13 9.811518e+12 1.219070e+13 1.100111e+13 1189591182715
## [46,] 1.191007e+13 1.108703e+13 1.368223e+13 1.238463e+13 1297598548582
## [47,] 1.271264e+13 1.201387e+13 1.409999e+13 1.305693e+13 1043061243138
## [48,] 1.472752e+13 1.311674e+13 1.520513e+13 1.416094e+13 1044193576561
## [49,] 1.629539e+13 1.457852e+13 1.726280e+13 1.592066e+13 1342140526588
## [50,] 1.476284e+13 1.526192e+13 1.714764e+13 1.620478e+13  942859542452
## [51,] 1.455612e+13 1.520479e+13 1.558421e+13 1.539450e+13  189712153287
## [52,] 1.576516e+13 1.502804e+13 1.475429e+13 1.489117e+13 -136873739344
## [53,] 1.464163e+13 1.498764e+13 1.481594e+13 1.490179e+13  -85849765904
## [54,] 1.529485e+13 1.523388e+13 1.553527e+13 1.538458e+13  150695096130
## [55,] 1.565137e+13 1.519595e+13 1.530954e+13 1.525275e+13   56795550403
## [56,] 1.355390e+13 1.483337e+13 1.432465e+13 1.457901e+13 -254362343174
## [57,] 1.388883e+13 1.436470e+13 1.349808e+13 1.393139e+13 -433308725389
## [58,] 1.476588e+13 1.406954e+13 1.336354e+13 1.371654e+13 -353000623455
## [59,] 1.598145e+13 1.487872e+13 1.576085e+13 1.531979e+13  441067183325
## [60,] 1.569405e+13 1.548046e+13 1.682223e+13 1.615135e+13  670886922293
## [61,] 1.538142e+13 1.568564e+13 1.636037e+13 1.602300e+13  337365962329
## [62,] 1.731513e+13 1.613020e+13 1.685973e+13 1.649497e+13  364766767370
## [63,] 1.676150e+13 1.648601e+13 1.725681e+13 1.687141e+13  385397089976
## [64,] 1.834939e+13 1.747534e+13 1.903165e+13 1.825349e+13  778154001476
## [65,]           NA           NA           NA           NA            NA
## [66,]           NA           NA           NA           NA            NA
## [67,]           NA           NA           NA           NA            NA
## [68,]           NA           NA           NA           NA            NA
## [69,]           NA           NA           NA           NA            NA
##           ramalan1
##  [1,]           NA
##  [2,]           NA
##  [3,]           NA
##  [4,]           NA
##  [5,]           NA
##  [6,] 4.583500e+11
##  [7,] 5.026337e+11
```

```
##  [8,] 5.462425e+11
##  [9,] 5.802434e+11
## [10,] 6.148393e+11
## [11,] 6.737566e+11
## [12,] 7.664904e+11
## [13,] 8.847005e+11
## [14,] 1.059308e+12
## [15,] 1.360555e+12
## [16,] 1.650938e+12
## [17,] 1.932528e+12
## [18,] 2.017453e+12
## [19,] 2.184928e+12
## [20,] 2.539681e+12
## [21,] 3.170257e+12
## [22,] 3.751782e+12
## [23,] 3.543030e+12
## [24,] 3.022785e+12
## [25,] 2.495606e+12
## [26,] 2.450546e+12
## [27,] 2.566327e+12
## [28,] 3.460547e+12
## [29,] 4.828086e+12
## [30,] 6.017955e+12
## [31,] 6.160524e+12
## [32,] 6.756166e+12
## [33,] 7.292438e+12
## [34,] 8.214203e+12
## [35,] 7.552176e+12
## [36,] 7.340769e+12
## [37,] 7.896424e+12
## [38,] 8.908760e+12
## [39,] 8.784297e+12
## [40,] 8.028643e+12
## [41,] 7.579256e+12
## [42,] 7.408338e+12
## [43,] 7.174173e+12
## [44,] 7.526526e+12
## [45,] 9.685125e+12
## [46,] 1.219070e+13
## [47,] 1.368223e+13
## [48,] 1.409999e+13
## [49,] 1.520513e+13
## [50,] 1.726280e+13
## [51,] 1.714764e+13
## [52,] 1.558421e+13
## [53,] 1.475429e+13
## [54,] 1.481594e+13
## [55,] 1.553527e+13
## [56,] 1.530954e+13
## [57,] 1.432465e+13
```

```
## [58,] 1.349808e+13
## [59,] 1.336354e+13
## [60,] 1.576085e+13
## [61,] 1.682223e+13
## [62,] 1.636037e+13
## [63,] 1.685973e+13
## [64,] 1.725681e+13
## [65,] 1.903165e+13
## [66,] 1.980980e+13
## [67,] 2.058795e+13
## [68,] 2.136611e+13
## [69,] 2.214426e+13
```

