Lampiran 01. *Data IHK Kota Singaraja*

a. Data IHK

Data Indeks Harga Konsumen Kota Singaraja

Januari 2020 – Oktober 2022

NO	BULAN	TAHUN		
		2020	2021	2022
1	Januari	104.33	107.2	109.43
2	Februari	105.06	107.44	108.51
3	Maret	105.22	108.31	109.89
4	April	104.84	108.15	110.87
5	Mei	104.61	107.61	111.51
6	Juni	104.94	107.05	113.96
7	Juli	105.06	107.25	114.51
8	Agustus	104.62	107.18	112.81
9	September	104.9	106.7	113.2
10	Oktober	104.68	106.79	113.02
11	Nopember	105.07	106.92	
12	Desember	106.2	108.74	

b. Data Deret Waktu

```
> dataihk = read.csv(choose.files(), header=TRUE)
> dataihknum = dataihk[,-1:-2]
> data=ts(dataihknum, start=c(2020,1), end=c(2022,10), frequency = 12)
> data
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
2020 104.33 105.06 105.22 104.84 104.61 104.94 105.06 104.62 104.90 104.68
2021 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18 106.70 106.79
2022 109.43 108.51 109.89 110.87 111.51 113.96 114.51 112.81 113.20 113.02
        Nov    Dec
2020 105.07 106.20
2021 106.92 108.74
2022
```

Lampiran 02. *Analisis Deskriptif Data IHK Kota Singaraja*

Analisis Deskriptif Data IHK Kota Singaraja

NO	BULAN	TAHUN		
		2020	2021	2022
1	Januari	104.33	107.2	109.43
2	Februari	105.06	107.44	108.51
3	Maret	105.22	108.31	109.89
4	April	104.84	108.15	110.87
5	Mei	104.61	107.61	111.51
6	Juni	104.94	107.05	113.96
7	Juli	105.06	107.25	114.51
8	Agustus	104.62	107.18	112.81
9	September	104.9	106.7	113.2
10	Oktober	104.68	106.79	113.02
11	Nopember	105.07	106.92	
12	Desember	106.2	108.74	
	Min	104.33	106.7	108.51
	Max	106.2	108.74	114.51
	Mean	104.9608	107.445	111.771
	Sd	0.464082	0.641978	2.041456

- Menentukan Rata-rata

$$\overline{x_i} = \frac{(x_{i1} + x_{i2} + \cdots + x_{in})}{n}; \quad i = 1,2,3$$

$$\overline{x_1} = \frac{(104.33 + 105.06 + \cdots + 106.2)}{12} = 104.9608$$

$$\overline{x_2} = \frac{(107.2 + 107.44 + \cdots + 108.74)}{12} = 107.445$$

$$\overline{x_3} = \frac{(109.43 + 108.51 + \cdots + 113.02)}{10} = 111.771$$

- Menentukan Standar Deviasi

$$\sigma = \sqrt{\sum \frac{(x_i - \bar{x_i})^2}{N}}$$

$$\sigma_1 = \sqrt{\frac{(104.33 - 104.96)^2 + (105.06 - 104.96)^2 + \cdots + (106.2 - 104.96)^2}{12}} = 0.464$$

$$\sigma_2 = \sqrt{\frac{(107.2 - 107.45)^2 + (107.44 - 107.45)^2 + \cdots + (108.74 - 107.45)^2}{12}} = 0.642$$

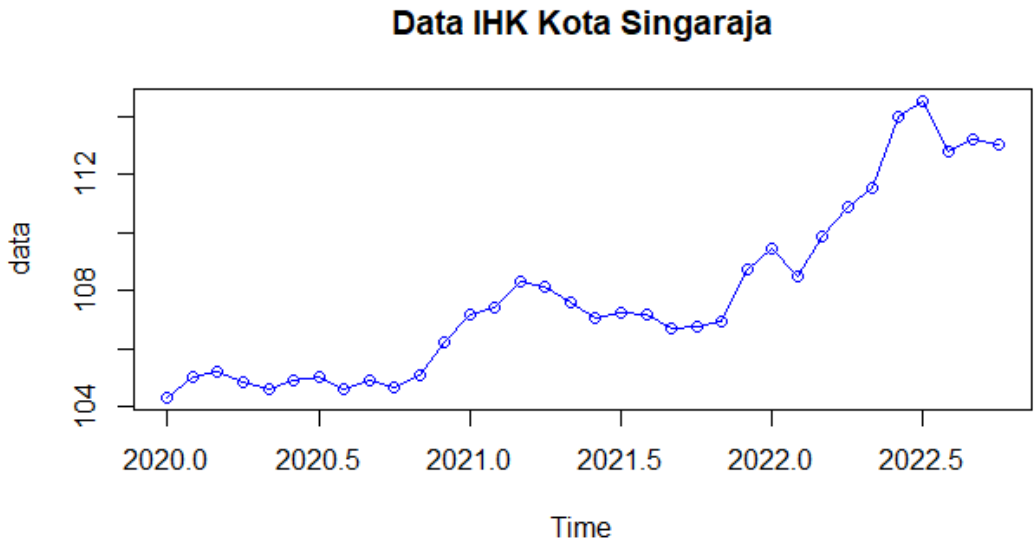
$$\sigma_3 = \sqrt{\frac{(109.43 - 111.77)^2 + (108.51 - 111.77)^2 + \cdots + (113.02 - 111.77)^2}{10}} = 2.041$$

Lampiran 03. *Plot Data IHK Kota Singaraja*

Plot Data Indeks Harga Konsumen Kota Singaraja

Untuk mengetahui kecenderungan data, maka dilakukan suatu ploting dari data yang dianalisis. Dengan bantuan *software R* maka didapatkan suatu plot data sebagai berikut.

```
> plot(data, type="o", col="blue", main=paste("Data IHK Kota Singaraja"))
```



Gambar tersebut memperlihatkan bahwa data IHK Kota Singaraja dari Januari 2020 sampai Oktober 2022 (34 bulan) tersebut memiliki pola tren cenderung naik dari tahun ke tahun dan memiliki pola musiman.

Lampiran 04. *Data In-sample dan Out-sample Penelitian*

Data *In-sample* dan *Out-sample* Penelitian

Pembagian data yang digunakan pada penelitian ini adalah 80% data *in-sample* dan 20% data *out-sample*. Sehingga data *in-sample* penelitian ini adalah dari Januari 2020 sampai April 2022 dan data *out-sample* dari Mei 2022 sampai Oktober 2022. Data *in-sample* digunakan untuk membuat matriks lintasan X, sedangkan data *out-sample* digunakan untuk memvalidasi keakuratan peramalan.

a. Data *In-sample* Penelitian

No	Tahun	Bulan	IHK
1	2020	Januari	104.33
2	2020	Februari	105.06
3	2020	Maret	105.22
4	2020	April	104.84
5	2020	Mei	104.61
6	2020	Juni	104.94
7	2020	Juli	105.06
8	2020	Agustus	104.62
9	2020	September	104.9
10	2020	Oktober	104.68
11	2020	Nopember	105.07
12	2020	Desember	106.2
13	2021	Januari	107.2
14	2021	Februari	107.44
15	2021	Maret	108.31
16	2021	April	108.15
17	2021	Mei	107.61
18	2021	Juni	107.05
19	2021	Juli	107.25
20	2021	Agustus	107.18
21	2021	September	106.7
22	2021	Oktober	106.79
23	2021	Nopember	106.92
24	2021	Desember	108.74
25	2022	Januari	109.43
26	2022	Februari	108.51
27	2022	Maret	109.89
28	2022	April	110.87

b. Data *Out-sample* Penelitian

No	Tahun	Bulan	IHK
1	2022	Mei	111.51
2	2022	Juni	113.96
3	2022	Juli	114.51
4	2022	Agustus	112.81
5	2022	September	113.2
6	2022	Oktober	113.02

Lampiran 05. *Embedding*

Tahap *Embedding*

Tahap *embedding* menentukan matriks lintasan X berdimensi $L \times K$, dengan L merupakan *window length* (jumlah baris pada matriks) dan K merupakan jumlah kolom pada matriks. Pemilihan *window length* (L), dilakukan secara coba-coba (*trial and error*) karena belum ada kriteria pemilihan L dengan tepat, tetapi rentang pemilihan nilai L adalah yang memenuhi $2 \leq L \leq \frac{N}{2}$ dan $K = N - L + 1$.

Pemilihan nilai *window length* (L) yang digunakan yaitu dengan menentukan nilai MAPE minimum. Adapun daftar nilai L beserta MAPE dapat dilihat sebagai berikut.

L	$MAPE$
7	2.855507
8	2.955306
9	2.92146
10	2.838356
11	3.13149
12	2.955972
13	$\mathbf{2.826412}$

Nilai L yang menghasilkan nilai $MAPE$ terendah berada pada $L = 13$ yaitu 2.826412. Sehingga berdasarkan rumus $K = N - L + 1$, dengan $L = 13$ dan $N = 28$, maka nilai $K = 28 - 13 + 1 = 16$. Selanjutnya dibentuk suatu matriks lintasan X dengan dimensi 13×16.

a. Dengan cara manual
   - Data *in-sample* diubah menjadi data satu dimensi.

$$\begin{bmatrix} Januari\ 2020 \\ Februari\ 2020 \\ Maret\ 2022 \\ April\ 2020 \\ Mei\ 2020 \\ Juni\ 2020 \\ Juli\ 2020 \\ \vdots \\ April\ 2022 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ \vdots \\ x_{28} \end{bmatrix} = \begin{bmatrix} 104.33 \\ 105.06 \\ 105.22 \\ 104.84 \\ 104.61 \\ 104.94 \\ 105.06 \\ \vdots \\ 110.87 \end{bmatrix}$$

   - Data satu dimensi diubah menjadi data multidimensi (Matriks Lintasan X) dengan dimensi 13×16.

$$X = [X_1, X_2, \dots, X_K] = \begin{bmatrix} x_1 & x_2 & \dots & x_K \\ x_2 & x_3 & \cdots & x_{K+1} \\ \vdots & \cdots & \ddots & \vdots \\ x_L & x_{L+1} & \cdots & x_N \end{bmatrix}$$

$$X = [X_1, X_2, \dots, X_{16}] = \begin{bmatrix} 104.33 & 105.06 & \dots & 108.15 \\ 105.06 & 105.22 & \cdots & 107.61 \\ \vdots & \cdots & \ddots & \vdots \\ 107.20 & 107.44 & \cdots & 110.87 \end{bmatrix}$$

b. Dengan bantuan *software R*

#1. Dekomposisi

```
> ###Embedding###
> z=as.matrix(insample)
> x=embed(z,L)
> id=1:L
> y=rbind(id,x)
> w=as.matrix(y)
> sort=w[,order(-w[1,])]
> THankel=as.matrix(sort[-1,(1:L)])
> Hankel=t(THankel)
> Hankel
```

```
 [1,] 104.33 105.06 105.22 104.84 104.61 104.94 105.06 104.62 104.90 104.68 105.07 106.20 107.20 107.44 108.31 108.15
 [2,] 105.06 105.22 104.84 104.61 104.94 105.06 104.62 104.90 104.68 105.07 106.20 107.20 107.44 108.31 108.15 107.61
 [3,] 105.22 104.84 104.61 104.94 105.06 104.62 104.90 104.68 105.07 106.20 107.20 107.44 108.31 108.15 107.61 107.05
 [4,] 104.84 104.61 104.94 105.06 104.62 104.90 104.68 105.07 106.20 107.20 107.44 108.31 108.15 107.61 107.05 107.25
 [5,] 104.61 104.94 105.06 104.62 104.90 104.68 105.07 106.20 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18
 [6,] 104.94 105.06 104.62 104.90 104.68 105.07 106.20 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18 106.70
 [7,] 105.06 104.62 104.90 104.68 105.07 106.20 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18 106.70 106.79
 [8,] 104.62 104.90 104.68 105.07 106.20 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18 106.70 106.79 106.92
 [9,] 104.90 104.68 105.07 106.20 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18 106.70 106.79 106.92 108.74
[10,] 104.68 105.07 106.20 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18 106.70 106.79 106.92 108.74 109.43
[11,] 105.07 106.20 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18 106.70 106.79 106.92 108.74 109.43 108.51
[12,] 106.20 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18 106.70 106.79 106.92 108.74 109.43 108.51 109.89
[13,] 107.20 107.44 108.31 108.15 107.61 107.05 107.25 107.18 106.70 106.79 106.92 108.74 109.43 108.51 109.89 110.87
```

Lampiran 06. *Singular Value Decomposition* (SVD)

Tahap *Singular Value Decomposition* (SVD

Tahap SVD dilakukan perhitungan untuk menentukan nilai *eigentriple* berdasarkan matriks lintasan $X_{13\times16}$ yang telah terbentuk.

