|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | |  |  |  | | 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:

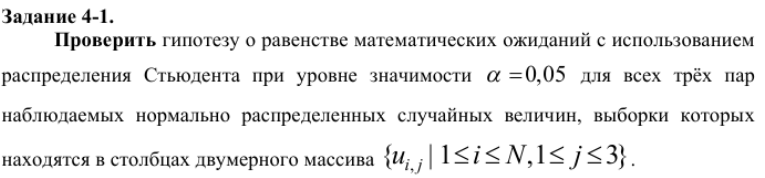
Студент 3-го курса

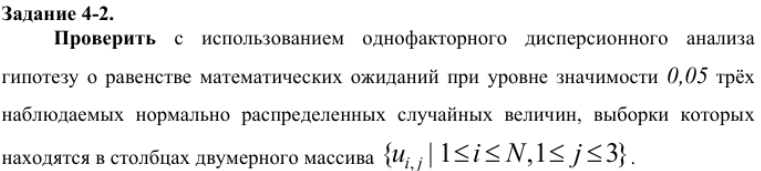
Успенский А.А.

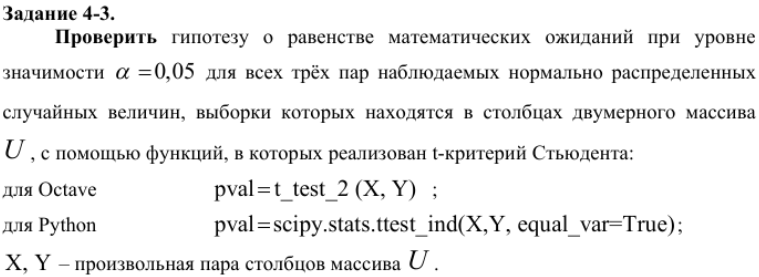
Группа: КМБО-03-19

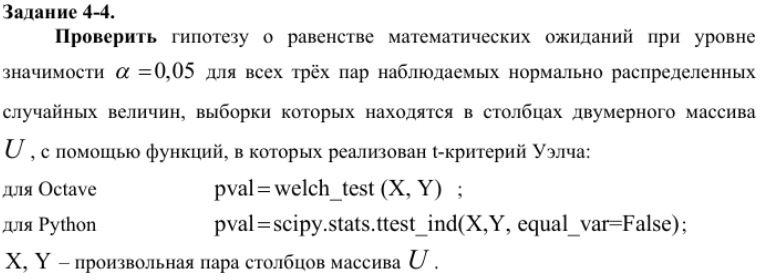
МОСКВА – 2022

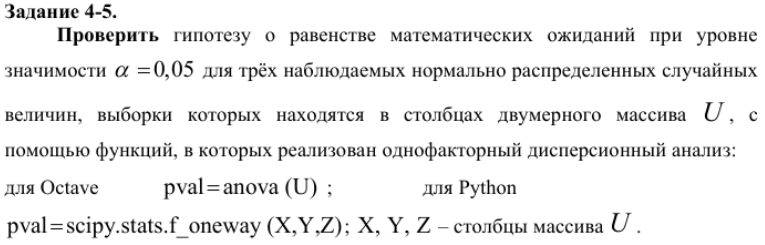
# **Задание**

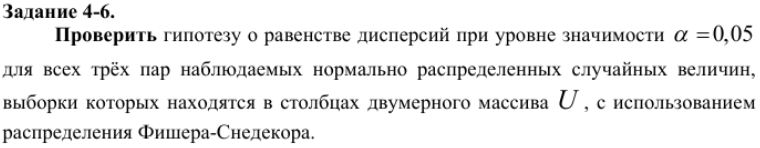
****

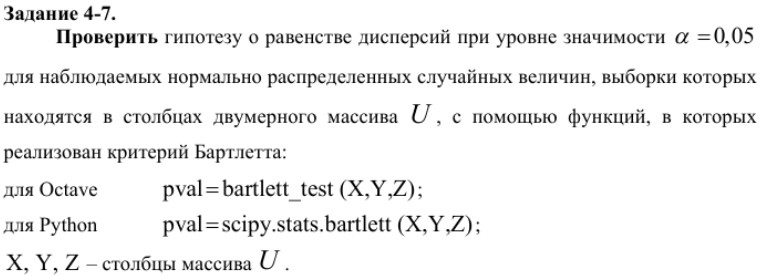
