МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

**ЛАБОРАТОРНАЯ РАБОТА № 1**

по дисциплине

«МАТЕМАТИЧЕСКАЯ СТАТИСТИКА»

Вариант № 3

**Выполнил**:

Студент группы P3217

Михайлов Павел

Максимович

**Преподаватель:** Оранский Станислав Игоревич

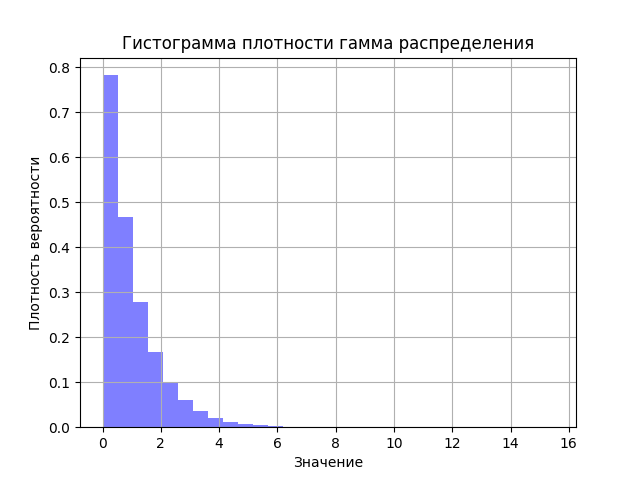
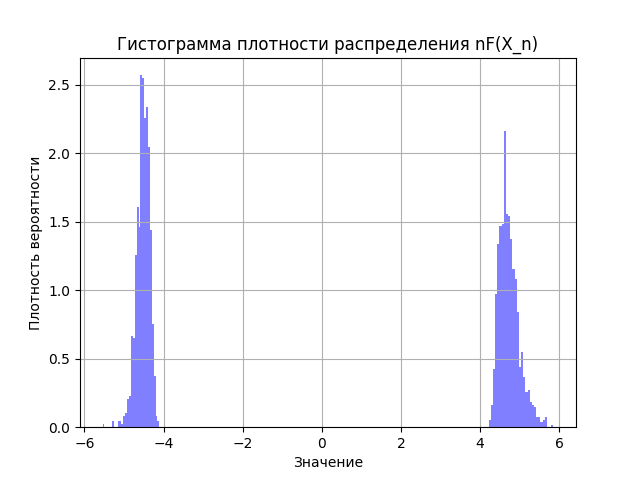
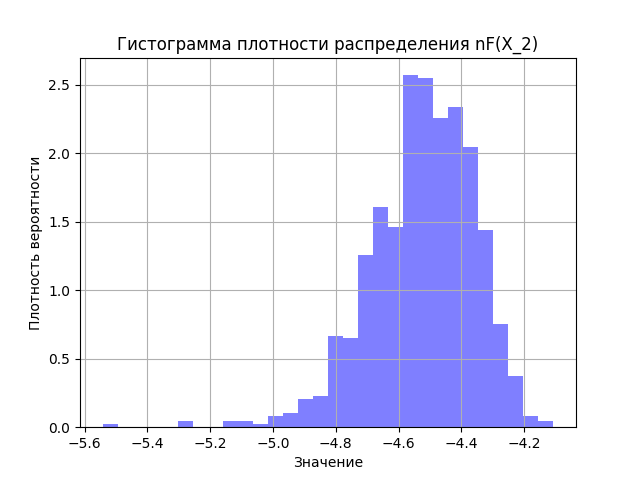
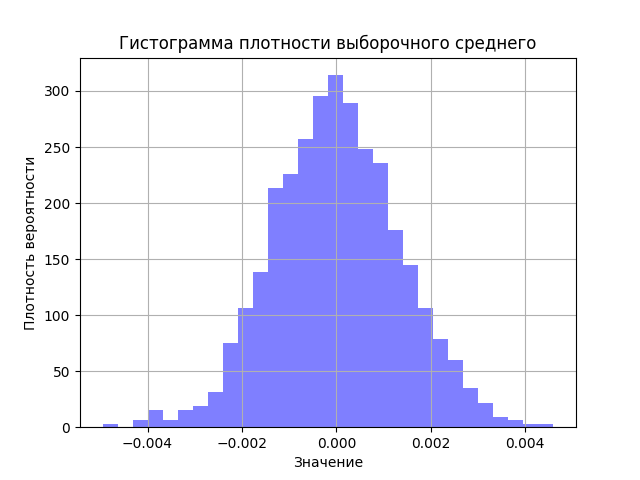
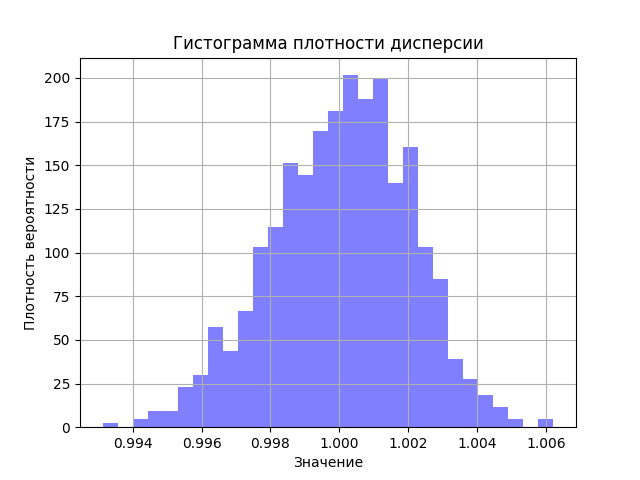
г. Санкт-Петербург, 2024

