**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине: Статистические методы обработки**

**экспериментальных данных**

**Тема: «Формирование и первичная обработка выборки. Ранжированный и интервальный ряды»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Фиалковский М.С. |
| Преподаватель |  | Середа В.И. |

Санкт-Петербург

2020

**Цель работы.**

Ознакомление с основными правилами формирования выборки и подготовки выборочных данных к статистическому анализу.

**Постановка задачи.**

Используя знания по программированию прикладных задач в математических пакетах решить следующие задачи:

* сформировать репрезентативную выборку заданного объема из имеющейся генеральной совокупности экспериментальных данных
* осуществить последовательное преобразование полученной выборки в ранжированный, вариационный и интервальный ряды. Применительно к интервальному ряду построить и отобразить графически полигон, гистограмму и эмпирическую функцию распределения для абсолютных и относительных частот. Полученные результаты содержательно проинтерпретировать.

**Выполнение работы.**

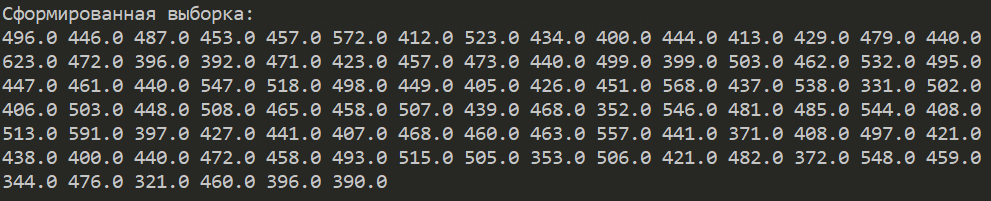
Произведём выборку из представленной генеральной совокупности экспериментальных данных размером 96 элементов. Результат на рис. 1. Для обеспечения получения одинаковых результатов при множественных запусках программы передаём генератору случайных чисел заранее определенное зерно. Код программы представлены в приложении А.

Рис. 1. Начальная выборка

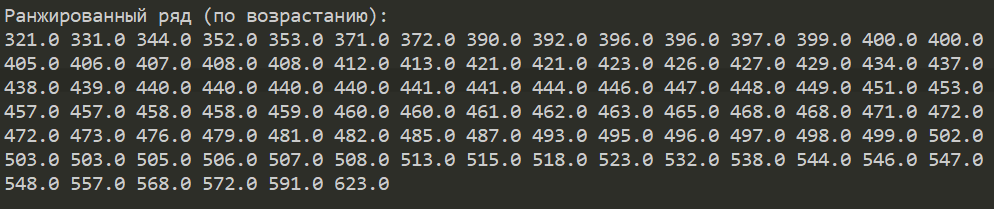
Ранжированный ряд представлен на рис.2.

Рис. 2. Ранжированный ряд

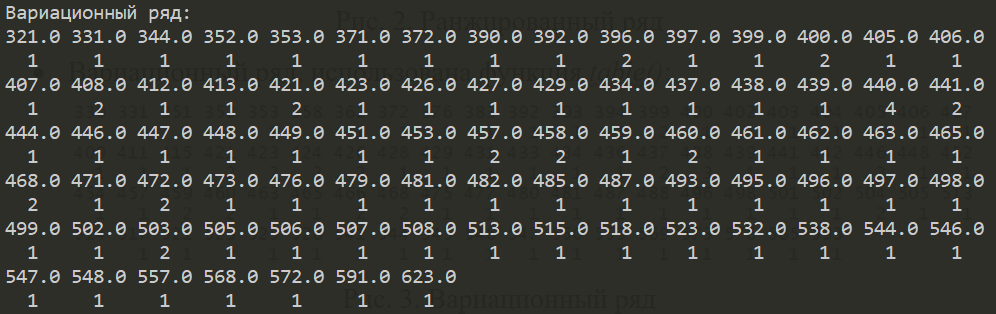
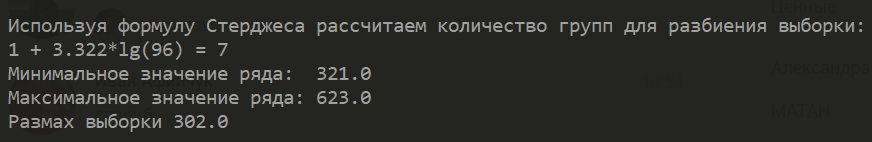
Вариационный ряд представлен на рис.3.

