МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени
первого Президента России Б. Н. Ельцина»

ОСНОВНЫЕ ПРОГРАММНЫЕ СРЕДСТВА РАБОТЫ

Методические указания к выполнению практического задания № 1

Екатеринбург
2020

Содержание

Введение

Как указано в Википедии, «**Python** – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода». Ну а если быть более точным, то, в первую очередь, Python – это **интерпретируемый** язык программирования, компилируемый только в **байт-код** некоторой виртуальной ретранслируемой машины. Это значит, что такой код всегда кроссплатформенный и аппаратно-независимый (он зависит только от того, установлены библиотеки Python или нет). При этом такой код выполняется по шагам, по мере поступления исходного кода на вход интерпретатора. А значит подобный язык программирования может работать в режиме диалога с программистом, в режиме цикла чтения-вычисления-печати. Это очень удобно как раз при математических вычислениях, где каждый промежуточный результат может потребовать изменить что-либо в предварительных расчетах. Интерпретируемость языка Python так же приводит к тому, что он поддерживает почти все известные парадигмы программирования: структурное, объектно-ориентированное (ООП), функциональное и другие.

Из других особенностей важно отметить следующие: динамическая типизация, автоматическое управление памятью (= сборка мусора) и огромный набор готовых библиотек (их так много, что иногда они дублируют друг друга). **Динамическая типизация** значит, что переменная связывается с типом в момент присваивания значения, и не при объявлении. То есть нет необходимости как в языке С для каждой переменной писать явный тип, интерпретатор сам поймет, что Вы хотите хранить в переменной – строчку, массив, целое, дробное число и т.д. Правда, проблемы начнутся, когда интерпретатор поймет Вас неправильно…

За все эти плюсы, правда, Python отделался низкой производительностью, но сейчас существует множество средств, который позволяют перевести код из Python в некоторый компилированный код, подходящий под заданное «железо», устраняя недостаток по скорости, но и лишая нас всех тех преимуществ, что уже были.

При работе со средствами анализа временных рядов (ВР) предполагается, что у студентов уже есть базовые навыки в области алгоритмизации процессов, языков программирования, а также знания математического анализа, понятия векторов, матриц.

1. Что потребуется для работы с ВР в Python

Так как нам уже стало ясно, что самое важное в Python – это набор его готовых библиотек, то сразу же оговоримся, какие из них нам пригодятся. Вот их примерный список:

Numpy (поддержка массивов, матриц и море полезных функций)

Scipy (различные сложные математические алгоритмы и прочая наука)

Pandas (библиотека для анализа данных)

Seaborn (визуализация статистических данных)

Matplotlib (красивые графики в стиле MATLAB)

Statsmodels (расчет статистических характеристик для ВР + ARIMA)

Spectrum (для метода Юле-Уолкера)

h5py (для загрузки файлов в формате HDF5, к которым относятся *.mat)

и другие по мере необходимости …

Часть из этих библиотек потребует подтянуть еще множество других, поэтому зачастую легче всего установить некоторый готовый пакет, содержащий в себе самое основное из них. Таковым, например, является пакет **Anaconda** (https://www.anaconda.com/distribution/), есть и другие, главное, чтобы они содержали нужные нам библиотеки.

Кроме того, понадобится средство python-тетрадей Jupyter (**Jupyter notebook**), где интерактивный код и отчет можно совместить в одной веб-странице, избавляя нас от необходимости оформления формальных отчетов с копированием и вырезанием картинок и т.д. Для обучения анализу данных в Python сложно найти что-то более подходящее, хотя **допускается** сдача лабораторных работ в других средах программирования на Python (Spyder, IPython, и др.).

2. Задание на лабораторную работу

Результатом выполнения лабораторной работы является оформленный отчет в виде *Jupyter*-тетради, в котором должны быть представлены и отражены все нижеперечисленные пункты:

1) Сначала импортируйте в свой код нужные библиотеки, функции и т.д.

```
import numpy as np
import numpy.random as rand
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot
from statsmodels.graphics.tsaplots import plot_acf
import h5py
%matplotlib inline
```

2) Создать ВР, являющийся выборкой случайной величины с нормальным распределением

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

где данная функция генерирует случайные величины из нормального распределения размером 1x10000.

3) Создать для него ряд временных отсчетов, на которых он будет определен:

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

где данная функция создает линейный равномерный массив из 10000 элементов в интервале от 3 до 5.

4) Построить ВР на заданной временной сетке с помощью функций

```
plt.figure(figsize = (10, 5))      # здесь задается размер рисуемой области
plt.plot(t, X)
```

5) Должно отобразиться получившееся изображение ВР.

6) Найти мат. ожидание данного ВР двумя способами:

Во-первых, с помощью функции

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

которая считает среднее значение ряда;

Во-вторых, с помощью **собственных расчетов** на основе формулы (2.7) из лекции 2.

Внимание! Нужно писать собственные расчеты над временными рядами в виде операций/функций над массивами, то есть **не использовать циклы**. Например, здесь нам пригодится функция `np.sum(X)`.

Сравнить полученные результаты расчетов (подсказка – они должны быть одинаковые).

Через функцию `print()` можно выводить полученные численные значения.

7) Найти дисперсию (*variance*) данного ВР двумя способами:

С помощью функции

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

которая считает дисперсию ряда;

С помощью **собственных расчетов** на основе формулы (2.8) из лекции 2.

Сравнить полученные результаты расчетов.

8) Найти асимметрию ВР по формуле (2.9). Найти в Python функцию, которая считает ту же самую характеристику, искать по ключевому слову `Skew`. Сравнить полученные результаты расчетов.

9) Найти эксцесс ВР по формуле (2.10). Найти в Python функцию, которая считает ту же самую характеристику, искать по ключевому слову `Kurtosis`. Для нее использовать параметр `fisher = False`. Сравнить полученные результаты расчетов.

10) Построить оценку выборочной автокорреляции ВР несколькими способами (до 20 лага) и построить ее на графике:

С помощью функции **plot_acf(X[0:20])**

На основе расчетов через функцию np.correlate(x, x, mode = 'full')

С помощью **собственных расчетов** на основе формулы (2.17).

Сравнить полученные результаты в отчете.

11) Написать **полную функцию**, которая имеет **один входной параметр** – это исходный временной ряд для анализа.

Функция должна выполнять все вышеперечисленные перечисленные действия (кроме 1 пункта, конечно же) для того ВР, что был передан ей в качестве параметра. То есть вычислять мат. ожидание, дисперсию, асимметрию, эксцесс и строить АКФ.

12) Получить у преподавателя **mat-файлы**, содержащие массивы некоторых ВР, **по вариантам**. Номер варианта определяется по последним двум цифрам студенческого билета.

13) Загрузить из этих **mat**-файлов массив ВР. Например, для 12-го варианта:

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

14) Используйте уже написанную функцию (пункт 11) от этого ВР для того, чтобы получить все его базовые характеристики.

15) Пояснить, с чем могут быть связаны особенности поведения или значений тех или иных характеристик.

3. Требования к оформлению отчета

Отчет в Jupyter-тетради должен обязательно содержать: номер лабораторной работы, ФИО студента, номер варианта (либо студенческий номер), номер группы, результаты выполнения работы с комментариями студента (комментарии пишутся после #) и изображениями, выводы.