Курс "Практикум по математической статистике"

3 курс ФПМИ МФТИ, осень 2020

Домашнее задание 6. Проверка гипотез.

Дедлайн --- 21 декабря 9:00

Мы предлагаем выполнять задания прямо в этом ноутбуке. Пожалуйста, не стирайте условия задач.

Информация о выполнении и курсе в целом есть в этой папке (https://docs.google.com/document/d/1kd85QRAS8fbxRxpMzP2IsbQ_YcVsU-Aczqd6ErXgIDg/edit#).

Настоятельно рекомендуемая форма оформления домашних заданий — это Jupyter Notebook и его pdf-версия с:

- условием задачи,
- решением (если требуется некоторый теоретический вывод),
- описанием плана решения, который потом реализуется в коде,
- собственно кодом,
- построенными графиками (если это требуется) и **выводом**, который как правило должен заключаться в объяснении практических результатов с использованием теоретических фактов. ***Вывод требуется даже в том случае, если в условии об этом явно не сказано!***
- некоторыми другими вещами, если об этом будет указано в задании.

Оценка за каждую задачу складывается из правильного выполнения всех этих пунктов. Закрывая на них глаза, вы сознательно понижаете свою оценку.

Каждая задача оценивается **в 10 баллов**.

In [1]:

```
import numpy as np

import seaborn as sns

from matplotlib import pyplot as plt

from ipywidgets import interactive
from IPython import display

from scipy import stats as sps

sns.set(style="whitegrid", font_scale=1.4)
```

Задание 1 (Корреляция)

В этом задании вы на практике увидите, как коэффициенты корреляции реагируют на выбросы в данных, а также познакомитесь с виджетами. Если некоторые импорты поломались, то продолжите работу в [колабе (https://colab.research.google.com)](https://colab.research.google.com).

Итак, наша задача – сгенерировать (почти) линейно зависимую выборку выборку и добавить в нее выбросы.

Начнем с первого: сгенерируйте выборку из двух столбцов $$ Y = 10 \cdot X + \varepsilon $$ где X и ε из стандартного нормального распределения. Выборка состоит из двух столбцов X и Y. **Размер выборки — 100 элементов**.

Совет: в данном случае проще использовать `np.random.randn` и `np.hstack` .

In [2]:

```
def generate_linear_sample():
    np.random.seed(42)
    size=100
    data_x = np.array([np.random.randn(size)]).T
    data_y = 10 * data_x + np.array([np.random.randn(size)]).T
    data = np.hstack((data_x, data_y))
    return data
```

Теперь займемся выбросами (outliers). Генерировать их будем следующим образом: для первых `outliers_num` элементов мы поменяем значение X на случайную величину из стандартного распределения (`np.random.randn`), а значение Y сделаем равное `outlier_bias` .

In [3]:

```
def add_outliers(
    data: np.ndarray, outliers_num: int, outlier_bias: float
) -> np.ndarray:
        np.random.seed(42)
        # YOUR_CODE_MAYBE_GOES_HERE
        data[list(range(outliers_num))] = np.hstack(
            (np.array([np.random.randn(outliers_num)]).T, np.array([np.repeat(outlier_bias, outliers_num)]).T)
        )
        return data
```

Теперь создадим функцию, которая генерирует выборку и считает коэффициенты корреляции. Итак, посчитайте коэффициенты корреляции Пирсона и Спирмана, а так же уровень достигаемой значимости при проверки гипотезы о некоррелированности X и Y для каждого из коэффициентов (все делается одной функцией из `scipy.stats`). Затем в этой же функции постройте график (scatter). Отметьте все выбросы красным, а остальные точки синим. **В заголовке графика** напишите коэффициенты корреляции, округленные до четырех знаков после запятой. Например так

```
f"Pearson correlation {p_corr:.4f}\n Spearman correlation {s_corr:.4f}"
```

И верните уровни достигаемой значимости (**p-value**) критериев о некоррелированности.