****

****

****

****

****

****

**Краткие теоретические сведения**

# **Нормальное распределение**



***Плотность***

***Функция распределение***

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

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

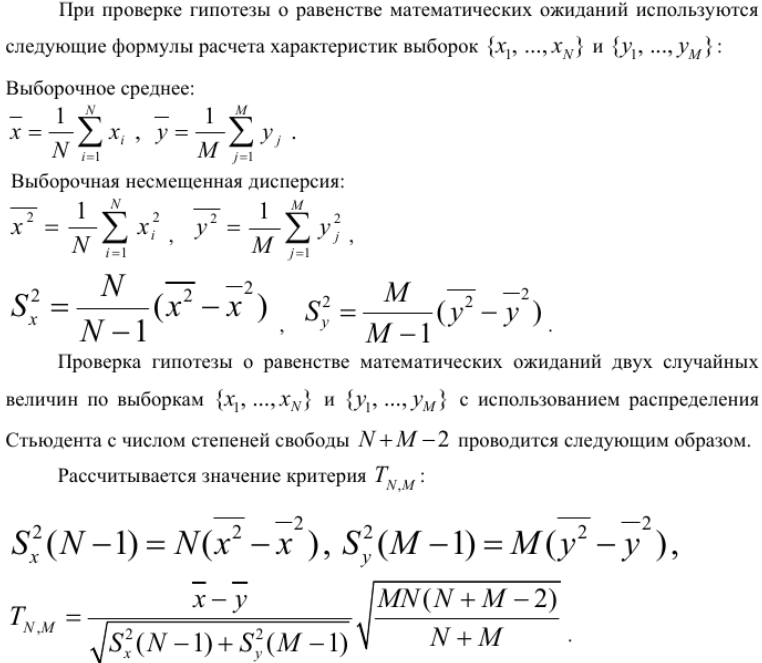
Дисперсия – отклонение величины от ее математического ожидания.

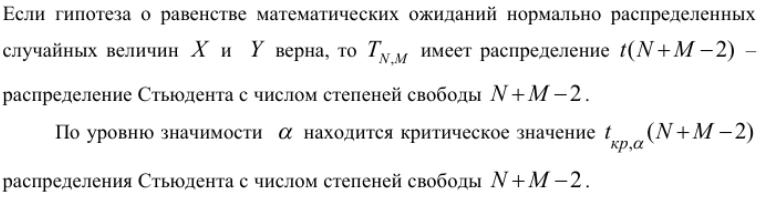
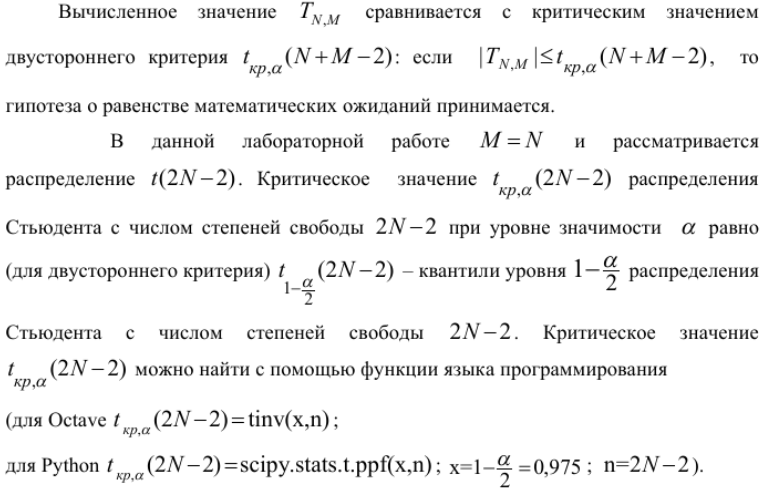
Среднеквадратическое отклонение – показатель рассеивания значений случайной величины относительно ее математического ожидания.