```r
#Plot time series
ts.plot(dt.gab1[,1], xlab="Tahun ", ylab="GDP",
        main= "DMA N=3 Data GDP", ylim=c(0,29932134719712))
points(dt.gab1[,1])
points(dt.gab1[,3])
points(dt.gab1[,6])
lines(dt.gab1[,3],col="green",lwd=2)
lines(dt.gab1[,6],col="red",lwd=2)
legend("topleft",c("data aktual","data pemulusan","data peramalan"),
       lty=8, col=c("black","green","red"), cex=0.8)
```



Selanjutnya akan dilihat keakuratan dari metode DMA

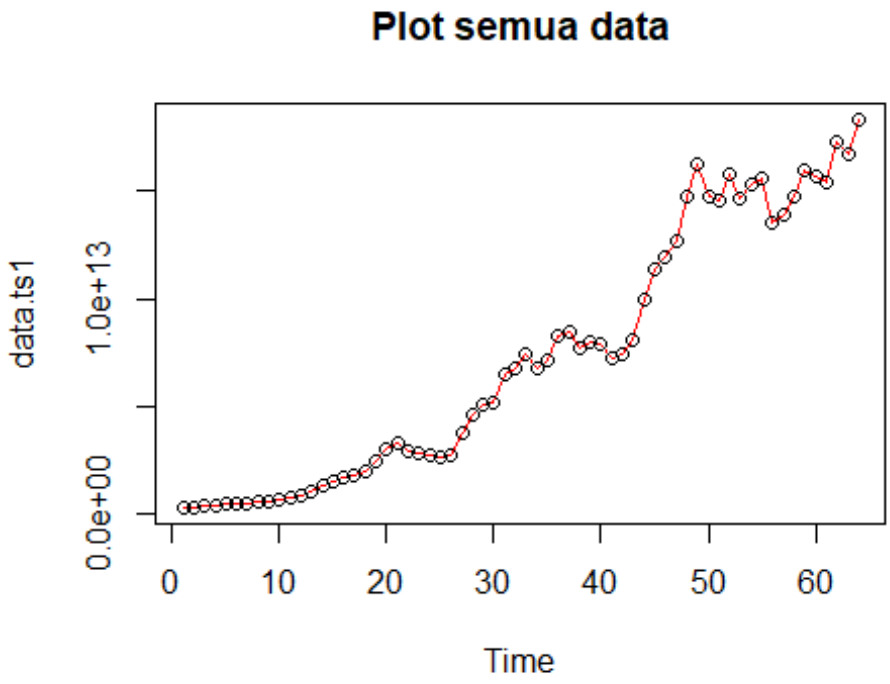
```r
#Menghitung nilai keakuratan
error.dma1 = data.ts1-dt.ramal1[1:length(data.ts1)]
SSE.dma1 = sum(error.dma1[6:length(data.ts1)]^2)
MSE.dma1 = mean(error.dma1[6:length(data.ts1)]^2)
MAPE.dma1 =
mean(abs((error.dma1[6:length(data.ts1)]/data.ts1[6:length(data.ts1)])*100))

akurasi.dma1 <- matrix(c(SSE.dma1, MSE.dma1, MAPE.dma1))
row.names(akurasi.dma1)<- c("SSE", "MSE", "MAPE")
colnames(akurasi.dma1) <- c("Akurasi m = 3")
akurasi.dma1
```

```
##       Akurasi m = 3
## SSE    5.256632e+25
## MSE    8.909545e+23
## MAPE   8.774146e+00
```

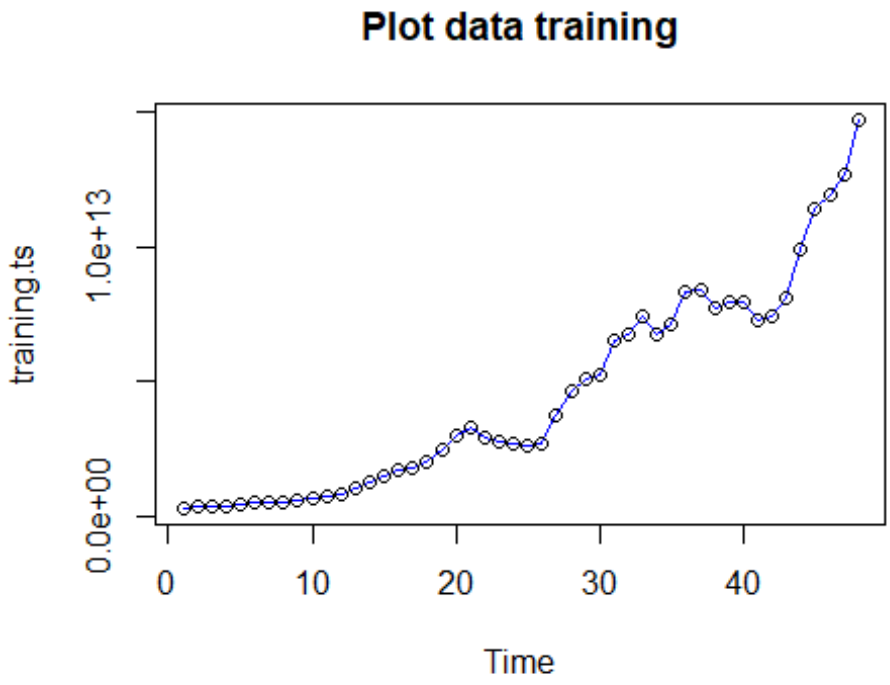
Dalam hal ini nilai MAPE pada metode pemulusan DMA kurang dari 10%, nilai ini dapat dikategorikan sebagai nilai akurasi yang sangat baik.Selanjutnya akan digunakan metode *Double Exponential Smoothing* dengan cara sebagai berikut.

Pertama akan data akan dibagi menjadi data *training* dan data *testing*.

```r
#membagi training dan testing
training<-DataEropa[1:48,2]
testing<-DataEropa[49:64,2]

#data time series
training.ts<-ts(training)
testing.ts<-ts(testing,start=49)