# Задание 1

## Листинг

import numpy as np  
from matplotlib import pyplot as plt  
from scipy.stats import gamma  
from sklearn.preprocessing import MinMaxScaler  
  
  
def generate\_gamma\_sample(shape, rate, size):  
 gamma\_dist = gamma(shape, scale=1 / rate)  
 return gamma\_dist.rvs(size=size)  
  
  
def compute\_average(\_data):  
 return sum(\_data) / len(\_data)  
  
  
def compute\_dispersion(\_data):  
 \_average = compute\_average(\_data)  
 return sum([(x - \_average) \*\* 2 for x in \_data]) / (len(\_data) - 1)  
  
  
def compute\_median(\_data):  
 temp\_data = sorted(\_data)  
 n = len(temp\_data)  
 if n % 2 != 0:  
 return temp\_data[n // 2]  
 else:  
 return (temp\_data[n // 2 - 1] + temp\_data[n // 2]) / 2  
  
  
# Задание параметров распределения  
shape = 1  
rate = 1  
sample\_size = 1000000 # размер каждой выборки  
  
# Ввод количества выборок с клавиатуры  
n = int(input("Введите количество выборок (n): "))  
  
  
plt.hist(generate\_gamma\_sample(shape, rate, sample\_size), bins=30, density=True, alpha=0.5, color='b')  
# Настройка отображения  
plt.xlabel('Значение')  
plt.ylabel('Плотность вероятности')  
plt.title('Гистограмма плотности медианы')  
plt.grid(True)  
plt.savefig("histogram\_gamma")  
plt.show()  
  
# Инициализация массивов для хранения результатов  
averages = np.zeros(n)  
dispersions = np.zeros(n)  
medians = np.zeros(n)  
F\_X\_2 = np.zeros(n)  
F\_X\_n = np.zeros(n)  
  
# Генерация данных из стандартного нормального распределения  
n\_data\_normalized = np.random.normal(loc=0, scale=1, size=(n, sample\_size))  
  
  
for i in range(n):  
 averages[i] = compute\_average(n\_data\_normalized[i])  
  
 dispersions[i] = compute\_dispersion(n\_data\_normalized[i])  
 medians[i] = compute\_median(n\_data\_normalized[i])  
 F\_X\_2[i] = sorted(n\_data\_normalized[i])[1]  
 F\_X\_n[i] = max(n\_data\_normalized[i])  
  
  
# Вывод результатов  
print("Статистики для каждой выборки:")  
for i in range(n):  
 print(f"Выборка {i + 1}:")  
 print(f"Выборочное среднее: {averages[i]}")  
 print(f"Выборочная дисперсия: {dispersions[i]}")  
 print(f"Выборочная медиана: {medians[i]}\n")  
  
  
plt.hist(F\_X\_2, bins=30, density=True, alpha=0.5, color='b')  
# Настройка отображения  
plt.xlabel('Значение')  
plt.ylabel('Плотность вероятности')  
plt.title('Гистограмма плотности распределения nF(X\_2)')  
plt.grid(True)  
plt.savefig('histogram\_nF\_x\_2.png')  
  
