Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ФУНДАМЕНТАЛЬНЫЕ НАУКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И МАТЕМАТИЧЕСКАЯ ФИЗИКА (ФН11)

НАПРАВЛЕНИЕ ПОДГОТОВКИ МАТЕМАТИКА И КОМПЬЮТЕРНЫЕ НАУКИ (02.03.01)

О т ч е т

по лабораторной работе № 8

Название лабораторной работы:
Применение критерия χ^2 Пирсона к проверке гипотезы о виде функции распределения

Вариант № 9

Дисциплина:

Теория вероятности и математическая статистика

Студент группы ФН11-52Б _____ (Подпись, дата) **Очкин Н.В.** (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) **Облакова Т.В.** (И.О. Фамилия)

Москва, 2024

Задание

1. Используя группированную выборку из задачи 1, проверьте на уровне α гипотезу H_0: выборка взята из генеральной совокупности, распределенной по закону $F(x)$.

2. Неизвестные параметры распределения $F(x)$, если это необходимо, найдите методом моментов или методом максимального правдоподобия по выборке.

3. Постройте совмещенные графики гистограммы относительных частот и плотности, соответствующей функции распределения $F(x)$.

4. Дайте анализ полученного решения.

Исходные данные

R[a, b] $\alpha = 0.01$

14.495	4.715	7.175	8.428	11.093	3.375	12.906	8.415	8.916	13.48
5.343	17.985	15.992	13.89	9.838	13.924	9.012	9.458	17.69	6.542
14.396	8.592	8.206	14.237	7.357	10.821	12.767	16.058	12.959	4.354
12.888	10.268	9.182	5.647	8.282	2.903	15.988	12.959	14.919	6.339
2.375	17.921	9.097	15.85	11.449	11.095	9.493	12.175	7.479	13.535
9.234	6.078	4.964	6.355	13.957	12.911	15.694	14.286	9.869	5.175
5.811	7.241	5.814	3.086	6.875	3.878	5.333	15.134	12.924	9.159
4.727	4.646	15.535	9.919	17.117	10.351	16.892	12.423	10.511	4.942
4.843	9.927	15.864	3.635	17.963	8.25	5.14	6.734	12.622	13.325
3.377	16.195	12.04	12.768	2.744	14.186	9.354	15.439	14.612	15.649

Ход выполнения работы

Сгруппируем данные:

```
def group(data):
    n_ = len(data)

    min_ = min(data)
    max_ = max(data)

    range_ = max_ - min_

    l_ = 1 + int(np.log2(n_))

    h_ = range_ / l_

    int_boundaries_ = np.array(
        [min_ + i * h_ for i in range(0, l_ + 1, 1)]
    )
    intervals_ = np.array(
        [(int_boundaries_[i], int_boundaries_[i+1]) for i in range(0, l_, 1)]
    )
    mid_ranges_ = np.array(
        [sum(interval)/2 for interval in intervals_]
    )

    present = lambda el, int_ : int_[0] <= el < int_[1]
    freqs_ = np.zeros(l_)
    for el in data:
        for j in range(0, l_, 1):
            if present(el, intervals_[j]):
                freqs_[j] += 1

    freqs_[-1] += np.count_nonzero(data == max_)

    rel_freqs_ = freqs_ / n_

    rel_freqs_density_ = rel_freqs_ / h_

    return n_, min_, max_, range_, l_, h_, int_boundaries_, \
           intervals_, mid_ranges_, freqs_, rel_freqs_, rel_freqs_density_
```

Параметр	**Значение**
Количество наблюдений, n	120
Минимальное значение	2.375
Максимальное значение	17.985
Размах	15.61
Количество интервалов, l	7
Ширина интервала, h	2.23

Границы интервалов	**Интервалы**	**Середины**	**Частоты**	**Относительные частоты**	**Плотность относительных частот**
2.375	[2.375, 4.605]	3.49	12	0.1	0.04484305
4.605	[4.605, 6.835]	5.72	20	0.167	0.07473842
6.835	[6.835, 9.065]	7.95	18	0.15	0.06726457
9.065	[9.065, 11.295]	10.18	18	0.15	0.06726457
11.295	[11.295, 13.525]	12.41	17	0.14167	0.06352765
13.525	[13.525, 15.755]	14.64	20	0.167	0.07473842
15.755	[15.755, 17.985]	16.87	14	0.1167	0.05231689
17.985					