#eksplorasi data
plot(data.ts1, col="red",main="Plot semua data")
points(data.ts1)
```

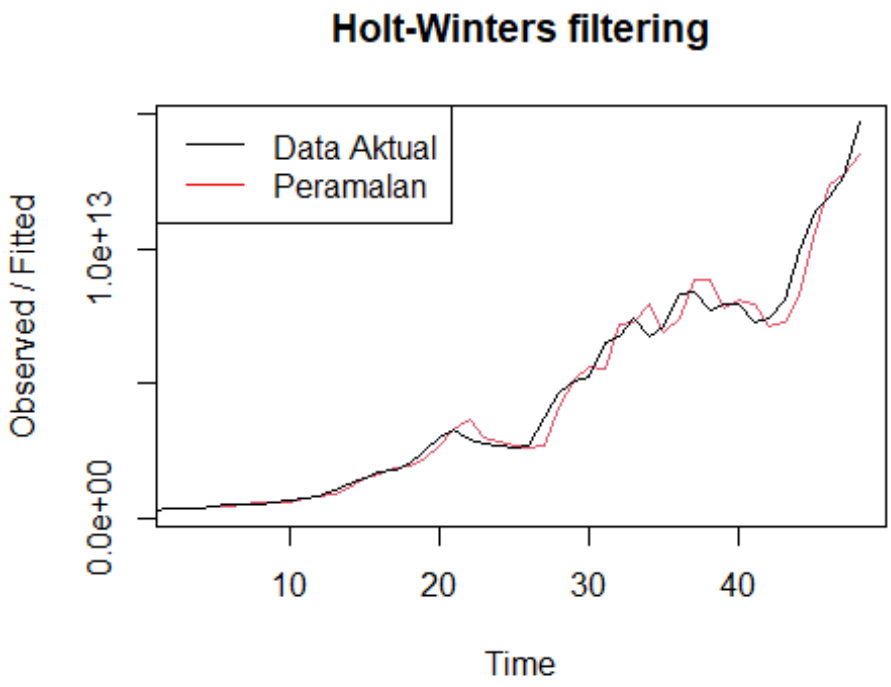



```
plot(training.ts, col="blue",main="Plot data training")
points(training.ts)
```



Selanjutnya akan dilakukan pemulusan dengan DES, kali ini langsung dicari lambda dan gamma optimum sebagai berikut. Nilai lambda dan gamma optimum dapat dilihat pada smoothing parameters alpha untuk nilai lambda dan beta untuk nilai gamma.

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

## Holt-Winters exponential smoothing with trend and without seasonal
component.
##
## Call:
## HoltWinters(x = training.ts, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 1
##  beta : 0.3414857
##  gamma: FALSE
##
## Coefficients:
##           [,1]
## a 1.472752e+13
## b 1.227295e+12

plot(des.opt)
legend("topleft", c("Data Aktual", "Peramalan"), col = c("black", "red"),
       lty = c(1,1))
```



```
#ramalan
ramalandesopt<- forecast(des.opt, h=16)
ramalandesopt
```

```
##    Point Forecast        Lo 80        Hi 80        Lo 95        Hi 95
## 49   1.595482e+13 1.525863e+13 1.665101e+13 1.489008e+13 1.701955e+13
## 50   1.718211e+13 1.601725e+13 1.834698e+13 1.540061e+13 1.896362e+13
## 51   1.840941e+13 1.675722e+13 2.006159e+13 1.588261e+13 2.093621e+13
## 52   1.963670e+13 1.746503e+13 2.180837e+13 1.631542e+13 2.295799e+13
## 53   2.086400e+13 1.813833e+13 2.358966e+13 1.669545e+13 2.503254e+13
## 54   2.209129e+13 1.877737e+13 2.540521e+13 1.702309e+13 2.715949e+13
## 55   2.331859e+13 1.938316e+13 2.725402e+13 1.729987e+13 2.933731e+13
## 56   2.454588e+13 1.995688e+13 2.913488e+13 1.752761e+13 3.156415e+13
## 57   2.577318e+13 2.049975e+13 3.104661e+13 1.770816e+13 3.383820e+13
## 58   2.700047e+13 2.101290e+13 3.298805e+13 1.784326e+13 3.615768e+13
## 59   2.822777e+13 2.149739e+13 3.495815e+13 1.793454e+13 3.852100e+13
## 60   2.945506e+13 2.195421e+13 3.695592e+13 1.798349e+13 4.092664e+13
## 61   3.068236e+13 2.238424e+13 3.898048e+13 1.799149e+13 4.337323e+13
## 62   3.190965e+13 2.278833e+13 4.103098e+13 1.795979e+13 4.585952e+13
## 63   3.313695e+13 2.316722e+13 4.310668e+13 1.788956e+13 4.838434e+13
## 64   3.436425e+13 2.352162e+13 4.520687e+13 1.778188e+13 5.094661e+13
```

Selanjutnya akan dicari akurasi dari metode DES.

```
ssedes.train<-des.opt$SSE
msedes.train<-ssedes.train/length(training.ts)
sisaandes<-ramalandesopt$residuals
head(sisaandes)
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1]         NA         NA 1554046676 2271764710 8932309704 4070685282
```

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

akurasides.opt <- matrix(c(ssedes.train,msedes.train,mapedes.train))
row.names(akurasides.opt)<- c("SSE", "MSE", "MAPE")
colnames(akurasides.opt) <- c("Akurasi lamda dan gamma optimum")
akurasides.opt
```

```
##      Akurasi lamda dan gamma optimum
## SSE                     1.354641e+25
## MSE                     2.822170e+23
## MAPE                    6.441161e+00
```

Dalam hal ini nilai MAPE pada metode DES kurang dari 10%, nilai ini dapat dikategorikan sebagai nilai akurasi yang sangat baik.

```
#Akurasi data testing
selisihdesopt<-ramalandesopt$mean-testing.ts
selisihdesopt

## Time Series:
## Start = 49
## End = 64
## Frequency = 1
##  [1] -3.405749e+11  2.419269e+12  3.853285e+12  3.871543e+12  6.222363e+12
##  [6]  6.796439e+12  7.667215e+12  1.099199e+13  1.188435e+13  1.223459e+13
## [11]  1.224632e+13  1.376101e+13  1.530094e+13  1.459453e+13  1.637545e+13
## [16]  1.601486e+13

SSEtestingdesopt<-sum(selisihdesopt^2)
SSEtestingdesopt<-SSEtestingdesopt/length(testing.ts)
MAPEtestingdesopt<-sum(abs(selisihdesopt/testing.ts)*100)/length(testing.ts)

akurasiDesTesting <-
matrix(c(SSEtestingdesopt,SSEtestingdesopt,MAPEtestingdesopt))
row.names(akurasiDesTesting)<- c("SSE", "MSE", "MAPE")
colnames(akurasiDesTesting) <- c("Akurasi lamda dan gamma optimum")
akurasiDesTesting

