МАТЕМАТИЧЕСКИЕ МЕТОДЫ ОПТИЧЕСКОЙ БИОСПЕКТРОСКОПИИ *IN VIVO*

Практикум №2 (29 марта 2023)

Проверка гипотез

Формируем два массива случайных величин с разными средними в Python numpy

Для создания последовательностей чисел, в NumPy имеется функция arange(), аналогичная встроенной в Python range(), только вместо списков она возвращает массивы, и принимает не только целые значения:

```
>>> np.arange(10, 30, 5)
array([10, 15, 20, 25])
>>> np.arange(0, 1, 0.1)
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9])
```

Заполним массив величинами из нормального распределения:

```
>>>c = 15
>>>w = 5
>>>num = 20
>>>inp_norm = np.random.normal(c,w,num)
>>>inp_norm
array([ 9.1742703 , 13.03650447,  5.93089495, 13.04962049, 19.29911111,
       12.74087993, 10.40564479, 14.35761725, -2.36282785, 14.28766252,
       19.80453278, 10.20603218, 11.44792894, 19.61000287,  7.87949471,
        9.157177  , 23.93331462, 14.00932658,  9.38303872,  9.84051895])
```

https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html

Проверяем на нормальность распределения наши данные

Есть четыре распространенных способа проверить это предположение в Python:

1. (Визуальный метод) Создайте гистограмму.

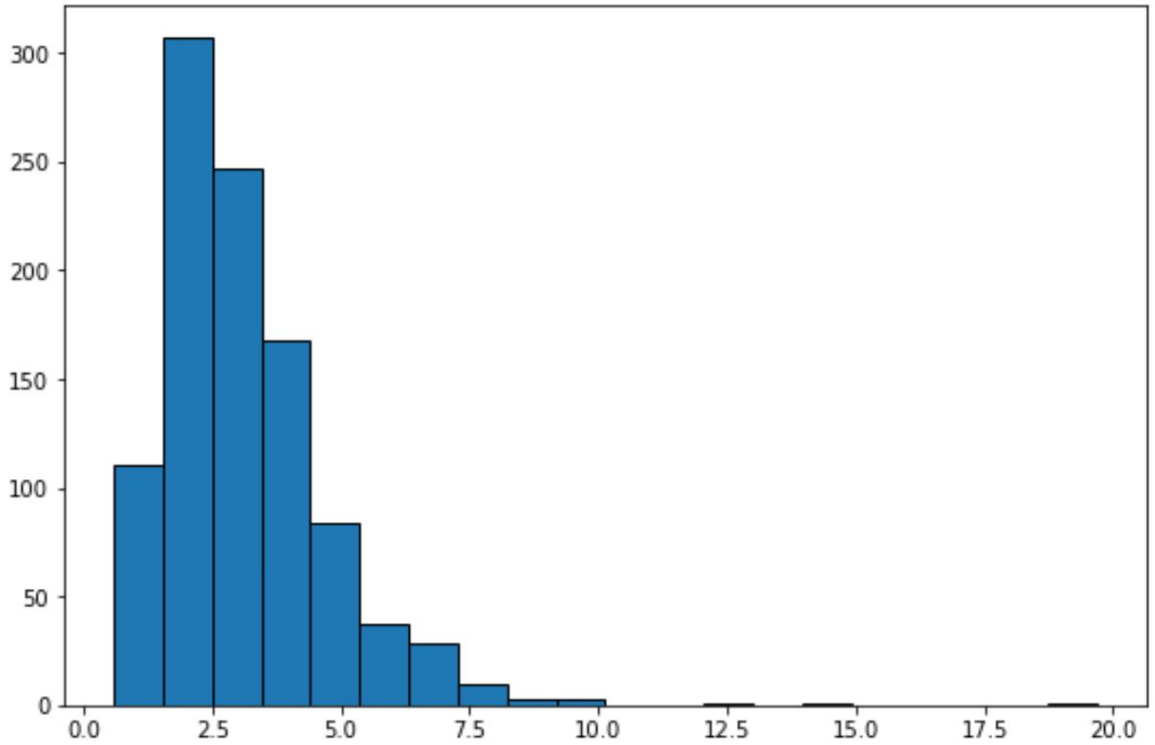
Если гистограмма имеет форму колокола, то считается, что данные распределены нормально.
2. (Визуальный метод) Создайте график QQ.

Если точки на графике примерно совпадают с прямой диагональной линией, предполагается, что данные распределены нормально.
3. (Формальный статистический тест) Выполните тест Шапиро-Уилка.

