Университет ИТМО

Факультет ФПИ и КТ

### Отчёт

### ЛР 1

# «Теория Верятности»

Вариант 12

Студент:

Ляо Ихун

Гр.P3211

Преподаватель:  
Селина Елена Георгиевна

**Цель работы**:

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

### Описание кода:

В конце нужно выбрать какую графику рисует программа.

1. Гистограмма приведенных частот группированной выборки
2. Полигон приведенных частот группированиой выборки
3. Графику эмпической функции распределения

### Формы:

Размах:

Частота:

Математическое ожидание:

, где - частота

Среднеквадратичное наклонение:

Исправленныое среднеквадратическое отклонение:

Величина интервала:

Эмпическая функция распределение:

### Результат:

图表, 散点图

描述已自动生成

图表, 直方图

描述已自动生成

### Код:

import math  
  
import matplotlib.pyplot as plt  
import numpy as np  
N = 20  
sum = 0  
average = 0  
all\_num = [0.41, 1.63, -1.53, -0.2, 0.85, 0.09, 1.54, 0.25, 1.24, -0.26,  
 1.08, 0.42, -0.92, -0.91, 1.15, -0.82, 0.26, 0.96, 1.57, 0.72]  
all\_num.sort()  
dictionary = {}  
sorted\_string = "[ "  
for i in all\_num:  
 dictionary[i] = 0  
for i in all\_num:  
 if i in dictionary:  
 a = dictionary[i]  
 a = a + 1  
 dictionary[i] = a  
 else:  
 dictionary[i] = 1  
for i in sorted(dictionary):  
 sum = sum + i  
 dictionary[i] = dictionary[i]/20  
 sorted\_string = sorted\_string + str(i) + " "  
average = sum/20  
sorted\_string = sorted\_string + "]"  
print("Вариационный ряд:")  
print(sorted\_string)  
print("Максимальное значение:")  
print(all\_num[19])  
print("Минимальное значение:")  
print(all\_num[0])  
print("Размах выборки:")  
gap = all\_num[19] - all\_num[0]  
print(gap)  
print("Таблица частот:")  
for i in sorted(dictionary):  
 print("|%.2f" % i, end="")  
print("|")  
frequences = []  
for i in sorted(dictionary):  
 frequence = dictionary[i]  
 frequences.append(frequence)  
 if i >= 0:  
 print("|%.2f" % frequence, end="")  
 else:  
 print("|%.3f" % frequence, end="")  
print("|")  
print("Математическое ожидание:")  
math\_exp = 0  
for i in dictionary:  
 math\_exp = math\_exp + i\*dictionary[i]  
print(math\_exp)  
print("Среднеквадратическое отклонение:")  
standard\_deviation = 0  
for i in dictionary:  
 standard\_deviation = standard\_deviation + (i-average)\*\*2\*dictionary[i]  
fixed = N\*standard\_deviation/(N-1)  
fixed = fixed\*\*0.5  
standard\_deviation = standard\_deviation\*\*0.5  
print("%.4f" % standard\_deviation)  
print("Исправленныое среднеквадратическое отклонение:")  
print("%.4f" % fixed)  
print("Эмпирическая функция распределения:")  
sum\_frequence = 0  
count = 0  
while count < len(dictionary):  
 if count == 0:  
 print(" F(x)=%.2f, x<=%.2f"%(sum\_frequence,all\_num[count]))  
 sum\_frequence = sum\_frequence + frequences[count]  
 count = count + 1  
 else:  
 print(" F(x)=%.2f, %.2f<x<=%.2f"%(sum\_frequence,all\_num[count-1],all\_num[count]))  
 sum\_frequence = sum\_frequence + frequences[count]  
 count=count+1  
 if count == len(dictionary):  
 print(" F(x)=%.2f, x>%.2f" % (sum\_frequence, all\_num[count-1]))  
print("Какую графику в хотите?:")  
print("1)Гистограмму и полигон приведенных частот группированной выборки")  
print("2)Графику эмпирической функции распределения")  
## dived into group  
choice = input()  
if choice == "1":  
 n = max(all\_num) - min(all\_num)  
 h = round(n/(1+math.log2(N)),2)  
 print("значение h:%.2f"% h)  
 x\_min = min(all\_num) - h/2  
 start = x\_min  
 internals\_x = []  
 internals\_y = []  
 while start < max(all\_num):  
 temp = start  
 start = start + h  
 internals\_x.append(round(temp, 3))  
 internals\_y.append(round(start, 3))  
 dictionary\_xy = {}  
 for i in all\_num:  
 internals\_amount = len(internals\_x)  
 while internals\_amount > 0:  
 internals\_amount = internals\_amount - 1  
 if i < internals\_y[internals\_amount] and i >= internals\_x[internals\_amount]:  
 s = "[" + str(round(internals\_x[internals\_amount], 2)) + "," + str(  
 round(internals\_y[internals\_amount], 2)) + ")"  
 if s not in dictionary\_xy:  
 dictionary\_xy[s] = 0.05  
 else:  
 dictionary\_xy[s] = dictionary\_xy[s] + 0.05  
 break  
 bar\_x = []  
 bar\_y = []  
 for i in dictionary\_xy:  
 bar\_x.append(i)  
 bar\_y.append(dictionary\_xy[i])  
 plt.xticks(size=8)  
 linar\_x = []  
 linar\_y = []  
 for a,b in zip(bar\_x,bar\_y):  
 plt.text(a,b+0.005,'%.2f'%b,ha='center',va='bottom',fontsize=10)  
 linar\_x.append(a)  
 linar\_y.append(b)  
 plt.plot(linar\_x,linar\_y,color="red")  
 plt.bar(bar\_x, bar\_y)  
 plt.ylim(0,max(bar\_y)+0.1)  
 plt.show()  
elif choice=="2":  
 dictionary\_exp = dictionary.copy()  
 for i in dictionary\_exp:  
 for j in dictionary:  
 if i > j:  
 dictionary\_exp[i] = round(dictionary\_exp[i] + dictionary[j],2)  
 count = 0  
 array\_x = []  
 array\_y = []  
 for i in dictionary\_exp:  
 array\_x.append(i)  
 array\_y.append(dictionary\_exp[i])  
 axis\_x = []  
 axis\_y = []  
 while count < len(array\_x):  
 if count != len(array\_x)-1:  
 axis\_x.append(array\_x[count])  
 axis\_x.append(array\_x[count+1])  
 axis\_y.append(array\_y[count])  
 axis\_y.append(array\_y[count])  
 plt.plot(array\_x[count+1],array\_y[count],'o',color="black",markersize=2)  
 plt.plot(axis\_x,axis\_y,color="blue")  
 plt.text(array\_x[count],array\_y[count],str(array\_y[count]),fontsize=10,verticalalignment="bottom")  
 else:  
 axis\_x.append(array\_x[count])  
 axis\_x.append(array\_x[count]+0.2)  
 axis\_y.append(array\_y[count])  
 axis\_y.append(array\_y[count])  
 plt.text(array\_x[count],array\_y[count],str(array\_y[count]),fontsize=10,verticalalignment="bottom")  
 plt.plot(axis\_x,axis\_y,color="blue")  
 axis\_x.clear()  
 axis\_y.clear()  
 count = count+1  
 plt.show()