**Математическое ожидание**:

**Дисперсия с поправкой Шеппарда**:

**Среднее квадратическое отклонение**:





# 

# **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

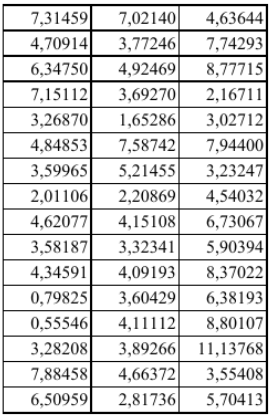
****

Таблица 1.1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Столбцы |  |  |  |  |  |  |  |
| (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 |  |  | 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**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| 242.91287 | 37.70483 | 205.20804 | 18.85241 | 4.56018 | 2 | 45 | 4.13414 |

Таблица 2.1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Вывод |
| 4.13414 | 0.05 | 3.20432 | НЕВЕРНА |

Таблица 2.2

## **Задание 4-3**

|  |  |  |  |
| --- | --- | --- | --- |
| Столбцы | p-value |  | 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 |  | 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 |  | Вывод |
| 0.03348 | 0.05 | НЕВЕРНА |

Таблица 5.1

## **Задание 4-6**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Столбцы |  |  |  |  |  |
| (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 |  |  | 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 |  | 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
14. **def** Normrnd(a,sd,size):
15. **distr = sps.norm.rvs(loc=a, scale=sd, size=200)**
16. distr.sort()
17. **print** (
18. '**\n**Параметр a = ', a,
19. '**\n**Параметр sd = ', sd,)
20. **print ('Нормальное распределение:\n',distr)**
21. **return** (distr)
23. **def** Exprnd(lm,size):
24. distr = sps.expon.rvs(scale=1/lm, size=size)
25. **distr.sort()**
26. **print** (
27. '**\n**Параметр lambda = ', lm,)
28. **print** ('Показательное распределение:**\n**',distr)
29. **return** (distr)
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. '**\n**Параметр b = ', b,)
37. **print** ('Равномерное распределение:**\n**',distr)
38. **return** (distr)

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)
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)
58. **def** Rel\_freq(ni, size):
59. wi = []
60. **for i in range(len(ni)):**
61. wi.append(ni[i]/size)
62. **return**(wi)
64. **def** koef\_srfq(wi, k):
65. **S=0**
66. **for** i **in** range (k+1):
67. S = S + wi[i]
68. **return**(S)
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)
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
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)
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)

