Прикладний статистичний аналіз

Посібник для студентів

Ігор Мірошниченко Юлія Хлевна

2025-04-16

Зміст

Передмова

Цей посібник створено для тих, хто хоче не просто вивчати статистику, а розуміти, як вона працює на практиці — у даних, рішеннях, дослідженнях. Ми починаємо з простого, але фундаментального — з біноміального критерію — і поступово розширюємо горизонти: від гіпотез й довірчих інтервалів до t- та Z-критеріїв та Монте-Карло моделювання.

Кожен розділ поєднує теоретичне підґрунтя, інтерпретації, візуалізації і, що важливо, реалізацію в Python. Це не академічна теорія заради теорії. Тут — статистика як інструмент: перевіряти гіпотези, оцінювати ефекти, шукати закономірності та ухвалювати обґрунтовані рішення.

Цей посібник буде корисним:

- студентам, які хочуть навчитися статистичного мислення;
- дослідникам, які працюють з експериментами;
- аналітикам, які мають справу з вибірками та оцінками;
- і всім, хто не хоче просто "вивчити формули", а прагне зрозуміти суть.

Ми не обіцяємо, що буде завжди легко. Але гарантуємо, що буде зрозуміло.

Як користуватись цим посібником

Цей посібник можна проходити послідовно — від першої теми до останньої — або звертатись до окремих розділів за потреби. Щоб отримати максимум користі, ось кілька порад:

1. Читайте, але й кодьте. Теорія дає розуміння, але практика — вміння. Якщо бачите фрагмент коду — спробуйте його виконати, змінити параметри, подивитись, що буде.
2. Грайте з прикладами. В багатьох місцях можна змінити дані, розміри вибірки чи рівень значущості — робіть це. Так найкраще приходить інтуїція.
3. Повертайтесь назад. Якщо на якомусь етапі щось здається складним, це нормально. Часто новий матеріал "вмикає" краще розуміння попереднього.
4. Не бійтесь помилятись. Статистика — це не про ідеальні відповіді, а про ймовірності, невизначеність і пояснення того, що ми бачимо.
5. Задавайте собі питання:
    - Яку гіпотезу ми тут перевіряємо?
    - Що означає результат?
    - Що буде, якщо змінити розмір вибірки чи очікувану різницю?
    - Як це застосувати в моїй сфері?
6. Не ігноруйте "інтерпретацію". Це не просто розділ — це ключ до розуміння. Бо мало просто отримати результат — важливо знати, що він означає.

Досліджуйте далі. Цей посібник — стартова платформа. Якщо вам цікаво — глибше копати тільки вітається.

Попередні налаштування Python

Для роботи з цим посібником вам знадобиться Python. В процесі написання ми використовували наступні глобальні бібліотеки та змінні:

```
import numpy as np                                                                    ①
import seaborn as sns
import matplotlib.pyplot as plt
import random
import time
from statsmodels.stats.proportion import proportion_confint
from IPython.display import Markdown, display

from scipy.stats import (
    norm,
    binom,
    expon,
    binomtest,
    chisquare,
    t,
    chi2,
    pareto,
    ttest_1samp,
    ttest_ind,
    sem,
    bernoulli,
    uniform,
    gamma
)

np.random.seed(73)                                                                    ②

sns.despine(left=True, bottom=True)                                                   ③
base_size = 13
sns.set_style("whitegrid")
palette = sns.color_palette("Set2")
sns.set_palette(palette)
sns.set_context("talk")

plt.rcParams['figure.figsize']      = (8, 3.5)                                        ④
plt.rcParams['font.size']           = 12
plt.rcParams['axes.labelsize']      = 12
plt.rcParams['axes.titlesize']      = 12
plt.rcParams['xtick.labelsize']     = 8
plt.rcParams['ytick.labelsize']     = 8
plt.rcParams['legend.fontsize']     = 8
plt.rcParams['figure.titlesize']    = 12
plt.rcParams['grid.color']          = (0.5, 0.5, 0.5, 0.1)
plt.rcParams['axes.spines.top']     = False
plt.rcParams['axes.spines.right']   = False
plt.rcParams['axes.spines.left']    = False
plt.rcParams['axes.spines.bottom']  = False

red_pink   = "#e64173"                                                                ⑤
turquoise  = "#20B2AA"
orange     = "#FFA500"
red        = "#fb6107"
blue       = "#181485"
navy       = "#150E37FF"
green      = "#8bb174"
yellow     = "#D8BD44"
purple     = "#6A5ACD"
slate      = "#314f4f"
```

① Ці бібліотеки використовуються для статистичних обчислень, візуалізації та роботи з даними.
② Встановлюємо випадкове зерно для відтворюваності результатів.
③ Налаштовуємо стиль графіків для кращої візуалізації.
④ Налаштовуємо параметри графіків для кращої візуалізації.
⑤ Визначаємо кольори для графіків.

Вступ

Статистика — це не просто набір формул чи методів. Це мова, якою ми говоримо про невизначеність, про вибірки, про те, що можна сказати про ціле, спостерігаючи лише частину. Зрештою, це інструмент, який допомагає ухвалювати рішення в умовах, коли 100% впевненості немає — й, чесно кажучи, рідко коли буває.

У цьому посібнику ми пройдемо шлях від базових понять (що таке генеральна сукупність й вибірка, як формулюються гіпотези) до застосування складніших методів, таких як t-критерій Стьюдента, Z-критерій Фішера і моделювання Монте-Карло.

Ми навмисно зосередилися на невеликій, але потужній добірці тем, які дозволяють:

- зрозуміти логіку перевірки гіпотез;
- інтерпретувати p-значення;
- оцінювати потужність дослідження та мінімальний ефект;
- будувати довірчі інтервали;
- і, найголовніше, — робити це все в Python.

Це практичний посібник. Підкріплений кодом, прикладами і поясненнями. Він не охоплює всю статистику (та й хто б це зміг!), але дає міцну базу і навички, які можна використовувати прямо зараз.

Почнемо з простого — але вже з першої сторінки будемо робити справжню статистику.

Chapter 1

Біноміальний критерій

1.1 Генеральна сукупність та вибірка

Ви вирішили створити платформу онлайн-курсів з програмування. Ви записали навчальні відео та запропонували користувачам доступ за передплатою. Вартість курсу для студента становить 1000 гривень, а витрати на підтримку платформи та індивідуальні консультації коштують вам 500 гривень з кожного студента.

Проте ви помічаєте, що деякі люди відмовляються від курсу після першого заняття, якщо матеріал їм здається складним або нецікавим. Інвестори готові підтримати ваш проєкт, якщо рівень відмов буде нижче 50%.

Щоб це перевірити, ви проводите експеримент: залучаєте 30 нових студентів. 20 із них проходять курс й оплачують доступ, а 11 відмовляються. 20 — це більше половини, але чи достатньо цього, щоб довести перспективність проєкту?

Розв'язуючи таку задачу, ми припускаємо, що існує певна аудиторія, яка користуватиметься нашим сервісом. Цю групу називають **генеральною сукупністю**. Якщо запустити сервіс для всіх потенційних користувачів, у ньому буде певна частка успішних випадків, позначимо її як μ. Це невідомий параметр, який ми не можемо визначити безпосередньо. Натомість ми можемо проводити експерименти та **досліджувати** результати. Оскільки протестувати продукт на всій аудиторії неможливо, ми беремо **вибірку** з генеральної сукупності та аналізуємо частку успішних випадків.

Згідно з результатами нашого експерименту, спостережувана ймовірність оплати становить $\hat{\mu} = 20/30 = 0.67$[1]. Це означає, що 67% студентів оплатили доступ. Чи можемо ми зробити висновок, що справжня частка успішних випадків перевищує 50%?

Розгляньмо, чому отримане значення *може не бути* переконливим доказом. Припустимо, що ймовірність успішної оплати дорівнює $\mu = 0.5$, і змоделюємо можливі результати для 30 студентів.

Давайте спростимо цю задачу до прикладу з підкиданням монетки та змоделюємо результати для 30 спроб:

- Якщо монетка випаде орлом, студент оплачує доступ.
- Якщо монетка випаде решкою, студент відмовляється від курсу.
- Використаємо метод `integers()` [2] до класу `Generator`, яка генерує випадкові цілі числа в заданому діапазоні.
- Підкинемо монетку 30 разів та порахуємо кількість успішних випадків.

```
rng = np.random.default_rng(seed=18)                                    ①

n = 30                                                                  ②
```

[1] У статистиці $\hat{\mu}$ позначається як оцінка параметра μ.

[2] Метод `integers()` генерує випадкові цілі числа в заданому діапазоні. Аргумент `endpoint` вказує, що верхня межа включається у діапазон.

```
results = rng.integers(0, 1, size = 30, endpoint = True)                    ③
success = np.sum(results) / n                                               ④

print(f"Кількість успішних випадків: {round(success, 3) * 100}%")          ⑤
```

① Ініціалізуємо генератор випадкових чисел з фіксованим `seed`.
② Кількість студентів.
③ Генеруємо випадкові числа для кожного студента.
④ Обчислюємо частку успішних випадків.
⑤ Виводимо результат.

```
Кількість успішних випадків: 70.0%
```

Ми бачимо, що в експерименті частка успішних випадків навіть перевищила 63%, тоді як у симуляції була закладена ймовірність 50%.

Тому, на жаль, ми не можемо з абсолютною точністю визначити, яким є справжнє значення μ у генеральній сукупності та чи перевищує воно 50%, незалежно від того, скільки спостережень ми проводимо. Однак, застосовуючи методи прикладної статистики, ми зможемо використати інструменти, які допоможуть ухвалити правильне рішення, зокрема й у цьому випадку.

1.2 Статистичні гіпотези

1.2.1 Постановка задачі

Ми з'ясували, що навіть за ймовірності $\mu = 0.5$ можна отримати значну кількість успішних випадків. Насправді ми спеціально підбирали `seed` для отримання такого результату. Якщо повторити цей експеримент з іншим значенням `seed` або збільшити кількість спостережень, результат може виявитися іншим.

> Порада
>
> Спробуйте змінити `seed` (наприклад 22) або кількість спостережень та перевірте, як змінюється результат.

Тож велика кількість успішних випадків може бути результатом випадковості. Щоб вирішити, чи можна вважати результати експерименту **статистично значущими** необхідно отримати відповідь на питання:

> Чи можна вважати, що спостережуване значення $\hat{\mu}$ є більшим від $\mu = 0.5$?

Звернімося до теорії ймовірностей. Факт підписки на наш сервіс для кожного окремого студента можна розглядати як випадкову величину ξ, яка підпорядковується розподілу Бернуллі[3]. Параметр цього розподілу, а саме ймовірність успіху, нам невідомий.

$$\xi \sim \text{Bernoulli}(\mu)$$

де μ — ймовірність успіху.

Нас цікавить підтвердження того, що $\mu > 0.5$. У статистиці для перевірки гіпотез розглядають дві можливості:

- **Нульова гіпотеза** (H_0) формулюється як твердження, яке ми прагнемо спростувати.
- **Альтернативна гіпотеза** (H_1) висловлює припущення, яке ми хочемо довести.

[3]Розподіл Бернуллі — це дискретний розподіл ймовірностей, який моделює випадковий експеримент з двома можливими результатами: успіхом або невдачею.

Скорочено це записують як:

$$H_0 : \mu \leq 0.5$$
$$H_1 : \mu > 0.5$$

Зауважимо, що якщо в нашому експерименті з 30 студентами можна дивитися не на частку успіхів, а на їх **кількість**.

Тоді питання можна переформулювати так:

> За умови вірності H_0 наскільки ймовірно отримати 20 або більше успішних випадків з 30?

Якщо ми проводимо n незалежних спостережень, то сума цих випадкових величин також підпорядковується біноміальному розподілу[4].

$$S_n = \sum_{i=1}^{n} \xi_i \sim \text{Binomial}(n, \mu)$$

де ξ_i — випадкова величина, яка показує успіх у i-му спостереженні, S_n — кількість успішних випадків у n спостереженнях, n — кількість спостережень, μ — ймовірність успіху.

Давайте подивимось, як це виглядає графічно. Для цього побудуємо графік функції щільності ймовірностей для біноміального розподілу з параметрами $n = 30$ та $\mu = 0.5$.

```
n = 30                                                                    ①
mu = 0.5                                                                  ②

x = np.arange(0, n + 1)                                                   ③
y = binom.pmf(x, n, mu)                                                   ④

plt.bar(x, y, color=turquoise)                                            ⑤
plt.bar(x[x >= 20], y[x >= 20], color=red_pink)                           ⑥
plt.xlabel("Кількість успішних випадків")
plt.ylabel("Ймовірність")
plt.show()
```

① Кількість студентів.
② Ймовірність успіху.
③ Створюємо масив з усіма можливими значеннями кількості успішних випадків.
④ Обчислюємо ймовірності для кожної кількості успішних випадків.
⑤ Створюємо гістограму з ймовірностями.
⑥ Виділяємо ймовірності для кількості успішних випадків, які є більшими або рівними 20.

[4] Біноміальний розподіл моделює кількість успішних випадків у послідовності незалежних випробувань. Сума n незалежних випадкових величин з розподілу Бернуллі підпорядковується біноміальному розподілу.

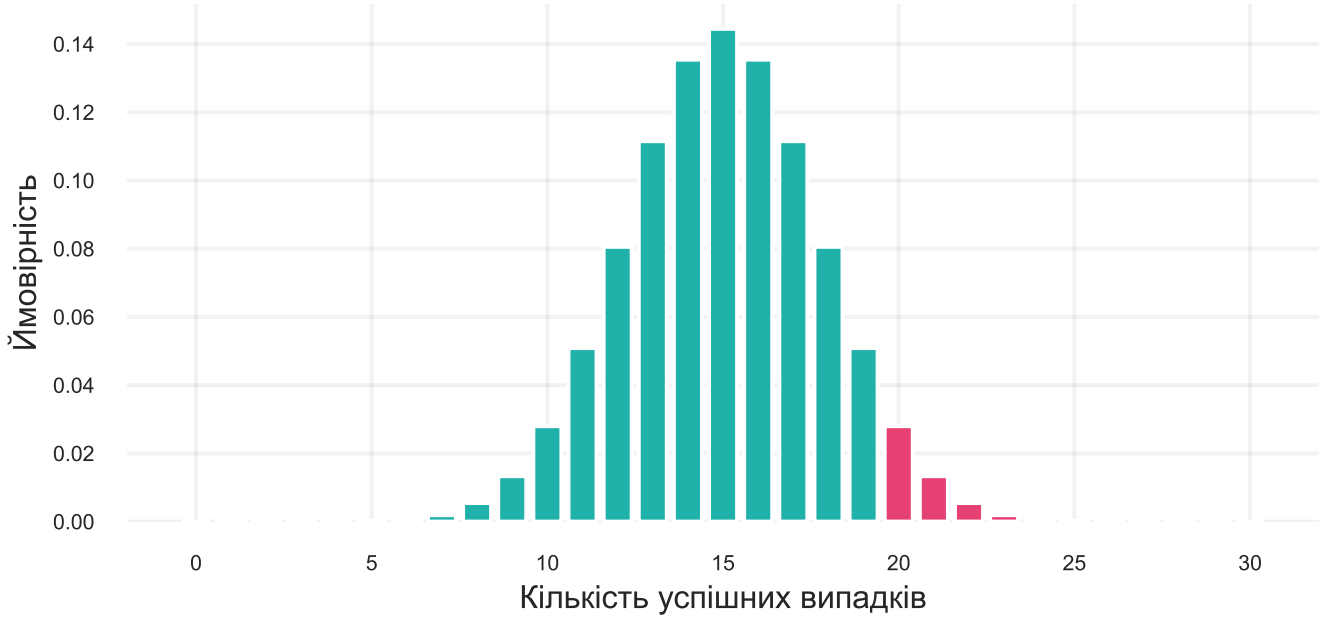


Рисунок 1.1: Візуалізація функції щільності ймовірностей для біноміального розподілу

Ціановим[5] кольором позначено ймовірності для кожної кількості успішних випадків. Рожевими виділено ймовірності для кількості успішних випадків, яка перевищує або дорівнює 20.

1.2.2 Критерій

Щойно ми розробили алгоритм, який на основі вибірки ξ або визнає наявність доказів на користь H_1, або повідомляє, що таких доказів немає. Відповідно, він або відхиляє H_0, або не відхиляє її.

Такий алгоритм називається **критерієм**. Його можна подати у вигляді функції S, яка приймає реалізацію вибірки та повертає 1, якщо слід відхилити H_0, та 0 в іншому випадку.

$$S(\xi) = \begin{cases} 1, & \text{якщо відхиляємо } H_0 \\ 0, & \text{в іншому випадку} \end{cases}$$

Давайте припустимо, що ми вирішили відхилити H_0, якщо кількість успішних випадків перевищує або дорівнює 21. Тоді критерій набуде вигляду:

$$S(\xi) = \begin{cases} 1, & \text{якщо } \sum \xi_i \geqslant 21 \\ 0, & \text{в іншому випадку} \end{cases}$$

Зазвичай скорочують запис і пишуть просто правило, за яким відхиляємо H_0

$$S = \{\sum \xi_i \geqslant 21\}$$

Позначимо $Q = \sum \xi_i$, $C = 21$, тоді критерій набуває вигляду:

$$S = \{Q(\xi) \geqslant C\}$$

{#sec-crit}

[5]Англ. *cyan*, від грец. κυανους — “блакитний”, “лазуровий”.

Так влаштована більшість класичних критеріїв у прикладній статистиці, тому величинам у ньому дано спеціальні назви. Q називається **статистикою критерію**, C — **критичним значенням**.

Q може бути будь-якою функцією від вибірки, яку ви вважаєте логічною для перевірки гіпотези. У нашому випадку це кількість успіхів, або сума всіх ξ_i. Але ви можете вибрати й інші: максимальне значення, суму перших 5 значень або навіть просто перший елемент.

1.2.3 Критична область

Знову перепишемо наше основне запитання, тільки тепер з використанням нашого критерію S:

Наскільки часто може бути таке, що за справедливості H_0 критерій S відхиляє гіпотезу?

Відповідь на це запитання залежить від критичного значення. Зараз ми взяли його рівним 21, побачивши на картинці, що великі відхилення відбуваються при H_0 рідко. Але що означає рідко й наскільки рідко, не сказали. Тепер наша мета зрозуміти, як вибрати критичне значення C, виходячи з **частоти помилок** нашого критерію.

Вибираючи C, ми можемо або часто відхиляти нульову гіпотезу, коли C мале, або можемо робити це рідше, коли C велике. Щоб вибрати правильне значення, потрібно визначитися, коли наш критерій помиляється.

- $C = 16$. Якщо відхиляти гіпотезу при отриманні хоча б 16 успішних підписок із 30, то це навряд чи влаштує інвесторів. Так, успіхів більше половини. Але якщо в генеральній сукупності ймовірність 0.5, то майже в половині випадків ми будемо відхиляти гіпотезу. Критерій помилково повертає 1, тобто це помилка **хибно позитивна** (false positive, **FP**).
- $C = 29$. У такому разі будемо відхиляти гіпотезу тільки за 29 або 30 успіхів. Ці значення, звісно, говорять про те, що відхилення від 50% успіхів сильне. Але якщо в генеральній сукупності ймовірність, наприклад, 60%, то такі значення будуть виходити рідко. Але ж такі ймовірності теж влаштували б інвесторів, й ми б змогли відкрити стартап! А з таким критерієм ми навряд чи доб'ємося цього. Не відхилити гіпотезу H_0, коли вона неправильна — це теж помилка. Вона називається **хибно негативна** (false negative, **FN**), оскільки критерій повернув 0 помилково.

$$\text{FP} - H_0 \text{відхиляється, коли вона вірна}$$

$$\text{FN} - H_0 \text{не відхиляється, коли вона не вірна}$$

У нашому завданні інвесторам важливіше хибно позитивна помилка. Їм дуже не хочеться потрапити в ситуацію, коли їм показали доказ успішності бізнесу, а виявилося, більшість користувачів відмовляється оформлювати підписку й компанія не отримує прибуток. Це призведе до збитків. Хибно негативна помилка призведе до того, що ви втратите успішний бізнес, але інвестори грошей не втратять.

Тому виберемо поріг, щоб ймовірність хибно позитивної помилки була задовільною, або ж **частота хибнопозитивних спрацьовувань** (False Positive Rate, FPR). Для цього треба зрозуміти, як часто ми будемо відхиляти гіпотезу, за умови вірності H_0.

Тепер знову переформулюємо основне питання, повністю з використанням нових термінів, й врешті-решт відповімо на нього.

Який FPR у критерію S для перевірки гіпотези H_0 проти H_1?

