Математическая статистика

Практикум 3

In [1]:
```
import numpy as np
import scipy.stats as sps
import scipy.stats as stats
import ipywidgets as widgets
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm

%matplotlib inline
import pandas as pd
```

Задача 1

Пусть $x_1, \ldots, x_n$ — реализация из модели сдвига показательного закона с плотностью распределения

$$f_\theta(u) = e^{-\lambda(u-\theta)} \cdot I_{\{u \geq \theta\}} = \begin{cases} e^{-\lambda(u-\theta)}, & u \geq \theta, \\ 0, & u < \theta. \end{cases}$$

Оценить θ с помощью метода моментов и метода максимального правдоподобия.
Реализуйте эту задачу в Python:

- сгенерируйте θ из равномерного распределения на $[1, N+1]$ и возьмите $\lambda = |N - 10|/N$, где `N` - ваш номер в журнале группы и ФИО(нумерация начинается с 01,02,03... и т.д.)
- сгенерируйте выборку из распределения с плотностью $f_\theta(u)$ размера $n = 1\,0000$;
- постройте алгоритм нахождения оценки методом моментов и методом правдоподобия (мы такой пример решали на занятии), а потом найдите значения оценок метода моментов и метода максимального правдоподобия, которые генерируются автоматически Питоном.

In [2]:
```
np.random.seed(100) # зафиксируем seed
```

Seed

Для того, чтобы каждый раз при использовании генератора псевдослучайных чисел получать идентичные последовательности, используется функция set.seed (от set - задать, установить, и seed - начальное число). Как следует из названия, эта функция фиксирует число, служащее начальной точкой для запуска алгоритма генерации псевдослучайных чисел. В качестве аргумента функции указывают любое целое число (не важно, какое именно).

- **np.random.seed(<число>)** —

настраивает генератор случайных чисел на новую последовательность. Если данная функция в программе не вызывается, по умолчанию используется системное время.

В качестве примера приведем еще фукции для распределений, которые мы упоминали в курсе по теории вероятностей:

- **np.random.binomial(n, p[, size])** — биномиальное распределение
- **np.random.exponential([scale, size])** — экспоненциальное распределение
- **np.random.poisson([lam, size])** — распределение Пуассона
- **np.random.standard_cauchy([size])** — распределение Коши

Обратите внимание на то, что в np.random.exponential параметр scale = $1/\lambda$.

In [3]:
```
N = 14 # номер варианта
λ = np.abs(N - 10) / N

N, λ
```

Out[3]:
```
(14, 0.2857142857142857)
```

In [4]:
```
n = 10000 # размер выборки
scale = 1 / λ
θ = np.random.uniform(1, N + 1)
```

In [5]:
```
sample = θ + np.random.exponential(scale, size = n) # генерируем выборку размера n
```

In [6]:
```
esimate_mme = np.mean(sample - scale) # оценка методом моментов
esimate_mle = np.min(sample) # оценка методом максимального правдоподобия
```

In [7]:
```
print(f'Истинное значение параметра: {θ}')
print(f'Оценка метода моментов: {esimate_mme}')
print(f'Оценка метода максимального правдоподобия: {esimate_mle}')
```

```
Истинное значение параметра: 8.607669185073515
Оценка метода моментов: 8.59730674766141
Оценка метода максимального правдоподобия: 8.607720350439438
```

Задача 2

Мы нашли точечные оценки методом моментов и методом правдоподобия, теперь надо оценить используя соответствующий параметрическому случаю критерий согласия, согласуется ли полученные распределения с данной выборкой. Good-to-fit критерии или критерии согласия с простой гипотезой. Используйте любой критерий, который проверит является ли распределение экспоненциальным с заданным показателем. Научитесь подбирать подходящие критерии. Какие критерии бывают можно посмотреть в книжке Медведева, Ивченко.

In [8]:
```
# оценка метода моментов
mme = esimate_mme + np.random.exponential(scale, size = n)
```

```
kstest_mme = sps.ks_2samp(sample, mme)
print('Результат с оценкой метода моментов')
print(f'Статистика: {kstest_mme.statistic}')
print(f'p-value: {kstest_mme.pvalue}\n')

# оценка метода максимального правдоподобия
mle = esimate_mle + np.random.exponential(scale, size = n)
kstest_mle = sps.ks_2samp(sample, mle)
print('Результат с оценкой метода максимального правдоподобия')
print(f'Статистика: {kstest_mle.statistic}')
print(f'p-value = {kstest_mle.pvalue}')
```

```
Результат с оценкой метода моментов
Статистика: 0.0079
p-value: 0.9139260465470475

Результат с оценкой метода максимального правдоподобия
Статистика: 0.0088
p-value = 0.8335435086261689
```

$p - value > 0.05 \Rightarrow$ нет оснований для отклонении гипотезы о том, что выборки принадлежат одному типу распределения.

Оценки методами 'моментов' и 'максимального правдоподобия' дают согласованные результаты.

Задача 3 Сравнить на выборках размера 100 для $\mathcal{N}(\theta, 4)$ доверительные интервалы: (1) теоретический, (2) на основе параметрического бутстрэпа, (3) на основе непараметрического бутстрэпа. Сам параметр θ сгенерировать из равномерного распределения на $[-5, 5]$.