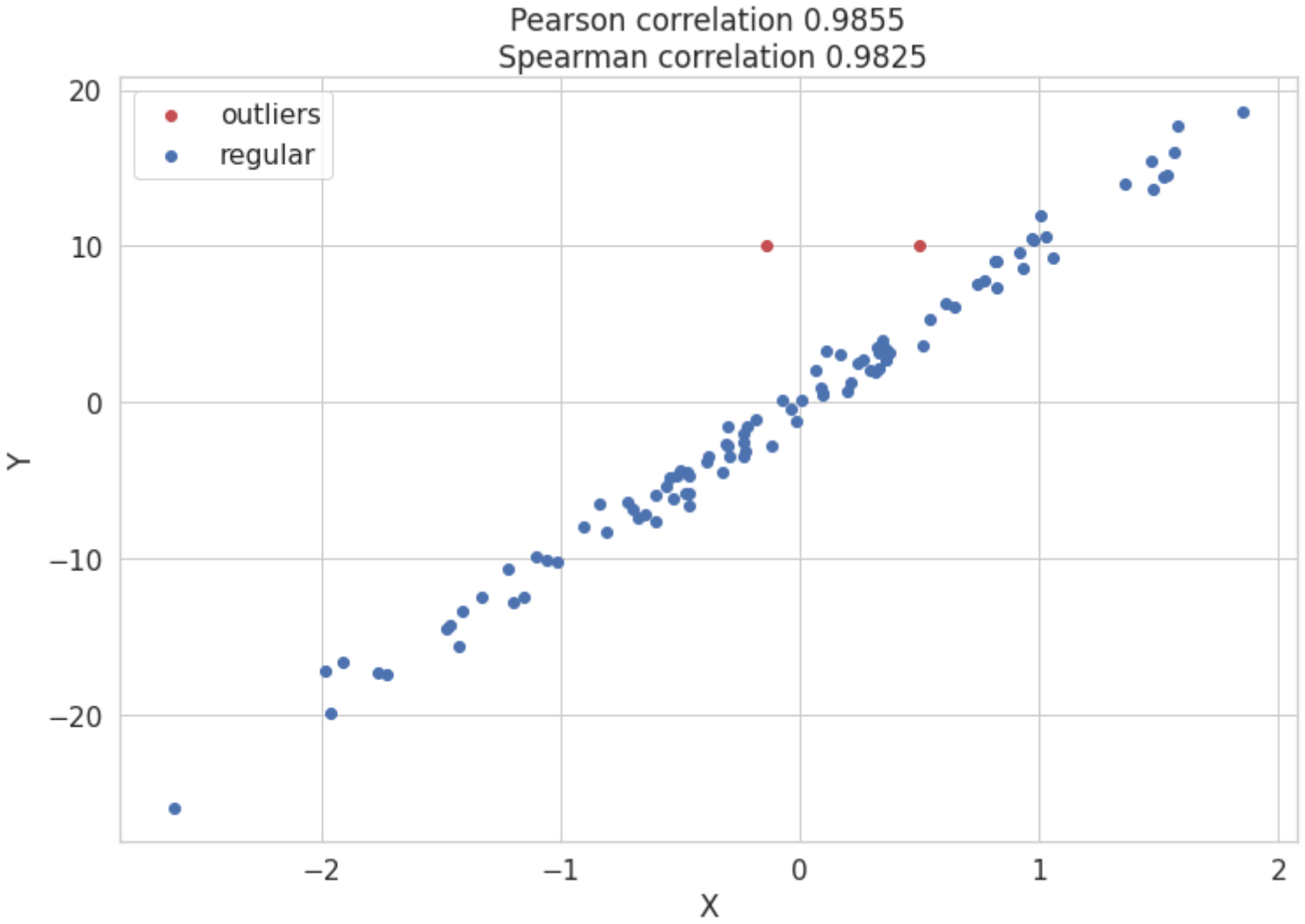
In [4]:

```
def plot_correlation(
    outliers_num: int = 2, outlier_bias: float = 10, show=True
):
    np.random.seed(42)
    data = generate_linear_sample()
    data = add_outliers(data, outliers_num, outlier_bias)
    plt.figure(figsize=(12, 8))
    p_corr, p_p_val = sps.pearsonr(data[:, 0], data[:, 1])
    s_corr, s_p_val = sps.spearmanr(data[:, 0], data[:, 1])
    plt.title(
        f"Pearson correlation {p_corr:.4f}\n Spearman correlation {s_corr:.4f}"
    )
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.scatter(data[:outliers_num, 0], data[:outliers_num, 1], color='r', label='outliers')
    plt.scatter(data[outliers_num:, 0], data[outliers_num:, 1], color='b', label='regular')
    plt.legend()
    if show:
        plt.show()
    else:
        plt.close()
    return p_p_val, s_p_val
```

Чтобы убедится, что все ок, вызовите функцию с параметрами по умолчанию ниже.