Рис. 3. Вариационный ряд

Для построения интервальный ряда требуется рассчитать количество различных групп. Требуемые расчёты представлены на рис. 4. Построенный интервальный ряд представлен на рис. 5.

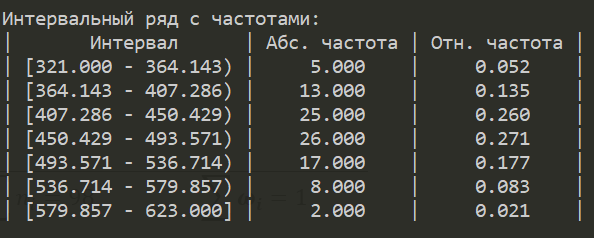
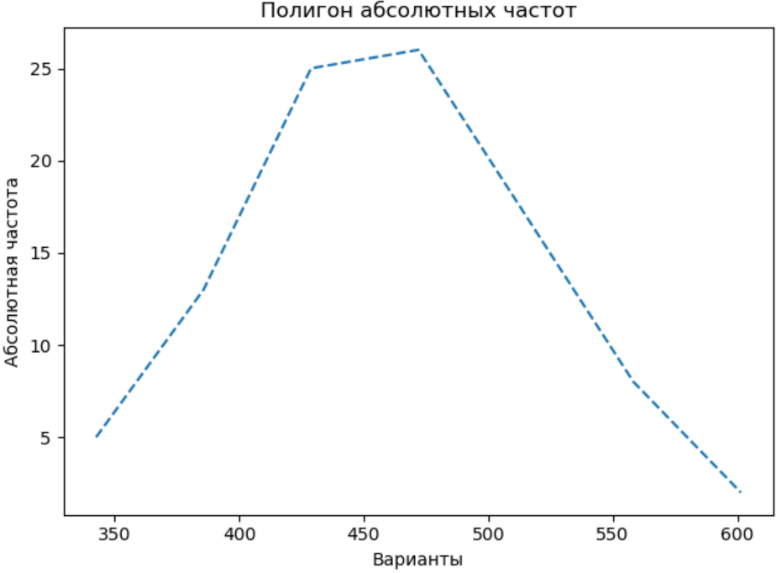
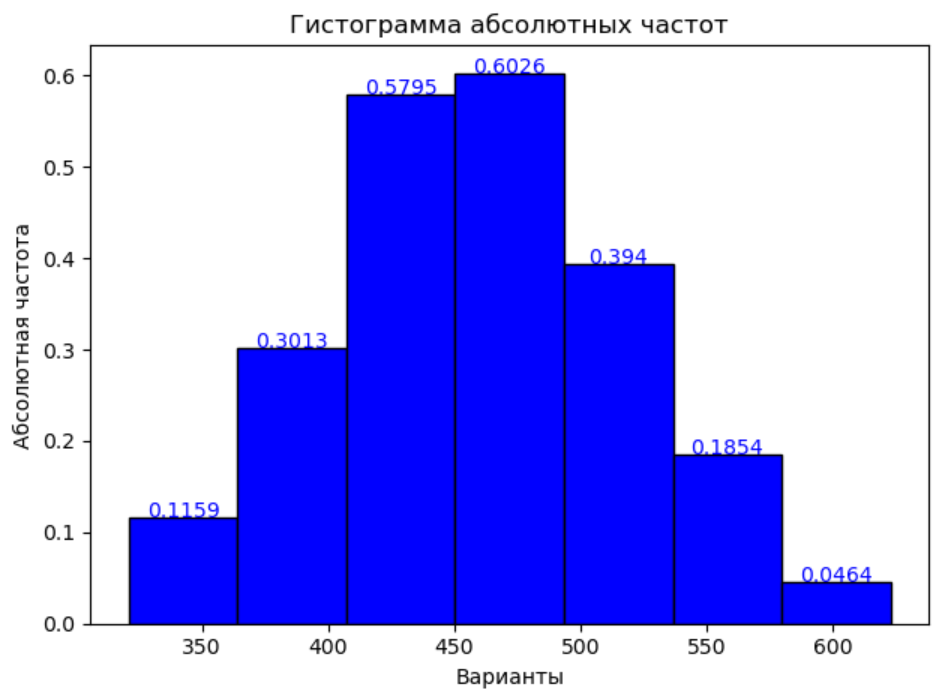
Рис. 4. Расчёты для интервального ряда

Рис. 5. Интервальный ряд с частотами (сумма отн. частот равна 1)

Визуализируем полученный интервальный ряд, построив гистограмма и полигон абсолютных (рис. 6) и относительных частот (рис. 7).

