ДЗ 2. Основные понятия математической статистики.

Шубин Никита СКБ172

1.1 Моделирование геометрического распределения:

Используется код написанный ранее в ДЗ 1.

```
smp = [5, 10, 100, 1000, 10000]
for i in range(len(smp)):
    for j in range(1, 6):
        file = open("Geom{}_{}.txt".format(smp[i], j), "w")
        for k in range(smp[i]):
            file.write(str(Geometric(0.2)) + ' ')
```

Все сгенерированные файлы находятся в репозитории.

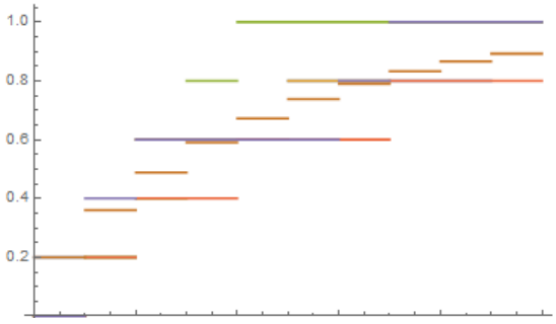
2.1 Графики эмпирической функции распределения и функции распределения.

Графики были построены с помощью Wolfram Mathematica, основываясь на данных, сгенерированных в Python.

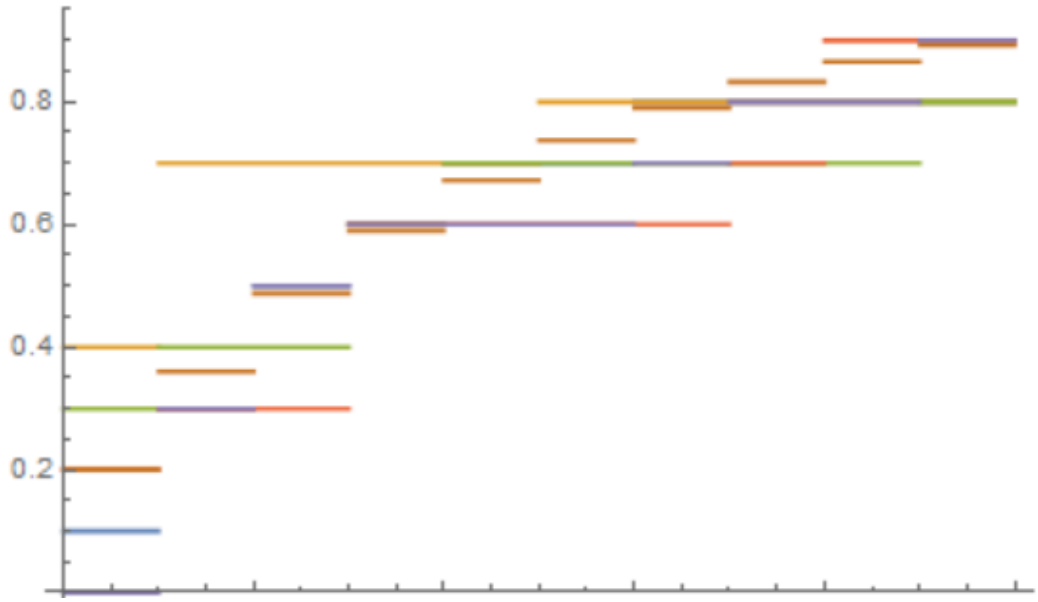
Геометрическое распределение:

При n = 5:

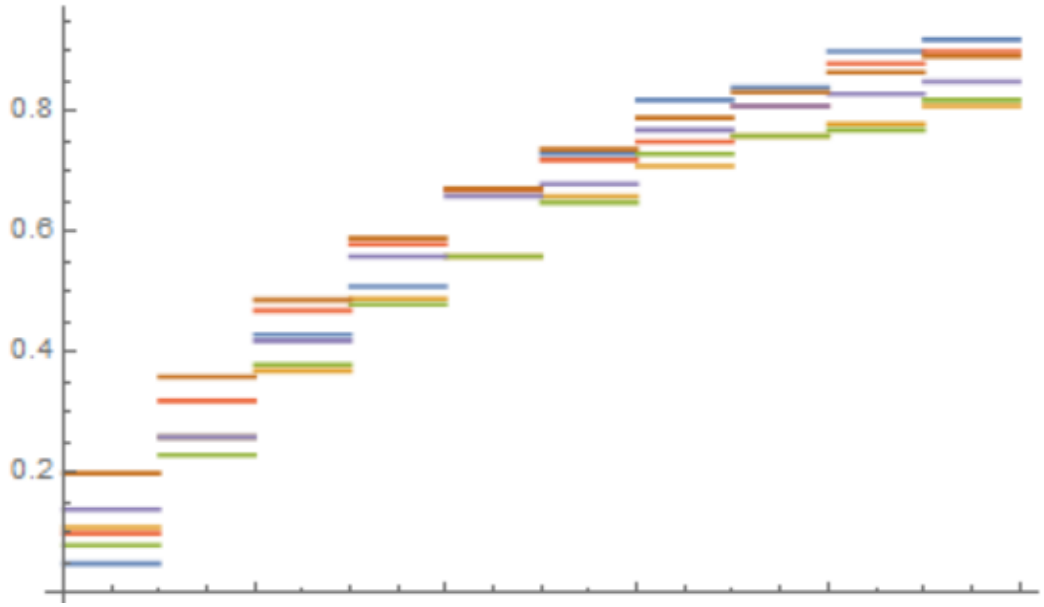
```
a1 = ReadList["C:\\Users\\MI\\Documents\\GitHub\\Math-Stat\\Geom\\Geom5_1.txt", Number];
     считать в список                                                          число
a2 = ReadList["C:\\Users\\MI\\Documents\\GitHub\\Math-Stat\\Geom\\Geom5_2.txt", Number];
     считать в список                                                          число
a3 = ReadList["C:\\Users\\MI\\Documents\\GitHub\\Math-Stat\\Geom\\Geom5_3.txt", Number];
     считать в список                                                          число
a4 = ReadList["C:\\Users\\MI\\Documents\\GitHub\\Math-Stat\\Geom\\Geom5_4.txt", Number];
     считать в список                                                          число
a5 = ReadList["C:\\Users\\MI\\Documents\\GitHub\\Math-Stat\\Geom\\Geom5_5.txt", Number];
     считать в список                                                          число
c1 = EmpiricalDistribution[a1];
     эмпирическое распределение
c2 = EmpiricalDistribution[a2];
     эмпирическое распределение
c3 = EmpiricalDistribution[a3];
     эмпирическое распределение
c4 = EmpiricalDistribution[a4];
     эмпирическое распределение
c5 = EmpiricalDistribution[a5];
     эмпирическое распределение
Plot[{CDF[c1, x], CDF[c2, x], CDF[c3, x], CDF[c4, x], CDF[c5, x], CDF[GeometricDistribution[0.2], x]}, {x, 0, 10}]
граф… функция ра… функция ра… функция ра… функция ра… функция рас… ф… геометрическое распределение
```



