МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

ИНСТИТУТ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ

Лабораторная работа 4

по курсу «**Теория вероятностей и математическая статистика, часть 2**»

ВАРИАНТ 46

Тема: **Проверка статистических гипотез о математическом ожидании и дисперсии нормального распределения случайных величин**

Выполнил:
Студент 3-го курса
Успенский А.А.

Группа: КМБО-03-19

МОСКВА – 2022

Задание

Задание 4-1.

Проверить гипотезу о равенстве математических ожиданий с использованием распределения Стьюдента при уровне значимости $\alpha = 0{,}05$ для всех трёх пар наблюдаемых нормально распределенных случайных величин, выборки которых находятся в столбцах двумерного массива $\{u_{i,j} \mid 1 \le i \le N, 1 \le j \le 3\}$.

Задание 4-2.

Проверить с использованием однофакторного дисперсионного анализа гипотезу о равенстве математических ожиданий при уровне значимости *0,05* трёх наблюдаемых нормально распределенных случайных величин, выборки которых находятся в столбцах двумерного массива $\{u_{i,j} \mid 1 \le i \le N, 1 \le j \le 3\}$.

Задание 4-3.

Проверить гипотезу о равенстве математических ожиданий при уровне значимости $\alpha = 0{,}05$ для всех трёх пар наблюдаемых нормально распределенных случайных величин, выборки которых находятся в столбцах двумерного массива U, с помощью функций, в которых реализован t-критерий Стьюдента:

для Octave pval = t_test_2 (X, Y) ;

для Python pval = scipy.stats.ttest_ind(X,Y, equal_var=True);

X, Y – произвольная пара столбцов массива U.

Задание 4-4.

Проверить гипотезу о равенстве математических ожиданий при уровне значимости $\alpha = 0{,}05$ для всех трёх пар наблюдаемых нормально распределенных случайных величин, выборки которых находятся в столбцах двумерного массива U, с помощью функций, в которых реализован t-критерий Уэлча:

для Octave pval = welch_test (X, Y) ;

для Python pval = scipy.stats.ttest_ind(X,Y, equal_var=False);

X, Y – произвольная пара столбцов массива U.

Задание 4-5.

Проверить гипотезу о равенстве математических ожиданий при уровне значимости $\alpha = 0{,}05$ для трёх наблюдаемых нормально распределенных случайных величин, выборки которых находятся в столбцах двумерного массива U, с помощью функций, в которых реализован однофакторный дисперсионный анализ:

для Octave $\text{pval} = \text{anova (U)}$; для Python

$\text{pval} = \text{scipy.stats.f\_oneway (X,Y,Z)}$; X, Y, Z – столбцы массива U.

Задание 4-6.

Проверить гипотезу о равенстве дисперсий при уровне значимости $\alpha = 0{,}05$ для всех трёх пар наблюдаемых нормально распределенных случайных величин, выборки которых находятся в столбцах двумерного массива U, с использованием распределения Фишера-Снедекора.

Задание 4-7.

Проверить гипотезу о равенстве дисперсий при уровне значимости $\alpha = 0{,}05$ для наблюдаемых нормально распределенных случайных величин, выборки которых находятся в столбцах двумерного массива U, с помощью функций, в которых реализован критерий Бартлетта:

для Octave $\text{pval} = \text{bartlett\_test (X,Y,Z)}$;

для Python $\text{pval} = \text{scipy.stats.bartlett (X,Y,Z)}$;

X, Y, Z – столбцы массива U.

Краткие теоретические сведения

Нормальное распределение

Характеристика	Значение
Математическое ожидание	a
Дисперсия	σ^2
Среднее квадратическое отклонение	σ
Мода	a
Медиана	a
Коэффициент асимметрии	**0**
Коэффициент эксцесса	**0**

Плотность

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{(-\frac{(x-a)^2}{2\sigma^2})}$$

Функция распределение

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{(-\frac{(t-a)^2}{2\sigma^2})} dt$$

Ряд распределения - структурная группировка с целью выделения характерных свойств и закономерностей изучаемой совокупности.

