numpy

Хашин С.И.
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский университет

.

Случайные числа в Питоне

.

Иваново-2023

План

Список слайдов

Выборочное среднее и дисперсия

Пусть в векторе x содержатся N значений СВ ξ: $(x_0, \ldots, x_{N-1})$. Вычисления среднего:

```
np.mean(x)
```

Вычисления среднекваратичного (стандартного) отклонения σ:

```
sigma =  x.std()
sigma = np.std(x) # другая форма
```

Если же мы не хотим строить весь вектор x, например, если значения СВ получаются в некотором процессе обработки больших файлов и длина вектора может оказаться весьма большой?

class sigma

```
class sigma:        # Класс для среднего и дисперсии
    __slots__ = ['cnt', 'sum', 'sum2']
    def __init__(self):
        self.cnt = self.sum = self.sum2 = 0.0

    def add(self, x):     # очередное значение (x)
        self.cnt +=1
        self.sum += x
        self.sum2 += x*x

    def mean(self): return self.sum/self.cnt
    def std(self):
        return np.sqrt(self.sum2/self.cnt
                       - (self.sum/self.cnt)**2)
```

class sigma

```
s = sigma()
for k in range(6): s.add(k)
print(s.mean())
print(s.std())
```

Задание

Найти погрешность выборочного среднего и ср.кв.отклонения при $N = 10, 100, 1000, 10\,000$ испытаний для следующих СВ:

- Бернуллиевская СВ(1/2) и (1/6)
- Биномиальная СВ (n=20, p=1/2)
- `np.random.randint(10,20)`
- `np.random.random()`
- `np.random.uniform(10,20)`
- Геометрическая СВ(p=1/6)
- Пуассоновская СВ ($\lambda = 5$)
- Экспоненциальная СВ ($\lambda = 5$)
- Нормальная СВ ($a = 2, \sigma = 1$)
- Нормальная СВ ($a = 2, \sigma = 10$)
- Нормальная СВ ($a = 2, \sigma = 0.1$)

Как строить вектора СВ На Питоне

Все функции имеют параметр size, задающий размер получаемого вектора.

```
random.binomial(1, p)  # Бернуллиевская
random.binomial(n, p)  # Биномиальная
np.random.randint(a,b) # Равномерная целая [a,b]
np.random.random()     # Равномерная [0,1]
np.random.uniform(a,b) # Равномерная [a,b]
np.random.geometric(p) # Геометрическая
np.random.exponential(lam)
np.random.normal(a, sigma)
```

Доверительный интервал для МО и σ^2

(Правило 3σ). Пусть в векторе x содержатся N значений СВ ξ: $(x_0, \ldots, x_{N-1})$ и $\overline{a}$, $\overline{\sigma}^2$ — её выборочное среднее и дисперсия. Тогда с надежностью 99.5% истинное значение МО попадает в интервал

$$\overline{a} \pm 3\overline{\sigma}$$

а дисперсии в интервал

$$\overline{\sigma}^2 \left(1 \pm \frac{3}{\sqrt{2N}}\right).$$

Замечание. Формула доверительного интервала для дисперсии лишь для нормальных СВ.
Задание. Проверить эти формулы для СВ из предыдущего слайда.

Задача об опросе

Пусть в предвыборном социологическом опросе ста человек голоса за кандидатов A,B,C,D на выборах распределились так: $30, 15, 10, 5$, остальные 40 человек не определились.
Каковы доверительные интервалы для доли голосов за A,B,C,D на выборах?
(Количество голосов за каждого из кандидатов — биномиальная СВ).

Задача об опросе

η_A: МО $= 30$, $\sigma^2 = 60 \cdot p_i \cdot (1 - p_i) = 15$. Таким образом, доверительных интервал на 60 голосов: $30 \pm 3\sqrt{(15)}$

η_A: МО $= 30$, $\sigma^2 = 60 \cdot 1/2 \cdot 1/2$
η_B: МО $= 15$, $\sigma^2 = 60 \cdot 1/3 \cdot 3/4$
η_C: МО $= 10$, $\sigma^2 = 60 \cdot 1/6 \cdot 5/6$
η_D: МО $= 10$, $\sigma^2 = 60 \cdot 5/60 \cdot 55/60$

А теперь в процентах:
A: 30.6% ... 69.4%.
B: 8.2% ... 41.8%.
C: 2.2% ... 31.1%.
D: -2.4% ... 19.3%.

