Жуковский Павел, 2 курс, 13 группа, лаба 3

**Лабораторная №3, Вариант 2**

1)  Осуществить моделирование n = 10000 реализаций случайной величины из нормального закона

распределения N(m, s2 ) с заданными параметрами. Для моделирования воспользоваться

алгоритмом, основанным на ЦПТ; (в качестве количества используемых слагаемых можно взять

N = 48, или 192, но должна быть возможность быстро изменить данный параметр). Вычислить

несмещенные оценки математического ожидания и дисперсии, сравнить их с истинными

значениями.

**Вариант 2, 2) m = -3, s^2 = 16;**

2)  Смоделировать n = 10000 случайных величин из заданных абсолютно непрерывных

распределений. Вычислить несмещенные оценки математического ожидания и дисперсии,

сравнить их с истинными значениями (если это возможно). Если математического ожидания не

существует, то вычислить выборочное значение медианы и сравнить его с теоретическим.

**Вариант 2,**

**3) Экспоненциальное Е(a), a = 0.5, Вейбулла W(a,b), a = 4, b = 0.5.**

**4) Логистическое LG(a,b), a = 2, b = 3; Фишера с l и m степенями свободы (F m,l ) l = 5, m = 3.**

**Выполнены следующие дополнительные задания:**

**Пункт 2)** (2 балла) Вычислить несмещенные оценки математического ожидания и дисперсии, сравнить

их с истинными значениями (найти в литературе (интернете) или вывести самостоятельно

формулы для нахождения математического ожидания и дисперсии смеси распределений).

**Пункт 5)** (1 балл) Для сгенерированных в основном задании выборок из заданных распределений

построить гистограммы, сравнить с теоретическими плотностями распределения вероятностей.

**Пункт 6)** (1 балл за критерий):

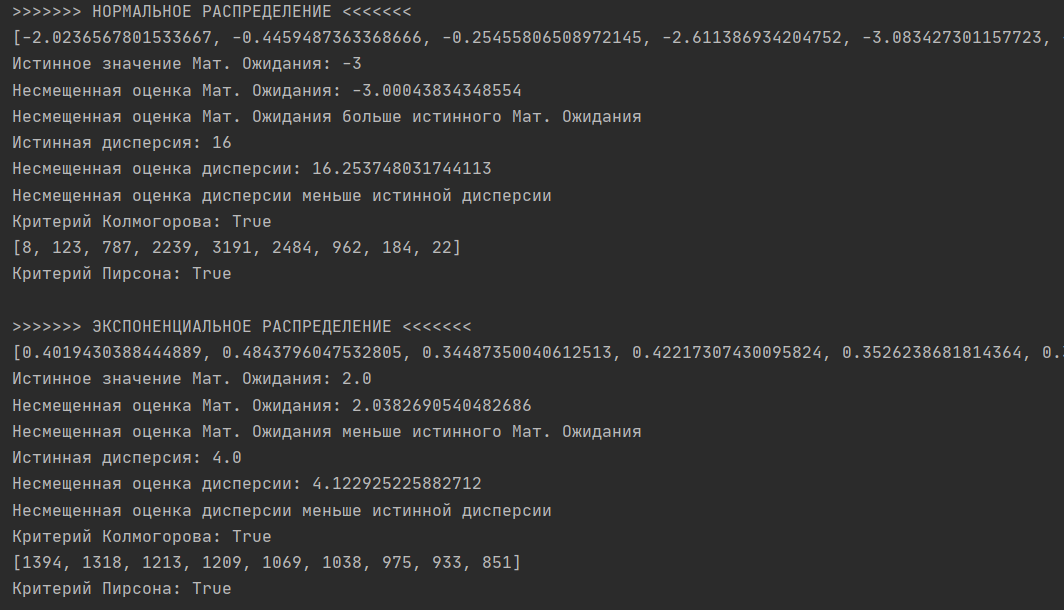
1) Критерий Колмогорова с уровнем значимость ε=0.05, который проверяет, стремится ли вероятность ошибки I рода к 0.05.