Если p-значение теста больше, чем α = 0,05, то предполагается, что данные распределены нормально.
4. (Формальный статистический тест) Выполните тест Колмогорова-Смирнова.

Если p-значение теста больше, чем α = 0,05, то предполагается, что данные распределены нормально.
https://www.codecamp.ru/blog/normality-test-python/

Способ 1: создание гистограммы

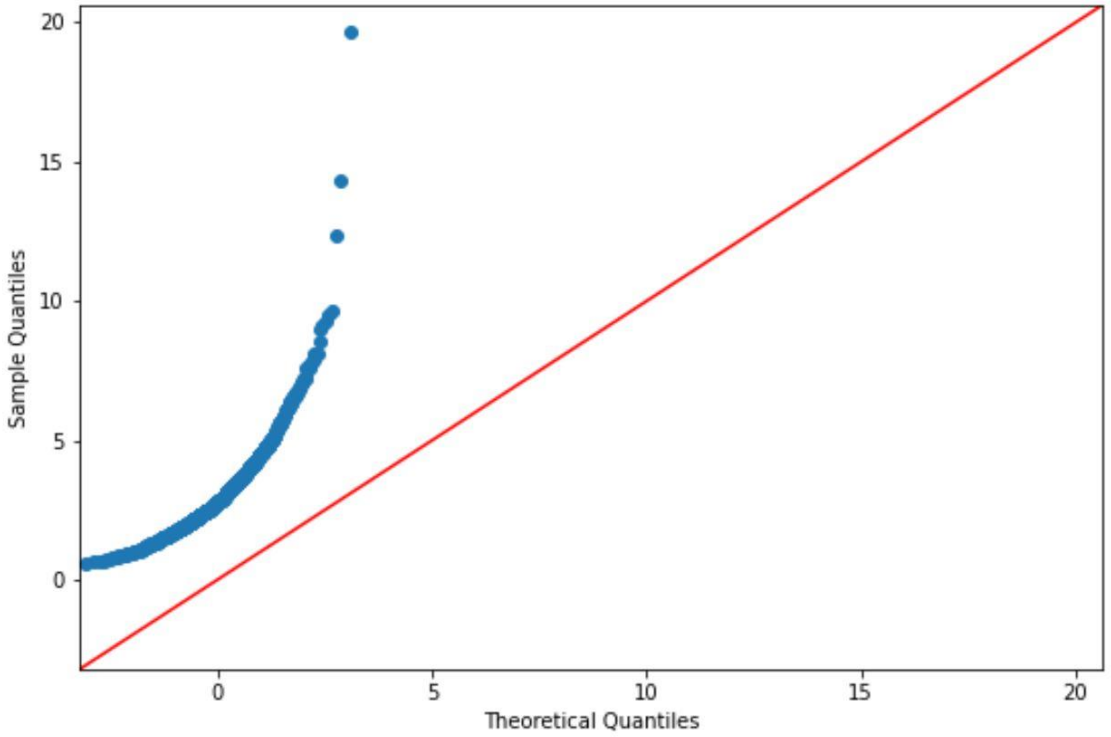



```
import math
import numpy as np
from scipy. stats import lognorm
import matplotlib.pyplot as plt

#make this example reproducible
np.random.seed (1)

#generate dataset that contains 1000 log-normal distributed values
lognorm_dataset = lognorm. rvs (s=.5, scale=math. exp (1), size=1000)

#create histogram to visualize values in dataset
plt.hist (lognorm_dataset, edgecolor='black', bins=20)
```

