Ural Federal University
named after the first President of Russia B.N.Yeltsin
Institute of radioelectronics and information technologies

BASIC TIME SERIES DATA ANALYSIS

Laboratory Task no. 1

Yekaterinburg
2019

Ural Federal University
Institute of radioelectronics and information technologies

Contents

1. Requirements for Python 3

2. Laboratory Task 4

3. Report requirements 7

1. Requirements for Python

The most of Data Analysis in the current time is evaluated through Python and its impressive number of libraries. Some of them are required for basic time series data analysis. The overall import list for libraries is presented below:

Numpy (for arrays, matrices and useful functions working with them)

Scipy (scientific and complex aggregate functions)

Pandas (data analysis library)

Seaborn (visualization of data)

Matplotlib (figures and plots in MATLAB style)

Statsmodels (statistical characteristics and ARIMA models)

Spectrum (some useful methods for spectral methods)

h5py (for files in HDF5, including support for files with *.mat extenstion)

and some others.

The presented code in laboratory tasks is written for Python 3. Most of libraries, required for import, can be installed through **pip**/**pip3** or are already installed with some data analysis Python package. For example, one of the most useful is **Anaconda** (https://www.anaconda.com/distribution/), although, there are others as well.

Moreover, we require the **Jupyter** (**Jupyter notebook**) for its simple python-notebooks, where interactive code can be mixed with plots, comments, texts and other fields, making it virtually a final report for completed task. Although, students can also use other interactive platforms for work, such as Spyder, IPython, IDLE and so on.

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 pandas.tools.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_acf
import h5py
%matplotlib inline
```

2) Let's create the random time series, as a sample of normal distribution with 10000 points of data:

```
X = rand.randn(10000)
```

3) To make a complete time series, one need's to also identify its time-scale, for example, those random observations were estimated on time interval between 3 and 5:

```
t = np.linspace(3, 5, num = 10000)
```

where function creates linear array of 10000 numbers between 3 and 5.

4) To plot our time series:

```
plt.figure(figsize = (10, 5))      # size of figure
plt.plot(t, X)
```

5) Now, let's estimate the mean value = expectation value:

First, there is a simple function for that:

```
M = np.mean(X)
```

Now, create your own code to estimate the expectation value, based on this formula (protip, use function np.sum(X) to sum up data-points):

$$M_x = \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Compare your result against mean function – they must be the same. For this, you can use function print();

6) Estimate the dispersion (*variance*) of your time series:

With imported function:

```
D = np.var(X)
```

And with your own calculations, based on this formula:

$$D_x = \sigma^2 = \frac{1}{N}\sum_{i=0}^{N}\left(x_i - \bar{x}\right)^2$$

Compare your result against variance function.

7) Calculate the asymmetry of time series with your own code:

$$a = \frac{M\left[\left(x - \bar{x}\right)^3\right]}{\sigma^3},$$

Try to find the similar function in python libraries with the key-word Skewness. Compare your result against imported function.

8) Calculate the excess of time series:

$$e = \frac{M\left[\left(x - \bar{x}\right)^4\right]}{\sigma^4}.$$

Try to find the similar function in python libraries with the key-word Kurtosis. Compare your result against imported function.

9) Estimate the ACF = Auto-Correlation Function, up to the 20 lag. Then plot it on figure.

First of all, there is function **plot_acf(X[0:20])**

Second, you can use function of correlation against the same data **np.correlate(x, x, mode = 'full')** to estimate the same value

And next, try to estimate the ACF with your own code:

$$r(l) = \frac{(N-l)\sum_{i=1}^{N-l} y_i y_{i+l} - \left(\sum_{i=1}^{N-l} y_i\right)\cdot\left(\sum_{i=1}^{N-l} y_{i+l}\right)}{\sqrt{(N-l)\sum_{i=1}^{N-l} y_i^2 - \left(\sum_{i=1}^{N-l} y_i\right)^2}\cdot\sqrt{(N-l)\sum_{i=1}^{N-l} y_{i+l}^2 - \left(\sum_{i=1}^{N-l} y_{i+l}\right)^2}}$$

Compare the results.

10) Now, **define the function** in Python, that has only one input parameter (the time series data) and calculates for it all of the characteristics, presented before: expectation value, variance, asymmetry, excess, plots the ACF.

11) Find in the folder “Materials for labs” the mat-file, corresponding to your variant number. This number can be defined through your student ID number, the last two numbers from it.

12) Load time series data for this file, for example, for ID xxxx12:

```
Xmat = h5py.File('12.mat', 'r')
Xmat = Xmat.get('z12')
Xmat = np.array(Xmat)
```

13) Use your function (see task 10 before) to estimate the characteristics of your time series.

14) Complete the formatting of your final report in *Jupyter*-notebook form.

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.