Проверка гипотезы

Пусть дискретная СВ ξ принимает значения $v_1, \ldots, v_k$ с вероятностями $p_1, \ldots, p_k$ соответственно. Если мы возьмём N независимых значений этой СВ, то значение v_i она будет принимать в среднем $p_i N$ раз.
Проведем N независимых испытаний СВ ξ и обозначим через η_i сколько раз появилось значение v_i.
η_i является биномиальной СВ с параметрами p_i, N.

Проверка гипотезы-2

Итак, пусть
x_i — сколько раз СВ ξ в серии из N испытаний приняла значение v_i,
$y_i = p_i \cdot N$ — сколько раз должна была принять в среднем.
Тогда СВ

$$\frac{x_i - y_i}{\sqrt{y_i}}$$

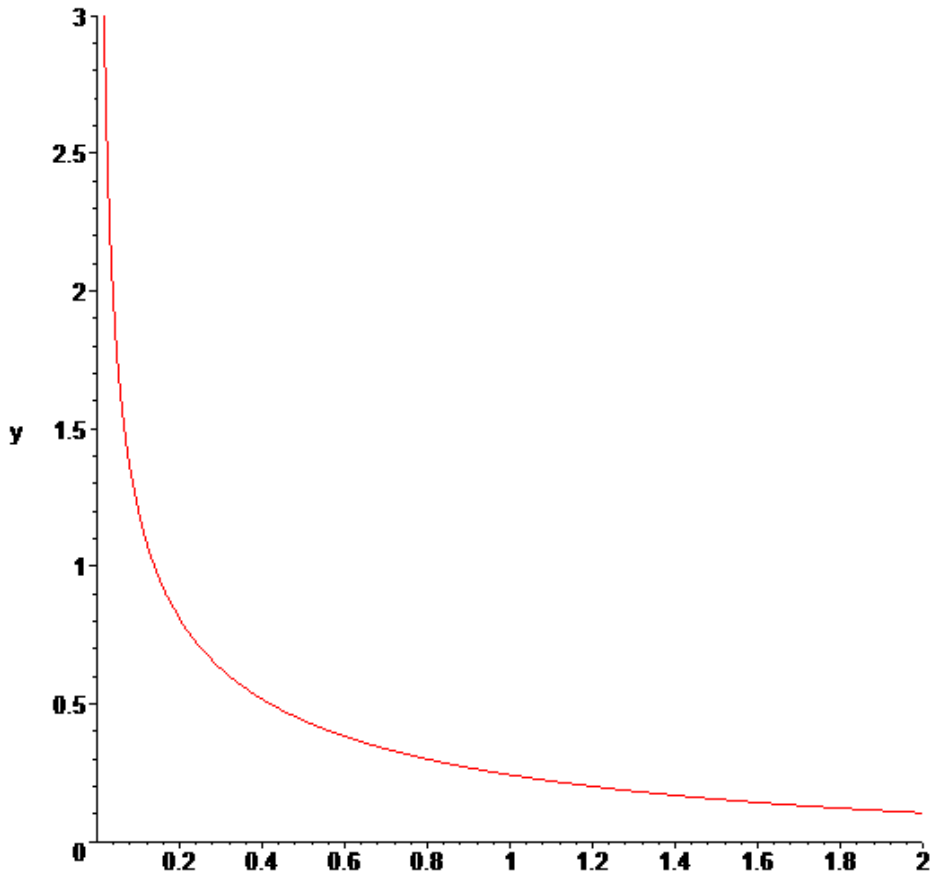
приближеённо можно считать стандартной, то есть нормальной с параметрами $(0, 1)$. Сумма квадратов таких СВ

$$\sum_{i=1}^{k} \frac{(x_i - y_i)^2}{y_i}$$

имеет распределение хи-квадрат с $k - 1$ степенями свободы.

