УСРС 2 “Сингулярно-спектральный анализ”

Кобака Ф.А. 18ДКК-1 ФЦЭ

Работа выполнена на языке программировани python3 в окружении jupyter notebook. Далее ячейки кода и основные результаты. Полностью работу можно выкачать тут https://github.com/Dranikf/multivariate_statistical_analysis/tree/main/ysrs2%20ssa

1. **Сглаживание ряда с использованием SSA**

подгружаем бибилиотеки/ данные и настраиваем окружение
в конце методом “гусеницы” получаем матрицу X

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from ssa_functions import *
import sys
sys.path.append(sys.path[0][:-9] + "/common/")
import functions
import statsmodels.api as sm

M = 12
data = pd.read_csv("data.csv", decimal = ',', sep = ';')
print("data shape" + str(data.shape))
X = create_matrix(data.to_numpy(), M)
pd.DataFrame(X).to_excel("matrixes/X.xlsx")
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	…	36
0	284	296	309	296	342	335	344	361	347	341	340	323	346	…	428
1	272	284	296	309	296	342	335	344	361	347	341	340	323	…	409
2	286	272	284	296	309	296	342	335	344	361	347	341	340	…	373
3	301	286	272	284	296	309	296	342	335	344	361	347	341	…	330
4	296	301	286	272	284	296	309	296	342	335	344	361	347	…	345
5	310	296	301	286	272	284	296	309	296	342	335	344	361	…	343
6	349	310	296	301	286	272	284	296	309	296	342	335	344	…	355
7	324	349	310	296	301	286	272	284	296	309	296	342	335	…	403
8	313	324	349	310	296	301	286	272	284	296	309	296	342	…	383
9	321	313	324	349	310	296	301	286	272	284	296	309	296	…	378
10	314	321	313	324	349	310	296	301	286	272	284	296	309	…	433
11	324	314	321	313	324	349	310	296	301	286	272	284	296	…	491

получаем корреляционную матрицу для X
затем получаем ее собсвенные значения и из низ получаем занчение r

```
# getting correlation matrix for X matrix
corr_x_matrix = np.corrcoef(np.transpose(X))
pd.DataFrame(corr_x_matrix).to_excel("matrixes/X_correlations.xlsx")
#getting own meanings
c_L = functions.eig_matlab(corr_x_matrix)[0]
pd.DataFrame(c_L).to_excel("matrixes/corr_lambdas.xlsx")

# gerring number of values > 1
for i in range(c_L.shape[1]):
    if c_L[i,i] < 1:
        break
```

	0	1	2	3	4	5	6	
0	16,07	0,00	0,00	0,00	0,00	0,00	0,00	……
1	0,00	9,64	0,00	0,00	0,00	0,00	0,00	
2	0,00	0,00	3,58	0,00	0,00	0,00	0,00	
3	0,00	0,00	0,00	2,69	0,00	0,00	0,00	
4	0,00	0,00	0,00	0,00	1,27	0,00	0,00	
5	0,00	0,00	0,00	0,00	0,00	0,95	0,00	
6	0,00	0,00	0,00	0,00	0,00	0,00	0,87	
……								

Итак r = 5
Далее вычисляем матрицу A, вычисляем ее собственные вектора и из них забираем только первые R в матрицу V. После чего вычисляем преобразованную матрицу X.

```
A = np.dot(X,np.transpose(X))
[L, V] = functions.eig_matlab(A)

V= V[:,0:i]

pd.DataFrame(V).to_excel("matrixes/vectors.xlsx")

X_n = np.dot(np.dot(V,np.transpose(V)), X)
pd.DataFrame(X_n).to_excel("matrixes/new_X_matrix.xlsx")
```

V

0	1	2	3	4
0,30	-0,30	-0,46	0,34	-0,25
0,29	-0,37	-0,28	0,27	0,08
0,29	-0,37	-0,05	-0,07	0,39
0,29	-0,30	0,16	-0,39	0,24
0,29	-0,19	0,29	-0,39	-0,13
0,29	-0,06	0,36	-0,11	-0,42
0,29	0,08	0,35	0,29	-0,37
0,29	0,21	0,26	0,46	0,08
0,29	0,31	0,13	0,22	0,43
0,28	0,36	-0,07	-0,11	0,32
0,28	0,36	-0,28	-0,26	-0,06
0,28	0,31	-0,42	-0,27	-0,31

