Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники



Вариант **№14**

Практическая работа **№6**

По дисциплине

**Теория вероятностей**

*Выполнил*:

Студент группы P3218

Хромов Даниил

Тимофеевич

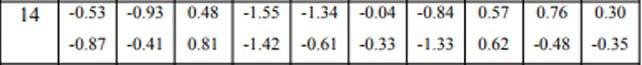
*Преподаватель*:

Селина Елена Георгиевна

Санкт-Петербург 2023 г.

1. **Текст задания**

Каждый студент получает выборку из 20 чисел. Необходимо определить следующие статистические характеристики: вариационный ряд, экстремальные значения и размах, оценки математического ожидания и среднеквадратического отклонения, эмпирическую функцию распределения и её график, гистограмму и полигон приведенных частот группированной выборки. Для расчета характеристик и построения графиков нужно написать программу на одном из языков программирования. Листинг программы и результаты работы должны быть представлены в отчете по практической работе. Стандартные функции статистики использовать нельзя.



1. **Выполнение задания и его результаты**

Далее мы можем видеть листинг кода для решения поставленной задачи:

import math

import matplotlib.pyplot as plt

import numpy as np

from collections import defaultdict

import os

class Chart:

def \_\_init\_\_(self, x\_label: str, y\_label: str, title: str):

self.x\_label = x\_label

self.y\_label = y\_label

self.title = title

self.fig, self.ax = plt.subplots()

self.ax.set\_title(title)

self.ax.set\_xlabel(x\_label)

self.ax.set\_ylabel(y\_label)

self.ax.grid(True)

def add\_line(self, x: list, y: list, label: str = None, linewidth: float = 2, marker: str = None):

self.ax.plot(x, y, label=label, linewidth=linewidth, marker=marker)

def add\_polygonal\_line(self, x: list, y: list, label: str = None, linewidth: float = 2):

self.ax.plot(x, y, label=label, linewidth=linewidth)

def add\_histogram(self, x: list, heights: list, width: float, label: str = None):

self.ax.bar(x, heights, width=width, align='edge', alpha=0.5, label=label, edgecolor='black')

def save\_png(self, name: str):

self.ax.legend(loc='upper left', bbox\_to\_anchor=(1.05, 1), fontsize='small', title="History")

plt.tight\_layout(rect=(0.0, 0.0, 0.85, 1.0))

plt.savefig(f"{name}.png")

plt.show()

plt.close(self.fig)

class ProbabilityTheory:

def \_\_init\_\_(self, values: list):

self.values = sorted(values)

self.n = len(values)

self.xi = []

self.ni = []

self.pi = []

def get\_var\_values(self):

print("! Вариационный ряд:")

print(self.values)

print("\n")

def calculate\_statistical\_series(self):

print("! Статистический ряд:")

frequency = defaultdict(int)

for x in self.values:

frequency[x] += 1

for value in sorted(frequency.keys()):

print(f"x = {value:.2f}, n = {frequency[value]}")

print("\n")

def get\_extreme\_values(self):

print("! Экстремальные значения:")

print(f"MIN = {self.values[0]}")

print(f"MAX = {self.values[-1]}\n")

def get\_selection\_size(self):

range\_val = self.values[-1] - self.values[0]

print("! Размах выборки:")

print(f"{range\_val}\n")

def calculate\_numeric\_characteristics(self):

print("! Числовые характеристики:")

# Медиана

if self.n % 2 == 1:

median = self.values[self.n // 2]

else:

median = (self.values[self.n // 2 - 1] + self.values[self.n // 2]) / 2

print(f"Медиана: {median:.2f}")

