Ural Federal University

named after the first President of Russia B.N.Yeltsin

Institute of radioelectronics and information technologies

STATISTICAL AND SPECTRUM ANALYSIS OF TIME SERIES

Laboratory Task no. 2

Yekaterinburg

2019

Contents

1. Introduction

On the previous task we have learned the basic concepts of Python time series data analysis. In this task we should learn the concept of statistical tests, usage of statistical criteria and spectrum methods of PSD (Power Spectrum Density) estimation.

2. Laboratory Task

The final result of laboratory task is presented as the report in form of *Jupyter*-notebook (or something similar), with all the needed commands for a complete Run of cells. Now, let's begin with our basic time series data analysis:

1) Import in to your code the following libraries:

```
import numpy as np
import numpy.random as rand
import matplotlib.pyplot as plt
from scipy import signal
import scipy.stats as stats
from statsmodels.tsa import api as tsa
from pandas.tools.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_acf
%matplotlib inline
```

2) Create your time series from the column of the tables below, based on your student ID card number, as a numpy array (np.array([…])) with 24 points.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1,65	23,46	0,54	30,42	20,89	12,60	3,54	15,48
2,59	14,86	2,16	30,56	20,11	18,92	7,81	9,29
6,18	20,14	5,39	29,90	16,41	17,08	12,83	8,26
6,26	21,59	3,48	21,67	18,95	15,51	6,73	5,45
6,44	18,98	4,54	26,31	21,43	8,97	6,29	10,49
7,16	21,77	7,99	28,13	16,54	14,52	15,88	14,47
10,56	20,27	7,95	24,06	11,55	12,77	12,27	9,46
10,93	16,86	7,01	20,55	14,39	12,96	7,84	8,79
9,53	16,23	9,89	24,35	20,66	5,55	10,71	12,96
10,64	18,55	12,35	18,12	15,31	11,09	14,60	15,37
17,43	14,87	12,91	18,69	9,34	9,23	17,48	11,82
14,72	11,98	14,42	14,88	11,39	5,03	12,97	11,34
15,50	14,41	14,13	11,66	11,34	2,15	11,34	20,84
15,01	13,42	18,67	19,83	10,07	8,95	23,82	16,58
17,83	10,44	16,95	14,10	5,95	8,04	19,97	12,47
18,43	8,26	15,84	10,16	4,59	5,68	11,51	7,05
17,69	8,86	19,23	10,08	8,74	0,14	18,07	15,08
19,80	9,53	22,05	5,82	9,96	5,85	22,11	16,97
22,64	6,88	22,59	8,46	3,03	4,21	23,12	13,51
22,86	4,10	21,15	5,50	3,17	2,56	15,52	13,45
21,56	7,61	23,98	3,60	4,45	0,08	20,03	16,55
22,16	4,92	26,45	8,44	4,06	3,87	24,36	18,47
25,82	1,79	29,80	3,04	0,16	1,10	27,02	21,73
26,50	0,10	27,41	0,00	1,52	0,85	21,31	14,04

9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
12,19	23,75	18,47	76,88	8,48	24,78	3,07	10,22
8,41	28,00	14,87	69,88	10,43	22,55	6,26	10,06
14,68	33,01	21,51	74,55	18,97	30,85	7,46	13,34
8,64	16,78	9,07	59,75	6,37	23,88	6,48	11,92
32,94	18,16	16,02	72,21	9,86	27,78	1,64	8,81
22,61	20,05	11,12	66,85	1,29	12,71	5,41	8,10
45,92	3,18	23,45	69,91	13,23	25,25	6,18	12,51
23,63	16,11	6,45	68,05	8,50	25,70	16,93	11,16
18,59	21,66	14,21	72,59	11,68	34,44	2,71	8,77
36,22	20,16	8,18	42,83	10,17	23,18	6,94	4,87
50,10	24,71	14,50	67,04	14,18	29,81	8,35	10,57
46,22	15,63	3,86	56,63	2,79	22,26	11,59	10,37
23,63	16,27	10,14	61,10	26,63	22,97	5,98	6,88
47,30	18,99	9,99	44,88	15,69	16,37	10,77	9,13
40,03	21,12	14,47	52,90	20,32	22,82	14,71	10,31
56,53	8,34	0,65	46,03	17,28	14,19	14,66	7,13
38,41	14,96	8,97	46,72	22,87	16,40	11,77	3,52
51,47	17,17	2,47	46,48	23,80	7,23	27,10	0,14
6,29	20,24	12,58	31,63	28,81	13,05	9,69	6,35
35,41	8,31	3,12	21,72	28,59	4,63	22,31	5,30
67,79	12,36	6,81	21,40	35,68	3,19	19,73	1,46
74,21	14,59	0,43	11,40	35,72	4,55	25,88	1,09
79,12	21,72	4,65	10,06	39,44	0,94	29,00	2,40
45,10	28,69	5,91	0,42	40,04	11,07	32,18	1,92