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)
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
124. **return** (sum(s))
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))
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)
139. **return** (md)
140. **if sk[k] == 0.5:**
141. md = (round(xi[k]+xi[k+1])/2, 5)
142. **return** (md)
143. **if** (sk[k] > 0.5) **and** (sk[k-1] < 0.5):
144. md = round(xi[k], 5)
145. **return (md)**
146. **if** sk[k] == 0.5:
147. md= (round(xi[k]+xi[k+1])/2, 5)
148. **return** (md)
150. **def Moment(xi,wi):**
151. M = []
152. **for** i **in** range (len(xi)):
153. help = 0
154. **for** j **in** range (len(xi)):
155. **help += (xi[j]\*\*i)\*wi[j]**
156. M.append(help)
157. **return** (M)
159. **def** Asym\_coef(M, sd):
160. **asym = (M[2]-3\*M[1]\*M[0]+2\*pow(M[0], 3))/(pow(sd, 3))**
161. **return** (asym)
163. **def** Exe\_coef(M, sd):
164. help = M[3]-4\*M[2]\*M[0]+6\*M[1]\*pow(M[0], 2)-3\*pow(M[0], 4)
165. **exe = help/(pow(sd, 4))-3**
166. **return**(exe)
168. v = 46
169. a\_norm = round((-1)\*\*v\*0.1\*v,2)
170. **sigma = 0.005\*v+1**
171. size=200
172. xi = []
173. ni = []
174. wi = []
175. **sk = []**
176. *#xi - значение*
177. *#ni - кол-во значения*
178. *#sum(ni) = size*
179. *#wi = ni/size*
180. ***#sk = sum(wj)***
182. distr1 = Normrnd(a\_norm,sigma,size)
183. xi = Xi(distr1, size)
184. ni = Freq(distr1, xi, size)
185. **wi= Rel\_freq(ni, size)**
186. sk = Sum\_rfq(wi, size)
188. *#Distr2 = Geomerty(p,size)*
189. *#Distr3 = Poisson(l,size)*
191. inter = intervals(xi,size)
192. inter
194. *#inter = [[a[0],a[1]], ... , [a[n-1],a[n]]]*
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)**
207. ni\_inter = Freq\_inter(distr1, inter, size)
208. **print** (ni\_inter)
209. **print** (sum(ni\_inter))
211. wi\_inter = Rel\_freq(ni\_inter,size)
212. **print** (wi\_inter)
213. **print** (sum(wi\_inter))
215. **x\_x = X\_X(inter)**
217. Table = pd.DataFrame({'center\_Intervals':x\_x, 'Ni':ni\_inter, 'Wi':wi\_inter})
218. Table
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()
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')
236. data = pd.DataFrame({'distr1': distr1})
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**')
243. *#Выборочный момент*
244. M = Moment(x\_x,wi\_inter)
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**')
251. *#Выборочное среднеквадратическое откл.(теор/эксп)*
252. SD\_teor = sigma
253. **print** (SD\_teor, '**\n**', distr1.std(),'**\n**')
255. ***#Выборочная мода (теор/эксп)***
256. mode\_teor = a\_norm
257. **print** (mode\_teor, '**\n**','**\n**')
259. *#Выборочная медиана (теор/эксп)*
260. **median\_teor = a\_norm**
261. **print** (median\_teor, '**\n**', data.describe(percentiles= [.50]).iloc[4,0],'**\n**')
263. *#Выборочный коэф.асимметрии (теор/эксп)*
264. asym\_teor = 0
265. **print (asym\_teor, '\n', data.skew()[0],'\n')**
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')**
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*
280. ***#wi = ni/size***
281. *#sk = sum(wj)*
283. distr2 = Exprnd(lm,size)
284. xi2 = Xi(distr2, size)
285. **ni2 = Freq(distr2, xi2, size)**
286. wi2 = Rel\_freq(ni2, size)
287. sk2 = Sum\_rfq(wi2, size)
289. inter2 = intervals(xi2,size)
290. **inter2**
292. *#inter = [[a[0],a[1]], ... , [a[n-1],a[n]]]*
294. ni\_inter2 = Freq\_inter(distr2, inter2, size)
295. **print (ni\_inter2)**
296. **print** (sum(ni\_inter2))
298. wi\_inter2 = Rel\_freq(ni\_inter2,size)
299. **print** (wi\_inter2)
300. **print (sum(wi\_inter2))**
302. *# график функции эмпирического распределения*
303. plt.figure(figsize=(12, 8))
304. fig, sec = plt.subplots()
305. **sec.xaxis.set\_major\_locator(ticker.MultipleLocator(0.5))**
306. sec.yaxis.set\_major\_locator(ticker.MultipleLocator(0.1))
307. **for** k **in** range(len(xi2) - 1):
308. sec.plot([xi2[k], xi2[k+1]], [sk2[k], sk2[k]])
309. plt.title("Эмпирическая функция распределения")
310. **plt.grid(True)**
311. plt.show()
313. m = 1+round(log(size,2))
314. plt.figure(figsize=(12, 7))
315. **sns.distplot(distr2, bins = m)**
316. plt.title('Гистограмма относительных частот')
318. data2 = pd.DataFrame({'distr2': distr2})
320. ***# Выборочное среднее (теор/эксп)***
321. X\_teor2 = lm\*\*(-1)
322. X2 = Mean(x\_x2,wi\_inter2)
323. **print** (X\_teor2, '**\n**', distr2.mean(), '**\n**')
325. ***#Выборочный момент***
326. 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**')
333. *#Выборочное среднеквадратическое откл.(теор/эксп)*
334. SD\_teor2 = lm\*\*(-1)
335. **print (SD\_teor2, '\n', distr2.std(),'\n')**
337. *#Выборочная мода (теор/эксп)*
338. mode\_teor2 = 0
339. **print** (mode\_teor2, '**\n**','**\n**')
341. *#Выборочная медиана (теор/эксп)*
342. median\_teor2 = log(2)/lm
343. **print** (median\_teor2, '**\n**', data2.describe(percentiles= [.50]).iloc[4,0],'**\n**')
345. ***#Выборочный коэф.асимметрии (теор/эксп)***
346. asym\_teor2 = 2
347. **print** (asym\_teor2, '**\n**', data2.skew()[0],'**\n**')
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**')
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)*
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)**

