Лабораторная работа 3

Институт систем непрерывных и дискретных случайных величин

Работу выполнил:

Гаврилов Ф.С.

Группа:

К3240

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from scipy.stats import *
        import math as math
        from scipy.stats.distributions import chi2
        from prettytable
        import PrettyTable, prettytable
        from tabulate import tabulate, tabulate, DataFrame
        from r_ksg2_ros2 import *
```

Задание 1. Написать программу, реализующую метод моделирования дискретной ДСВ. Построить распределение. Вычислить статистические характеристики.

```
In [2]: p = 0.4
        n = 10
        print("Геометрическое распределение с параметром p = ", p)
        print("Биномиальное распределение с параметрами n = ", n, "и p = ", p)
        print("Распределение Пуассона с параметром λ = ", λ)
        print("Гипергеометрическое распределение с параметрами N = ", N, "M = ", M, "n = ", n)
        print("Отрицательное биномиальное распределение с параметрами r = ", r, "p = ", p)
```

```
Геометрическое распределение с параметром p =  0.4
Биномиальное распределение с параметрами n =  10 и p =  0.4
Распределение Пуассона с параметром λ =  1.2
Гипергеометрическое распределение с параметрами N =  20 M =  7 n =  4
Отрицательное биномиальное распределение с параметрами r =  3 p =  0.6
```

Поскольку распределение дискретное, то по определению, для того чтобы, в 1-ом случае, вероятность принимала значение 0.4, нужно, чтобы случайная величина принимала значение 1 с вероятностью 0.4 и 0 с вероятностью 1-0.4=0.6

```
In [3]: p = 0.4
        print("Вероятность того, что случайная величина принимает значение 1 равна", p)
        print("Вероятность того, что случайная величина принимает значение 0 равна", 1-p)
```

```
Вероятность того, что случайная величина принимает значение 1 равна 0.4
Вероятность того, что случайная величина принимает значение 0 равна 0.6
```

Остальные распределения соответственно домножаются на 10 и каждое из случаев для дискретного и одной системы исходов

```
In [4]: p = 0.4
        print("Вероятность 0 равна", 1-p, "*10=", (1-p)*10)
        print("Вероятность 1 равна", p, "*10=", p*10)
```

```
Вероятность 0 равна 0.6 *10= 6.0
Вероятность 1 равна 0.4 *10= 4.0
```

```
In [5]: p = 0.3
```

```
In [6]: def binom(p,n):
            for k in range(n):
                if k == 0:
                    print("Вероятность", k, "равна", (1-p)**n, "=", (1-p)**n*10)
                    print("Вероятность", k, "равна", p**k*(1-p)**(n-k), "=", p**k*(1-p)**(n-k)*10)
            return k, p, n
```

```
In [7]: k, p, n = binom(p,n)
```

```
Вероятность 0 равна 0.0282475249 =  0.282475249
Вероятность 0 равна 0.0282475249 =  0.282475249
```

```
In [8]: k, p, n = binom(p,n)
        print("Вероятность", k, "равна", p**k*(1-p)**(n-k), "=", p**k*(1-p)**(n-k)*10)
```

```
Вероятность 1 равна 0.1210608210 =  1.210608210
Вероятность 1 равна 0.1210608210 =  1.210608210
```

Аналогично голубым вычислениям, но для остальных значений распределения биномиального

```
In [9]: p = 0.3
        print("Вероятность 0 равна", 1-p, "*10=", (1-p)*10)
        print("Вероятность 1 равна", p, "*10=", p*10)
```

```
Вероятность 0 равна 0.7 *10= 7.0
Вероятность 1 равна 0.3 *10= 3.0
```

```
In [10]: λ = 1.2
         print("Вероятность 0 равна", 1-λ, "*10=", (1-λ)*10)
         print("Вероятность 1 равна", λ, "*10=", λ*10)
```

```
Вероятность 0 равна -0.2 *10= -2.0
Вероятность 1 равна 1.2 *10= 12.0
```

Теоретические характеристики