17	**18**	**19**	**20**	**21**	**22**	**23**	**24**
11,54	0,54	6,86	11,43	10,41	4,89	15,45	5,93
0,80	4,33	3,91	7,60	7,70	3,10	11,94	3,88
12,76	3,73	6,66	12,15	10,39	5,19	11,93	5,08
11,18	5,18	6,38	10,39	10,73	1,02	18,66	5,98
8,90	2,50	8,35	11,44	12,31	6,25	12,69	7,77
8,49	3,72	6,16	10,94	9,58	5,06	10,01	6,67
11,38	4,78	7,68	13,54	11,53	5,96	8,81	6,55
10,93	5,72	7,12	11,87	11,55	6,27	10,86	6,27
9,40	3,69	8,61	13,35	13,98	6,56	11,49	8,23
9,30	4,80	5,87	11,72	10,07	6,43	10,78	6,61
12,43	6,35	7,76	13,58	11,44	6,45	10,38	7,40
11,03	6,89	7,07	10,56	11,00	6,26	13,07	7,48
10,88	6,38	8,37	11,04	11,16	7,00	10,81	8,08
11,33	5,93	8,69	8,96	9,49	4,51	12,73	7,00
13,86	9,17	6,83	11,38	10,41	5,93	12,11	6,16
14,98	9,31	6,17	9,26	9,15	6,53	15,74	5,73
12,66	4,07	6,98	9,38	8,48	6,98	17,71	7,23
12,98	9,47	3,84	8,04	5,41	8,96	15,31	3,86
18,09	12,28	4,75	10,98	6,44	5,78	11,15	5,63
17,49	13,32	4,05	7,95	6,15	5,87	18,12	5,66
14,97	9,87	4,88	7,67	3,17	6,21	20,81	5,71
14,42	12,73	0,51	4,69	0,47	3,33	19,90	2,62
21,29	16,73	2,60	7,16	1,80	5,21	19,15	3,89
20,66	17,05	0,90	4,17	1,26	4,63	22,43	3,44

3) Plot on figure your time series data.
4) Calculate its expectation mean value and its variance.
5) Plot its ACF figure with 20 lags.
6) By using this function, estimate the periodogram of sample data:

   ```
   pds, pdden = signal.periodogram(X)
   ```

   Make some statements about character of your data, based on those figures and plots, Compare the results against the ACF.
7) Now, try to estimate another evaluation of PSD with Welch spectrum method:

   ```
   pdw, pddenw = signal.welch(X, nperseg = …)
   ```

   Compare the results with periodogram.
8) Plot those functions on the same plot.

9) Create **two-periodic signal** with these commands and plot it:

   ```
   t = np.linspace(0, 1, 4096)
   x1 = np.sin(2*np.pi*10*t) + np.sin(2*np.pi*120*t)
   plt.figure(figsize = (10, 5))
   plt.plot(t, x1)
   ```

10) Estimate periodogram and Welch spectrum for it:

   ```
   pd1, pdden1 = signal.periodogram(x1)
   pdw1, pddenw1 = signal.welch(x1, nperseg = 1024)
   plt.figure(figsize = (10, 5))
   plt.semilogy(pd1, pdden1)
   plt.semilogy(pdw1, pddenw1)
   ```

11) Create periodic signal with **frequency break** and plot it:

```
t = np.linspace(0, 1, 4096)
x2 = np.zeros(4096)
for i in range(0, len(t)//2):
   x2[i] = np.sin(2*np.pi*10*t[i])
for i in range(len(t)//2, len(t)):
   x2[i] = np.sin(2*np.pi*120*t[i])
plt.figure(figsize = (10, 5))
plt.plot(t, x2)
```

12) Estimate periodogram and Welch spectrum for it:

```
pd2, pdden2 = signal.periodogram(x2)
pdw2, pddenw2 = signal.welch(x2, nperseg = 1024)
plt.figure(figsize = (10, 5))
plt.semilogy(pd2, pdden2)
plt.semilogy(pdw2, pddenw2)
```

13) Plot them against each other:

```
plt.figure(figsize = (10, 5))
plt.semilogy(pd1, pdden1)
plt.semilogy(pd2, pdden2)
plt.figure(figsize = (10, 5))
plt.semilogy(pdw1, pddenw1)
plt.semilogy(pdw2, pddenw2)
```

14) Note how close the spectrum estimations for those two signals, even though the initial data is different. This is the main problem with spectrum methods – they lack the feature to describe the change of period with time. This problem will be solved later with the introduction of spectrograms.