Квадрат стандартной СВ

Пусть ξ — стандарная СВ, то есть нормальная с параметрами $(0, 1)$ и пусть $\eta = \xi^2$.
Тогда $\mathrm{MO}(\eta) = 1$ и
дисперсия $\sigma^2(\eta) = 2$.
Её плотность распределения:

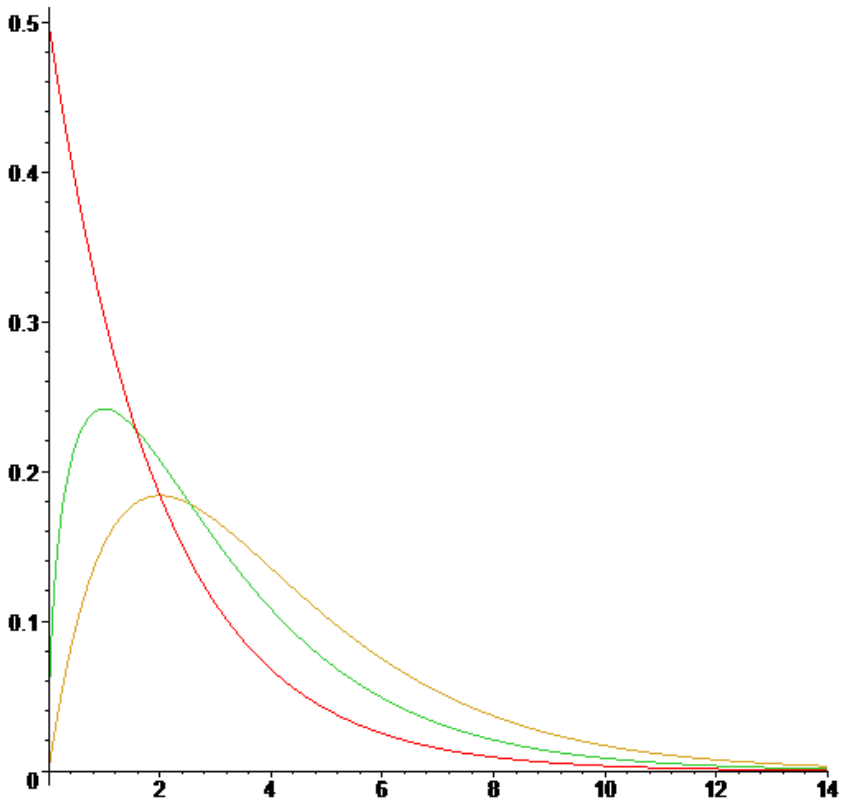


Распределение хи-квадрат

Пусть ξ_i — независимые стандарные СВ и пусть

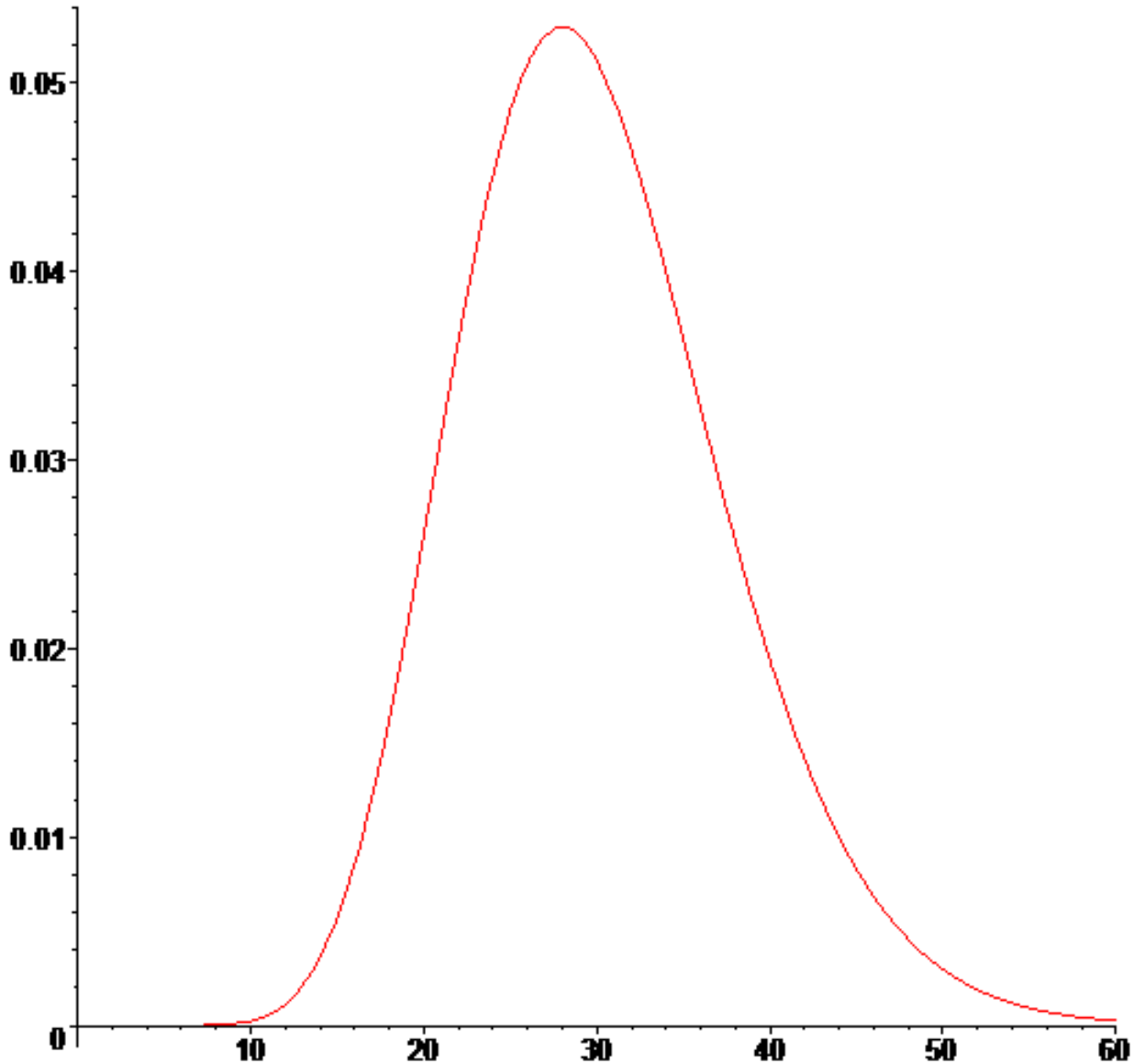
$$\eta = \xi_1^2 + \dots \xi_n^2.$$

СВ η имеет распределение хи-квадрат с n степенями свободы.
$\mathrm{MO}(\eta) = n$ и дисперсия $\sigma^2(\eta) = 2n$.
Её плотность распределения (2-3-4 степени свободы):



Распределение хи-квадрат(30)

При 30 степенях свободы:



Критерий Пирсона

Сумма квадратов k независимых стандартных СВ не может быть ни слишком большой, ни слишком маленькой! А какой может быть?

Требуется задать надежность. Обычно берут два варианта: надежность 95% и 99.5%. Для каждого k и заданной надежности существует свой доверительный интервал.

k	95	%	99.5	%
1	0.00	3.84	0.00	7.88
2	0.10	5.99	0.00	10.60
4	0.21	9.49	0.07	14.86
10	3.94	18.31	2.16	25.19
20	10.85	31.41	7.43	40.00
50	34.76	70.00	27.99	79.49
100	77.93	124.84	67.33	140.17
200	168.28	233.99	152.24	255.26

Критерий Пирсона

Матожидание СВ η равно k, дисперсия $2k$.
Согласно центральной предельной теореме, сумма любых независимых СВ аппроксимируется нормальной величиной.
Уже при $k \geq 10$ для верхней границы достаточно хорошую аппроксимацию даёт «правило 3σ», или «правило 2σ»:

«правило 2σ», с надежностью 95%: $\eta < k + 2\sqrt{2k}$.

«правило 3σ», с надежностью 99.5%: $\eta < k + 3\sqrt{2k}$.