https://www.codecamp.ru/blog/normality-test-python/

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.lognorm.html

Способ 2: создать график QQ

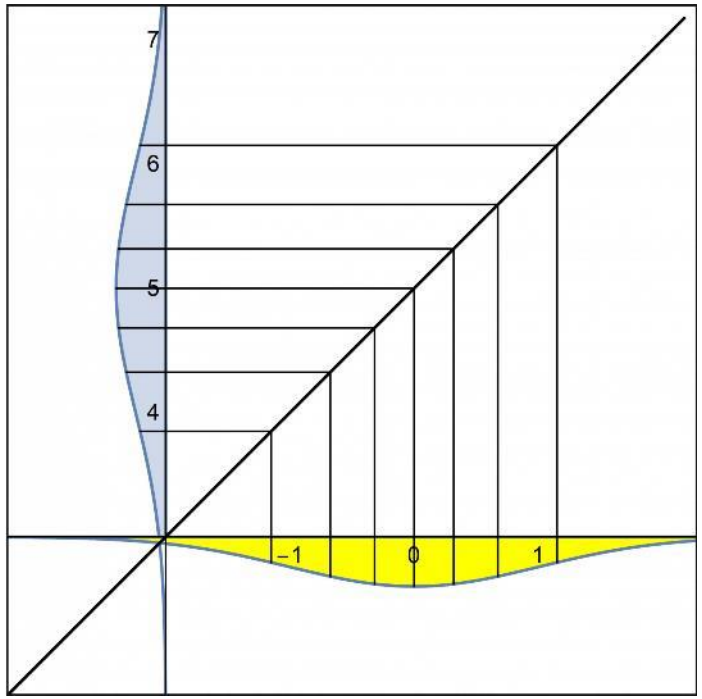


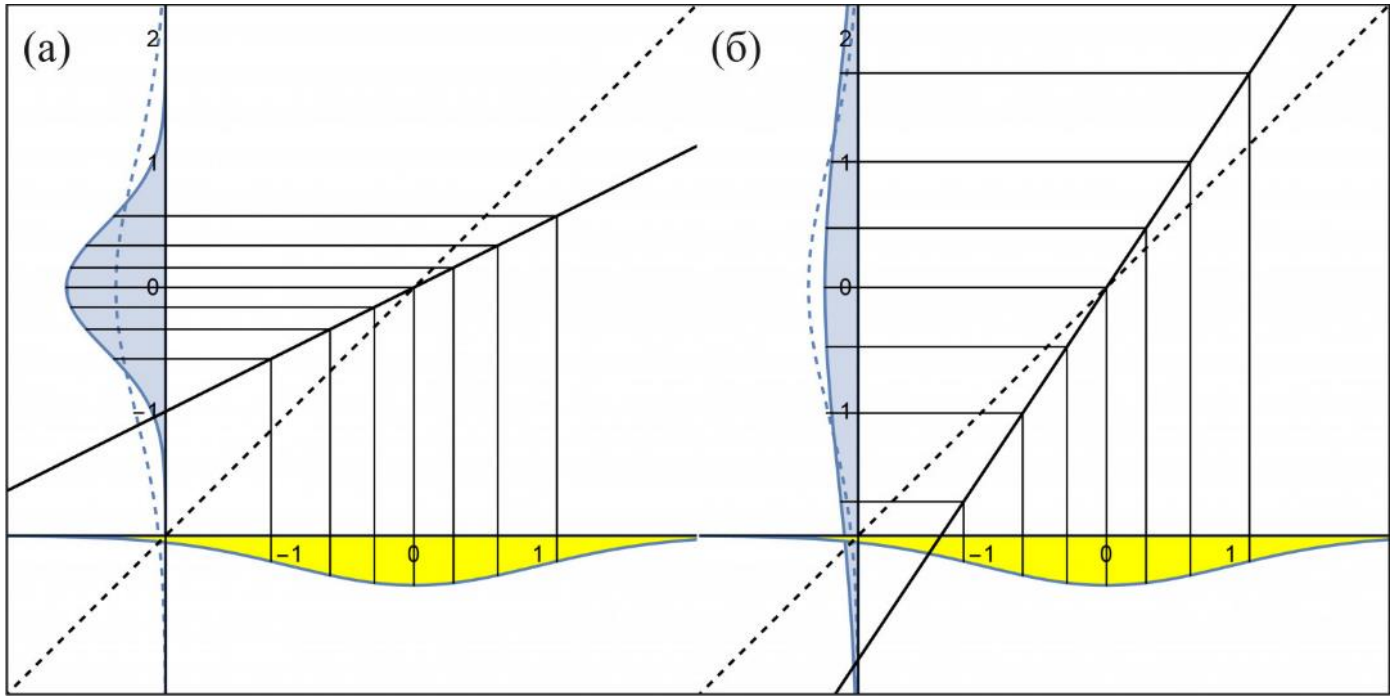

```
import math
import numpy as np
from scipy. stats import lognorm
import statsmodels.api as sm
import matplotlib.pyplot as plt

#make this example reproducible
np.random.seed (1)

#generate dataset that contains 1000 log-normal distributed values
lognorm_dataset = lognorm. rvs (s=.5, scale=math. exp (1), size=1000)

#create Q-Q plot with 45-degree line added to plot
fig = sm. qqplot (lognorm_dataset, line='45')

plt.show()
```

https://www.codecamp.ru/blog/normality-test-python/

Q-Q Plot – сопоставление квантилей

Квантиль непрерывного распределения — это одна из точек, делящих функцию плотности распределения на участки, вероятность попадания в которые одинакова, то есть на участки одинаковой площади.
Квантиль-функция — это функция, которая по значению вероятности P возвращает такое число (квантиль) q, что вероятность того, что случайная величина примет значение меньше q равняется P.
Главный квантильный график - теоретических квантилей стандартного нормального распределения от теоретических квантилей стандартного нормального распределения. В случае одинаковых распределений Q-Q Plot представляет собой прямую линию y = x, причем масштаб нормальных распределений не имеет значения, главное, чтобы у них совпадали средние значения и стандартное отклонение.