Коли H_0 є вірною, щоб порахувати кількість успіхів ми проводили 30 разів підкидання монетки з ймовірністю орла 0.5. Кількість орлів (тобто успіхів) у такому експерименті має розподіл, який називається біноміальним, тобто при $\mu = 0.5$ наша статистика має біноміальний розподіл $Q \sim Binom(0.5, 30)$.

Обчислимо FPR для $C = 21$

$$
\begin{aligned}
FPR &= P(S(\xi) = 1 \mid H_0) \\
&= P(Q \geqslant 21 \mid H_0) \\
&= P(Q \geqslant 21 \mid \mu = 0.5) = \\
&= P(Q \geqslant 21 \mid Q \sim Binom(0.5, 30))
\end{aligned}
$$

Це вже ймовірність події за конкретного розподілу випадкової величини. Його можна подивитися за таблицею або, що зручніше, обчислити з використанням мов програмування.

1.2.4 Обчислення FPR

Давайте порахуємо суму ймовірностей для кількостей успіхів від 21 до 30 включно. Покажемо графічно, як це виглядає на Рисунку 1.2.

```
x = np.arange(0, n + 1)                                                    ①
y = binom.pmf(x, n, 0.5)                                                   ②

plt.bar(x, y, color=turquoise)                                             ③
plt.bar(x[x >= crit_subs], y[x >= crit_subs], color=red_pink)              ④
for i in range(crit_subs - 2, crit_subs + 4):                              ⑤
    plt.text(i + 0.5, y[i] + 0.001, f"{round(y[i] * 100, 1)}%",
    ha='center', va='bottom', size=8, rotation = 30)
plt.xlabel("Кількість успішних випадків")
plt.ylabel("Ймовірність")
plt.show()
```

① Створюємо масив з усіма можливими значеннями кількості успішних випадків.
② Обчислюємо ймовірності для кожної кількості успішних випадків.
③ Створюємо гістограму з ймовірностями.
④ Виділяємо ймовірності для кількості успішних випадків, яка є більшою або рівною 21.
⑤ Додаємо текст до гістограми.

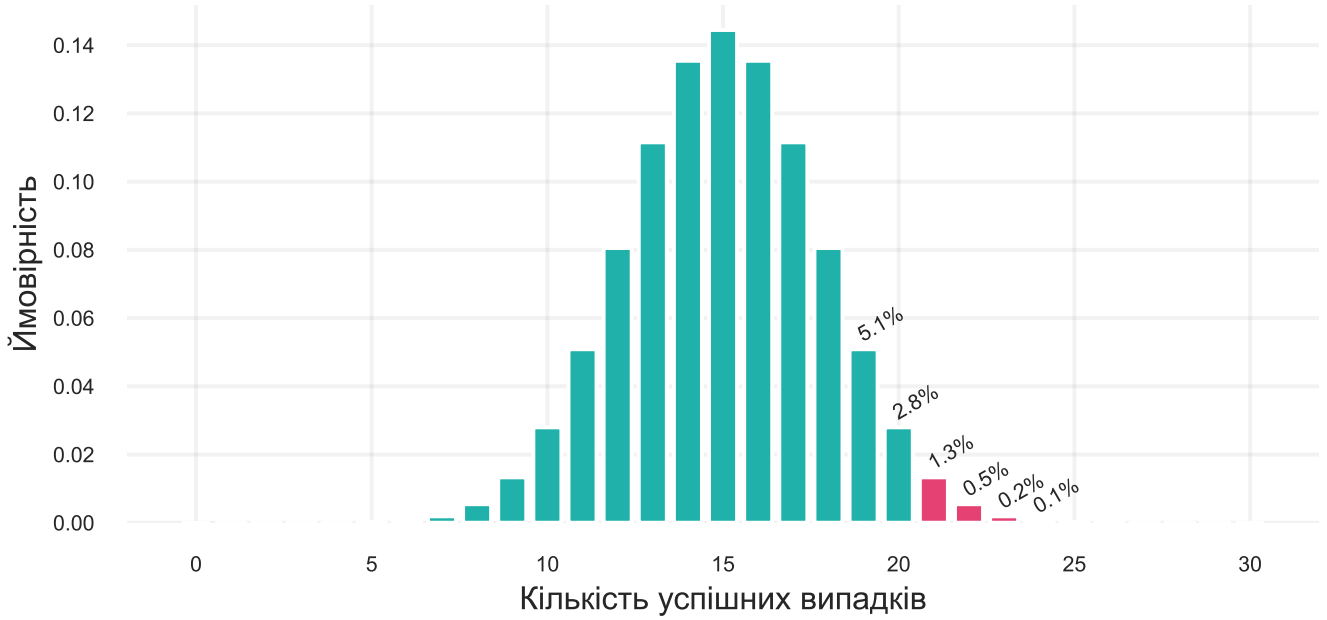


Рисунок 1.2: Ймовірність хибно відхилити H_0 за умови її вірності

Залишається лише обчислити суму ймовірностей для кількостей успіхів від 21 до 30 включно. Це і буде нашим FPR.

$$FPR_{21} = \sum_{i=21}^{30} P(Q = i) \approx 0.021$$

У нашому випадку це буде 2.1%. Якщо FPR не перевищує деякої константи α, то критерій називається критерієм **рівня значущості** α. Статистичний критерій з α = 100% створити тривіально — достатньо завжди відхиляти H_0 — тому така постановка не має сенсу.

Рівень значущості зазвичай обирають на основі бізнес-міркувань. Він позначає те, який ризик неправильного прийняття позитивного рішення ми вважаємо прийнятним. Зазвичай беруть $\alpha = 0.05$, але якщо потрібне більш точне ухвалення рішення, можуть вибрати 0.01, 0.005, 0.001. Якщо ж рішення не таке критичне, можуть вибрати 0.1.

Припустимо, вибрали значення $\alpha = 0.05$, скористаємося критерієм S: тобто якщо кількість успішних випадків перевищує або дорівнює 21, то відхиляємо H_0.

Якщо уважно подивитись на Рисунок 1.2, то можна помітити, що ми можемо відхиляти H_0 при кількості успіхів від 20, а не 21, оскільки такий все ще буде відповідати $\alpha = 0.05$:

$$FPR_{20} = \sum_{i=20}^{30} P(Q = i) \approx 0.049$$

Якщо ж обрати 19, то FPR буде більше α:

$$FPR_{19} = \sum_{i=20}^{30} P(Q = i) \approx 0.1002$$

1.3 Статистичні функції в Python

У цій частині подивимося, як вивести те, що ми отримали в частині 2, за допомогою Python. А також зрозуміємо, як знайти відповідне C за допомогою Python.

1.3.1 Біноміальний розподіл

Ми з'ясували, що статистика Q має біноміальний розподіл.

Біноміальний розподіл $Binom(n, \mu)$ — розподіл кількості успіхів у послідовності з n незалежних випадкових експериментів, ймовірність успіху в кожному з яких дорівнює μ.

Щоб працювати з розподілом, можна створити об'єкт-розподіл за допомогою бібліотеки `scipy.stats`.

```
from scipy.stats import binom

n = 30                                                                    ①
mu = 0.5                                                                  ②

binom_dist = binom(n, mu)                                                 ③
```

① Кількість спостережень.
② Ймовірність успіху.
③ Створюємо об'єкт біноміального розподілу.

1.3.2 Функція ймовірностей

Функція ймовірності дискретного розподілу $p_\xi(x)$ — ймовірність, з якою ξ приймає значення x.

У Python це функція `pmf` (probability mass function).

```
result = binom_dist.pmf(20)
print(f"Ймовірність отримати 20 успішних випадків: {result:.4f}")
```

```
Ймовірність отримати 20 успішних випадків: 0.0280
```

Зобразимо розподіл статистики Q за справедливості H_0 на графіку. Для цього можна передати відразу масив точок, для яких треба розрахувати ймовірність.

```
x = np.arange(0, n + 1)                                                    ①
y = binom_dist.pmf(x)                                                      ②

crit_subs = 21                                                             ③

plt.bar(x, y, color=turquoise, label="Ймовірність успіхів")                ④
plt.bar(x[x >= crit_subs], y[x >= crit_subs],                              ⑤
        color=red_pink, label="Критичне значення")
plt.xlabel("Кількість успішних випадків")
plt.ylabel("Ймовірність")
plt.show()
```

① Масив точок.
② Розрахунок ймовірностей.
③ Критичне значення.
④ Ймовірність успіхів.
⑤ Критичне значення.

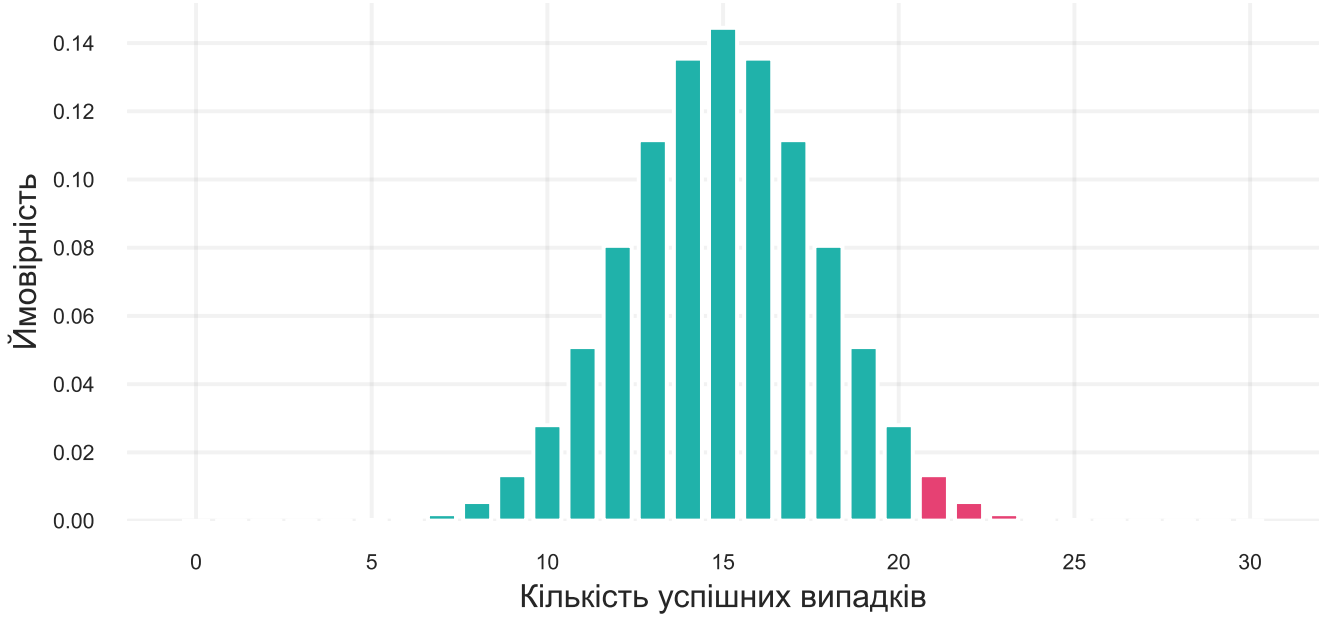


Рисунок 1.3: Функція щільності ймовірностей біноміального розподілу

Насправді вже зараз ми можемо порахувати ймовірність потрапляння в критичну область. Потрібно просто підсумувати ймовірності для кількостей успіхів від 21 до 30.

```
result = np.sum(y[crit_subs:])
print(f"Ймовірність потрапляння в критичну область: {result:.4f}")
```

```
Ймовірність потрапляння в критичну область: 0.0214
```

Отже, ми дійсно побудували критерій рівня значущості $\alpha = 0.05$. Ба більше, це критерій рівня значущості 0.021.

А що якби ми взяли $C = 19$?

```
crit_subs = 19
result = np.sum(y[crit_subs:])
print(f"Ймовірність потрапляння в критичну область: {result:.4f}")
```

```
Ймовірність потрапляння в критичну область: 0.1002
```

Тоді ймовірність помилки вже навіть більше 10%, що зовсім нам не підходить.

А якщо $C = 20$?

```
crit_subs = 20
result = np.sum(y[crit_subs:])
print(f"Ймовірність потрапляння в критичну область: {result:.4f}")
```

```
Ймовірність потрапляння в критичну область: 0.0494
```

Видно, що немає такого C, щоб FPR був рівно 5%.

1.3.3 Кумулятивна функція розподілу

Кумулятивна функція розподілу задається формулою:

$$F_\xi(x) = P(\xi \leqslant x) \tag{1.1}$$

У Python це метод `cdf()` (Cumulative Distribution Function).

```
result = binom_dist.cdf(19)
print(f"Ймовірність отримати 19 або менше успішних випадків: {result:.4f}")
```

```
Ймовірність отримати 19 або менше успішних випадків: 0.9506
```

А оскільки $P(\xi \leqslant 19) + P(\xi \geqslant 20) = 1$, можемо обчислити рівень значущості нашого критерію.

```
result = 1 - binom_dist.cdf(19)
print(f"Ймовірність потрапляння в критичну область: {result:.4f}")
```

```
Ймовірність потрапляння в критичну область: 0.0494
```

1.3.4 Квантиль

Щоб вибрати критичну область для критерію, ми хотіли б знайти точку, площа стовпців праворуч від якої була б 5%. Тобто площа стовпців зліва — 95%. Така точка називається *квантилью*.

$$u_p(\xi) = \{x \,|F_\xi(x) = p\} \tag{1.2}$$

Але при $p = 0.95$ й нашому біноміальному розподілі, такої точки немає. Ми з'ясували, що є точка, праворуч від якої площа 0.494, а в наступної вже 0.1. Щоб визначити квантиль у цьому випадку, модифікуємо визначення. Квантиль $u_p(\xi)$ — величина, яку ξ не перевищує з імовірністю хоча б p. Тобто $F_\xi(u_p) \geqslant p$.

$$u_p(\xi) = min\,\{x \mid F_\xi(x) \geqslant p\} \tag{1.3}$$

Приклад 1.1. Для величини $\xi \sim Bin(30, 0.5)$ порахуємо 0.95-квантиль. Вирішимо задачу просто підбором.

$$P(\xi \leqslant 18) \approx 0.90$$

$$P(\xi \leqslant 19) \approx 0.951$$

$$P(\xi \leqslant 20) \approx 0.97$$

Бачимо, що 18 нам ще не підходить, а 19 й більші значення вже підійдуть. У них функція розподілу буде більшою за p. Відповідь — найменше відповідне значення, тобто 19. При цьому немає точки, де функція розподілу дорівнювала б p в точності.

Якби розподіл був неперервним, можна було б сказати, що квантиль — це таке x, на якому функція розподілу дорівнює p. Але для дискретного розподілу такого може не бути.

У Python квантиль можна порахувати через методу `ppf()` (Percent Point Function).

```
result = binom_dist.ppf(0.95)
print(f"0.95-квантиль: {result}")
```

```
0.95-квантиль: 19.0
```

Як тепер підібрати C для будь-яких n, μ й для будь-якого рівня значущості α?

1. Потрібно знайти C, таке що $P(Q \geqslant C) \leqslant \alpha$
2. Тобто потрібно $P(Q < C) \geqslant 1 - \alpha$
3. Q приймає тільки цілі значення: $P(Q \leqslant C - 1) \geqslant 1 - \alpha$, або $F(C - 1) \geqslant 1 - \alpha$
4. Отже, з визначення квантилі, $C - 1 = u_{1-\alpha}$
5. Значить $C = u_{1-\alpha} + 1$

```
def find_crit_subs(n, mu, alpha):                                    ①
    binom_dist = binom(n, mu)                                        ②
    return binom_dist.ppf(1 - alpha) + 1                             ③


result = find_crit_subs(30, 0.5, 0.05)
print(f"Критичне значення для n = 30, mu = 0.5, alpha = 0.05: {result}")
```

① Функція, що приймає кількість спостережень, ймовірність успіху та рівень значущості.
② Створюємо об'єкт біноміального розподілу.
③ Повертаємо критичне значення.

```
Критичне значення для n = 30, mu = 0.5, alpha = 0.05: 20.0
```

Критичне значення 20, отже підсумковий критерій має такий вигляд:

$$S = \{Q \geqslant 20\}$$

$Q = 19$, значить гіпотезу ми не відкидаємо.

При цьому нам вдалося побудувати процес, за яким ми ухвалюємо рішення для будь-якого рівня значущості та значення статистики критерію.

1.4 p-значення

Зауважимо, що зараз, якщо нам зададуть іншу α, нам доведеться перебудовувати критерій заново. Це не зовсім зручно. У статистиці є механізм *p-значення*, який дає змогу прийняти рішення для всіх α відразу.

1.4.1 Більш екстремальні значення

Припустимо, ми провели експеримент й порахували для критерію його статистику $Q(\xi)$. Позначимо отримане значення q, у поточній задачі це $q = 19$. Якби кількість успішних підписок була більшою, це б сильніше свідчило на користь альтернативної гіпотези H_1. Тобто в разі значення 25 ми були б ще сильніше впевнені в тому, що наш бізнес буде окупатися. Тоді значення 25 називається *більш екстремальним*, ніж значення 19. У нашій задачі більш екстремальним із двох значень є те, яке більше.

Визначимо поняття екстремальності формально:

$$S = \{Q(\xi) \geqslant C\} : \ t \text{ екстремальніше } q \Leftrightarrow t > q \tag{1.4}$$

Найчастіше критерії інших видів можна привести до цього, тоді для них теж визначено поняття екстремальності.

1.4.2 p-значення

p-value — це ймовірність отримати таке або більш екстремальне значення статистики q за умови вірності H_0.

$$P_{H_0}(Q \geqslant q) \tag{1.5}$$

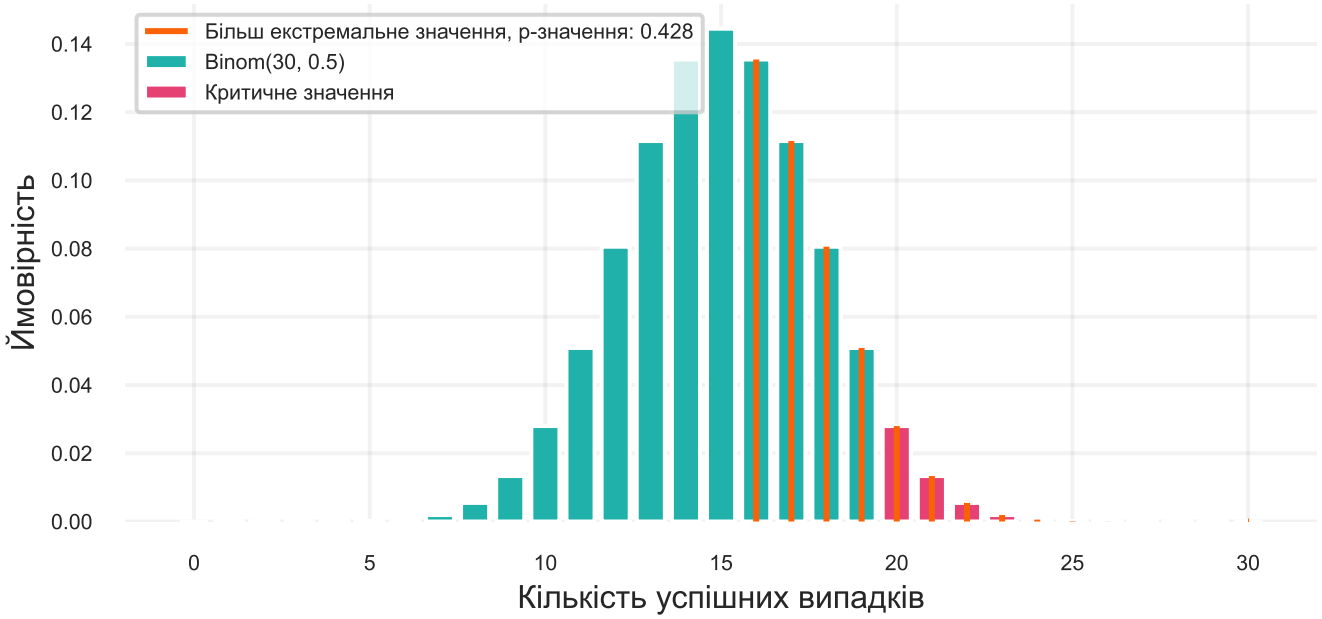


Рисунок 1.4: p-значення для критерію $Q = 15$

Тепер виведемо формулу через функції Python:

$$P_{H_0}(Q \geqslant q) = 1 - P_{H_0}(Q < q) = 1 - F(q) \tag{1.6}$$

Зобразимо на графіку область більш екстремальних значень й p-value для різних значень статистики.