Математическое ожидание – понятие среднего значения случайной величины в теории вероятностей.

Дисперсия – отклонение величины от ее математического ожидания.

Среднеквадратическое отклонение – показатель рассеивания значений случайной величины относительно ее математического ожидания.

Математическое ожидание:

$$M(X) = \bar{x} = \frac{1}{N}\sum_{k=1}^{m} x_k^* n_k = \sum_{k=1}^{m} x_k^* w_k$$

Дисперсия с поправкой Шеппарда:

$$D(X) = s_B^2 = \sum_{i=1}^{m} (x_i^* - \bar{x})^2 \cdot w_i - \frac{h^2}{12}, \text{где } h = \frac{a_m - a_0}{m}$$

Среднее квадратическое отклонение:

$$\sigma = \bar{\sigma} = \sqrt{s_B^2}$$

При проверке гипотезы о равенстве математических ожиданий используются следующие формулы расчета характеристик выборок $\{x_1, ..., x_N\}$ и $\{y_1, ..., y_M\}$:

Выборочное среднее:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \ , \ \bar{y} = \frac{1}{M}\sum_{j=1}^{M} y_j \ .$$

Выборочная несмещенная дисперсия:

$$\overline{x^2} = \frac{1}{N}\sum_{i=1}^{N} x_i^2 \ , \ \overline{y^2} = \frac{1}{M}\sum_{j=1}^{M} y_j^2 \ ,$$

$$S_x^2 = \frac{N}{N-1}(\overline{x^2} - \bar{x}^2) \ , \ S_y^2 = \frac{M}{M-1}(\overline{y^2} - \bar{y}^2) \ .$$

Проверка гипотезы о равенстве математических ожиданий двух случайных величин по выборкам $\{x_1, ..., x_N\}$ и $\{y_1, ..., y_M\}$ с использованием распределения Стьюдента с числом степеней свободы $N+M-2$ проводится следующим образом.

Рассчитывается значение критерия $T_{N,M}$:

$$S_x^2(N-1) = N(\overline{x^2} - \bar{x}^2), \ S_y^2(M-1) = M(\overline{y^2} - \bar{y}^2),$$

$$T_{N,M} = \frac{\bar{x} - \bar{y}}{\sqrt{S_x^2(N-1) + S_y^2(M-1)}} \sqrt{\frac{MN(N+M-2)}{N+M}} \ .$$

Если гипотеза о равенстве математических ожиданий нормально распределенных случайных величин X и Y верна, то $T_{N,M}$ имеет распределение $t(N+M-2)$ – распределение Стьюдента с числом степеней свободы $N+M-2$.

По уровню значимости α находится критическое значение $t_{кр,\alpha}(N+M-2)$ распределения Стьюдента с числом степеней свободы $N+M-2$.

Вычисленное значение $T_{N,M}$ сравнивается с критическим значением двустороннего критерия $t_{кр,\alpha}(N+M-2)$: если $|T_{N,M}| \le t_{кр,\alpha}(N+M-2)$, то гипотеза о равенстве математических ожиданий принимается.

В данной лабораторной работе $M=N$ и рассматривается распределение $t(2N-2)$. Критическое значение $t_{кр,\alpha}(2N-2)$ распределения Стьюдента с числом степеней свободы $2N-2$ при уровне значимости α равно (для двустороннего критерия) $t_{1-\frac{\alpha}{2}}(2N-2)$ – квантили уровня $1-\frac{\alpha}{2}$ распределения Стьюдента с числом степеней свободы $2N-2$. Критическое значение $t_{кр,\alpha}(2N-2)$ можно найти с помощью функции языка программирования

(для Octave $t_{кр,\alpha}(2N-2)=\text{tinv(x,n)}$;

для Python $t_{кр,\alpha}(2N-2)=\text{scipy.stats.t.ppf(x,n)}$; $\text{x}=1-\frac{\alpha}{2}=0{,}975$; $\text{n}=2N-2$).