Оценим параметры:

$$\text{a} = \min = 2.375$$
$$\text{b} = \max = 17.985$$

Найдем $\chi^2_{\text{в}}$:

$$\chi^2_{\text{в}} = \sum_{i=1}^{l} \frac{(\nu_i - np_i)^2}{np_i}$$

Количество значений, попавших в j-ый интервал группировки (ν) нам уже известно и записано в столбце **Частоты** в таблице выше.

Определим теоретическую вероятность попадания в j-ый интервал группировки (p):

```
theorIntHitProbs_  = [] # p_j
theorIntHitProbsN_ = [] # n*p_j

cdf_ = lambda x : sp.stats.uniform.cdf(x, loc=a, scale=b-a)

for interval in intervals_:
    beg = interval[0]
    end = interval[1]

    theorIntHitProb = cdf_(end) - cdf_(beg)
    theorIntHitProbs_.append(theorIntHitProb)

    theorIntHitProbsN_.append(n_ * theorIntHitProb)
```

$$p_i : [0.143 \quad 0.143 \quad 0.143 \quad 0.143 \quad 0.143 \quad 0.143 \quad 0.143]$$
$$n \cdot p_i : [17.143 \quad 17.143 \quad 17.143 \quad 17.143 \quad 17.143 \quad 17.143 \quad 17.143]$$

$$\text{sum}(p) == 1$$

Итого имеем:

```
Chi2_v = sum([((freqs_[i] - theorIntHitProbsN_[i])**2) / \
                    theorIntHitProbsN_[i] for i in range(l_)])
```

$$\chi^2_{\text{в}} \approx 3.1583$$

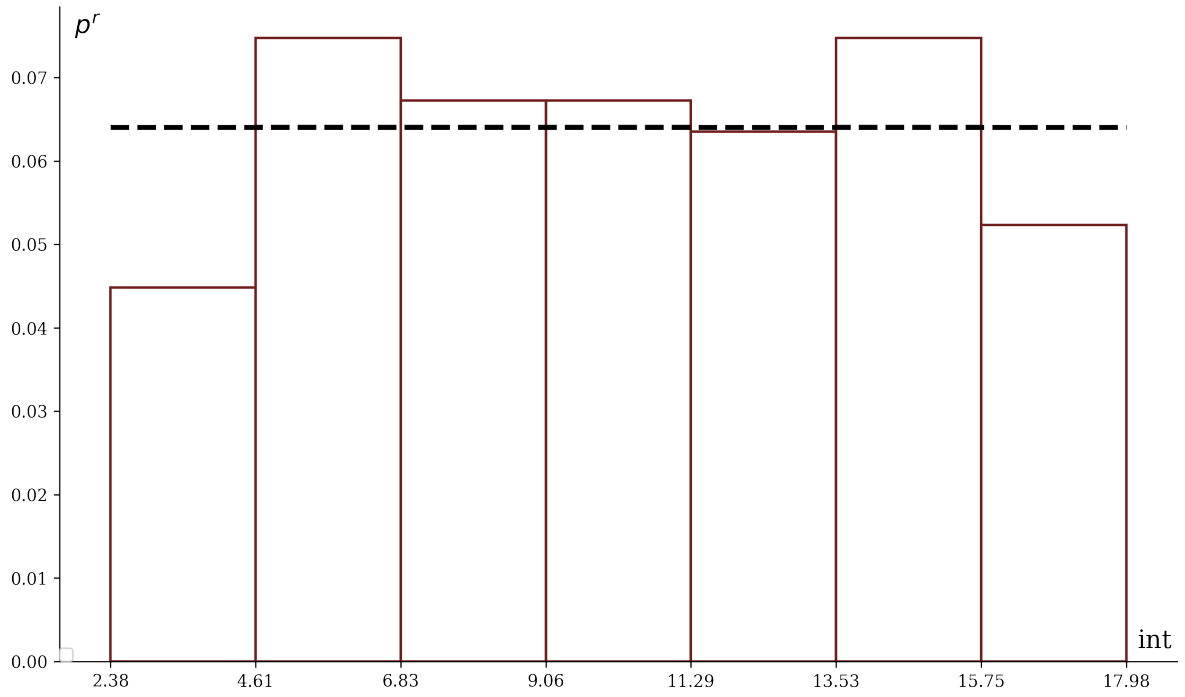
Теперь определим квантиль с $l - 1 - 2 = 4$ степенями свободы:

```
quantile = sp.stats.chi2.ppf(1 - alpha_, l_ - 1 - 2)
```