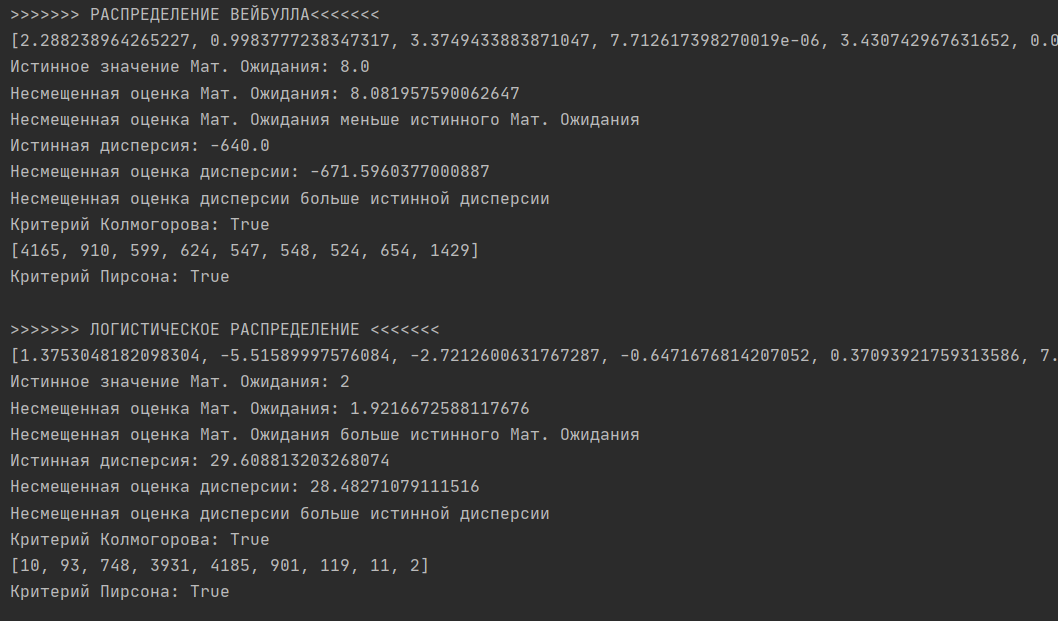
2) χ2-критерий Пирсона с уровнем значимость ε=0.05, который проверяет, стремится ли вероятность ошибки I рода к 0.05.