Проверка с использованием однофакторного дисперсионного анализа гипотезы о равенстве математических ожиданий трёх случайных величин по выборкам, являющимся столбцами массива $U=\{u_{i,j} \mid 1\le i\le N, 1\le j\le m\}$, $m=3$, проводится по следующей схеме.

Расчет общего среднего значения и групповых средних

$$\overline{u}=\frac{1}{Nm}\sum_{j=1}^{m}\sum_{i=1}^{N}u_{ij}\ ,\ \overline{u}_{\cdot j}=\frac{1}{N}\sum_{i=1}^{N}u_{ij},\ j=1,\ldots,m.$$

Расчет общей суммы квадратов отклонений $S_{общ}=\sum_{j=1}^{m}\sum_{i=1}^{N}(u_{ij}-\overline{u})^2$.

Расчет факторной суммы квадратов отклонений $S_{факт}=N\sum_{j=1}^{m}(u_{\cdot j}-\overline{u})^2$.

Расчет остаточной суммы квадратов отклонений

$$S_{ост}=\sum_{j=1}^{m}\sum_{i=1}^{N}(u_{ij}-u_{\cdot j})^2=S_{общ}-S_{факт}.$$

Расчет значения критерия: $F_{N,m}=\frac{s^2_{факт}}{s^2_{ост}}$, где $s^2_{факт}=\frac{S_{факт}}{m-1}$, $s^2_{ост}=\frac{S_{ост}}{m(N-1)}$.

Если гипотеза о равенстве математических ожиданий m нормально распределенных случайных величин верна, то $F_{N,m}$ имеет распределение Фишера-Снедекора с числом степеней свободы (k_1,k_2), $k_1=m-1$, $k_2=m(N-1)$.

Средства языка программирования

Для расчёта статистических исследований я использую язык Python. В программе расчёта используются следующие библиотеки:

NumPy — это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами.

Pandas – это библиотека для обработки и анализа данных Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временны́ми рядами.

Matplotlib – это библиотека для визуализации данных. Построение графиков диаграмм и гистограмм.

Scipy.stats – этот модуль содержит большое количество вероятностных распределений, а также растущую библиотеку статистических функций.

Стоит описать некоторые команды для генерации и визуализации данных:

sps.norm.cdf(x, loc, scale) – Функция распределения случайной непрерывной величины. Позволяет узнать значение функции в точке.;

sps.norm.ppf(x, loc, scale) – Плотность распределения случайной непрерывной величины. Позволяет узнать значение плотности в точке;

sps.ttest_ind (X, Y, equal_var=True/False) – Это проверка нулевой гипотезы о том, что две независимые выборки имеют одинаковые средние (ожидаемые) значения. Этот тест предполагает, что популяции по умолчанию имеют одинаковые дисперсии. Если True (по умолчанию), выполните стандартный независимый тест с двумя выборками, который предполагает равные дисперсии генеральной совокупности. Если False, выполните t-критерий Уэлча, который не предполагает равной дисперсии генеральной совокупности.

sps.f_oneway(X, Y, Z) – Однофакторный дисперсионный анализ проверяет нулевую гипотезу о том, что две или более групп имеют одинаковое среднее значение генеральной совокупности. Тест применяется к образцам из двух или более групп, возможно, с разными размерами.

sps.bartlett(X, Y, Z) – Тест Бартлетта проверяет нулевую гипотезу о том, что все входные выборки взяты из популяций с одинаковыми дисперсиями. Для выборок из значительно ненормальных популяций тест Левена более надежен.

pd.DataFrame – визуализация данных в виде таблицы. Используется для визуализации статистического ряда.