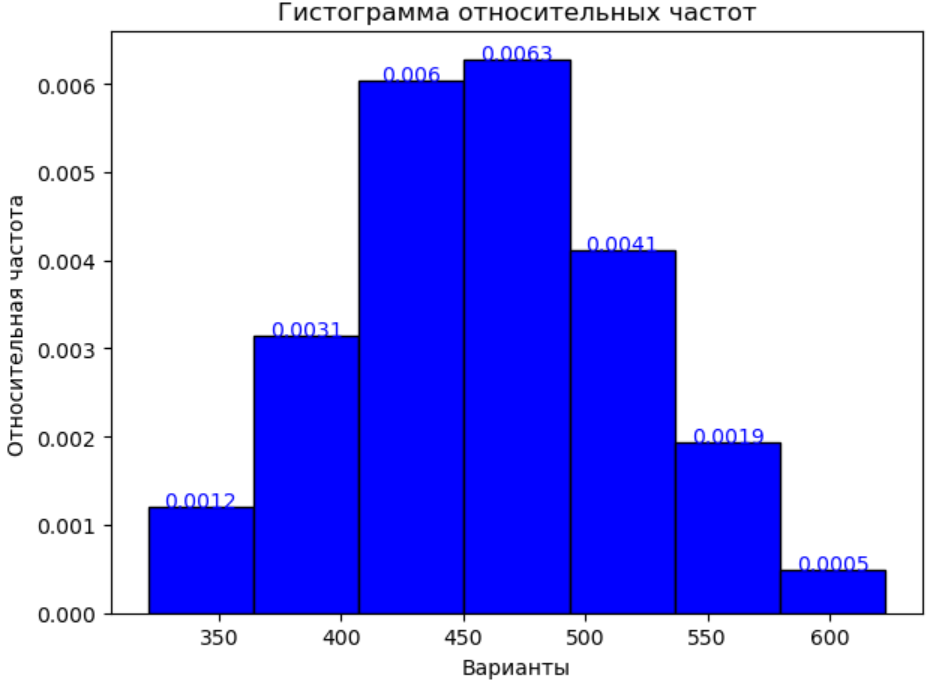
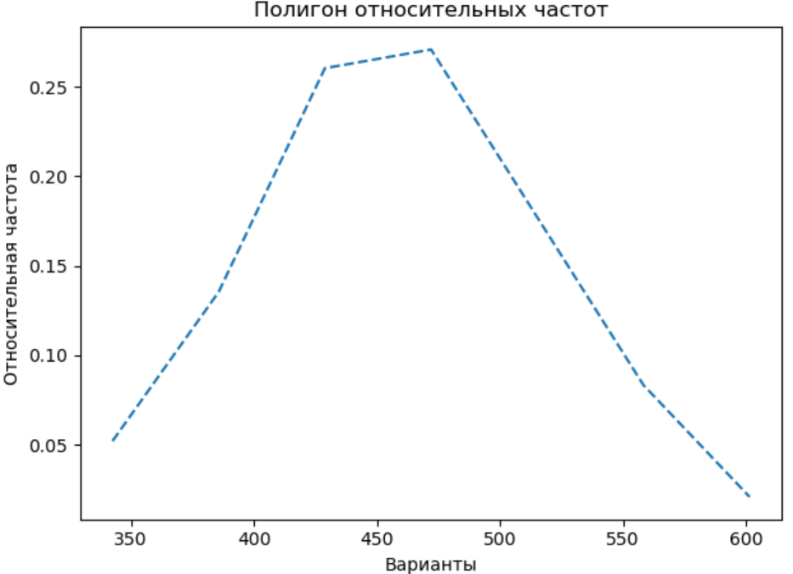
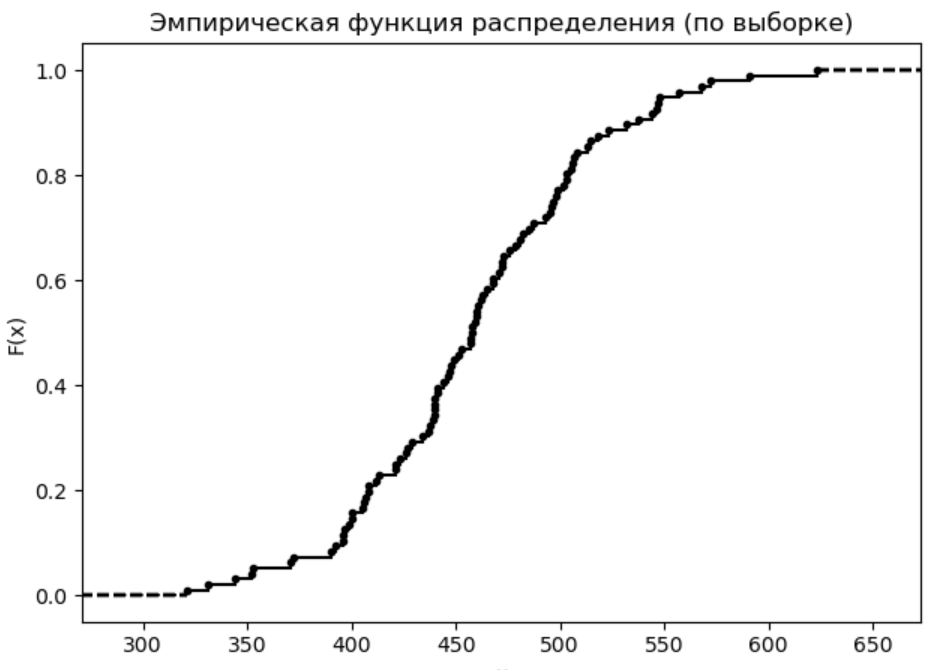
Рис. 6. Гистограмма и полигон абсолютных частот