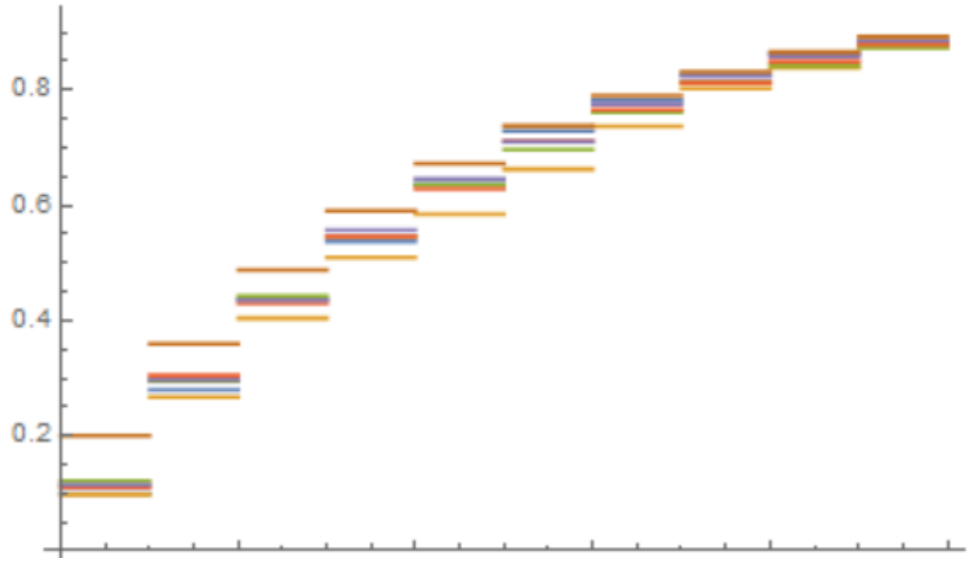
При n = 10:



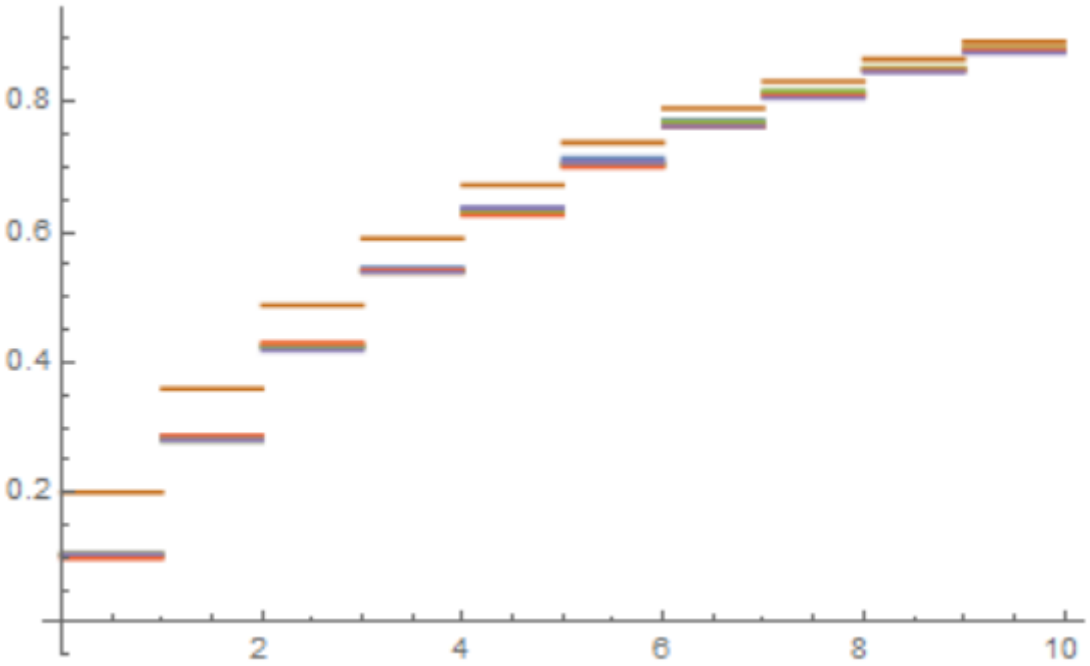
При n = 100:



При n = 1000:



При n = 10000:



3 Построение вариационного ряда выборки:

3.1 Геометрическое распределение.

```
var = []
file = open('Geom5_1.txt','r')
data = file.read()
data = ''.join(data)
data = data.split()
var.append(sorted(list(map(float,data))))

file = open('Geom5_2.txt','r')
data = file.read()
data = ''.join(data)
data = data.split()
var.append(sorted(list(map(float,data))))

file = open('Geom5_3.txt','r')
data = file.read()
data = ''.join(data)
data = data.split()
var.append(sorted(list(map(float,data))))

file = open('Geom5_4.txt','r')
data = file.read()
data = ''.join(data)
data = data.split()
var.append(sorted(list(map(float,data))))

file = open('Geom5_5.txt','r')
data = file.read()
data = ''.join(data)
data = data.split()
var.append(sorted(list(map(float,data))))

pprint.pprint(var)
```

Для n = 5:

```
[0.0, 2.0, 2.0, 5.0, 9.0]
[1.0, 2.0, 2.0, 5.0, 7.0]
[1.0, 2.0, 3.0, 3.0, 4.0]
[1.0, 2.0, 4.0, 7.0, 24.0]
[1.0, 1.0, 2.0, 6.0, 7.0]
```

Для n = 10:

```
[0.0, 1.0, 1.0, 2.0, 2.0, 3.0, 4.0, 6.0, 10.0, 14.0]
[0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 5.0, 11.0, 14.0]
[0.0, 0.0, 0.0, 1.0, 3.0, 3.0, 4.0, 9.0, 15.0, 17.0]
[0.0, 0.0, 1.0, 3.0, 3.0, 3.0, 7.0, 8.0, 8.0, 18.0]
[1.0, 1.0, 1.0, 2.0, 2.0, 3.0, 6.0, 7.0, 9.0, 15.0]
```

Квантили уровней:

Функция, вычисляющая квантили, основываясь на вариационном ряде выборки.

```
def quantils(data, f, j, k):
    quant1 = []
    quant2 = []
    quant3 = []
    n1 = int(f * len(data) + 1)
    n2 = int(j * len(data) + 1)
    n3 = int(k * len(data) + 1)
    for i in range(5):
        quant1.append(data[i][n1-1])
    for i in range(5):
        quant2.append(data[i][n2-1])
    for i in range(5):
        quant3.append(data[i][n3-1])
    print('Квантили уровня {} для n = {}'.format(f,len(data)), quant1)
    print('Квантили уровня {} для n = {}'.format(j,len(data)), quant2)
    print('Квантили уровня {} для n = {}'.format(k,len(data)), quant3)

quantils(var, 0.1, 0.5, 0.7)
```

Результаты:

```
Квантили уровня 0.1 для n = 5 [0.0, 1.0, 1.0, 1.0, 1.0]
Квантили уровня 0.5 для n = 5 [2.0, 2.0, 3.0, 4.0, 2.0]
Квантили уровня 0.7 для n = 5 [5.0, 5.0, 3.0, 7.0, 6.0]
```

```
Квантили уровня 0.1 для n = 10 [1.0, 0.0, 0.0, 0.0, 1.0]
Квантили уровня 0.5 для n = 10 [3.0, 1.0, 3.0, 3.0, 3.0]
Квантили уровня 0.7 для n = 10 [6.0, 5.0, 9.0, 8.0, 7.0]
```

```
Квантили уровня 0.1 для n = 100 [1.0, 0.0, 1.0, 1.0, 0.0]
Квантили уровня 0.5 для n = 100 [3.0, 4.0, 4.0, 3.0, 3.0]
Квантили уровня 0.7 для n = 100 [5.0, 6.0, 6.0, 5.0, 6.0]
```

```
Квантили уровня 0.1 для n = 1000 [1.0, 1.0, 0.0, 0.0, 0.0]
Квантили уровня 0.5 для n = 1000 [3.0, 3.0, 3.0, 3.0, 3.0]
Квантили уровня 0.7 для n = 1000 [5.0, 6.0, 6.0, 5.0, 5.0]
```

```
Квантили уровня 0.1 для n = 10000 [0.0, 0.0, 0.0, 1.0, 0.0]
Квантили уровня 0.5 для n = 10000 [3.0, 3.0, 3.0, 3.0, 3.0]
Квантили уровня 0.7 для n = 10000 [5.0, 5.0, 5.0, 5.0, 5.0]
```

