Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Лабораторная работа 3

СПОСОБЫ ПОЛУЧЕНИЯ СЛУЧАЙНЫХ ЧИСЕЛ С ТРЕБУЕМЫМ ЗАКОНОМ РАСПРЕДЕЛЕНИЯ

(Часть 2. Непрерывные случайные величины)

Выполнил
студент гр. 3530904/00103 Плетнева А. Д.

Руководитель Чуркин В. В.

Санкт-Петербург
2023

Оглавление

Цель работы

1. Практическое освоение методов получения случайных величин, имеющих характер распределения.

2. Разработка программных датчиков дискретных случайных величин.

3. Оценка точности моделирования: вычисление математического ожидания и дисперсии, сравнение полученных оценок с соответствующими теоретическими значениями.

4. Графическое представление плотности распределения и интегральной функции распределения.

Ход работы

Равномерное распределение

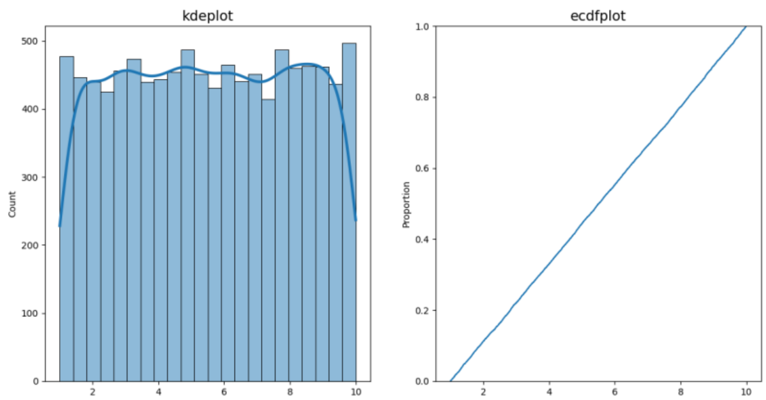
По следующей формуле получили 100000 случайных чисел от 1 до 10:

```
r = (b-a)*u + a ,
```

u - псевдослучайное вещественное число, равномерно распределенное в интервале [0,1], полученное с помощью программного датчика случайных чисел - np.random.random()
Посчитали для них математическое ожидание и дисперсию, сравнили с теоретическими значениями:

Оценка распределений	Эксперимент	Теоретическое значение	Отклонение
M	5.523077401447167	5.5	0.023077401447166856
D	6.796391630873283	6.75	0.04639163087328324

Построили графики распределения и плотности распределения:



Нормальное распределение

Получим 10000 случайных нормально распределенных чисел с помощью алгоритма, основанного на использовании центральной предельной теоремы:

```
z(12) = R(12) - 6
```

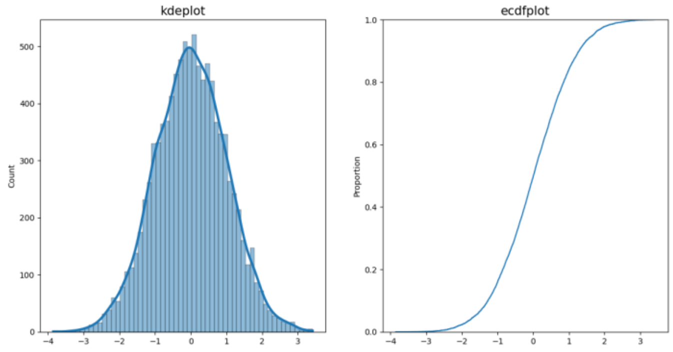
А также 10000 случайных нормально распределенных чисел с помощью алгоритма Бокс-Миллера (метод точного преобразования)