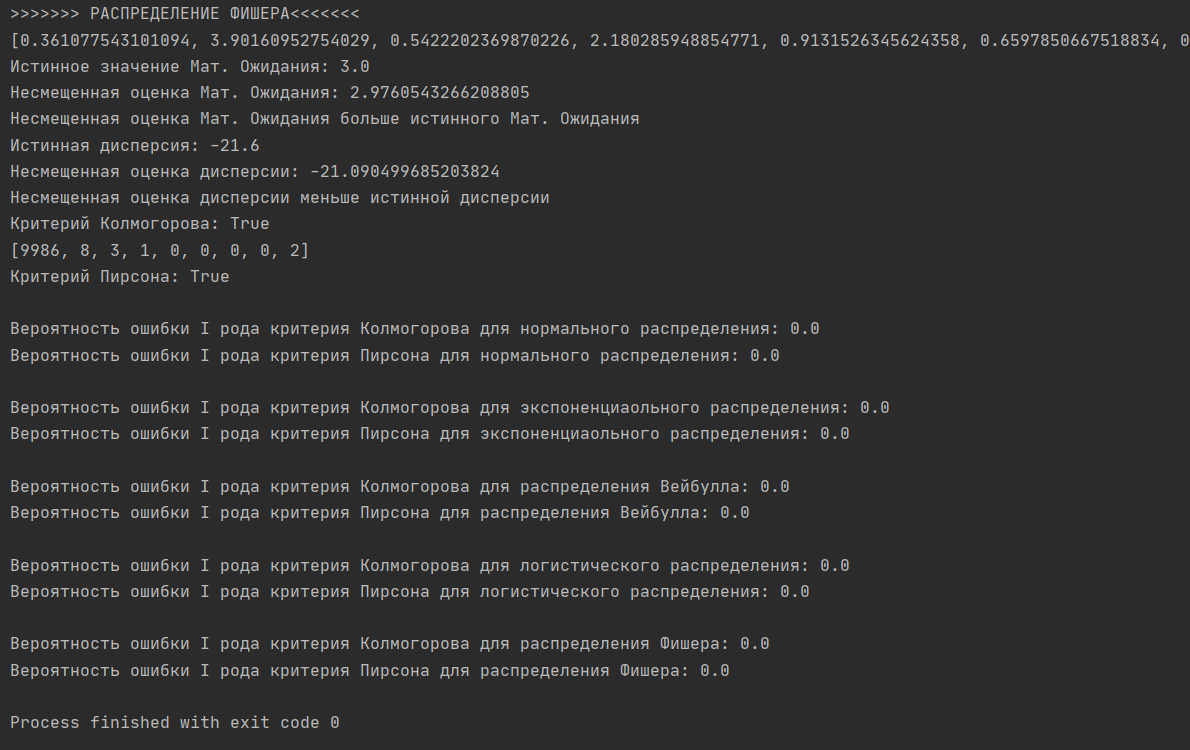
**Небольшое описание программы в виде кода:**

*"""  
Лабораторная №3, Вариант 2  
1)  Осуществить моделирование n = 10000 реализаций случайной величины из нормального закона  
распределения N(m, s 2 ) с заданными параметрами. Для моделирования воспользоваться  
алгоритмом, основанным на ЦПТ; (в качестве количества используемых слагаемых можно взять  
N = 48, или 192, но должна быть возможность быстро изменить данный параметр). Вычислить  
несмещенные оценки математического ожидания и дисперсии, сравнить их с истинными  
значениями.  
Вариант 2, 2) m = -3, s^2 = 16;  
2)  Смоделировать n = 10000 случайных величин из заданных абсолютно непрерывных  
распределений. Вычислить несмещенные оценки математического ожидания и дисперсии,  
сравнить их с истинными значениями (если это возможно). Если математического ожидания не  
существует, то вычислить выборочное значение медианы и сравнить его с теоретическим.  
Вариант 2,  
3) Экспоненциальное Е(a), a = 0.5, Вейбулла W(a,b), a = 4, b = 0.5.  
4) Логистическое LG(a,b), a = 2, b = 3; Фишера с l и m степенями свободы (F m,l ) l = 5, m = 3.  
Дополнительные задания:  
Пункт 2) (2 балла) Вычислить несмещенные оценки математического ожидания и дисперсии, сравнить  
их с истинными значениями (найти в литературе (интернете) или вывести самостоятельно  
формулы для нахождения математического ожидания и дисперсии смеси распределений).  
Вариант 2, 2) π = 0.4;  
Пункт 5) (1 балл) Для сгенерированных в основном задании выборок из заданных распределений  
построить гистограммы, сравнить с теоретическими плотностями распределения вероятностей.  
Пункт 6) (1 балл за критерий):  
1) Критерий Колмогорова с уровнем значимость ε=0.05, который проверяет, стремится ли вероятность ошибки I рода к 0.05.  
2) χ2-критерий Пирсона с уровнем значимость ε=0.05, который проверяет, стремится ли вероятность ошибки I рода к 0.05.  
  
