Лабораторная работа №3

Исследование распределений и моментов связанных с преобразованием случайных величин

Логинов Сергей НФИмд-01-22

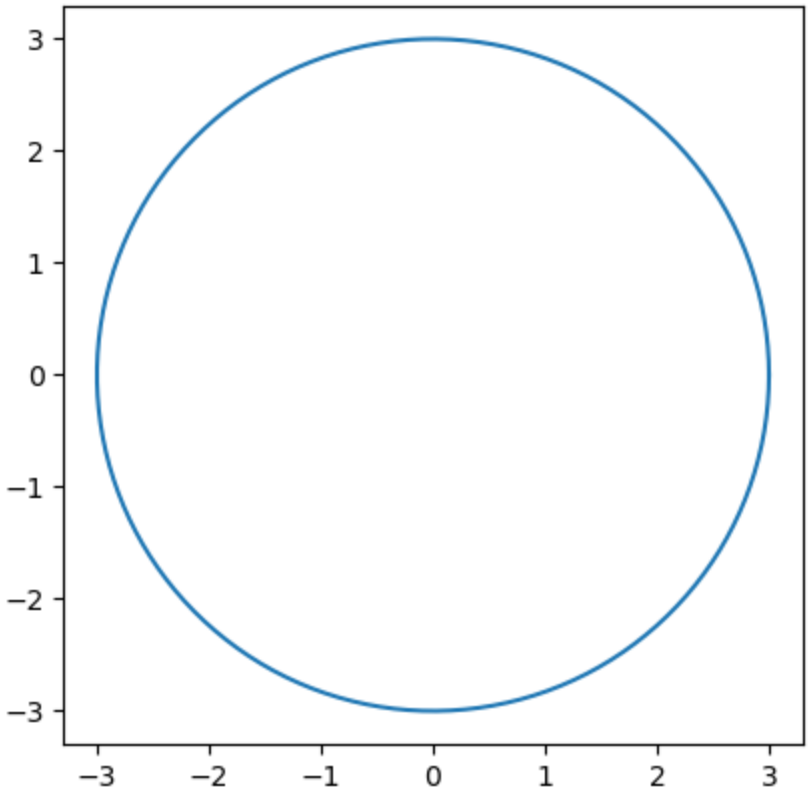
Цель работы:

Применение распределений для преобразования случайных величин

In [262…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import math
```

Для начала просто нарисуем круг

In [263…
```
theta = np.linspace(0, 2 * np.pi, 150)

radius = 3

a = radius * np.cos(theta)
b = radius * np.sin(theta)

figure, axes = plt.subplots(1)

axes.plot(a, b)
axes.set_aspect(1)
plt.show()
```



Способ 1

Генерируем выборку по случайному радиусу и углу.

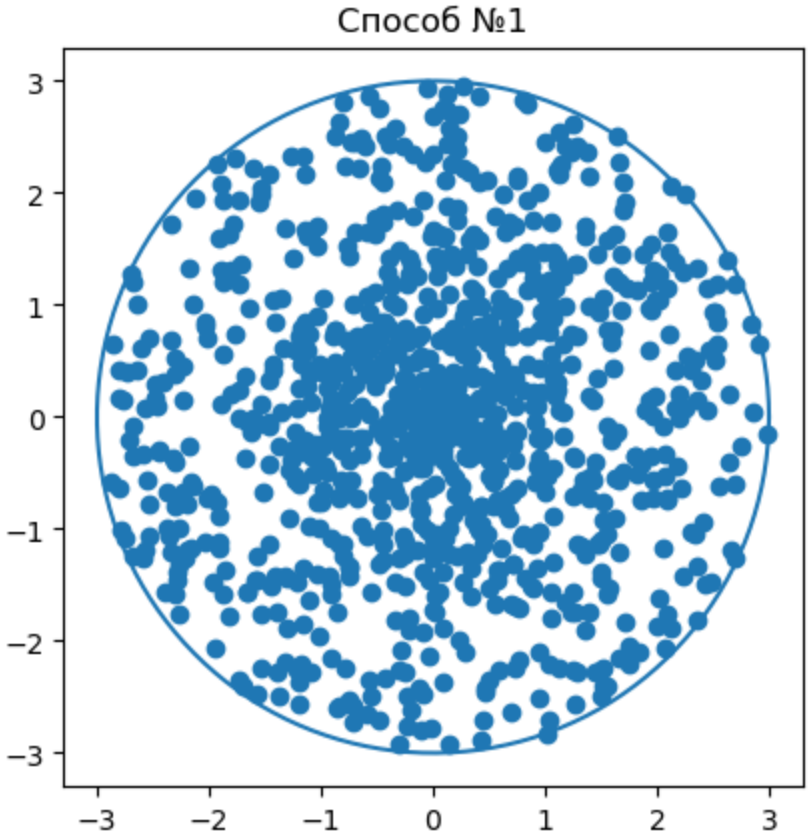
In [264…
```
r = stats.uniform.rvs(0, 3, size=1000)
angle = stats.uniform.rvs(0, 360, size=1000)
```

Переводим координаты в полярные

In [265…
```
x = [r[i] * math.cos(angle[i]) for i in range(1000)]
y = [r[i] * math.sin(angle[i]) for i in range(1000)]
```

Рисуем круг и точки на графике

In [266…
```
theta = np.linspace(0 , 2 * np.pi, 150)