1. Matriks Simetris (*Trajectory*)

   Langkah awal dalam menentukan *eigentriple* yaitu dengan membentuk matriks simetris $S = XX^T$. Adapun pembentukan matriks simetris dapat dilihat sebagai berikut.

   a. Dengan cara manual

$$S = X_{13\times16}(X_{13\times16})^T$$

$$S_{13\times13} = \begin{bmatrix} 104.33 & 105.06 & \dots & 108.15 \\ 105.06 & 105.22 & \cdots & 107.61 \\ \vdots & \cdots & \ddots & \vdots \\ 107.20 & 107.44 & \cdots & 110.87 \end{bmatrix} \times \begin{bmatrix} 104.33 & 105.06 & \dots & 108.15 \\ 105.06 & 105.22 & \cdots & 107.61 \\ \vdots & \cdots & \ddots & \vdots \\ 107.20 & 107.44 & \cdots & 110.87 \end{bmatrix}^T$$

$$S_{13\times13} = \begin{bmatrix} 104.33 & 105.06 & \dots & 108.15 \\ 105.06 & 105.22 & \cdots & 107.61 \\ \vdots & \cdots & \ddots & \vdots \\ 107.20 & 107.44 & \cdots & 110.87 \end{bmatrix} \times \begin{bmatrix} 104.33 & 105.06 & \dots & 107.20 \\ 105.06 & 105.22 & \cdots & 107.44 \\ \vdots & \cdots & \ddots & \vdots \\ 108.15 & 107.61 & \cdots & 110.87 \end{bmatrix}$$

$$S_{13\times13} = \begin{bmatrix} 178666.4 & 179011.6 & \dots & 182614.3 \\ 179011.6 & 179361.5 & \cdots & 182965.8 \\ \vdots & \cdots & \ddots & \vdots \\ 182614.3 & 182965.8 & \cdots & 186655.0 \end{bmatrix}$$

   b. Dengan bantuan *software R*

```
> ###Matriks S=X.X^T
> trajectory=Hankel%*%(t(Hankel))
> trajectory
```

> dim(trajectory)

```
        [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]    [,10]    [,11]    [,12]    [,13]
 [1,] 178666.4 179011.6 179218.3 179429.5 179672.5 179887.9 180080.1 180274.6 180712.1 181196.2 181602.3 182114.9 182614.3
 [2,] 179011.6 179361.5 179570.3 179781.9 180025.1 180240.5 180431.2 180624.9 181061.1 181543.6 181951.6 182465.6 182965.8
 [3,] 179218.3 179570.3 179783.6 179997.0 180241.0 180457.1 180647.4 180839.4 181274.1 181754.8 182161.9 182676.7 183176.9
 [4,] 179429.5 179781.9 179997.0 180214.9 180460.8 180677.5 180868.5 181060.2 181493.6 181972.9 182378.0 182892.2 183393.2
 [5,] 179672.5 180025.1 180241.0 180460.8 180711.1 180929.6 181121.3 181313.7 181746.6 182224.6 182628.3 183140.5 183641.2
 [6,] 179887.9 180240.5 180457.1 180677.5 180929.6 181152.7 181346.3 181539.4 181971.9 182449.1 182852.0 183362.2 183860.7
 [7,] 180080.1 180431.2 180647.4 180868.5 181121.3 181346.3 181544.4 181739.3 182172.9 182649.8 183051.8 183561.2 184057.4
 [8,] 180274.6 180624.9 180839.4 181060.2 181313.7 181539.4 181739.3 181938.7 182374.4 182852.4 183254.0 183762.6 184258.0
 [9,] 180712.1 181061.1 181274.1 181493.6 181746.6 181971.9 182172.9 182374.4 182817.7 183299.2 183700.1 184211.0 184708.0
[10,] 181196.2 181543.6 181754.8 181972.9 182224.6 182449.1 182649.8 182852.4 183299.2 183788.6 184192.5 184703.5 185203.1
[11,] 181602.3 181951.6 182161.9 182378.0 182628.3 182852.0 183051.8 183254.0 183700.1 184192.5 184605.2 185117.9 185617.0
[12,] 182114.9 182465.6 182676.7 182892.2 183140.5 183362.2 183561.2 183762.6 184211.0 184703.5 185117.9 185641.3 186143.0
[13,] 182614.3 182965.8 183176.9 183393.2 183641.2 183860.7 184057.4 184258.0 184708.0 185203.1 185617.0 186143.0 186655.0
> dim(trajectory)
[1] 13 13
```

2. *Eigen Values* dan *Eigen Vectors*

   Perhitungan ini menggunakan persamaan:

$$\det(S - \lambda I) = 0$$

- Dengan cara manual

$$\det\left(\begin{bmatrix} 178666.4 & 179011.6 & \dots & 182614.3 \\ 179011.6 & 179361.5 & \cdots & 182965.8 \\ \vdots & \cdots & \ddots & \vdots \\ 182614.3 & 182965.8 & \cdots & 186655.0 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix} 178666.4 - \lambda & 179011.6 & \dots & 182614.3 \\ 179011.6 & 179361.5 - \lambda & \cdots & 182965.8 \\ \vdots & \cdots & \ddots & \vdots \\ 182614.3 & 182965.8 & \cdots & 186655.0 - \lambda \end{bmatrix}\right) = 0$$

$$\lambda_1 = 2366732$$
$$\lambda_2 = 68.63761$$
$$\vdots$$
$$\lambda_{13} = 0.2058444$$

- Dengan bantuan *software R*

```
> ###eigen value
> egen=(eigen(trajectory)$values)
> print(as.matrix(egen), ncol=1)

              [,1]
 [1,] 2.366732e+06
 [2,] 6.863761e+01
 [3,] 5.728275e+01
 [4,] 6.278692e+00
 [5,] 4.724340e+00
 [6,] 3.970485e+00
 [7,] 3.434226e+00
 [8,] 1.438876e+00
 [9,] 1.341226e+00
[10,] 1.030682e+00
[11,] 5.384341e-01
[12,] 4.209757e-01
[13,] 2.058444e-01

> u=as.matrix(eigen(trajectory)$vectors)
> u
           [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]       [,9]       [,10]
 [1,] -0.2747484 -0.33773530 -0.05232813  0.04024696  0.159737712  0.4090326 -0.370325920 -0.15390744  0.212889929  0.40993898
 [2,] -0.2752821 -0.28389024 -0.22640344 -0.17219101  0.159203218  0.4645376  0.143669803  0.13580254 -0.350860647  0.05784840
 [3,] -0.2756060 -0.14669477 -0.34979660 -0.16387768  0.049580622  0.0666015  0.485356762 -0.20306169  0.091295702 -0.39213505
 [4,] -0.2759366  0.01783552 -0.37982515  0.07426391  0.048280990 -0.3429099  0.252481452  0.20012576  0.495883007  0.10259088
 [5,] -0.2763159  0.18410533 -0.33189303  0.11454816  0.122554938 -0.3778954 -0.155206568  0.25412566 -0.326779514  0.43930380
 [6,] -0.2766523  0.32920622 -0.22091724 -0.07858178  0.008511562 -0.1129262 -0.318476204 -0.40119802 -0.413539456 -0.28610440
 [7,] -0.2769505  0.40106521 -0.04050259 -0.09438978 -0.230551968  0.2000746 -0.247494631 -0.29520131  0.418566421 -0.03670726
 [8,] -0.2772505  0.38791661  0.15850857 -0.01060200 -0.231877268  0.3349772 -0.000190393  0.57452981  0.088063589 -0.07574180
 [9,] -0.2779199  0.24574423  0.34261335  0.34567961  0.016275792  0.1496621  0.450771528 -0.07258174 -0.275573009  0.08942522
[10,] -0.2786567  0.03851086  0.43572093  0.14990723  0.548621855 -0.1690321  0.083667121 -0.25916108  0.177788433  0.07238591
[11,] -0.2792758 -0.13184610  0.36302051 -0.63483417  0.187902291 -0.2521842 -0.165372285  0.27644885 -0.017953268 -0.20552778
[12,] -0.2800586 -0.29661742  0.22606876 -0.15631669 -0.680731818 -0.2428509  0.152352497 -0.23278372 -0.093229876  0.31132152
[13,] -0.2808204 -0.40331257  0.05939694  0.58191521 -0.149604643 -0.1163329 -0.308829074  0.17542233 -0.004868869 -0.47956402
           [,11]       [,12]       [,13]
 [1,] -0.01388494  0.09442868 -0.47953713
 [2,]  0.01381789 -0.34233352  0.48852930
 [3,]  0.05269429  0.53348562 -0.14184779
 [4,] -0.01058694 -0.52403774 -0.13744316
 [5,] -0.11637096  0.42927829  0.14964018
 [6,]  0.25328923 -0.30756764 -0.27068731
 [7,] -0.37583435  0.07439295  0.43241496
 [8,]  0.45757953  0.11356243 -0.12484653
 [9,] -0.47438178 -0.11480300 -0.26871944
[10,]  0.42302039  0.05280532  0.29366116
[11,] -0.32707850 -0.01844515 -0.13138758
[12,]  0.21791763 -0.01787930  0.06964724
[13,] -0.09973279  0.02705658  0.11652571
```

3. *Singular Value* $\sqrt{\lambda_i}$

Matriks nilai singular adalah matriks diagonal dengan diagonal utamanya berisi akar positif dari *eigenvalue* $\sqrt{\lambda_i}$ dengan urutan menurun $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L > 0$. Matriks nilai singularnya adalah sebagai berikut.

$$\sqrt{\lambda_i} = \begin{bmatrix} \sqrt{\lambda_1} & \cdots & 0_L \\ \vdots & \ddots & \vdots \\ 0_L & \cdots & \sqrt{\lambda_L} \end{bmatrix}$$

- Dengan cara manual

$$\sqrt{\lambda_i} = \begin{bmatrix} \sqrt{2366732} & \cdots & 0_L \\ \vdots & \ddots & \vdots \\ 0_L & \cdots & \sqrt{0.2058444} \end{bmatrix}$$

$$\sqrt{\lambda_i} = \begin{bmatrix} 1538.4185919 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0.4537008 \end{bmatrix}$$

- Dengan bantuan *software R*

```
> evec=sqrt(egen)
> evec
 [1] 1538.4185919    8.2847816    7.5685371    2.5057317    2.1735547
[6] 1.9926076    1.8531664    1.1995315    1.1581130    1.0152250
[11]    0.7337807    0.6488264    0.4537008
```

4. *Matrix Principal Component*

Setelah mendapatkan nilai *eigenvalue, eigenvector*, dan *singular value* maka dicari nilai *principal component* dengan rumus sebagai berikut.

$$V_i = \frac{X^T U_i}{\sqrt{\lambda_L}}$$

V_i^T merupakan transpose dari matriks V_i yaitu:

$$V_i^T = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1K} \\ v_{21} & v_{22} & \cdots & v_{2K} \\ \vdots & \cdots & \ddots & \vdots \\ v_{K1} & v_{K2} & \cdots & v_{KK} \end{bmatrix}$$

- Dengan cara manual

$$V_i = \frac{\begin{bmatrix} 104.33 & 105.06 & \ldots & 108.15 \\ 105.06 & 105.22 & \cdots & 107.61 \\ \vdots & \cdots & \ddots & \vdots \\ 107.20 & 107.44 & \cdots & 110.87 \end{bmatrix}^T \begin{bmatrix} -0.27475 & -0.33773 & \ldots & -0.47954 \\ -0.27528 & -0.28389 & \cdots & 0.48853 \\ \vdots & \cdots & \ddots & \vdots \\ -0.28082 & 0.403313 & \cdots & 0.11653 \end{bmatrix}}{\begin{bmatrix} 1538.4185919 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0.4537008 \end{bmatrix}}$$

- Dengan bantuan *software R*

```
> Vbaru<-list()
> for(i in 1:L)
+ {
+   Vbaru[[i]]<-evecs[i]*((t(Hankel))%*%(as.matrix(u[,i])))
+ }
> Vi<-do.call(cbind, Vbaru)
> Vi
```

[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
-0.2464005	-0.10621102	-0.18185831	0.15742682	-0.25235768	0.23361578	0.03258397	-0.17638068
-0.2469641	-0.20148968	-0.07389549	-0.09704926	-0.30011722	0.05264569	-0.31580610	0.07670573
-0.2475550	-0.25181014	0.07657900	0.02124847	-0.07575166	-0.34743699	-0.43293317	0.14883731
-0.2480867	-0.21407574	0.23024487	0.05414587	-0.13215846	-0.50465436	0.14280916	-0.23156535
-0.2485862	-0.10086117	0.34601746	-0.17641512	-0.04836332	-0.21104463	0.40127213	0.35995489
-0.2490236	0.06240621	0.39234132	-0.18204100	0.15866809	0.18595248	0.22878903	0.25975893
-0.2494383	0.22434534	0.34147672	0.09763735	0.11256570	0.36181247	0.02683926	-0.32062201
-0.2498184	0.35252719	0.20516734	0.16361621	-0.18998576	0.24605299	-0.15186592	0.08064765
-0.2501891	0.41463191	0.02326637	-0.06846616	-0.24255475	-0.05378976	-0.31019361	0.12630872
-0.2505262	0.39749171	-0.16648562	-0.14777640	0.09372294	-0.34093810	-0.12738800	-0.20431286
-0.2509260	0.28999052	-0.31785595	-0.04036386	0.30394067	-0.24645982	0.27775238	-0.25921902
-0.2515878	0.06553018	-0.38869100	0.30240537	0.23207549	0.01724501	0.15248262	0.56346252
-0.2521727	-0.14169571	-0.34800864	0.17890508	-0.29408412	0.15530846	0.20241964	0.06578088
-0.2524098	-0.21558882	-0.21991393	-0.60737847	-0.16247960	0.23713685	0.20375166	-0.21026314
-0.2528554	-0.28748242	-0.04324777	-0.20477228	0.63045129	0.18932460	-0.39425266	0.02328893
-0.2533233	-0.28757653	0.13607273	0.54717399	0.15017058	0.01802072	0.05884841	-0.29903856