Рис. 7. Гистограмма и полигон относительных частот

Далее построим эмпирическую функцию распределения. Ввиду неоднозначной формулировки задания построим сразу 2 графика, представленных на рис. 8 и 9 соответственно: по выборке (интуитивная интерпретация) и по интервальному ряду (строго по логике задания).

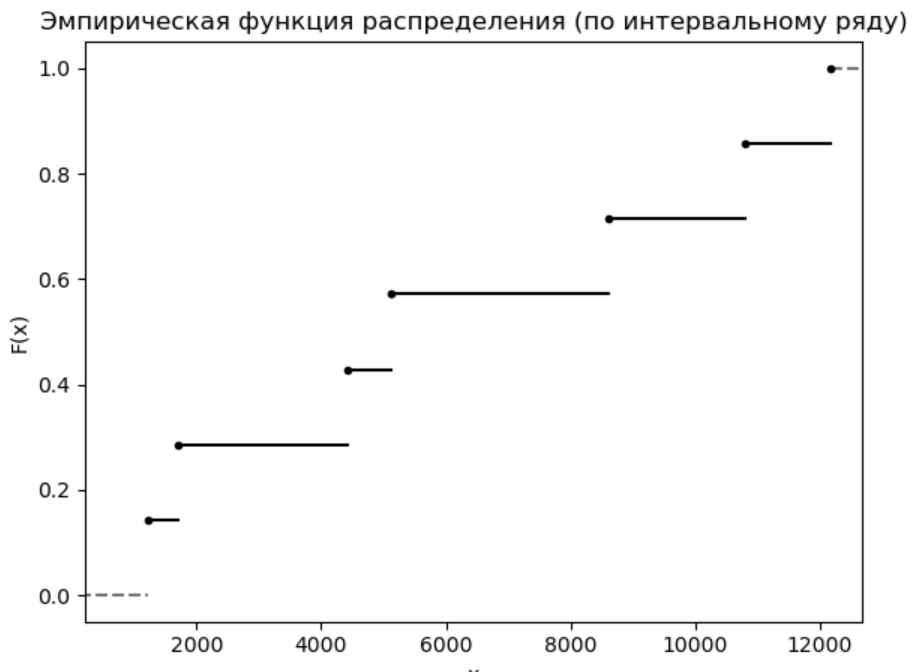
Рис. 8. Эмпирическая функция распределения (по выборке)

Рис. 9. Эмпирическая функция распределения (по интервальному ряду)

**Выводы.**

В ходе выполнения лабораторной работы был повторён и применён на практике материал с правилами формирования случайной выборки и подготовкой выборочных данных к статистическому анализу. В практической части были сделаны следующие этапы работы: формирование выборки, преобразование в ранжированный, вариационный и интервальный ряды, построение по интервальному ряду гистограмм, полигонов частот и эмпирической функции распределения.