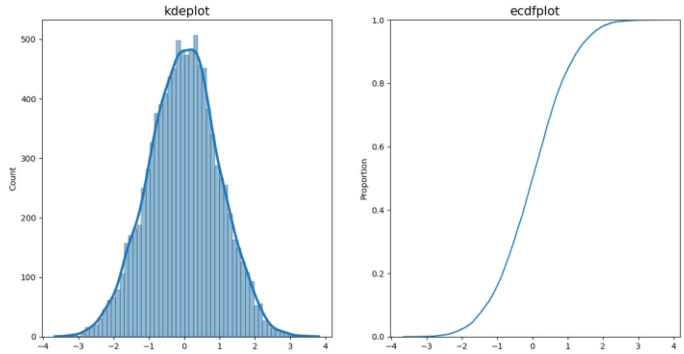
```
z(1) = sqrt( -2ln(u(2)) )* cos(2pi*u(1))
```

Посчитали для полученных чисел математическое ожидание и дисперсию, сравнили с теоретическими значениями:

Оценка распределений	Эксперимент1	Эксперимент2	Теоретическое значение	Отклонение1	Отклонение2
M	-0.008795901114992523	0.0022504433430112387	0	0.008795901114992523	0.0022504433430112387
D	1.0067087275972426	1.0160382194279967	1	0.006708727597242614	0.01603821942799666

Построили графики распределения и плотности распределения:





Экспоненциальное распределение

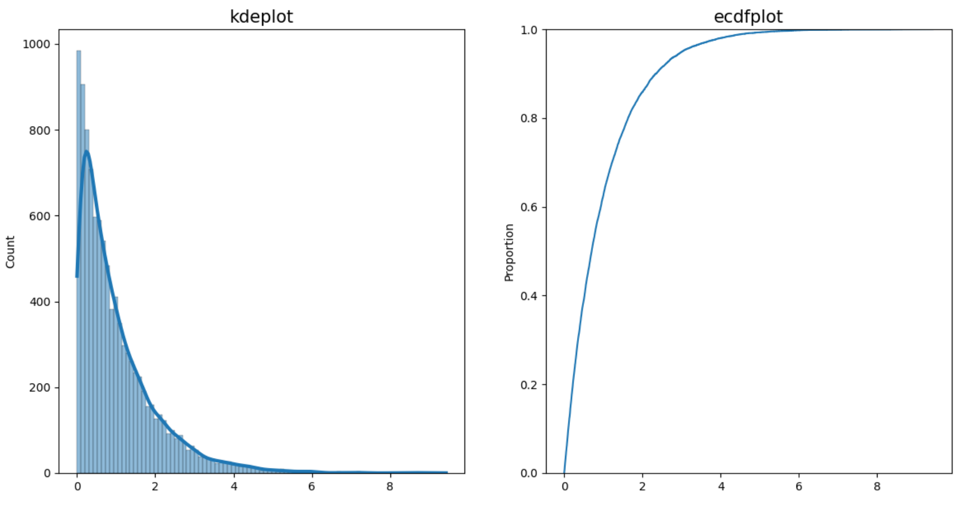
По следующей формуле получили 100000 случайных чисел с параметром beta=1:

```
x = -beta *ln(u)
```

u - псевдослучайное вещественное число, равномерно распределенное в интервале [0,1], полученное с помощью программного датчика случайных чисел - np.random.random()
Посчитали для них математическое ожидание и дисперсию, сравнили с теоретическими значениями:

Оценка распределений	Эксперимент	Теоретическое значение	Отклонение
M	1.0149768115140292	1	0.014976811514029187
D	1.0408281206904484	1	0.04082812069044839

Построили графики распределения и плотности распределения:



Хи-квадрат распределение

По следующей формуле получили 100000 случайных чисел с параметром N=10:

```
YN =z(1)**2 + z(2)**2 + ... + z(N)**2
```