##      Akurasi lamda dan gamma optimum
## SSE                      1.18896e+26
## MSE                      1.18896e+26
## MAPE                     6.17501e+01
```

Setelah didapatkan nilai akurasi untuk metode DMA dan DES, selanjutnya akan dibandingkan keakuratan antar metode keduanya.

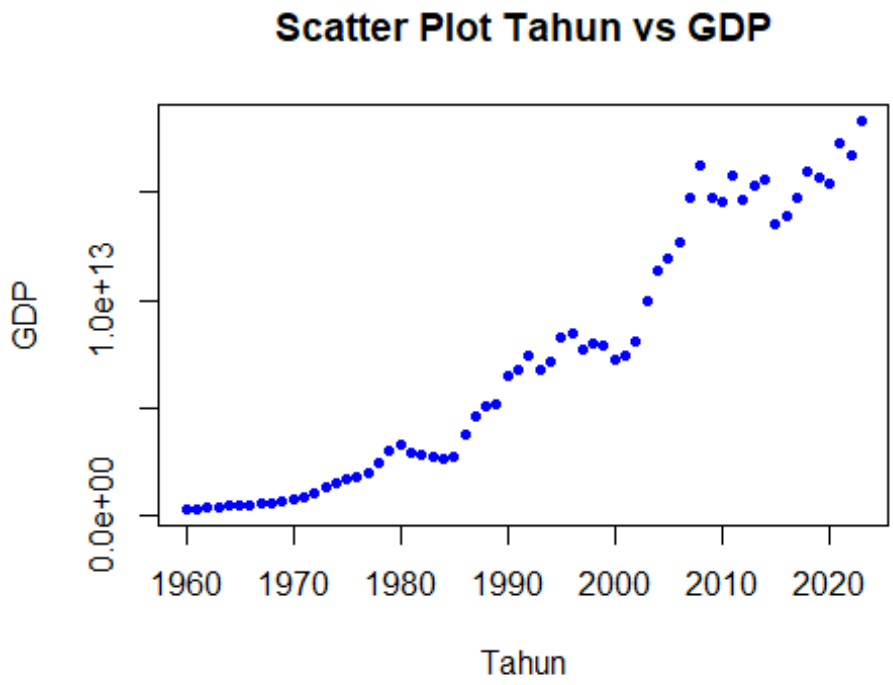
```
cbind(akurasi.dma1, akurasides.opt)

##      Akurasi m = 3 Akurasi lamda dan gamma optimum
## SSE   5.256632e+25                    1.354641e+25
## MSE   8.909545e+23                    2.822170e+23
## MAPE  8.774146e+00                    6.441161e+00
```

Berdasarkan perbandingan akurasi tersebut, terlihat nilai SSE, MSE, dan MAPE metode DES lebih kecil dibandingkan dengan metode DMA. Oleh karena itu, metode peramalan dan pemulusan yang terbaik antara keduanya adalah dengan metode DES.

Setelah melakukan peramalan, data yang telah dimasukkan kemudian dieksplorasi. Eksplorasi pertama yang dilakukan adalah dengan menggunakan *scatter plot*.

```
#Eksplorasi Data

Tahun <- c(DataEropa$Tahun)
GDP <- c(DataEropa$GDP)
#Pembuatan Scatter Plot
plot(Tahun, GDP, pch = 20, col = "blue",
```

```
     main = "Scatter Plot Tahun vs GDP",
     xlab = "Tahun",
     ylab = "GDP")
```




```
#Menampilkan Nilai Korelasi
cor(Tahun,GDP)

## [1] 0.9686201
```

Berdasarkan scatter plot di atas, terlihat adanya hubungan / korelasi positif antara peubah tahun dengan nilai IPM, terlihat titik-titik pada plot yang naik ke arah kanan atas. Hal tersebut juga diperkuat dengan hasil perhitungan aplikasi R di mana didapatkan nilai korelasi sebesar 0.9686201.

Setalah mengetahui adanya hubungan antar dua peubah, maka model regresi dapat ditentukan.

Regresi

```
#Pembuatan Model Regresi
#model regresi
model<- lm(GDP~Tahun, data = DataEropa)
summary(model)