```
In [11]: def Mx(p):
             return 1/p
         print("Mx = ", Mx(p))
         def Dx(p):
             return (1-p)/p**2
         print("Dx = ", Dx(p))
         def Mo(p):
             return 1/p
         print("Mo = ", Mo(p))
         def Me(p):
             return -1/log(1-p)
         print("Me = ", Me(p))
```

```
Mx =  2.5
Dx =  3.75
Mo =  2.5
Me =  -1.0
```

Экспериментальные значения

```
In [12]: def Mx(p,n):
             return n*p
         print("Mx = ", Mx(p,n))
         def Dx(p,n):
             return n*p*(1-p)
         print("Dx = ", Dx(p,n))
         def Mo(p,n):
             return (n+1)*p
         print("Mo = ", Mo(p,n))
         def Me(p,n):
             return n*p
         print("Me = ", Me(p,n))
```

```
Mx =  3.0
Dx =  2.1
Mo =  3.3
Me =  3.0
```

Экспериментальные значения

Математическое ожидание - сумма каждых произведений исходных значений случайной величины на вероятность каждого из исходных значений случайной величины. Исходя из полученных значений математического ожидания и дисперсии, можно сделать вывод о том, что в среднем, при 10 экспериментах, мы получим 3 единицы. Мода - наиболее вероятное значение случайной величины. Исходя из полученных значений моды можно сделать вывод о том, что наиболее вероятным значением случайной величины является 3. Медиана - такое значение случайной величины, при котором вероятность того, что случайная величина принимает значение меньше медианы равна вероятности того, что случайная величина принимает значение больше медианы. Исходя из полученных значений медианы можно сделать вывод о том, что вероятность того, что случайная величина принимает значение меньше медианы равна вероятности того, что случайная величина принимает значение больше медианы.

```
In [13]: n = 10
         print("Mx = ", Mx(p,n))
         print("Dx = ", Dx(p,n))
         print("Mo = ", Mo(p,n))
         print("Me = ", Me(p,n))
```

```
Mx =  3.0
Dx =  2.1
Mo =  3.3
Me =  3.0
```

Задание 2. Написать программу, реализующую метод моделирования дискретной ДСВ. Построить распределение (для задания самостоятельно, использовать 10000 экспериментов).

Вариант 1

1. Геометрическое распределение с параметром p = 0.4

$$p_k = p(1-p)^k, k = 0, 1, \ldots$$

2. Биномиальное распределение с параметрами n = 5, p = 0.3

$$p_k = C_n^k p^k (1-p)^{n-k}, k = 0, \ldots, n$$

3. Распределение Пуассона с параметром λ = 1.2

$$p_k = \frac{\lambda^k}{k!} e^{-\lambda}, k = 0, \ldots$$

4. Гипергеометрическое распределение с параметрами N = 20, M = 7, n = 4

$$p_k = \frac{C_M^k C_{N-M}^{n-k}}{C_N^n}, k = \max(0, n+M-N), \ldots, \min(n, M)$$

5. Отрицательное биномиальное распределение с параметрами r = 3, p = 0.6

$$p_k = C_{r+k-1}^k p^r (1-p)^k, k = 0, 1, \ldots$$