Результаты расчётов и выводы

Задание 4-1

Выборка из файла «MC_D3_Norm»:

N = 16, m = 3

7,31459	7,02140	4,63644
4,70914	3,77246	7,74293
6,34750	4,92469	8,77715
7,15112	3,69270	2,16711
3,26870	1,65286	3,02712
4,84853	7,58742	7,94400
3,59965	5,21455	3,23247
2,01106	2,20869	4,54032
4,62077	4,15108	6,73067
3,58187	3,32341	5,90394
4,34591	4,09193	8,37022
0,79825	3,60429	6,38193
0,55546	4,11112	8,80107
3,28208	3,89266	11,13768
7,88458	4,66372	3,55408
6,50959	2,81736	5,70413

Таблица 1.1

Столбцы	$\bar{x}$	$\bar{y}$	$\overline{x^2}$	$\overline{y^2}$	S_x^2	S_y^2	$T_{N,N}$
(1,2)	4.42680	4.17065	24.22689	19.59651	4.93903	2.34903	0.37954
(1,3)	4.42680	6.16570	24.22689	44.00885	4.93903	6.39248	-2.06629
(2,3)	4.17065	6.16570	19.59651	44.00885	2.34903	6.39248	-2.06629

Таблица 1.2

Столбцы	$\lvert T_{N,N} \rvert$	$t_{\text{кр},\alpha}(2N-2)$	Вывод
(1,2)	0.37954	2,04227	ВЕРНА
(1,3)	2.06629	2,04227	НЕВЕРНА
(2,3)	2.06629	2,04227	НЕВЕРНА

Таблица 1.3

Задание 4-2

$S_{\text{общ}}$	$S_{\text{факт}}$	$S_{\text{ост}}$	$S^2_{\text{факт}}$	$S^2_{\text{ост}}$	k_1	k_2	$F_{N,m}$
242.91287	37.70483	205.20804	18.85241	4.56018	2	45	4.13414

Таблица 2.1

$F_{N,m}$	α	$F_{\text{кр},\alpha}(k_1,k_2)$	Вывод
4.13414	0.05	3.20432	НЕВЕРНА

Таблица 2.2

Задание 4-3

Столбцы	p-value	α	Вывод
(1,2)	0.70696	0.05	ВЕРНА
(1,3)	0.04753	0.05	НЕВЕРНА
(2,3)	0.01131	0.05	НЕВЕРНА

Таблица 3.1

Задание 4-4

Столбцы	p-value	α	Вывод
(1,2)	0.70729	0.05	ВЕРНА
(1,3)	0.04768	0.05	НЕВЕРНА
(2,3)	0.01235	0.05	НЕВЕРНА

Таблица 4.1

Задание 4-5

p-value	α	Вывод
0.03348	0.05	НЕВЕРНА

Таблица 5.1

Задание 4-6

Столбцы	S_1^2	S_2^2	k_1	k_2	$F_{N,N}$
(1,2)	4.93903	2.34903	15	15	2.10258
(1,3)	4.93903	6.39248	15	15	1.29428
(2,3)	2.34903	6.39248	15	15	2.72133

Таблица 6.1

Столбцы	$F_{N,N}$	α	Вывод
(1,2)	2.10258	4.00850	ВЕРНА
(1,3)	1.29428	4.00850	ВЕРНА
(2,3)	2.72133	4.00850	ВЕРНА

Таблица 6.2

Задание 4-7

p-value	α	Вывод
0.16689	0.05	ВЕРНА

Таблица 7.1

Список литературы

1) Математическая статистика [Электронный ресурс]: метод. указания по выполнению лаб. работ / А.А. Лобузов – М.: МИРЭА, 2017.

2) Гмурман В.Е. Теория вероятностей и математическая статистика. – М.:Юрайт, 2020.

3) Ивченко Г.И., Медведев Ю.И. Математическая статистика. – М.: URSS, 2020.