Однако, нижняя граница даёт удовлетворительное приближение лишь при $k \geq 100$.

Пирсон на Excel

	A	B	C	D	E	F
1	X	Y	Пирсон:	11.72	*=СУММ(C2:C11)	
2	82	100	3.24	3.32511	*=ХИ2ОБР(0.95, 9)	
3	110	100	1	16.919	*=ХИ2ОБР(0.05, 9)	
4	102	100	0.04			
5	120	100	4			
6	93	100	0.49			
7	88	100	1.44			
8	101	100	0.01			
9	95	100	0.25			
10	111	100	1.21	*=(A10-B10)^2/B10		
11	98	100	0.04	*=(A11-B11)^2/B11		

на Питоне

```
from scipy.stats import chi2
...
n = 7  # количество степеней свободы
# точное значение, "2sigma":
x0, x1 = chi2.ppf(0.05, n), chi2.ppf(0.95, n)
# точное значение, "3sigma":
x0, x1 = chi2.ppf(0.005, n), chi2.ppf(0.995, n)

# приближённое, "2sigma":
sigma = np.sqrt(2*n)
y0, y1 = n - 2*sigma, n+2*sigma
# приближённое, "3sigma":
y0, y1 = n - 3*sigma, n+3*sigma
```

Критерий Пирсона

```
def Pirson(X,Y): # X-реальные частоты, Y-теоретич.
  sum_x = sum(X)
  sum_y = sum(Y)
  co = sum_x/sum_y    # Y[i]-> co*Y[i]
  xi2 = 0
  for i in range(len(X)):
    yi = Y[i]*co
    xi2 += (X[i]-yi)**2/yi
  return xi2
```

Критерий Пирсона-2

```
def Pirson2(X,Y): # X-реальные частоты, Y - теория
  minY = 10       # наименьшее допустимое значение Y
  k_svob = 0      # количество степеней свободы у xi2
  sum_x, sum_y = sum(X), sum(Y)
  Y = Y*sum_x/sum_y   # теперь sum(X)=sum(Y)
  xi2 = y_rest = x_rest = 0 # сумма "хвостов" от Y, X
  for i in range(len(Y)):
    Xi = X[i] if i<len(X) else 0
     if Y[i]<minY:
       y_rest += Y[i]; x_rest += Xi
       continue
     xi2 += (Xi-Y[i])**2/Y[i]; k_svob += 1
    if y_rest==0: k_svob -= 1  # если нет "хвостов"
    else: xi2 += (x_rest-y_rest)**2/y_rest
  return xi2, k_svob
```

Задание

Рассмотрим файлы
rnd_vec_4289.csv, rnd_vec_5212.csv
rnd_vec_6894.csv, rnd_vec_9942.csv.
В каждом из них лежат по 4000 значений целочисленных СВ:

- Биномиальная
- Равномерно распределенная.
- Геометрическая
- Пусассоновская

Определить в каком файле лежит какая величина и каковы её параметры.

Для справки

- Биномиальная(n, p): МО$= np$, $\sigma^2 = np(1-p)$.
- Равномерное(a, b): МО$= (a+b)/2$, $\sigma^2 = (n^2-1)/12$ где $n = b - a + 1$.
- Геометрическая(p): МО$= q/p$, $\sigma^2 = q/p^2$.
- Пусассоновская(λ): МО$= \sigma^2 = \lambda$.

Вероятность того, что СВ принимает значение k:

$$Binomial(n, p) : \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

$$Geom(p) : p \cdot (1-p)^k$$

$$Puass(\lambda) : \frac{\lambda^k}{k!} e^{-\lambda}$$

Таблица равномерного распределения

```
def r_int_uniform(a,b):
  if a<0:
    raise Exception(f'r_int_uniform: a={a}<0')
  # now 0 <=a <b
  k = b-a+1
  res = np.ones(b+1)
  res[:a] = 0
  return res

print('Uniform(0,10):', r_int_uniform(0, 10))
print('Uniform(2,5):', r_int_uniform(2, 5))
print('Uniform(-2,10):', r_int_uniform(-2, 5))
```