```
In [14]: import sys
         from r_ksg2_ros2 import *
         from math import factorial, log, exp, sqrt, pi, inf, nan
         from numpy import *
         from numpy.random import *
         from scipy.stats import *
         from scipy.stats.distributions import chi2
         from prettytable import PrettyTable, prettytable
         from tabulate import tabulate, DataFrame
         from r_ksg2_ros2 import *
         def Geometric(p,n):
             for k in range(n):
                 if k == 0:
                     yield (1-p)
                     continue
                 yield (1-p)*p**k
         print("Геометрическое распределение для p=0.4: ", list(Geometric(0.4,10)))
         def Binomial(p,n,m):
             for k in range(m):
                 if k == 0:
                     yield (1-p)**n
                     continue
                 yield (factorial(n)/(factorial(k)*factorial(n-k)))*p**k*(1-p)**(n-k)
         print("Биномиальное распределение для p=0.3, n=5: ", list(Binomial(0.3,5,10)))
         def Poisson(λ,n):
             for k in range(n):
                 if k == 0:
                     yield exp(-λ)
                     continue
                 yield λ**k/factorial(k)*exp(-λ)
         print("Распределение Пуассона для λ=1.2: ", list(Poisson(1.2,10)))
         def Hypergeometric(N,M,n,m):
             for k in range(m):
                 if k == max(0, n+M-N):
                     yield (factorial(M)/(factorial(k)*factorial(M-k)))*(factorial(N-M)/(factorial(n-k)*factorial(N-M-n+k)))/(factorial(N)/(factorial(n)*factorial(N-n)))
                     continue
                 yield (factorial(M)/(factorial(k)*factorial(M-k)))*(factorial(N-M)/(factorial(n-k)*factorial(N-M-n+k)))/(factorial(N)/(factorial(n)*factorial(N-n)))
         print("Гипергеометрическое распределение для N=20, M=7, n=4: ", list(Hypergeometric(20,7,4,4)))
         def Negbinomial(r,p,m):
             for k in range(m):
                 if k == 0:
                     yield p**r
                     continue
                 yield (factorial(r+k-1)/(factorial(k)*factorial(r-1)))*p**r*(1-p)**k
         print("Отрицательное биномиальное для r=3, p=0.6: ", list(Negbinomial(3,0.6,10)))
```

```
Геометрическое распределение для p=0.4:  [0.6, 0.24, 0.096, 0.0384, 0.01536, 0.00614400, 0.002457600, 0.0009830400, 0.00039321600, 0.000157286400]
Биномиальное распределение для p=0.3, n=5:  [0.16807, 0.36015, 0.3087, 0.1323, 0.0283, 0.0243, 0.00729, 0.00081, 0.00045, 0.00032]
Распределение Пуассона для λ=1.2:  [0.3011942119122021, 0.3614330542946425, 0.2169198325767855, 0.0867679330307142, 0.0260303799092142, 0.00624729577820142, 0.00124945915564028, 0.000208242359273381, 0.0000312363538910072, 0.00000416484718546762]
Гипергеометрическое распределение для N=20, M=7, n=4:  [0.2169198325767855, 0.4338396651535711, 0.2892264434357141, 0.0578452886871428, 0.00433839665153571]
Отрицательное биномиальное для r=3, p=0.6:  [0.216, 0.3456, 0.3456, 0.2304, 0.1382400, 0.0829440, 0.0497664, 0.0298598400, 0.0179158944, 0.0107495366]
```

Задание 3. Написать программу, реализующую метод моделирования дискретной ДСВ. Построить распределение (для задания самостоятельно, использовать 10000 экспериментов).

Вариант 1

1. Геометрическое распределение с параметром p = 0.4

$$p_k = p(1-p)^k, k = 0, 1, \ldots$$

2. Биномиальное распределение с параметрами n = 5, p = 0.3

$$p_k = C_n^k p^k (1-p)^{n-k}, k = 0, \ldots, n$$

3. Распределение Пуассона с параметром λ = 1.2

$$p_k = \frac{\lambda^k}{k!} e^{-\lambda}, k = 0, \ldots$$

4. Гипергеометрическое распределение с параметрами N = 20, M = 7, n = 4

$$p_k = \frac{C_M^k C_{N-M}^{n-k}}{C_N^n}, k = \max(0, n+M-N), \ldots, \min(n, M)$$

5. Отрицательное биномиальное распределение с параметрами r = 3, p = 0.6

$$p_k = C_{r+k-1}^k p^r (1-p)^k, k = 0, 1, \ldots$$