Приложение

```
1. import scipy.stats as sps
2. import numpy as np
3. import pandas as pd
4. import matplotlib.pyplot as plt
5. import pylab
6. import statistics
7. import cmath
8. import seaborn as sns
9. from matplotlib import ticker
10.     from math import factorial
11.     from math import exp
12.     from math import log
13.
14.     def Normrnd(a,sd,size):
15.         distr = sps.norm.rvs(loc=a, scale=sd, size=200)
16.         distr.sort()
17.         print (
18.             '\nПараметр a = ', a,
19.            '\nПараметр sd = ', sd,)
20.         print ('Нормальное распределение:\n',distr)
21.         return (distr)
22.
23.     def Exprnd(lm,size):
24.         distr = sps.expon.rvs(scale=1/lm, size=size)
25.         distr.sort()
26.         print (
27.             '\nПараметр lambda = ', lm,)
28.         print ('Показательное распределение:\n',distr)
29.         return (distr)
30.
31.     def Uniformrnd(a,b,size):
32.         distr = sps.uniform.rvs(loc=a,scale=6,size=size)
33.         distr.sort()
34.         print (
35.             '\nПараметр a = ', a,
36.            '\nПараметр b = ', b,)
37.         print ('Равномерное распределение:\n',distr)
38.         return (distr)
39.
40.
41.     def Xi(distr, size):
42.         xi = []
43.         for i in range(size):
44.             if distr[i]!=distr[i-1]:
45.                 xi.append(distr[i])
```

```
46.         return(xi)
47.
48.     def Freq(distr, xi, size):
49.         ni=[]
50.         for j in range(len(xi)):
51.             help = 0
52.             for i in range(size):
53.                 if distr[i]==xi[j]:
54.                     help += 1
55.             ni.append(help)
56.         return(ni)
57.
58.     def Rel_freq(ni, size):
59.         wi = []
60.         for i in range(len(ni)):
61.             wi.append(ni[i]/size)
62.         return(wi)
63.
64.     def koef_srfq(wi, k):
65.         S=0
66.         for i in range (k+1):
67.             S = S + wi[i]
68.         return(S)
69.
70.     def Sum_rfq(wi, k):
71.         sk = []
72.         for i in range (len(wi)):
73.             sk.append(koef_srfq(wi,i))
74.         return(sk)
75.
76.     def intervals(xi, size):
77.         m = 1+round(log(size,2))
78.         d = xi[-1] - xi[0]
79.         a = [xi[0]]
80.         for i in range (1,(m+1)):
81.             a.append(d/m+a[i-1])
82.         return a
83.
84.     def Freq_inter(distr, inter, size):
85.         ni=[]
86.         for j in range(1,len(inter)):
87.             count = 0
88.             h=0
89.             for i in range(size):
90.                 if j-1 == 0:
91.                     if distr[i]>=inter[j-1] and
  distr[i]<=inter[j]:
92.                         count +=1
```

```
93.                 elif j == len(inter)-1:
94.                     if distr[i]>inter[j-1] and
   distr[i]<=inter[j]:
95.                         count += 1
96.                     if distr[i]>inter[j]:
97.                         count += 1
98.                 else:
99.                     if distr[i]>inter[j-1] and
   distr[i]<=inter[j]:
100.                        count += 1
101.            ni.append(count)
102.        return(ni)
103.
104.    def X_X(inter):
105.        x_x = []
106.        for i in range(len(inter)-1):