In [9]:
```
n = 100 # размер выборки
α = 0.05 # уровень значимости

θ = np.random.uniform(-5, 5)
σ = np.sqrt(4)
```

In [10]:
```
sample = np.random.normal(θ, σ, n)
```

In [11]:
```
print(f'Значение θ: {θ}')
```

```
Значение θ: -3.6100308551787066
```

Теоретический доверительный интервал

Напомним, что теоретический доверительный интервал вычисляется следующим образом:

$$\mathbb{P}\left(\bar{X} - \frac{c_{1-\alpha/2}\sigma}{\sqrt{n}} < \mu < \bar{X} + \frac{c_{1-\alpha/2}\sigma}{\sqrt{n}}\right) = 1 - \alpha,$$

где c_α — квантиль распределения $\mathcal{N}(0, 1)$ уровня α.

Вычисляем теоретический доверительный интервал используя функции $stats.norm.ppf(1 - alpha/2)$ для вычисления кваниля c_α.

In [12]:
```
n = len(sample)
avg = np.mean(sample)
std = np.std(sample, ddof = 1)

quantile = stats.norm.ppf(1 - α / 2)

confidence_interval = (avg - quantile * std / np.sqrt(n), avg + quantile * std / np
print(f'Теоретический доверительный интервал: {confidence_interval}')
```

```
Теоретический доверительный интервал: (-3.888687526547435, -3.0792696230134338)
```

Доверительный интервал на основе параметрического бутстрэпа

In [13]:
```
number_of_bootstrap_samples = 10 # количество бутстрэп-выборок
size_of_bootstrap_samples = 20 # размер бутстрэп-выборок
```

In [14]:
```
bootstrap_samples = np.random.normal(np.mean(sample), σ, size = [number_of_bootstra
bootstrap_estimates = np.apply_along_axis(np.mean, 1, bootstrap_samples)

parametric = [np.quantile(bootstrap_estimates, α / 2), np.quantile(bootstrap_estima

print(f'Параметрический бутстрэп доверительный интервал: {parametric}')
```

```
Параметрический бутстрэп доверительный интервал: [-4.140715248471226, -2.894936580
7765646]
```

Доверительный интервал на основе непараметрического бутстрэпа

In [15]:
```
number_of_bootstrap_samples = 10 # количество бутстрэп-выборок
size_of_bootstrap_samples = 20 # размер бутстрэп-выборок
```

In [16]:
```
bootstrap_samples = np.random.choice(sample, size = [number_of_bootstrap_samples, s
bootstrap_estimates = np.apply_along_axis(np.mean, 1, bootstrap_samples)

nonparametric = [np.quantile(bootstrap_estimates, α / 2), np.quantile(bootstrap_est
print("Непараметрический бутстрэп доверительный интервал:", nonparametric)
```

```
Непараметрический бутстрэп доверительный интервал: [-4.634893425709553, -3.0533332
066753043]
```

Выводы: как сравнить полученные доверительные интервалы? Как сравнить методы? Какие лучше и почему?

In [17]:
```
print(f'Значение θ: {θ}\n')
print(f'Теоретический доверительный интервал: {confidence_interval}')
print(f'Параметрический бутстрэп доверительный интервал: {parametric}')
print(f'Непараметрический бутстрэп доверительный интервал: {nonparametric}')
```

```
Значение θ: -3.6100308551787066

Теоретический доверительный интервал: (-3.888687526547435, -3.0792696230134338)
Параметрический бутстрэп доверительный интервал: [-4.140715248471226, -2.894936580
7765646]
Непараметрический бутстрэп доверительный интервал: [-4.634893425709553, -3.0533332
066753043]
```

Как сравнить полученные доверительные интервалы?

Сравниваем интервалы по ширине

Как сравнить методы?

Доврительные интервалы содержат значение параметра θ. Разница в ширине:

In [18]:
```python
print(f'Значение θ: {θ}\n')

def is_inside(interval):
    return 'θ ' + ('' if interval[0] <= θ <= interval[1] else 'не ') + 'входит'

def print_stat(message, interval):
    print(f'{message}: {interval}, {is_inside(interval)}')
    print(f' - Ширина {interval[1] - interval[0]}\n')

print_stat('Теоретический доверительный интервал', confidence_interval)
print_stat('Параметрический бутстрэп доверительный интервал', parametric)
print_stat('Непараметрический бутстрэп доверительный интервал', nonparametric)
```

```
Значение θ: -3.6100308551787066

Теоретический доверительный интервал: (-3.888687526547435, -3.0792696230134338), θ
входит
 - Ширина 0.8094179035340012

Параметрический бутстрэп доверительный интервал: [-4.140715248471226, -2.894936580
7765646], θ входит
 - Ширина 1.2457786676946614

Непараметрический бутстрэп доверительный интервал: [-4.634893425709553, -3.0533332
066753043], θ входит
 - Ширина 1.5815602190342486
```

Какие лучше и почему?

θ входит во все интервалы, а теоретический самый узкий - поэтому он подходит лучше всего.