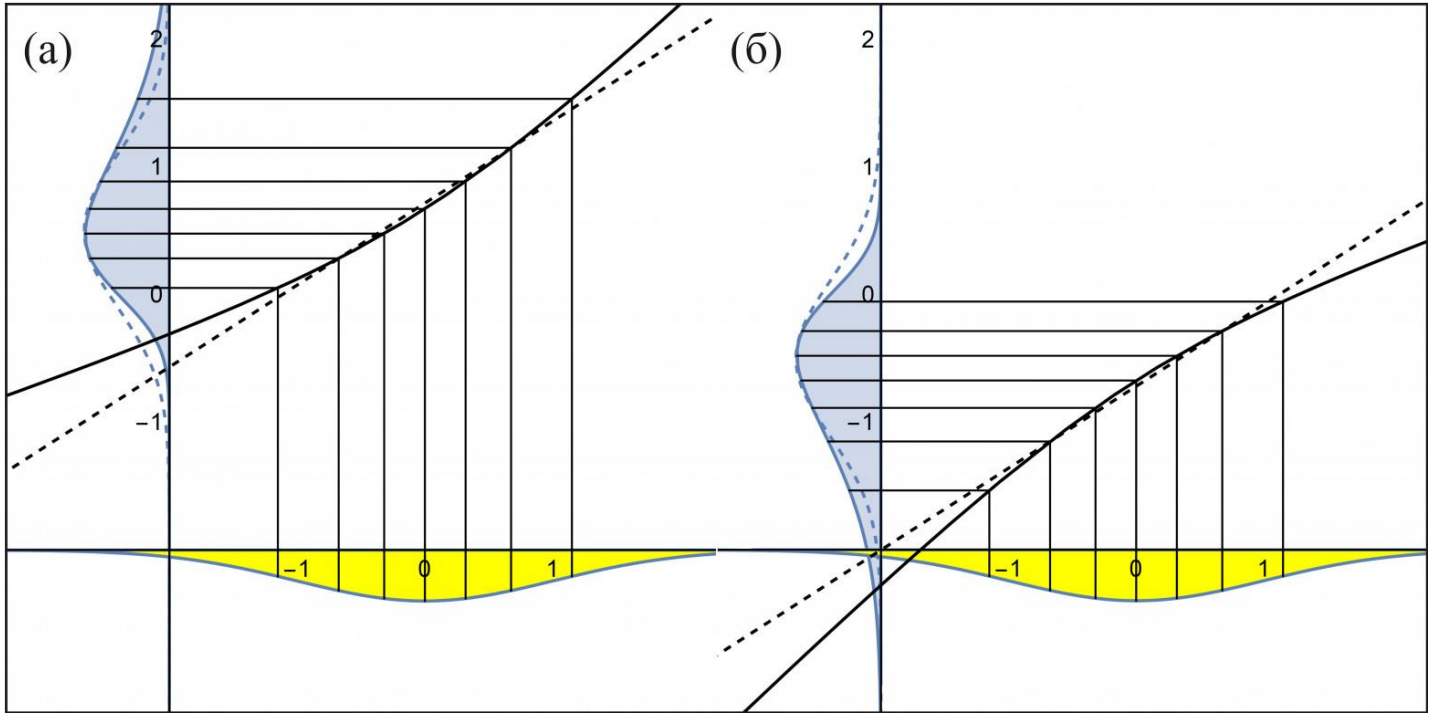


https://habr.com/ru/post/578754/

Q-Q Plot – асимметричные распределения

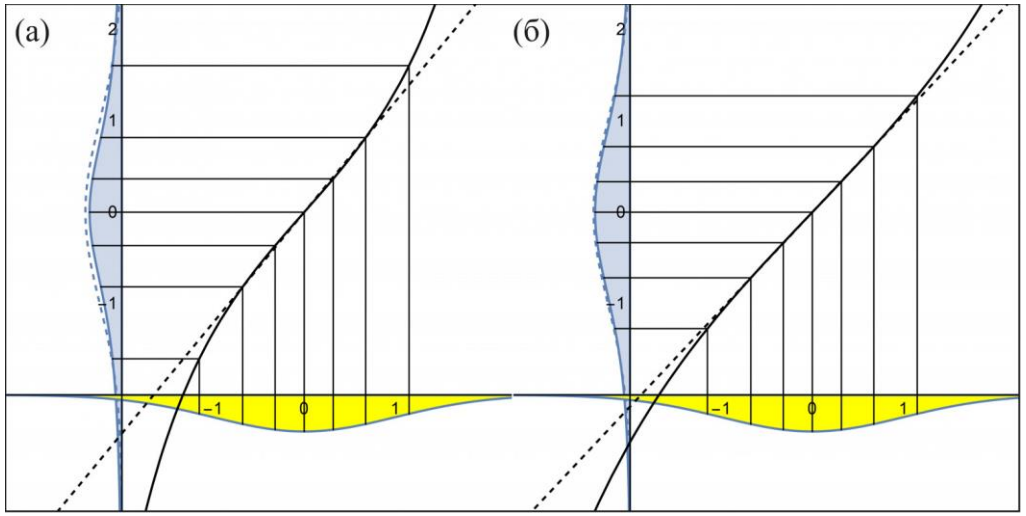
При построении Q-Q Plot многие программные пакеты подбирают и изображают некоторую прямую, которая называется линией главного тренда (англ. Reference Line).
Intercept и slope этой контрольной прямой имеют смысл среднего и стандартного