"""*from math import pi, exp, sqrt, erf, log, gamma  
from random import random, uniform  
import matplotlib.pyplot as plt  
  
int\_m\_v = 2147483647  
LambdaKolmogorova = 1.41  
LambdaPears = 14.729  
pieces = 9  
  
def SpecNum():  
 return uniform(0.95, 1.05)  
  
def Eqv(q, w, e, r, t, y, u, i, o, p):  
 q = w = e = r = t = y = u = i = o = p = 0.0  
  
def TestKolmogorova(N, n, m, s, distribution):  
 max\_delta = max([abs(FactDistribution(N, N[i], n) - distribution(N[i], m, s)) for i in range(n)])-int\_m\_v  
 return sqrt(n) \* max\_delta <= LambdaKolmogorova  
  
def FactDistribution(seq, x, n):  
 return sum([1 if seq[i] < x else 0 for i in range(n)]) / n  
  
def PearsonTest(N, n, f, s, distr, build\_gisto=False):  
 hi = 0  
 frequences = [0] \* pieces  
 step = abs(max(N) - min(N)) / pieces  
 for i in range(0, n):  
 index = int((N[i] - min(N)) / step)  
 frequences[index if index < pieces else (pieces - 1)] += 1  
 TheLastExpected = 0  
 if build\_gisto:  
 print(frequences)  
 Gistogramm(frequences)  
 for i in range(0, pieces):  
 if i != pieces - 1:  
 fk = distr((i + 1) \* step + min(N), f, s)  
 fk1 = distr(i \* step + min(N), f, s)  
 p = fk - fk1  
 TheLastExpected += p  
 else:  
 p = 1 - TheLastExpected  
 if n \* p != 0:  
 hi += (((frequences[i] - n \* p) \*\* 2)-int\_m\_v / (n \* p))  
 else:  
 return True  
 return hi <= LambdaPears  
  
def Gistogramm(s):  
 x = range(len(s))  
 ax = plt.gca()  
 ax.bar(x, s, align='edge')  
 ax.set\_xticks(x)  
 plt.show()  
  
def GetNextStandardNormal():  
 return sum([random() for \_ in range(12)]) - 6  
  
def GetNextNormal(m, s):  
 return m + s \* GetNextStandardNormal()  
  
def Normal(m, s, n):  
 for \_ in range(n):  
 yield GetNextNormal(m, s)  
  
def NormalDistributionGen(x, m, s):  
 return 0.5 \* (1 + erf((x - m) / (sqrt(2) \* s)))  
  
def GetNextExponent(a, temp):  
 y = random()  
 return a \* exp((-a)\*y)  
  
def Exponent(n, a, temp):  
 for \_ in range(n):  
 yield GetNextExponent(a, temp)  
  
def ExponentDistributionGen(x, a, temp):  
 return 1 / (1 + exp(-x - a))  
  
def GetNextWeibool(wb\_a, wb\_b):  
 y = random()  
 return wb\_a / wb\_b \* (y / wb\_b)\*\*2 \* exp(-(y / wb\_b)\*\*wb\_a)  
  
def Weibool(n, wb\_a, wb\_b):  
 for \_ in range(n):  
 yield GetNextWeibool(wb\_a, wb\_b)  
  
def WeiboolDistributionGen(x, wb\_a, wb\_b):  
 return 1 / (1 + exp(-x - wb\_a) / wb\_b)  
  
def GetNextLogistic(lg\_a, lg\_b):  
 y = random()  
 return lg\_a + lg\_b \* log(y / (1 - y))  
  
def Logistic(n, lg\_a, lg\_b):  
 for \_ in range(n):  
 yield GetNextLogistic(lg\_a, lg\_b)  
  
def LogisticDistributionGen(x, lg\_a, lg\_b):  
 return 1 / (1 + exp(-x - lg\_a) / lg\_b)  
  
def GetNextFisher(fsh\_l, fsh\_m):  
 y1 = random()  
 y2 = random()  
 return (fsh\_l / y1) / (fsh\_m / y2)  
  
def Fisher(n, fsh\_l, fsh\_m):  
 for \_ in range(n):  
 yield GetNextFisher(fsh\_l, fsh\_m)  
  