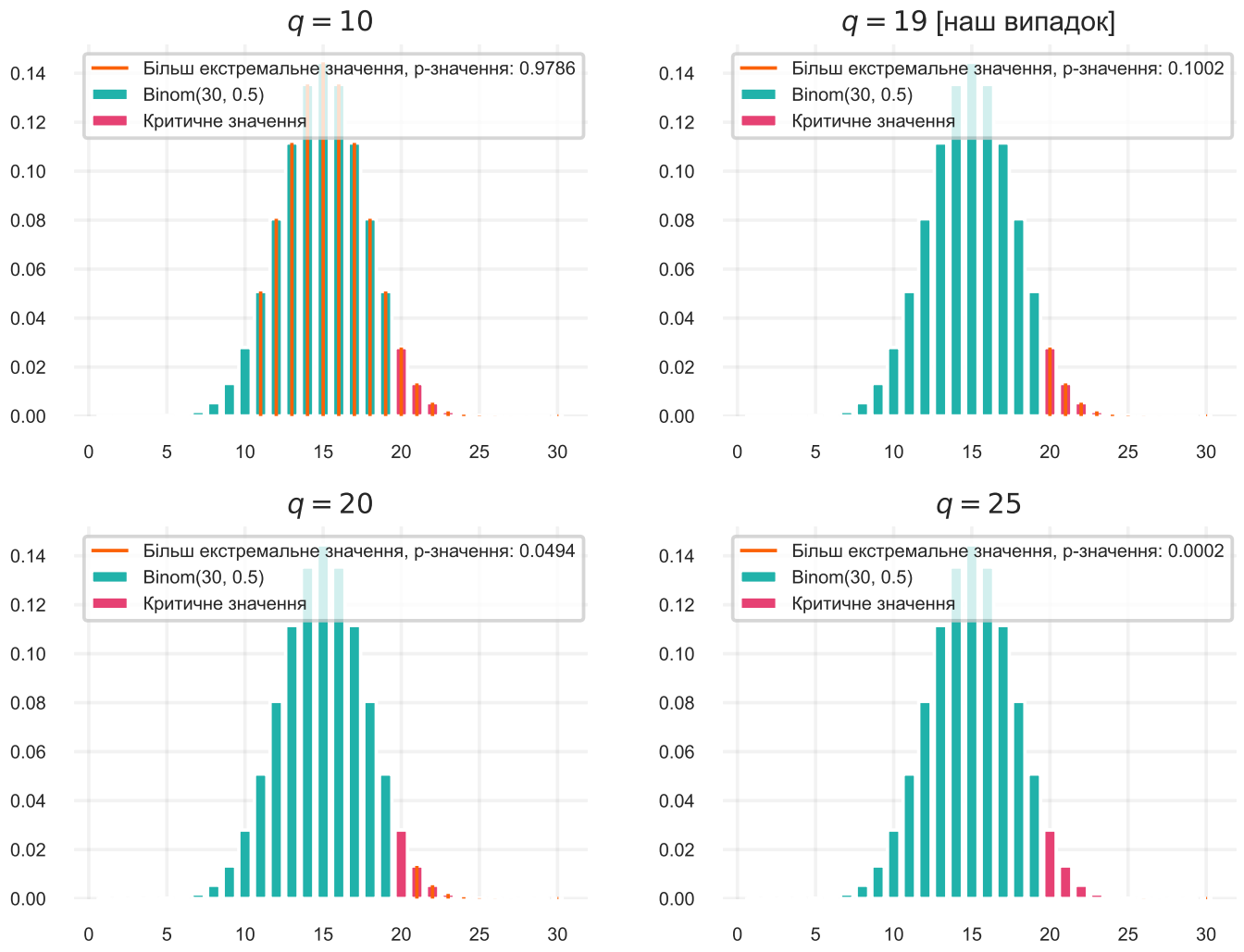


Рисунок 1.5: p-значення для критерію $Q = 10, 19, 20, 25$

Можна побачити, що в критичній області p-значення $\leqslant \alpha$, а поза нею p-значення $> \alpha$. Саме таке правило й використовується для прийняття рішення.

$$H_0 \text{ відкидається } \Leftrightarrow p\text{-значення} \leqslant \alpha \tag{1.7}$$

Причому за p-значення одразу видно, що якби в нашу критичну область включили значення 19, наш критерій допускав би FPR у 10% випадків, що вже неприпустимо. Тому й гіпотезу ми не відкидаємо.

Зауважимо, що для обчислення p-значення не знадобилося знання α, а потрібна була тільки статистика й форма критерію.

1.5 Двосторонні критерії

До цього моменту нас цікавили відхилення від ймовірності в 50% тільки в один бік. І логічно, адже це продиктовано бізнесом. Тільки велика частка успішних підписок призведе до успіху. І зазвичай при прийнятті рішень так й буває. **При тестуванні нового рішення або продукту розглядають альтернативну гіпотезу тільки в бік поліпшення**, тому що в іншому разі немає сенсу впроваджувати рішення на всіх користувачів.

Однак **іноді** може знадобитися доводити відхилення в обидва боки, якщо ви перевіряєте якесь припущення. Нехай вам дали монетку й просять перевірити, чесна вона чи ні. Монетка чесна, якщо під час підкидання ймовірність випадання орла дорівнює 0.5. Ви підкидаєте монетку 30 разів, кожен кидок — бернуллівська величина, аналогічно завданню з сервісом освітніх послуг. Нульова гіпотеза та ж сама: $\mu = 0.5$. Але тепер ми хочемо відкидати цю гіпотезу як у разі великої ймовірності орла, так і в разі маленької, відповідно перевіряємо *двосторонню гіпотезу*.

$$H_0 : \mu = 0.5$$

$$H_1 : \mu \neq 0.5$$

Виберемо критичну область для критерію за такої альтернативи. Скористаємося тією ж статистикою $Q(\xi) = \sum \xi_i$. Тільки тепер відхилення в кожну сторону однаково важливі. Відкидати гіпотезу будемо не тільки на досить великих значеннях, а й на досить маленьких. Наприклад, якщо у нас було всього 2 орла з 30 — це свідчення на користь того, що $\mu \neq 0.5$, але не на користь $\mu > 0.5$.

Оскільки відхилення в різні боки однаково важливі, а розподіл симетричний, шукати критерій можна в такому вигляді:

$$S = \{Q \geqslant C\} \cup \{Q \leqslant n - C\} \tag{1.8}$$

1.5.1 Як вибрати критичну область

Подивимося, який вигляд матиме критична область у такому разі.

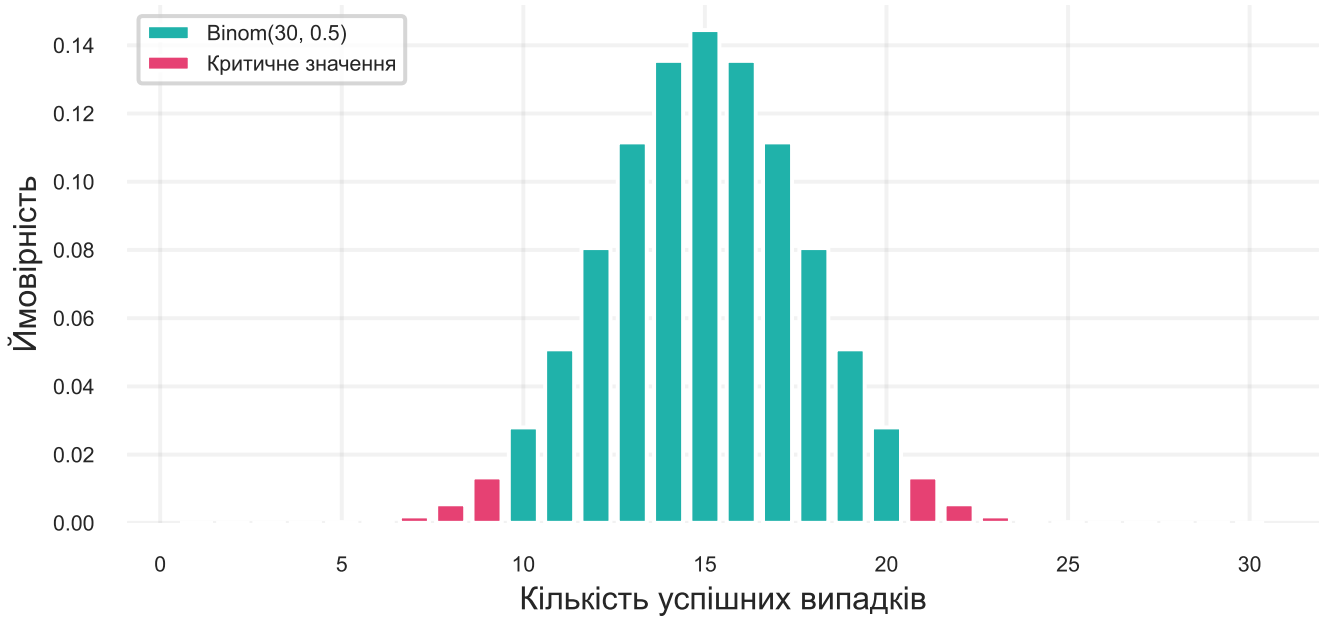


Рисунок 1.6: Двостороння критична область для критерію $= 6$

З картинки видно, що якщо тепер відкидати відхилення за $Q \geqslant 20$, то необхідно відкидати й $Q \leqslant 10$, а отже, загальна площа стовпців буде вже приблизно 0.1. Тому за рівня значущості 0.05 й 20 успіхів гіпотеза вже не відкинеться.

Якщо ж виставити $C = 6$, то така область уже підходить, площа стовпців $\approx 0.043 < 0.05$.

Щоб вибрати порогову константу за формулою, можна помітити, що критична область симетрична, а значить праворуч площа не повинна бути більшою, ніж $\frac{\alpha}{2}$. А таку задачу ми вже вміємо розв'язувати.

Реалізуємо функцію на Python.

```
def find_crit_subs_two_sided(n, mu, alpha):                    ①
    binom_dist = binom(n, mu)                                  ②
    return n / 2 - binom_dist.ppf(alpha / 2) + 1               ③

result = find_crit_subs_two_sided(30, 0.5, 0.05)
print(f"Критичне значення для n = 30, mu = 0.5, alpha = 0.05: {result}")
```

① Функція для знаходження критичного значення:
② Створюємо об'єкт біноміального розподілу.
③ Знаходимо критичне значення.

```
Критичне значення для n = 30, mu = 0.5, alpha = 0.05: 6.0
```

1.5.2 Як знайти p-значення

Критерій має вигляд

$$S = \{|Q(\xi) - 15| \geqslant C\}$$

Позначимо відхилення суми від 15 як $\Delta(\xi) = |Q(\xi) - 15|$, тоді ми маємо критерій

$$S = \{\Delta(\xi) \geqslant C\} \tag{1.9}$$

Тобто більш екстремальними вважатимуться ті значення суми, що знаходяться далі від 15. Щоб обчислити p-значення, доведеться порахувати суму площ із двох сторін окремо.

```
def pvalue_two_sided_sym(n, q):                                ①
    binom_h0 = binom(n=n, p=0.5)                               ②
    diff = np.abs(q - 15)                                      ③
    right_sq = 1 - binom_h0.cdf(15 + diff - 1)                 ④
    left_sq = binom_h0.cdf(15 - diff)                          ⑤
    return left_sq + right_sq                                  ⑥

result = pvalue_two_sided_sym(30, 21)
print(f"p-значення для q = 21: {result:.4f}")
```

① Функція для обчислення p-значення:
② Створюємо об'єкт біноміального розподілу.
③ Обчислюємо відхилення від 15.
④ Обчислюємо праву площу.
⑤ Обчислюємо ліву площу.
⑥ Повертаємо суму площ.

```
p-значення для q = 21: 0.0428
```

Насправді через симетричність розподілу ліва й права площа виходять однаковими, тому можна порахувати площу з одного боку й помножити на 2.

```
def pvalue_two_sided_sym_simple(n, q):
    binom_h0 = binom(n=n, p=0.5)
    diff = np.abs(q - 15)
    right_sq = 1 - binom_h0.cdf(15 + diff - 1)
    return 2 * right_sq
```

```
result = pvalue_two_sided_sym_simple(30, 21)
print(f"p-значення для q = 21: {result:.4f}")
```

```
p-значення для q = 21: 0.0428
```

Тепер навіть у разі 20 орлів p-значення > 0.05, тому відкидати будемо значення, починаючи з 21 й менші або такі, що дорівнюють 9.

1.5.3 Випадок із несиметричним розподілом

Коли розподіл за справедливості H_0 несиметричний, відхилення від очікуваного значення в різні боки можуть бути по-різному критичними. Як приклад розглянемо також біноміальний розподіл, але з імовірністю успіху 0.8.

Тоді можна ліву і праву критичні області побудувати окремо, виділивши на них по $\frac{\alpha}{2}$ площі. Праву область ми вже вміємо шукати, знайдемо ліву.

```
binom_h0_nonsym = binom(n=30, p=0.8)                                    ①
probs = binom_h0_nonsym.pmf(np.arange(31))                              ②

plt.bar(np.arange(31), probs, color=turquoise, label="Binom(30, 0.8)")  ③
plt.legend()
plt.show()
```

① Створюємо об'єкт біноміального розподілу з ймовірністю успіху 0.8.
② Обчислюємо ймовірності для всіх можливих значень.
③ Будуємо графік.

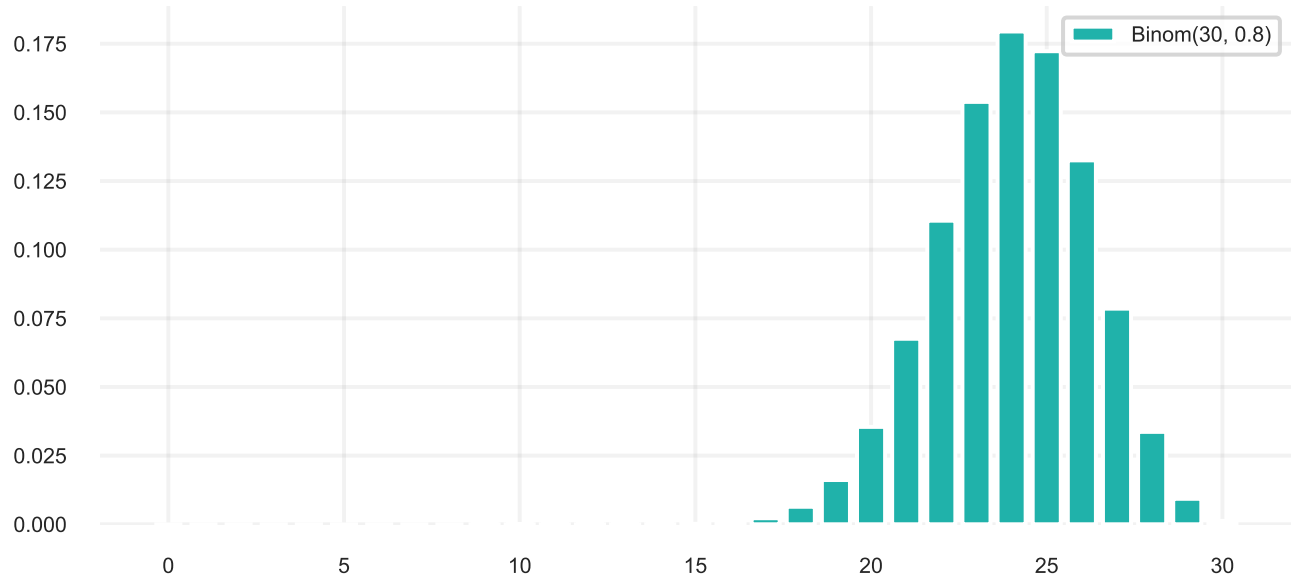


Рисунок 1.7: Біноміальний розподіл з імовірністю успіху 0.8

Для того, щоб побудувати двосторонній критерій, потрібно знайти ліворуч і праворуч області, площа яких становить не більше, ніж $\frac{\alpha}{2}$. Для правого боку ми вже розв'язували таку задачу, розв'яжемо для лівого.

Шукаємо C, таке що

$$P(Q(\xi) \leqslant C) \leqslant \frac{\alpha}{2} \tag{1.10}$$

Спочатку знайдемо перше число, де ймовірність $\geqslant \frac{\alpha}{2}$. А це за визначенням $\frac{\alpha}{2}$-квантиль. Достатньо взяти попереднє число, і воно буде задовольняти нашій умові.

```
def two_sided_criterion_nonsym(n, mu, alpha):

    binom_h0 = binom(n=n, p=mu)
    c2 = binom_h0.ppf(1 - alpha/2) + 1
    c1 = binom_h0.ppf(alpha/2) - 1
    return c1, c2

result = two_sided_criterion_nonsym(30, 0.8, 0.05)
print(f"Критичні значення для n = 30, mu = 0.8, alpha = 0.05: {result}")
```

```
Критичні значення для n = 30, mu = 0.8, alpha = 0.05: (18.0, 29.0)
```

Отже, наш критерій для перевірки гіпотези

$$H_0 : \mu = 0.8$$

$$H_1 : \mu \neq 0.8$$

має вигляд

$$S = \{Q(\xi) \leqslant 18\} \cup \{Q(\xi) \geqslant 29\}$$

Тут межа 29 уже має логічний вигляд, бо треба спростувати 80% орлів/успіхів, а для цього потрібна велика їхня кількість.

Зобразимо критичну область на графіку.

```
C1, C2 = two_sided_criterion_nonsym(30, 0.8, 0.05)                    ①

x = np.arange(31)                                                     ②

plt.bar(x, probs, color=turquoise, label="Binom(30, 0.8)")            ③
plt.bar(x[x <= C1], probs[x <= C1],                                   ④
        color=red_pink, label="Критичне значення")
plt.bar(x[x >= C2], probs[x >= C2],                                   ⑤
        color=red_pink)

plt.xlabel("Кількість успішних випадків")
plt.ylabel("Ймовірність")
plt.legend(loc = 'upper left')
plt.show()
```

① Знаходимо критичні значення
② Створюємо масив значень для графіка.
③ Будуємо графік біноміального розподілу.
④ Відзначаємо ліву критичну область.
⑤ Відзначаємо праву критичну область.

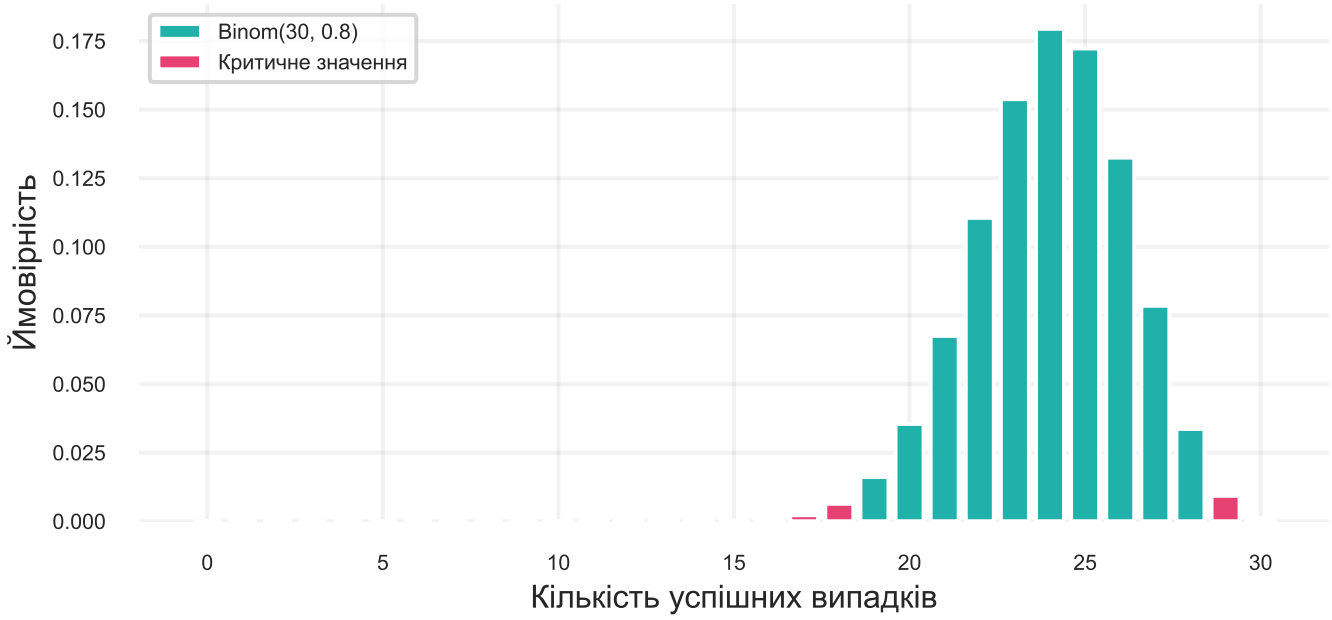


Рисунок 1.8: Двостороння критична область для критерію $C_1 = 18, C_2 = 29$

1.5.4 p-значення для несиметричного розподілу

Цей критерій — об'єднання двох критеріїв рівня значущості $\frac{\alpha}{2}$, для кожного з яких можна порахувати p-значення. Позначимо їх як p_1, p_2. Перший критерій відкидається при $p_1 \leqslant \frac{\alpha}{2}$, другий при $p_2 \leqslant \frac{\alpha}{2}$. А наш об'єднаний, коли виконано одну з цих умов, тобто

$$2p_1 \leqslant \alpha \vee 2p_2 \leqslant \alpha \Leftrightarrow 2 \cdot \min(p_1, p_2) \leqslant \alpha \tag{1.11}$$

Отже, можна рахувати p-значення як $2\min(p_1, p_2)$ й порівнювати з α.