восстановленная матрица X

	0	1	2	3	4	5		33	34	35	36
0	286,75	293,62	304,48	303,39	330,83	344,17	…	332,90	366,23	416,83	447,35
1	278,49	285,60	295,91	297,46	315,15	327,32	…	330,88	337,39	368,19	404,62
2	277,54	279,21	286,91	293,97	300,97	311,37	…	353,82	334,14	331,02	358,29
3	289,87	280,29	279,80	287,12	290,67	300,67	…	372,43	352,41	331,39	335,92
4	307,57	290,37	280,61	281,23	284,24	292,73	…	378,53	370,25	352,04	338,06
5	322,45	305,38	289,34	280,06	280,37	284,69	…	380,22	378,32	371,57	350,97
6	329,75	321,38	305,95	289,55	282,79	280,02	…	395,06	383,50	382,01	367,08
7	325,86	329,38	322,08	307,79	291,57	281,12	…	432,93	396,88	381,30	375,43
8	320,26	327,43	328,00	322,23	303,51	289,69	…	476,43	429,80	393,15	382,56
9	320,02	323,14	327,25	327,93	318,46	306,74	…	498,27	472,05	433,21	407,90
10	320,06	318,41	323,36	325,47	330,94	324,20	…	488,55	498,04	477,92	442,44
11	314,47	310,99	316,74	318,92	334,89	332,61	…	464,60	497,34	495,28	459,78

далее восстановление ряда:

```
def reconstruct_series(Matrix):
    '''return series average by side diagonals of Matrix '''
    result = []
    M, n = Matrix.shape
    N = n + M - 1
    for t in range(N):
        sum = 0
        if t < M-1:
            for i in range(t+1):
                #print(str(M-t+i-1) + "," + str(i))
                #print(t+1)
                sum += Matrix[M-t+i-1, i]
            result.append(sum/(t+1))
        elif t < N-M+1:
            for i in range(M):
                #print(str(i) + "," + str(t-M+i+1))
                #print(M-1)
                sum += Matrix[i, t-M+i+1]
            result.append(sum/(M-1))
        elif t <= N:
            for i in range(N-t):
                #print(str(i) + "," + str(t-M+i+1))
                #print(N-t)
                sum += Matrix[i, t-M+i+1]
            result.append(sum/(N-t))
    return result

new_data = reconstruct_series(X_n)
data['ssa_series'] = new_data
```

в результате получи ряд:

[314.4669733670622, 315.5245825889568, 318.39084369202584, 321.4174874685699, 328.1817192414686, 329.76812829824667, 319.89419121250364, 305.19777183006977, 291.05131256015045, 281.47659542276136, 279.81383241403444, 310.72379029579565, 319.6576055164192, 329.9063391770636, 341.2087016247901, 355.8406600097143, 368.70096609762294, 380.7772741653993, 388.2746858782006, 383.77186088271077, 371.5420166405482, 362.1674035476253, 364.2700238315061, 376.67188097779086, 383.93754055717415, 375.5532620771144, 353.46967797271856, 337.38097182430903, 344.5170326694373, 370.1673384844085, 398.87379370179036, 426.02390725334374, 456.9138493178453, 497.2328255107741, 533.659837004023, 543.865173054627, 518.8985868587857, 434.26856042173864, 396.6336576412622, 380.9774243843087, 378.99884947757334, 370.1359667338576, 351.66383573159436, 334.1608347915662, 334.30816144778424, 364.23648776405986, 410.72424719919525, 447.35337190790466]

2. **Моделирование с помощью ARIMA.**

Опустим процесс идентификации и построения модели ARIMA, сразу к результату:
получена модель вида

$$\Delta y_t = -0{,}3632\,\Delta y_{t-9} + \varepsilon_t$$

получена следующая аппроксимация ряда:
[0.0, 323.79999999999995, 314.1, 321.0, 312.9, 323.7, 349.3, 310.4, 295.8, 301.12815900883913, 289.3228408473472, 269.1940616652887, 286.5417536972695, 291.77766173697364, 300.0026055987524, 309.82768133627076, 347.30242024446153, 333.1388308684869, 349.99296381950046, 366.0208305100621, 342.1781643212951, 336.2055284275352, 335.3607592243372, 328.2392407756628, 328.7847905946186, 351.8059383347113, 356.3224309401718, 314.0075387648209, 305.12978435070175, 320.6427588659125, 343.20895384063965, 366.9740509695785, 389.70109784578824, 407.7562359654447, 457.3229335244932, 505.81822491130976, 497.79990732285455, 484.24486188034354, 424.06578506755113, 371.5717234022624, 369.16235965444685, 399.2960720025743, 335.8146676224657, 331.99566209539836, 345.72686152191886, 330.2184102656006, 393.8644091903266, 429.1748707839304]