##
## Call:
## lm(formula = GDP ~ Tahun, data = DataEropa)
```

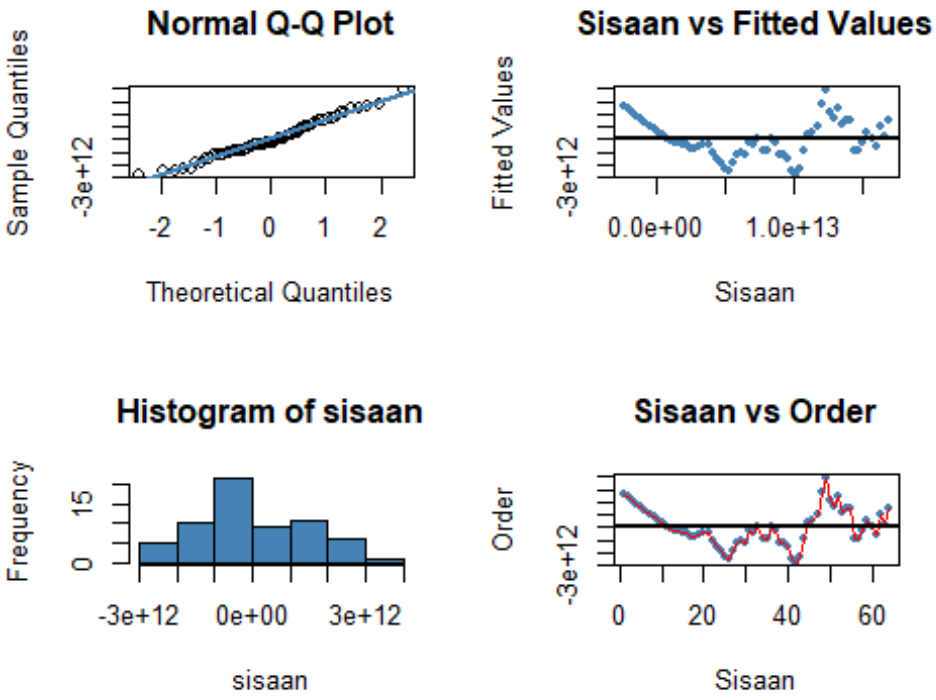
```
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.761e+12 -9.086e+11 -2.684e+11  1.041e+12  3.994e+12
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.036e+14  1.991e+13  -30.32   <2e-16 ***
## Tahun        3.067e+11  9.995e+09   30.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.477e+12 on 62 degrees of freedom
## Multiple R-squared:  0.9382, Adjusted R-squared:  0.9372
## F-statistic: 941.6 on 1 and 62 DF,  p-value: < 2.2e-16
```

Model yang dihasilkan adalah

$$y_i = -6.036e14 + 3.067e11x_t$$

Berdasarkan ringkasan model dapat diketahui bahwa hasil uji F memiliki *p-value* < α (5%). Artinya, minimal terdapat satu variabel yang berpengaruh nyata terhadap model. Hasil uji-t parsial kedua parameter regresi, yaitu intersep dan koefisien regresi juga menunjukkan hal yang sama, yaitu memiliki *p-value* < α (5%) sehingga nyata dalam taraf 5%. Selanjutnya dapat dilihat juga nilai $R^2 = 0.9372$. Artinya, sebesar 93.72% keragaman nilai GDP dapat dijelaskan oleh peubah tahun. Hasil ini menunjukkan hasil yang bagus, seolah mendapatkan hasil terbaik. Namun, kita perlu melakukan uji terhadap sisaannya seperti berikut ini.

```
#sisaan dan fitted value
sisaan<- residuals(model)
fitValue<- predict(model)

#Diagnostik dengan eksploratif
par(mfrow = c(2,2))
qqnorm(sisaan)
qqline(sisaan, col = "steelblue", lwd = 2)
plot(fitValue, sisaan, col = "steelblue", pch = 20, xlab = "Sisaan", ylab =
"Fitted Values", main = "Sisaan vs Fitted Values")
abline(a = 0, b = 0, lwd = 2)
hist(sisaan, col = "steelblue")
plot(seq(1,64,1), sisaan, col = "steelblue", pch = 20, xlab = "Sisaan", ylab
= "Order", main = "Sisaan vs Order")
lines(seq(1,64,1), sisaan, col = "red")
abline(a = 0, b = 0, lwd = 2)
```

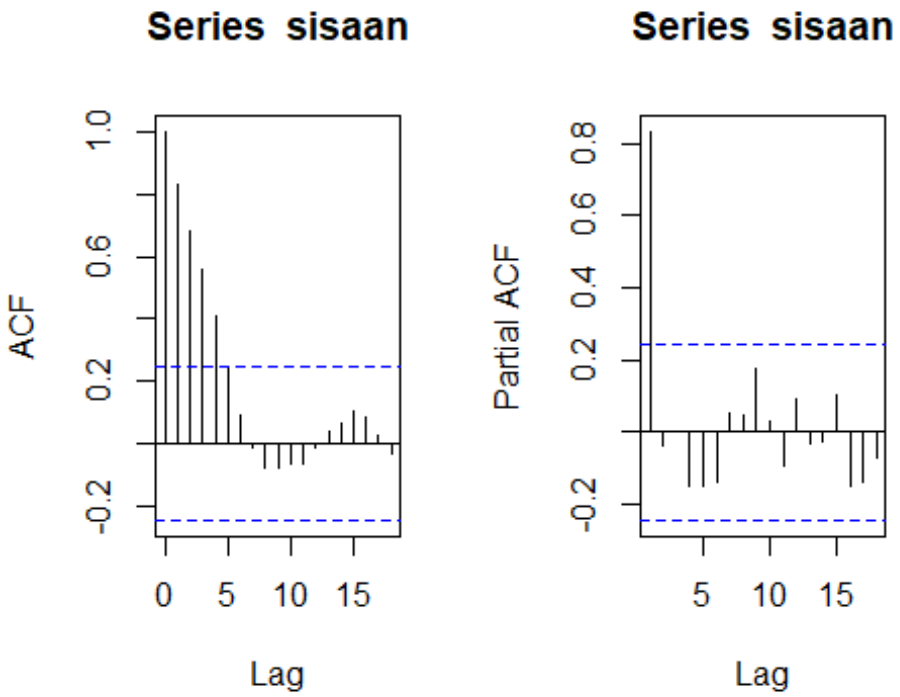


Dua plot di samping kiri digunakan untuk melihat apakah sisaan menyebar normal. Normal Q-Q Plot di atas menunjukkan bahwa sisaan cenderung menyebar normal, tetapi histogram dari sisaan kurang menunjukkan demikian. Selanjutnya, dua plot di samping kanan digunakan untuk melihat autokorelasi. Plot Sisaan vs *Fitted Value* dan Plot Sisaan vs *Order* menunjukkan adanya pola pada sisaan. Untuk lebih lanjut akan digunakan uji formal melihat normalitas sisaan dan plot ACF dan PACF untuk melihat apakah ada autokorelasi atau tidak.

```
#Melihat Sisaan Menyebar Normal/Tidak
#H0: sisaan mengikuti sebaran normal
#H1: sisaan tidak mengikuti sebaran normal
shapiro.test(sisaan)

##
##  Shapiro-Wilk normality test
##
## data:  sisaan
## W = 0.97986, p-value = 0.3791

ks.test(sisaan, "pnorm", mean=mean(sisaan), sd=sd(sisaan))

##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  sisaan
## D = 0.088724, p-value = 0.6618
## alternative hypothesis: two-sided
```

Berdasarkan uji formal Saphiro-Wilk dan Kolmogorov-Smirnov didapatkan nilai *p-value* > α (5%). Artinya, cukup bukti untuk menyatakan sisaan berdistribusi normal.

