Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №5**

по дисциплине

‘Теория вероятностей’

Вариант №6

*Выполнил:*

Студент группы P32091

Кравец Роман

Денисович

*Преподаватель:*

Селина Елена

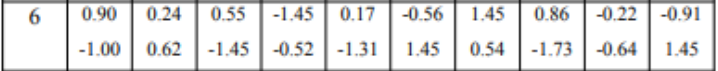
Георгиевна

Изображение выглядит как текст, коллекция картинок

Автоматически созданное описание

Санкт-Петербург, 2023

**Исходные данные:**



**Результат работы программы:**

Вариационный ряд: [-1.73, -1.45, -1.45, -1.31, -1.0, -0.91, -0.64, -0.56, -0.52, -0.22, 0.17, 0.24, 0.54, 0.55, 0.62, 0.86, 0.9, 1.45, 1.45, 1.45]

Экстремальные значения:

Max: 1.45

Min: -1.73

Размах выборки: 3.1799999999999997

Оценка математического ожидания -0.07799999999999999

Дисперсия 1.029226

Среднеквадратическое отклонение 1.014507762414857

Эмпирическая функция:

0 , при x <= -1.73

0.05 , при -1.73 < x <= -1.45

0.15 , при -1.45 < x <= -1.31

0.2 , при -1.31 < x <= -1.0

0.25 , при -1.0 < x <= -0.91

0.3 , при -0.91 < x <= -0.64

0.35 , при -0.64 < x <= -0.56

0.4 , при -0.56 < x <= -0.52

0.45 , при -0.52 < x <= -0.22

0.5 , при -0.22 < x <= 0.17

0.55 , при 0.17 < x <= 0.24

0.6 , при 0.24 < x <= 0.54

0.65 , при 0.54 < x <= 0.55

0.7 , при 0.55 < x <= 0.62

0.75 , при 0.62 < x <= 0.86

0.8 , при 0.86 < x <= 0.9

0.85 , при 0.9 < x <= 1.45

1.0 , при x > 1.45

[ -2.048 : -1.4120000000000001 ] - 3

[ -1.4120000000000001 : -0.7760000000000002 ] - 3

[ -0.7760000000000002 : -0.14000000000000035 ] - 4

[ -0.14000000000000035 : 0.49599999999999955 ] - 2

[ 0.49599999999999955 : 1.1319999999999995 ] - 5

[ 1.1319999999999995 : 1.7679999999999993 ] - 3

Изображение выглядит как диаграмма

Автоматически созданное описание

Изображение выглядит как диаграмма

Автоматически созданное описание

Изображение выглядит как диаграмма

Автоматически созданное описание

**Листинг программы:**

FILE\_IN = "files/input.txt"

x\_i, n\_i, p\_i = [], [], []

def getdata\_file():

    # Получить данные из файла

    with open(FILE\_IN, 'rt', encoding="utf-8") as fin:

        try:

            with open(FILE\_IN, 'rt', encoding='UTF-8') as fin:

                data = [list(map(float, row.split())) for row in fin.readlines()]

                return data[0]

        except (ValueError):

            print("Ошибка! С файлом проблемы!")

def numerical\_characteristics\_of\_distribution\_stat(data):

    #Получить числовые характеристики статистического распределения

    n = len(data)

    m, dispersion, q = 0, 0, 0

    data\_un = np.unique(data)

    for x in data\_un:

        counter = 0

        x\_i.append(x)

        for i in data:

            if (i == x):

                counter+=1

        n\_i.append(counter)

        p\_i.append(counter / n)

    for i in range(len(x\_i)):

        m += x\_i[i] \* p\_i[i]

    for i in range(len(x\_i)):

        dispersion += (x\_i[i] - m)\*\*2 \* p\_i[i]

    q = dispersion\*\*(1/2)

    return m, dispersion, q, data\_un

def variation\_series\_info(data):

    info = {}

    data.sort()

    n = len(data)

    info["varser"] = data

    info["min"] = data[0]

    info["max"] = data[n-1]

    info["scope"] = info["max"] - info["min"]

    info["interval"] = round(1 + math.log(n,2), 0)

    info["h"] = info["scope"] / info["interval"]

    info["xStart"] = info["min"] - info["h"]/2

    info["num\_characteristics"] = numerical\_characteristics\_of\_distribution\_stat(data)

    return info

def calculateEmpiricFunction(data):