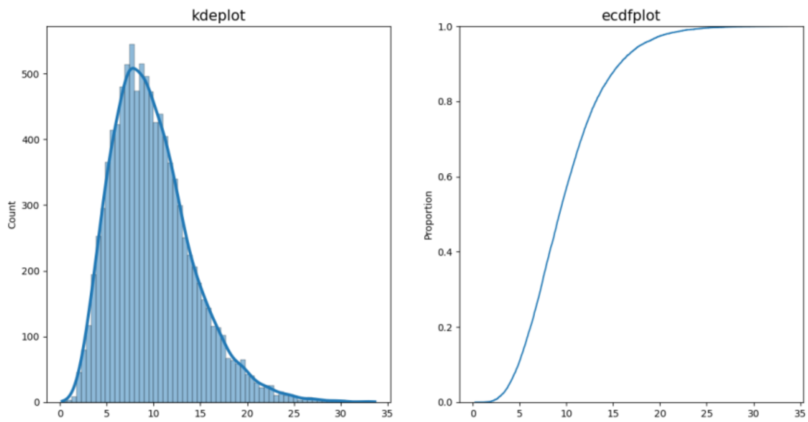
,

где z(i)- нормированно нормально распределенная N(0,1) псевдослучайная величина.

Посчитали для них математическое ожидание и дисперсию, сравнили с теоретическими значениями:

Оценка распределений	Эксперимент	Теоретическое значение	Отклонение
M	9.884008175110539	10	0.11599182488946091
D	19.104244448219806	20	0.8957555517801943

Построили графики распределения и плотности распределения:



Распределение Стьюдента

По следующей формуле получили 100000 случайных чисел с параметром N=10:

```
t = z/sqrt(Yn/N)
```

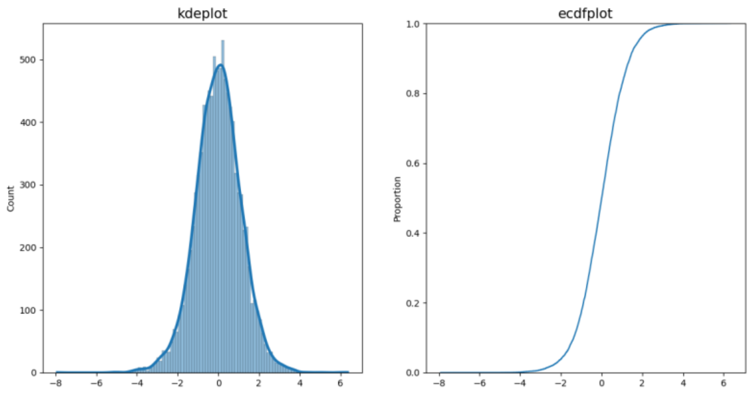
,

где z- нормированно нормально распределенная N(0,1) псевдослучайная величина, Yn - псевдослучайная величина, имеющая Хи-квадрат распределение

Посчитали для них математическое ожидание и дисперсию, сравнили с теоретическими значениями:

Оценка распределений	Эксперимент	Теоретическое значение	Отклонение
M	-0.0012220922344216027	0	0.0012220922344216027
D	1.278086101858261	1.25	0.028086101858260992

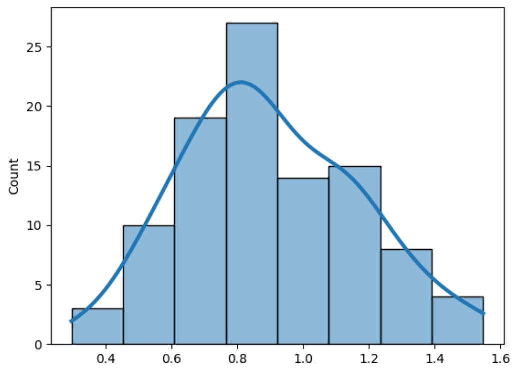
Построили графики распределения и плотности распределения:



Индивидуальное задание

4. Смоделировать случайную величину X, имеющую закон распределения Вейбулла с параметрами m=4 и λ=1 (алгоритм приведен в лекции №3). На основе выборки объема 100 исследовать статистические характеристики случайной величины X: построить гистограмму распределения, провести проверку согласия эмпирического распределения теоретическому критерием Колмогорова с уровнем значимости 0,1.