radius = 3

a = radius * np.cos(theta)
b = radius * np.sin(theta)

figure, axes = plt.subplots( 1 )

axes.plot(a, b)
axes.set_aspect(1)
plt.scatter(x, y)
plt.title('Способ №1')
plt.show()
```



Все точки лежат строго в пределах окружности

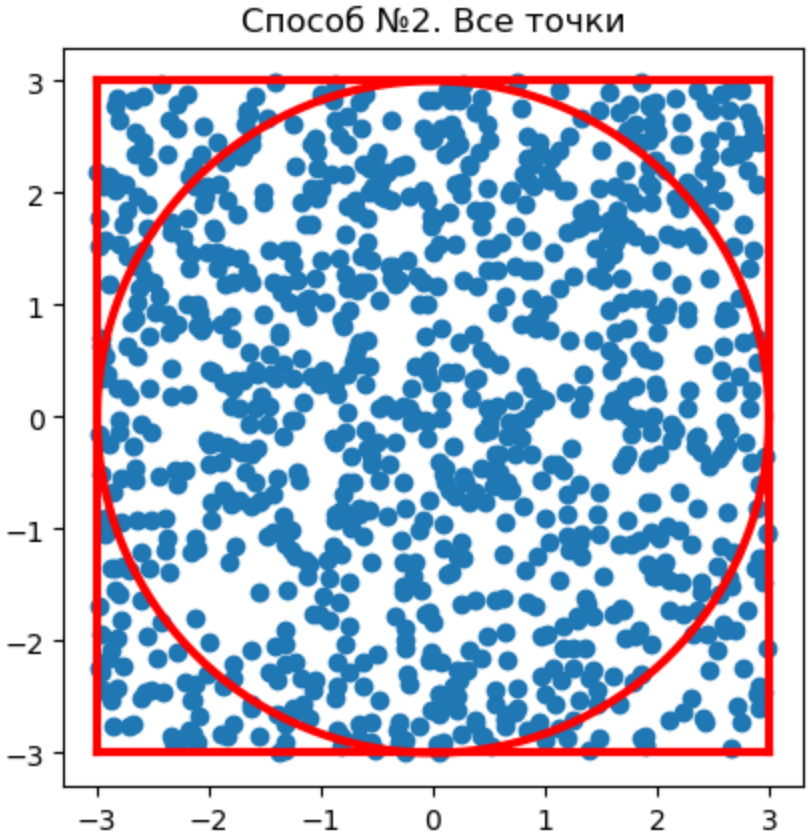
Способ 2

Генерируем значения в пределах квадрата

In [267…
```
x1 = stats.uniform.rvs(-3, 6, 1000)
y1 = stats.uniform.rvs(-3, 6, 1000)
```

Переводим координаты в полярные и рисуем график

In [268…
```
theta = np.linspace(0, 2 * np.pi , 150)

