Санкт-Петербургский Политехнический
Университет им. Петра Великого

Институт прикладной математики и механики

Кафедра прикладной математики

# Лабораторная работа №3
# Боксплот Тьюки

3 курс, группа 33631/2

Студент Д. А. Плаксин

Преподаватель Баженов А. Н.

Санкт-Петербург
2019 г.

# Содержание

# 1 Список иллюстраций



# 2 Список таблиц

## 3 Постановка задачи

Для, приведённых ниже, пяти распределений сгенерировать выборки объёмом 20, 100, для каждой выборки построить боксплот Тьюки. Для каждого распределения определить процент выбросов экспериментально. Сгенерировать выборку, соответствующую распределению 1000 раз и, вычислив средний процент, сравнить его с результатами, полученными теоретически.

Распределения:

1. Стандартное нормальное распределение
2. Стандартное распределение Коши
3. Распределение Лапласа с коэффициентом масштаба $\sqrt{2}$ и нулевым коэффициентом сдвига.
4. Равномерное распределение на отрезке $\left[-\sqrt{3}, \sqrt{3}\right]$
5. Распределение Пуассона со значением матожидания равным двум.

## 4 Теория

Боксплот Тьюки - график, использующийся в описательной статистике, изображающий одномерное распределение вероятностей.

Такой вид диаграммы в удобной форме показывает медиану, нижний и верхний квартили, минимальное и максимальное значение выборки и выбросы.

1. Выборочная медиана [1]:

$$med\ x = \begin{cases} x_{k+1}, & n = 2k+1 \\ \frac{1}{2}\left(x_k + x_{k+1}\right), & n = 2k \end{cases} \tag{1}$$

2. Квартиль [2]:

$$z_{[p]} = \begin{cases} x_{np}, & np \in \mathbb{Z} \\ x_{[np]+1}, & np \notin \mathbb{Z} \end{cases} \tag{2}$$

Выбросом в статистике называют результат измерения, выделяющийся из общей выборки.

## 5 Реализация

Для генерации выборки был использован *Python* 3.7: модуль *random* библиотеки *numpy* [4].

Боксплот Тьюки был построен средствами библиотеки matplotlib [5].

Правая и левые границы: $R - LQ - l(UQ - LQ),\ L = UQ - k(UQ - LQ)$, где $k$ обычно полагают равным 1.5 [3]

Число выбросов определялось таким образом: если значение из выборки находится вне установленных левой и правых границ, то оно является выбросом.