```
#ACF dan PACF identifikasi autokorelasi
par(mfrow = c(1,2))
acf(sisaan)
pacf(sisaan)
```



Berdasarkan plot ACF dan PACF, terlihat semua diluar rentang batas dan terdapat lag yang signifikan. Namun, untuk lebih memastikan akan dilakukan uji formal dengan uji Durbin Watson.

```
#Deteksi autokorelasi dengan uji-Durbin Watson
#H0: tidak ada autokorelasi
#H1: ada autokorelasi
dwtest(model)

##
##  Durbin-Watson test
##
## data:  model
## DW = 0.2647, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

(nilai Dl dan DU berasal dari tabel durbin watson, dimana k= jumlah parameter dan n jumlah observasi) Berdasarkan hasil DW Test, didapatkan nilai $DW = 0.2647$ dan *p-value* = $2.2e-16$. Berdasarkan tabel Durbin-Watson diperoleh nilai $DU = 1.5634$ dan $DL =$

1.6268. Nilai DW berada di bawah nilai DL. Artinya, dapat dikatakan berada di daerah autokorelasi positif. Dengan nilai *p-value* < 0.05 juga dapat disimpulkan bahwa tolak H0, cukup bukti mengatakan adanya autokorelasi. Oleh karena itu, diperlukan penangan autokorelasi. Penanganan yang akan digunakan menggunakan dua metode, yaitu Cochrane-Orcutt dan Hildret-Lu.

Penanganan Autokorelasi

Metode Cochrane-Orcutt

Penanganan metode Cochrane-Orcutt dapat dilakukan dengan bantuan packages Orcutt pada aplikasi R maupun secara manual. Berikut ini ditampilkan cara menggunakan bantuan `library` *packages* `Orcutt`.

```
#Penanganan Autokorelasi Cochrane-Orcutt
modelCO<-cochrane.orcutt(model)
modelCO
```

```
## Cochrane-orcutt estimation for first order autocorrelation
##
## Call:
## lm(formula = GDP ~ Tahun, data = DataEropa)
##
##  number of interaction: 10
##  rho 0.847797
##
## Durbin-Watson statistic
## (original):    0.26470 , p-value: 1.804e-22
## (transformed): 1.86790 , p-value: 2.548e-01
##
##  coefficients:
##   (Intercept)         Tahun
## -6.830045e+14  3.463974e+11
```

Hasil keluaran model setelah dilakukan penanganan adalah sebagai berikut.

$$y_i = -6.830045e14 + 3.463974e11x_t$$

Hasil juga menunjukkan bahwa nilai DW dan p-value meningkat menjadi 1.86790 dan $2.548e-01$. Nilai DW sudah berada pada rentang DU < DW < 4-DU atau $1.5634 < DW < 2.436$. Hal tersebut juga didukung dengan nilai *p-value* > 0.05, artinya belum cukup bukti menyatakan bahwa sisaan terdapat autokorelasi pada taraf nyata 5%. Untuk nilai $\hat{\rho}$ optimum yang digunakan adalah 0.8477969. Nilai tersebut dapat diketahui dengan *syntax* berikut.

```
#Rho optimum
rho<- modelCO$rho
rho
```

```
## [1] 0.8477969
```

Selanjutnya akan dilakukan transformasi secara manual dengan syntax berikut ini.

```
GDP
```

```
##  [1] 2.850731e+11 3.168704e+11 3.502218e+11 3.848216e+11 4.268577e+11
##  [6] 4.670825e+11 5.057039e+11 5.384998e+11 5.779985e+11 6.450426e+11
## [11] 7.292302e+11 8.242884e+11 9.944014e+11 1.286732e+12 1.456742e+12
## [16] 1.696432e+12 1.777203e+12 2.012781e+12 2.451546e+12 2.962559e+12
## [21] 3.314876e+12 2.893851e+12 2.792073e+12 2.714737e+12 2.615738e+12
## [26] 2.690006e+12 3.755825e+12 4.642293e+12 5.096307e+12 5.205659e+12
## [31] 6.509871e+12 6.733633e+12 7.392186e+12 6.751223e+12 7.155503e+12
## [36] 8.296091e+12 8.431591e+12 7.733844e+12 7.969881e+12 7.925845e+12
## [41] 7.276391e+12 7.393612e+12 8.083520e+12 9.932135e+12 1.141890e+13
## [46] 1.191007e+13 1.271264e+13 1.472752e+13 1.629539e+13 1.476284e+13
## [51] 1.455612e+13 1.576516e+13 1.464163e+13 1.529485e+13 1.565137e+13
## [56] 1.355390e+13 1.388883e+13 1.476588e+13 1.598145e+13 1.569405e+13
## [61] 1.538142e+13 1.731513e+13 1.676150e+13 1.834939e+13
```

```
GDP[-1]
```

```
##  [1] 3.168704e+11 3.502218e+11 3.848216e+11 4.268577e+11 4.670825e+11
##  [6] 5.057039e+11 5.384998e+11 5.779985e+11 6.450426e+11 7.292302e+11
## [11] 8.242884e+11 9.944014e+11 1.286732e+12 1.456742e+12 1.696432e+12
## [16] 1.777203e+12 2.012781e+12 2.451546e+12 2.962559e+12 3.314876e+12
## [21] 2.893851e+12 2.792073e+12 2.714737e+12 2.615738e+12 2.690006e+12
## [26] 3.755825e+12 4.642293e+12 5.096307e+12 5.205659e+12 6.509871e+12
## [31] 6.733633e+12 7.392186e+12 6.751223e+12 7.155503e+12 8.296091e+12
## [36] 8.431591e+12 7.733844e+12 7.969881e+12 7.925845e+12 7.276391e+12
## [41] 7.393612e+12 8.083520e+12 9.932135e+12 1.141890e+13 1.191007e+13
## [46] 1.271264e+13 1.472752e+13 1.629539e+13 1.476284e+13 1.455612e+13
## [51] 1.576516e+13 1.464163e+13 1.529485e+13 1.565137e+13 1.355390e+13
## [56] 1.388883e+13 1.476588e+13 1.598145e+13 1.569405e+13 1.538142e+13
## [61] 1.731513e+13 1.676150e+13 1.834939e+13
```

```
#Transformasi Manual
GDP.trans<- GDP[-1]-GDP[-12]*rho
Tahun.trans<- Tahun[-1]-Tahun[-12]*rho
modelCOmanual<- lm(GDP.trans~Tahun.trans)
summary(modelCOmanual)
```

```
##
## Call:
## lm(formula = GDP.trans ~ Tahun.trans)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -4.095e+11 -1.202e+11  8.436e+09  8.973e+10  5.890e+11
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.003e+14  2.826e+12   -35.5   <2e-16 ***
## Tahun.trans  3.345e+11  9.317e+09    35.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.899e+11 on 61 degrees of freedom
## Multiple R-squared:  0.9548, Adjusted R-squared:  0.9541
## F-statistic:  1289 on 1 and 61 DF,  p-value: < 2.2e-16
```

