

## 0.1 Getting started

Полезные ссылки - Python

<https://habr.com/ru/sandbox/80371/>

<https://pythonworld.ru/samouchitel-python>

<https://pythonworld.ru/uploads/pythonworldru.pdf>

<https://jenyay.net/Matplotlib/Matplotlib>

Полезные ссылки - R

<https://www.statmethods.net/>

<http://www.inp.nsk.su/baldin/DataAnalysis/R/R-01-intro.pdf>

<https://stepik.org/course/497/promo>

<https://stepik.org/course/129/promo>

## 0.2 Теория вероятностей

### 0.2.1 Центральная предельная теорема

1.1. Независимые с.в.  $\xi_i \sim \exp(1)$ ,  $S_n = \sum_{i=1}^n \xi_i$ . Построить графики плотностей  $S_n$ ,  $n = 1, 2, \dots, 10$ . Изобразить на одном графике плотность централизованной и нормированной с.в.  $S_n$  и плотность  $N(0, 1)$  для  $n = 1, 2, \dots, 20$ . Также изобразить графики ф.р.

1.2. С. в.  $\xi_i$  независимы и имеют плотность  $(e^{-(x-1)}\mathbb{I}\{x > 1\} + e^{-x}\mathbb{I}\{x > 0\})/2$ ,  $S_n = \sum_{i=1}^n \xi_i$ . Найти формулу для плотности распределения с.в.  $S_n$ . 1) Построить графики плотности для  $n = 1, 2, \dots, 10$  на одном графике.

2) Построить графики плотности централизованной и нормированной суммы и стандартного нормального распределения (2 плотности на одном графике) для  $n = 1, 2, \dots, 10$ .

1.3. Распределение случайной величины  $\xi$  является смесью двух нормальных распределений:  $\mathcal{N}(a, b)$  с весом  $1/3$  и  $\mathcal{N}(c, d)$  с весом  $2/3$ . Найти формулу для плотности распределения суммы  $n$  независимых реализаций случайной величины  $\xi$ .

1) Построить графики плотности для  $n = 1, 2, \dots, 10$ . По графикам найти числа локальных максимумов плотности для каждого из 10 значений  $n$ . Параметры  $a, b, c, d$  выбрать так, чтобы плотность  $\xi$  имела два выраженных локальных максимума.

2) Построить графики плотности централизованной и нормированной суммы и стандартного нормального распределения (2 плотности на одном графике) для  $n = 1, 2, \dots, 10$ .

### 0.2.2 Байесовские оценки

2.1.  $X_1, \dots, X_n \sim \text{Bern}(\theta)$ ,  $\theta \sim R[0, 1]$ . Построить график апостериорной плотности  $\theta$  а) для выборок размера  $n = 5, 10, 20, 50, 100$  (можно  $n = 10, 20, \dots, 100$ ), б) для значения  $\sum_{i=1}^n X_i = 9n/10, 99n/100$ .

2.2. Сделать то же, что в 2.1 для  $X_1, \dots, X_n \sim \text{Bern}(\theta)$ ,  $\theta \sim \text{Beta}(a, b)$ .

2.3.  $X_1, \dots, X_n \sim \text{Bern}(\theta)$ . Сравнить отклонения оценок  $\hat{\theta}_1 = \bar{X}$  и  $\hat{\theta}_2 = \bar{X} + (\frac{1}{2} - \bar{X}) / (1 + \sqrt{n})$  от  $\theta$ : генерировать 100 выборок и оценить вероятность того, что  $\hat{\theta}_2$  ближе к  $\theta$ , чем  $\hat{\theta}_1$  (при разных значениях  $\theta$ .)

## 0.3 Базовый синтаксис

1) `math.exp(x)` –  $e^x$   
`math.factorial(x)` –  $x!$   
`math.gamma(x)` – гамма-функция  
`math.pi`, `math.e` – числа  $\pi$  и  $e$

2) `scipy.special.binom(m,n)` – биномиальный коэффициент  
или  
`scipy.special.comb(m,n)`

Т.е. получается так  
`import scipy.special`  
`scipy.special.binom(10, 5)`  
252.0  
`scipy.special.comb(10, 5)`  
252.0

получим одинаковые результаты, однако, если вам интересна разница, то можно посмотреть тут  
<https://qa-help.ru/questions/kakaya-raznicza-mezhdu-scipyspecialbinom-i-scipymisccomb>  
<https://stackoverflow.com/questions/26560726/python-binomial-coefficient>

3) `scipy.special.gamma` - Гамма функция  
Хорошее описание в документации библиотеки `scipy`  
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.gamma.html>

4) Цикл `While`.  
Выполняет тело цикла до тех пор, пока условие цикла истинно.

```
>i = 5
>while i < 15:
... print(i)
... i = i + 2
...
5
7
9
11
13
```

5) Цикл `for`.  
Этот цикл проходится по любому итерируемому объекту (например строке или списку), и во время каждого прохода выполняет тело цикла.

```
>for i in 'hello world':
... print(i * 2, end="")
...
hheelllloo wwoorrlldd
```

6) Оператор `continue`  
Оператор `continue` начинает следующий проход цикла, минуя оставшееся тело цикла (`for` или `while`)

```
>for i in 'hello world':
... if i == 'o':
... continue
... print(i * 2, end="")
...
hheellll wwrrlldd
```

### 7) Оператор break

Оператор break досрочно прерывает цикл.

```
>for i in 'hello world':  
... if i == 'o':  
... break  
... print(i * 2, end="")  
...  
hheellll
```

### 8) Else.

Слово else, примененное в цикле for или while, проверяет, был ли произведен выход из цикла инструкцией break, или же “естественным” образом.