In [5]:

```
plot_correlation()
```


Pearson correlation 0.9855
Spearman correlation 0.9825
outliers
regular
20
10
0
−10
−20
Y
−2
−1
0
1
2
X

Out[5]:

```
(2.920497051061722e-77, 2.2377121996722538e-73)
```

Часто, чтобы смотреть за изменением величин в зависимости от параметров бывает удобно использовать виджеты. Мы будем пользоваться самой простой функцией из этой библиотеки – `interactive`. Она позволяет быстро создать ползунки соответствующие аргументу функции, нужно лишь указать краевые значения параметров.

In [6]:

```
v = interactive(
    plot_correlation, outliers_num=(1, 10), outlier_bias=(-300., 300.)
)
```

После этого осталось лишь отобразить наш виджет.

In [7]:

```
display.display(v)
```

Поиграйтесь с ползунками, посмотрите как коэффициенты корреляции реагируют на выбросы. Сделайте вывод.

Вывод: Зависимость выборок почти линейная. Следовательно, коэффициент корреляции должен быть близким к 1.

Так как коэффициент корреляции Пирсона обращает внимание на значения элементов выборок, он крайне чувствителен к выбросам. Если выбросы сделать отрицательными и большими по модулю, можно получить даже отрицательные значения, то есть якобы обратную корреляцию.

Коэффициент корреляции Спирмэна не обращает внимание на сами значения: только на их ранги. Благодаря этому он менее чувствителен к выбросам, но всё равно: при 10% выбросов с большим по модулю отрицательным значением коэффициент корреляции Спирмэна ≈ 0.7.

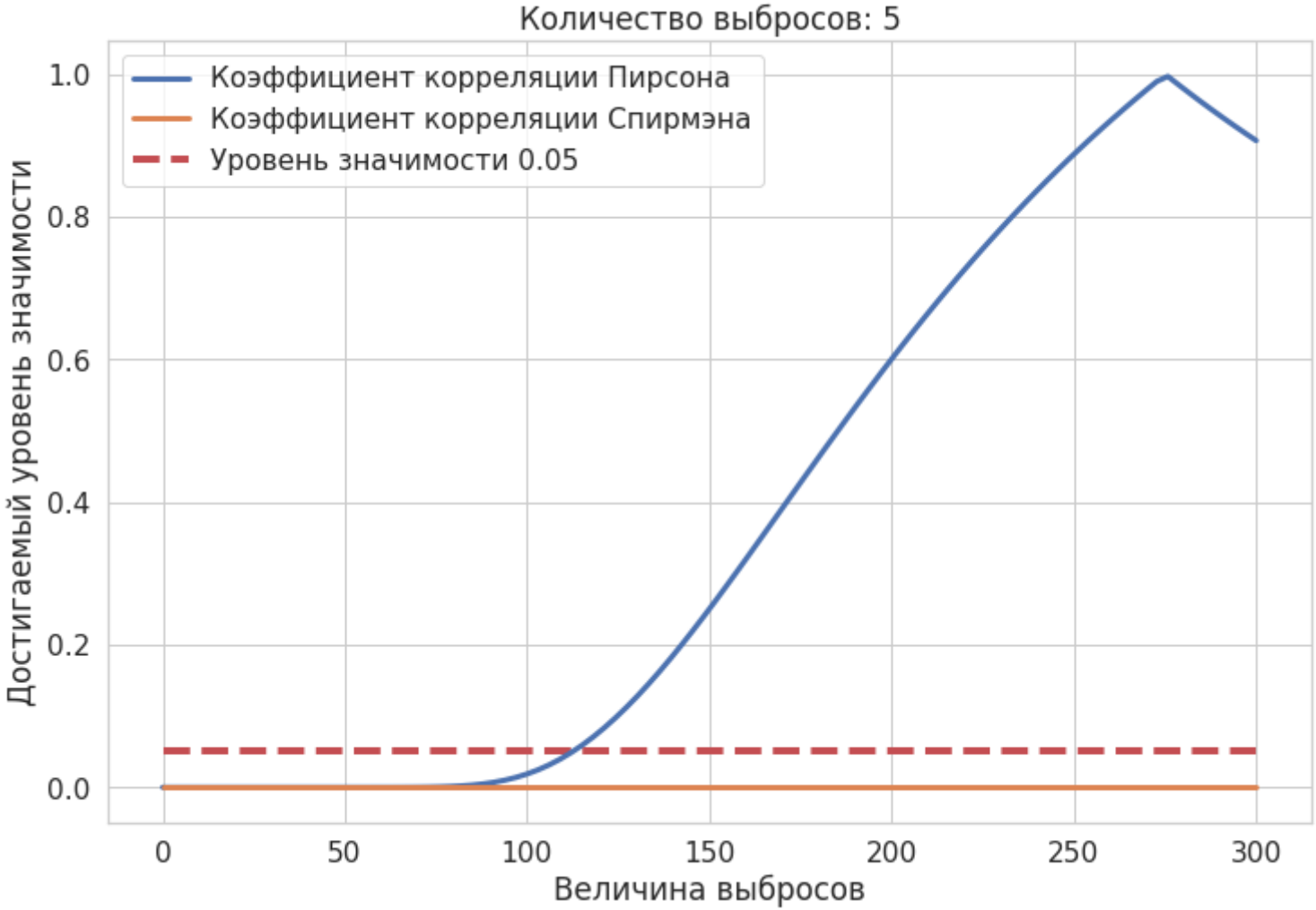
Давайте зафиксируем число выбросов `outliers_num=5`. Для параметра `outlier_bias` в интервале от -300 до 0 (`np.linspace`) посчитайте значения `p_value` (достигаемый уровень значимости) для обоих критериев. Постройте график зависимости `p_value` от **модуля** отклонения остатков (`outlier_bias`). Должны получиться две линии для двух коэффициентов корреляции. А также постройте вспомогательную красную линию `plt.hlines` на уровне 0.05. Не забудьте про легенду!