radius = 3

a = radius * np.cos(theta)
b = radius * np.sin(theta)

figure, axes = plt.subplots(1)

axes.plot( a, b, linewidth=3, c='r')
plt.plot(-3, -3, -3, 3)
axes.set_aspect( 1 )
plt.plot([-3, -3], [-3, 3],
         [-3, 3], [3, 3],
         [3, 3], [3, -3],
         [-3, 3], [-3, -3],
         c='r',
         linewidth=3)
plt.scatter(x1, y1)
plt.title( 'Способ №2. Все точки' )
plt.show()
```



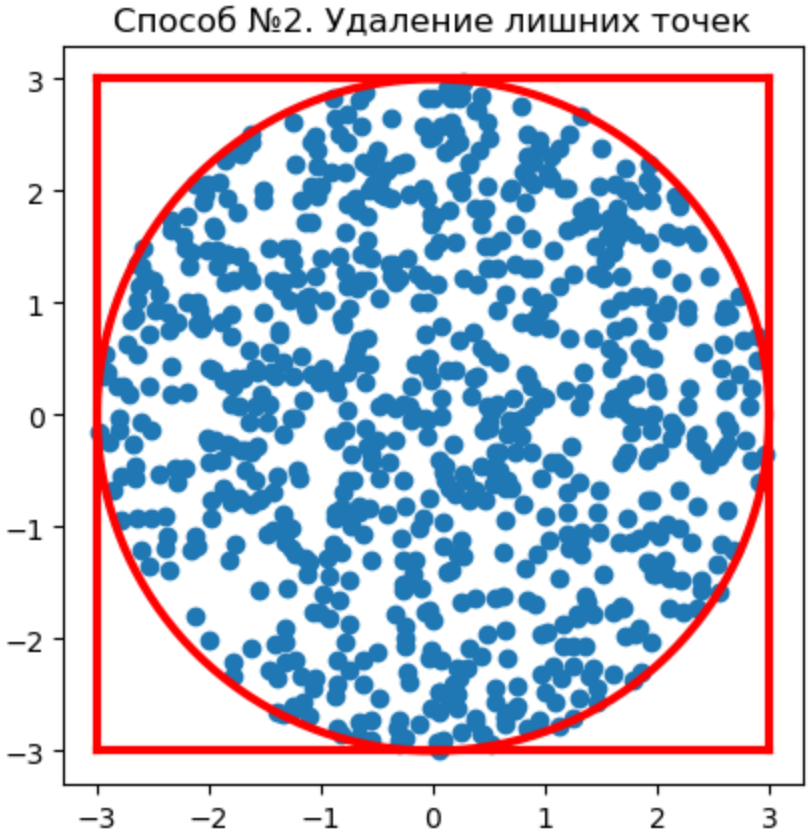
Функция для удаления лишних точек

In [269…
```
for_del = []
for i in range(len(x)):
    if (x1[i] ** 2 + y1[i] ** 2) > 9:
        for_del.append(i)
```

In [270…
```
x_clear = np.delete(x1, for_del)
y_clear = np.delete(y1, for_del)
```

Отрисовка после удаления

In [271…
```
theta = np.linspace(0, 2 * np.pi, 150)

radius = 3

a = radius * np.cos(theta)
b = radius * np.sin(theta)

figure, axes = plt.subplots(1)