Проведемо аналогію із симетричним випадком: якщо сума опинилася в лівій частині, то потрібно порахувати p-значення лівого критерію і помножити на 2. Якщо сума опинилася в правій частині, то потрібно порахувати p-значення правого критерію і помножити на 2.

```python
def pvalue_two_sided(n, q, mu=0.5):                    ①
    binom_h0 = binom(n=n, p=mu)                        ②
    pvalue_left = binom_h0.cdf(q)                      ③
    pvalue_right = 1 - binom_h0.cdf(q - 1)             ④
    return 2 * min(pvalue_left, pvalue_right)          ⑤

result = pvalue_two_sided(30, 28, 0.8)
print(f"p-значення для q = 28: {result:.4f}")
```

① Функція для обчислення p-значення:
② Створюємо об'єкт біноміального розподілу.
③ Обчислюємо p-значення для лівого критерію.
④ Обчислюємо p-значення для правого критерію.
⑤ Повертаємо p-значення.

```
p-значення для q = 28: 0.0884
```

Видно, що p-значення > 0.05, отже, на рівні значущості 0.05 навіть 28 успіхів недостатньо, щоб відкинути ймовірність успіху в 80%.

Зауважимо, що ця ж функція працює і для симетричного випадку, повертаючи той самий результат.

```
result = pvalue_two_sided(n=30, q=20, mu=0.5)
print(f"p-значення для q = 20: {result:.4f}")
```

```
p-значення для q = 20: 0.0987
```

```
result = pvalue_two_sided_sym(n=30, q=20)
print(f"p-значення для q = 20: {result:.4f}")
```

```
p-значення для q = 20: 0.0987
```

1.6 Готові функції

Звісно, можна використати готові функції з бібліотеки `scipy`. Для цього використаємо функцію `binomtest`, котра має параметри:

- `k` — кількість успіхів
- `n` — кількість спостережень
- `p` — ймовірність успіху
- `alternative` — тип гіпотези:
  - `two-sided`: двостороння
  - `greater`: правостороння
  - `less`: лівостороння

```
from scipy.stats import binomtest                                    ①

result = binomtest(19, 30, 0.5, alternative='two-sided')             ②

print(f"Статистика: {result.statistic:.2f}")                         ③
print(f"p-значення: {result.pvalue:.4f}")
```

① Імпортуємо функцію `binomtest`.
② Викликаємо функцію з параметрами:
③ Виводимо статистику та p-значення.

```
Статистика: 0.63
p-значення: 0.2005
```

1.7 Питання для самоперевірки

Загальні поняття та постановка задачі:

1. Що таке генеральна сукупність та вибірка в контексті наведеного прикладу з онлайн-курсами?
2. Що позначають символи μ та $\hat{\mu}$? Яка між ними різниця?
3. Чому спостережувана частка успіхів у вибірці ($\hat{\mu}$) може не точно відображати справжню частку (μ) в генеральній сукупності?
4. Який розподіл ймовірностей описує результат для одного студента (успіх/невдача)?
5. Який розподіл ймовірностей описує загальну кількість успішних випадків серед n студентів? Які параметри має цей розподіл?

Статистичні гіпотези та критерій:

6. Що таке нульова (H_0) та альтернативна (H_1) гіпотези? Сформулюйте їх для прикладу з онлайн-курсами.
7. Що таке статистичний критерій? Яка його мета?
8. Що таке статистика критерію (Q) та критичне значення (C) у контексті біноміального тесту?
9. Що таке критична область критерію?

Помилки та рівень значущості:

10. Які два типи помилок можливі при перевірці статистичних гіпотез? Опишіть їх (хибно позитивна та хибно негативна). Яка помилка була важливішою для інвесторів у прикладі?
11. Що таке рівень хибнопозитивних спрацьовувань (FPR)? Як він пов'язаний з нульовою гіпотезою?
12. Що таке рівень значущості (α)? Як він використовується для вибору критичного значення (C)?
13. Як розрахувати FPR для заданого критичного значення C, використовуючи функцію щільності ймовірностей (PMF) біноміального розподілу?

Обчислення в Python (`scipy.stats`):

14. Які функції з `scipy.stats.binom` використовуються для обчислення:
    - Ймовірності конкретної кількості успіхів ($P(Q = k)$)?
    - Кумулятивної ймовірності ($P(Q \leq k)$)?
    - Квантиля розподілу?
15. Як можна знайти критичне значення C для правостороннього тесту ($H_1 : \mu > \mu_0$) з заданим рівнем значущості α, використовуючи функцію квантиля (`ppf`)?

P-значення:

16. Що таке p-значення (p-value)? Як воно інтерпретується?
17. Як визначається поняття "більш екстремального" значення статистики для правостороннього тесту?
18. Як розраховується p-значення для правостороннього біноміального тесту, знаючи спостережувану кількість успіхів q?
19. Яке правило прийняття рішення щодо H_0 використовується на основі p-значення та рівня значущості α?
20. Яка перевага використання p-значення порівняно з визначенням критичної області?

Двосторонні критерії:

21. Коли використовують двосторонні критерії? Як формулюються гіпотези (H_0 та H_1) для двостороннього тесту?
22. Як зазвичай визначається критична область для двостороннього критерію при симетричному розподілі (наприклад, $Binom(n, 0.5)$)? Як рівень значущості α розподіляється між "хвостами"?
23. Як розраховується p-значення для двостороннього критерію при симетричному розподілі?
24. Як визначається критична область для двостороннього критерію, коли розподіл статистики при H_0 є несиметричним (наприклад, $Binom(n, 0.8)$)?
25. Як розраховується p-значення для двостороннього критерію при несиметричному розподілі?

Готові функції:

26. Як можна виконати біноміальний тест (односторонній або двосторонній) за допомогою функції `binomtest` з бібліотеки `scipy.stats` ? Які основні параметри вона приймає і що повертає?

Chapter 2

Статистична потужність, ефект та довірчі інтервали

2.1 Статистична потужність

2.1.1 Хибно негативні помилки

Раніше під час побудови критеріїв ми звертали увагу тільки на α, рівень значущості критерію. Але цей параметр контролює лише хибнопозитивну помилку (False Positive), а саме ймовірність, що критерій прийме H_1 за умови вірності H_0.

Але є ще один вид помилок, які може допустити критерій — хибно негативні помилки (False Negative). Це випадки, коли критерій приймає H_0 за умови вірності H_1. Це важливо, оскільки вони можуть вказувати на те, що критерій не чутливий до змін, які відбуваються в даних.

Випадок, коли ймовірність FPR $< \alpha$, але при цьому ймовірність хибно негативні помилки (False Negative Rate, FNR) величезна, можна навести легко. Для цього достатньо **ніколи** не відкидати гіпотезу, взявши критерій $S \equiv 0$.

Наведемо приклад, коли помилки False Negative відбуваються не завжди, але критерії є все одно нечутливими.

2.1.2 Критерій пори року

Поставимо гіпотезу про те, що зараз на вулиці літо. Для перевірки можна було б, звісно, подивитися в календар, але ми зробимо інакше.

$$H_0: \text{ на вулиці літо}$$

$$H_1: \text{ на вулиці не літо}$$

Подивимося у вікно і визначимо, чи йде там сніг. Якщо він йде, то це непоганий доказ того, що зараз не літо, а отже можна відкинути H_0.

Порахуємо FPR та FNR для цього критерію. Ми знаємо, що влітку сніг іде дуже рідко (ймовірність помилки нижча за 0.1%), тож це точно критерій рівня значущості 0.001, чого зазвичай достатньо для критеріїв.

$$\text{FPR}(S) = P(\text{йде сніг} \mid \text{сьогодні літо}) < 0.001$$

Але що з FNR? Розглянемо конкретний випадок: зараз вересень. Оскільки у вересні майже завжди немає снігу, можна сказати, що FNR більша за 90%, отже, цей критерій насправді мало дієвий.

$$\text{FPR}(S) = P(\text{не йде сніг} \mid \text{зараз вересень}) > 0.9$$

Сформулюємо інший критерій рівня значущості α, причому в цьому разі рівень значущості можна вибрати довільним.

$$S(\xi) = \begin{cases} 1, \text{ якщо монетка з імовірністю орла } \alpha \text{ випала орлом} \\ 0, \text{ інакше} \end{cases}$$

Виходить, цей критерій випадковий, і він не використовує взагалі жодної інформації про погоду. Однак вимогам до рівня значущості він задовольняє.

$$\text{FPR} = P(\text{випав орел} \mid \text{сьогодні літо}) = P(\text{випав орел}) = \alpha$$

Обчислимо FNR.

$$\text{FNR} = P(\text{не випав орел} \mid \text{сьогодні не літо}) = P(\text{не випав орел}) = 1 - \alpha$$

За $\alpha = 0.001$, як у першому випадку, отримуємо ймовірність FNR $0.999 > 0.9$, тобто за однакового рівня значущості з першим критерієм, другий критерій частіше припускається хибно негативної помилки.

2.1.3 Потужність

У статистиці заведено позитивним результатом вважати відкидання нульової гіпотези, бо зазвичай підтвердження альтернативи означає наявність бізнес-результату. Тому вважається хорошим критерій, який частіше дає змогу виявити бізнес-результат. І рахують тоді не ймовірність хибно негативної помилки, а *потужність*, що дорівнює ймовірності відкинути нульову гіпотезу за вірності H_1, тобто ймовірність **істинно позитивного результату** (True Positive Rate, TPR).

$$\text{Power}_S = 1 - FNR \tag{2.1}$$

Коли альтернатива H_1 складається з множини результатів, потужність розглядають як функцію від результату. Наприклад, можна порахувати потужність першого та другого критеріїв взимку й восени.

$$\text{Power}_S(\mu) = 1 - FNR(\mu) \tag{2.2}$$

Перший критерій

$$\text{Power}_S(\text{травень}) = P(\text{їде сніг} \mid \text{травень}) \approx 0.00001$$

$$\text{Power}_S(\text{жовтень}) = P(\text{їде сніг} \mid \text{жовтень}) \approx 0.1$$

$$\text{Power}_S(\text{січень}) = P(\text{їде сніг} \mid \text{січень}) \approx 0.5$$

Другий критерій

$$\text{Power}_S(\text{травень}) = P(\text{випав орел} \mid \text{травень}) = \alpha = 0.001$$

$$\text{Power}_S(\text{жовтень}) = P(\text{випав орел} \mid \text{жовтень}) = \alpha = 0.001$$

$$\text{Power}_S(\text{січень}) = P(\text{випав орел} \mid \text{січень}) = \alpha = 0.001$$

Зазвичай завдання пошуку найкращого критерію формулюється як пошук якомога потужнішого критерію за заданого рівня значущості $FPR \leqslant \alpha$. Але ми сказали, що потужність — функція від параметра, у нашому випадку від місяця.

Якщо ми застосовуватимемо критерій у січні, то потужнішим буде перший критерій, а якщо в травні, то потужнішим буде другий критерій. Тому потрібно розуміти, коли буде застосовуватися критерій, а отже, ми шукаємо найпотужніший критерій у галузі, яка нас цікавить.

Хоча в реальності в травні потужність обох критеріїв настільки низька, що вони просто не приносять користі, й використовувати їх не має сенсу.

2.2 Потужність для біноміального розподілу

Застосуємо нові знання про потужність для нашої задачі з освітнім сервісом. З бізнес-міркувань ми вже вибрали $\alpha = 0.05$, а отже, знаємо, що ми неправильно відкидаємо гіпотезу $H_0 : \mu = 0.5$ з ймовірністю не більше, ніж 5%. Тобто цим обмежена ймовірність хибно позитивної помилки.

А з якою ймовірністю ми будемо *правильно* відкидати гіпотезу? І яка в нас буде ймовірність хибно негативної помилки? На це запитання якраз відповість формула потужності.

Згадаймо критерій, за яким ми приймаємо рішення:

$$Q(\xi) = \sum_{i=1}^{n} \xi_i - \text{кількість підписок}$$

$$S = \{Q \geqslant 20\}$$

Тобто якщо отримуємо хоча б 20 успішних підписок, то відкидаємо H_0.

Зауважимо, що потужність залежить від того, яке значення μ у нашій генеральній сукупності. Зафіксуємо спочатку параметр $\mu = 0.6$ й порахуємо потужність для нього. Якщо істинний параметр такий, то статистика Q має розподіл $Binom(30, 0.6)$.

```
binom_h0 = binom(n=30, p=0.5)                                                    ①
binom_alternative = binom(n=30, p=0.6)                                           ②

x_grid = np.arange(1, 31)                                                        ③
crit_reg = x_grid >= 20                                                          ④

probs_h0 = binom_h0.pmf(x_grid)                                                  ⑤
plt.bar(x_grid, probs_h0, color=turquoise, label='PMF, $Binom(0.5, 30)$')        ⑥

probs_alternative = binom_alternative.pmf(x_grid)                                ⑦
plt.bar(x_grid, probs_alternative, color=slate, label='PMF, $Binom(0.6, 30)$')   ⑧
plt.bar(x_grid[crit_reg], probs_alternative[crit_reg], color=red_pink, label='Критична область

plt.legend()
plt.show()
```

① Розподіл нульової гіпотези
② Розподіл альтернативної гіпотези
③ Вибираємо значення для осі x
④ Критична область
⑤ Обчислюємо ймовірності для нульової гіпотези

⑥ Будуємо гістограму для нульової гіпотези
⑦ Обчислюємо ймовірності для альтернативної гіпотези
⑧ Будуємо гістограму для альтернативної гіпотези
⑨ Будуємо гістограму для критичної області

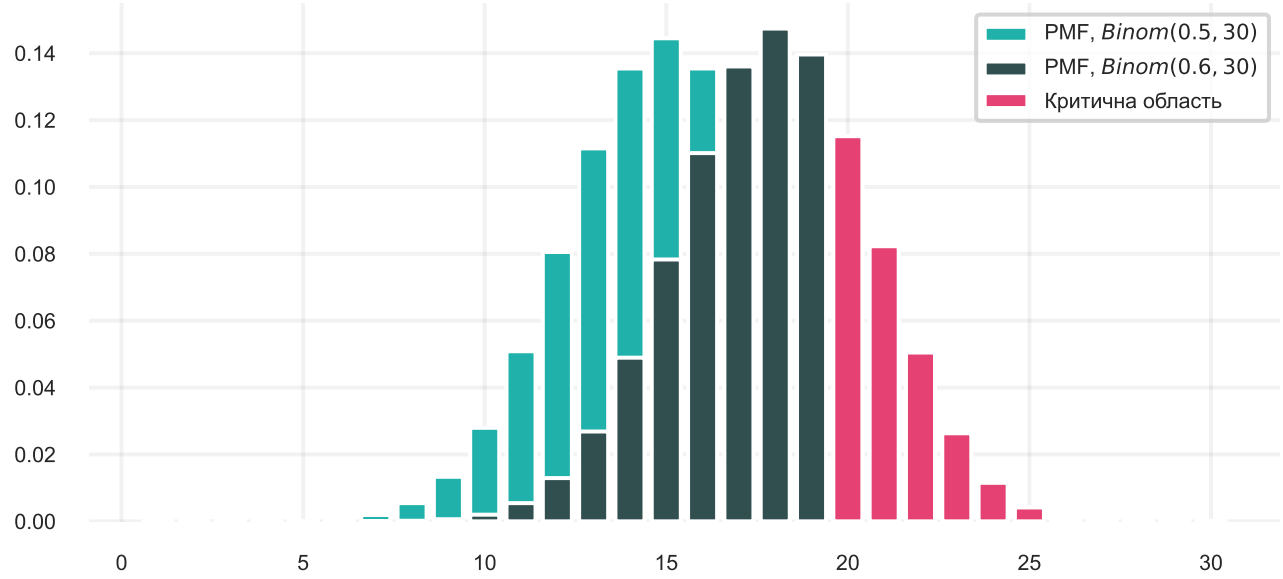


Рисунок 2.1: Потужність критерію для $\mu = 0.6$

Як і раніше, нас цікавить імовірність отримати 20 або більше успіхів. Але якщо раніше ми дивилися на неї для розподілу з $\mu = 0.5$ й хотіли, щоб вона була меншою за 5%, то тепер ми дивимося за $\mu = 0.6$ та прагнемо зробити цю величину якомога більшою. Порівняно з обчисленням FPR формула не зміниться, змінюється тільки μ

```
critical_value = 20
power = 1 - binom(n=30, p=0.6).cdf(critical_value - 1)
fpr   = 1 - binom(n=30, p=0.5).cdf(critical_value - 1)

print(f"Хибно позитивна помилка: {fpr:.1%}")
print(f"Потужність: {power:.1%}")
```

```
Хибно позитивна помилка: 4.9%
Потужність: 29.1%
```

Видно, що потужність близько 30%. Це досить маленьке значення, адже якщо наш продукт прибутковий, то ми побачимо це за допомогою нашого тесту тільки з імовірністю в 30 відсотків. Ми легко можемо пропустити ефект.

Що ж можна зробити, щоб зробити потужність вищою? Щоб розібратися, реалізуємо функцію потужності в загальному вигляді.

```
def get_stat_power(N, mu_h0, mu_alternative, alpha):                    ①

    binom_h0 = binom(n=N, p=mu_h0)                                      ②
    binom_alternative = binom(n=N, p=mu_alternative)                    ③
    critical_value = binom_h0.ppf(1 - alpha) + 1                        ④
    return 1 - binom_alternative.cdf(critical_value - 1)                ⑤

result = get_stat_power(30, 0.5, 0.6, alpha=0.05)
```

```
print(f"Потужність: {result:.1%}")
```

① Функція потужності:
② Розподіл нульової гіпотези
③ Розподіл альтернативної гіпотези
④ Критичне значення
⑤ Повертаємо потужність

```
Потужність: 29.1%
```

Коли в житті ми спостерігаємо якесь явище і бачимо його лише кілька разів, ми не впевнені в тому, що воно не випадкове. Якщо ж бачимо його досить часто, то вже складаємо закономірності. Так і в статистиці. Коли ми подивилися на 30 потенційних підписок, ми помічаємо, що частка доставок більше половини. Але ми все ще не впевнені. Щоб отримати більше впевненості, потрібно провести більше спостережень, тобто знайти більше пробних клієнтів.

Подивимося, що буде, якщо ми проведемо експеримент на 300 клієнтах.

```
result = get_stat_power(300, 0.5, 0.6, alpha=0.05)
print(f"Потужність при N=300: {result:.1%}")
```

```
Потужність при N=300: 96.6%
```

Бачимо, що потужність уже дуже близька до 100%. Але провести 300 пробних занять набагато затратніше, ніж 30. І за ресурсами, і за часом. Тому зазвичай балансують між потужністю і тривалістю/витратами експерименту.