отклонения нормального распределения, "наилучшим образом" подходящего к нашим данным.

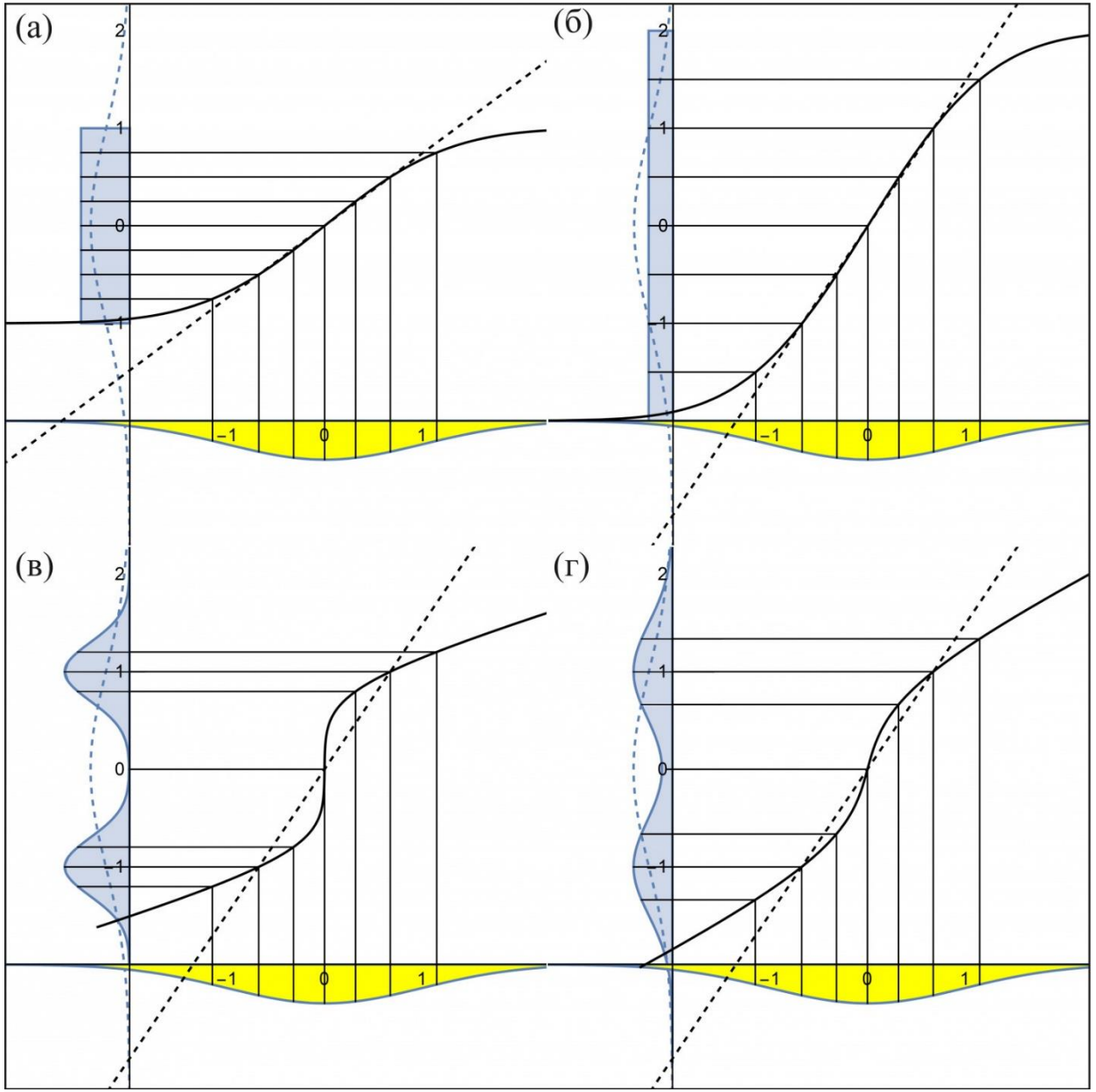
- Если оба конца квантильного графика находятся выше прямой главного тренда, то скорее всего это распределение скошено вправо.
- Если оба конца квантильного графика находится ниже прямой главного тренда, то скорее всего это распределение скошено влево.