15) Find in your initial time series the **anomalous observations** with **Irvine method**. In order to do this, you need to write a **function** that receives the input time series and returns the numbers or indices of those anomalous points.

16) First of all, you need to estimate the criterion $k_i = \frac{|y_i - y_{i-1}|}{\sigma}$. You can find the σ with this: $\sigma = \sqrt{\frac{\sum_{i=1}^{N}\left(y_i - \overline{y}\right)^2}{N-1}}$ or just simply use **std()** function.

17) For *N*=24 you need to compare these *k_i* with critical criterion **1.23** ($\alpha = 0.05$). If $k_i > k_{critical}$, then the k_i is an anomaly. Return the *i*.

18) Plot the time series with these denoted points on the same figure.

19) Now create a function to perform the **Run Test**. To do this you need to follow these steps.

20) Rearrange the initial data into **ascending order** and estimate its **median** value (for the *N*=24): $x_{med} = 0.5\left(x_{(12)} + x_{(13)}\right)$.

21) Shape the run of "+" and "–" for the **initial time series**:

«+», if $x_i > x_{med}$; «–», if $x_i < x_{med}$;

if $x_i = x_{med}$, then disregard these points;

22) Number of alternations ν and length τ of the max run should be:

$$\begin{cases} \nu > 0.5\left(N + 2 - 1.96\sqrt{N-1}\right) \\ \tau < 1.43\ln\left(N+1\right) \end{cases}$$

for **random** data. If not, then the data is not random (just what we need).

23) Your final function should provide the results of **accepting** or **rejecting** some kind of statistical null-hypothesis, stated in your notebook, for

example, as print('Run Test: the data is random.') or print('The null hypothesis of random data has been accepted'). It also should print the series of "+" and "-" from Run Test, in order to visually confirm the achieved results.

24) Test your time series for a stationarity. Let's use the **KPSS-test**. There is a function, called **tsa.kpss(X)**, which returns the test statistics **kpss_stat** and p-value of null hypothesis **p_value**, as well as some other stats.

25) If the test statistics is close to 0, then the time series can be noted as **stationary**, meaning, the **null hypothesis** for trend-stationarity of data is **accepted**.

26) If the test statistics is significantly larger than 0, then the time series is **non-stationary**, meaning the **null hypothesis** for trend-stationarity of data is **rejected** and the **alternative hypothesis** is **accepted**.

27) A better value to watch closely is the **p-value**. If this value is **lower than 0.05** (for, example, equal **0.01**), then the **null hypothesis is rejected**. And if the p-value is **higher than 0.05** (for, example, equal **0.1**), then the **null hypothesis is accepted**. In case of p-value **close to 0.05**, the overall answer is unreliable, meaning the sample size was insufficient for effective data analysis.

28) Now, let's try the **Fisher-test** to compare the variance in the time series data between its first half and the second half of time-span.

29) For this you can use function **stats.f_oneway(x, y)**.

30) Stipulate, which null hypothesis has been accepted or rejected with this F-test.

31) Similarly, compare the expectation value between its first half and the second half of time-span with **t-Student test**.

32) Estimate the criterion for test with equation:

$$K_S = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{(N_1 - 1)D_1 + (N_2 - 1)D_2}} \cdot \sqrt{\frac{N_1 N_2 (N_1 + N_2 - 2)}{N_1 + N_2}}$$

33) Test for $K_S > t(1-\alpha, N_1 + N_2 - 2)$, where $t(\ldots)$ can be estimated in Python as **stats.t.ppf(1-a, N1+N2-2)**.

34) If it's true – then the mean value is not constant, data is non-stationary.

35) The final function should only show the message: "The null hypothesis has been accepted" or "The null hypothesis has been rejected". And maybe some small statistics, for example, which test failed, how and so on.

36) Compare your calculations against the results from the imported function **scipy.stats.ttest_ind(x, y)**

3. Report requirements

Report in Jupyter-notebook should include: number of laboratory task, name of student, student ID or variant number, student group number and the complete task (code, plots and figures) with student comments in the report.