In [8]:

```
x = np.linspace(0, -300, 100)
samples = [add_outliers(generate_linear_sample(), 5, bias) for bias in x]
# p_values = # p_values for pearson and spearman
p_values = [
    [sps.pearsonr(sample[:, 0], sample[:, 1])[1] for sample in samples],
    [sps.spearmanr(sample[:, 0], sample[:, 1])[1] for sample in samples]
]
```

In [9]:

```
plt.figure(figsize=(12, 8))
plt.title(r'Количество выбросов: 5')
plt.xlabel('Величина выбросов')
plt.ylabel('Достигаемый уровень значимости')
plt.plot(np.abs(x), p_values[0], lw=3, label='Коэффициент корреляции Пирсона')
plt.plot(np.abs(x), p_values[1], lw=3, label='Коэффициент корреляции Спирмэна')
plt.hlines(
    0.05,
    np.min(np.abs(x)),
    np.max(np.abs(x)),
    linestyles='dashed',
    lw=4,
    color='r',
    label='Уровень значимости 0.05'
)
plt.legend()
plt.show()
```


Количество выбросов: 5
Коэффициент корреляции Пирсона
Коэффициент корреляции Спирмэна
Уровень значимости 0.05
Достигаемый уровень значимости
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
Величина выбросов

Сделайте вывод по полученному графику.

Вывод: При больших по модулю выбросах достигаемый уровень значимости коэффициента корреляции Пирсона начинает сильно расти вплоть до 1. Это значит, что он становится очень неточным или даже вовсе неприменимым.

Коэффициент корреляции Спирмэна же, напротив, даже при больших выбросах сохраняет низкий, близкий к нулю уровень значимости, то есть очень точен.

Это снова может быть объяснено тем, что коэффициент корреляции Пирсона учитывает величины выборки напрямую, то есть сильно от них зависит, в то время как коэффициент корреляции Спирмэна учитывает лишь их ранг, поэтому более устойчив к выбросам.