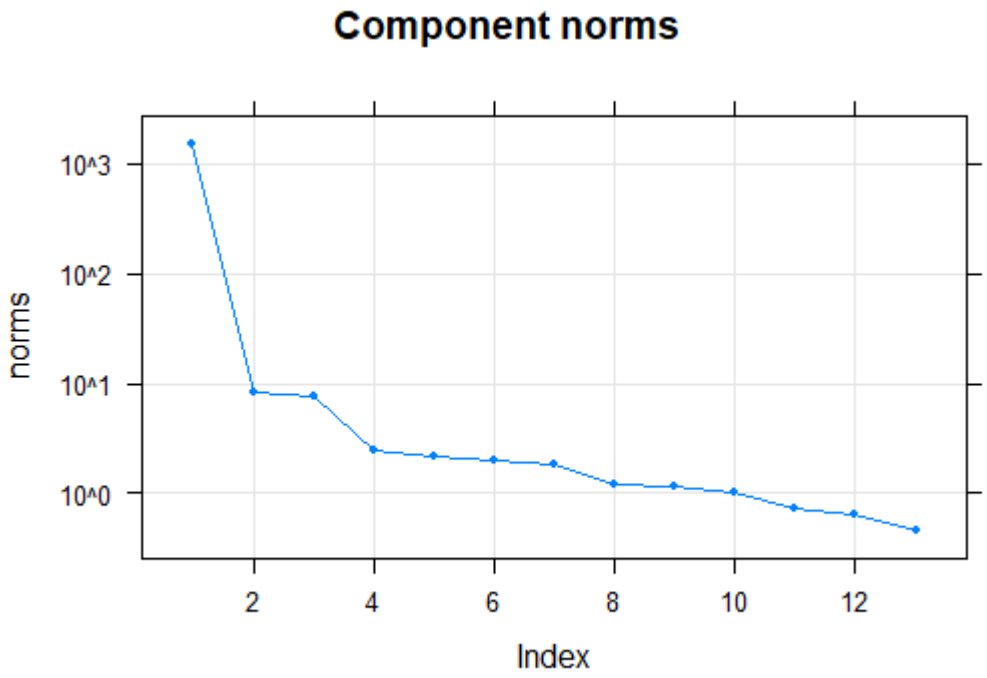
[,9]	[,10]	[,11]	[,12]	[,13]
-0.04797506	-0.48343163	-0.22410523	-0.069076465	0.465065088
-0.35120017	0.02093394	0.25790896	0.007067195	-0.134256278
0.16642144	-0.05660349	-0.31847621	0.080970663	0.360001166
0.02950615	-0.16431607	0.24776432	-0.215398139	-0.253777769
-0.26470150	-0.10935921	-0.21219508	0.362847365	-0.127346900
0.37682820	0.12290698	0.29596893	-0.241793069	0.442961424
0.14721949	-0.27949925	-0.33310064	0.166935891	-0.334119326
-0.44382615	-0.02808577	0.35530612	-0.095476443	0.068624518
0.17036749	0.45884109	-0.28198757	0.115153210	-0.103192412
0.25506715	-0.24904206	0.15745080	-0.233334382	-0.119924306
-0.31877756	0.16967000	0.02723034	0.322634945	0.343539400
-0.04833733	-0.11204158	-0.20045936	-0.381871982	-0.239886038
0.45875082	0.04534565	0.34714236	0.467650705	-0.134552617
-0.06108974	0.23363808	-0.21487352	-0.323682639	-0.103063550
-0.02708733	-0.08774553	0.17787199	0.210271911	-0.116536903
-0.04594081	0.50143506	-0.08409896	-0.172666808	-0.002474008

Lampiran 07. *Plot Eigenvalues, Eigenvectors, dan W-correlations*

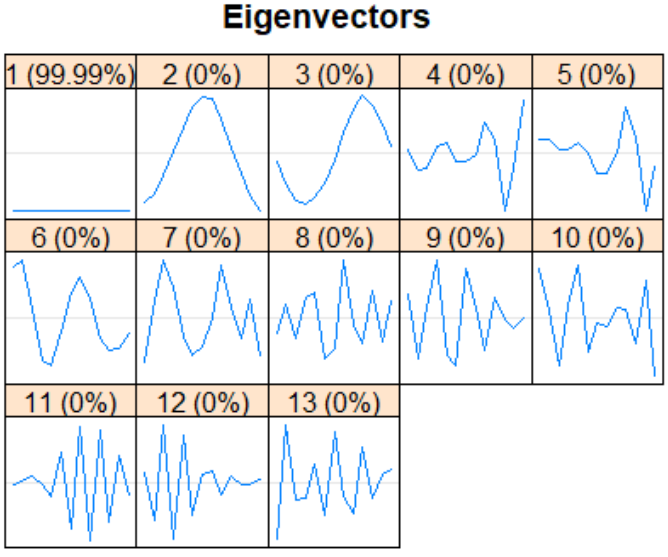
Plot *Eigenvalues, Eigenvectors, dan W-correlations*

Untuk membantu dalam menentukan *grouping*, maka dapat dilihat dari plot *Eigenvalues*, *Eigenvectors*, dan *W-correlations*.

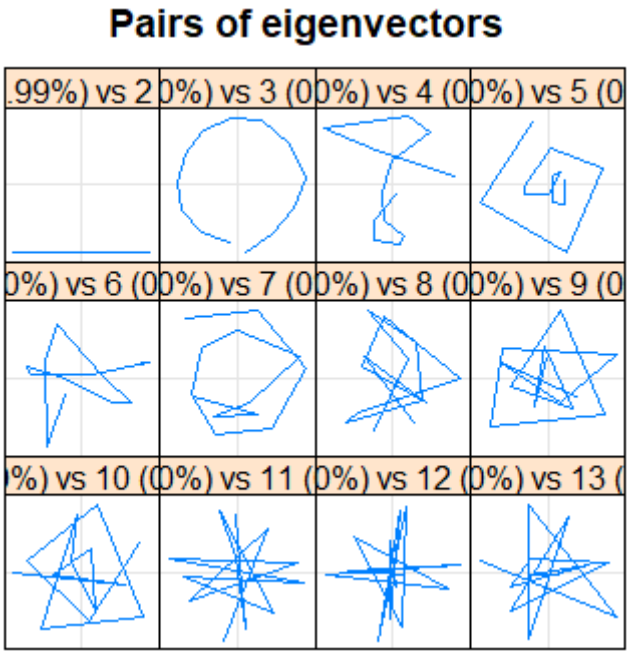
- Plot *Eigenvalues*



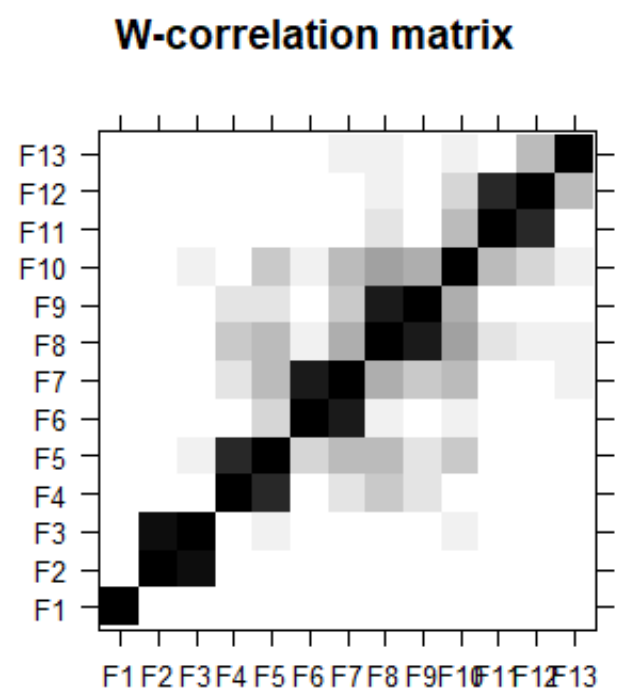
- Plot *Eigenvector* 1 Dimensi



- Plot *Eigenvector* 2 Dimensi (*Paired*)



- Plot *W-correlation*



Lampiran 08. *Periode Musiman Data*

Periode Musiman Data

> #Periode masing-masing data

> print(parestimate(S, groups=list(2:3,4:5,6:7,8:9,11:12), method="esprit"))

```
$F1
   period      rate   |    Mod     Arg   |     Re        Im
   12.569   0.012362  |  1.01244   0.50  |  0.88855   0.48529
  -12.569   0.012362  |  1.01244  -0.50  |  0.88855  -0.48529

$F2
   period      rate   |    Mod     Arg   |     Re        Im
    4.236   0.001437  |  1.00144   1.48  |  0.08765   0.99759
   -4.236   0.001437  |  1.00144  -1.48  |  0.08765  -0.99759

$F3
   period      rate   |    Mod     Arg   |     Re        Im
    5.957  -0.230670  |  0.79400   1.05  |  0.39176   0.69062
   -5.957  -0.230670  |  0.79400  -1.05  |  0.39176  -0.69062

$F4
   period      rate   |    Mod     Arg   |     Re        Im
    3.097  -0.229456  |  0.79497   2.03  | -0.35134   0.71312
   -3.097  -0.229456  |  0.79497  -2.03  | -0.35134  -0.71312

$F5
   period      rate   |    Mod     Arg   |     Re        Im
    2.082  -0.075108  |  0.92764   3.02  | -0.92056   0.11440
   -2.082  -0.075108  |  0.92764  -3.02  | -0.92056  -0.11440
```

> print(parestimate(S, groups=list(6:7), method="pairs"))

```
> print(parestimate(S, groups=list(6:7), method="pairs"))
   period      rate   |    Mod     Arg   |     Re        Im
    6.485   0.000000  |  1.00000   0.97  |  0.56623   0.82425
Warning message:
In parestimate.pairs(span(x, group), normalize = normalize.roots) :
  too big deviation of estimates, period estimates might be unreliable
```

Lampiran 09. *Grouping*

Grouping

Proses *grouping* ini dilakukan dengan cara mengelompokkan set-set indeks $i = \{1,2, \dots, d\}$ ke dalam m disjoin $I_1, I_2, \dots, I_m$. Kemudian X_i disesuaikan dengan kelompok $I = I_1, I_2, \dots, I_m$, sehingga $X_i = X_1 + X_2 + \cdots + X_d$ dapat dieksplanasi menjadi:

$$X_I = X_{I_1} + X_{I_2} + \cdots + X_{I_m}$$

Tahapan untuk memilih kelompok atau set $I = \{I_1 + I_2 + \cdots + I_m\}$ pada SVD yang didapatkan disebut *eigentriple grouping* yang dilakukan dengan cara *trial and error*. Berdasarkan plot *eigenvalue, eigenvector, w-correlation*, dan nilai periode per masing-masing data maka dipilih *grouping* sebagai berikut.

Grouping	***Eigentriple***
Trend	1
Season 1	2,3
Season 2	4,5
Season 3	8,9
Noise	6,7,10,11,12,13

Dapat dilihat pada tabel tersebut didapatkan *trend*, 3 musiman, dan *noise* yang membentuk 5 matriks dari 13 *eigentriple* yang berbeda. 5 grup tersebut merupakan matriks $X_{I1}, X_{I2}, \dots, X_{I5}$ yang masing-masing berukuran 13×16 yaitu sebagai berikut.

- Dengan bantuan *software R*

```
> Tbaru<-list()
> for(i in 1:L)
+ {
+   Tbaru[[i]]<-evec[i]*((as.matrix(u[,i]))%*%as.matrix(t(Vi[,i])))
+ }
> Tbaru.1=(Tbaru[[1]])
> Tbaru.1
```

```
 [1,] 104.1481 104.3863 104.6361 104.8608 105.0719 105.2568 105.4321 105.5927
 [2,] 104.3504 104.5891 104.8393 105.0645 105.2760 105.4613 105.6369 105.7979
 [3,] 104.4731 104.7121 104.9627 105.1881 105.3999 105.5854 105.7612 105.9223
 [4,] 104.5985 104.8378 105.0886 105.3143 105.5263 105.7120 105.8881 106.0494
 [5,] 104.7423 104.9819 105.2330 105.4590 105.6714 105.8573 106.0336 106.1952
 [6,] 104.8698 105.1097 105.3612 105.5874 105.8000 105.9862 106.1627 106.3245
 [7,] 104.9828 105.2230 105.4747 105.7013 105.9141 106.1005 106.2771 106.4391
 [8,] 105.0965 105.3370 105.5890 105.8158 106.0288 106.2154 106.3922 106.5544
 [9,] 105.3503 105.5913 105.8439 106.0712 106.2848 106.4718 106.6491 106.8116
[10,] 105.6296 105.8712 106.1245 106.3524 106.5666 106.7541 106.9319 107.0948
[11,] 105.8642 106.1064 106.3603 106.5887 106.8033 106.9913 107.1694 107.3327
[12,] 106.1610 106.4038 106.6584 106.8875 107.1027 107.2912 107.4698 107.6336
[13,] 106.4498 106.6933 106.9486 107.1782 107.3940 107.5830 107.7621 107.9264

 [1,] 105.7494 105.8919 106.0609 106.3407 106.5879 106.6881 106.8764 107.0742
 [2,] 105.9549 106.0976 106.2670 106.5472 106.7949 106.8954 107.0840 107.2822
 [3,] 106.0795 106.2224 106.3920 106.6726 106.9206 107.0211 107.2100 107.4084
 [4,] 106.2068 106.3499 106.5196 106.8006 107.0488 107.1495 107.3386 107.5373
 [5,] 106.3527 106.4961 106.6660 106.9473 107.1960 107.2968 107.4862 107.6851
 [6,] 106.4822 106.6257 106.7959 107.0776 107.3265 107.4274 107.6170 107.8162
 [7,] 106.5970 106.7407 106.9110 107.1930 107.4422 107.5432 107.7330 107.9324
 [8,] 106.7125 106.8563 107.0268 107.3091 107.5586 107.6597 107.8497 108.0493
 [9,] 106.9701 107.1142 107.2852 107.5682 107.8182 107.9196 108.1101 108.3102
[10,] 107.2537 107.3982 107.5696 107.8534 108.1041 108.2058 108.3967 108.5974
[11,] 107.4920 107.6368 107.8086 108.0930 108.3442 108.4461 108.6376 108.8386
[12,] 107.7933 107.9385 108.1108 108.3959 108.6479 108.7501 108.9421 109.1437
[13,] 108.0865 108.2322 108.4049 108.6908 108.9435 109.0459 109.2384 109.4406
```