Таблица биномиального распределения

$$P(\xi = k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

```
import math
def binom(n,k):
  return math.factorial(n)/
    (math.factorial(k)*math.factorial(n-k))

def r_binom(n,p):
  res = np.zeros(n+1)
  for i in range(n+1):
    res[i] = binom(n,i) * (1-p)**i * p**(n-i)
 return res

Y_binom = r_binom(9, 0.4)
print('Binom:', Y_binom)
```

Таблица геометрического распределения

Напомню, что $\mathrm{MO}(\xi) = M = q/p = (1-p)/p$, то есть $p = 1/(M+1)$.

$$P(\xi = k) = p \cdot (1-p)^k$$

```
def r_geom(p):
    max_ = 40
    res = np.zeros(max_)
    res[0] = p
    for i in range(1, max_):
        res[i] = (1-p)*res[i-1]
    return res

Y_geom = r_geom(0.44)
print('Geom(0.44):', Y_geom)
```

Таблица пуассоновского распределения

$$P(\xi = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

```
def r_puass(lam):
  max_ = 40
  res = np.zeros(max_)
  for i in range(max_):
    res[i] = lam**i/math.factorial(i)*math.exp(-lam)
  return res

Y_puass = r_puass(4.4)
print('Puass(4.4):', Y_puass)
```

Задание

Рассмотрим файлы
rnd_vec_2417.csv, rnd_vec_5476.csv
rnd_vec_7529.csv.
В каждом из них лежат по 4000 значений непрерывных СВ:

- Равномерно распределенная.
- Экспоненциальная
- Нормальная

Определить в каком файле лежит какая величина и каковы её параметры.

Для справки

- Равномерная(a, b): МО$= (a + b)/2$, $\sigma^2 = (b - a)^2/12$.
- Экспоненциальная(λ): МО$= \sigma^2 = \lambda$.
- Нормальная(a, σ).

Равномерное распределение (a,b)

В качестве a естественно взять $min(x_i)$, а в качестве b $max(x_i)$.
Но, например, для СВ равномерно распределённой на отрезке $[0, 1]$ минимум никогда не будет равен 0, а максимум 1.
Поэтому пределы надо немого расширить.
Пусть $\alpha = min(x_i)$, $\beta = max(x_i)$. Тогда положим

$$a = \alpha - \frac{\beta - \alpha}{N - 1}, \quad b = \beta + \frac{\beta - \alpha}{N - 1}.$$

Затем разобьем отрезок $[a, b]$ на k равных частей,
$k = 4, 5, 6, 10$.
Количество значений x_i, попавших в каждый из отрезков должно быть примерно равно $1/k$.

Частоты попаданий

```
# частота попаданий выборки X в разбивку по borders
def freqs(X,borders):
  N = len(borders)+1
  res = np.zeros(N, dtype=int)
  for x1 in X:
    k = N-1
    for i,b in enumerate(borders):
      if b>x1: k=i; break
    res[k] += 1
  return res
# ------> пример вызова:
k, N=10, 2000
X = np.random.uniform(0,k, N)
b = 1+np.arange(k-1); print('b=', b)
print('freqs=', freqs(X, b))
```