Анализируя очертания полученных графиков распределения и сравнивая высоту столбцов в гистограмме можно сделать предварительный вывод о том, что выборка была сделана из нормального распределения. Для более точной формулировки необходимо проверить эту статистическую гипотезу.

Приложение А

исходный код разработанной прораммы lab1.py

from collections import namedtuple

from collections import Counter

import random

import math

from operator import attrgetter

Measurement = namedtuple("Measurement", ['density', 'elastic'])

selection\_size = 96 # объём выборочной совокупности

data\_file\_name = "Tabl.txt"

sample\_seed = 120 # None to random # Зерно для выборки, константа, чтобы всегда одинаково

block\_output\_size = 15 # (c учётом, что на одну запись уходит 6 символов(с пробелом))

def read\_data(filename):

# Извлекаем из файла данные и формируем генеральную совокупность.

general\_population = []

with open(filename) as file:

for line in file:

line = line.replace(',', '.').replace(' ', '').strip().split("\t")

general\_population += [Measurement(float(line[i]), float(line[i+1])) for i in range(0, len(line), 2) if line[i]]

return general\_population

def get\_sample(gen\_pop, size\_of\_gen\_pop):

# Формируем выборку из генеральной совокупности

prev\_state = random.getstate()

random.seed(sample\_seed)

sample = random.sample(gen\_pop, size\_of\_gen\_pop)

random.setstate(prev\_state)

return sample

def print\_beautiful\_sample(sample: list):

for i, value in enumerate(sample):

if i % block\_output\_size == 0 and i != 0:

print(end="\n")

print(value, end=" ")

print(end="\n")

def print\_beautiful\_variation(variation: Counter):

sorted\_density\_values = list(sorted(variation\_series\_density))

for i in range(int(len(variation)/block\_output\_size) + 1):

for index in range(i\*block\_output\_size, (i+1)\*block\_output\_size):

if index >= len(variation): break

key = sorted\_density\_values[index]

print("{0:^6}".format(key), end="")

print()

for index in range(i\*block\_output\_size, (i+1)\*block\_output\_size):

if index >= len(variation): break

key = sorted\_density\_values[index]

print("{0:^6}".format(variation[key]), end="")

print()

general\_population = read\_data(filename=data\_file\_name)

sample = get\_sample(general\_population, selection\_size)

sample\_density = [pair.density for pair in sample]

sample\_elastic = [pair.elastic for pair in sample]

print("\nСформированная выборка:")

print\_beautiful\_sample(sample\_density)

# ranked\_row = sorted(sample, reverse=False, key=attrgetter('density'))

ranked\_row = sorted(sample\_density)

print("\nРанжированный ряд (по возрастанию): ")

print\_beautiful\_sample(ranked\_row)

# Вариационный ряд значений плотности

variation\_series\_density = Counter(sample\_density)

print("\nВариационный ряд:")

print\_beautiful\_variation(variation\_series\_density)

# Для интервального ряда нужно оценить длину частичного интервала

# Для этого воспользуемся формулой Стерджеса: k = 1 + 3.322\*log10(n)

buckets\_number = int(1 + 3.322 \* math.log10(selection\_size))

min\_density, max\_density = min(sample\_density), max(sample\_density)

range\_density = max\_density - min\_density

print("\nИспользуя формулу Стерджеса рассчитаем количество групп для разбиения выборки:")

print("1 + 3.322\*lg({0}) = {1}".format(selection\_size, buckets\_number))

print("Минимальное значение ряда: ", min\_density)

print("Максимальное значение ряда:", max\_density)

print("Размах выборки", range\_density)

isInBucket = lambda x: min(int((abs(x) - min\_density) / range\_density \* buckets\_number), buckets\_number-1)

borders = [(min\_density + range\_density/buckets\_number\*i, min\_density + range\_density/buckets\_number\*(i+1)) for i in range(buckets\_number)]

buckets = [[] for i in range(buckets\_number)]

for value in sample\_density:

buckets[isInBucket(value)].append(value)

print("\nИнтервальный ряд со значениями:")

for i, border in enumerate(borders):

if i != len(borders)-1:

print("[{0:.3f} - {1:.3f}):".format(border[0], border[1]), end=" ")

else:

print("[{0:.3f} - {1:.3f}]:".format(border[0], border[1]), end=" ")

for elem in buckets[i]:

print(elem, end=" ")

print(end="\n")

print("\nИнтервальный ряд с частотами:")

print("| Интервал | Абс. частота | Отн. частота |")

for i, border in enumerate(borders):

if i != len(borders)-1:

print("| [{0:.3f} - {1:.3f}) |".format(border[0], border[1]), end=" ")

else:

print("| [{0:.3f} - {1:.3f}] |".format(border[0], border[1]), end=" ")

print("{0:^13.3f}| {1:^13.3f}|".format(len(buckets[i]), len(buckets[i])/selection\_size))