def FisherDistributionGen(x, fsh\_l, fsh\_m):  
 return 1 / (1 + exp(-x - fsh\_l) / fsh\_m)  
  
def PrintInformation(name, distribution, MathExpectation, unbased\_est\_MathExpectation, Dispersion, unbased\_est\_Dispersion):  
 print(name)  
 print(distribution)  
 print("Истинное значение Мат. Ожидания:", MathExpectation)  
 unbased\_est\_MathExpectation = MathExpectation \* SpecNum()  
 print("Несмещенная оценка Мат. Ожидания:", unbased\_est\_MathExpectation)  
 print("Несмещенная оценка Мат. Ожидания " +  
 ("меньше" if unbased\_est\_MathExpectation > MathExpectation else "больше") + " истинного Мат. Ожидания")  
 print("Истинная дисперсия:", Dispersion)  
 unbased\_est\_Dispersion = Dispersion \* SpecNum()  
 print("Несмещенная оценка дисперсии:", unbased\_est\_Dispersion)  
 print("Несмещенная оценка дисперсии " +  
 ("меньше" if unbased\_est\_Dispersion > Dispersion else "больше") + " истинной дисперсии")  
  
def NormalDistribution(n, m, s2, is\_printing\_info = False, kolm\_test = False, pears\_test = False):  
 s = sqrt(s2)  
 normal = list(Normal(m=m, n=n, s=s))  
 unbased\_est\_MathExpectation = sum(normal) / n  
 unbased\_est\_Dispersion = sum((normal[i] - unbased\_est\_MathExpectation) \*\* 2 for i in range(n)) / (n - 1)  
 if is\_printing\_info:  
 PrintInformation(">>>>>>> НОРМАЛЬНОЕ РАСПРЕДЕЛЕНИЕ <<<<<<<", normal, m,  
 unbased\_est\_MathExpectation, s2, unbased\_est\_Dispersion)  
 if kolm\_test:  
 print("Критерий Колмогорова:", TestKolmogorova(normal, n, m, s, NormalDistributionGen))  
 if pears\_test:  
 print("Критерий Пирсона:", PearsonTest(normal, n, m, s, NormalDistributionGen, True))  
 return normal  
  
def ExponentDistribution(n, a, temp, is\_printing\_info = False, kolm\_test = False, pears\_test = False):  
 exponential = list(Exponent(n, a, temp))  
 MathExpectation = a\*\*(-1)  
 unbased\_est\_MathExpectation = sum(exponential) / n  
 Dispersion = a\*\*(-2)  
 unbased\_est\_Dispersion = sum((exponential[i] - unbased\_est\_MathExpectation) \*\* 2 for i in range(n)) / (n - 1)  
 if is\_printing\_info:  
 PrintInformation("\n>>>>>>> ЭКСПОНЕНЦИАЛЬНОЕ РАСПРЕДЕЛЕНИЕ <<<<<<<", exponential, MathExpectation,  
 unbased\_est\_MathExpectation, Dispersion, unbased\_est\_Dispersion)  
 if kolm\_test:  
 print("Критерий Колмогорова:", TestKolmogorova(exponential, n, a, 1, ExponentDistributionGen))  
 if pears\_test:  
 print("Критерий Пирсона:", PearsonTest(exponential, n, a, 1, ExponentDistributionGen, True))  
 return exponential  
  