Экспоненциальная СВ, разбиение

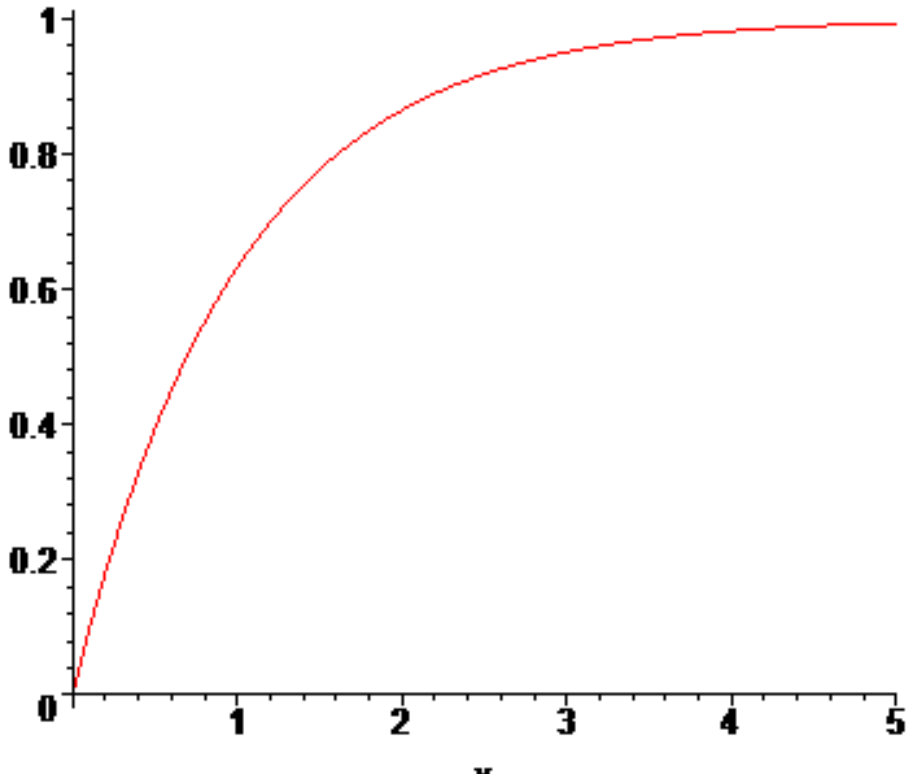
СВ принимает значения от 0 до ∞.

Плотность распределения

$$f(x) = \lambda e^{-\lambda x} \text{ при } x > 0.$$

Функция распределения:

$$F(x) = 1 - e^{-\lambda x} \text{ при } x > 0.$$



Экспоненциальная СВ, разбиение

Найдем такое разбиении на k участков, что вероятность попадания в каждый из них одинакова.
При $k = 2$: $0,\ 0.6931/\lambda,\ \infty$.
Далее деление на λ указываться не будет.
При $k = 3$: $0,\ 0.4055,\ 1.0986,\ \infty$.
При $k = 4$: $0,\ 0.2877,\ 0.6931,\ 1.3863,\ \infty$.
При $k = 5$: $0,\ 0.2231,\ 0.5108,\ 0.9163,\ 1.6094,\ \infty$.
При $k = 6$: $0,\ 0.1823,\ 0.4055,\ 0.6931,\ 1.0986,\ 1.7918,\ \infty$.
В общем виде:

$$r_i = \frac{ln(k) - ln(i)}{\lambda}$$

Например, при $\lambda = 0.1$ и $k = 4$ вероятность попадания СВ в интервалы $[0 - 2.8], [2.8 - 6.9], [6.9 - 13.9], [13.9 - \infty]$ одинакова и равна 0.25.

Частоты попаданий

```
# частота попаданий выборки X в разбивку по borders
def freqs(X,borders):
  N = len(borders)+1
  res = np.zeros(N, dtype=int)
  for x1 in X:
    k = N-1
    for i,b in enumerate(borders):
      if b>x1: k=i; break
    res[k] += 1
  return res
# ------> пример вызова:
k, N=10, 2000
X = np.random.uniform(0,k, N)
b = 1+np.arange(k-1); print('b=', b)
print('freqs=', freqs(X, b))
```

Квантили стандартной СВ

```
quantils = (None, None,                          # 0, 1
 ( 0.0) ,                                        # 2
 (-0.430727,  0.430727) ,                        # 3
 (-0.674490,  0.       ,  0.67449),              # 4
 (-0.841621, -0.253347,  0.253347, 0.841621),    # 5
 (-0.967422, -0.430727,  0.,
   0.430727,  0.967422),                         # 6
 (-1.067571, -0.565949, -0.180012,
   0.180012,  0.565949,  1.067571),              # 7
 (-1.150349, -0.67449 , -0.318639,  0.,
   0.318639,  0.67449 ,  1.150349),              # 8
 (-1.220640, -0.76471 , -0.430727, -0.13971,
   0.13971 ,  0.430727,  0.76471 ,  1.22064),    # 9
 )
```