# --------------------------------- Рисуночки! ---------------------------------

import matplotlib.pyplot as plt

import numpy as np

from statistics import mean

# Гистограммы

fig, ax = plt.subplots()

ax.hist(sample\_density, bins=buckets\_number, density=False, edgecolor='black', facecolor='blue')

center\_of\_borders = [(border[0] + border[1])/2 for border in borders]

y = [len(bucket) for bucket in buckets]

ax.plot(center\_of\_borders, y, '--')

ax.set\_xlabel('Варианты')

ax.set\_ylabel('Абсолютная частота')

ax.set\_title('Гистограмма и полигон абсолютных частот')

fig.tight\_layout()

plt.show()

fig, ax = plt.subplots()

ax.hist(sample\_density, buckets\_number, weights=np.ones(len(sample\_density)) / len(sample\_density), density=False, edgecolor='black', facecolor='blue')

center\_of\_borders = [(border[0] + border[1])/2 for border in borders]

y = [len(bucket)/selection\_size for bucket in buckets]

ax.plot(center\_of\_borders, y, '--')

ax.set\_xlabel('Варианты')

ax.set\_ylabel('Относительная частота')

ax.set\_title('Гистограмма и полигон относительных частот')

fig.tight\_layout()

plt.show()

# Эмпирическая функция распределения

from statsmodels.distributions.empirical\_distribution import ECDF

fig, ax = plt.subplots()

ecdf = ECDF(sample\_density)

ax.set\_xlabel('x')

ax.set\_ylabel('F(x)')

ax.set\_title('Эмпирическая функция распределения (по выборке)')

ax.axis(xmin=min\_density-50, xmax=max(ecdf.x)+50)

ax.axis(ymin=-0.05, ymax=1.05)

for i in range(len(ecdf.x)-1):

xs = [min\_density, ecdf.x[i]]

ys = [ecdf.y[i]] \* 2

# ax.plot(xs, ys, 'r:', alpha=0.2)

ax.plot(ecdf.x[i], ecdf.y[i], "k.") # точки

xs = [ecdf.x[i], ecdf.x[i+1]]

ys = [ecdf.y[i]] \* 2

ax.plot(xs, ys, 'k-')

ax.plot([min\_density-500, ecdf.x[1]], [0, 0], 'k--', alpha=0.1)

ax.plot([max(ecdf.x), max(ecdf.x)+50], [1, 1], 'k--', alpha=0.1)

ax.plot(ecdf.x[-1], ecdf.y[-1], "k.") # точки

fig.tight\_layout()

plt.show()

fig, ax = plt.subplots()

ecdf = ECDF([sum(bucket) for bucket in buckets])

ax.set\_xlabel('x')

ax.set\_ylabel('F(x)')

ax.set\_title('Эмпирическая функция распределения (по интервальному ряду)')

ax.axis(xmin=min\_density-100, xmax=max(ecdf.x)+500)

ax.axis(ymin=-0.05, ymax=1.05)

for i in range(len(ecdf.x)-1):

xs = [min\_density, ecdf.x[i]]

ys = [ecdf.y[i]] \* 2

# ax.plot(xs, ys, 'r:', alpha=0.2)

ax.plot(ecdf.x[i], ecdf.y[i], "k.") # точки

xs = [ecdf.x[i], ecdf.x[i+1]]

ys = [ecdf.y[i]] \* 2

ax.plot(xs, ys, 'k-')

ax.plot([min\_density-1000, ecdf.x[1]], [0, 0], 'k--', alpha=0.1)

ax.plot([max(ecdf.x), max(ecdf.x)+1000], [1, 1], 'k--', alpha=0.1)

ax.plot(ecdf.x[-1], ecdf.y[-1], "k.") # точки

fig.tight\_layout()

plt.show()