> Tbaru.2=(Reduce('+', Tbaru[c(2,3)]))

> Tbaru.2

```
 [1,]  0.36920965  0.59304698  0.6742518  0.5078095  0.14517660 -0.33000230 -0.76297342 -1.0676493
 [2,]  0.56142697  0.60052049  0.4610282  0.1089650 -0.35569343 -0.81907111 -1.11278819 -1.1806959
 [3,]  0.61054239  0.44051180  0.1032945 -0.3493879 -0.79348338 -1.11454785 -1.17669660 -0.9716083
 [4,]  0.50709775  0.18265610 -0.2573515 -0.6935223 -1.00960713 -1.11865036 -0.94850018 -0.5377082
 [5,]  0.29481728 -0.12170527 -0.5764413 -0.9048862 -1.02301748 -0.89035323 -0.51558288  0.0223310
 [6,]  0.01439069 -0.42598852 -0.8148290 -0.9688446 -0.85363712 -0.48579584  0.04092384  0.6184393
 [7,] -0.29716353 -0.64684504 -0.8601740 -0.7818980 -0.44120523  0.08708908  0.64076263  1.1084623
 [8,] -0.55951283 -0.73619921 -0.7173984 -0.4117778  0.09096145  0.67124460  1.13066414  1.3790882
 [9,] -0.68781244 -0.60183741 -0.3140940  0.1611990  0.69190457  1.14442829  1.34223013  1.2497389
[10,] -0.63361393 -0.30797643  0.1721989  0.6909929  1.10890583  1.31376240  1.19768998  0.7890699
[11,] -0.38364591  0.01706005  0.4854597  0.8664440  1.06086707  1.00980386  0.69316292  0.1786331
[12,] -0.05015727  0.36870697  0.7498283  0.9200232  0.83989654  0.51794213  0.03296138 -0.5152607
[13,]  0.27313485  0.64002930  0.8758134  0.8188095  0.49256455 -0.03214519 -0.59610763 -1.0856865

 [1,] -1.169380883 -1.04627072 -0.6855258 -0.0294175  0.5343015  0.6903275  0.82152213  0.75076605
 [2,] -1.015069169 -0.64960778 -0.1373888  0.5119138  0.9295916  0.8838901  0.75025652  0.44320452
 [3,] -0.565512853 -0.04232314  0.4890719  0.9493984  1.0935426  0.8442233  0.46388344 -0.01074398
 [4,] -0.005616916  0.53733437  0.9565970  1.1270610  0.9794902  0.6003350  0.08184588 -0.43366438
 [5,]  0.573982799  1.02448584  1.2407511  1.0763217  0.6580539  0.2235800 -0.32985301 -0.78043975
 [6,]  1.091965910  1.36248687  1.3223822  0.8286264  0.1954164 -0.2202967 -0.71176887 -1.01185308
 [7,]  1.370580886  1.37179607  1.0609997  0.3368910 -0.3641373 -0.6489315 -0.94197126 -0.99725384
 [8,]  1.360458031  1.07773148  0.5506481 -0.2557029 -0.8728806 -0.9566862 -0.97579565 -0.76097127
 [9,]  0.904496187  0.37755735 -0.2338240 -0.8744922 -1.1908985 -1.0091801 -0.69744117 -0.23264005
[10,]  0.209017130 -0.42220996 -0.9556932 -1.2609059 -1.1928610 -0.7940101 -0.23434361  0.35698413
[11,] -0.388983984 -0.89161155 -1.1900814 -1.1395215 -0.8013891 -0.3687299  0.19519697  0.68798898
[12,] -0.979111860 -1.26165885 -1.2564804 -0.8260886 -0.2472416  0.1535154  0.63246498  0.93951539
[13,] -1.374973711 -1.40300479 -1.1118532 -0.3936949  0.3170093  0.6214973  0.94113931  1.02206694
```

> Tbaru.3=(Reduce('+', Tbaru[c(4,5)]))

> Tbaru.3

```
 [1,] -0.071742054 -0.113987526 -0.024158005 -0.040424732 -0.03458281  0.036730859  0.048929199 -0.049462420
 [2,] -0.155249147 -0.061978324 -0.035380830 -0.069093721  0.05938136  0.133449237 -0.003175162 -0.136336665
 [3,] -0.091840350  0.007509187 -0.016888803 -0.036476320  0.06723003  0.091851188 -0.027962389 -0.087660337
 [4,]  0.002812077 -0.049554167 -0.003995451 -0.003793133 -0.03790360 -0.017224334  0.029981688  0.010509228
 [5,] -0.022037262 -0.107800938 -0.014079830 -0.019663028 -0.06351891 -0.009984695  0.058009787 -0.003646117
 [6,] -0.035666809  0.013557199 -0.005585360 -0.013106561  0.03384226  0.038780172 -0.017142764 -0.035731628
 [7,]  0.089226924  0.173347585  0.032934873  0.053420557  0.06596060 -0.036455810 -0.079501364  0.056507390
 [8,]  0.123005583  0.153836642  0.037614195  0.065169163  0.02906162 -0.075132327 -0.059326689  0.091405840
 [9,]  0.127432531 -0.094679209  0.015725196  0.042224806 -0.15451824 -0.152067250  0.088553717  0.135000150
[10,] -0.241792531 -0.394332011 -0.082349283 -0.137254960 -0.12393764  0.120825713  0.170905301 -0.165091635
[11,] -0.353489523  0.031805956 -0.064738642 -0.140106751  0.26087544  0.354379582 -0.109340466 -0.337861455
[12,]  0.311728254  0.482068788  0.103759978  0.174334423  0.14065838 -0.163463322 -0.204796437  0.217017969
[13,]  0.311607850 -0.043919522  0.055615357  0.121925833 -0.24150854 -0.317032963  0.105763882  0.300351029

 [1,] -0.091119365  0.01763751  0.10145717  0.11107330 -0.08406338 -0.11766565  0.19824095  0.10732047
 [2,] -0.054392179  0.09619186  0.12259026 -0.05017049 -0.17895514  0.20583829  0.30651137 -0.18412157
 [3,]  0.001975296  0.07078211  0.04932928 -0.09916788 -0.10515675  0.23190014  0.15202769 -0.20850465
 [4,] -0.038194585 -0.01766362  0.02438482  0.08062756  0.00243004 -0.13007512  0.02805528  0.11758022
 [5,] -0.084263336 -0.01744991  0.06937816  0.14861873 -0.02698750 -0.21761529  0.10916436  0.19705611
 [6,]  0.008993973  0.03083180  0.01357085 -0.05525161 -0.04066794  0.11658984  0.05198422 -0.10496301
 [7,]  0.137741692 -0.01201478 -0.14276326 -0.18782059  0.10505683  0.22507574 -0.26749824 -0.20466819
 [8,]  0.124065948 -0.04331037 -0.15211317 -0.12499919  0.14346504  0.09802485 -0.31230621 -0.09022176
 [9,] -0.067884741 -0.12468545 -0.02421008  0.27014758  0.14456046 -0.53184724 -0.15506668  0.47926385
[10,] -0.314954506  0.05625197  0.34727521  0.39033236 -0.28348181 -0.42189829  0.67486964  0.38460584
[11,]  0.009847556  0.27334943  0.18834197 -0.38626020 -0.40469726  0.89981233  0.58322254 -0.80907081
[12,]  0.385703268 -0.08079100 -0.43390298 -0.46182917  0.36505425  0.47830876 -0.85261389 -0.43651489
[13,] -0.020959645 -0.24595249 -0.15768906  0.36547947  0.35649416 -0.83279865 -0.50358954  0.74901565
```

> Tbaru.4=(Reduce('+', Tbaru[c(8,9)]))

> Tbaru.4

```
 [1,]  0.020734561 -0.10074978  0.01355344  0.05002562 -0.13171604  0.044951352  0.095489272 -0.12431452
 [2,] -0.009238302  0.15520070 -0.04337756 -0.04971128  0.16619427 -0.110804395 -0.112049762  0.19348013
 [3,]  0.037890174 -0.05581655 -0.01865776  0.05952414 -0.11566450 -0.023429469  0.093662382 -0.06657013
 [4,] -0.069892977 -0.18327649  0.13130339 -0.03864387 -0.06560530  0.278765125  0.007578866 -0.23552425
 [5,] -0.035610382  0.15629321 -0.01761146 -0.08175500  0.20990133 -0.063426968 -0.153450630  0.19254891
 [6,]  0.107859625  0.13128409 -0.15133137  0.09730950 -0.04645589 -0.305481514  0.083792075  0.17374798
 [7,]  0.039201197 -0.19740506  0.02796856  0.09630107 -0.25577452  0.090684941  0.184897542 -0.24370110
 [8,] -0.126448539  0.01704497  0.11954662 -0.15657785  0.22107262  0.217448985 -0.205947420  0.01031491
 [9,]  0.030667411  0.10540536 -0.06607090  0.01074429  0.05313894 -0.142878398 -0.019069657  0.13462325
[10,]  0.044953772 -0.09615745 -0.01200331  0.07806245 -0.16640165 -0.003163159  0.129984727 -0.11645449
[11,] -0.057491948  0.03273844  0.04589559 -0.07740267  0.12486797  0.078303449 -0.109382153  0.03597149
[12,]  0.054430929  0.01650069 -0.05952870  0.06147452 -0.07193069 -0.113219270  0.073632336  0.02540088
[13,] -0.036844319  0.01812108  0.03038063 -0.04889342  0.07723594  0.052534851 -0.068296887  0.01947285

 [1,]  0.018685493  0.10060655 -0.03073861 -0.11594226  0.10096103  0.023756401 -0.010977926  0.04388082
 [2,] -0.048650877 -0.13692549  0.08730426  0.11142893 -0.17569144 -0.009428766  0.014800319 -0.03004579
 [3,] -0.012753057  0.07673474  0.02943570 -0.14235832  0.03248123  0.044756600 -0.008536655  0.06798213
 [4,]  0.128161415  0.09743539 -0.24529769  0.10750363  0.27924653 -0.085558309 -0.009965273 -0.09816963
 [5,] -0.025972264 -0.15881060  0.04162242  0.19005441 -0.15356105 -0.040975612  0.017350341 -0.07377026
 [6,] -0.142379353 -0.02383265  0.27741975 -0.24801617 -0.25136446  0.130446495  0.001765025  0.16591443
 [7,]  0.037858839  0.19599094 -0.06273624 -0.22295531  0.19908496  0.044841783 -0.021377195  0.08362085
 [8,]  0.104423121 -0.11479191 -0.21115645  0.38338974  0.09212088 -0.151136728  0.013287378 -0.21077278
 [9,] -0.065368834 -0.06361502  0.12430483 -0.03363075 -0.15213504  0.037802834  0.006617150  0.04069728
[10,] -0.004187316  0.11603322  0.01494789 -0.18511727  0.07400696  0.052786574 -0.012817130  0.08350351
[11,]  0.038342856 -0.07305534 -0.07933139  0.18785433  0.01227523 -0.068454999  0.008286017 -0.09820871
[12,] -0.053664064  0.02951078  0.10680074 -0.15211741 -0.06789975  0.065307975 -0.003578360  0.08846122
[13,]  0.025617811 -0.04443070 -0.05274857  0.11883894  0.01125516 -0.043900072  0.005053300 -0.06266603
```

```
> Tbaru.5=(Reduce('+', Tbaru[c(6,7,10,11,12,13)]))
> Tbaru.5
```

```
 [1,] -0.1362811577  0.295365657 -0.079739818 -0.53821359 -0.44080731 -0.06850500  0.24646568  0.268676454
 [2,]  0.3126786360 -0.062833151 -0.381613154 -0.44465038 -0.20590914  0.39514466  0.21112687  0.225693563
 [3,]  0.1902622502 -0.264339256 -0.420430277  0.07824583  0.50203824  0.08077390  0.24983315 -0.116486694
 [4,] -0.1985041442 -0.177588752 -0.018567810  0.48166550  0.20678281  0.04508121 -0.29711089 -0.216682668
 [5,] -0.3694197014  0.031358061  0.435084759  0.16726390  0.10526383 -0.21355672 -0.35256197 -0.206396615
 [6,] -0.0163629213  0.231471202  0.230571007  0.19719924 -0.25378110 -0.16371144 -0.07026046  0.119083206
 [7,]  0.2459159555  0.067927861  0.224526087 -0.38907967 -0.21305546 -0.04177028  0.17672021  0.079662211
 [8,]  0.0864167730  0.128363251 -0.348759300 -0.24256730 -0.16989862  0.17105647  0.18236740  0.274824877
 [9,]  0.0794530802 -0.320143905 -0.409466413 -0.08537866  0.32470075  0.11871361  0.24919480 -0.180968399
[10,] -0.1191267846 -0.002748288 -0.002381347  0.21575752  0.05486171  0.12447764 -0.28043472  0.007675568
[11,]  0.0003827127  0.011979372  0.373083526  0.20235221  0.06007097 -0.28374770 -0.03385332 -0.159464116
[12,] -0.2769810362 -0.071105885 -0.012484312  0.26668957  0.13869056  0.07758620 -0.32160311 -0.110729634
[13,]  0.2023392745  0.132495867  0.399629512  0.07992036 -0.11232204 -0.23636852  0.04649127  0.019502057

 [1,]  0.3923774500 -0.28391041 -0.37612924 -0.106382143  0.06092836  0.155463221  0.424780565  0.173804995
 [2,] -0.1567384800 -0.33728587 -0.13946063  0.079597350  0.07014049  0.334345103 -0.005605030  0.098748441
 [3,] -0.4332113796 -0.12764023  0.24018975  0.059549203  0.36857880  0.008006941 -0.207391022 -0.207160777
 [4,] -0.0911208925  0.23300685  0.18469855  0.194248616 -0.15999892  0.075786690 -0.388577759  0.126963141
 [5,]  0.3835085902  0.09571773  0.29222780 -0.212343345 -0.06346808 -0.211769749 -0.032832459  0.152061321
 [6,] -0.0008149598  0.31479233 -0.25926496  0.007078725 -0.17986362 -0.204159941  0.220978618 -0.165303341
 [7,]  0.1668060405 -0.14642666 -0.15650854 -0.069097616 -0.13217229  0.015796702  0.197804362 -0.024119831
 [8,] -0.1514277344 -0.16590731 -0.16419499 -0.061783882  0.25874334  0.050087756  0.215073487 -0.067369834
 [9,] -0.1313448008 -0.25350371  0.09853224  0.249817342  0.08025760  0.373605919 -0.344217381  0.142494327
[10,] -0.0935923157  0.10168096  0.20382286 -0.097667766  0.08825699 -0.122629170 -0.084454351  0.007554091
[11,]  0.0988032133  0.23447921 -0.02755160  0.034963296 -0.23043027 -0.168766756  0.005736333 -0.109320240
[12,]  0.0337853195  0.07439784  0.26277572 -0.035910317  0.04215991 -0.017242836 -0.208338843  0.154855985
[13,] -0.0161952863  0.25122812 -0.16260331 -0.041431911 -0.19823419 -0.280735563  0.208982377 -0.278995760
```