axes.plot(a, b,
          linewidth=3,
          c='r')
plt.plot(-3, -3, -3, 3)
axes.set_aspect( 1 )
plt.plot([-3, -3], [-3, 3],
         [-3, 3], [3, 3],
         [3, 3], [3, -3],
         [-3, 3], [-3, -3],
         color='r',
         linewidth=3)
plt.scatter(x_clear, y_clear)
plt.title( 'Способ №2. Удаление лишних точек' )
plt.show()
```



In [279…
```
print('Способ 1: \nВыборочное среднее по x =', np.mean(x),  'дисперсия по x =', np.var(x
print('\nВыборочное среднее по y =', np.mean(y),  'дисперсия по y =', np.var(y))
print('\nСпособ 2: \nВыборочное среднее по x =', np.mean(x_clear),  'дисперсия по x =',
print('\nВыборочное среднее по y =', np.mean(y_clear),  'дисперсия по y =', np.var(y_cle
```

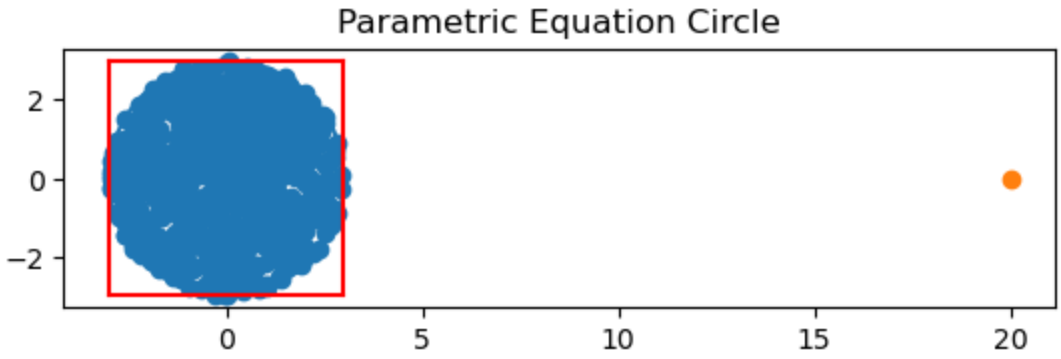
```
Способ 1:
Выборочное среднее по x = 0.023374425440254323 дисперсия по x = 1.5078110567741312

Выборочное среднее по y = -0.004921604400315207 дисперсия по y = 1.5695688667902168

Способ 2:
Выборочное среднее по x = -0.020827881990829352 дисперсия по x = 2.0965506217210828

Выборочное среднее по y = 0.059448301876579714 дисперсия по y = 2.383734151928817
```

Координаты точки, до которой ищем расстояние

In [232…
```
dot_x = 20
dot_y = 0
```

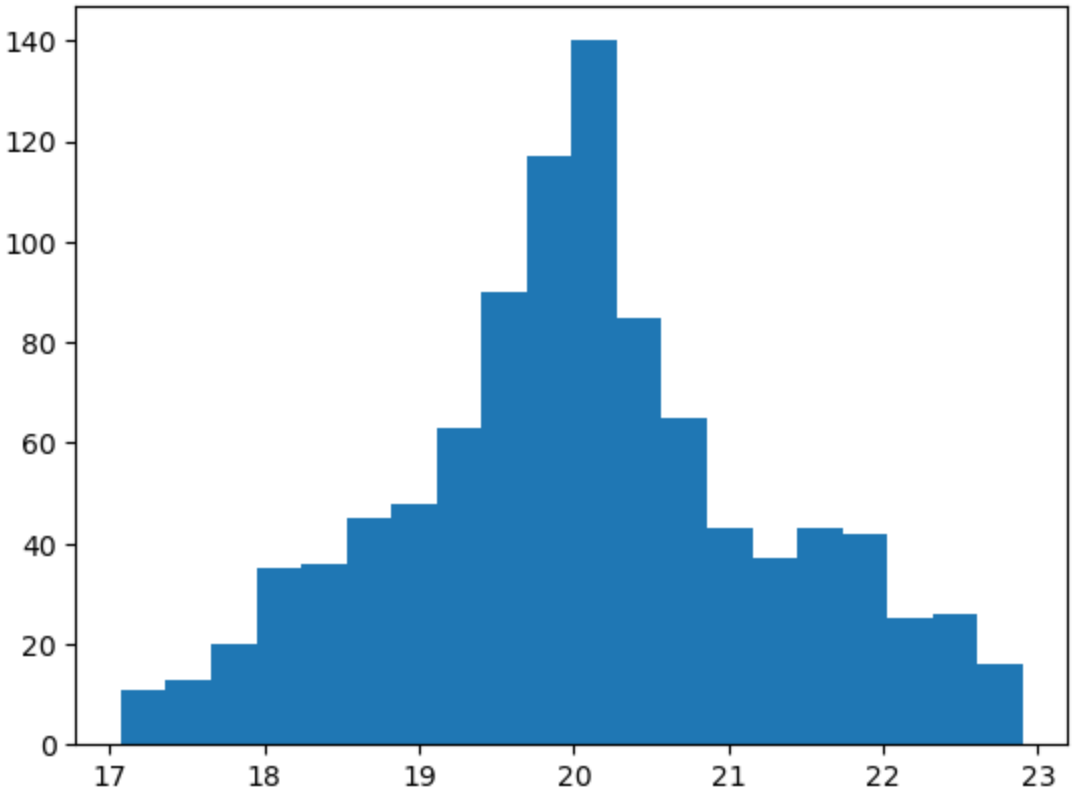
In [233…
```
theta = np.linspace(0 , 2 * np.pi, 150)