Сгенерируем выборку объемом 100 и построим гистограмму распределения:



Проводим тест Колмогорова, используя функцию kstest библиотеки scipy.stats, получаем значение максимальной абсолютной разницы между двумя функциями распределения - эмпирической и теоретической:

```
Критерий Колмогорова: статистика = 0.16880339557293594
```

По таблице квантилей распределения Колмогорова для уровня значимости α=0.1 и выборки размера n=100, критическое значение статистики Колмогорова равно 1.22.
Результат теста Колмогорова показывает, что гипотеза о том, что эмпирическое распределение выборки соответствует теоретическому распределению Вейбулла с параметрами m=4 и λ=1, не может быть отвергнута на уровне значимости 0,1.

Текст программы

Построение графиков

```
from matplotlib import pyplot as plt
import seaborn as sns


def get_plots(array):
    fig = plt.figure()

    ax_1 = fig.add_subplot(1, 2, 1)
    ax_2 = fig.add_subplot(1, 2, 2)

    sns.histplot(
        ax=ax_1,
        data=array,
        kde=True,
        bins=20,
        line_kws={"lw": 3})
    sns.ecdfplot(
        ax=ax_2,
        data=array)

    ax_1.set_title('kdeplot',
                   fontsize=15)
    ax_2.set_title('ecdfplot',
                   fontsize=15)

    fig.set_figwidth(14)
    fig.set_figheight(7)

    plt.show()
```

Экспоненциальное распределение

```
def uniform_dist(n, beta):
    r_array = []
    D = 0
    for i in range(n):
        r = -1 * beta * math.log(np.random.random())
        r_array.append(r)
    M = sum(r_array) / n
    for i in range(n):
        D += (r_array[i] - M) ** 2
    D /= n
    return r_array, M, D


n = 10000
beta = 1
array, M, D = uniform_dist(n, beta)
M_theor = 1
D_theor = 1
table = PrettyTable(['Оценка распределений', 'Эксперимент', 'Теоретическое
значение', 'Отклонение'])
table.add_row(["M", M, M_theor, abs(M - M_theor)])
table.add_row(["D", D, D_theor, abs(D - D_theor)])
print(table)

get_plots(array)
```

Хи-квадрат распределение

```
import math
```

```
import numpy as np
from prettytable import PrettyTable

from Lab_2.get_plots import get_plots


def uniform_dist(n, N):
    r_array = []
    D = 0
    for i in range(n):
        r = 0
        for i in range(N):
            a = np.random.random(2)
            m = math.sqrt(-2 * math.log(a[1])) * math.cos(2 * math.pi * a[0])
            r += m ** 2
        r_array.append(r)
    M = sum(r_array) / n
    for i in range(n):
        D += (r_array[i] - M) ** 2
    D /= n
    return r_array, M, D


n = 10000
N = 10
array, M, D = uniform_dist(n, N)
M_theor = 10
D_theor = 20
table = PrettyTable(['Оценка распределений', 'Эксперимент', 'Теоретическое
значение', 'Отклонение'])
table.add_row(["M", M, M_theor, abs(M - M_theor)])
table.add_row(["D", D, D_theor, abs(D - D_theor)])
print(table)

get_plots(array)
```