Lampiran 10. *Diagonal Averaging*

Diagonal Averaging

Diagonal Averaging adalah tahapan mengubah setiap grup hasil *grouping* menjadi data deret waktu berukuran N. Pada langkah akhir ini yaitu mengubah setiap matriks X_I dari dekomposisi yang dikelompokkan yaitu $X_I = X_{I_1} + X_{I_2} + \cdots + X_{I_m}$ menjadi suatu deret baru dengan panjang N.

- Dengan bantuan *software R*

$$X_I = X_{I_1} + X_{I_2} + \cdots + X_{I_5}$$

$$X_I = \begin{bmatrix} 104.15 & 104.38 & \dots & 107.07 \\ 104.35 & 104.59 & \cdots & 107.28 \\ \vdots & \cdots & \ddots & \vdots \\ 106.45 & 106.69 & \cdots & 109.44 \end{bmatrix} + \begin{bmatrix} 0.3692 & 0.5930 & \dots & 0.7507 \\ 0.5624 & 0.6005 & \cdots & 0.4432 \\ \vdots & \cdots & \ddots & \vdots \\ 0.2731 & 0.6400 & \cdots & 1.0221 \end{bmatrix}$$

$$+ \cdots + \begin{bmatrix} -0.136 & 0.2953 & \dots & 0.1738 \\ 0.3127 & -0.063 & \cdots & 0.0987 \\ \vdots & \cdots & \ddots & \vdots \\ 0.2023 & 0.1325 & \cdots & -0.279 \end{bmatrix}$$

```
 [1,] 104.4663 104.7646 105.2997 105.3782 105.0508 105.0085 104.8135 104.3513
 [2,] 104.7473 105.2828 105.2216 105.0547 105.1459 104.6649 104.4089 104.6743
 [3,] 105.0297 105.1043 105.0304 104.8618 104.5580 104.5392 104.6502 104.7965
 [4,] 105.0385 104.7876 104.9586 104.5783 104.4132 104.8549 104.9771 105.2867
 [5,] 104.9794 104.9086 104.6249 104.4527 104.7947 104.8936 105.4226 106.4064
 [6,] 104.9564 104.8285 104.3894 104.7028 104.9338 105.2337 106.2703 107.0809
 [7,] 104.8141 104.5521 104.6755 105.0691 105.2831 106.2418 107.0233 107.3603
 [8,] 104.5336 104.7716 105.0288 105.3126 106.3699 107.0289 107.2576 108.0352
 [9,] 104.8205 105.0001 105.4795 106.2854 106.8753 107.3213 108.0608 108.3310
[10,] 104.7991 105.0727 106.2024 106.9842 107.3851 108.1855 108.4304 107.6023
[11,] 105.0696 106.1880 106.8269 107.2376 108.2499 108.4337 107.6439 107.2095
[12,] 106.4770 107.2711 107.4525 108.0433 108.0113 107.5324 107.3716 107.3607
[13,] 106.9977 107.3075 107.9104 108.0701 107.7223 107.2864 107.2035 107.1605

 [1,] 104.5076 104.9639 105.4461 106.3064 107.1391 107.2845 107.8852 107.9762
 [2,] 104.8367 105.4073 106.3395 107.1204 107.3699 107.9757 108.1556 107.5113
 [3,] 105.5032 106.3276 106.9598 107.3805 107.9414 108.1420 107.8174 107.2572
 [4,] 106.2911 106.9670 107.2553 108.1158 108.3100 107.5342 107.4386 107.1230
 [5,] 106.8165 107.3443 108.0178 108.3623 107.6735 107.2618 107.2828 107.0279
 [6,] 107.4408 107.9952 108.4093 107.6029 107.2299 107.4542 106.9590 106.8653
 [7,] 108.1432 108.2964 107.7665 107.1191 107.3822 107.1642 106.5022 106.8141
 [8,] 108.3014 107.7759 107.2142 107.3118 106.9213 106.6499 106.5749 106.9874
 [9,] 107.7413 107.3035 107.1515 106.9302 106.6197 106.4164 107.2642 108.5975
[10,] 107.1436 107.1483 106.9762 106.7977 106.7017 107.0426 108.8245 109.4224
[11,] 107.1512 106.9455 106.7276 106.7550 107.1504 108.9088 109.4243 108.6193
[12,] 107.1462 106.6256 106.5272 106.9559 108.6978 109.4472 108.7183 109.7351
[13,] 106.7162 106.5388 107.0826 108.7814 109.6282 108.7907 109.6810 111.1490
```

- Dengan bantuan *software R*

```
> recon1<-reconstruct(S, groups=list(c(1), c(2,3), c(4,5), c(8,9)), len=p)
```

> recon1

> komponen=cbind(recon1$F1, recon1$F2, recon1$F3, recon1$F4)

> komponen

```
          [,1]        [,2]         [,3]         [,4]
 [1,] 104.1481  0.36920965 -0.071742054  0.020734561
 [2,] 104.3684  0.57723697 -0.134618337 -0.054994042
 [3,] 104.5661  0.62843822 -0.059325560  0.068881435
 [4,] 104.7527  0.47911181 -0.016371075 -0.029765367
 [5,] 104.9358  0.16698189 -0.038431352 -0.083794393
 [6,] 105.1102 -0.23329163 -0.014637882  0.111020998
 [7,] 105.2762 -0.62409193  0.047788518 -0.002392810
 [8,] 105.4351 -0.90383320  0.034051848 -0.110292377
 [9,] 105.6046 -0.99127461 -0.003896020  0.107498321
[10,] 105.7826 -0.84601865 -0.031601274  0.024928838
[11,] 105.9651 -0.48167571 -0.026731186 -0.136995111
[12,] 106.1656  0.02723831  0.045123145  0.070855589
[13,] 106.3768  0.54517306  0.004612918  0.080472049
[14,] 106.5741  0.95316372 -0.056842329 -0.132193835
[15,] 106.7676  1.13781685  0.108393246  0.029233229
[16,] 106.9569  1.04397176  0.141944002  0.105589215
[17,] 107.1335  0.71083917 -0.098147537 -0.085869578
[18,] 107.3040  0.19487828 -0.188908834 -0.052190785
[19,] 107.4773 -0.37600405  0.068806335  0.086788682
[20,] 107.6554 -0.86673511  0.243823070 -0.006774664
[21,] 107.8393 -1.14862860  0.043380968 -0.046006826
[22,] 108.0345 -1.13915676 -0.342631302  0.065563198
[23,] 108.2517 -0.83195891 -0.281900370 -0.057326635
[24,] 108.4984 -0.29533002  0.556895911  0.002072869
[25,] 108.7321  0.25567645  0.450657825  0.042088164
[26,] 108.9422  0.64731707 -0.831494449 -0.048562381
[27,] 109.1910  0.94032735 -0.470052218  0.046757260
[28,] 109.4406  1.02206694  0.749015654 -0.062666025
```

> diagonal.averaging=rowSums(komponen)

> diagonal.averaging

```
 [1] 104.4663 104.7560 105.2041 105.1857 104.9806 104.9733
 [7] 104.6975 104.4551 104.7169 104.9299 105.3197 106.3088
[13] 107.0070 107.3383 108.0430 108.2484 107.6603 107.2577
[19] 107.2569 107.0257 106.6880 106.6183 107.0805 108.7620
[25] 109.4805 108.7095 109.7081 111.1490
```

Lampiran 11. *Plot Komponen Grouping yang Direkonstruksi*

Plot Rekonstruksi

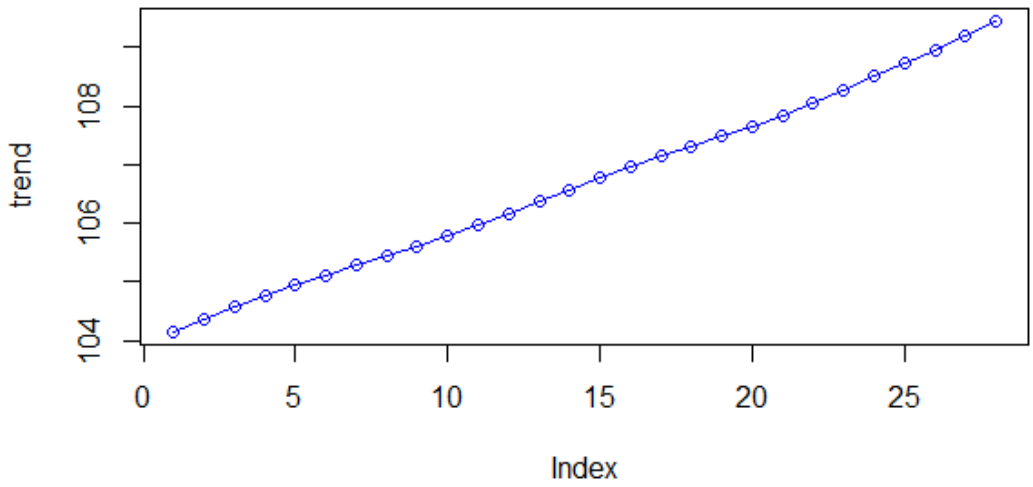
- Plot *trend* yang direkonstruksi

  > recon1<-reconstruct(S, groups=list(c(1), c(2,3), c(4:5),c(8:9)), len=p)

  > recon1

  > trend=recon1$F1

  > plot(trend, type="o", col="blue")

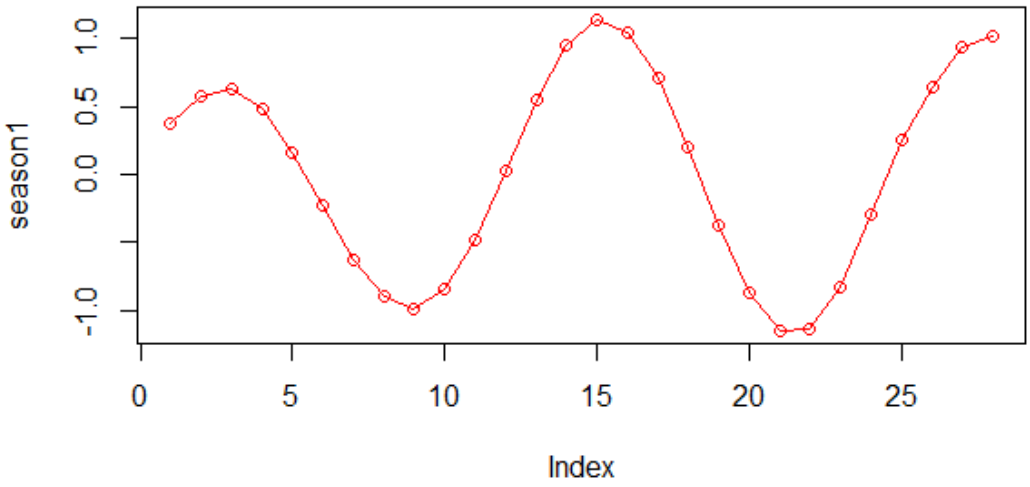


Gambar tersebut merupakan komponen *trend* yang direkonstruksi oleh *eigentriple* 1.