107.            x_x.append((inter[i]+inter[i+1])/2)
108.        return(x_x)
109.
110.
111.    def Mean(xi ,wi):
112.        X = 0
113.        for i in range (len(xi)):
114.            help = xi[i]*wi[i]
115.            X += help
116.        return (X)
117.
118.    def DispSh (inter,m,xi,wi,mean):
119.        h = (inter[-1]-inter[0])/m
120.        s=xi
121.        for i in range (len(xi)):
122.            s[i]=((xi[i]-mean)**2)*wi[i]-(h**2)/12
123.
124.        return (sum(s))
125.
126.    def Mode(xi,wi):
127.        max_w = max(wi)
128.        h = (inter[-1]-inter[0])/m
129.        for k in range(len(wi_inter)-1):
130.            mode = xi[k]
131.            flag += 1
132.        return(round(mode, 5))
133.
134.    def Median(xi,sk):
135.        for k in range(len(xi)):
136.            if k == 0:
137.                if sk[k] > 0.5:
138.                    md = round(xi[k], 5)
```

```
                    return (md)
                if sk[k] == 0.5:
                    md = (round(xi[k]+xi[k+1])/2, 5)
                    return (md)
            if (sk[k] > 0.5) and (sk[k-1] < 0.5):
                md = round(xi[k], 5)
                return (md)
            if sk[k] == 0.5:
                md= (round(xi[k]+xi[k+1])/2, 5)
                return (md)

    def Moment(xi,wi):
        M = []
        for i in range (len(xi)):
            help = 0
            for j in range (len(xi)):
                help += (xi[j]**i)*wi[j]
            M.append(help)
        return (M)

    def Asym_coef(M, sd):
        asym = (M[2]-3*M[1]*M[0]+2*pow(M[0], 3))/(pow(sd, 3))
        return (asym)

    def Exe_coef(M, sd):
        help = M[3]-4*M[2]*M[0]+6*M[1]*pow(M[0], 2)-3*pow(M[0],
4)
        exe = help/(pow(sd, 4))-3
        return(exe)

    v = 46
    a_norm = round((-1)**v*0.1*v,2)
    sigma = 0.005*v+1
    size=200
    xi = []
    ni = []
    wi = []
    sk = []
    #xi - значение
    #ni - кол-во значения
    #sum(ni) = size
    #wi = ni/size
    #sk = sum(wj)

    distr1 = Normrnd(a_norm,sigma,size)
    xi = Xi(distr1, size)
    ni = Freq(distr1, xi, size)
    wi= Rel_freq(ni, size)
```

```
186.     sk = Sum_rfq(wi, size)
187.
188.     #Distr2 = Geomerty(p,size)
189.     #Distr3 = Poisson(l,size)
190.
191.     inter = intervals(xi,size)
192.     inter
193.
194.     #inter = [[a[0],a[1]], ... , [a[n-1],a[n]]]
195.
196.     interval_all = [[inter[0], inter[1]],
197.                     [inter[1], inter[2]],
198.                     [inter[2], inter[3]],
199.                     [inter[3], inter[4]],
200.                     [inter[4], inter[5]],
201.                     [inter[5], inter[6]],
202.                     [inter[6], inter[7]],
203.                     [inter[7], inter[8]],
204.                     [inter[8], inter[9]],]
205.     print (interval_all)
206.
207.     ni_inter = Freq_inter(distr1, inter, size)
208.     print (ni_inter)
209.     print (sum(ni_inter))
210.
211.     wi_inter = Rel_freq(ni_inter,size)
212.     print (wi_inter)
213.     print (sum(wi_inter))
214.
215.     x_x = X_X(inter)
216.
217.     Table = pd.DataFrame({'center_Intervals':x_x, 'Ni':ni_inter,
    'Wi':wi_inter})
218.     Table
219.
220.     # график функции эмпирического распределения