Прийнято вважати, що прийнятною для роботи потужністю вважається 80%. Подивимося, як змінюється потужність при зростанні розміру вибірки, і скільки потрібно провести експериментів, щоб детектувати ефект при $\mu = 0.6$ у 80% випадків.

```
n_grid = np.arange(10, 600, 10)                                          ①
power = get_stat_power(n_grid, 0.5, 0.6, alpha=0.05)                     ②

plt.plot(n_grid, power, color=turquoise)                                 ③
plt.axhline(0.8, ls='--', color=red_pink, label='Потужність = 80%')      ④

min_n = n_grid[power >= 0.8].min()                                       ⑤
plt.axvline(min_n, ls='--', color=slate, label=f'N = {min_n}')           ⑥

plt.xlabel('Кількість пробних занять')
plt.ylabel('Потжність')
plt.legend()
plt.show()
```

① Вибираємо значення для осі x
② Обчислюємо потужність для різних розмірів вибірки
③ Будуємо графік
④ Додаємо горизонтальну лінію для потужності 80%
⑤ Знаходимо мінімальну кількість пробних занять, щоб потужність була більшою за 80%
⑥ Додаємо вертикальну лінію для мінімальної кількості пробних занять

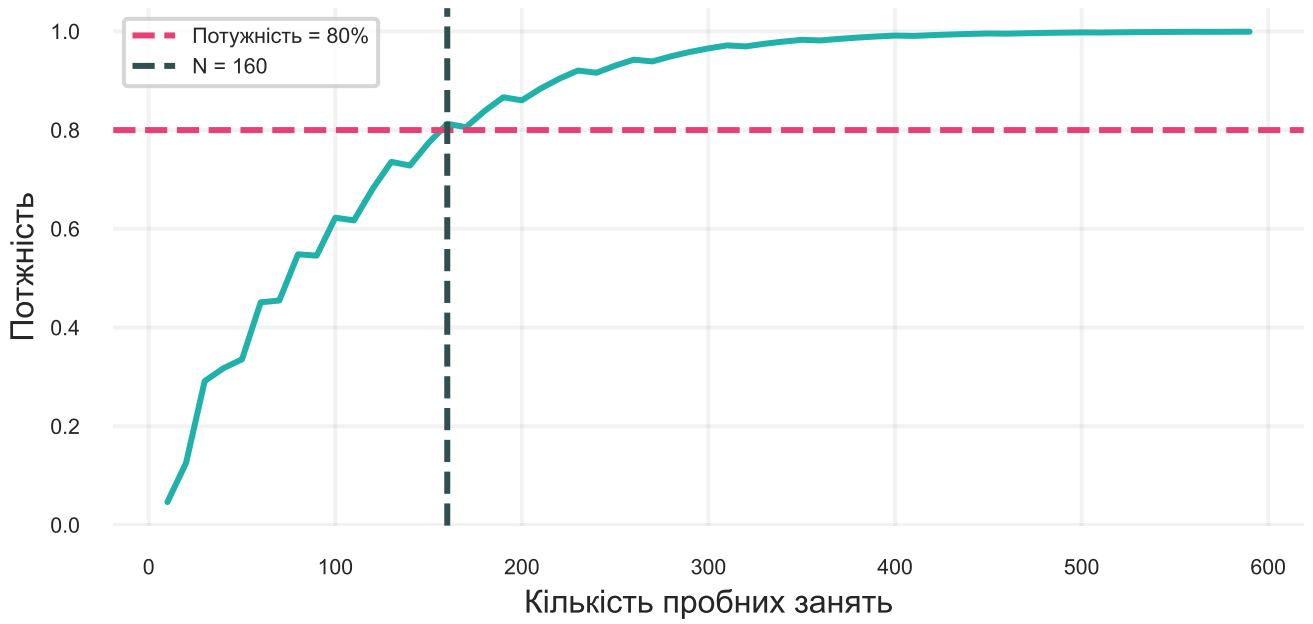


Рисунок 2.2: Залежність потужності від розміру вибірки для $\mu = 0.6$

Бачимо, що для потужності в 80% достатньо набрати 160 пробних занять.

А що, якщо ми хочемо детектувати ще менший ефект? Наприклад, якщо хочемо відкидати гіпотезу за $\mu = 0.51$. Часто поліпшення ймовірності успіху на 1% може бути значущим для продукту, тому це питання не позбавлене сенсу.

```
n_grid = np.arange(10, 30000, 59)                                    ①
power = get_stat_power(n_grid, 0.5, 0.51, alpha=0.05)                ②

plt.plot(n_grid, power, color=turquoise)                             ③
plt.axhline(0.8, ls='--', color=red_pink, label='Потужність = 80%')  ④

min_n = n_grid[power >= 0.8].min()                                   ⑤
plt.axvline(min_n, ls='--', color=slate, label=f'N = {min_n}')       ⑥

plt.xlabel('Кількість пробних занять')
plt.ylabel('Потжність')
plt.legend()
plt.show()
```

① Вибираємо значення для осі x
② Обчислюємо потужність для різних розмірів вибірки
③ Будуємо графік
④ Додаємо горизонтальну лінію для потужності 80%
⑤ Знаходимо мінімальну кількість пробних занять, щоб потужність була більшою за 80%
⑥ Додаємо вертикальну лінію для мінімальної кількості пробних занять

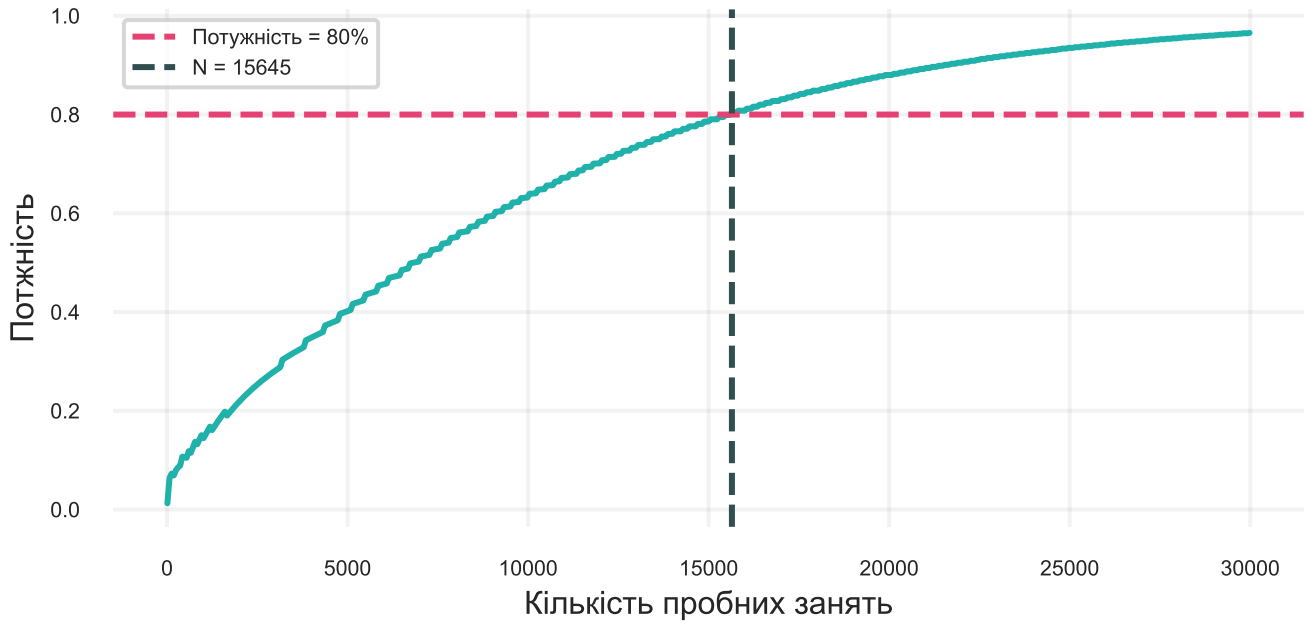


Рисунок 2.3: Залежність потужності від розміру вибірки для $\mu = 0.51$

Бачимо, що потрібно понад 15 тисяч клієнтів, щоб детектувати такий ефект! Дуже складно знайти стільки пробних клієнтів. Але потрібно замислитися над питанням, а чи варто це робити? У нашому випадку, якщо ймовірність успіху 51%, то прибуток із замовлень буде невеликий, і вкладення інвесторів, звісно, окупатимуться, але дуже довго. Тому збільшення на 1 для нашого завдання не значуще *практично*, а отже, не потрібно намагатися набирати 15 тисяч людей, а можна зупинитися і на 160.

Перед кожним експериментом аналітику варто замислюватися над питанням тривалості тесту і кількості учасників. Для цього потрібно зрозуміти:

- Який ефект є для завдання практично значущим?
- Скільки знадобиться випробовуваних, щоб детектувати цей ефект частіше, ніж у 80% випадків?

З графіків видно, що для детектування меншого ефекту потрібен більший розмір вибірки. Подивимося, як для фіксованого $N = 30$ змінюється потужність для різних параметрів μ.

```
mu_grid = np.linspace(0.5, 0.9, 100)                                    ①
power = get_stat_power(30, 0.5, mu_grid, alpha=0.05)                    ②

plt.plot(mu_grid, power, color=turquoise)                               ③
plt.axhline(0.8, ls='--', color=red_pink, label='Потужність = 80%')     ④

min_mu = mu_grid[power >= 0.8].min()                                    ⑤
plt.axvline(min_mu, ls='--', color=slate, label=f'$\mu = {min_mu:.2f}$')  ⑥

plt.xlabel('Ймовірність успіху')
plt.ylabel('Потужність')
plt.legend()
plt.show()
```

① Вибираємо значення для осі x
② Обчислюємо потужність для різних параметрів μ
③ Будуємо графік
④ Додаємо горизонтальну лінію для потужності 80%
⑤ Знаходимо мінімальну ймовірність успіху, щоб потужність була більшою за 80%

⑥ Додаємо вертикальну лінію для мінімальної ймовірності успіху

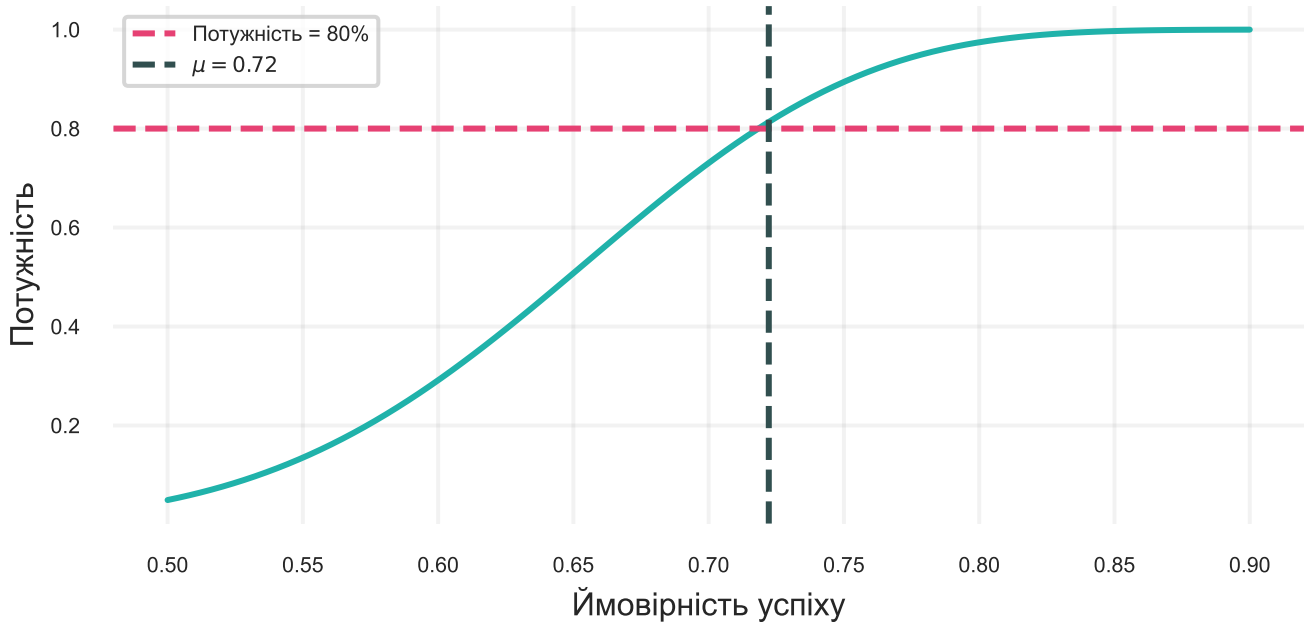


Рисунок 2.4: Залежність потужності від параметра μ

У нашому експерименті ми добре детектуємо ефект, тільки якщо ймовірність успіху в генеральній сукупності хоча б 72%.

2.3 Мінімальна величина ефекту

Вище на Рисунок 2.4 ми побачили, що з хорошою потужністю понад 80% ми можемо помітити ефект у 22 процентних пункти. Причому це можна порахувати навіть до проведення експерименту. У нашому випадку таке збільшення успішності щодо 0.5 цілком можливо, і з ним можна працювати. Але коли аналітики перевіряють зміни, найчастіше очікуваний ефект коливається в районі одного, максимум двох відсотків! Для подібних змін не підійде обрана постановка експерименту, а значить і проводити його не має сенсу.

Тому перед запуском експериментів аналітики повідомляють **мінімальну величину ефекту, яку можна задетектувати** (Minimal Detectable Effect, MDE). У нашому випадку $MDE = +22$ процентних пункти.

Більш формально, MDE для гіпотези $H_0 : \mu = \mu_0$ — це мінімальний ефект δ, за якого критерій рівня значущості α для перевірки цієї гіпотези за істинного параметра $\mu = \mu_0 + \delta$ та розміру вибірки N відкидатиме H_0 з потужністю більшою, ніж $1 - \beta$.

Найчастіше беруть $1 - \beta = 80\%$. Напишемо функцію, яка обчислюватиме MDE підбором.

```
def binom_test_mde_one_sided(N, mu0, alpha=0.05, min_power=0.8):                ①

    delta_grid = np.linspace(0, 1 - mu0, 500)                                   ②
    power = get_stat_power(N, mu0, mu0 + delta_grid, alpha=alpha)               ③
    fit_delta = delta_grid[power >= min_power]                                  ④
    return fit_delta[0]                                                         ⑤

result = binom_test_mde_one_sided(30, 0.5)
print(f"MDE: {result:.1%}")
```

① Функція `binom_test_mde_one_sided`, яка обчислює MDE:

② Створюємо сітку для δ
③ Обчислюємо потужність для різних значень δ
④ Знаходимо значення δ, для яких потужність більша за 80%
⑤ Повертаємо перше значення δ, для якого потужність більша за 80%

```
MDE: 21.8%
```

Результат збігається з обчисленнями за графіком Рисунок 2.4. Тобто ми можемо детектувати ефект у 22 процентних пункти.

Зазвичай MDE розраховують не просто так, а нерозривно з ним іде питання про визначення **розміру вибірки**.

У нашому завданні ми знайшли 30 клієнтів, не обчислюючи спочатку, скільки їх знадобиться. Але що якщо отриманий MDE занадто великий й потрібно зробити його меншим, оскільки очікувані зміни набагато менші? Тоді вирішується зворотне завдання, За необхідним MDE визначити обсяг вибірки. Якщо ми говоримо, що хочемо детектувати +10 в.п., тобто 60% успішних підписок, то потрібно знайти 160 тестових клієнтів, це видно з попередніх графіків. Якщо 30 осіб нам, наприклад, шукати місяць, такий тест може затягнутися майже на півроку. Тому варто подумати про те, щоб виділити додаткові ресурси на пошук клієнтів, наприклад, залучити маркетологів.

2.4 Довірчі інтервали

Раніше ми навчилися перевіряти гіпотезу $H_0 : \mu = 0.5$. Як відповідь ми отримуємо лише вердикт "відкидаємо H_0" або "не відкидаємо H_0". Однак у вибірці міститься набагато більше інформації, й ми можемо більше зрозуміти про параметр, ніж порівняння з числом 0.5.

Якщо гіпотеза H_0 не відкидається, це означає, що значення $\mu = 0.5$ припустиме для нашої вибірки. Отримані значення можна пояснити значенням $\mu = 0.5$. Але якщо у нас є механізм перевірки для будь-якого μ, ми можемо для всіх значень дізнатися, які з них допустимі, і отримати множину можливих значень μ. Така множина називається *довірчим інтервалом*.

Довірчий інтервал рівня $1 - \alpha$ — множина значень параметра μ_0, для яких гіпотеза $\mu = \mu_0$ не відкидається критерієм рівня значущості α.

З визначення випливає, що різні критерії можуть породжувати різні довірчі інтервали. У цій частині розглянемо, які інтервали породжуються двостороннім критерієм. Для цього з кроком 0.001 переберемо значення $\mu \in [0, 1]$ і перевіримо гіпотези.

```python
def two_sided_criterion_nonsym(n, mu, alpha):                                    ①

    binom_h0 = binom(n=n, p=mu)                                                  ②
    c2 = binom_h0.ppf(1 - alpha/2) + 1                                           ③
    c1 = binom_h0.ppf(alpha/2) - 1                                               ④
    return c1, c2                                                                ⑤


success_cnt = 19                                                                 ⑥
mu_grid = np.arange(0, 1, 0.001)                                                 ⑦
mu_no_rejection = []                                                             ⑧

for mu_h0 in mu_grid:                                                            ⑨
    c1, c2 = two_sided_criterion_nonsym(30, mu_h0, alpha=0.05)
    if success_cnt > c1 and success_cnt < c2:
        mu_no_rejection.append(mu_h0)

print(f'95% довірчий інтервал: [{min(mu_no_rejection)} - {max(mu_no_rejection)}]')
```

① Функція, що обчислює критичні значення для двостороннього критерію.

② Розподіл нульової гіпотези
③ Права межа критичної області
④ Ліва межа критичної області
⑤ Повертаємо межі критичної області
⑥ Кількість успіхів
⑦ Сітка для μ
⑧ Список для значень μ, для яких H_0 не відкидається
⑨ Перебираємо значення μ та обчислюємо критичні значення

```
95% довірчий інтервал: [0.439 - 0.8]
```

Отримавши такий інтервал, ми відразу можемо зробити висновок, що гіпотеза $H_0 : \mu = 0.5$ не відкидається, оскільки 0.5 лежить у довірчому інтервалі. Але при цьому відразу зрозуміло, що $\mu \neq 0.4$ на рівні значущості α.

Звичайно ж, у довірчому інтервалі лежить значення $\mu = \frac{19}{30}$, для якого 19 успіхів — це найбільш правдоподібний результат. При цьому інтервал несиметричний щодо точки $\frac{19}{30}$.