    #Посчитать Эмпирическую функцию

    draw = []

    counter = 0

    f = 0

    print("Эмпирическая функция:")

    for i in p\_i:

        if (f == 0):

            print(round(f,4), f", при x <= {data[counter]}")

        else:

            print(str(round(f, 4)), f", при {prev} < x <= {data[counter]}")

        draw.append((data[counter], round(f,4)))

        f += i

        prev = data[counter]

        counter += 1

    draw.append((data[-1], round(f,4)))

    print(str(round(f, 4)), f", при x > {data[-1]}")

    return draw

def drawEmpiricFunction(fun\_data):

    #Нарисовать Эмпирическую функцию

    ax = plt.gca()

    plt.grid()

    ax.spines['top'].set\_visible(False)

    ax.spines['right'].set\_visible(False)

    x, y = zip(\*fun\_data)

    x\_new = list(x)

    y\_new = list(y)

    plt.title("График Эмпирической функции распределения")

    plt.plot([x\_new[0] - 0.5, x\_new[0]], [y\_new[0], y\_new[0]], color='r')

    plt.plot([x\_new[-1], x\_new[-1] + 0.5], [y\_new[-1], y\_new[-1]], color='r')

    for i in range(len(x\_new) - 1):

        plt.plot([x\_new[i], x\_new[i + 1]], [y\_new[i + 1], y\_new[i + 1]], color='r')

    plt.savefig(f"empiric\_function.png")

    plt.show(block=True)

def drawHistogram(max, h, xStart, data):

    #Нарисовать гистограмму относительной частоты

    histogram\_data = []

    while xStart < max:

        counter = 0

        for val in data:

            if (val >= xStart and val < (xStart+h)):

                counter+=1

        p\_w = counter / (len(data) \* h)

        histogram\_data.append((f"[{round(xStart, 2)}; {round(xStart+h, 2)})", p\_w))

        xStart += h

    x, y = zip(\*histogram\_data)

    fig, ax = plt.subplots()

    ax.bar(x, y)

    fig.set\_figwidth(12)

    fig.set\_figheight(6)

    plt.title("Гистограмма относительных частот")

    plt.savefig("histogram.png")

    plt.show(block=True)

def drawFrequencyPolygon(max, h, xStart, data):

    #Нарисовать полигон частот

    polygon\_data = []

    while xStart < max:

        counter = 0

        for val in data:

            if (val >= xStart and val < (xStart + h)):

                counter += 1

        polygon\_data.append(((xStart+h)/2, counter))

        print(f"[ {xStart} : {xStart+h} ] - {counter}")

        xStart += h

    ax = plt.gca()

    plt.grid()

    ax.spines['top'].set\_visible(False)

    ax.spines['right'].set\_visible(False)

    x, y = zip(\*polygon\_data)

    x\_new = list(x)

    y\_new = list(y)

    plt.title("Полигон частот")

    plt.plot(x\_new, y\_new, '-o')

    plt.savefig(f"poligon.png")

    plt.show(block=True)

def main():

    data = variation\_series\_info(getdata\_file())

    print("Вариационный ряд: ", data["varser"])

    print("Экстремальные значения: ")

    print("Max: ", data["max"])

    print("Min: ", data["min"])

    print("Размах выборки: ", data["scope"])

    print("Оценка математического ожидания", data["num\_characteristics"][0])

    print("Дисперсия", data["num\_characteristics"][1])

    print("Среднеквадратическое отклонение", data["num\_characteristics"][2])

    data\_for\_draw = calculateEmpiricFunction(data["num\_characteristics"][3])

    drawEmpiricFunction(data\_for\_draw)

    drawFrequencyPolygon(data["max"], data["h"], data["xStart"], data["varser"])

    drawHistogram(data["max"], data["h"], data["xStart"], data["varser"])

main()