$$\chi^2_{1-\alpha}(l - 1 - 2) = \chi^2_{0.99}(4) \approx 13.2767$$

$$\chi^2_{\text{в}} < \chi^2_{0.99}(4) \Rightarrow \text{гипотеза принимается}$$

Построим совмещенные графики гистограммы относительных частот и плотности, соответствующей функции непрерывного равномерного распределения с параметрами a = 2.375 и b = 17.985.



Вывод

В ходе выполнения лабораторной работы были перепроверены выводы, сделанные в первой лабораторной работе, и действительно показано, что на уровне доверия $\alpha = 0.01$ выборка взята из генеральной совокупности, распределенной по непрерывному нормальному закону с найденными параметрами a и b.

Приложение

Программный код, с помощью которого была выполнена данная лабораторная работа.

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt

alpha_ = 0.01

data_ = [
    14.495, 4.715, 7.175, 8.428, 11.093, 3.375, 12.906, 8.415, 8.916, 13.48,
    5.343, 17.985, 15.992, 13.89, 9.838, 13.924, 9.012, 9.458, 17.69, 6.542,
    14.396, 8.592, 8.206, 14.237, 7.357, 10.821, 12.767, 16.058, 12.959, 4.354,
    12.888, 10.268, 9.182, 5.647, 8.282, 2.903, 15.988, 12.959, 14.919, 6.339,
    2.375, 17.921, 9.097, 15.85, 11.449, 11.095, 9.493, 12.175, 7.479, 13.535,
    9.234, 6.078, 4.964, 6.355, 13.957, 12.911, 15.694, 14.286, 9.869, 5.175,
    5.811, 7.241, 5.814, 3.086, 6.875, 3.878, 5.333, 15.134, 12.924, 9.159,
    4.727, 4.646, 15.535, 9.919, 17.117, 10.351, 16.892, 12.423, 10.511, 4.942,
    4.843, 9.927, 15.864, 3.635, 17.963, 8.25, 5.14, 6.734, 12.622, 13.325,
    3.377, 16.195, 12.04, 12.768, 2.744, 14.186, 9.354, 15.439, 14.612, 15.649,
    8.681, 5.006, 3.608, 2.867, 12.177, 15.506, 7.683, 14.022, 17.103, 8.905,
    12.173, 17.757, 6.883, 2.666, 9.861, 5.743, 16.175, 15.308, 7.039, 15.238
]

def decorate_plot(ax, x_ticks, xname, yname, loc=(-0.025, -0.3)):
    SIZE_TICKS = 10

    # Eliminate upper and right axes
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')

    # Show ticks in the left and lower axes only
    ax.xaxis.set_ticks_position('bottom')
    ax.yaxis.set_ticks_position('left')

    # axis names
    ax.set_xlabel(xname, fontsize=15)
    ax.xaxis.set_label_coords(0.98, 0.05)

    ax.set_ylabel(yname, rotation=0, fontsize=15)
    ax.yaxis.set_label_coords(0.025, 0.95)

    ax.set_xticks(x_ticks)

    # Adjust the font size of the tick labels
    ax.tick_params(axis='both', which='major', labelsize=SIZE_TICKS)

    plt.legend(fontsize=10, loc=loc)