## 6 Результаты

Рис. 1: Boxplot нормальное распределение

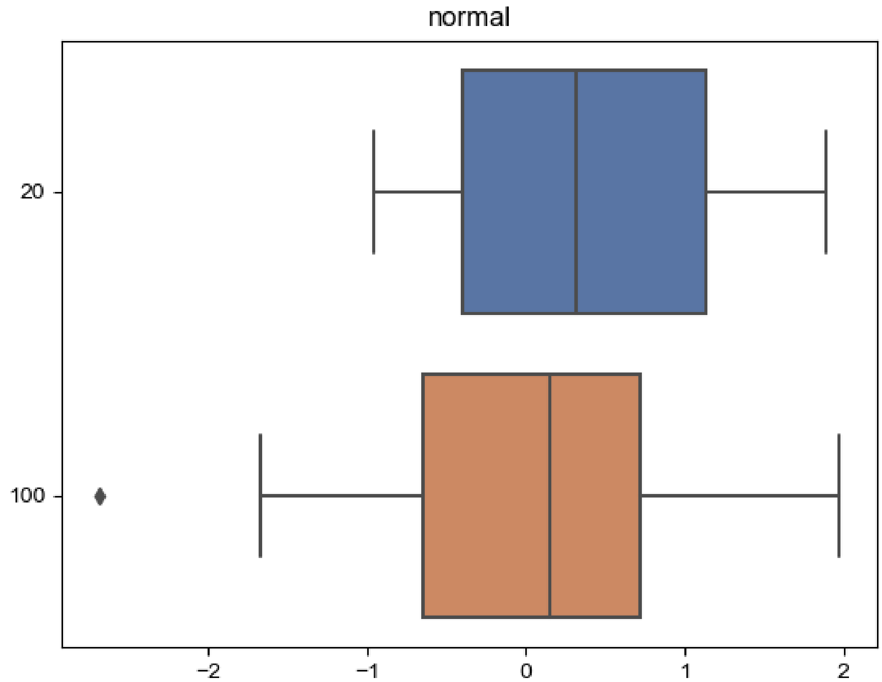

Рис. 2: Boxplot стандартное распределение Лапласа

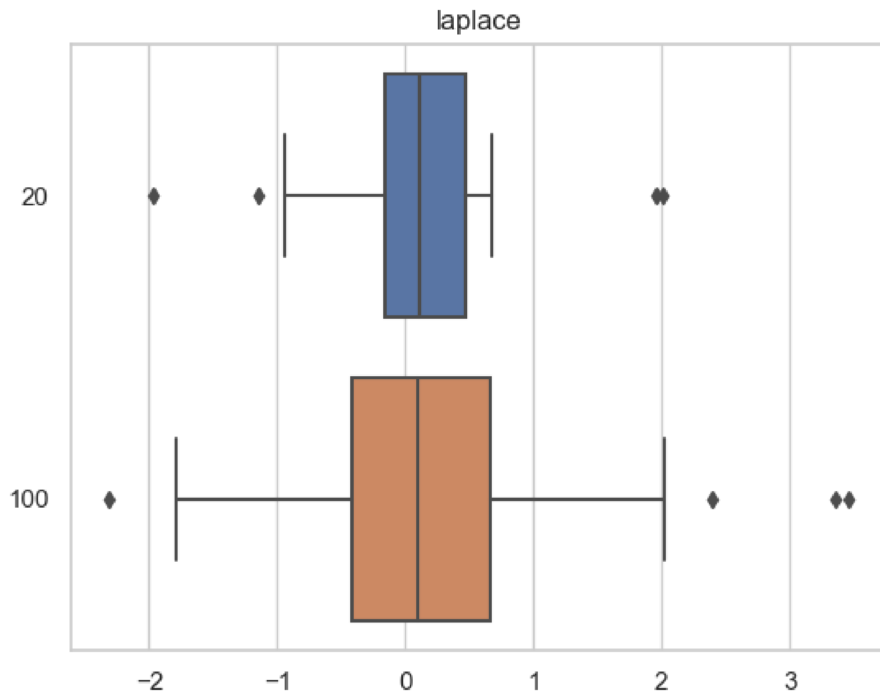

Рис. 3: Boxplot стандартное распределение Коши

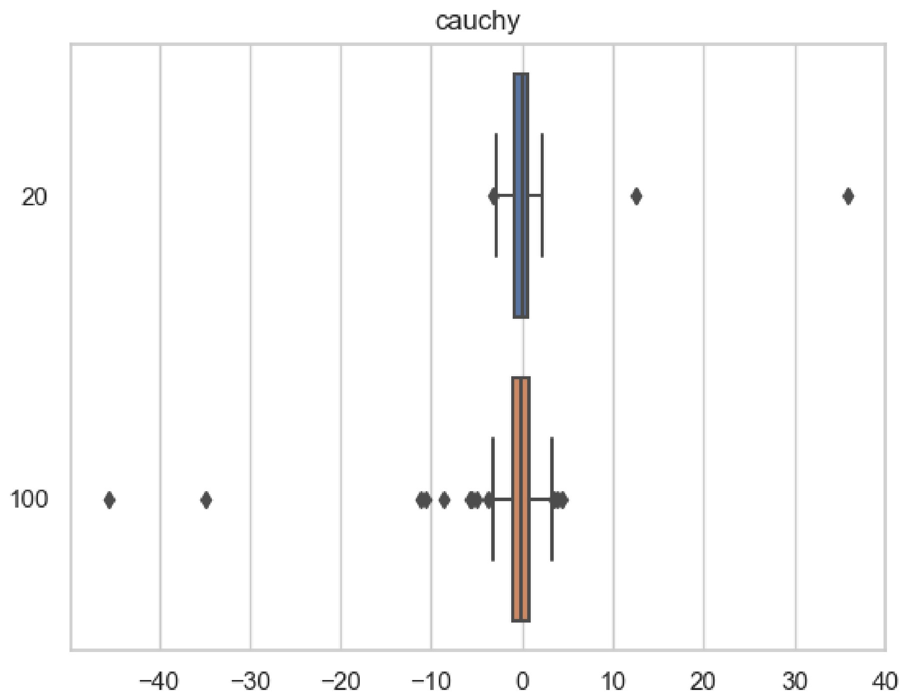

Рис. 4: Boxplot распределение Пуассона

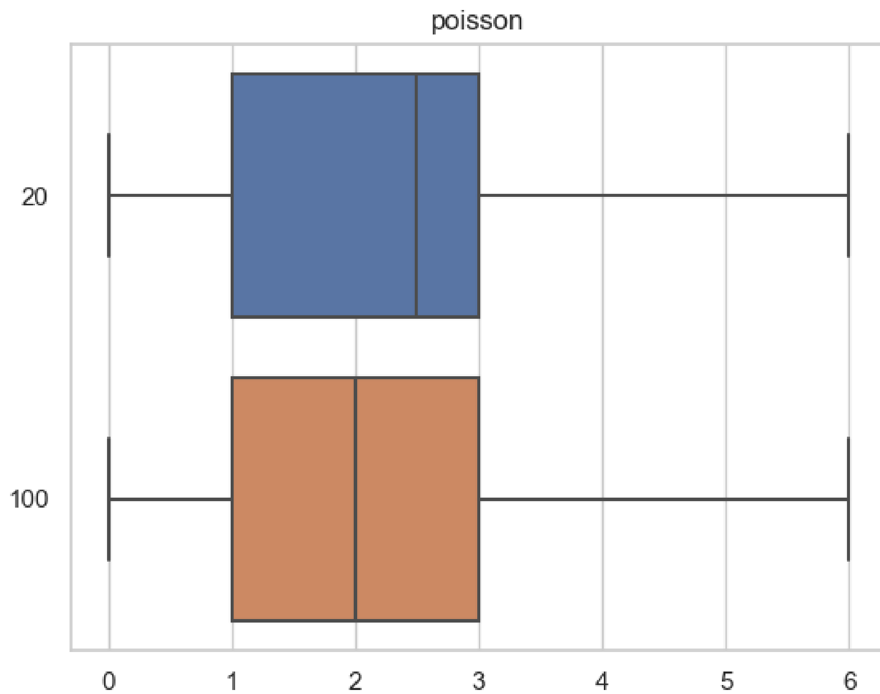

Рис. 5: Boxplot равномерное распределение

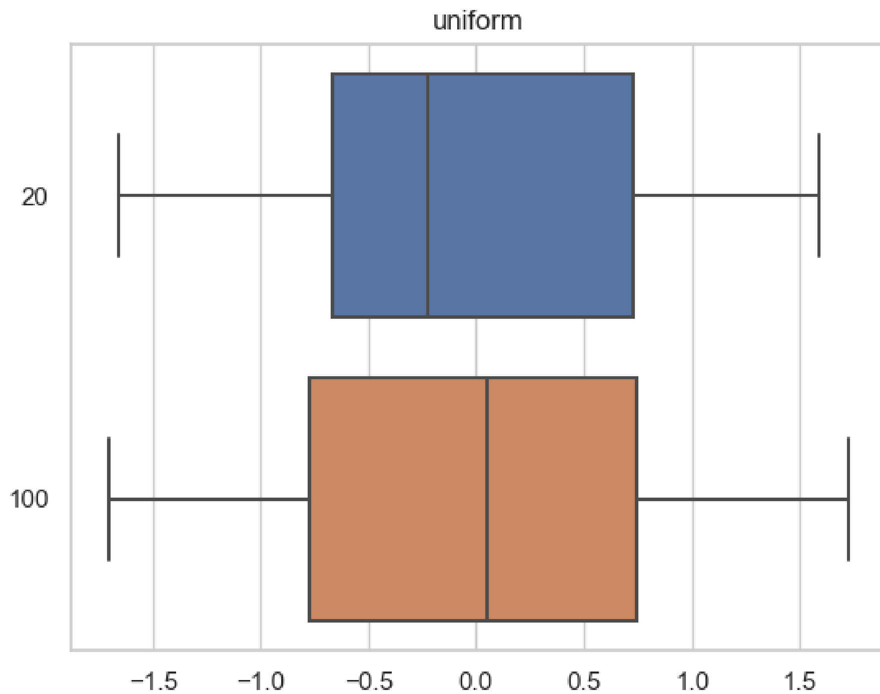


Таблица 1: Зависимость выбросов от размера выборки

| Выборка | Процент выбросов |
|---|---|
| normal | |
| n = 20 | 2 |
| n = 100 | 1 |
| cauchy | |
| n = 20 | 15 |
| n = 100 | 16 |
| laplace | |
| n = 20 | 7 |
| n = 100 | 6 |
| uniform | |
| n = 20 | 0 |
| n = 100 | 0 |
| poisson | |
| n = 20 | 3 |
| n = 100 | 0 |

# 7 Выводы

Экспериментально полученные проценты выбросок, близки к теоретическим Можно вывести соотношение между процентами выбросов:

$$uniform < normal < poisson < laplace < cauchy \tag{3}$$

По полученным данным видно, что наименьший процент выбросов у равномерного распределения, а наибольший процент выбросов у распределения Коши