# Мода

frequency = defaultdict(int)

for x in self.values:

frequency[x] += 1

mode = [k for k, v in frequency.items() if v == max(frequency.values())]

if len(mode) == len(frequency):

mode\_value = "Нет явной моды"

else:

mode\_value = ", ".join(f"{m:.2f}" for m in mode)

print(f"Мода: {mode\_value}\n")

def discrepancy\_calculation(self):

frequency = defaultdict(int)

for x in self.values:

frequency[x] += 1

for x in sorted(frequency.keys()):

self.xi.append(x)

self.ni.append(frequency[x])

self.pi.append(frequency[x] / self.n)

expected\_value = sum(x \* p for x, p in zip(self.xi, self.pi))

print("! Оценка математического ожидания")

print(f"{expected\_value:.2f}\n")

variance = sum(((x - expected\_value) \*\* 2) \* n for x, n in zip(self.xi, self.ni)) / self.n

corrected\_variance = (self.n / (self.n - 1)) \* variance if self.n > 1 else 0

print("! Дисперсия")

print(f"{variance:.2f}\n")

print("! Исправленная дисперсия")

print(f"{corrected\_variance:.2f}\n")

print("! Cреднеквадратическоe отклонение")

print(f"{math.sqrt(variance):.2f}\n")

print("! Исправленное СКО")

print(f"{math.sqrt(corrected\_variance):.2f}\n")

def get\_h(self):

return (self.values[-1] - self.values[0]) / (1 + math.log2(self.n)) # Формула Стерджеса

def get\_m(self):

return math.ceil(1 + math.log2(self.n))

def calculate\_empiric\_function(self):

print("\t\t\t! Функция")

print(f"\t\t\tx\t<=\t{self.xi[0]:.2f}\t->\t0.00")

h = 0

chart = Chart("x", "F(X)", "Эмпирическая функция")

x\_start = self.values[0] - 1

chart.add\_line([x\_start, self.xi[0]], [0, 0], label=f"x <= {self.xi[0]:.2f}")

for i in range(len(self.xi)):

h += self.pi[i]

if i < len(self.xi) - 1:

print(f"{self.xi[i]:.2f}\t<\tx\t<=\t{self.xi[i + 1]:.2f}\t->\t{h:.2f}")

chart.add\_line([self.xi[i], self.xi[i + 1]], [h, h],

label=f"{self.xi[i]:.2f} < x <= {self.xi[i + 1]:.2f}")

print(f"{self.xi[-1]:.2f}\t<\tx\t\t\t\t->\t{h:.2f}")

chart.add\_line([self.xi[-1], self.xi[-1] + 1], [h, h], label=f"{self.xi[-1]:.2f} < x")

chart.save\_png("EmpiricFunction")

def draw\_frequency\_polygon(self):

h = self.get\_h()

m = self.get\_m()

frequency = [0] \* m

x\_start = self.values[0] - h / 2

for value in self.values:

index = int((value - x\_start) // h)

if index >= m:

index = m - 1

frequency[index] += 1

print("! Полигон частот:")

xs = []

ys = []

chart = Chart("x", "p\_i", "Полигон частот")

for i in range(m):

freq = frequency[i] / self.n

bin\_center = x\_start + h / 2 + i \* h

xs.append(bin\_center)

ys.append(freq)

print(f"[ {x\_start + i \* h:.2f} : {x\_start + (i + 1) \* h:.2f} ) -> {freq:.2f}")

chart.add\_polygonal\_line(xs, ys, label="Frequency Polygon")

chart.save\_png("FrequencyPolygon")

def draw\_histogram(self):

h = self.get\_h()

m = self.get\_m()

frequency = [0] \* m

x\_start = self.values[0] - h / 2

for value in self.values:

index = int((value - x\_start) // h)

if index >= m:

index = m - 1

frequency[index] += 1

bins = [x\_start + i \* h for i in range(m + 1)]

heights = [(freq / self.n) / h for freq in frequency]

chart = Chart("x", "p\_i / h", "Гистограмма частот")

chart.add\_histogram(bins[:-1], heights, width=h, label="Histogram")

chart.save\_png("Histogram")

def print\_data(self):

print("\n\n! DEBUG:\n")

for x, n, p in zip(self.xi, self.ni, self.pi):

print(f"{x} {n} {p}")

def read\_input(file\_path: str, n: int = 20) -> list:

if not os.path.exists(file\_path):

raise FileNotFoundError(f"File {file\_path} not found.")