221.     plt.figure(figsize=(12, 8))
222.     fig, sec = plt.subplots()
223.     sec.xaxis.set_major_locator(ticker.MultipleLocator(0.5))
224.     sec.yaxis.set_major_locator(ticker.MultipleLocator(0.1))
225.     for k in range(len(xi) - 1):
226.         sec.plot([xi[k], xi[k+1]], [sk[k], sk[k]])
227.     plt.title("Эмпирическая функция распределения")
228.     plt.grid(True)
229.     plt.show()
230.
231.     m = 1+round(log(size,2))
232.     plt.figure(figsize=(12, 7))
```

```
233.     sns.distplot(distr1, bins = m)
234.     plt.title('Гистограмма относительных частот')
235.
236.     data = pd.DataFrame({'distr1': distr1})
237.
238.     # Выборочное среднее (теор/эксп)
239.     X_teor = a_norm
240.     X = Mean(x_x,wi_inter)
241.     print (X_teor, '\n', distr1.mean(), '\n')
242.
243.     #Выборочный момент
244.     M = Moment(x_x,wi_inter)
245.
246.     # Выборочная дисперсия с поправкой Шепарда (теор/эксп)
247.     D_teor = sigma**2
248.     D = DispSh(inter,m,x_x,wi_inter,X)
249.     print (D_teor,'\n', distr1.var(), '\n')
250.
251.     #Выборочное среднеквадратическое откл.(теор/эксп)
252.     SD_teor = sigma
253.     print (SD_teor, '\n', distr1.std(),'\n')
254.
255.     #Выборочная мода (теор/эксп)
256.     mode_teor = a_norm
257.     print (mode_teor, '\n','\n')
258.
259.     #Выборочная медиана (теор/эксп)
260.     median_teor = a_norm
261.     print (median_teor, '\n', data.describe(percentiles=
    [.50]).iloc[4,0],'\n')
262.
263.     #Выборочный коэф.асимметрии (теор/эксп)
264.     asym_teor = 0
265.     print (asym_teor, '\n', data.skew()[0],'\n')
266.
267.     #Выборочный коэф. эксцесса (теор/эксп)
268.     exe_teor = 0
269.     exe = Exe_coef(M, distr1.std())
270.     print (exe_teor, '\n', data.kurtosis()[0],'\n')
271.
272.     lm = 3 + (-1)**v * 0.01*v
273.     xi2 = []
274.     ni2 = []
275.     wi2 = []
276.     sk2 = []
277.     #xi - значение
278.     #ni - кол-во значения
279.     #sum(ni) = size
```

```
    #wi = ni/size
    #sk = sum(wj)

    distr2 = Exprnd(lm,size)
    xi2 = Xi(distr2, size)
    ni2 = Freq(distr2, xi2, size)
    wi2 = Rel_freq(ni2, size)
    sk2 = Sum_rfq(wi2, size)

    inter2 = intervals(xi2,size)
    inter2

    #inter = [[a[0],a[1]], ... , [a[n-1],a[n]]]

    ni_inter2 = Freq_inter(distr2, inter2, size)
    print (ni_inter2)
    print (sum(ni_inter2))

    wi_inter2 = Rel_freq(ni_inter2,size)
    print (wi_inter2)
    print (sum(wi_inter2))

    # график функции эмпирического распределения
    plt.figure(figsize=(12, 8))
    fig, sec = plt.subplots()
    sec.xaxis.set_major_locator(ticker.MultipleLocator(0.5))
    sec.yaxis.set_major_locator(ticker.MultipleLocator(0.1))
    for k in range(len(xi2) - 1):
        sec.plot([xi2[k], xi2[k+1]], [sk2[k], sk2[k]])
    plt.title("Эмпирическая функция распределения")
    plt.grid(True)
    plt.show()

    m = 1+round(log(size,2))
    plt.figure(figsize=(12, 7))
    sns.distplot(distr2, bins = m)
    plt.title('Гистограмма относительных частот')

    data2 = pd.DataFrame({'distr2': distr2})

    # Выборочное среднее (теор/эксп)
    X_teor2 = lm**(-1)
    X2 = Mean(x_x2,wi_inter2)
    print (X_teor2, '\n', distr2.mean(), '\n')

    #Выборочный момент
    M2 = Moment(x_x2,wi_inter2)