https://habr.com/ru/post/578754/

Изогнутые Q-Q Plots: равномерное, бимодальное и t-распределения





https://habr.com/ru/post/578754/

Способ 3: выполнить тест Шапиро-Уилка

```
import math
import numpy as np
from scipy.stats import shapiro
from scipy. stats import lognorm

#make this example reproducible
np.random.seed (1)

#generate dataset that contains 1000 log-normal distributed values
lognorm_dataset = lognorm. rvs (s=.5, scale=math. exp (1), size=1000)

#perform Shapiro-Wilk test for normality
shapiro(lognorm_dataset)

ShapiroResult(statistic=0.8573324680328369, pvalue=3.880663073872444e-29)
```

- Нулевая гипотеза H0 теста Шапиро–Уилка заключается в том, что случайная величина, выборка x которой известна, распределена по нормальному закону.
- Из вывода мы видим, что тестовая статистика равна 0,857 , а соответствующее значение p равно 3,88e-29 (чрезвычайно близко к нулю).
- Поскольку p-значение меньше 0,05, мы отвергаем нулевую гипотезу теста Шапиро-Уилка.
- Это означает, что у нас есть достаточно доказательств, чтобы сказать, что данные выборки не получены из нормального распределения.

https://www.codecamp.ru/blog/normality-test-python/

Способ 4: тест Колмогорова-Смирнова

Метод 4: выполнить тест Колмогорова-Смирнова
В следующем коде показано, как выполнить тест Колмогорова-Смирнова для набора данных, который соответствует логарифмически нормальному распределению:

```
import math
import numpy as np
from scipy.stats import kstest
from scipy. stats import lognorm

#make this example reproducible
np.random.seed (1)

#generate dataset that contains 1000 log-normal distributed values
lognorm_dataset = lognorm. rvs (s=.5, scale=math. exp (1), size=1000)

#perform Kolmogorov-Smirnov test for normality
kstest(lognorm_dataset, 'norm')

KstestResult(statistic=0.84125708308077, pvalue=0.0)
```

Из вывода мы видим, что статистика теста равна 0,841 , а соответствующее значение p равно 0,0 .

Поскольку p-значение меньше 0,05, мы отвергаем нулевую гипотезу теста Колмогорова-Смирнова.

Это означает, что у нас есть достаточно доказательств, чтобы сказать, что данные выборки не получены из нормального распределения.

https://www.codecamp.ru/blog/normality-test-python/

Как обращаться с ненормальными данными

Если данный набор данных не имеет нормального распределения, мы часто можем выполнить одно из следующих преобразований, чтобы сделать его более нормально распределенным:

1. Log-преобразование: преобразование значений из x в log(x) .

2. Преобразование квадратного корня: преобразование значений из x в $\sqrt{x}$.