Задание 2 (Мощность критерия)

В реальной жизни аналитика самая распространенная задача это A/B-тестирование. Оно применяется практически везде, где это возможно. Идея проста: разбиваем людей на две группы A и B. Группе A мы даем продукт без изменений (принято называть эту группу `placebo`, даже когда речь не идет о медицине), а группе B (`treatment`) мы даем продукт, с каким-то изменением. Мы хотим понять, полезно ли нам предложенное изменение. Поэтому мы считаем какую-то метрику для двух этих групп и пытаемся понять, значимо ли изменение. Однако важный вопрос понять необходимый размер групп A и B.

Давайте представим себе такой случай: студент ведет два паблика с мемами. Один про лектора по статам, а другой про лектора по теории меры. Но наступает сессия, и времени у него остается немного, поэтому он думает о том чтобы закрыть первый паблик, потому что мемы в нем уже не актуальны и лайкают их мало. Вы, как опытный эксперт, можете довольно точно прикинуть распределение лайков под записями. По вашему мнению оба распределения нормальные. Для первого паблика: $\mu_1 = 35$ $\sigma_1 = 30$, для второго: $\mu_2 = 55$ $\sigma_2 = 30$. Считаем, что аудитории пабликов не пересекаются. Вы хотите убедить студента, что ему выгодно закрывать именно первый паблик и сконцентрироваться на втором, но он вам не верит и требует статистически доказать правоту: он требует, чтобы мощность критерия была не менее не менее 0.95, а уровень значимости $\alpha = 0.05$. Вы предлагаете ему следующий сценарий: чтобы избежать влияние других факторов нужно выкладывать мемы в двух пабликах одновременно раз в день. Необходимо понять сколько дней требуется для подведения итогов.

Введем обозначения для количества лайков: X ~ $N(\mu_1, \sigma^2_1)$; Y ~ $N(\mu_2, \sigma^2_2)$.

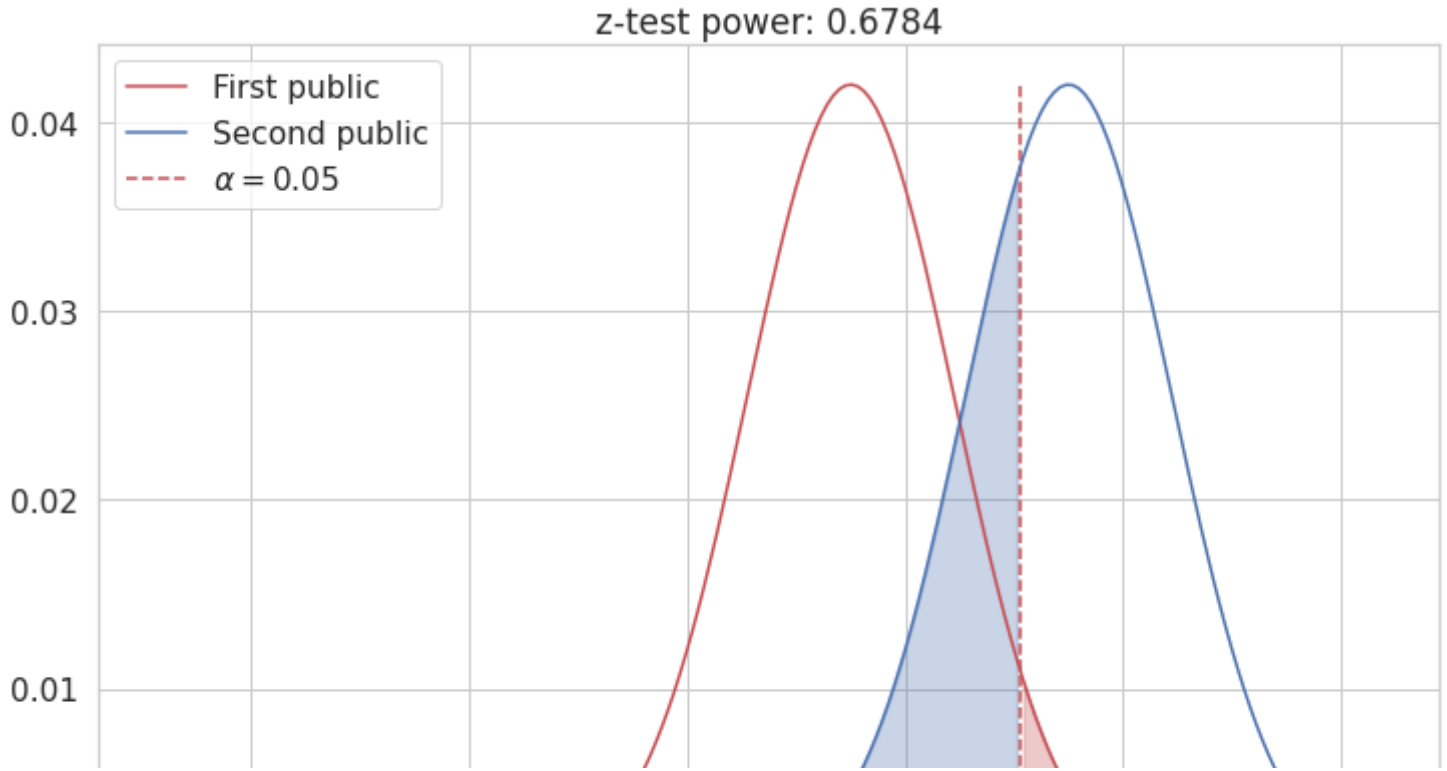
Для проверки гиоптезы вы предложили воспользоваться Z-критерием для односторонней альтернативы:

$H_0: \mu_1 = \mu_2$ vs $H_1: \mu_2 > \mu_1$.

Статистика этого критерия выглядит так: $$\dfrac{\overline{Y} - \overline{X}}{\sigma / \sqrt{n}} = \dfrac{\overline{Y}}{\sigma/\sqrt{n}} - \dfrac{\overline{X}}{\sigma / \sqrt{n}}.$$

Напишите функцию которая строит плотности $\overline{X}$ и $\overline{Y}$ в зависимости от n красным и синим цветом соответственно. **Не гистограмму или kde, а именно теоретическую плотность.** Так же закрасьте область соответствующую ошибке первого рода красным (α), а ошибке второго рода синим (β) (`plt.fill_between`). В заголовке напишите мощность критерия ($1 - \beta$).

Должен получится примерно такой график:


z-test power: 0.6784
First public
Second public
$\alpha = 0.05$
0.04
0.03
0.02
0.01

In [10]:

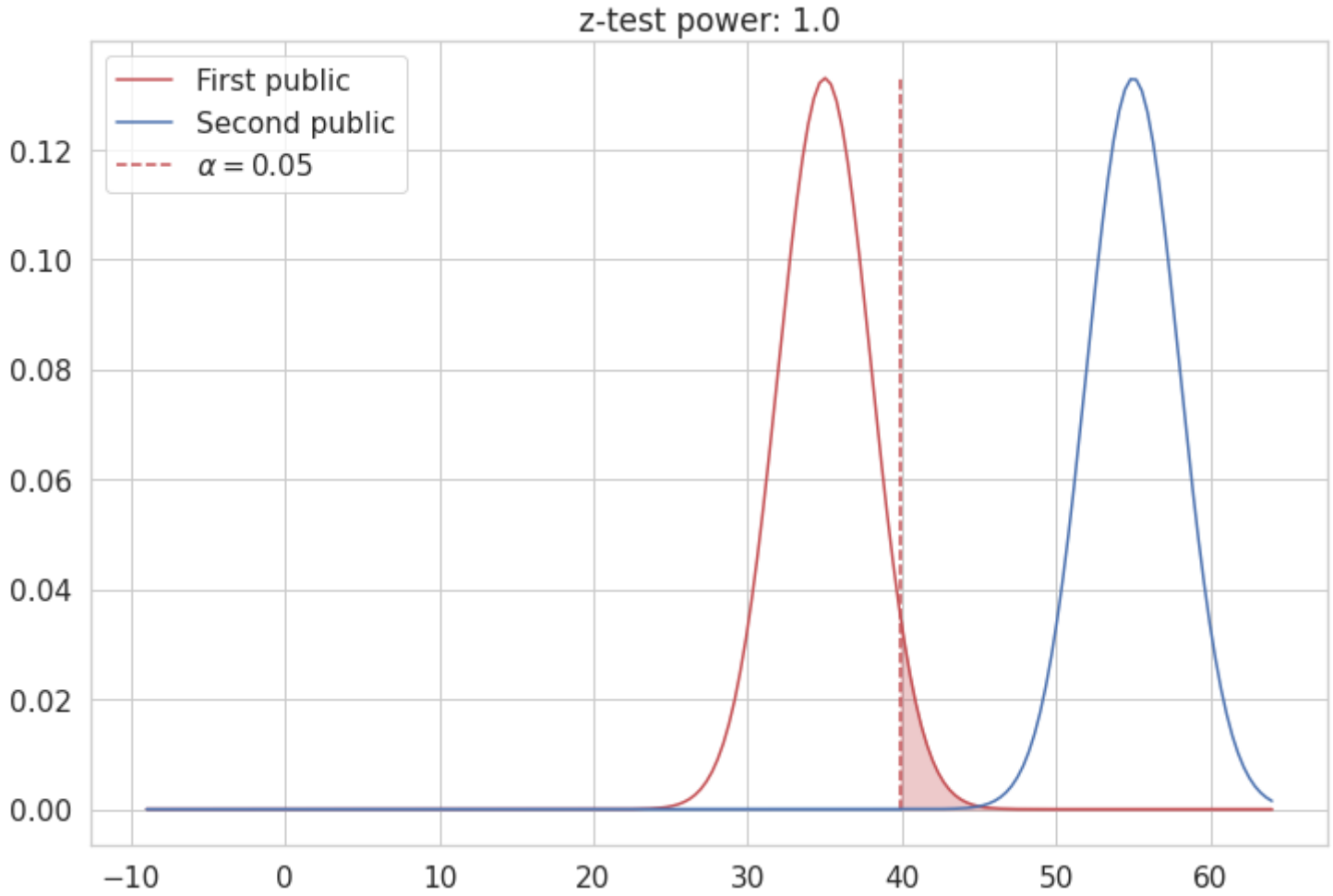
```
def plot_power_and_pvalue(
    n: int = 100,
    mu_1: float = 35,
    mu_2: float = 55,
    sigma_1: float = 30,
    sigma_2: float = 30,
    alpha: float = 0.05,
    show: bool = True,
):
    assert mu_1 < mu_2
    assert n > 0
    sigma_n_1 = sigma_1 / np.sqrt(n)
    sigma_n_2 = sigma_2 / np.sqrt(n)
    x = np.linspace(-3*sigma_n_1, mu_2+3*sigma_n_2, 200)
    pdf_n_1 = sps.norm(mu_1, sigma_n_1).pdf
    pdf_n_2 = sps.norm(mu_2, sigma_n_2).pdf
    quantile = sps.norm(loc=mu_1, scale=sigma_n_1).ppf(1-alpha)