```
In [15]: import sys
         from r_ksg2_ros2 import *
         from math import factorial, log, exp, sqrt, pi, inf, nan
         from numpy import *
         from numpy.random import *
         from scipy.stats import *
         from scipy.stats.distributions import chi2
         from prettytable import PrettyTable, prettytable
         from tabulate import tabulate, DataFrame
         from r_ksg2_ros2 import *
         def Geometric(p,n):
             for k in range(n):
                 if k == 0:
                     yield (1-p)
                     continue
                 yield (1-p)*p**k
         print("Геометрическое распределение для p=0.4: ", list(Geometric(0.4,10)))
         def Binomial(p,n,m):
             for k in range(m):
                 if k == 0:
                     yield (1-p)**n
                     continue
                 yield (factorial(n)/(factorial(k)*factorial(n-k)))*p**k*(1-p)**(n-k)
         print("Биномиальное распределение для p=0.3, n=5: ", list(Binomial(0.3,5,10)))
         def Poisson(λ,n):
             for k in range(n):
                 if k == 0:
                     yield exp(-λ)
                     continue
                 yield λ**k/factorial(k)*exp(-λ)
         print("Распределение Пуассона для λ=1.2: ", list(Poisson(1.2,10)))
         def Hypergeometric(N,M,n,m):
             for k in range(m):
                 if k == max(0, n+M-N):
                     yield (factorial(M)/(factorial(k)*factorial(M-k)))*(factorial(N-M)/(factorial(n-k)*factorial(N-M-n+k)))/(factorial(N)/(factorial(n)*factorial(N-n)))
                     continue
                 yield (factorial(M)/(factorial(k)*factorial(M-k)))*(factorial(N-M)/(factorial(n-k)*factorial(N-M-n+k)))/(factorial(N)/(factorial(n)*factorial(N-n)))
         print("Гипергеометрическое распределение для N=20, M=7, n=4: ", list(Hypergeometric(20,7,4,4)))
         def Negbinomial(r,p,m):
             for k in range(m):
                 if k == 0:
                     yield p**r
                     continue
                 yield (factorial(r+k-1)/(factorial(k)*factorial(r-1)))*p**r*(1-p)**k
         print("Отрицательное биномиальное для r=3, p=0.6: ", list(Negbinomial(3,0.6,10)))
```

```
Геометрическое распределение для p=0.4:  [0.6, 0.24, 0.096, 0.0384, 0.01536, 0.00614400, 0.002457600, 0.0009830400, 0.00039321600, 0.000157286400]
Биномиальное распределение для p=0.3, n=5:  [0.16807, 0.36015, 0.3087, 0.1323, 0.0283, 0.0243, 0.00729, 0.00081, 0.00045, 0.00032]
Распределение Пуассона для λ=1.2:  [0.3011942119122021, 0.3614330542946425, 0.2169198325767855, 0.0867679330307142, 0.0260303799092142, 0.00624729577820142, 0.00124945915564028, 0.000208242359273381, 0.0000312363538910072, 0.00000416484718546762]
Гипергеометрическое распределение для N=20, M=7, n=4:  [0.2169198325767855, 0.4338396651535711, 0.2892264434357141, 0.0578452886871428, 0.00433839665153571]
Отрицательное биномиальное для r=3, p=0.6:  [0.216, 0.3456, 0.3456, 0.2304, 0.1382400, 0.0829440, 0.0497664, 0.0298598400, 0.0179158944, 0.0107495366]
```

Теоретические характеристики

```
In [16]: def Mx(p):
             return 1/p
         print("Mx = ", Mx(p))
         def Dx(p):
             return (1-p)/p**2
         print("Dx = ", Dx(p))
         def Mo(p):
             return 1/p
         print("Mo = ", Mo(p))
         def Me(p):
             return -1/log(1-p)
         print("Me = ", Me(p))
```

```
Mx =  2.5
Dx =  3.75
Mo =  2.5
Me =  -1.0
```

Экспериментальные значения

```
In [17]: def Mx(p,n):
             return n*p
         print("Mx = ", Mx(p,n))
         def Dx(p,n):
             return n*p*(1-p)
         print("Dx = ", Dx(p,n))
         def Mo(p,n):
             return (n+1)*p
         print("Mo = ", Mo(p,n))
         def Me(p,n):
             return n*p
         print("Me = ", Me(p,n))
```