with open(file\_path, 'r') as file:

data = []

for line in file:

parts = line.strip().split()

for part in parts:

if part:

data.append(float(part.replace(',', '.')))

if len(data) == n:

break

if len(data) == n:

break

if len(data) < n:

raise ValueError(f"Expected {n} numbers, but got {len(data)}.")

return data

def main():

input\_file = "input.txt"

try:

elements = read\_input(input\_file)

except Exception as e:

print(f"Error reading input: {e}")

return

runner = ProbabilityTheory(elements)

# Вариационный ряд

runner.get\_var\_values()

# Статистический ряд

runner.calculate\_statistical\_series()

# Экстремальные значения

runner.get\_extreme\_values()

# Размах

runner.get\_selection\_size()

# Числовые характеристики

runner.calculate\_numeric\_characteristics()

# Оценки математического ожидания и среднеквадратического отклонения

runner.discrepancy\_calculation()

# Эмпирическая функция распределения и её график

runner.calculate\_empiric\_function()

# Полигон частот

runner.draw\_frequency\_polygon()

# Гистограмма

runner.draw\_histogram()

# Отладочные данные

# runner.print\_data()

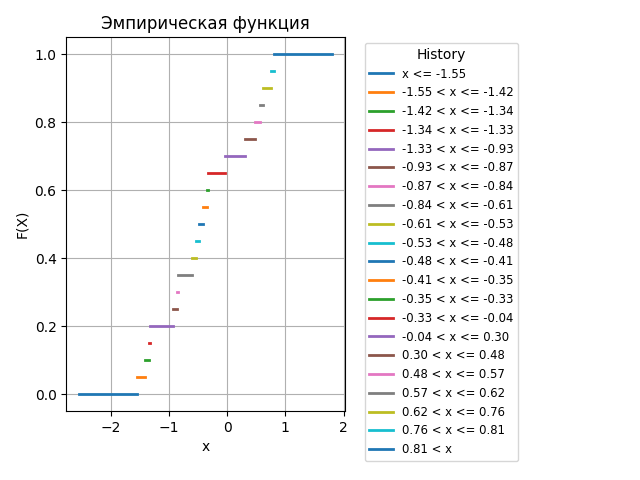
if \_\_name\_\_ == "\_\_main\_\_":

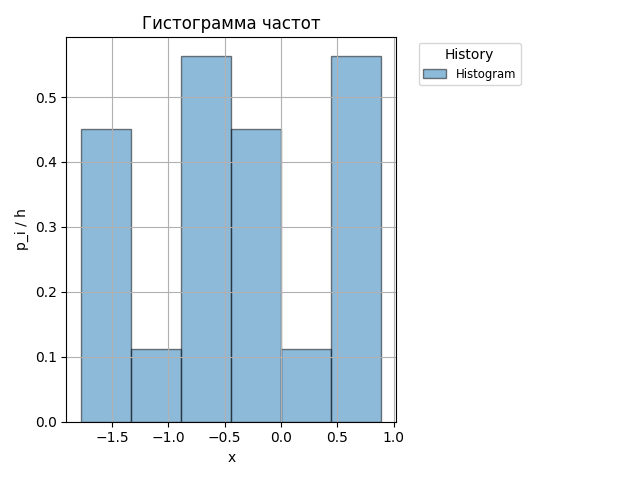
main()

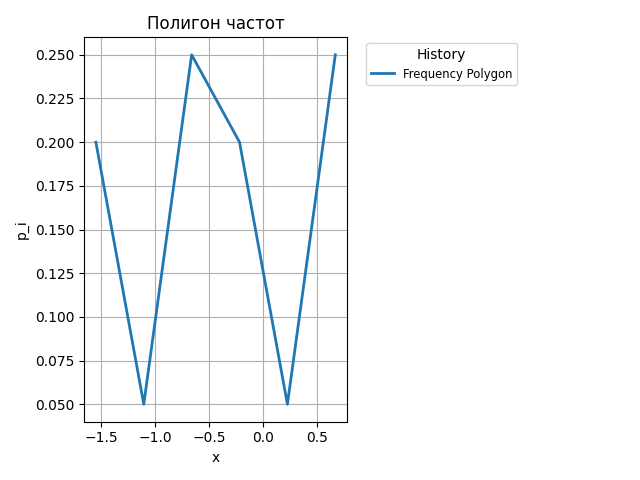
(Листинг №1. Код программы)