```

```
328.     # Выборочная дисперсия с поправкой Шепарда (теор/эксп)
329.     D_teor2 = lm**(-2)
330.     D2 = DispSh(inter2,m,x_x2,wi_inter2,X2)
331.     print (D_teor2,'\n', distr2.var(), '\n')
332.
333.     #Выборочное среднеквадратическое откл.(теор/эксп)
334.     SD_teor2 = lm**(-1)
335.     print (SD_teor2, '\n', distr2.std(),'\n')
336.
337.     #Выборочная мода (теор/эксп)
338.     mode_teor2 = 0
339.     print (mode_teor2, '\n','\n')
340.
341.     #Выборочная медиана (теор/эксп)
342.     median_teor2 = log(2)/lm
343.     print (median_teor2, '\n', data2.describe(percentiles=
    [.50]).iloc[4,0],'\n')
344.
345.     #Выборочный коэф.асимметрии (теор/эксп)
346.     asym_teor2 = 2
347.     print (asym_teor2, '\n', data2.skew()[0],'\n')
348.
349.     #Выборочный коэф. эксцесса (теор/эксп)
350.     exe_teor2 = 6
351.     exe2 = Exe_coef(M2, distr2.std())
352.     print (exe_teor2, '\n', data2.kurtosis()[0],'\n')
353.
354.     a_unif = (-1)**v *0.02*v
355.     b_unif = a_unif + 6
356.     xi3 = []
357.     ni3 = []
358.     wi3 = []
359.     sk3 = []
360.     #xi - значение
361.     #ni - кол-во значения
362.     #sum(ni) = size
363.     #wi = ni/size
364.     #sk = sum(wj)
365.
366.     distr3 = Uniformrnd(a_unif,b_unif,size)
367.     xi3 = Xi(distr3, size)
368.     ni3 = Freq(distr3, xi3, size)
369.     wi3 = Rel_freq(ni3, size)
370.     sk3 = Sum_rfq(wi3, size)
371.
372.
373.     # график функции эмпирического распределения
374.     plt.figure(figsize=(12, 8))
```

```
    fig, sec = plt.subplots()
    sec.xaxis.set_major_locator(ticker.MultipleLocator(0.5))
    sec.yaxis.set_major_locator(ticker.MultipleLocator(0.1))
    for k in range(len(xi3) - 1):
        sec.plot([xi3[k], xi3[k+1]], [sk3[k], sk3[k]])
    plt.title("Эмпирическая функция распределения")
    plt.grid(True)
    plt.show()

    m = 1+round(log(size,2))
    plt.figure(figsize=(12, 7))
    sns.distplot(distr3, bins = m)
    plt.title('Гистограмма относительных частот')

    data3 = pd.DataFrame({'distr3': distr3})

    # Выборочное среднее (теор/эксп)
    X_teor3 = (a_unif+b_unif)/2
    X3 = Mean(x_x3,wi_inter3)
    print (X_teor3, '\n', distr3.mean(), '\n')

    #Выборочный момент
    M3 = Moment(x_x3,wi_inter3)

    # Выборочная дисперсия с поправкой Шепарда (теор/эксп)
    D_teor3 = (b_unif - a_unif)**2/12
    D3 = DispSh(inter3,m,x_x3,wi_inter3,X3)
    print (D_teor3,'\n', distr3.var(), '\n')

    #Выборочное среднеквадратическое откл.(теор/эксп)
    SD_teor3 = (b_unif - a_unif)/(2*3**(1/2))
    print (SD_teor3, '\n', distr3.std(),'\n')

    #Выборочная мода (теор/эксп)
    mode_teor3 = (a_unif + b_unif)/2
    print (mode_teor3, '\n','\n')

    #Выборочная медиана (теор/эксп)
    median_teor3 = (a_unif + b_unif)/2
    print (median_teor3, '\n', data3.describe(percentiles=
  [.50]).iloc[4,0],'\n')

    #Выборочный коэф.асимметрии (теор/эксп)
    asym_teor3 = 0
    print (asym_teor3, '\n', data3.skew()[0],'\n')

    #Выборочный коэф. эксцесса (теор/эксп)
    exe_teor3 = -6/5
```

```
422.     exe3 = Exe_coef(M3, distr3.std())
423.     print (exe_teor3, '\n', data3.kurtosis()[0],'\n')
424.
```