Подивимося, як можна візуально знайти межу інтервалу. Ми отримали 19 успіхів. Для кожного μ_0 статистика Q має розподіл $Binom(30, \mu_0)$. Будемо малювати цей розподіл і дивитися, чи потрапляє 19 у критичну область.

```
mus_h0 = [0.2, 0.438, 0.439, 0.8, 0.81, 0.9]                                    ①

fig, axes = plt.subplots(3, 2, figsize=(8, 10))                                 ②

for mu_h0, ax in zip(mus_h0, axes.flatten()):                                   ③
    binom_h0 = binom(n=30, p=mu_h0)                                             ④
    probs = binom_h0.pmf(x_grid)                                                ⑤

    ax.bar(x_grid, probs, color=turquoise, label=f'PMF, $Binom({mu_h0}, 30)$')  ⑥
    c1, c2 = two_sided_criterion_nonsym(30, mu_h0, alpha=0.05)                  ⑦
    crit_reg = (x_grid <= c1) | (x_grid >= c2)                                  ⑧
    ax.bar(x_grid[crit_reg], probs[crit_reg],                                   ⑨
           color=red_pink, label='Критична область')
    is_rejection = success_cnt <= c1 or success_cnt >= c2                       ⑩
    ax.axvline(success_cnt, ls='--',                                            ⑪
        label=f'Q = {success_cnt} ' + ('відхилено' if is_rejection else 'не відхилено'),
        color='gray', alpha=0.4)

    rejection_prob = probs[crit_reg].sum()                                      ⑫
    ax.set_title(f'$\mu = {mu_h0}$', fontsize=10)
    ax.legend()
```

① Сітка для μ
② Створюємо сітку для графіків
③ Перебираємо значення μ та осі
④ Розподіл нульової гіпотези
⑤ Обчислюємо ймовірності
⑥ Будуємо гістограму
⑦ Обчислюємо критичні значення
⑧ Критична область
⑨ Підсвічуємо критичну область
⑩ Чи потрапляє 19 у критичну область
⑪ Вертикальна лінія для 19 успіхів
⑫ Ймовірність потрапляння в критичну область

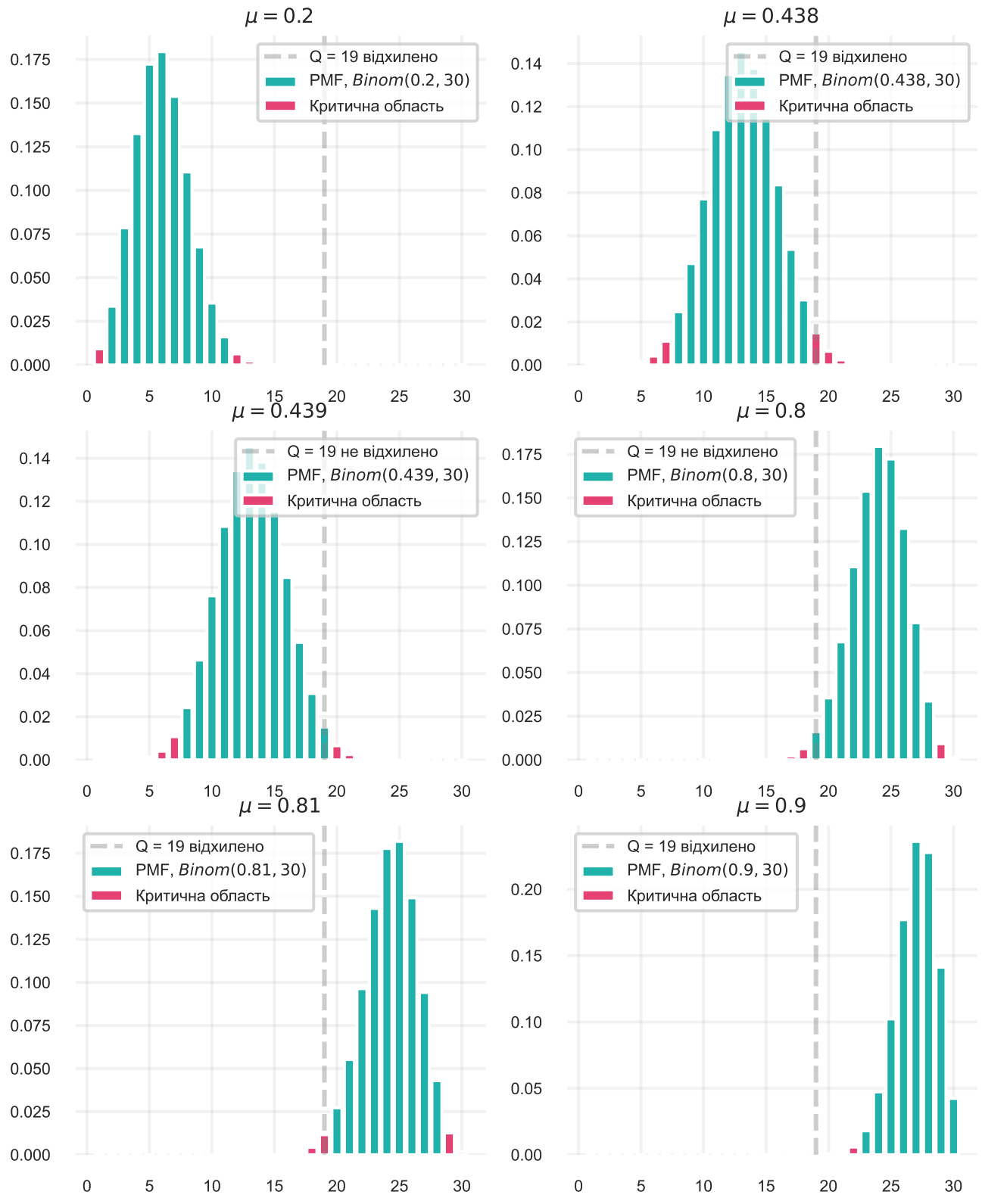


Рисунок 2.5: Довірчі інтервали для біноміального розподілу

Видно, що зі зростанням μ_0 гістограма зсувається вправо. І спочатку 19 потрапляє в праву критичну область. Потім, починаючи з точки 0.439, значення 19 вже опиняється поза критичною областю, і тільки з $\mu_0 = 0.81$

починає потрапляти в ліву критичну область.

Таким чином, ліва межа довірчого інтервалу — це перша точка, коли значення статистики перестало потрапляти до критичної області, а права межа - остання точка, коли значення не потрапляє до правої критичної області.

2.5 Односторонні довірчі інтервали

Насправді, двосторонній критерій потрібен вкрай рідко. Контролювати хибно похитивну помилку нам потрібно тільки для відхилень у бік, корисний для бізнесу. У випадку завдання з освітнім сервісом це отримання *більшої* конверсії в успіх.

Спробуємо скористатися одностороннім критерієм для побудови довірчого інтервалу.

```
def make_binom_criterion(n, mu=0.5, alpha=0.05):                    ①

    binom_h0 = binom(n=n, p=mu)                                     ②
    q = binom_h0.ppf(1 - alpha)                                     ③
    return q + 1                                                    ④

success_cnt = 19                                                    ⑤
mu_grid = np.arange(0, 1.001, 0.001)                                ⑥
mu_no_rejection = []                                                ⑦

for mu_h0 in mu_grid:                                               ⑧
    crit_val = make_binom_criterion(n=30, mu=mu_h0, alpha=0.05)
    if success_cnt < crit_val:
        mu_no_rejection.append(mu_h0)

print(f'95% довірчий інтервал: [{min(mu_no_rejection)} - {max(mu_no_rejection)}]')
```

① Функція, що обчислює критичне значення для одностороннього критерію.
② Розподіл нульової гіпотези
③ Критичне значення
④ Повертаємо критичне значення
⑤ Кількість успіхів
⑥ Сітка для μ
⑦ Список для значень μ, для яких H_0 не відкидається
⑧ Перебираємо значення μ та обчислюємо критичне значення

```
95% довірчий інтервал: [0.467 - 1.0]
```

Коли ми використовували двосторонній інтервал, ми отримали ліву межу $0.439 < 0.467$. Виходить, що односторонній інтервал з точки зору лівої межі дає нам більше інформації. При цьому з точки зору правої межі ми втрачаємо інформацію зовсім. Вона дорівнює 1 просто тому, що ймовірність не може бути більшою.

Насправді зазвичай на праву межу не дивляться під час аналізу, коли ми шукаємо позитивний ефект.

Припустимо, ми отримали не 19 успіхів, а 22. Побудуємо 2 види інтервалів.

```
success_cnt = 22
mu_grid = np.arange(0, 1, 0.001)
mu_no_rejection = []

for mu_h0 in mu_grid:
    c1, c2 = two_sided_criterion_nonsym(30, mu_h0, alpha=0.05)
    if success_cnt > c1 and success_cnt < c2:
        mu_no_rejection.append(mu_h0)
```

```
print(f'Двосторонній 95% ДІ: [{min(mu_no_rejection):.3f} - {max(mu_no_rejection):.3f}]')
```

```
Двосторонній 95% ДІ: [0.542 - 0.877]
```

```
success_cnt = 22
mu_grid = np.arange(0, 1.001, 0.001)
mu_no_rejection = []

for mu_h0 in mu_grid:
    crit_val = make_binom_criterion(n=30, mu=mu_h0, alpha=0.05)
    if success_cnt < crit_val:
        mu_no_rejection.append(mu_h0)

print(f'Односторонній 95% ДІ: [{min(mu_no_rejection):.3f} - {max(mu_no_rejection):.3f}]')
```

```
Односторонній 95% ДІ: [0.571 - 1.000]
```

За обома довірчими інтервалами ми робимо висновок, що конверсія значимо відрізняється від 50%. Але односторонній інтервал дає кращу нижню оцінку на ймовірність успіху. Ми можемо зрозуміти, що наша конверсія більша за 57%. А інформація з двостороннього інтервалу про те, що ймовірність менша за 88% не додає нам користі.

Навіщо ж ми тоді взагалі використовуємо двосторонній інтервал? Щоб це зрозуміти, подивимося, як виглядають візуально межі для одностороннього інтервалу.

```
fig, axes = plt.subplots(3, 2, figsize=(8, 10))

for mu_h0, ax in zip(mus_h0, axes.flatten()):                                    ①
    binom_h0 = binom(n=30, p=mu_h0)                                              ②
    probs = binom_h0.pmf(x_grid)                                                 ③

    ax.bar(x_grid, probs, color=turquoise, label=f'PMF, $Binom({mu_h0}, 30)$')   ④
    c = make_binom_criterion(30, mu_h0, alpha=0.05)                              ⑤
    crit_reg = (x_grid >= c)                                                     ⑥
    ax.bar(x_grid[crit_reg], probs[crit_reg],                                    ⑦
           color=red_pink, label='Критична область')

    is_rejection = success_cnt >= c                                              ⑧
    ax.axvline(success_cnt, ls='--',                                             ⑨
    label=f'Q = {success_cnt} ' + ('відхилено' if is_rejection else 'не відхилено'),
    color='gray', alpha=0.4)

    rejection_prob = probs[crit_reg].sum()                                       ⑩
    ax.set_title(f'$\mu = {mu_h0}$', fontsize=8)                                 ⑪
    ax.legend()
```

① Сітка для μ
② Розподіл нульової гіпотези
③ Обчислюємо ймовірності
④ Будуємо гістограму
⑤ Обчислюємо критичні значення
⑥ Критична область
⑦ Підсвічуємо критичну область
⑧ Чи потрапляє 22 у критичну область
⑨ Вертикальна лінія для 22 успіхів
⑩ Ймовірність потрапляння в критичну область
⑪ Заголовок для графіка

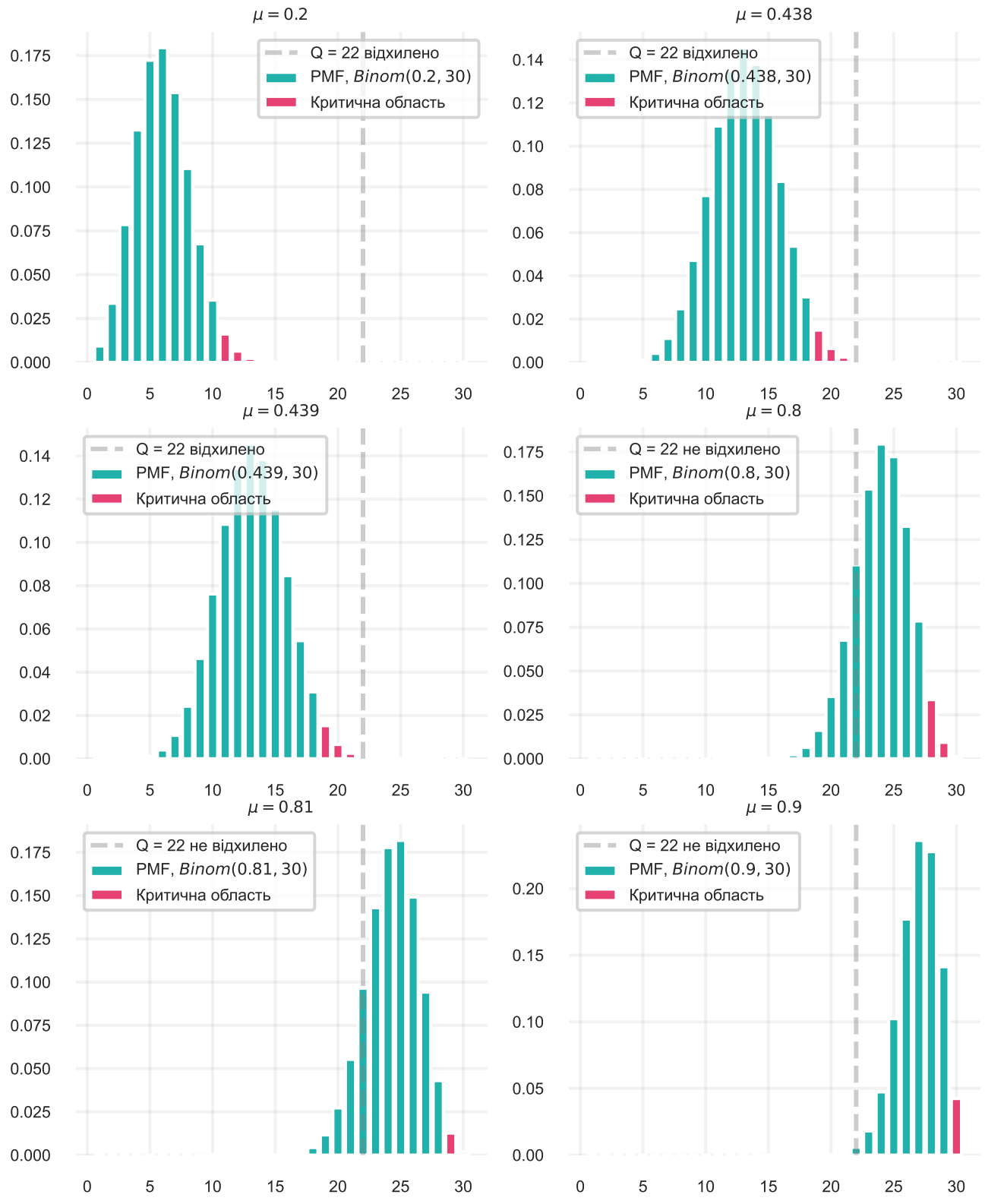


Рисунок 2.6: Односторонній довірчий інтервал для 22 успіхів

Порівняно з Рисунок 2.5 ми бачимо, що права критична область стала більшою через те, що там тепер знаходиться не 2.5%, а 5% від усіх значень. При цьому лівої критичної області просто не існує, тому за великих

μ не відбувається потрапляння 19 до неї, а значить ми не відкидаємо гіпотезу.

Зауважимо, що якби ми будували двосторонній інтервал, але з удвічі більшою α, потрапляння в праву критичну область траплялися б за тих самих μ, що й в односторонньому критерії. Тому часто для пошуку односторонньої межі будують двосторонній довірчий інтервал із більшою α, ігноруючи при цьому праву межу. Це зручно, оскільки можна користуватися тільки однією функцією для критерію.

Перевіримо, що вийде за $\alpha = 0.1$.

```
success_cnt = 19
mu_grid = np.arange(0, 1, 0.001)
mu_no_rejection = []

for mu_h0 in mu_grid:
    c1, c2 = two_sided_criterion_nonsym(30, mu_h0, alpha=0.1)
    if success_cnt > c1 and success_cnt < c2:
        mu_no_rejection.append(mu_h0)

print(f'95% довірчий інтервал: [{min(mu_no_rejection):.3f} - {max(mu_no_rejection):.3f}]')
```

```
95% довірчий інтервал: [0.467 - 0.778]
```

Бачимо, що отримали таку саму ліву межу, як і в односторонньому інтервалі.

2.6 Властивості довірчих інтервалів

Згадаймо визначення довірчого інтервалу.

Нехай є критерій $S = \{Q(\xi) \leqslant C\}$ рівня значущості α для перевірки гіпотези $H_0 : \mu = \mu_0$, Q — статистика критерію, а q — її реалізація на конкретній вибірці $\xi = \xi_1, \dots, \xi_n$. Тоді **довірчим інтервалом** називається множина таких μ_0, на яких критерій S не відкидає гіпотезу $H_0 : \mu = \mu_0$.

Процедура підрахунку інтервалу — це довгий перебір значень із деяким кроком. Але це все ще залишається деякою функцією від вибірки, тобто *статистикою* й випадковою величиною, причому її розподіл залежить від статистики Q, а отже, і від початкової вибірки, та від параметра μ у генеральній сукупності.

Позначимо межі інтервалу за $\mathcal{L}(Q), \mathcal{R}(Q)$ — статистики критерію, які відповідають лівій та правій межі інтервалу.

2.6.1 Ймовірність попадання в інтервал

Яким би не було істинне значення $\mu = \mu_0$, ймовірність того, що воно перебуває між $\mathcal{L}(Q)$ та $\mathcal{R}(Q)$, не нижча, ніж $1 - \alpha$. Значення $1 - \alpha$ називається **рівнем довіри** довірчого інтервалу.

$$P(\mathcal{L}(Q) < \mu_0 < \mathcal{R}(Q)) \geqslant 1 - \alpha \tag{2.3}$$

Важливо, що випадковість тут прихована саме в $\mathcal{L}$ і $\mathcal{R}$, а не в μ_0. Параметр μ_0 невідомий, але ми припускаємо його константним і не випадковим.

Перевіримо справедливість цієї властивості. Для цього зафіксуємо μ_0 й проведемо множину експериментів:

- Генеруємо вибірку з розподілу з параметром μ_0.
- Обчислюємо статистику q.
- Рахуємо довірчий інтервал для $\alpha = 0.05$.

Перевіряємо, що частка випадків, коли параметр μ_0 опинився всередині інтервалу, хоча б 95%.

```
import time                                                    ①

start_time = time.time()                                       ②
```

```python
def my_binomial_confint(n, alpha, q):                                    ③
    mu_grid = np.arange(0, 1.1, 0.1) # np.arange(0, 1.001, 0.001)        ④
    mu_no_rejection = []                                                 ⑤

    for mu_h0 in mu_grid:                                                ⑥
        c1, c2 = two_sided_criterion_nonsym(30, mu_h0, alpha=0.05)       ⑦
        if q > c1 and q < c2:                                            ⑧
            mu_no_rejection.append(mu_h0)                                ⑨

    return min(mu_no_rejection), max(mu_no_rejection)                    ⑩
```

① Імпортуємо бібліотеку для вимірювання часу
② Запускаємо таймер
③ Функція, що обчислює довірчий інтервал
④ Сітка для μ
⑤ Список для значень μ, для яких H_0 не відкидається
⑥ Перебираємо значення μ та осі
⑦ Обчислюємо критичні значення
⑧ Чи потрапляє q у критичну область
⑨ Додаємо значення μ, для яких H_0 не відкидається
⑩ Повертаємо межі довірчого інтервалу

Тепер запустимо експеримент. Для цього зафіксуємо $\mu_0 = 0.5$ й проведемо 1000 експериментів. У кожному з них будемо генерувати 30 успіхів, а потім перевіряти, чи потрапила μ_0 у довірчий інтервал.

```python
N_EXPERIMENTS = 1000                                                     ①
SAMPLE_SIZE = 30                                                         ②
latent_mu = 0.5                                                          ③
binom_true = binom(n=SAMPLE_SIZE, p=latent_mu)                           ④

confint_fail_cases = 0                                                   ⑤

for i in range(N_EXPERIMENTS):                                           ⑥
    q = binom_true.rvs()                                                 ⑦
    L, R = my_binomial_confint(n=SAMPLE_SIZE, alpha=0.05, q=q)           ⑧
    if L < latent_mu < R:                                                ⑨
        pass
    else:
        confint_fail_cases += 1

success_cases = 100 * (N_EXPERIMENTS - confint_fail_cases) / N_EXPERIMENTS  ⑩
end_time = time.time()                                                   ⑪

print(f"Відсоток успішних випадків: {success_cases:.2f}%")
print(f"Час виконання: {end_time - start_time:.4f} секунди")
```

① Кількість експериментів
② Розмір вибірки
③ Істинне значення ймовірності успіху
④ Розподіл нульової гіпотези
⑤ Лічильник невдалих випадків
⑥ Перебираємо експерименти
⑦ Генеруємо вибірку
⑧ Обчислюємо довірчий інтервал
⑨ Перевіряємо, чи потрапила μ_0 у довірчий інтервал
⑩ Обчислюємо частку успішних випадків

⑪ Вимірюємо час виконання

```
Відсоток успішних випадків: 61.40%
Час виконання: 3.6399 секунди
```

Зазначимо, що цей код працював понад 5 хвилин. Це через те, що під час кожного експерименту потрібно побудувати довірчий інтервал, а значить перевірити 1000 можливих параметрів μ_0.

Бачимо, що властивість виконалася. Ми очікували хоча б 95% влучень, отримали навіть 61.4%. Насправді це значно більше, ніж ми очікували. Це відбувається через дискретність розподілу. З тієї ж причини під час пошуку критичної області ми не могли вибрати стовпці із сумарною висотою рівно α.

2.6.1.1 Доведення

Під час формулювання властивості ми припускаємо, що є деяка μ_0 — ймовірність успіху в генеральній сукупності. Коли ми проводимо штучний експеримент, ми фіксуємо її й можемо вважати істинною μ.