373. *# график функции эмпирического распределения*
374. plt.figure(figsize=(12, 8))
375. **fig, sec = plt.subplots()**
376. sec.xaxis.set\_major\_locator(ticker.MultipleLocator(0.5))
377. sec.yaxis.set\_major\_locator(ticker.MultipleLocator(0.1))
378. **for** k **in** range(len(xi3) - 1):
379. sec.plot([xi3[k], xi3[k+1]], [sk3[k], sk3[k]])
380. **plt.title("Эмпирическая функция распределения")**
381. plt.grid(True)
382. plt.show()
384. m = 1+round(log(size,2))
385. **plt.figure(figsize=(12, 7))**
386. sns.distplot(distr3, bins = m)
387. plt.title('Гистограмма относительных частот')
389. data3 = pd.DataFrame({'distr3': distr3})
391. *# Выборочное среднее (теор/эксп)*
392. X\_teor3 = (a\_unif+b\_unif)/2
393. X3 = Mean(x\_x3,wi\_inter3)
394. **print** (X\_teor3, '**\n**', distr3.mean(), '**\n**')
396. *#Выборочный момент*
397. M3 = Moment(x\_x3,wi\_inter3)
399. *# Выборочная дисперсия с поправкой Шепарда (теор/эксп)*
400. **D\_teor3 = (b\_unif - a\_unif)\*\*2/12**
401. D3 = DispSh(inter3,m,x\_x3,wi\_inter3,X3)
402. **print** (D\_teor3,'**\n**', distr3.var(), '**\n**')
404. *#Выборочное среднеквадратическое откл.(теор/эксп)*
405. **SD\_teor3 = (b\_unif - a\_unif)/(2\*3\*\*(1/2))**
406. **print** (SD\_teor3, '**\n**', distr3.std(),'**\n**')
408. *#Выборочная мода (теор/эксп)*
409. mode\_teor3 = (a\_unif + b\_unif)/2
410. **print (mode\_teor3, '\n','\n')**
412. *#Выборочная медиана (теор/эксп)*
413. median\_teor3 = (a\_unif + b\_unif)/2
414. **print** (median\_teor3, '**\n**', data3.describe(percentiles= [.50]).iloc[4,0],'**\n**')
416. *#Выборочный коэф.асимметрии (теор/эксп)*
417. asym\_teor3 = 0
418. **print** (asym\_teor3, '**\n**', data3.skew()[0],'**\n**')
420. ***#Выборочный коэф. эксцесса (теор/эксп)***
421. exe\_teor3 = -6/5
422. exe3 = Exe\_coef(M3, distr3.std())
423. **print** (exe\_teor3, '**\n**', data3.kurtosis()[0],'**\n**')