Блок инструкций внутри else выполнится только в том случае, если выход из цикла произошел без помощи break.

```
>for i in 'hello world':  
... if i == 'a':  
... break  
... else:  
... print('Буквы а в строке нет')  
...  
Буквы а в строке нет
```

### 9) Пример построения графика <https://jenyay.net/Matplotlib/Matplotlib>

```
import math  
import pylab  
from matplotlib import mlab
```

Будем рисовать график этой функции

```
def func (x):  
if x == 0:  
return 1.0  
return math.sin (x) / x
```

Интервал изменения переменной по оси X

```
xmin = -20.0  
xmax = 20.0
```

Шаг между точками

```
dx = 0.01
```

!!! Создадим список координат по оси X на отрезке [-xmin; xmax], включая концы

```
xlist = mlab.frange (xmin, xmax, dx)
```

Вычислим значение функции в заданных точках

```
ylist = [func (x) for x in xlist]
```

!!! Нарисуем одномерный график

```
pylab.plot (xlist, ylist)
```

!!! Покажем окно с нарисованным графиком  
pylab.show()

Еще пример <http://profitraders.com/Math/Norm.html>

```
import math
def gaussian(x, mu, sigma):
    return math.exp(-0.5*((x-mu)/sigma)**2) / sigma /
    math.sqrt(2*math.pi)

import numpy as np
import matplotlib.pyplot as plt

xs = np.arange(-8, 8, 0.05) # Сетка значений по оси абсцисс.
p1, = plt.plot(xs, [gaussian(x, 0, .5) for x in xs], label='σ = 0.5')
p2, = plt.plot(xs, [gaussian(x, 0, 1) for x in xs], label='σ = 1')
p3, = plt.plot(xs, [gaussian(x, 0, 2) for x in xs], label='σ = 2')
plt.legend()
plt.show()
```

## 0.4 Статистическая практика

### 0.4.1 Генерирование случайных величин

Нам понадобится генерировать случайные величины из различных распределений.

Для этого есть два наиболее простых варианта:

- Напрямую с помощью специальных функций. В R для этого пригодятся функции с буквой r: rnorm, runif, rcauchy и так далее. Более полный список вы можете получить с помощью команды `help.search("distribution package="stats")`. В Python можно либо использовать библиотеку random: `import random as rnd` загрузит ее в память под именем rnd, а внутри нее есть различные генераторы: `rnd.gauss`, `rnd.random`, `rnd.exprovariate`, либо использовать `scipy.stats`, где выбирая распределение можно применять функцию `rvs`, скажем `stats.norm.rvs(5,1,2)` создаст выборку из 5 величин  $\mathcal{N}(1, 2)$ .
- С помощью метода обратной функции. Если вы знаете функцию распределения  $F$  величины  $X$ , возьмете к ней обратную  $F^{-1}$  и сгенерируете равномерную  $R \sim R[0, 1]$ , то  $F^{-1}(R)$  будет иметь функцию распределения  $F$ . Кстати, обратные к функциям распределения в R задаются буквой q: `qnorm`, `qunif`, а в Python все в том же `scipy.stats` можно использовать метод `ppf` вместо `rvs`.
- В Python За генерацию случайных величин отвечает библиотека random: `import random as rnd` Внутри нее есть различные генераторы: `rnd.gauss`, `rnd.random`, `rnd.exprovariate` и так далее. Отметим также `rnd.sample(array,number)`, выбирающее `number` случайных элементов из набора `sample`.
- Почитать про генерацию случайных чисел можно здесь <https://ps.readthedocs.io/ru/latest/random.html>

Для того, чтобы посмотреть что получилось, можно использовать, например, гистограммы. Простое описание нескольких базовых методов визуализации в Python можно найти <https://medium.com/nuances-of-programming/5-%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%8B%D1%85-%D1%81%D0%BF%D0%BE%D0%B2%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-%D0%BD%D0%B0-python-%D1%81-%D0%BA%D0%Be0053808c83d>, а в R — <https://r-analytics.blogspot.com/2011/11/r06.html>. *XUHN5OgzaM8*, <https://rpubs.com/>

Для построения графиков в Python можно использовать matplotlib.pyplot: `import matplotlib.pyplot as plt`  
Гистограмма массива array с разбиением на columns колонок — `plt.hist(array,colums)`  
Двумерный точечный график (scatterplot)x,y — `plt.plot(x,y)`

Пример визуализации и генерации случайных величин можно посмотреть здесь  
[https : //habr.com/ru/post/331560/](https://habr.com/ru/post/331560/)

1. Сгенерировать тысячу величин а) экспоненциальных б) Коши в) нормальных каждым из методов
2. Построить для полученных распределений гистограммы

## 0.4.2 Графики

1. Построить точечный график  $(i, X_i)$ , где  $X_i$  — экспоненциальные с.в.
2. Построить график функции  $y = 2x^2 - 3x^3 + 5x$
3. Построить два предыдущих графика на одном