radius = 3

a = radius * np.cos(theta)
b = radius * np.sin(theta)
plt.style.use('default')
figure, axes = plt.subplots(1)
axes.plot(a, b)
plt.plot(-3, -3, -3, 3)
axes.set_aspect( 1 )
plt.plot([-3, -3], [-3, 3],
         [-3, 3], [3, 3],
         [3, 3], [3, -3],
         [-3, 3], [-3, -3], color='r')
plt.scatter(x_clear, y_clear)
```

```
plt.scatter(dot_x, dot_y)
plt.show()
```



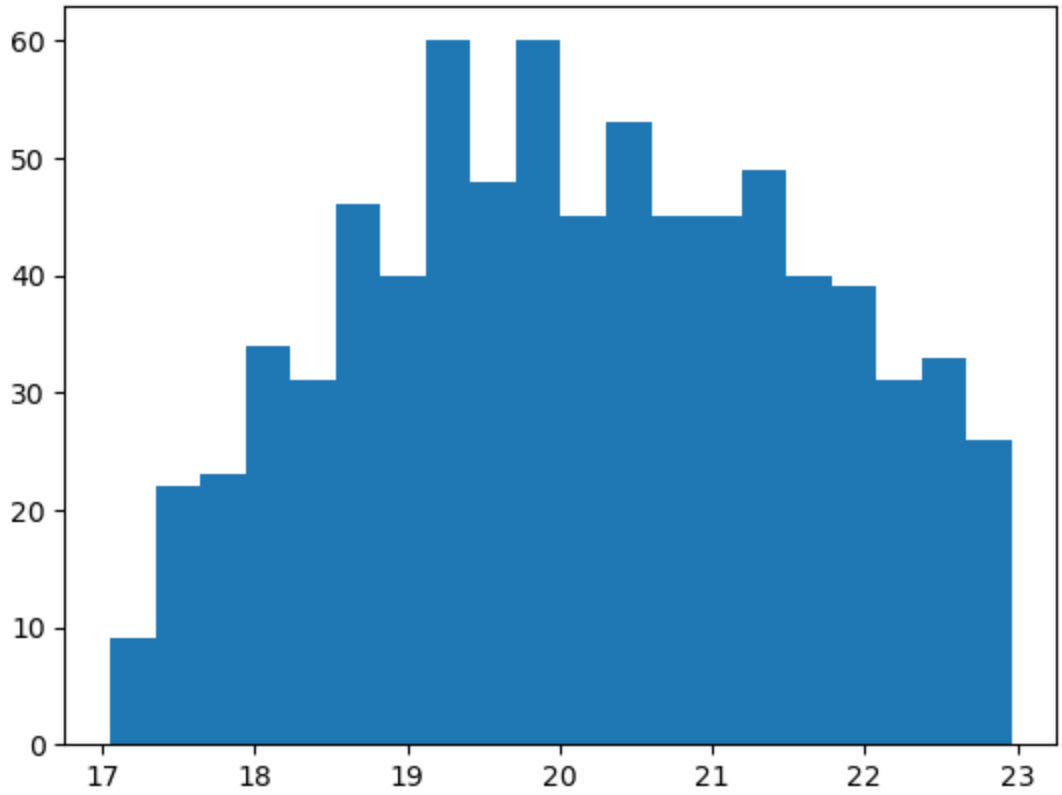
Найдем расстояние от точки до точек из первого способа. Построим гистограмму распределения

In [234…
```
distances1 = [math.sqrt((x[i] - dot_x)**2 + (y[i] - dot_y)**2) for i in range(len(x))]
plt.hist(distances1, bins=20)
plt.show()
```



То же самое для второго способа генерирования точек

In [235…
```
distances2 = [math.sqrt((x_clear[i] - dot_x)**2 + (y_clear[i] - dot_y)**2) for i in rang