# 8 Список литературы


[1] Выборочная медиана - http://femto.com.ua/articles/part_1/2194.html

[2] Квартили - https://studfiles.net/preview/2438125/page:13/

[3] Боксплот - https://en.wikipedia.org/wiki/Box_plot

[4] Модуль numpy - https://physics.susu.ru/vorontsov/language/numpy.html

[5] Модуль matplotlib - https://matplotlib.org/users/index.html


# 9 Приложения

Код отчёта: https://github.com/MisterProper9000/MatStatLabs/blob/master/MatStatLab3/MatStatLab3.tex

Код лаборатрной: https://github.com/MisterProper9000/MatStatLabs/blob/master/MatStatLab3/MatStatLab3.py

```python

import numpy
import plotly.graph_objs as go
import plotly.plotly as py
import seaborn as sns
import matplotlib.pyplot as mplot
import sys


POISSON_PARAM = 2
UNIFORM_LEFT = -numpy.sqrt(3)
UNIFORM_RIGHT = numpy.sqrt(3)
LAPLAS_COEF = numpy.sqrt(2)
selection = [20, 100]
selection = numpy.sort(selection)

def standart_normal(x):
    return (1 / numpy.sqrt(2*numpy.pi)) * numpy.exp(- x * x / 2)


def standart_cauchy(x):
    return 1 / (numpy.pi * (1 + x*x))


def laplace(x):
    return 1 / LAPLAS_COEF * numpy.exp (-LAPLAS_COEF * numpy.abs(x))


def uniform(x):
    flag2 = x <= UNIFORM_RIGHT
    flag1 = x >= UNIFORM_LEFT
```

```python
    return 1 / (UNIFORM_RIGHT - UNIFORM_LEFT) * flag1 * flag2


def poisson(x):
    k = POISSON_PARAM
    return (numpy.power(x, k) / numpy.math.factorial(k)) * numpy.exp(-x)


func_dict = {
    'normal': standart_normal,
    'cauchy': standart_cauchy,
    'laplace': laplace,
    'uniform': uniform,
    'poisson': poisson
}


def generate_laplace(x):
    return numpy.random.laplace(0, 1/LAPLAS_COEF, x)


def generate_uniform(x):
    return numpy.random.uniform(UNIFORM_LEFT, UNIFORM_RIGHT, x)


def generate_poisson(x):
    return numpy.random.poisson(POISSON_PARAM, x)


generate_dict = {
    'normal': numpy.random.standard_normal,
    'cauchy': numpy.random.standard_cauchy,
    'laplace': generate_laplace,
    'uniform': generate_uniform,
    'poisson': generate_poisson
}


def Zr(x):
    return (numpy.amin(x) + numpy.amax(x))/2


def Zq(x):
    return (numpy.quantile(x, 1/4) + numpy.quantile(x, 3/4)) / 2


def Ztr(x):
    length = x.size
    r = (int)(length / 4)
    sum = 0
    for i in range(r, length - r):
        sum += x[i]
    return sum/(length - 2 * r)


def IQR(x):
    return numpy.abs(numpy.quantile(x, 1 / 4) - numpy.quantile(x, 3 / 4))


def ejection(x):
    length = x.size
    count = 0
    left = numpy.quantile(x, 1 / 4) - 1.5 * IQR(x)
    right = numpy.quantile(x, 3 / 4) + 1.5 * IQR(x)
    for i in range(0, length):
        if(x[i] < left or x[i] > right):
```

```
            count += 1
    return count / length


pos_characteristic_dict = {
    'average': numpy.mean,
    'med': numpy.median,
    'Zr': Zr,
    'Zq': Zq,
    'Ztr r = n/4': Ztr
}

pos_char_name = [
    'average',
    'med',
    'Zr',
    'Zq',
    'Ztr r = n/4'
]


def E(z):
    return numpy.mean(z)


def D(z):
    return numpy.var(z)


f = open('out1.csv', 'w')
std = sys.stdout
sys.stdout = f

def research(dist_type):
   # print('-------------------------------------')
    print()
    print(dist_type)

    data = []

    for num in selection:
        eject = []
        arr = numpy.sort(generate_dict[dist_type](num))
        data.append(arr)

        for i in range(0, 1000):
            arr = numpy.sort(generate_dict[dist_type](num))
            eject.append(ejection(arr))

        print("%-10s;" % ('n = %i' % num), end="")
        print("%-12f;" % E(eject), end="")
        print()

    mplot.figure(dist_type)
    mplot.title(dist_type)
    sns.set(style="whitegrid")
    ax = sns.boxplot(data=data, orient='h')
    mplot.yticks(numpy.arange(2), ('20', '100'))
    mplot.show()




research('normal')
research('cauchy')
research('laplace')
```

```
research('uniform')
research('poisson')


f.close()
sys.stdout = std
print("Done")
```