  
plt.hist(F\_X\_n, bins=30, density=True, alpha=0.5, color='b')  
# Настройка отображения  
plt.xlabel('Значение')  
plt.ylabel('Плотность вероятности')  
plt.title('Гистограмма плотности распределения nF(X\_n)')  
plt.grid(True)  
plt.savefig('histogram\_nF\_x\_n.png')  
plt.show()  
  
  
plt.hist(averages, bins=30, density=True, alpha=0.5, color='b')  
# Настройка отображения  
plt.xlabel('Значение')  
plt.ylabel('Плотность вероятности')  
plt.title('Гистограмма плотности выборочного среднего')  
plt.grid(True)  
plt.savefig('histogram\_average.png')  
plt.show()  
  
plt.hist(dispersions, bins=30, density=True, alpha=0.5, color='b')  
# Настройка отображения  
plt.xlabel('Значение')  
plt.ylabel('Плотность вероятности')  
plt.title('Гистограмма плотности дисперсии')  
plt.grid(True)  
plt.savefig('histogram\_dispersion.png')  
plt.show()  
  
plt.hist(medians, bins=30, density=True, alpha=0.5, color='b')  
# Настройка отображения  
plt.xlabel('Значение')  
plt.ylabel('Плотность вероятности')  
plt.title('Гистограмма плотности медианы')  
plt.grid(True)  
plt.savefig('histogram\_median.png')  
plt.show()

## Гистограммы



Из этих гистограмм можно сделать вывод, что дисперсия, медиана и среднее выборочное при большом объеме выборки стремятся к нормальному распределению, а величины имеют схожую с гистограммой плотности гамма-распределения форму.

# Задание 2

## Листинг

import csv  
  
import numpy as np  
  
import plot\_by\_data  
  
  
def compute\_average(\_data):  
 return sum(\_data) / len(\_data)  
  
  
def compute\_dispersion(\_data):  
 \_average = compute\_average(\_data)  
 sm = 0  
 for x in \_data:  
 sm += (x - \_average) \*\* 2  
 return sum([(x - \_average) \*\* 2 for x in \_data]) / (len(\_data) - 1)  
  
  
def compute\_median(\_data):  
 temp\_data = sorted(\_data)  
 n = len(temp\_data)  
 if n % 2 != 0:  
 return temp\_data[n // 2]  
 else:  
 return (temp\_data[n // 2 - 1] + temp\_data[n // 2]) / 2  
  
  
def compute\_iqr(\_data):  
 q1 = np.percentile(\_data, 25)  
 q3 = np.percentile(\_data, 75)  
 return q3 - q1  
  
  
# Создаем список для хранения данных из CSV файла  
data = []  
  
# Читаем данные из CSV файла и сохраняем их в список  
with open("cars93.csv", encoding='utf-8') as r\_file:  
 file\_reader = csv.DictReader(r\_file, delimiter=",")  
 for row in file\_reader:  
 data.append(row)  
  
# Теперь вы можете использовать данные из списка data в любом месте вашей программы  
# Например, для подсчета количества автомобилей каждого производителя  
cars = {}  
types = {}  
  
for row in data:  
 manufacturer = row["Manufacturer"]  
 type = row["Type"]  
 if manufacturer in cars:  
 cars[manufacturer] += 1  
 else:  
 cars[manufacturer] = 1  
  
 if type in types:  
 types[type] += 1  
 else:  
 types[type] = 1  
  
print(cars)  
  
# Находим ключ с максимальным значением  
max\_key = max(cars, key=cars.get)  
  
print("Производитель с наибольшим количеством автомобилей:", max\_key)  
print("Количество автомобилей:", cars[max\_key])  
  
# Находим ключ с максимальным значением  
min\_key = min(cars, key=cars.get)  
  
print("Производитель с наибольшим количеством автомобилей:", min\_key)  
print("Количество автомобилей:", cars[min\_key])  
  
print("Статистические данные всей выборки автомобилей:")  
horsepowers = [int(row["Horsepower"]) for row in data]  
print(f'\tВыборочное среднее: {compute\_average(horsepowers)}')  
print(f'\tВыборочная дисперсия: {compute\_dispersion(horsepowers)}')  
print(f'\tВыборочная медиана: {compute\_median(horsepowers)}')  
print(f"\tМежквартильный размах: {compute\_iqr(horsepowers)}")  
  