Нормальное распределение

```
import math

import numpy as np
from prettytable import PrettyTable

from Lab_2.get_plots import get_plots


def norm_dist_alg_1(n):
    r_array = []
    D = 0
    for i in range(n):
        a = np.random.random(12)
        m = sum(a) - 6
        r_array.append(m)
    M = sum(r_array) / n
    for i in range(n):
        D += (r_array[i] - M) ** 2
    D /= n
    return r_array, M, D


def norm_dist_alg_2(n):
    r_array = []
    D = 0
    for i in range(n):
        a = np.random.random(2)
        m = math.sqrt(-2*math.log(a[1]))*math.cos(2*math.pi*a[0])
```

```
        r_array.append(m)
    M = sum(r_array) / n
    for i in range(n):
        D += (r_array[i] - M) ** 2
    D /= n
    return r_array, M, D


n = 10000
array_1, M_1, D_1 = norm_dist_alg_1(n)
array_2, M_2, D_2 = norm_dist_alg_2(n)
M_theor = 0
D_theor = 1
table = PrettyTable(
    ['Оценка распределений', 'Эксперимент1', 'Эксперимент2', 'Теоретическое
значение', 'Отклонение1', 'Отклонение2'])
table.add_row(["M", M_1, M_2, M_theor, abs(M_1 - M_theor), abs(M_2 -
M_theor)])
table.add_row(["D", D_1, D_2, D_theor, abs(D_1 - D_theor), abs(D_2 -
D_theor)])
print(table)

get_plots(array_1)
get_plots(array_2)
```

Распределение Стьюдента

```
import math

import numpy as np
from prettytable import PrettyTable

from Lab_2.get_plots import get_plots


def uniform_dist(n, N):
    r_array = []
    D = 0
    for i in range(n):
        YN = 0
        for i in range(N):
            a = np.random.random(2)
            m = math.sqrt(-2 * math.log(a[1])) * math.cos(2 * math.pi * a[0])
            YN += m ** 2
        a = np.random.random(2)
        m = math.sqrt(-2 * math.log(a[1])) * math.cos(2 * math.pi * a[0])
        r = m / math.sqrt(YN / N)
        r_array.append(r)
    M = sum(r_array) / n
    for i in range(n):
        D += (r_array[i] - M) ** 2
    D /= n
    return r_array, M, D


n = 10000
N = 10
array, M, D = uniform_dist(n, N)
M_theor = 0
D_theor = 1.25
table = PrettyTable(['Оценка распределений', 'Эксперимент', 'Теоретическое
значение', 'Отклонение'])
table.add_row(["M", M, M_theor, abs(M - M_theor)])
table.add_row(["D", D, D_theor, abs(D - D_theor)])
print(table)
```

```
get_plots(array)
```

Равномерное распределение

```
import math

import numpy as np
from prettytable import PrettyTable

from Lab_2.get_plots import get_plots


def uniform_dist(n, a, b):
    r_array = []
    D = 0
    for i in range(n):
        r = (b - a) * np.random.random() + a
        r_array.append(r)
    M = sum(r_array) / n
    for i in range(n):
        D += (r_array[i] - M) ** 2
    D /= n
    return r_array, M, D


n = 10000
r_low = 1
r_up = 10
array, M, D = uniform_dist(n, r_low, r_up)
M_theor = (r_low + r_up) / 2
D_theor = ((r_up - r_low) ** 2) / 12
table = PrettyTable(['Оценка распределений', 'Эксперимент', 'Теоретическое значение', 'Отклонение'])
table.add_row(["M", M, M_theor, abs(M - M_theor)])
table.add_row(["D", D, D_theor, abs(D - D_theor)])
print(table)

get_plots(array)
```

Индивидуальное задание

```
import math

import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from scipy.stats import weibull_min, kstest


def weibull_dist(n, lam, m):
    r_array = []
    for i in range(n):
        r = (lam ** (-1)) * ((-math.log(np.random.random())) ** (1 / m))
        r_array.append(r)
    return r_array


n = 100
lam = 1
m = 4
array = weibull_dist(n, lam, m)

sns.histplot(
    data=array,
    kde=True,
    line_kws={"lw": 3})
```

```
plt.show()

x = np.linspace(0, 4, 100)
pdf = weibull_min.pdf(x, c=m, scale=1/lam)
ks_stat = kstest(array, weibull_min.cdf, args=(m, 0, lam))
print(f'Критерий Колмогорова: статистика = {ks_stat.statistic}')
```