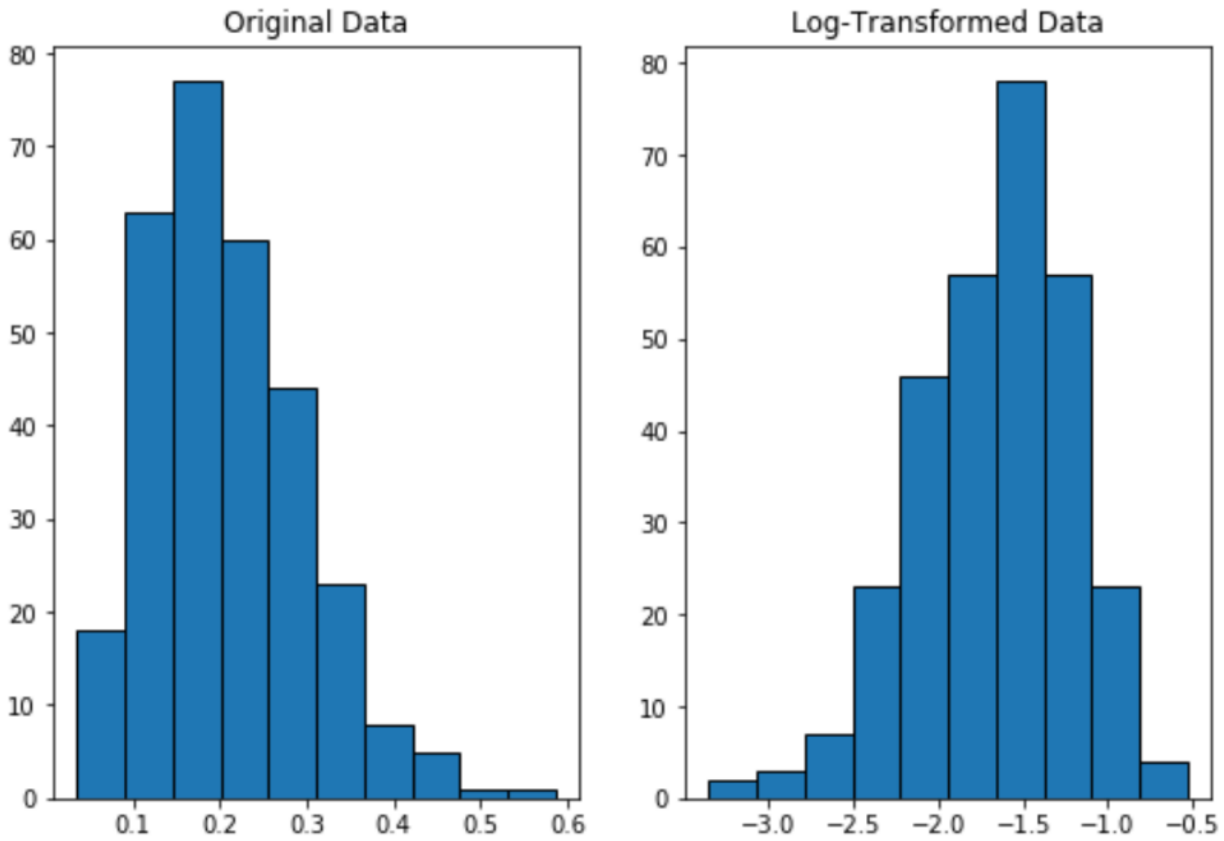
3. Преобразование кубического корня: преобразование значений от x до x 1/3 .

Выполняя эти преобразования, набор данных обычно становится более нормально распределенным.

https://www.codecamp.ru/blog/normality-test-python/
https://www.codecamp.ru/blog/transform-data-in-python/

Логарифмическое преобразование в Python

```
import numpy as np
import matplotlib.pyplot as plt

#make this example reproducible
np.random.seed (0)

#create beta distributed random variable with 200 values
data = np.random.beta (a= 4 , b= 15 , size= 300 )

#create log-transformed data
data_log = np.log (data)

#define grid of plots
fig, axs = plt.subplots(nrows= 1 , ncols= 2 )

#create histograms
axs[0]. hist (data, edgecolor='black')
axs[1]. hist (data_log, edgecolor='black')

#add title to each histogram
axs[0].set_title('Original Data')
axs[1].set_title('Log-Transformed Data')
```



https://www.codecamp.ru/blog/transform-data-in-python/

Однофакторный дисперсионный анализ в Python

```
from scipy.stats import f_oneway

#perform one-way ANOVA
f_oneway(group1, group2, group3)

(statistic=2.3575, pvalue=0.1138)
```

Однофакторный дисперсионный анализ использует следующие нулевую и альтернативную гипотезы :

H 0 (нулевая гипотеза): μ 1 = μ 2 = μ 3 = … = μ k (все средние значения совокупности равны)
H 1 (нулевая гипотеза): по крайней мере одно среднее значение популяции отличаетсяот остальных

https://www.codecamp.ru/blog/one-way-anova-python/

Независимый двухвыборочный t-тест в Pandas

Например, предположим, что профессор хочет знать, приводят ли два разных метода обучения к разным средним баллам на экзаменах.