Щоразу ми генеруємо $Q \sim Binom(\mu_0, 30)$ й перевіряємо, чи потрапила μ_0 у довірчий інтервал. Намалюємо розподіл статистики Q, який уже нам знайомий. Намалюємо й область ймовірності $\leqslant \alpha$, як ми робили це раніше.

```
mu0 = 0.5                                                                    ①
binom_mu0 = binom(n=30, p=mu0)                                               ②
probs = binom_mu0.pmf(x_grid)                                                ③

plt.bar(x_grid, probs, color=turquoise, label=f'PMF, $Binom({mu0}, 30)$')    ④
c1, c2 = two_sided_criterion_nonsym(30, mu0, alpha=0.05)                     ⑤
crit_reg = (x_grid >= c2) | (x_grid <= c1)                                   ⑥
plt.bar(x_grid[crit_reg], probs[crit_reg],                                   ⑦
        color=red_pink, label='Критична область')
plt.legend()
plt.show()
```

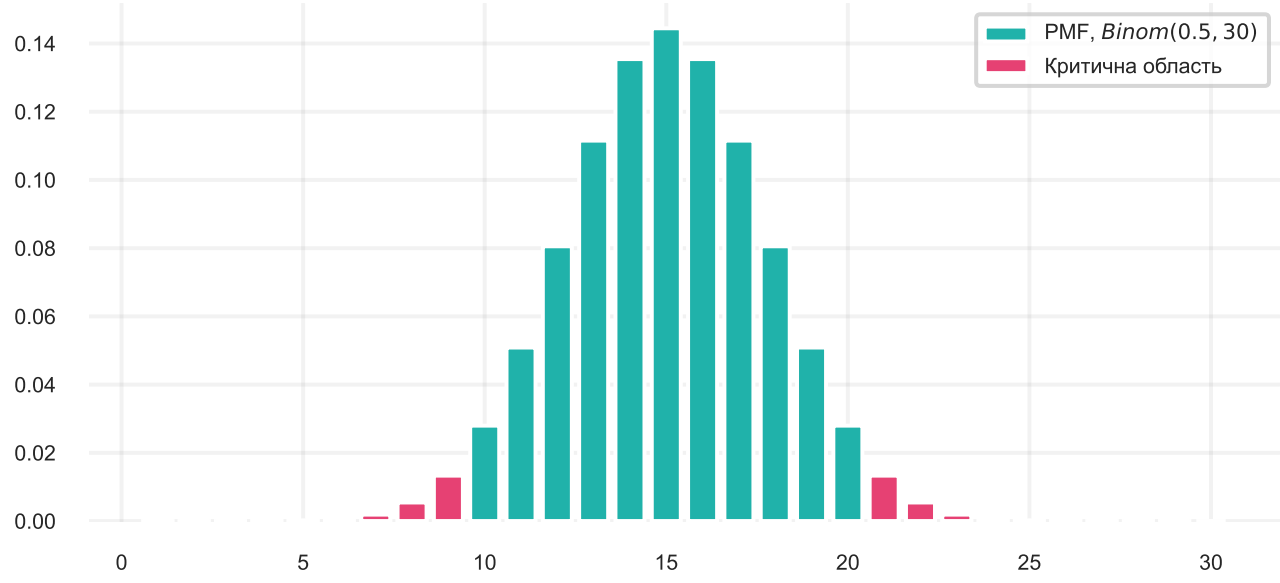


Рисунок 2.7: Розподіл статистики при істинній ймовірності успіху

Нехай реалізувалося значення статистики q. За такою вибіркою можна побудувати довірчий інтервал на μ. Він буде якось розташований, але зараз нас цікавить, чи потрапить у нього μ_0. За визначенням потрапляння в інтервал відбудеться, якщо не відкидається гіпотеза $H_0 : \mu = \mu_0$. Але тоді за справедливості H_0 статистика

має той розподіл, що і на малюнку. І гіпотеза відкидається тільки в разі потрапляння в критичну область, а це трапляється з ймовірністю $\leqslant \alpha$.

Отже, з ймовірністю хоча б $1 - \alpha$ μ_0 перебуватиме в довірчому інтервалі.

Часто так й вводять визначення довірчого інтервалу. Для вибірки $\xi_1, \dots, \xi_n$ — це така пара статистик $\mathcal{L}(\xi)$ і $\mathcal{R}(\xi)$, що хоч яким би не було μ_0,

$$P(L(\xi) < \mu_0 < R(\xi)) \geqslant 1 - \alpha \tag{2.4}$$

де $L(\xi)$ і $R(\xi)$ — статистики, що залежать від вибірки. Знову звертаємо увагу, що випадковість тут прихована не в параметрі μ_0, а в статистиках від вибірки.

2.6.2 Довірчий інтервал Вілсона

Розглянутий зараз алгоритм побудови довірчого інтервалу працює занадто довго. У Python є функції, які дозволяють швидше розрахувати інтервал. Наприклад, можна скористатися методом Вілсона і функцією `proportion_confint`.

Повторимо експерименти з новим типом довірчого інтервалу, тут можемо дозволити більше реалізацій вибірки, оскільки інтервал рахується недовго.

```
from statsmodels.stats.proportion import proportion_confint

start_time = time.time()

N_EXPERIMENTS = 1000
SAMPLE_SIZE = 30
latent_mu = 0.5
binom_true = binom(n=SAMPLE_SIZE, p=latent_mu)

confint_fail_cases = 0

for i in range(N_EXPERIMENTS):
    q = binom_true.rvs()
    L, R = proportion_confint(
        count=q,
        nobs=SAMPLE_SIZE,
        alpha=0.05,
        method='wilson'
    )
    if L < latent_mu < R:
        pass
    else:
        confint_fail_cases += 1

success_cases = 100 * (N_EXPERIMENTS - confint_fail_cases) / N_EXPERIMENTS
end_time = time.time()

print(f"Відсоток успішних випадків: {success_cases:.2f}%")
print(f"Час виконання: {end_time - start_time:.4f} секунди")
```

```
Відсоток успішних випадків: 96.60%
Час виконання: 0.0739 секунди
```

Зауважимо, що наше μ може знаходитись в довірчому інтервалі менше, ніж у 95% випадків. Це відбувається через те, що швидкі методи працюють наближено, оцінюючи розподіл статистики при збільшенні розміру вибірки. Чим розмір вибірки більший, тим ближчим буде інтервал до 95%-ного.

Залежність частки успішних влучень μ у довірчий інтервал від розміру вибірки зобразимо на Рисунок 2.8.

```
n_grid = np.arange(1, 1000, 25).tolist()                                    ①
interval_success_rate = []                                                  ②

for n in n_grid:                                                            ③
    confint_fail_cases = 0                                                  ④
    for i in range(N_EXPERIMENTS):                                          ⑤
        binom_true = binom(n=n, p=latent_mu)                                ⑥
        q = binom_true.rvs()                                                ⑦
        L, R = proportion_confint(                                          ⑧
            count=q,
            nobs=n,
            alpha=0.05,
            method='wilson'
        )
        if L < latent_mu < R:                                               ⑨
            pass
        else:
            confint_fail_cases += 1
    interval_success_rate.append(1 - confint_fail_cases / N_EXPERIMENTS)    ⑩
```

① Сітка для розміру вибірки
② Список для частки успішних влучень
③ Перебираємо розміри вибірки
④ Лічильник невдалих випадків
⑤ Перебираємо експерименти
⑥ Генеруємо вибірку
⑦ Генеруємо q з розподілу
⑧ Обчислюємо довірчий інтервал
⑨ Перевіряємо, чи потрапила μ_0 у довірчий інтервал
⑩ Обчислюємо частку успішних випадків

Тепер побудуємо графік.

```
plt.plot(n_grid, interval_success_rate,                                     ①
         label='Частка успішних влучень', color=turquoise)
plt.axhline(0.95, ls='--',                                                  ②
            label='Желаемая успешность', color=red_pink)

plt.xlabel('Розмір вибірки $n$')
plt.ylabel('Частка успішних влучень')
plt.legend()
plt.show()
```

① Графік частки успішних влучень
② Лінія бажаної частки успішних влучень

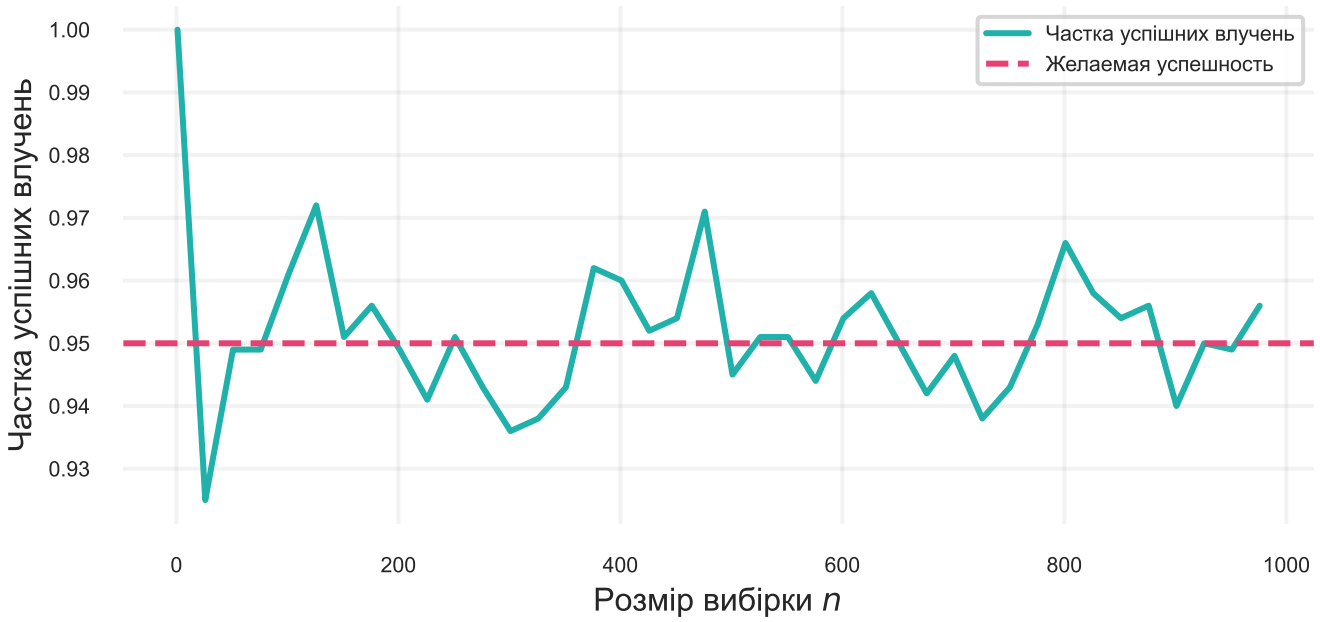


Рисунок 2.8: Залежність частки успішних влучень μ у довірчий інтервал від розміру вибірки

Видно, що на будь-якому розмірі вибірки під час використання інтервалу Вілсона можна отримати менше 95% влучень, але що більший розмір вибірки, то менше графік відхиляється від 95%.

2.7 Питання для самоперевірки

Загальні поняття та Помилки

1. Які два основні типи помилок можна зробити при перевірці статистичних гіпотез? Назвіть їх (FPR та FNR) та поясніть, що кожна з них означає.
2. Як рівень значущості (α) пов'язаний з одним із типів помилок?
3. Поясніть на прикладі "критерію пори року", чому критерій з низьким рівнем значущості (α) не обов'язково є хорошим чи корисним.

Статистична Потужність

4. Що таке статистична потужність критерію? Як вона пов'язана з хибно негативною помилкою (FNR)?
5. Чому в статистиці часто фокусуються на максимізації потужності (або мінімізації FNR) при фіксованому рівні значущості α?
6. Від яких основних факторів залежить потужність статистичного критерію? Як кожен з них (розмір вибірки, величина ефекту, рівень значущості) впливає на потужність?
7. Як змінюється потужність критерію для біноміального розподілу при збільшенні розміру вибірки (N)? Наведіть приклад з тексту.
8. Як змінюється потужність критерію для біноміального розподілу при збільшенні різниці між параметром нульової гіпотези (μ_0) та параметром альтернативної гіпотези ($\mu_{alternative}$)? Наведіть приклад з тексту.
9. Яке значення потужності часто вважається прийнятним мінімальним рівнем у практичних дослідженнях?

Мінімальна Величина Ефекту (MDE)

10. Що таке Мінімальна величина ефекту, яку можна задетектувати (MDE)?
11. Як MDE пов'язаний з потужністю, розміром вибірки та рівнем значущості?
12. Чому важливо розраховувати MDE перед початком експерименту?

13. Поясніть різницю між *статистичною* значущістю та *практичною* значущістю ефекту. Чому ця різниця важлива?

Довірчі Інтервали (ДІ)

14. Дайте визначення довірчого інтервалу рівня $1 - \alpha$. Як він пов'язаний з перевіркою гіпотез?
15. Опишіть основний принцип побудови довірчого інтервалу, який використовувався в тексті для біноміального розподілу (через перебір значень μ_0).
16. Чим відрізняються двосторонні та односторонні довірчі інтервали? В якій ситуації може бути доцільніше використовувати односторонній ДІ?
17. Як межа одностороннього довірчого інтервалу пов'язана з межами двостороннього довірчого інтервалу, побудованого з подвоєним рівнем значущості (наприклад, $\alpha = 0.1$ для 95% одностороннього інтервалу)?
18. Поясніть ключову властивість довірчого інтервалу: $P(\mathcal{L}(Q) < \mu_0 < \mathcal{R}(Q)) \geqslant 1 - \alpha$. Що є випадковим у цій нерівності (параметр μ_0 чи межі інтервалу $\mathcal{L}, \mathcal{R}$)?
19. Чому при використанні точних методів побудови ДІ для дискретних розподілів (як біноміальний) фактична ймовірність покриття істинного параметра може бути *більшою* за $1 - \alpha$?
20. Які переваги та недоліки використання наближених методів побудови ДІ, таких як метод Вілсона? Коли їх використання виправдане?

Chapter 3

Z-критерій Фішера

У цьому розділі ми розглянемо Z-критерій Фішера, який використовується для перевірки гіпотез про середнє значення генеральної сукупності з відомою дисперсією.

Далі, для виведення критеріїв нам потрібен нормальний розподіл. *Потому що саме цьому розподілу підпорядковується середнє вибірок.* Тож давайте подивимося, що це взагалі таке, як з ним працювати в Python й які в нього є властивості.

3.1 Нормальний розподіл

Нормальний розподіл $\mathcal{N}(\mu, \sigma^2)$ — неперервний розподіл, у якому щільність спадає зі збільшенням відстані від математичного сподівання μ за швидкістю, пропорційною квадрату відстані (див. формулу 3.1).

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{3.1}$$

де x — випадкова величина, μ — математичне сподівання, σ^2 — дисперсія.

На графіку нижче показано, як виглядає нормальний розподіл з різними параметрами μ та σ^2.

Лістинг 3.1 Візуалізація нормального розподілу з різними параметрами μ та σ^2.

```
x = np.linspace(-5, 5, 1000)
params = [(0, 1), (0, 2), (1, 1), (1, 2), (2, 1), (2, 2)]

for mu, sigma in params:
    plt.plot(x, norm.pdf(x, mu, sigma), label=f'μ={mu}, σ={sigma}')

plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.show()
```

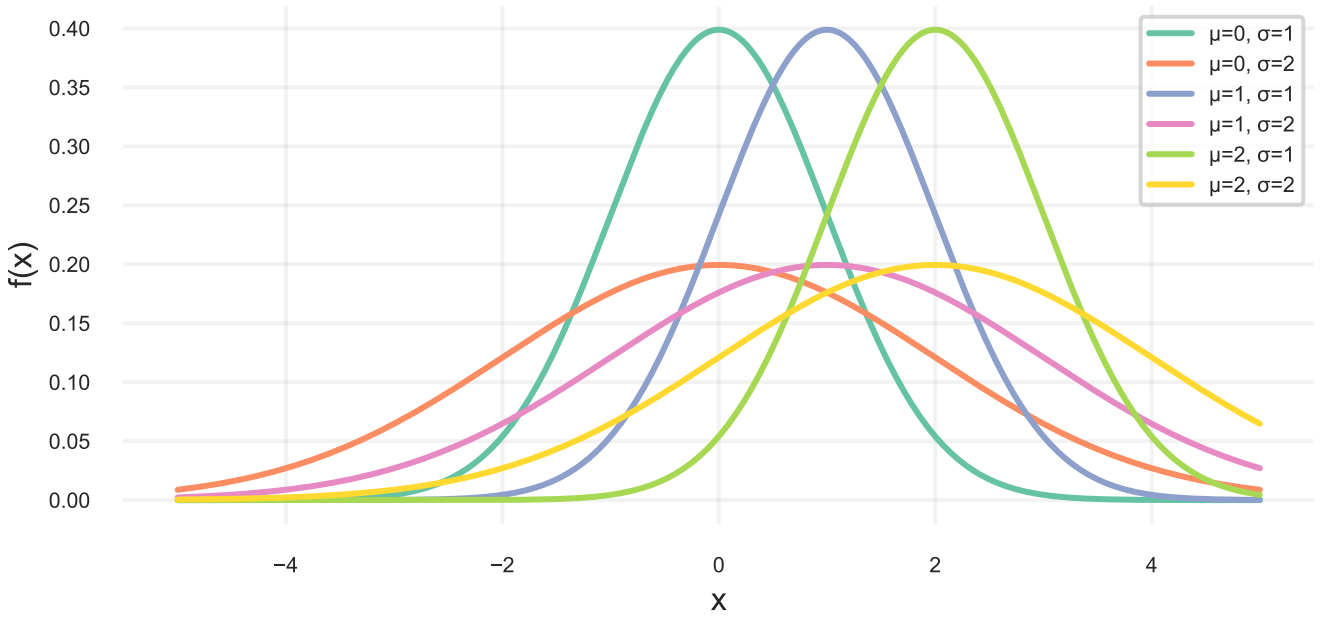


Рисунок 3.1: Нормальний розподіл з різними параметрами

3.2 Нормальний розподіл у Python

Нехай ми хочемо задати розподіл $\mathcal{N}(\mu, \sigma^2)$. Для цього є клас `norm`[1].

Параметри класу:

- `loc` — це μ
- `scale` — це σ, або **стандартне відхилення**. Не дисперсія!

Методи класу:

- `rvs()` — згенерувати випадкові числа з розподілу $\mathcal{N}(\mu, \sigma^2)$
- `cdf(x)` — кумулятивна функція розподілу (cumulative distribution function, CDF) в точці x, ймовірність того, що випадкова величина X менша або дорівнює x.
- `ppf(q)` — квантиль функції розподілу (percent-point function, PPF) для ймовірності q, ймовірність того, що випадкова величина X менша або дорівнює q.
- `pdf(x)` — щільність ймовірності (probability density function, PDF) в точці x, ймовірність того, що випадкова величина X дорівнює x.

CDF та PPF — це функції, які пов'язані між собою. CDF визначає ймовірність того, що випадкова величина X менша або дорівнює x, а PPF визначає значення x, для якого ймовірність X менша або дорівнює q.

Ініціалізуємо клас `norm` з параметрами $\mu = 0$ та $\sigma = 1$ (стандартний нормальний розподіл). Далі, згенеруємо випадкову вибірку з 50 спостережень, а також обчислимо PDF, CDF та PPF для $x = 1.5$.

Візуалізація методів класу `norm` показана на рисунку 3.2.

[1]Документація доступна за посиланням https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html.