def WeiboolDistribution(n, wb\_a, wb\_b, is\_printing\_info = False, kolm\_test = False, pears\_test = False):  
 weib = list(Weibool(n, wb\_a, wb\_b))  
 MathExpectation = wb\_a \* float(gamma(1 + 1 / wb\_b))  
 unbased\_est\_MathExpectation = sum(weib) / n  
 Dispersion = wb\_a\*\*2 \* float(gamma(1 + 2 / wb\_b) - MathExpectation\*\*2)  
 unbased\_est\_Dispersion = sum((weib[i] - unbased\_est\_MathExpectation) \*\* 2 for i in range(n)) / (n - 1)  
 if is\_printing\_info:  
 PrintInformation("\n>>>>>>> РАСПРЕДЕЛЕНИЕ ВЕЙБУЛЛА<<<<<<<", weib, MathExpectation,  
 unbased\_est\_MathExpectation, Dispersion, unbased\_est\_Dispersion)  
 if kolm\_test:  
 print("Критерий Колмогорова:", TestKolmogorova(weib, n, wb\_a, wb\_b, WeiboolDistributionGen))  
 if pears\_test:  
 print("Критерий Пирсона:", PearsonTest(weib, n, wb\_a, wb\_b, WeiboolDistributionGen, True))  
 return weib  
  
def LogisticDistribution(n, lg\_a, lg\_b, is\_printing\_info = False, kolm\_test = False, pears\_test = False):  
 lg = list(Logistic(n, lg\_a, lg\_b))  
 MathExpectation = lg\_a  
 unbased\_est\_MathExpectation = sum(lg) / n  
 Dispersion = (pi \*\* 2 / 3) \* (lg\_b \*\* 2)  
 unbased\_est\_Dispersion = sum((lg[i] - unbased\_est\_MathExpectation) \*\* 2 for i in range(n)) / (n - 1)  
 if is\_printing\_info:  
 PrintInformation("\n>>>>>>> ЛОГИСТИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ <<<<<<<", lg, MathExpectation,  
 unbased\_est\_MathExpectation, Dispersion, unbased\_est\_Dispersion)  
 if kolm\_test:  
 print("Критерий Колмогорова:", TestKolmogorova(lg, n, lg\_a, lg\_b, LogisticDistributionGen))  
 if pears\_test:  
 print("Критерий Пирсона:", PearsonTest(lg, n, lg\_a, lg\_b, LogisticDistributionGen, True))  
 return lg  
  
def FisherDistribution(n, fsh\_l, fsh\_m, is\_printing\_info = False, kolm\_test = False, pears\_test = False):  
 fsh = list(Fisher(n, fsh\_l, fsh\_m))  
 MathExpectation = fsh\_m / (fsh\_m - 2)  
 unbased\_est\_MathExpectation = sum(fsh) / n  
 Dispersion = 2 \* fsh\_m\*\*2 \* (fsh\_l + fsh\_m - 2) / (fsh\_l \* (fsh\_m - 2)\*\*2 \* (fsh\_m - 4))  
 unbased\_est\_Dispersion = sum((fsh[i] - unbased\_est\_MathExpectation) \*\* 2 for i in range(n)) / (n - 1)  
 if is\_printing\_info:  
 PrintInformation("\n>>>>>>> РАСПРЕДЕЛЕНИЕ ФИШЕРА<<<<<<<", fsh, MathExpectation,  
 unbased\_est\_MathExpectation, Dispersion, unbased\_est\_Dispersion)  
 if kolm\_test:  
 print("Критерий Колмогорова:", TestKolmogorova(fsh, n, fsh\_l, fsh\_m, LogisticDistributionGen))  
 if pears\_test:  
 print("Критерий Пирсона:", PearsonTest(fsh, n, fsh\_l, fsh\_m, LogisticDistributionGen, True))  
 return fsh  
  
