Отчет №2.2 Гаязов Ильназ Дамирович, группа 09-012

Проверка статистических гипотез о типе распределения

Вариант 1. Критерий согласия хи-квадрат

**Задание**: на заданном уровне значимости α = 0.1 проверить нулевую гипотезу *H­­0*: *генеральная совокупность распределена по закону N (µ, σ2)* против альтернативной гипотезы *H1: распределение генеральной совокупности отлично от нормального.*

**Решение**:

1) Для применения критерия данные должны быть сгруппированы. По данной выборке строим интервальный ряд:

|  |  |
| --- | --- |
| Интервалы | ni |
| (-0.383, -0.091] | 4 |
| (-0.092, 0.199] | 6 |
| (0.199, 0.491] | 18 |
| (0.491, 0.782] | 12 |
| (0.782, 1.073] | 13 |
| (1.073, 1.364] | 10 |
| (1.364, 1.655] | 12 |
| (1.655, 1.947] | 8 |
| (1.947, 2.238] | 5 |

Вычислим выборочное среднее: Хв = 0.933

Вычислим выборочную дисперсию: Sв = 0.392

Вычислим выборочную дисперсию: σв = 0.626

3) Вид критической области *С = {χ2: χ2>χкр2}*

4) Вычисление статистики хи-квадрат: *χ2 = 8.09*

5) Критическое значение *χ*кр2= 10.64

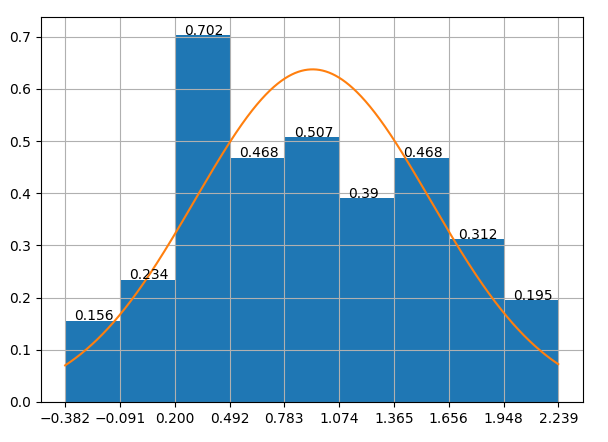
6) p – значение

pr-3 = 0.232 > α = 0.1 => p = 0.232

pr-1 = 0.425 > α = 0.1

**Вывод**: так как *χ2*<*χ*кр2, то на уровне значимости α = 0.1 нет оснований говорить о том, что нулевая гипотеза отвергается.

7) Гистограмма выборки с наложением графика теоретической плотности:



8) Код на Python

**import** openpyxl  
**import** math  
*# библиотеки для работы с статистическими данными***import** numpy **as** np  
**import** matplotlib.pylab **as** plt  
**from** scipy.stats **import** norm, chi2  
**import** scipy  
  
**def** vk(data: list, start: float, end: float):  
 *'''подсчет количества элементов выборки в интервале'''* count = 0  
 **for** x **in** data:  
 **if** (start < x <= end):  
 count += 1  
 **return** count  
  
**def** normal\_dist(x: float):  
 *'''функция плотности стандартного нормального распределения'''* prob\_density = (np.exp(-0.5 \* (x)\*\*2)) / math.sqrt(2 \* np.pi)  
 **return** prob\_density  
  
*# открыть файл для чтения*book = openpyxl.open(**'r2z2.xlsx'**, read\_only=**True**)  
*# позиционирование на листе*sheet = book.active  
*# коприуем данные из листа в список*data = []  
**for** row **in** range(2, sheet.max\_row + 1):  
 data.append(sheet[row][0].value)  
  
*# построение гистограммы*fig = plt.figure(figsize=(7, 5))  
ax = fig.add\_subplot()  
n, bins, pitches = ax.hist(data, bins=9, density=**True**)  
ax.xaxis.set\_major\_locator(plt.IndexLocator(base=(max(data) - min(data)) / 9, offset=0))  
**for** i **in** pitches:  
 height = round(i.get\_height(), 3)  
 ax.annotate(height, (i.get\_x() + 0.05, height+0.001))  
ax.grid()  
*# выборочное среднее*sum = 0  
**for** x **in** data:  
 sum += x  
mean = round(sum / len(data), 3)  
print(**"Выборочное среднее "**, mean)  
*#выборочная дисперсия*sum = 0  
**for** x **in** data:  
 sum += (x - mean)\*\*2  
disp = round(sum / len(data), 3)  
print(**"Выборочная дисперсия "**, disp)  
*# Выборочное стандартное отклонение*sigma = round(math.sqrt(disp), 3)  
print(**"Выборочное стандартное отклонение"**, sigma)  
  
*# график теоретичесой функции плотности*x\_axis = np.arange(min(data), max(data), 0.001)  
plt.plot(x\_axis, norm.pdf(x\_axis, mean, sigma))  
plt.show()  
  
step = round((max(data) - min(data)) / math.ceil(len(data)/10), 3)  
r = math.ceil(len(data)/10)  
T = 0  
count = 1  
n = len(data)  
left = min(data) - 0.001  
right = left + step  
**while** count <= r:  
 **if** count == 1:  
 pk = norm(loc = mean, scale = sigma).cdf(right)  
 T += ((vk(data, left, right) - n \* pk) \*\* 2) / (n \* pk)

print((vk(data, left, right))  
 **elif** count == r:  
 pk = 1 - norm(loc = mean, scale = sigma).cdf(left)  
 T += ((vk(data, left, right) - n \* pk) \*\* 2) / (n \* pk)

print((vk(data, left, right))  
 **else**:  
 pk = norm(loc = mean, scale = sigma).cdf(right) - norm(loc = mean, scale = sigma).cdf(left)

print((vk(data, left, right))  
 T += ((vk(data, left, right) - n \* pk) \*\* 2) / (n \* pk)  
 left = right  
 right += step  
 count += 1  
print(**"Статистика хи-квадрат "**, round(T, 2))  
*# вычисление критической константы*print(**"Критическая константа "**, round(chi2.ppf(0.9, r-3), 2))  
*# р-значение*print(**"p1-value"**, round(1 - chi2.cdf(T, r-3), 3))  
print(**"p2-value"**, round(1 - chi2.cdf(T, r-1), 3))