Квантили для геометрического распределения можно сосчитать по формуле:

log(1-r)/log(1-q), где r – уровень квантиля, q – параметр геом. распр.

```
In[100]:= Log[1 - 0.1] / Log[1 - 0.2] // N
Out[100]= 0.472165

In[98]:= Log[1 - 0.5] / Log[1 - 0.2] // N
Out[101]= 3.10628

In[97]:= Log[1 - 0.7] / Log[1 - 0.2] // N
Out[102]= 5.39551
```

Нахождение разностей эмпирических функций:

Код:

```
import numpy as np
import math
smp = [5,10,100,1000,10000]
data = []
def same(k):
    mas = []
    for i in k:
        if i not in mas:
            mas.append(i)
    return(mas)
def f(a,b):
    s = 0
    for i in range(len(a)):
        if a[i] < b:
            s += 1
        else:
            break
    return(s / len(a))
for k1 in range(len(smp)):
    var = []
    maxs = []
    for i in range(len(smp)):
        file = open('C:\\Users\MI\\Documents\\GitHub\\Math-Stat\\Geom\\Geom{}_{}.txt'.format(smp[k1], i+1),'r')
        data = file.read()
        data = ''.join(data)
        data = data.split()
        var.append(sorted(list(map(float,data))))
    for i in range(4):
        for j in range(i+1, 5):
            max = -100000
            mas = sorted(np.concatenate([var[i], var[j]]))
            for k in range(len(mas)):
                if math.fabs(f(var[i], mas[k]) - f(var[j], mas[k])) > max:
                    max = math.fabs(f(var[i], mas[k]) - f(var[j], mas[k]))
            maxs.append(max)
    print("n = {}".format(smp[k1]), maxs, '\n')
```

Результат выполнения кода:

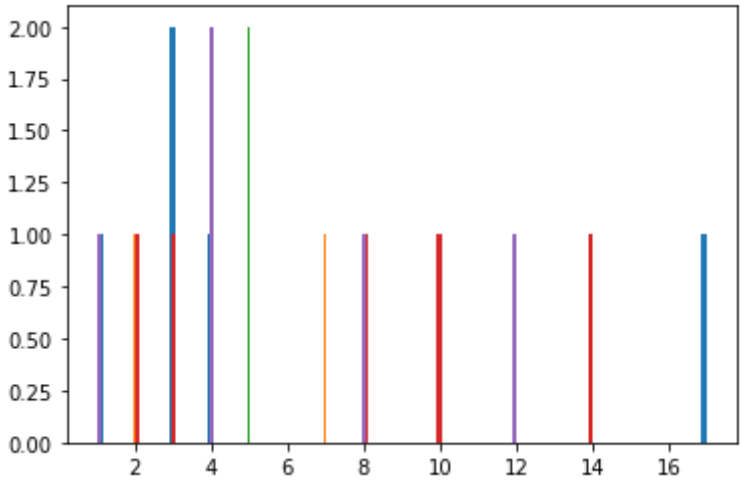
```
5 [0.2, 0.4, 0.20000000000000007, 0.20000000000000007, 0.4, 0.20000000000000007, 0.20000000000000007, 0.4, 0.4, 0.20000000000000007]

10 [0.39999999999999997, 0.19999999999999998, 0.2000000000000007, 0.1000000000000009, 0.29999999999999993, 0.39999999999999997, 0.4, 0.2000000000000007, 0.3, 0.2]

100 [0.12, 0.13, 0.06999999999999995, 0.0900000000000001, 0.03000000000000027, 0.10999999999999999, 0.09999999999999998, 0.10999999999999999, 0.09999999999999998, 0.06]

1000 [0.06599999999999995, 0.0320000000000003, 0.024999999999999967, 0.019000000000000017, 0.05000000000000044, 0.04899999999999993, 0.061000000000000054, 0.015000000000000013, 0.016
0000000014, 0.017000000000000015]
```

4 Построение гистограмм и полигон частот.