Hasil model transformasi bukan merupakan model sesungguhnya. Koefisien regresi masih perlu dicari kembali mengikuti $\beta_0^* = \beta_0 + \hat{\rho}\beta_0$ dan $\beta_1^* = \beta_1$.

```
#Mencari Penduga Koefisien Regresi setelah Transformasi ke Persamaan Awal
b0bintang <- modelCOmanual$coefficients[-2]
b0 <- b0bintang/(1-rho)
b1 <- modelCOmanual$coefficients[-1]
b0

##   (Intercept)
## -6.592499e+14

b1

##  Tahun.trans
## 334499843695
```

Hasil perhitungan koefisien regresi tersebut akan menghasilkan hasil yang sama dengan model yang dihasilkan menggunakan *packages*.

Metode Hildreth-Lu

Penanganan kedua adalah menggunakan metode Hildreth-Lu. Metode ini akan mencari nilai SSE terkecil dan dapat dicari secara manual maupun menggunakan packages. Jika menggunakan packages, gunakan `library` *packages* `HORM`.

```
#Penanganan Autokorelasi Hildreth lu
# Hildreth-Lu
hildreth.lu.func<- function(r, model){
  x <- model.matrix(model)[,-1]
  y <- model.response(model.frame(model))
  n <- length(y)
  t <- 2:n
  y <- y[t]-r*y[t-1]
  x <- x[t]-r*x[t-1]

  return(lm(y~x))
}

#Pencariab rho yang meminimumkan SSE
r <- c(seq(0.8,0.9, by= 0.01))
```

```
tab <- data.frame("rho" = r, "SSE" = sapply(r,
function(i){deviance(hildreth.lu.func(i, model))}))
round(tab, 4)
```

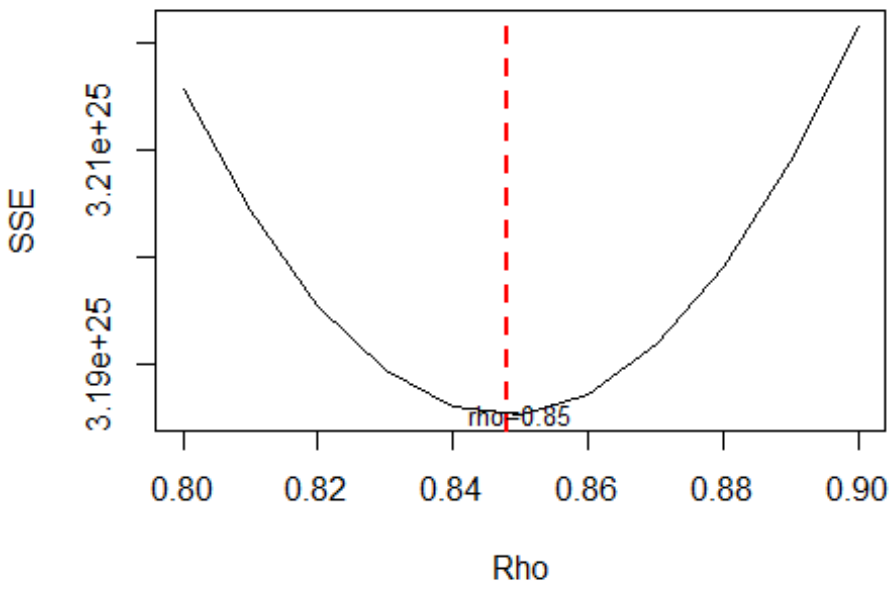
```
##     rho          SSE
## 1  0.80 3.215608e+25
## 2  0.81 3.204220e+25
## 3  0.82 3.195493e+25
## 4  0.83 3.189427e+25
## 5  0.84 3.186022e+25
## 6  0.85 3.185277e+25
## 7  0.86 3.187194e+25
## 8  0.87 3.191772e+25
## 9  0.88 3.199010e+25
## 10 0.89 3.208910e+25
## 11 0.90 3.221470e+25
```