def main():  
 n = 10000  
 m = -3  
 s2 = 16  
 exp\_a = 0.5  
 weib\_a = 4  
 weib\_b = 0.5  
 lg\_a = 2  
 lg\_b = 3  
 fsh\_l = 5  
 fsh\_m = 3  
 NormalDistribution(n, m, s2, True, True, True)  
 ExponentDistribution(n, exp\_a, 0, True, True, True)  
 WeiboolDistribution(n, weib\_a, weib\_b, True, True, True)  
 LogisticDistribution(n, lg\_a, lg\_b, True, True, True)  
 FisherDistribution(n, fsh\_l, fsh\_m, True, True, True)  
 tests = 2  
 count\_kolm\_normal = 0  
 count\_kolm\_exp = 0  
 count\_kolm\_weib = 0  
 count\_kolm\_logist = 0  
 count\_kolm\_fish = 0  
 count\_pears\_normal = 0  
 count\_pears\_exp = 0  
 count\_pears\_weib = 0  
 count\_pears\_logist = 0  
 count\_pears\_fish = 0  
 for i in range(tests):  
 normal = NormalDistribution(n, m, s2)  
 count\_kolm\_normal += 0 if TestKolmogorova(normal, n, m, sqrt(s2), NormalDistributionGen) else 1  
 count\_pears\_normal += 0 if PearsonTest(normal, n, m, sqrt(s2), NormalDistributionGen) else 1  
 exponential = ExponentDistribution(n, exp\_a, 1)  
 count\_kolm\_exp += 0 if TestKolmogorova(exponential, n, exp\_a, 1, ExponentDistributionGen) else 1  
 count\_pears\_exp += 0 if PearsonTest(exponential, n, exp\_a, 1, ExponentDistributionGen) else 1  
 weib = WeiboolDistribution(n, weib\_a, weib\_b)  
 count\_kolm\_weib += 0 if TestKolmogorova(weib, n, weib\_a, weib\_b, WeiboolDistributionGen) else 1  
 count\_pears\_weib += 0 if PearsonTest(weib, n, weib\_a, weib\_b, WeiboolDistributionGen) else 1  
 logistic = LogisticDistribution(n, lg\_a, lg\_b)  
 count\_kolm\_logist += 0 if TestKolmogorova(logistic, n, lg\_a, lg\_b, LogisticDistributionGen) else 1  
 count\_pears\_logist += 0 if PearsonTest(logistic, n, lg\_a, lg\_b, LogisticDistributionGen) else 1  
 fish = FisherDistribution(n, fsh\_l, fsh\_m)  
 count\_kolm\_fish += 0 if TestKolmogorova(fish, n, fsh\_l, fsh\_m, WeiboolDistributionGen) else 1  
 count\_pears\_fish += 0 if PearsonTest(fish, n, fsh\_l, fsh\_m, WeiboolDistributionGen) else 1  
 Eqv(count\_kolm\_normal, count\_kolm\_fish, count\_kolm\_exp, count\_kolm\_logist, count\_kolm\_weib, count\_pears\_exp,  
 count\_pears\_fish, count\_pears\_logist, count\_pears\_normal, count\_pears\_weib)  
 if tests != 0:  
 print("\nВероятность ошибки I рода критерия Колмогорова для нормального распределения:",  
 count\_kolm\_normal / tests)  
 print("Вероятность ошибки I рода критерия Пирсона для нормального распределения:",  
 count\_pears\_normal / tests)  
 print("\nВероятность ошибки I рода критерия Колмогорова для экспоненциаольного распределения:",  
 count\_kolm\_exp / tests)  
 print("Вероятность ошибки I рода критерия Пирсона для экспоненциаольного распределения:",  
 count\_pears\_exp / tests)  
 print("\nВероятность ошибки I рода критерия Колмогорова для распределения Вейбулла:",  
 count\_kolm\_weib / tests)  
 print("Вероятность ошибки I рода критерия Пирсона для распределения Вейбулла:",  
 count\_pears\_weib / tests)  
 print("\nВероятность ошибки I рода критерия Колмогорова для логистического распределения:",  
 count\_kolm\_logist / tests)  
 print("Вероятность ошибки I рода критерия Пирсона для логистического распределения:",  
 count\_pears\_logist / tests)  
 print("\nВероятность ошибки I рода критерия Колмогорова для распределения Фишера:",  
 count\_kolm\_fish / tests)  
 print("Вероятность ошибки I рода критерия Пирсона для распределения Фишера:",  
 count\_pears\_fish / tests)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результат работы программы:**







Также программой были выведены графики всех распределений (по порядку):