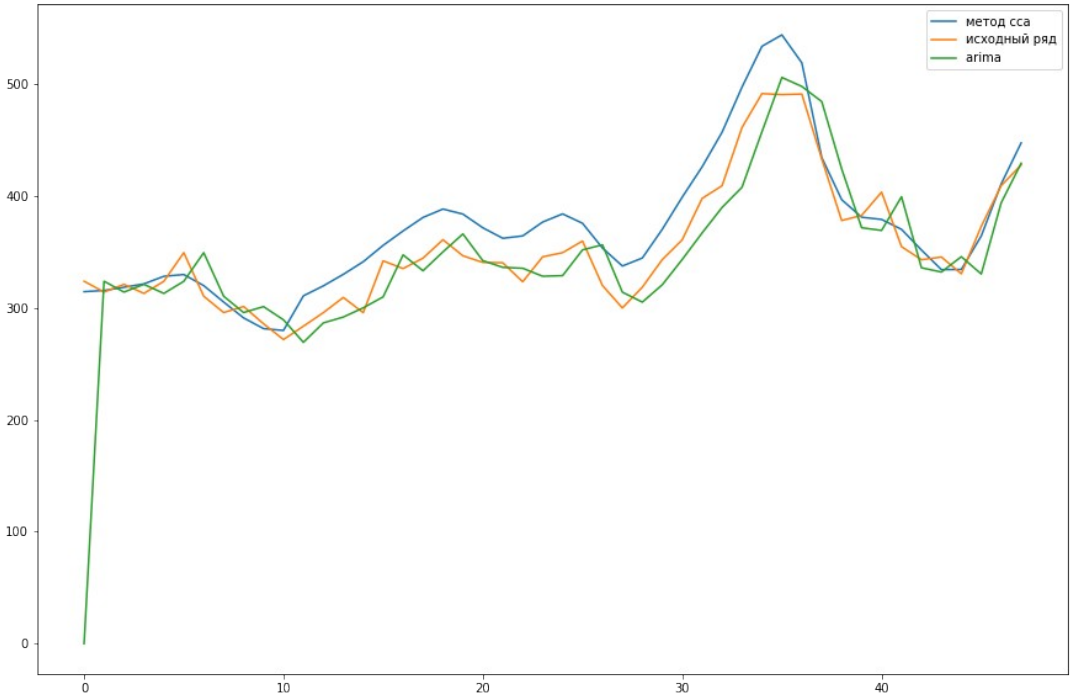
3. **Графическое отображение полученных результатов и вычисление дисперсий.**

Нанесение на график требуемых элементов

```
fig = plt.figure(figsize = [15,10])

plt.plot(data['ssa_series'])
plt.plot(data['y'])
plt.plot(data['pred'])
plt.legend(['метод сса', 'исходный ряд', 'arima '])

plt.savefig("result.png")
```



Вычисление дисперсий остатков

```
print("variance of ssa method " + str((((data['y'] - data['ssa_series']).to_numpy())**2).sum()))
print("variance of arima " + str((((data['y'] - data['pred']).to_numpy())**2).sum()))
```

variance of ssa method 32106.752916063568
variance of arima 130176.78188051633

4. Вычисление прогноза на 3 шага вперед используя представленную формулу

```
def ssa_forecaster(V, N, M, np_y, l):
    '''funciton for forecasting using ssa basics'''
    result = copy.copy(np_y)
    result = np.append(result, np.dot(np.dot(V[M-1,:], np.transpose(V[0:M-1,:])),np_y[N-M+1:])/(1-np.dot(V[M-1,:],V[M-1,:])))
    if l > 1:
        result = ssa_forecaster(V,N+1,M,result, l-1)
    return result
N = data['y'].shape[0]
result = ssa_forecaster(V, N, M, data['y'].to_numpy(), 3)
print(result[-3:])
```

В результате прогноз:

[443.150165 426.1512954 417.2603125]