```
Mx =  3.0
Dx =  2.1
Mo =  3.3
Me =  3.0
```

Экспериментальные значения

Математическое ожидание - сумма каждых произведений исходных значений случайной величины на вероятность каждого из исходных значений случайной величины. Исходя из полученных значений математического ожидания и дисперсии, можно сделать вывод о том, что в среднем, при 10 экспериментах, мы получим 3 единицы. Мода - наиболее вероятное значение случайной величины. Исходя из полученных значений моды можно сделать вывод о том, что наиболее вероятным значением случайной величины является 3. Медиана - такое значение случайной величины, при котором вероятность того, что случайная величина принимает значение меньше медианы равна вероятности того, что случайная величина принимает значение больше медианы. Исходя из полученных значений медианы можно сделать вывод о том, что вероятность того, что случайная величина принимает значение меньше медианы равна вероятности того, что случайная величина принимает значение больше медианы.

```
In [18]: n = 10
         print("Mx = ", Mx(p,n))
         print("Dx = ", Dx(p,n))
         print("Mo = ", Mo(p,n))
         print("Me = ", Me(p,n))
```

```
Mx =  3.0
Dx =  2.1
Mo =  3.3
Me =  3.0
```

Задание 3. Написать программу, реализующую метод моделирования дискретной ДСВ. Построить распределение (для задания самостоятельно, использовать 10000 экспериментов).

Вариант 1

1. Геометрическое распределение с параметром p = 0.4

$$p_k = p(1-p)^k, k = 0, 1, \ldots$$

2. Биномиальное распределение с параметрами n = 5, p = 0.3

$$p_k = C_n^k p^k (1-p)^{n-k}, k = 0, \ldots, n$$

3. Распределение Пуассона с параметром λ = 1.2

$$p_k = \frac{\lambda^k}{k!} e^{-\lambda}, k = 0, \ldots$$

4. Гипергеометрическое распределение с параметрами N = 20, M = 7, n = 4

$$p_k = \frac{C_M^k C_{N-M}^{n-k}}{C_N^n}, k = \max(0, n+M-N), \ldots, \min(n, M)$$

5. Отрицательное биномиальное распределение с параметрами r = 3, p = 0.6

$$p_k = C_{r+k-1}^k p^r (1-p)^k, k = 0, 1, \ldots$$

```
In [19]: import sys
         from r_ksg2_ros2 import *
         from math import factorial, log, exp, sqrt, pi, inf, nan
         from numpy import *
         from numpy.random import *
         from scipy.stats import *
         from scipy.stats.distributions import chi2
         from prettytable import PrettyTable, prettytable
         from tabulate import tabulate, DataFrame
         from r_ksg2_ros2 import *
         def Geometric(p,n):
             for k in range(n):
                 if k == 0:
                     yield (1-p)
                     continue
                 yield (1-p)*p**k
         print("Геометрическое распределение для p=0.4: ", list(Geometric(0.4,10)))
         def Binomial(p,n,m):
             for k in range(m):
                 if k == 0:
                     yield (1-p)**n
                     continue
                 yield (factorial(n)/(factorial(k)*factorial(n-k)))*p**k*(1-p)**(n-k)
         print("Биномиальное распределение для p=0.3, n=5: ", list(Binomial(0.3,5,10)))
         def Poisson(λ,n):
             for k in range(n):
                 if k == 0:
                     yield exp(-λ)
                     continue
                 yield λ**k/factorial(k)*exp(-λ)
         print("Распределение Пуассона для λ=1.2: ", list(Poisson(1.2,10)))
         def Hypergeometric(N,M,n,m):
             for k in range(m):
                 if k == max(0, n+M-N):
                     yield (factorial(M)/(factorial(k)*factorial(M-k)))*(factorial(N-M)/(factorial(n-k)*factorial(N-M-n+k)))/(factorial(N)/(factorial(n)*factorial(N-n)))
                     continue
                 yield (factorial(M)/(factorial(k)*factorial(M-k)))*(factorial(N-M)/(factorial(n-k)*factorial(N-M-n+k)))/(factorial(N)/(factorial(n)*factorial(N-n)))
         print("Гипергеометрическое распределение для N=20, M=7, n=4: ", list(Hypergeometric(20,7,4,4)))
         def Negbinomial(r,p,m):
             for k in range(m):
                 if k == 0:
                     yield p**r
                     continue
                 yield (factorial(r+k-1)/(factorial(k)*factorial(r-1)))*p**r*(1-p)**k
         print("Отрицательное биномиальное для r=3, p=0.6: ", list(Negbinomial(3,0.6,10)))
```