Pertama-tama akan dicari di mana kira-kira ρ yang menghasilkan SSE minimum. Pada hasil di atas terlihat ρ minimum ketika 0.85. Namun, hasil tersebut masih kurang teliti sehingga akan dicari kembali ρ yang lebih optimum dengan ketelitian yang lebih. Jika sebelumnya jarak antar ρ yang dicari adalah 0.01, kali ini jarak antar ρ adalah 0.001 dan dilakukan pada selang 0.8 sampai dengan 0.9.

```
#Rho optimal di sekitar 0.4
rOpt <- seq(0.8,0.9, by= 0.001)
tabOpt <- data.frame("rho" = rOpt, "SSE" = sapply(rOpt,
function(i){deviance(hildreth.lu.func(i, model))}))
head(tabOpt[order(tabOpt$SSE),])
```

```
##      rho          SSE
## 49 0.848 3.185213e+25
## 48 0.847 3.185221e+25
## 50 0.849 3.185232e+25
## 47 0.846 3.185256e+25
## 51 0.850 3.185277e+25
## 46 0.845 3.185317e+25
```

```
#Grafik SSE optimum
par(mfrow = c(1,1))
plot(tab$SSE ~ tab$rho , type = "l", xlab = "Rho", ylab = "SSE")
abline(v = tabOpt[tabOpt$SSE==min(tabOpt$SSE),"rho"], lty = 2,
col="red",lwd=2)
text(x=0.85, y=3.185277e+25 , labels = "rho=0.85", cex = 0.8)
```



Perhitungan yang dilakukan aplikasi R menunjukkan bahwa nilai ρ optimum, yaitu saat SSE terkecil terdapat pada nilai $\rho = 0.85$. Hal tersebut juga ditunjukkan pada plot. Selanjutnya, model dapat didapatkan dengan mengevaluasi nilai ρ ke dalam fungsi `hildreth.lu.func`, serta dilanjutkan dengan pengujian autokorelasi dengan uji Durbin-Watson. Namun, setelah pengecekan tersebut tidak lupa koefisien regresi tersebut digunakan untuk transformasi balik. Persamaan hasil transformasi itulah yang menjadi persamaan sesungguhnya.

```
#Model terbaik
modelHL <- hildreth.lu.func(0.85, model)
summary(modelHL)

##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.293e+12 -4.051e+11  4.500e+09  3.174e+11  1.795e+12
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.026e+14  1.000e+13  -10.26 6.58e-15 ***
## x            3.470e+11  3.338e+10   10.40 3.94e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 7.226e+11 on 61 degrees of freedom
## Multiple R-squared:  0.6392, Adjusted R-squared:  0.6333
## F-statistic: 108.1 on 1 and 61 DF,  p-value: 3.943e-15

#Transformasi Balik
cat("y = ", coef(modelHL)[1]/(1-0.85), "+", coef(modelHL)[2],"x", sep = "")

## y = -6.842399e+14+347012930176x
```

Setelah dilakukan tranformasi balik, didapatkan model dengan metode Hildreth-Lu sebagai berikut.

$$y_i = -6.842399e14 + 347012930176x_t$$

```
#Deteksi autokorelasi
dwtest(modelHL)

##
##  Durbin-Watson test
##
## data:  modelHL
## DW = 1.8718, p-value = 0.2598
## alternative hypothesis: true autocorrelation is greater than 0
```

Hasil uji Durbin-Watson juga menunjukkan bawah nilai DW sebesar 1.8718 berada pada selang daerah tidak ada autokorelasi, yaitu pada rentang DU < DW < 4-DU atau $1.5634 < DW < 2.4366$. Hal tersebut juga didukung oleh *p-value* sebesar 0.2598, di mana *p-value* > α=5%. Artinya tak tolak H_0 atau belum cukup bukti menyatakan bahwa ada autokorelasi dalam data nilai GDP dengan metode Hildreth-Lu pada taraf nyata 5%.

Terakhir, akan dibandingkan nilai SSE dari ketiga metode (metode awal, metode Cochrane-Orcutt, dan Hildreth-Lu).

```
#Perbandingan
sseModelawal <- anova(model)$`Sum Sq`[-1]
sseModelCO <- anova(modelCOmanual)$`Sum Sq`[-1]
sseModelHL <- anova(modelHL)$`Sum Sq`[-1]
mseModelawal <- sseModelawal/length(GDP)
mseModelCO <- sseModelCO/length(GDP)
mseModelHL <- sseModelHL/length(GDP)
akurasi <- matrix(c(sseModelawal,sseModelCO,sseModelHL,
                    mseModelawal,mseModelCO,mseModelHL),nrow=2,ncol=3,byrow =
T)
colnames(akurasi) <- c("Model Awal", "Model Cochrane-Orcutt", "Model
Hildreth-Lu")
row.names(akurasi) <- c("SSE","MSE")
akurasi
```

```
##       Model Awal Model Cochrane-Orcutt Model Hildreth-Lu
## SSE 1.352759e+26          2.198923e+24      3.185277e+25
## MSE 2.113687e+24          3.435818e+22      4.976996e+23
```

Berdasarkan hasil tersebut dapat diketahui bahwa hasil penanganan autokorelasi dengan metode Cochrane-Orcutt memiliki nilai SSE yang lebih rendah yaitu sebesar $2.198923e + 24$, metode Hildreth-Lu sebesar $3.185277e + 25$ dan lebih baik dibandingkan model awal ketika autokorelasi masih terjadi, yaitu sebesar $1.352759e + 26$.

Simpulan

Autokorelasi yang terdapat pada data GDP terjadi akibat adanya korelasi di antara unsur penyusunnya. Indikator GDP yang erat hubungannya dengan perekonomian sangat rawan menjadi penyebab adanya autokorelasi. Adanya autokorelasi menyebabkan model regresi kurang baik karena akan meingkatkan galatnya. Autokorelasi dapat dideteksi secara eksploratif melalui plot sisaan, ACF, dan PACF, serta dengan uji formal Durbin-Watson. Namun, autokorelasi tersebut dapat ditangani dengan metode Cochrane-Orcutt dan Hildreth-Lu. Pada kasus ini, metode Cochrane-Orcutt menghasilkan nilai SSE dan MSE yang lebih rendah, artinya metode Cochrane-Orcutt lebih baik untuk digunakan.