- Plot *season 1* yang direkonstruksi

  > season1=recon1$F2

  > plot(season1,type="o", col="red")

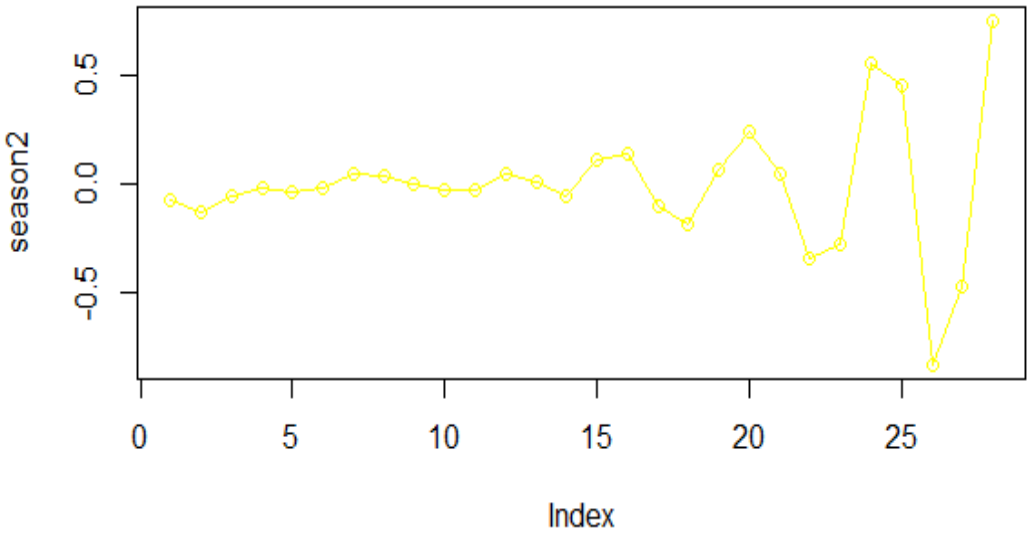


Gambar tersebut merupakan komponen *season* 1 yang direkonstruksi oleh *eigentriple* 2 dan 3.

- Plot *season 2* yang direkonstruksi

  > season2=recon1$F3

  > plot(season2,type="o", col="yellow")

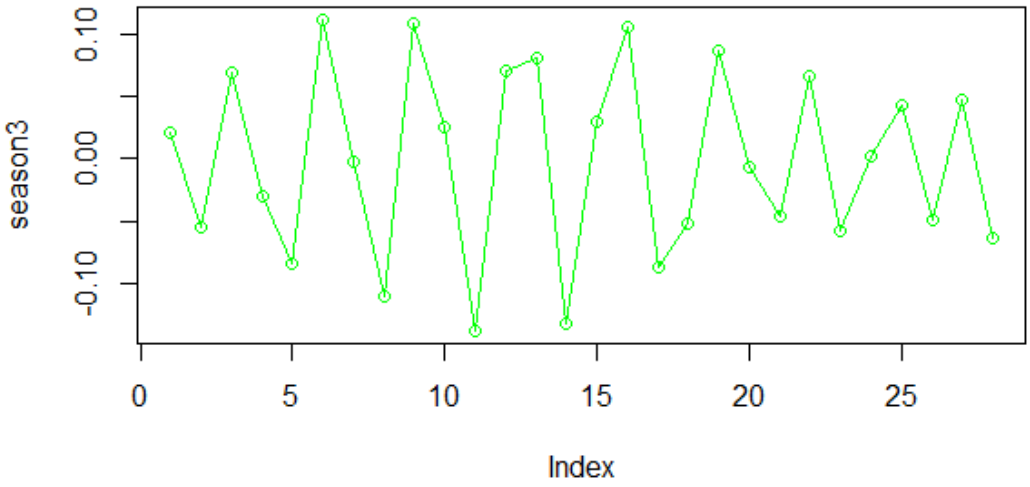


Gambar tersebut merupakan komponen *season* 2 yang direkonstruksi oleh *eigentriple* 4 dan 5.

- Plot *season 3* yang direkonstruksi

  > season3=recon1$F4

  > plot(season3, type="o", col="green")

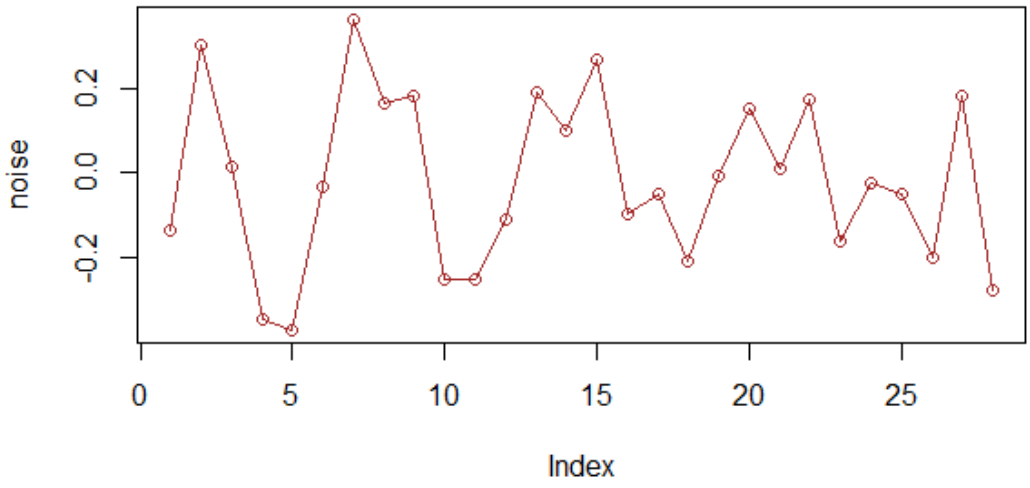


Gambar tersebut merupakan komponen *season* 3 yang direkonstruksi oleh *eigentriple* 8 dan 9.

- Plot *noise* yang direkonstruksi

  > noise=residuals(recon1)

  > plot(noise, type="o", col="brown")



Gambar tersebut merupakan komponen *noise* yang direkonstruksi oleh *eigentriple* 6, 7, 10, 11, 12, dan 13.

Lampiran 12. *Hasil Peramalan Data Out-sample*

Peramalan Data *Out-sample*

Linear Recurrence Formula (LRF) yaitu rumus berulang linier yang digunakan untuk peramalan dengan metode SSA. Metode SSA dengan LRF ini juga dikenal dengan *R-forecasting* dengan deret waktu yang digunakan adalah deret hasil rekonstruksi yang diperoleh dari hasil *diagonal averaging*.

- Dengan bantuan *software R*

```
> #Linear Recurrence Formula#
> S<-ssa(insample, L=L, kind="1d-ssa")
> lrr.coeff = lrr(S, groups=list(c(1,2:3,4:5,8:9)))
> lrr.coeff
 [1]  0.50047792  0.21992908 -0.06351053  0.32058417  0.21512651 -0.50550316
 [7] -0.44236884  0.16439038  0.51388120  0.13060739 -0.53913152  0.50172349
attr(,"class")
[1] "lrr"
```

Setelah mendapatkan nilai LRF, selanjutnya menentukan hasil peramalan dari *out-sample*.

```
> ###Recurrence Forecasting###
> ###CARA 1
> diagonal.averaging[29]=(lrr.coeff[12]*diagonal.averaging[28])+
>  (lrr.coeff[11]*diagonal.averaging[27])+(lrr.coeff[10]*diagonal.averaging[26])+
>  (lrr.coeff[9]*diagonal.averaging[25])+(lrr.coeff[8]*diagonal.averaging[24])+
>  (lrr.coeff[7]*diagonal.averaging[23])+(lrr.coeff[6]*diagonal.averaging[22])+
>  (lrr.coeff[5]*diagonal.averaging[21])+(lrr.coeff[4]*diagonal.averaging[20])+
>  (lrr.coeff[3]*diagonal.averaging[19])+(lrr.coeff[2]*diagonal.averaging[18])+
>  (lrr.coeff[1]*diagonal.averaging[17])
> diagonal.averaging[29]
> diagonal.averaging[30]=(lrr.coeff[12]*diagonal.averaging[29])+
>  (lrr.coeff[11]*diagonal.averaging[28])+(lrr.coeff[10]*diagonal.averaging[27])+
>  (lrr.coeff[9]*diagonal.averaging[26])+(lrr.coeff[8]*diagonal.averaging[25])+
>  (lrr.coeff[7]*diagonal.averaging[24])+(lrr.coeff[6]*diagonal.averaging[23])+
>  (lrr.coeff[5]*diagonal.averaging[22])+(lrr.coeff[4]*diagonal.averaging[21])+
>  (lrr.coeff[3]*diagonal.averaging[20])+(lrr.coeff[2]*diagonal.averaging[19])+
```

```
>  (lrr.coeff[1]*diagonal.averaging[18])
> diagonal.averaging[30]
> diagonal.averaging[31]=(lrr.coeff[12]*diagonal.averaging[30])+
>  (lrr.coeff[11]*diagonal.averaging[29])+(lrr.coeff[10]*diagonal.averaging[28])+
>  (lrr.coeff[9]*diagonal.averaging[27])+(lrr.coeff[8]*diagonal.averaging[26])+
>  (lrr.coeff[7]*diagonal.averaging[25])+(lrr.coeff[6]*diagonal.averaging[23])+
>  (lrr.coeff[5]*diagonal.averaging[23])+(lrr.coeff[4]*diagonal.averaging[22])+
>  (lrr.coeff[3]*diagonal.averaging[21])+(lrr.coeff[2]*diagonal.averaging[20])+
>  (lrr.coeff[1]*diagonal.averaging[19])
> diagonal.averaging[31]
> diagonal.averaging[32]=(lrr.coeff[12]*diagonal.averaging[31])+
>  (lrr.coeff[11]*diagonal.averaging[30])+(lrr.coeff[10]*diagonal.averaging[29])+
>  (lrr.coeff[9]*diagonal.averaging[28])+(lrr.coeff[8]*diagonal.averaging[27])+
>  (lrr.coeff[7]*diagonal.averaging[26])+(lrr.coeff[6]*diagonal.averaging[25])+
>  (lrr.coeff[5]*diagonal.averaging[24])+(lrr.coeff[4]*diagonal.averaging[23])+
>  (lrr.coeff[3]*diagonal.averaging[22])+(lrr.coeff[2]*diagonal.averaging[21])+
>  (lrr.coeff[1]*diagonal.averaging[20])
> diagonal.averaging[32]
> diagonal.averaging[33]=(lrr.coeff[12]*diagonal.averaging[32])+
>  (lrr.coeff[11]*diagonal.averaging[31])+(lrr.coeff[10]*diagonal.averaging[30])+
>  (lrr.coeff[9]*diagonal.averaging[29])+(lrr.coeff[8]*diagonal.averaging[28])+
>  (lrr.coeff[7]*diagonal.averaging[27])+(lrr.coeff[6]*diagonal.averaging[26])+
>  (lrr.coeff[5]*diagonal.averaging[25])+(lrr.coeff[4]*diagonal.averaging[24])+
>  (lrr.coeff[3]*diagonal.averaging[23])+(lrr.coeff[2]*diagonal.averaging[22])+
>  (lrr.coeff[1]*diagonal.averaging[21])
> diagonal.averaging[33]
> diagonal.averaging[34]=(lrr.coeff[12]*diagonal.averaging[33])+
>  (lrr.coeff[11]*diagonal.averaging[32])+(lrr.coeff[10]*diagonal.averaging[32])+
>  (lrr.coeff[9]*diagonal.averaging[30])+(lrr.coeff[8]*diagonal.averaging[29])+
>  (lrr.coeff[7]*diagonal.averaging[28])+(lrr.coeff[6]*diagonal.averaging[27])+
>  (lrr.coeff[5]*diagonal.averaging[26])+(lrr.coeff[4]*diagonal.averaging[25])+
>  (lrr.coeff[3]*diagonal.averaging[24])+(lrr.coeff[2]*diagonal.averaging[23])+
```

```
>  (lrr.coeff[1]*diagonal.averaging[22])
> diagonal.averaging[34]
> hasil.forecasting=diagonal.averaging[29:34]
> as.matrix(hasil.forecasting)
         [,1]
[1,] 111.1478
[2,] 109.5957
[3,] 109.3147
[4,] 110.4218
[5,] 110.3915
[6,] 108.8733
```

```
> ###CARA 2
> forecast=rforecast(S, groups=list(c(1),c(2:3),c(4:5),c(8:9)), len=p)
> forecast
> hasil.forecast=as.matrix(forecast$F1+forecast$F2+forecast$F3+forecast$F4)
> hasil.forecast
         [,1]
[1,] 111.1478
[2,] 109.5957
[3,] 109.3147
[4,] 110.4218
[5,] 110.3915
[6,] 108.8733
```

Lampiran 13. *Akurasi Peramalan Data Out-sample*

Akurasi Peramalan Data *Out-sample*

Berikut ditampilkan perbandingan hasil peramalan data dengan data *out-sample*.