plot\_by\_data.plot(horsepowers, "general")  
  
print(types)  
for type in types.keys():  
 print(f"Статистические данные всей выборки автомобилей типа {type}:")  
 horsepowers\_for\_curr\_type = []  
 for car in data:  
 if car["Type"] == type:  
 horsepowers\_for\_curr\_type.append(int(car["Horsepower"]))  
  
 plot\_by\_data.plot(horsepowers\_for\_curr\_type, type)  
 print(horsepowers\_for\_curr\_type)  
 print(f'\tВыборочное среднее: {compute\_average(horsepowers\_for\_curr\_type)}')  
 print(f'\tВыборочная дисперсия: {compute\_dispersion(horsepowers\_for\_curr\_type)}')  
 print(f'\tВыборочная медиана: {compute\_median(horsepowers\_for\_curr\_type)}')  
 print(f"\tМежквартильный размах: {compute\_iqr(horsepowers\_for\_curr\_type)}")

## Вывод программы:

{'Acura': 2, 'Audi': 2, 'BMW': 1, 'Buick': 4, 'Cadillac': 2, 'Chevrolet': 8, 'Chrylser': 1, 'Chrysler': 2, 'Dodge': 6, 'Eagle': 2, 'Ford': 8, 'Geo': 2, 'Honda': 3, 'Hyundai': 4, 'Infiniti': 1, 'Lexus': 2, 'Lincoln': 2, 'Mazda': 5, 'Mercedes-Benz': 2, 'Mercury': 2, 'Mitsubishi': 2, 'Nissan': 4, 'Oldsmobile': 4, 'Plymouth': 1, 'Pontiac': 5, 'Saab': 1, 'Saturn': 1, 'Subaru': 3, 'Suzuki': 1, 'Toyota': 4, 'Volkswagen': 4, 'Volvo': 2}

Производитель с наибольшим количеством автомобилей: Chevrolet

Количество автомобилей: 8

Производитель с наибольшим количеством автомобилей: BMW

Количество автомобилей: 1

Статистические данные всей выборки автомобилей:

Выборочное среднее: 143.8279569892473

Выборочная дисперсия: 2743.0787751285648

Выборочная медиана: 140

Межквартильный размах: 67.0

{'Small': 21, 'Midsize': 22, 'Compact': 16, 'Large': 11, 'Sporty': 14, 'Van': 9}

Статистические данные всей выборки автомобилей типа Small:

[140, 92, 93, 92, 63, 127, 55, 102, 81, 124, 82, 103, 92, 110, 74, 85, 73, 90, 70, 82, 81]

Выборочное среднее: 91.0

Выборочная дисперсия: 447.6

Выборочная медиана: 90

Межквартильный размах: 21.0

Статистические данные всей выборки автомобилей типа Midsize:

[200, 172, 208, 110, 170, 295, 110, 100, 140, 128, 278, 185, 225, 160, 217, 140, 202, 160, 110, 200, 130, 168]

Выборочное среднее: 173.0909090909091

Выборочная дисперсия: 2756.08658008658

Выборочная медиана: 169.0

Межквартильный размах: 69.0

Статистические данные всей выборки автомобилей типа Compact:

[172, 110, 110, 141, 100, 96, 140, 164, 130, 150, 155, 110, 140, 130, 134, 114]

Выборочное среднее: 131.0

Выборочная дисперсия: 518.5333333333333

Выборочная медиана: 132.0

Межквартильный размах: 33.25

Статистические данные всей выборки автомобилей типа Large:

[170, 180, 200, 170, 153, 147, 214, 190, 210, 170, 170]

Выборочное среднее: 179.45454545454547

Выборочная дисперсия: 477.0727272727273

Выборочная медиана: 170

Межквартильный размах: 25.0

Статистические данные всей выборки автомобилей типа Sporty:

[160, 300, 300, 105, 115, 90, 160, 92, 255, 100, 92, 160, 135, 178]

Выборочное среднее: 160.14285714285714

Выборочная дисперсия: 5536.285714285714

Выборочная медиана: 147.5

Межквартильный размах: 72.25

Статистические данные всей выборки автомобилей типа Van:

[170, 165, 142, 145, 155, 151, 170, 138, 109]

Выборочное среднее: 149.44444444444446

Выборочная дисперсия: 370.27777777777777

Выборочная медиана: 151

Межквартильный размах: 23.0

Process finished with exit code 0