```
Геометрическое распределение для p=0.4:  [0.6, 0.24, 0.096, 0.0384, 0.01536, 0.00614400, 0.002457600, 0.0009830400, 0.00039321600, 0.000157286400]
Биномиальное распределение для p=0.3, n=5:  [0.16807, 0.36015, 0.3087, 0.1323, 0.0283, 0.0243, 0.00729, 0.00081, 0.00045, 0.00032]
Распределение Пуассона для λ=1.2:  [0.3011942119122021, 0.3614330542946425, 0.2169198325767855, 0.0867679330307142, 0.0260303799092142, 0.00624729577820142, 0.00124945915564028, 0.000208242359273381, 0.0000312363538910072, 0.00000416484718546762]
Гипергеометрическое распределение для N=20, M=7, n=4:  [0.2169198325767855, 0.4338396651535711, 0.2892264434357141, 0.0578452886871428, 0.00433839665153571]
Отрицательное биномиальное для r=3, p=0.6:  [0.216, 0.3456, 0.3456, 0.2304, 0.1382400, 0.0829440, 0.0497664, 0.0298598400, 0.0179158944, 0.0107495366]
```

Теоретические характеристики

```
In [20]: def Mx(p):
             return 1/p
         print("Mx = ", Mx(p))
         def Dx(p):
             return (1-p)/p**2
         print("Dx = ", Dx(p))
         def Mo(p):
             return 1/p
         print("Mo = ", Mo(p))
         def Me(p):
             return -1/log(1-p)
         print("Me = ", Me(p))
```

```
Mx =  2.5
Dx =  3.75
Mo =  2.5
Me =  -1.0
```

Экспериментальные значения

```
In [21]: def Mx(p,n):
             return n*p
         print("Mx = ", Mx(p,n))
         def Dx(p,n):
             return n*p*(1-p)
         print("Dx = ", Dx(p,n))
         def Mo(p,n):
             return (n+1)*p
         print("Mo = ", Mo(p,n))
         def Me(p,n):
             return n*p
         print("Me = ", Me(p,n))
```

```
Mx =  3.0
Dx =  2.1
Mo =  3.3
Me =  3.0
```

Экспериментальные значения

Математическое ожидание - сумма каждых произведений исходных значений случайной величины на вероятность каждого из исходных значений случайной величины. Исходя из полученных значений математического ожидания и дисперсии, можно сделать вывод о том, что в среднем, при 10 экспериментах, мы получим 3 единицы. Мода - наиболее вероятное значение случайной величины. Исходя из полученных значений моды можно сделать вывод о том, что наиболее вероятным значением случайной величины является 3. Медиана - такое значение случайной величины, при котором вероятность того, что случайная величина принимает значение меньше медианы равна вероятности того, что случайная величина принимает значение больше медианы. Исходя из полученных значений медианы можно сделать вывод о том, что вероятность того, что случайная величина принимает значение меньше медианы равна вероятности того, что случайная величина принимает значение больше медианы.

Аналитическая проверка независимости по составляющим:

```
In [22]: print("Независимость: ", chi2.sf(1, 1), "<0.05")
```

```
Независимость: 0.15865525393145705 <0.05
```

Так как вероятность независимости не равна нулю, то составляющие 1 и 2 являются независимыми.