No	*Out-sample*	Hasil Peramalan	*Residual*	*Error*
29	111.51	111.1478	0.362185	0.003248
30	113.96	109.5957	4.364305	0.038297
31	114.51	109.3147	5.195272	0.04537
32	112.81	110.4218	2.388220	0.02117
33	113.20	110.3915	2.808509	0.02481
34	113.02	108.8733	4.146687	0.03669

1. *Mean Absolute Persentage Error* (MAPE)

Tingkat akurasi peramalan dapat diukur dari nilai *Mean Absolute Persentage Error* (MAPE). MAPE merupakan rata-rata persentase kesalahan pertama dari beberapa periode. Tingkat keakuratan ini dapat dijelaskan dengan membandingkan nilai yang diproyeksikan dengan nilai aktual. Semakin kecil nilai akurasinya, maka semakin akurat sebuah model untuk melakukan prediksi.

Adapun rumus dari MAPE ini adalah sebagai berikut.

$$MAPE = \frac{\sum_{t=1}^{n} \left| \frac{Y_t - \hat{Y}_t}{Y_t} \right|}{n} . 100\%$$

Keterangan:

Y_t : Nilai observasi

$\hat{Y}_t$: Nilai peramalan

- Dengan cara manual

$$MAPE = \frac{\sum_{t=1}^{n} \left| \frac{Y_t - \hat{Y}_t}{Y_t} \right|}{n} . 100\%$$

$$MAPE$$

$$= \frac{\left|\frac{111.51 - 111.1478}{111.51}\right| + \left|\frac{113.96 - 109.5957}{113.96}\right| + \cdots + \left|\frac{113.02 - 108.8733}{113.02}\right| +}{6} . 100\%$$

$$MAPE = \frac{\left|\frac{0.362185}{111.51}\right| + \left|\frac{4.364305}{113.96}\right| + \cdots + \left|\frac{4.146687}{113.02}\right|}{\mathbf{6}} . 100\%$$

$$MAPE = \frac{0.3248005 + 3.8296812 + \cdots + 3.6689848}{6}$$

$$MAPE = 2.826412$$

- Dengan bantuan *software R*

```
> ##Akurasi Peramalan##
> residu=aktual-ramalan
> residu
[1] 0.362185 4.364305 5.195272 2.388220
[5] 2.808509 4.146687
> MAD=(sum(abs(residu))/(length(aktual)))
> MAD
[1] 3.210863
> MSE=(sum(residu^2)/(length(aktual)))
> MSE
[1] 12.82592
> PEI=(residu/aktual)*100
> MAPE=(sum(abs(PEI)))/(length(aktual))
> MAPE
[1] 2.826412
```

Lampiran 14. *Hasil Peramalan IHK Kota Singaraja*

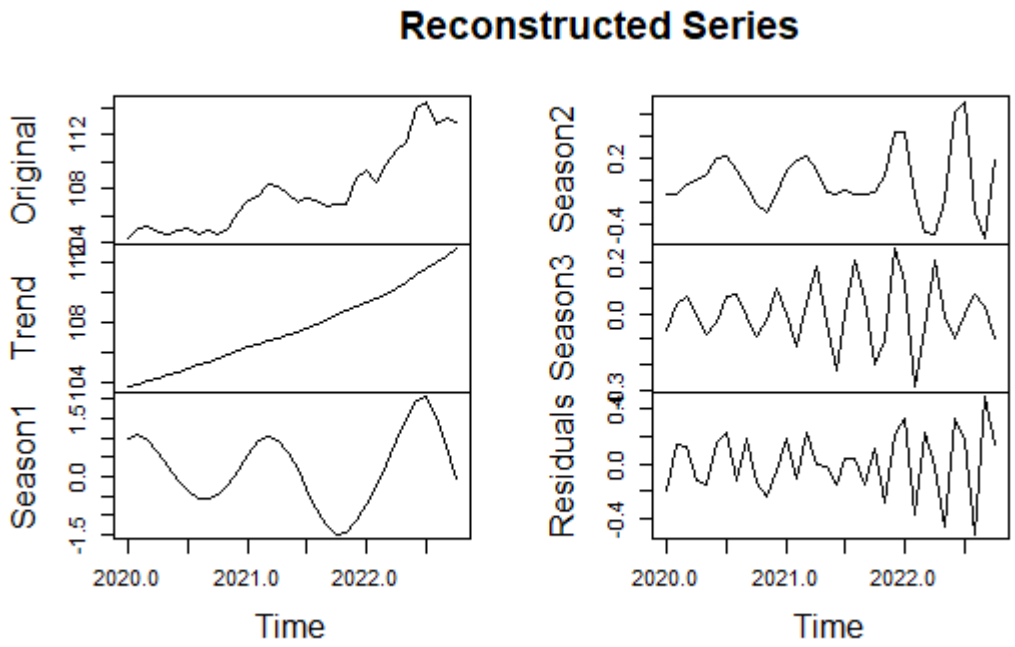
Peramalan IHK Kota Singaraja

Peramalan IHK Kota Singaraja untuk 6 bulan kedepan menggunakan keseluruhan data IHK, yaitu *in-sample* dan *out-sample*. Adapun hasil peramalan dapat dilihat sebagai berikut.

- Hasil Rekonstruksi Data

	r.complete$Trend	r.complete$Season1	r.complete$Season2	r.complete$Season3
Jan 2020	103.7439	0.96962152	-0.123095307	-0.066608335
Feb 2020	103.9200	1.08007252	-0.113962204	0.033338859
Mar 2020	104.1173	0.95189308	-0.037897577	0.064779556
Apr 2020	104.3198	0.63684639	0.005501726	-0.002215637
May 2020	104.5163	0.26726124	0.054697696	-0.082719086
Jun 2020	104.7175	-0.09446880	0.197579249	-0.036944587
Jul 2020	104.9244	-0.38503214	0.230498185	0.067174651
Aug 2020	105.1359	-0.55391382	0.094219862	0.071684154
Sep 2020	105.3647	-0.58937521	-0.049215330	-0.008281944
Oct 2020	105.6052	-0.48477642	-0.211481697	-0.089726872
Nov 2020	105.8505	-0.22205759	-0.289287024	-0.023883740
Dec 2020	106.1087	0.16171355	-0.126310706	0.100874733
Jan 2021	106.3699	0.56038414	0.091529784	-0.005255349
Feb 2021	106.5673	0.90970024	0.193601945	-0.131003661
Mar 2021	106.7608	1.04845632	0.227503409	0.043490219
Apr 2021	106.9502	0.94053528	0.086220512	0.184035711
May 2021	107.1420	0.62482878	-0.108289357	-0.035884265
Jun 2021	107.3552	0.18622582	-0.122689659	-0.220715471
Jul 2021	107.5981	-0.30004258	-0.083220719	-0.004211300
Aug 2021	107.8603	-0.80074101	-0.116409971	0.204898018
Sep 2021	108.1455	-1.23187610	-0.120347007	0.056236118
Oct 2021	108.4557	-1.48141376	-0.103150524	-0.196606660
Nov 2021	108.7072	-1.44426479	0.045825571	-0.108986570
Dec 2021	108.9786	-1.14661855	0.442305388	0.252132837
Jan 2022	109.2589	-0.72339666	0.432475768	0.124769289
Feb 2022	109.5521	-0.25691792	-0.132261974	-0.285938493
Mar 2022	109.8854	0.30749690	-0.468658538	-0.056175282
Apr 2022	110.2518	0.90907207	-0.488806884	0.211124084
May 2022	110.6611	1.49394291	-0.181586346	-0.008333182
Jun 2022	111.1214	1.97694934	0.628654939	-0.099212239
Jul 2022	111.5770	2.05008159	0.717429785	-0.002758411
Aug 2022	111.9900	1.54419389	-0.287302548	0.074635112
Sep 2022	112.4089	0.79397666	-0.519808589	0.030103113
Oct 2022	112.8148	-0.03200652	0.190135502	-0.100067621

- Grafik Hasil Rekonstruksi Data



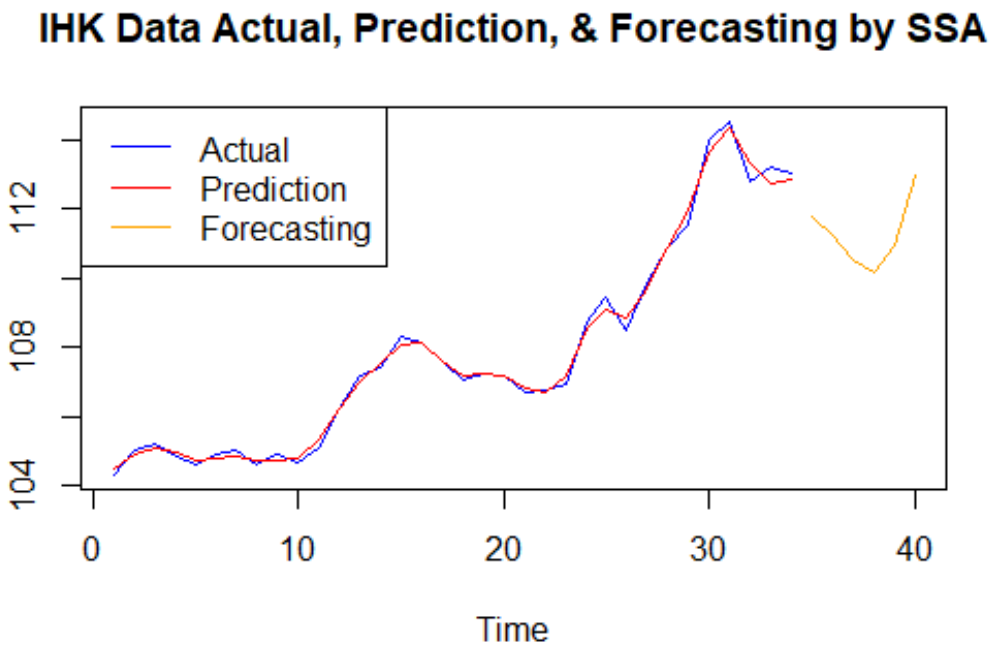
- Diagonal Averaging

```
 [1] 104.5239 104.9194 105.0961 104.9599 104.7555 104.7837 104.8371 104.7479 104.7178
[10] 104.8192 105.3153 106.2450 107.0166 107.5396 108.0803 108.1610 107.6227 107.1980
[19] 107.2106 107.1480 106.8495 106.6746 107.1997 108.5264 109.0928 108.8770 109.6680
[28] 110.8832 111.9651 113.6278 114.3418 113.3215 112.7132 112.8729
```

- Hasil Peramalan IHK Kota Singaraja Selama 6 Bulan

Tahun	Bulan	Hasil Peramalan
2022	November	111.7449
2022	Desember	111.2046
2023	Januari	110.5413
2023	Februari	110.1675
2023	Maret	110.9427
2023	April	112.9699

- Grafik Peramalan



Lampiran 15. *Script Analisis Data dengan Software R*

1. *Script* Analisis Data Musiman

```
#Program untuk mengidentifikasi musiman
#panggil data
dataihk = read.csv(choose.files(), header=TRUE)
dataihk
dataihknum = dataihk[,-1:-2]
data=ts(dataihknum, start=c(2020,1), end=c(2022,10), frequency = 12)
data

x<-diff(diff(diff(data)))
x
n<-length(x)
n
k<-round((n-1)/2)
k
omega<-c()
omega
t<-c(1:n)
t
md=matrix(0,nrow=k,ncol=n)
md
mt=matrix(0,nrow=k,ncol=n)
mt
mdata=matrix(0,nrow=k,ncol=n)
mdata
cosomega=matrix(0,nrow=k,ncol=n)
cosomega
md1=matrix(0,nrow=k,ncol=n)
md1
```

```
mt1=matrix(0,nrow=k,ncol=n)
mt1
mdata1=matrix(0,nrow=k,ncol=n)
mdata1
sinomega=matrix(0,nrow=k,ncol=n)
sinomega
for(i in 1:k)
{
  omega[i]<-(2*pi*i)/n
  md[i,]<-omega[i]
  mt[i,]<-t
  mdata[i,]<-x[t]
}
cosomega<-mdata*cos(md*mt)
cosomega
sinomega<-mdata*sin(md*mt)
sinomega
a<-c()
b<-c()
periodogram<-c()
for (i in 1:k)
{
  a[i]<-(2/n)*sum(cosomega[i,])
  b[i]<-(2/n)*sum(sinomega[i,])
  periodogram[i]<-(a[i]^2+b[i]^2)
}
a
b
periodogram
max(periodogram)
k<-which.max(periodogram)
k
```

```
omega<-(2*pi*k)/n
omega
Periode<-(2*pi)/omega
Periode
Thitung<-(max(periodogram))/sum(periodogram)
Thitung
Nbintang<-c(seq(5,50,by=5))
Nbintang
galpha<-c(0.68377, 0.444, 0.33462, 0.27040, 0.22805, 0.19784, 0.17513, 0.15738, 0.14310, 0.13135)
Tabel<-cbind(Nbintang, galpha)
Tabel
Ttabel<-Tabel[3,2]
Ttabel
Thitung>Ttabel

plot(periodogram, type="h", col="blue")
```

2. *Script* Analisis Peramalan Data

```
#Script Peramalan Data IHK Kota Singaraja
rm(list=ls())     #remove variables
graphics.off()    #close figures

#install package
install.packages("Rssa")
library(Rssa)
library(tseries)

#panggil data
dataihk = read.csv(choose.files(), header=TRUE)
dataihk
dataihknum = dataihk[,-1:-2]
data=ts(dataihknum, start=c(2020,1), end=c(2022,10), frequency = 12)
data
plot(data, type="o", col="blue", main=paste("Data IHK Kota Singaraja"))

n<-length(data)
p<-6           #20% dari total series = data outsample
insample=data[1:(n-p)]
insample
n1<-length(insample)
n1

outsample=data[(n-p+1):n]
head(outsample)

#parameter window length (L)
L=13           #trial and error (L=< N/2)
K=n1-L+1
```