plt.hist(distances2, bins=20)
plt.show()
```



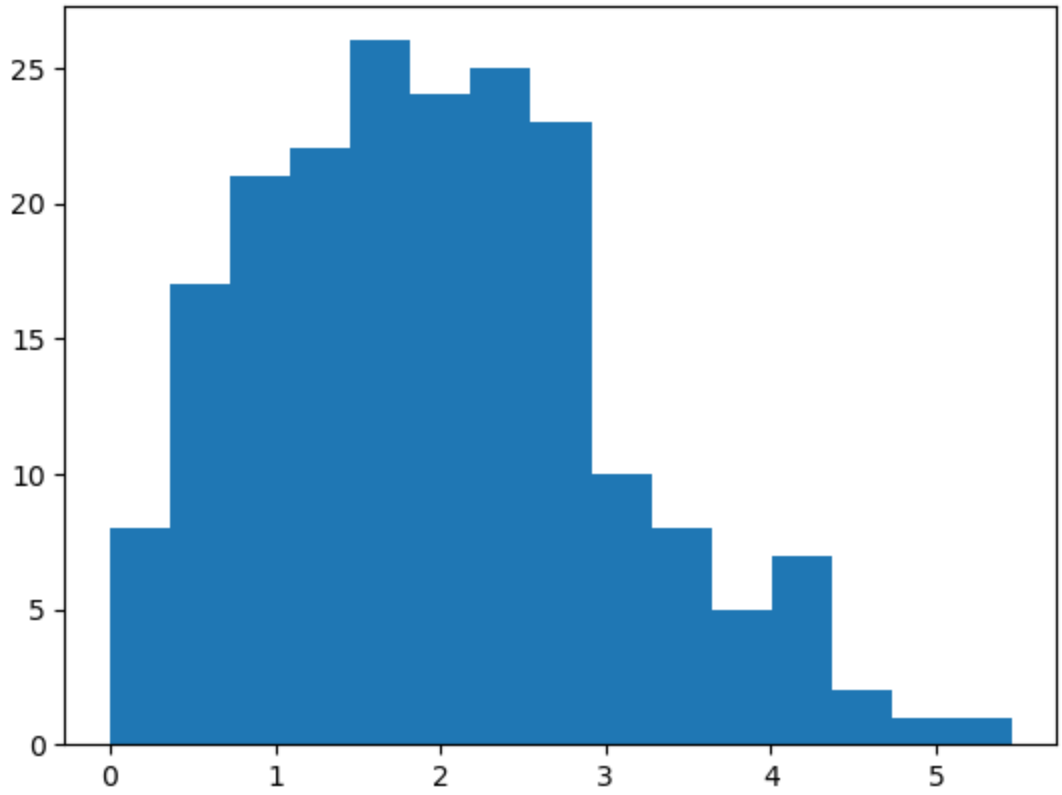
Функция для поиска расстояния между случайными точками

In [248…
```
def find_random_dist(x, y, size):
    inner_dist = []
    for i in range(size):
        num1 = round(np.random.uniform(0, size - 1, 1)[0])
        num2 = round(np.random.uniform(0, size - 1, 1)[0])
        inner_dist.append(math.sqrt((x[num1] - x[num2])**2 + (y[num1] - y[num2])**2))
    return inner_dist
```

In [249…
```
innder_dist1 = find_random_dist(x, y, 1000)
```

Строим гистограмму плотности для первого способа

In [250…
```
plt.hist(inner_dist1, bins=15)
plt.show()
```



Повторяем для второго способа

In [255…
```
innder_dist2 = find_random_dist(x_clear, y_clear, len(x_clear))
```

In [256…
```
plt.hist(inner_dist2, bins=10)
plt.show()
```