    # Update font settings
    plt.rcParams.update({'font.family': 'serif', 'font.size': 12})
```

```
    # Adjust layout
    plt.tight_layout()

def clean(data):
    res = []
    for el in data:
        res.append(round(el, 3))
    return res

def group(data):
    n_ = len(data)
    print(f'n: {n_}')

    min_ = min(data)
    max_ = max(data)
    print(f'min: {min_}     max: {max_}')

    range_ = max_ - min_
    print(f'range: {range_}')

    l_ = 1 + int(np.log2(n_))
    print(f'l: {l_}')

    h_ = range_ / l_
    print(f'h: {h_}')

    int_boundaries_ = np.array(
        [min_ + i * h_ for i in range(0, l_ + 1, 1)]
    )
    print(f'interval boundaries: {int_boundaries_}')
    intervals_ = np.array(
        [(int_boundaries_[i], int_boundaries_[i+1]) for i in range(0, l_, 1)]
    )
    print(f'intervals: {intervals_}')
    mid_ranges_ = np.array(
        [sum(interval)/2 for interval in intervals_]
    )
    print(f'intervals\' midpoints: {mid_ranges_}')

    present = lambda el, int_ : int_[0] <= el < int_[1]
    freqs_ = np.zeros(l_)
    for el in data:
        for j in range(0, l_, 1):
            if present(el, intervals_[j]):
                freqs_[j] += 1

    freqs_[-1] += np.count_nonzero(data == max_)
    print(f'frequencies: {freqs_}')

    rel_freqs_ = freqs_ / n_
    print(f'relative frequencies: {rel_freqs_}')

    rel_freqs_density_ = rel_freqs_ / h_
    print(f'relative frequencies\' density: {rel_freqs_density_}')
```

```
    print(f'-'*100)

    space_ = ' ' * 5
    for i in range(l_):
        print(f'{intervals_[i]}{space_}{freqs_[i]}{space_}{rel_freqs_[i]}{space_}{re

    return n_, \
           min_, \
           max_, \
           range_, \
           l_, \
           h_, \
           int_boundaries_, \
           intervals_, \
           mid_ranges_, \
           freqs_, \
           rel_freqs_, \
           rel_freqs_density_

n_, \
min_, \
max_, \
range_, \
l_, \
h_, \
int_boundaries_, \
intervals_, \
mid_ranges_, \
freqs_, \
rel_freqs_, \
rel_freqs_density_ = group(data_)

a = min_
b = max_

theorIntHitProbs_  = [] # p_j
theorIntHitProbsN_ = [] # n*p_j

cdf_ = lambda x : sp.stats.uniform.cdf(x, loc=a, scale=b-a)

for interval in intervals_:
    beg = interval[0]
    end = interval[1]

    theorIntHitProb = cdf_(end) - cdf_(beg)
    theorIntHitProbs_.append(theorIntHitProb)

    theorIntHitProbsN_.append(n_ * theorIntHitProb)

print(f'p_i: {clean(theorIntHitProbs_)}')
print(f'n * p_i: {clean(theorIntHitProbsN_)}')

sum(theorIntHitProbs_)
```

```
Chi2_v = sum([((freqs_[i] - theorIntHitProbsN_[i])**2)/theorIntHitProbsN_[i] for i i
Chi2_v

quantile = sp.stats.chi2.ppf(1 - alpha_, l_ - 1 - 2)
quantile

def buildBar(filename):
    RED = '#6F1D1B'

    _, ax = plt.subplots(figsize=(10, 6))

    x_values = mid_ranges_
    y_values = rel_freqs_density_

    ax.bar(x_values,
           y_values,
           width=h_,
           color='white',
           edgecolor=RED,
           linestyle='-',
           linewidth=1.5,
           align='center')

    x_values = np.linspace(min_, max_, 100)
    y_values = sp.stats.uniform.pdf(x_values, loc=a, scale=b-a)
    ax.plot(x_values,
            y_values,
            color='black',
            linestyle='--',
            linewidth=3)

    decorate_plot(ax, int_boundaries_, 'int', '$p^r$', loc=(0, 0))

    plt.savefig(f'{filename}.png', dpi=300, transparent=True)

    plt.show()

buildBar('hist')
```