    beta = sps.norm(loc=mu_2, scale=sigma_n_2).cdf(quantile)
    # PLOT (plot, fill_between, vlines)
    plt.figure(figsize=(12, 8))
    plt.title(f'z-test power: {np.round(1-beta, 4)}')
    plt.plot(x, pdf_n_1(x), color='r', label='First public')
    plt.plot(x, pdf_n_2(x), color='b', label='Second public')
    plt.vlines(
        quantile,
        0, np.max(np.hstack((pdf_n_1(x), pdf_n_2(x)))),
        linestyles='dashed', color='r', label=r'$\alpha = 0.05$'
    )
    type1_error_x = np.linspace(quantile, x[-1], 100)
    plt.fill_between(type1_error_x, pdf_n_1(type1_error_x), alpha=0.3, color='r')
    type2_error_x = np.linspace(x[0], quantile, 100)
    plt.fill_between(type2_error_x, pdf_n_2(type2_error_x), alpha=0.3, color='b')

    if show:
        plt.legend()
        plt.show()
    else:
        plt.close()
```

```
    return beta
```

Запустите функцию с параметрами по умолчанию для проверки.

In [11]:

```
plot_power_and_pvalue()
```



Out[11]:

```
2.55929977464261e-07
```

Теперь снова поиграйтесь с ползунком.

In [12]:

```
v = interactive(
    lambda n: plot_power_and_pvalue(n), n=(10, 100)
)
```

In [13]:

```
display.display(v)
```

Сколько дней нужно для достижения мощности критерия ≥ 0.95 на уровне $\alpha=0.05$?

Ответ: 25 дней

In [13]:

```

```