K

```
#1. Dekomposisi
###Embedding###
z=as.matrix(insample)
z
x=embed(z,L)
x
id=1:L
id
y=rbind(id,x)
y
w=as.matrix(y)
w
sort=w[,order(-w[1,])]
sort
THankel=as.matrix(sort[-1,(1:L)])
THankel
Hankel=t(THankel)
Hankel
t(Hankel)

#SVD (Singular Value Decomposition)##
###Matriks S=X.X^T
trajectory=Hankel%*%(t(Hankel))
trajectory
dim(trajectory)
###eigen value
eigen(trajectory)
egen=(eigen(trajectory)$values)
egen
print(as.matrix(egen), ncol=1)
```

```
#kumulatif varian data yang bisa dijelaskan dari eigenvalue
total=sum(egen)
total
propv=egen/total
propv
kumv=cumsum(propv)
kumv

#eigen vektor ##untuk penentuan grouping
u=as.matrix(eigen(trajectory)$vectors)
u

###matriks singular value (evec) dan matriks prinsipal component (Vi baru)
evec=sqrt(egen)
evec
print(as.matrix(sqrt(egen), ncol=1))
evecs=1/sqrt(egen)
evecs

Vbaru<-list()
for(i in 1:L)
{
  Vbaru[[i]]<-evecs[i]*((t(Hankel))%*%(as.matrix(u[,i])))
}
Vi<-do.call(cbind, Vbaru)
Vi


###Hasil SVD=hankel (gabungan dari semua eigentriple dengan i=1,2,3,...,d)
T<-list()
for(i in 1:L)
```

```
{
  T[[i]]<-((as.matrix(u[,i])%*%(as.matrix(evec[i])))%*%as.matrix(t(Vi[,i])))
}
T1=(T[[1]])
T1
T2=(T[[2]])
T2
T3=(T[[3]])
T3
T4=(T[[4]])
T4
T5=(T[[5]])
T5
T6=(T[[6]])
T6
T7=(T[[7]])
T7
T8=(T[[8]])
T8
T9=(T[[9]])
T9
T10=(T[[10]])
T10
T11=(T[[11]])
T11
T12=(T[[12]])
T12
T13=(T[[13]])
T13
T=as.matrix(T1+T2+T3+T4+T5+T6+T7+T8+T9+T10+T11+T12+T13)
T
```

```
#Rekonstruksi
##Grouping
#plot tree eigenvalues
S<-ssa(insample,L=L, kind="1d-ssa")
S
plot(S)

#plot principal component
opar=par(no.readonly = TRUE)
par(mfrow=c(3,5))
plot(Vi[,1], type="l", col="blue")
plot(Vi[,2], type="l", col="red")
plot(Vi[,3], type="l", col="red")
plot(Vi[,4], type="l", col="yellow")
plot(Vi[,5], type="l", col="yellow")
plot(Vi[,6], type="l", col="orange")
plot(Vi[,7], type="l", col="orange")
plot(Vi[,8], type="l", col="pink")
plot(Vi[,9], type="l", col="pink")
plot(Vi[,10], type="l", col="black")
plot(Vi[,11], type="l", col="purple")
plot(Vi[,12], type="l", col="purple")
plot(Vi[,13], type="l", col="brown")


#plot eigen vector 1D
plot(S,type="vectors",plot.method="matplot", idx=1:L)

#plot eigen vector 2D (pairs)
plot(S,type="paired", idx=1:(L-1))

#plot W corr matrix
```

```
plot(wcor(S))

#Periode masing-masing data
print(parestimate(S, groups=list(2:3,4:5,6:7,8:9,11:12), method="esprit"))
print(parestimate(S, groups=list(8:9), method="pairs"))

#plot W corr matrix hasil grouping
plot(wcor(S, groups=list(c(1), c(2,3), c(4,5), c(8,9))))

###Tahap rekonstruksi grouping yaitu dengan eigentriple grouping dari SVD
Tbaru<-list()
for(i in 1:L)
{
  Tbaru[[i]]<-evec[i]*((as.matrix(u[,i]))%*%as.matrix(t(Vi[,i])))
}
Tbaru.1=(Tbaru[[1]])
Tbaru.1
Tbaru.2=(Reduce('+', Tbaru[c(2,3)]))
Tbaru.2
Tbaru.3=(Reduce('+', Tbaru[c(4,5)]))
Tbaru.3
Tbaru.4=(Reduce('+', Tbaru[c(8,9)]))
Tbaru.4
Tbaru.5=(Reduce('+', Tbaru[c(6,7,10,11,12,13)]))
Tbaru.5
Tbaru=as.matrix(Tbaru.1+Tbaru.2+Tbaru.3+Tbaru.4)
Tbaru

#Rekonstruksi
recon1<-reconstruct(S, groups=list(c(1), c(2:3), c(4:5), c(8:9)), len=p)
recon1
plot(recon1)
```

```
#Plot rekonstruksi trend, musiman, dan noise
trend=recon1$F1
plot(trend, type="o",col="blue")
season1=recon1$F2
plot(season1,type="o",col="red")
season2=recon1$F3
plot(season2, type="o",col="yellow")
season3=recon1$F4
plot(season3, type="o",col="green")
noise=residuals(recon1)
plot(noise, type="o",col="brown")

##Diagonal Averaging##
komponen=cbind(recon1$F1, recon1$F2, recon1$F3, recon1$F4)   #ganti banyak r
sebanyak groupingnya
komponen
diagonal.averaging=rowSums(komponen)
diagonal.averaging



#Linear Recurrence Formula#
S<-ssa(insample,L=13, kind="1d-ssa")
lrr.coeff=lrr(S, groups=list(c(1,2:3,4:5,8:9)))
lrr.coeff

###Recurrence Forecasting###
###CARA 1
diagonal.averaging[29]=(lrr.coeff[12]*diagonal.averaging[28])+
  (lrr.coeff[11]*diagonal.averaging[27])+(lrr.coeff[10]*diagonal.averaging[26])+
  (lrr.coeff[9]*diagonal.averaging[25])+(lrr.coeff[8]*diagonal.averaging[24])+
  (lrr.coeff[7]*diagonal.averaging[23])+(lrr.coeff[6]*diagonal.averaging[22])+
```

(lrr.coeff[5]*diagonal.averaging[21])+(lrr.coeff[4]*diagonal.averaging[20])+
(lrr.coeff[3]*diagonal.averaging[19])+(lrr.coeff[2]*diagonal.averaging[18])+
(lrr.coeff[1]*diagonal.averaging[17])

diagonal.averaging[29]

diagonal.averaging[30]=(lrr.coeff[12]*diagonal.averaging[29])+
(lrr.coeff[11]*diagonal.averaging[28])+(lrr.coeff[10]*diagonal.averaging[27])+
(lrr.coeff[9]*diagonal.averaging[26])+(lrr.coeff[8]*diagonal.averaging[25])+
(lrr.coeff[7]*diagonal.averaging[24])+(lrr.coeff[6]*diagonal.averaging[23])+
(lrr.coeff[5]*diagonal.averaging[22])+(lrr.coeff[4]*diagonal.averaging[21])+
(lrr.coeff[3]*diagonal.averaging[20])+(lrr.coeff[2]*diagonal.averaging[19])+
(lrr.coeff[1]*diagonal.averaging[18])

diagonal.averaging[30]

diagonal.averaging[31]=(lrr.coeff[12]*diagonal.averaging[30])+
(lrr.coeff[11]*diagonal.averaging[29])+(lrr.coeff[10]*diagonal.averaging[28])+
(lrr.coeff[9]*diagonal.averaging[27])+(lrr.coeff[8]*diagonal.averaging[26])+
(lrr.coeff[7]*diagonal.averaging[25])+(lrr.coeff[6]*diagonal.averaging[23])+
(lrr.coeff[5]*diagonal.averaging[23])+(lrr.coeff[4]*diagonal.averaging[22])+
(lrr.coeff[3]*diagonal.averaging[21])+(lrr.coeff[2]*diagonal.averaging[20])+
(lrr.coeff[1]*diagonal.averaging[19])

diagonal.averaging[31]

diagonal.averaging[32]=(lrr.coeff[12]*diagonal.averaging[31])+
(lrr.coeff[11]*diagonal.averaging[30])+(lrr.coeff[10]*diagonal.averaging[29])+
(lrr.coeff[9]*diagonal.averaging[28])+(lrr.coeff[8]*diagonal.averaging[27])+
(lrr.coeff[7]*diagonal.averaging[26])+(lrr.coeff[6]*diagonal.averaging[25])+
(lrr.coeff[5]*diagonal.averaging[24])+(lrr.coeff[4]*diagonal.averaging[23])+
(lrr.coeff[3]*diagonal.averaging[22])+(lrr.coeff[2]*diagonal.averaging[21])+
(lrr.coeff[1]*diagonal.averaging[20])

diagonal.averaging[32]

diagonal.averaging[33]=(lrr.coeff[12]*diagonal.averaging[32])+
(lrr.coeff[11]*diagonal.averaging[31])+(lrr.coeff[10]*diagonal.averaging[30])+
(lrr.coeff[9]*diagonal.averaging[29])+(lrr.coeff[8]*diagonal.averaging[28])+
(lrr.coeff[7]*diagonal.averaging[27])+(lrr.coeff[6]*diagonal.averaging[26])+

```
  (lrr.coeff[5]*diagonal.averaging[25])+(lrr.coeff[4]*diagonal.averaging[24])+
  (lrr.coeff[3]*diagonal.averaging[23])+(lrr.coeff[2]*diagonal.averaging[22])+
  (lrr.coeff[1]*diagonal.averaging[21])
diagonal.averaging[33]
diagonal.averaging[34]=(lrr.coeff[12]*diagonal.averaging[33])+
  (lrr.coeff[11]*diagonal.averaging[32])+(lrr.coeff[10]*diagonal.averaging[32])+
  (lrr.coeff[9]*diagonal.averaging[30])+(lrr.coeff[8]*diagonal.averaging[29])+
  (lrr.coeff[7]*diagonal.averaging[28])+(lrr.coeff[6]*diagonal.averaging[27])+
  (lrr.coeff[5]*diagonal.averaging[26])+(lrr.coeff[4]*diagonal.averaging[25])+
  (lrr.coeff[3]*diagonal.averaging[24])+(lrr.coeff[2]*diagonal.averaging[23])+
  (lrr.coeff[1]*diagonal.averaging[22])
diagonal.averaging[34]
hasil.forecasting=diagonal.averaging[29:34]
as.matrix(hasil.forecasting)

###CARA 2
forecast=rforecast(S,groups=list(c(1), c(2:3), c(4:5), c(8:9)), len=p)
forecast

hasil.forecast=as.matrix(forecast$F1+forecast$F2+forecast$F3+forecast$F4)
hasil.forecast

#Perbandingan data outsample dengan hasil ramalan
aktual=outsample[1:6]
aktual
ramalan=hasil.forecast[1:6]
ramalan

##Akurasi Peramalan##
###MAPE
residu=aktual-ramalan
residu
```

```
MAD=(sum(abs(residu))/(length(aktual)))
MAD
MSE=(sum(residu^2)/(length(aktual)))
MSE
PEI=(residu/aktual)*100
MAPE=(sum(abs(PEI)))/(length(aktual))
MAPE

###Peramalan IHK Lengkap
s.complete=ssa(data, L=L, kind="1d-ssa")
s.complete
r.complete=reconstruct(s.complete, groups=list(Trend=c(1), Season1=c(2:3), Season2=c(4,5), Season3=c(8,9)), len=6)  #ganti groupingnya
r.complete
plot(r.complete)
komponen.complete = cbind(r.complete$Trend, r.complete$Season1, r.complete$Season2, r.complete$Season3)  #ganti banyak r sebanyak groupingnya
komponen.complete
diag.avr.complete=rowSums(komponen.complete)
diag.avr.complete
forecast.complete=rforecast(s.complete, groups=list(c(1), c(2:3), c(4,5), c(8,9)), len=6)
forecast.complete
hasil.forecast.comp=as.matrix(forecast.complete$F1+forecast.complete$F2+forecast.complete$F3+forecast.complete$F4)
hasil.forecast.comp

##Plot
#data actual complete
data.complete<-as.matrix(data)
data.comp.kosong<-matrix(NA, 6)
data.comp.gab<-rbind(data.complete, data.comp.kosong)
```

```
dim(data.comp.gab)

#data prediksi complete
data.pred.comp<-rbind(as.matrix(diag.avr.complete))
data.pred.kosong<-matrix(NA, 6)
data.pred.gab<-rbind(data.pred.comp, data.pred.kosong)
dim(data.pred.gab)

#data forecasting menggunakan data actual complete
hasil.fore.comp<-as.matrix(hasil.forecast.comp)
forecast_kosong<-matrix(NA, 34)
forecast_gab<-rbind(forecast_kosong, hasil.fore.comp)
dim(forecast_gab)

#grafik hasil peramalan
opar=par(no.readonly = TRUE)
par(mfrow=c(1,1))
ts.plot(cbind(data.comp.gab, data.pred.gab, forecast_gab), type="l",
main=paste("IHK Data Actual, Prediction, & Forecasting by SSA"),
col=c("blue","red","orange"))
legend("topleft",c("Actual","Prediction","Forecasting"), col=c("blue", "red",
"orange"), lty=1)
```