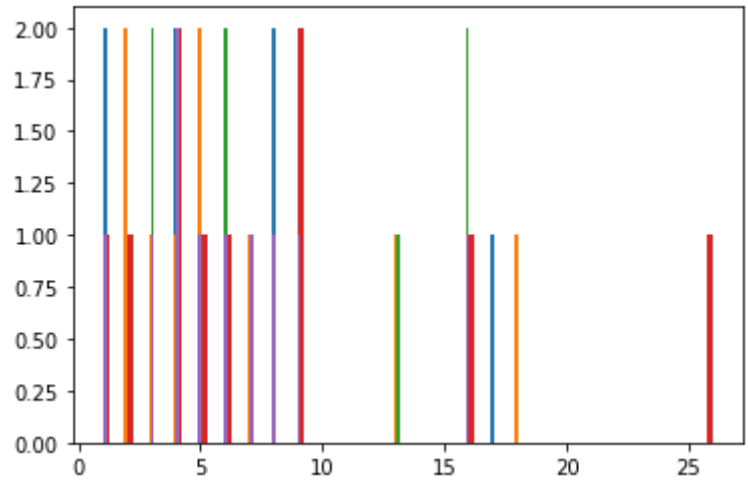
Способ построения:

```
fig = plt.figure()
sb = fig.add_subplot(1,1,1)
sb.hist(data1, 100)
sb.hist(data2, 100)
sb.hist(data3, 100)
sb.hist(data4, 100)
sb.hist(data5, 100)
plt.show
```

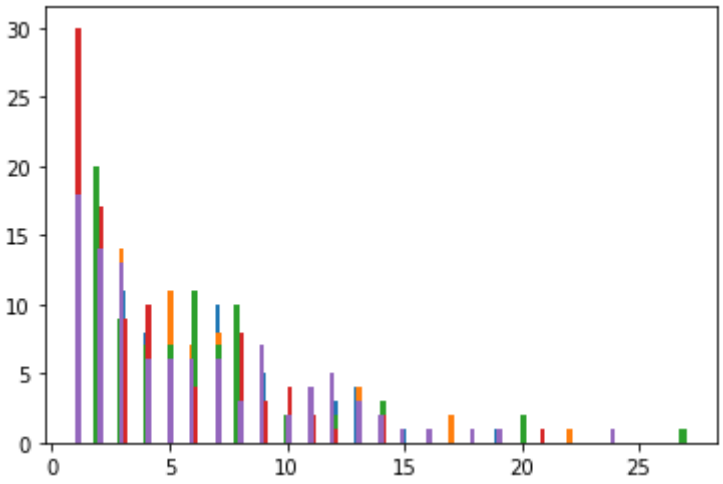
N = 5:


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
2
4
6
8
10
12
14
16

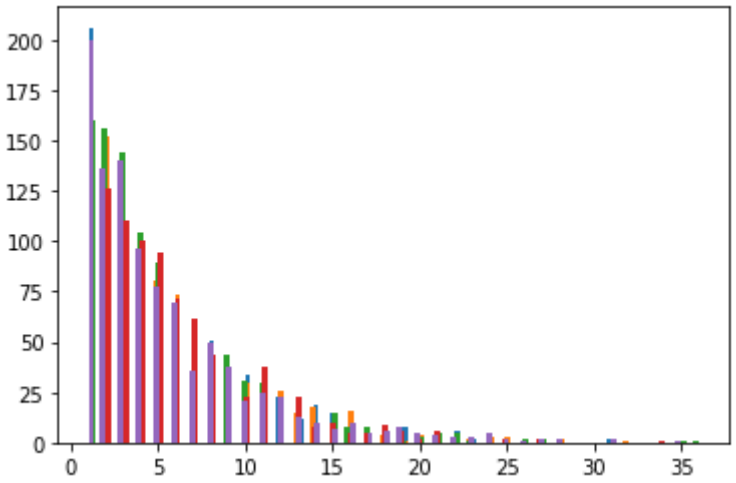
N = 10:


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
25

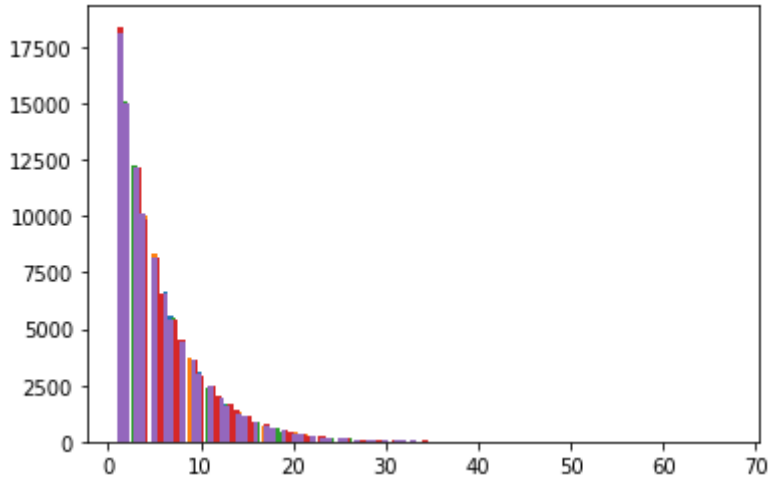
N = 100:



N = 1000:

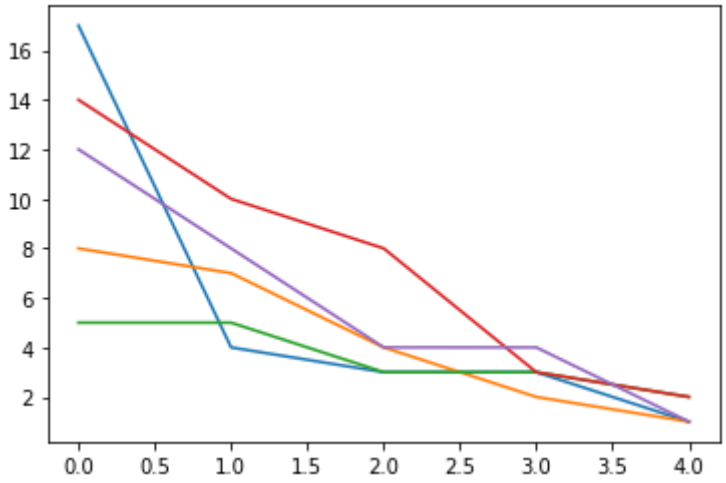


N = 100000:

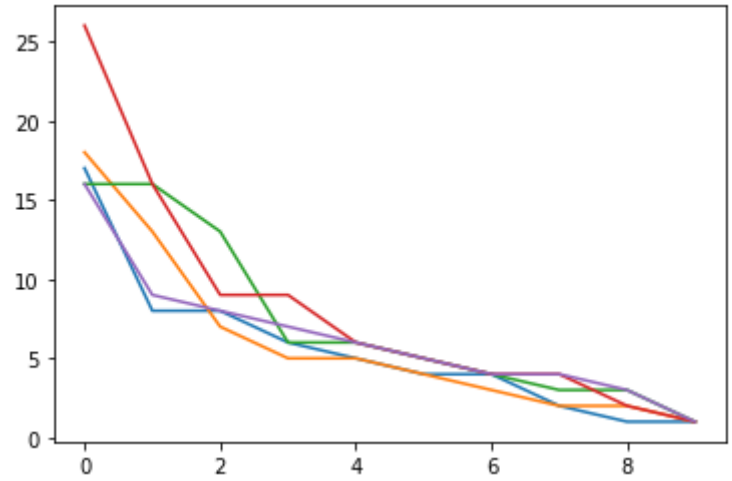


Полигоны:

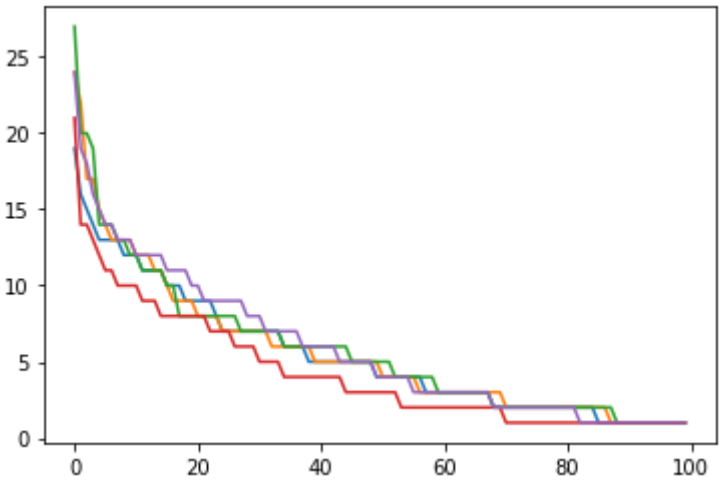
N = 5:



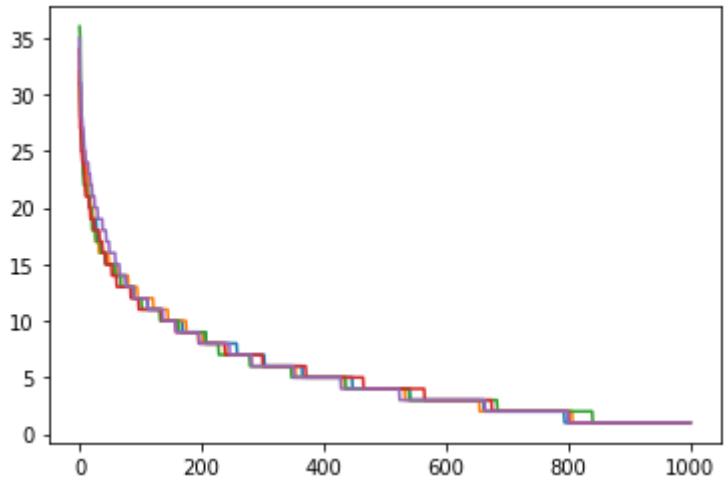
N = 10:



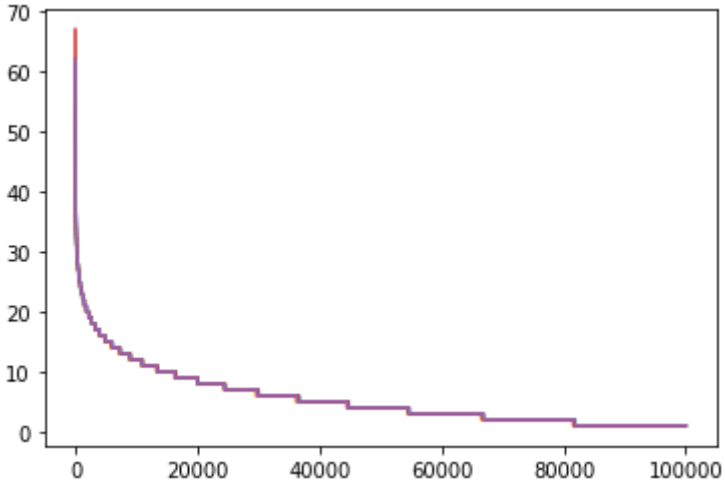
N = 100:



N = 1000:


35
30
25
20
15
10
5
0
0
200
400
600
800
1000

N = 100000:


70
60
50
40
30
20
10
0
0
20000
40000
60000
80000
100000