Чтобы проверить это, он набирает 10 студентов для использования метода А и 10 студентов для использования метода Б.

В следующем коде показано, как ввести баллы каждого учащегося в кадр данных pandas, а затем использовать функцию ttest_ind() из библиотеки SciPy для выполнения независимого двухвыборочного t-теста:

```
import pandas as pd
from scipy. stats import ttest_ind
#create pandas DataFrame
df = pd.DataFrame({'method': ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B'],
 'score': [71, 72, 72, 75, 78, 81, 82, 83, 89, 91, 80, 81, 81,
 84, 88, 88, 89, 90, 90, 91]})
#view first five rows of DataFrame
df.head ()
method score
0 A 71
1 A 72
2 A 72
3 A 75
4 A 78
#define samples
group1 = df[df['method']=='A']
group2 = df[df['method']=='B']
#perform independent two sample t-test
ttest_ind(group1['score'], group2['score'])
Ttest_indResult(statistic=-2.6034304605397938, pvalue=0.017969284594810425)
```

https://www.codecamp.ru/blog/one-way-anova-python/

t-критерий парных выборок в Pandas

Например, предположим, что профессор хочет знать, приводят ли два разных метода обучения к разным средним баллам на экзаменах.

Чтобы проверить это, он набирает 10 студентов для использования метода А, а затем проходит тест. Затем он позволяет тем же 10 учащимся, которые использовали метод Б, подготовиться и сдать еще один тест аналогичной сложности.

Поскольку все учащиеся присутствуют в обеих выборках, в этом сценарии мы можем выполнить t-критерий парных выборок.

```
import pandas as pd
from scipy. stats import ttest_rel
#create pandas DataFrame
df = pd.DataFrame({'method': ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B'],
 'score': [71, 72, 72, 75, 78, 81, 82, 83, 89, 91, 80, 81, 81,
 84, 88, 88, 89, 90, 90, 91]})
#view first five rows of DataFrame
df.head ()
 method score
0 A 71
1 A 72
2 A 72
3 A 75
4 A 78
#define samples
group1 = df[df['method']=='A']
group2 = df[df['method']=='B']
#perform independent two sample t-test
ttest_rel(group1['score'], group2['score'])
Ttest_relResult(statistic=-6.162045351967805, pvalue=0.0001662872100210469)
```

Ранговый тест Уилкоксона в Python

Исследователи хотят знать, приводит ли новая обработка топлива к изменению среднего расхода топлива на галлон определенного автомобиля. Чтобы проверить это, они измеряют количество миль на галлон 12 автомобилей с обработкой топлива и без нее.

Используйте следующие шаги, чтобы выполнить знаковый ранговый тест Уилкоксона в Python, чтобы определить, есть ли разница в среднем показателе миль на галлон между двумя группами.

Шаг 1: Создайте данные.

Во-первых, мы создадим два массива для хранения значений миль на галлон для каждой группы автомобилей

Шаг 2: Проведите знаковый ранговый тест Уилкоксона.

Далее мы воспользуемся функцией wilcoxon() из библиотеки scipy.stats для проведения теста Wilcoxon Signed-Rank, который использует следующий синтаксис:

Уилкоксон (x, y, альтернатива = 'двусторонний')

x: массив выборочных наблюдений из группы 1

y: массив выборочных наблюдений из группы 2

альтернатива: определяет альтернативную гипотезу. По умолчанию используется «двусторонний», но другие варианты включают «меньше» и «больше».

Статистика теста равна 10,5 , а соответствующее двустороннее значение p равно 0,044 .

Шаг 3: Интерпретируйте результаты.

В этом примере критерий знакового ранга Уилкоксона использует следующие нулевую и альтернативную гипотезы:

H 0 : миль на галлон равны между двумя группами

H A : Расход топлива в милях на галлон для двух групп неодинаков .

Поскольку p-значение (0,044) меньше 0,05, мы отвергаем нулевую гипотезу. У нас есть достаточно доказательств, чтобы сказать, что истинное среднее значение миль на галлон не равно между двумя группами.

```
import scipy.stats as stats

group1 = [20, 23, 21, 25, 18, 17, 18, 24, 20, 24, 23, 19]
group2 = [24, 25, 21, 22, 23, 18, 17, 28, 24, 27, 21, 23]

#perform the Wilcoxon-Signed Rank Test
stats.wilcoxon(group1, group2)
```

```
(statistic=10.5, pvalue=0.044)
```

СПАСИБО ЗА ВНИМАНИЕ!