Здесь есть исчервыпающие комментарии по описанию каждого блока кода, то есть за что именно он отвечает, для более подробных объяснений, а именно работы с библиотеками и более детальному разъяснению каждого фрагмента кода или команды, не вижу целесообразным.

Результатами работы нашей программы станут вот такие графики:







Ну и собственно мы можем наблюдать еще вот такие выводы в нашей консоли:

! Вариационный ряд:

[-1.55, -1.42, -1.34, -1.33, -0.93, -0.87, -0.84, -0.61, -0.53, -0.48, -0.41, -0.35, -0.33, -0.04, 0.3, 0.48, 0.57, 0.62, 0.76, 0.81]

! Статистический ряд:

x = -1.55, n = 1

x = -1.42, n = 1

x = -1.34, n = 1

x = -1.33, n = 1

x = -0.93, n = 1

x = -0.87, n = 1

x = -0.84, n = 1

x = -0.61, n = 1

x = -0.53, n = 1

x = -0.48, n = 1

x = -0.41, n = 1

x = -0.35, n = 1

x = -0.33, n = 1

x = -0.04, n = 1

x = 0.30, n = 1

x = 0.48, n = 1

x = 0.57, n = 1

x = 0.62, n = 1

x = 0.76, n = 1

x = 0.81, n = 1

! Экстремальные значения:

MIN = -1.55

MAX = 0.81

! Размах выборки:

2.3600000000000003

! Числовые характеристики:

Медиана: -0.44

Мода: Нет явной моды

! Оценка математического ожидания

-0.37

! Дисперсия

0.55

! Исправленная дисперсия

0.58

! Cреднеквадратическоe отклонение

0.74

! Исправленное СКО

0.76

! Функция

x <= -1.55 -> 0.00

-1.55 < x <= -1.42 -> 0.05

-1.42 < x <= -1.34 -> 0.10

-1.34 < x <= -1.33 -> 0.15

-1.33 < x <= -0.93 -> 0.20

-0.93 < x <= -0.87 -> 0.25

-0.87 < x <= -0.84 -> 0.30

-0.84 < x <= -0.61 -> 0.35

-0.61 < x <= -0.53 -> 0.40

-0.53 < x <= -0.48 -> 0.45

-0.48 < x <= -0.41 -> 0.50

-0.41 < x <= -0.35 -> 0.55

-0.35 < x <= -0.33 -> 0.60

-0.33 < x <= -0.04 -> 0.65

-0.04 < x <= 0.30 -> 0.70

0.30 < x <= 0.48 -> 0.75

0.48 < x <= 0.57 -> 0.80

0.57 < x <= 0.62 -> 0.85

0.62 < x <= 0.76 -> 0.90

0.76 < x <= 0.81 -> 0.95

0.81 < x -> 1.00

! Полигон частот:

[ -1.77 : -1.33 ) -> 0.20

[ -1.33 : -0.88 ) -> 0.05

[ -0.88 : -0.44 ) -> 0.25

[ -0.44 : 0.00 ) -> 0.20

[ 0.00 : 0.45 ) -> 0.05

[ 0.45 : 0.89 ) -> 0.25

1. **Вывод**

По проделанной работе я научился на практике применять формулы из теории вероятностей на базе чего прогнозировать и строить функции при помощи языков программирования и определенных библиотек. Наиболее интересным являлся переход из функционального вида, будь то, численный или в виде формулы, в графический на базе определенной лексики в рамках выбранной среды языка программирования.