Лістинг 3.2 Нормальний розподіл у Python.

```
std_norm = norm(loc=0, scale=1)                                                    ①

rnorm = std_norm.rvs(size=50, random_state=42)                                     ②

CDF = std_norm.cdf(1.5)                                                            ③
PDF = std_norm.pdf(1.5)                                                            ④
PPF = std_norm.ppf(0.933)                                                          ⑤

display(
    Markdown(f"$P(X \\leq 1.5) = {CDF:.3f}$"),                                     ⑥
    Markdown(f"$f(1.5) = {PDF:.3f}$"),                                             ⑦
    Markdown(f"$z_{{0.933}} = \Phi^{{-1}}(0.933) = {PPF:.3f}$")                    ⑧
)
```

① Ініціалізація класу `norm` з параметрами $\mu = 0$ та $\sigma = 1$.
② Генерація випадкової вибірки з 50 спостережень.
③ Обчислення PDF для $x = 1.5$.
④ Обчислення CDF для $x = 1.5$.
⑤ Обчислення PPF для $q = 0.933$.
⑥ Ймовірність того, що випадкова величина X менша або дорівнює 1.5.
⑦ Ймовірність того, що випадкова величина X дорівнює 1.5.
⑧ Значення x, для якого ймовірність X менша або дорівнює 0.933.

$$P(X \leq 1.5) = 0.933$$
$$f(1.5) = 0.130$$
$$z_{0.933} = \Phi^{-1}(0.933) = 1.499$$

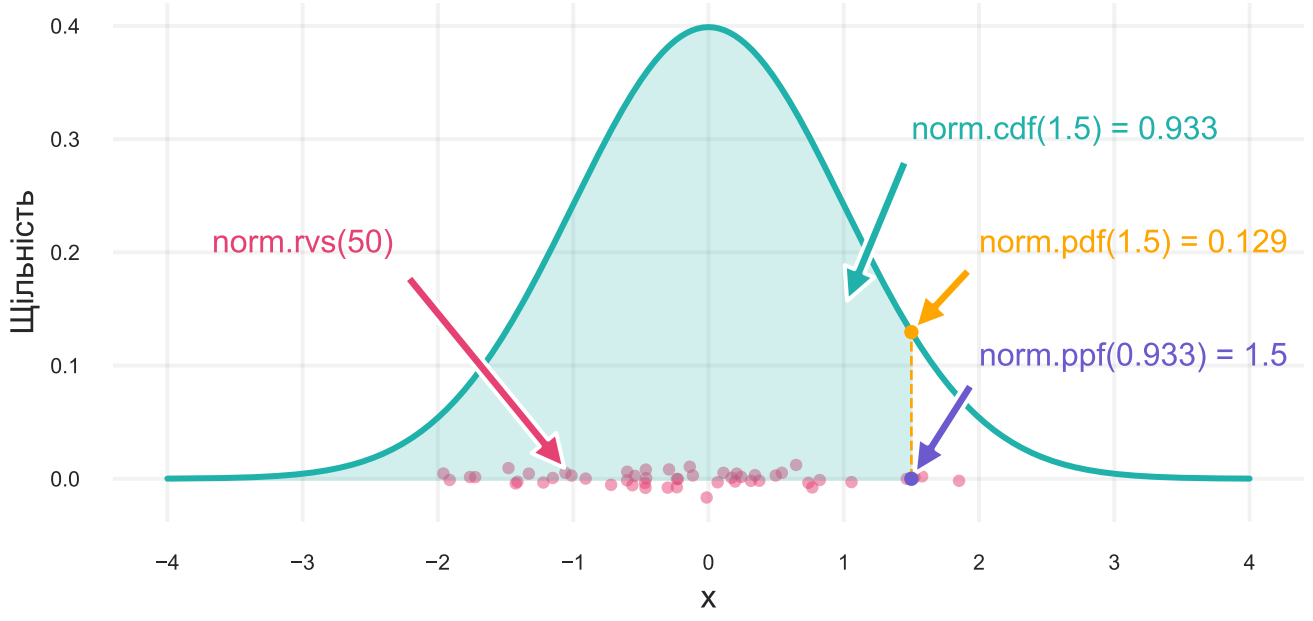


Рисунок 3.2: Демонстрація методів класу `norm`

3.3 Властивості нормального розподілу

Нормальний розподіл має кілька важливих властивостей[2]:

1. **Сума двох незалежних нормально розподілених випадкових величин також має нормальний розподіл**:

$$\begin{aligned} \xi_1 &\sim \mathcal{N}(\mu_1, \sigma_1^2) \\ \xi_2 &\sim \mathcal{N}(\mu_2, \sigma_2^2) \\ \xi_1 + \xi_2 &\sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2) \end{aligned} \tag{3.2}$$

де ξ_1 та ξ_2 — незалежні нормально розподілені випадкові величини з параметрами μ_1, σ_1^2 та μ_2, σ_2^2 відповідно.

2. **Множення нормально розподіленої випадкової величини на константу також дає нормально розподілену величину**:

$$a\xi_1 \sim \mathcal{N}(a\mu_1, a^2\sigma_1^2) \tag{3.3}$$

де a — константа, ξ_1 — нормально розподілена випадкова величина з параметрами μ_1, σ_1^2.

3.3.1 Перевірка властивостей в Python

За допомогою мови Python ми можемо перевірити ці властивості. Почнемо з Рівняння 3.2. Для цього ми згенеруємо дві нормально розподілені випадкові величини ξ_1 та ξ_2 з параметрами $\mu_1 = 0$, $\sigma_1^2 = 1$ та $\mu_2 = 1$, $\sigma_2^2 = 4$. Потім, ми обчислимо їхню суму та перевіримо, чи має вона нормальний розподіл з параметрами $\mu_1 + \mu_2$ та $\sigma_1^2 + \sigma_2^2$.

Видно, що розподіли приблизно збіглися! А значить ми переконалися, що формула правильна.

Другу властивість Рівняння 3.3 можна перевірити аналогічно. Для цього ми згенеруємо нормально розподілену випадкову величину ξ_1 з параметрами $\mu_1 = 0$, $\sigma_1^2 = 1$ та помножимо її на константу $a = 2$. Потім, ми перевіримо, чи має вона нормальний розподіл з параметрами $a\mu_1$ та $a^2\sigma_1^2$.

Цього разу розподіли також збіглися. А значить ми переконалися, що формула правильна.

3.4 Центральна гранична теорема

Для початку пригадаємо теорему, яка є основоположною теоремою для всіх критеріїв, які ми розглянемо найближчим часом.

Теорема 3.1 (Центральна гранична теорема, ЦГТ)**.** *Нехай $\xi_1, ..., \xi_n$ — **незалежно** однаково розподілені випадкові величини, в яких існують математичне сподівання та дисперсія: $E[\xi_i] = \mu < \infty$ і $Var[\xi_i] = \sigma^2 < \infty$, тоді $\sqrt{n}\frac{\overline{\xi} - \mu}{\sqrt{\sigma^2}}$ збігається за розподілом[3] до $\mathcal{N}(0, 1)$.*

Це означає, що якщо випадкові величини в експерименті **незалежні й однаково розподілені** й ваша вибірка **досить велика**, то можна вважати, що

$$\sqrt{n}\frac{\overline{\xi} - \mu}{\sqrt{\sigma^2}} \sim \mathcal{N}(0, 1), \tag{3.4}$$

де $\overline{\xi}$ — середнє арифметичне вибірки, n — кількість спостережень, μ — математичне сподівання генеральної сукупності, σ^2 — дисперсія генеральної сукупності.

[2] Доведення цих властивостей можна знайти в роботі Lemons (2002).

[3] Послідовність випадкових величин ξ_n збігається за розподілом до ξ, позначаємо $\xi_n \xrightarrow{d} \xi$, якщо $\lim_{n\to\infty} F_{\xi_n}(x) = F_\xi(x)$ для всіх x, в яких $F_\xi(x)$ неперервна.

> **Примітка**
>
> Випадкові величини можуть бути слабко залежні одна від одної й злегка по-різному розподілені. Центральна гранична теорема все ще буде правильною, Gnedenko and Kolmogorov (2021).

3.4.1 Візуалізація ЦГТ

Щоб краще розуміти, як працює ЦГТ, я пропоную візуалізувати теорему: подивимося на розподіл середніх значень у різних вибірках. Як ми це зробимо?

- Щоб подивитися, що деяка випадкова величина з нормального розподілу, нам потрібна вибірка цих випадкових величин.
- У цьому випадку нам потрібна вибірка статистик із ЦГТ. Тому нам потрібно згенерувати N вибірок по M елементів у кожній.
  - По кожній вибірці треба порахувати середнє за M елементами.
  - У підсумку ми отримаємо вибірку з N елементів.
  - Вона і має бути з нормального розподілу.

Емпірична щільність достатньо близько збігається з теоретичним розподілом. А що якщо зменшити вибірку, за якою рахується середнє?

Стало значно гірше: з'явилися прогалини в розподілі, та й сама емпірична функція розподілу зміщена. Тож наш експеримент підтвердив важливість розміру вибірки для коректної роботи ЦГТ.

Тепер подивимось на експоненціальний розподіл.

Бачимо, що і тут усе добре працює!

3.4.2 Інші формулювання ЦГТ

Наступні формулювання є еквівалентними, тому що ми можемо перетворити одне в інше за допомогою простих алгебраїчних перетворень. Вони можуть бути корисними в різних ситуаціях, залежно від того, що ми хочемо перевірити.

$$
\begin{aligned}
&\sqrt{n}\frac{\bar{\xi}-\mu}{\sqrt{\sigma^2}} \sim \mathcal{N}(0,1) \Leftrightarrow \\
&\bar{\xi}-\mu \sim \mathcal{N}\left(0, \frac{\sigma^2}{n}\right) \Leftrightarrow \\
&\frac{\sum\limits_{i=1}^{n}\xi_i}{n} \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right) \Leftrightarrow \\
&\sum_{i=1}^{n}\xi_i \sim \mathcal{N}\left(n\cdot\mu, n\cdot\sigma^2\right)
\end{aligned}
\tag{3.5}
$$

3.5 Нормальна апроксимація й застосування Z-критерію

3.5.1 Апроксимація нормальним розподілом

Згадайте задачу на самому початку розділу 1.1. У нас є вибірка користувачів $X_1,\ X_2,\ ...,\ X_n,\ X_i \sim \text{Bernoulli}(\mu)$ з параметром μ, й ми хочемо перевірити гіпотезу:

$$H_0 : \mu = \mu_0 = 0.5 \text{ проти } H_1 : \mu > 0.5$$

де μ_0 — гіпотетичне значення параметра μ.

Раніше, ми вирішували цю задачу через біноміальний розподіл:

- $T(X^n) = \sum_{i=1}^{n} X_i,\ T \overset{H_0}{\sim} \text{Binom}(n, \mu_0)$
- Нехай реалізація $T(X^n) = t$. Тоді
- p-значення $= P_{H_0}(T(X^n) \geq t) = 1 - P_{H_0}(T(X^n) < t)$

Згадаємо, як вирішити цю задачу за допомогою Python.

А тепер подивимося, що нам говорить ЦГТ:

- За досить великого розміру вибірки $\sum_{i=1}^{n} X_i \sim \mathcal{N}\left(n \cdot \mu_0, n \cdot \sigma^2\right)$,
- $X_i \overset{H_0}{\sim} \text{Bernoulli}(\mu_0)$
- $\sigma^2 = \mu_0 \cdot (1 - \mu_0)$
- p-value $= P_{H_0}(T(X^n) \geq t)$.

При цьому цього разу ми дивимося статистику не в точці $t - 1$, як робили раніше, а в точці t, **оскільки у нас неперервний розподіл, то нам не потрібно віднімати 1**:

- у разі нормального розподілу: $P(T(X^n) \geq t) = P(T(X^n) > t) = 1 - P(T(X^n) \leq t)$;
- у разі біноміального розподілу: $P(T(X^n) \geq t) = 1 - P(T(X^n) \leq t - 1)$.

Подивимось, як це виглядає в Python. Для цього створимо функцію `get_pvalue_by_normal_approx`, яка буде приймати на вхід параметри n, μ_0, t та повертати p-значення. Порівняємо результати за точним біноміальним тестом та нашим наближенням.

Ми бачимо, що значення не дуже-то й збіглися. Але, як ми пам'ятаємо, нормальна апроксимація працює тільки з деякого великого n. Тому давайте спробуємо повторити експеримент із більшаимо кількість спостережень.

Ми бачимо, що відмінність тепер тільки в 3 знаку після коми, а не в другому, як раніше. Що більше ми братимемо вибірку, то меншою буде помилка про що говорить ЦГТ.

3.5.2 Z-критерій Фішера

Z-критерій Фішера використовується для перевірки гіпотез про математичне сподівання випадкової величини з відомою дисперсією. Він є одним із найпоширеніших критеріїв у статистиці, оскільки дозволяє оцінити, чи є різниця між середніми значеннями двох груп статистично значущою.

Для **двостороннього** критерію ми можемо використовувати Z-критерій Фішера, але з урахуванням того, що ми перевіряємо гіпотезу про те, що μ не дорівнює μ_0. Тобто, ми хочемо перевірити, чи є різниця між середніми значеннями двох груп статистично значущою в обидва боки.

Нульова та альтернативна гіпотези для двостороннього Z-критерію Фішера мають вигляд:

$$H_0 : \mu = \mu_0 \quad \text{проти} \quad H_1 : \mu \neq \mu_0 \tag{3.6}$$

де μ_0 — гіпотетичне значення параметра μ.

Статистика Z-критерію Фішера має вигляд:

$$Z = \frac{\overline{X} - \mu_0}{\sigma / \sqrt{n}} \tag{3.7}$$

де $\overline{X}$ — середнє арифметичне вибірки, σ — відома дисперсія генеральної сукупності, n — кількість спостережень.

При достатньо великій вибірці згідно ЦГТ Z-критерій Фішера має нормальний розподіл:

$$Z(X) \overset{H_0}{\sim} \mathcal{N}(0, 1) \tag{3.8}$$

Двосторонній критерій набуває вигляду:

$$P_{H_0}(Z(X) \geq z) = 1 - P_{H_0}(Z(X) < z) = 1 - \Phi(z) = \frac{\alpha}{2} \tag{3.9}$$

де α — рівень значущості.

А p-значення для двостороннього критерію розраховується за формулою:

$$p\text{-значення} = 2 \cdot \min\left[\Phi(z), 1 - \Phi(z)\right] \tag{3.10}$$

Односторонній критерій перевіряє гіпотезу про те, що μ більше або менше μ_0. Нульова та альтернативна гіпотези для одностороннього Z-критерію Фішера мають вигляд:

$$H_0 : \mu = \mu_0 \quad \text{проти} \quad H_1 : \mu > \mu_0 \tag{3.11}$$

Тоді односторонній Z-критерій Фішера має вигляд:

$$P_{H_0}(Z(X) \geq z) = 1 - P_{H_0}(Z(X) < z) = 1 - \Phi(z) = \alpha \tag{3.12}$$

де $\Phi(z)$ — функція розподілу стандартного нормального розподілу, α — рівень значущості, z — реалізація статистики Z-критерію Фішера.

3.6 Z-критерій Фішера в Python

Напишемо функцію `z_test_pvalue`, яка буде приймати на вхід параметри `sample_mean` (середнє арифметичне вибірки), `sample_size` (кількість спостережень), `population_mean` (гіпотетичне значення параметра μ), `population_variance` (дисперсія генеральної сукупності) та `alternative` (альтернативна гіпотеза). Функція буде повертати p-значення для двостороннього або одностороннього Z-критерію Фішера.

Тепер ми можемо перевірити гіпотезу про те, що μ не дорівнює μ_0, за допомогою Z-критерію Фішера. Для цього ми можемо використати функцію `z_test_pvalue` та порівняємо з результатами, які ми отримали раніше за допомогою біноміального тесту та нормальної апроксимації.

Ми бачимо, що p-значення за Z-критерієм Фішера та нормальною апроксимацією збігаються, а точний біноміальний тест дає трохи інше значення. Залишається питання: чи можна уточнити результати Z-тесту при малих вибірках? Відповідь: так, можна. Для цього існує поправка на неперервність, яка дозволяє покращити точність апроксимації і її ми розглянемо далі.

3.7 Поправка на неперервність

Задля кращого розуміння, давайте спочатку візуалізуємо p-значення в залежності від величини успіхів експерименту t для біноміального тесту та Z-критерію Фішера. Для цього побудуємо три варіанти:

- p-значення *за нормальною апроксимацією*.
  - Розрахунок в Python: `1 - norm.cdf(t)`.
- p-значення *біноміального тесту за умови, що t — нецілe число*.
  - Розглянемо на прикладі $t = 19.5$, тоді p-значення буде дорівнювати

$$\begin{aligned} P(T(X) \geq t) &= P(T(X) \geq 19.5) \\ &= 1 - P(T(X) < 19.5) \end{aligned}$$

  - Розрахунок в Python: `1 - binom.cdf(t, n, mu_0)`.
- p-значення *біноміального тесту за умови, що t — ціле число*.

– Розглянемо на прикладі $t = 19$, тоді p-значення буде дорівнювати

$$\begin{aligned} P(T(X) \geq t) &= P(T(X) \geq 19) \\ &= 1 - P(T(X) \leq 18) \end{aligned}$$

– Розрахунок в Python: `1 - binom.cdf(t - 1, n, mu_0)`.

Якщо порівняти різницю між p-значеннями біноміального та нормального розподілів, то ми отримаємо, що p-значення біноміального розподілу завжди більше за p-значення нормального розподілу. При цьому із збільшенням вибірки ця різниця зменшується. Давайте подивимось на ці різниці для різних значень t.

Для початку візьмемо $n = 20$ та $t = 10$ (Лістинг 3.13).

Тепер візьмемо $n = 20$ та $t = 16$ (Лістинг 3.14).

І накінці візьмемо $n = 200$ та $t = 100$ (Лістинг 3.15).

Ми бачимо, що з ростом вибірки різниця між p-значеннями біноміального та нормального розподілів зменшується. Але як зробити так, щоб два p-значення збіглися? Для цього слід звернути увагу на точки перетину двох ліній: біноміального та нормального розподілів. Зауважимо, що вони перетинаються приблизно на середині відрізка: між $t - 1$ та t. Тому спробуємо "змістити" графік нормального розподілу на 0.5 праворуч.

$$F_{\text{new}}(x) = F_{\text{old}}(x - 0.5) \tag{3.13}$$

Це означає, що ми повинні відняти 0.5 від x-координати точки перетину. Тобто, ми можемо використовувати поправку на неперервність, яка дозволяє покращити точність апроксимації. Тоді p-значення для біноміального розподілу буде дорівнювати:

$$p\text{-значення} = 1 - \Phi(t - 0.5) \tag{3.14}$$

де $\Phi(t - 0.5)$ — функція розподілу стандартного нормального розподілу.

Подивимось на графік з поправкою на неперервність (Лістинг 3.16).

Ми бачимо, що p-значення біноміального та нормального розподілів тепер збігаються.

Порівняємо p-значення біноміального та нормального розподілів з поправкою на неперервність (Лістинг 3.17).

Ми бачимо, що p-значення біноміального та нормального розподілів з поправкою на неперервність тепер збігаються.

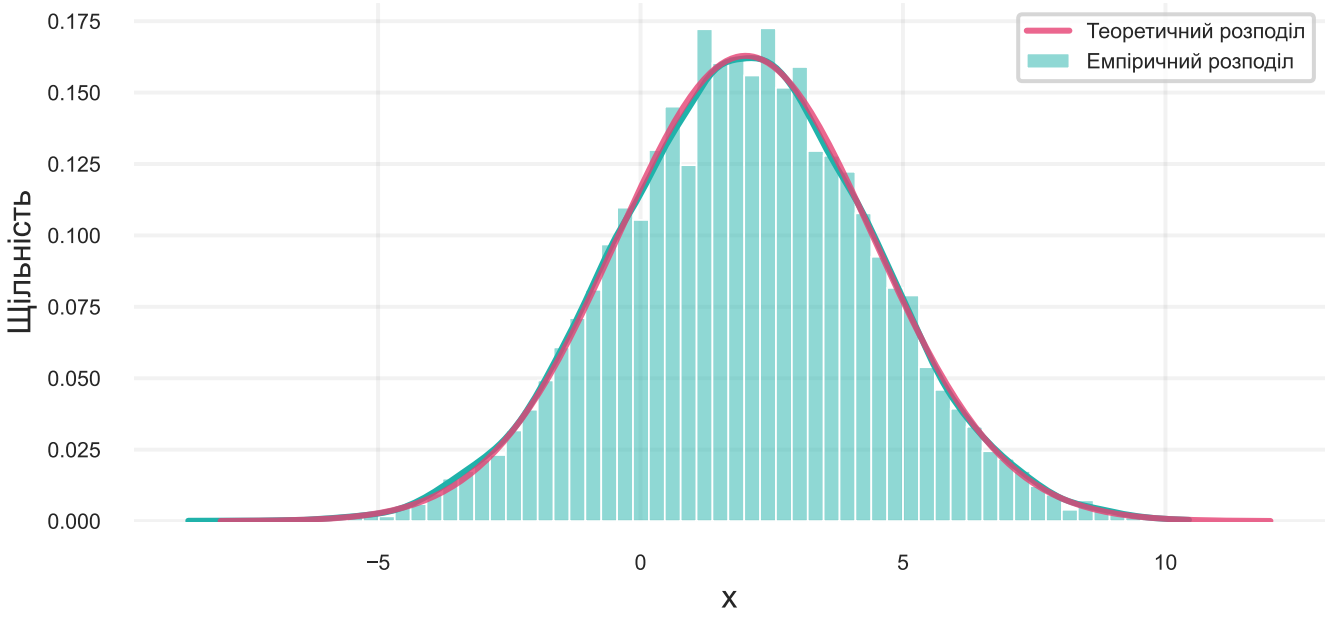
Додамо поправку на неперервність до нашої функції `z_test_pvalue` (Лістинг 3.18).

Чудово, тепер ми можемо використовувати Z-критерій Фішера з поправкою на неперервність для перевірки гіпотез про математичне сподівання випадкової величини *з відомою дисперсією*. Але що робити, якщо дисперсія невідома? Для цього існує t-критерій Стьюдента, який ми розглянемо далі.

Лістинг 3.3 Візуалізація нормального розподілу суми двох нормально розподілених випадкових величин.

```
mean_one, mean_two = 3, -1                                                    ①
var_one, var_two = 4, 2                                                       ②

n = 10000                                                                     ③

x1 = norm.rvs(loc=mean_one, scale=np.sqrt(var_one), size=n)                   ④
x2 = norm.rvs(loc=mean_two, scale=np.sqrt(var_two), size=n)

x_sum = x1 + x2                                                               ⑤
check_sum = norm(loc=mean_one + mean_two, scale=np.sqrt(var_one + var_two))   ⑥

x_grid = np.linspace(-8, 12, 1000)                                            ⑦

fig, ax = plt.subplots()
sns.histplot(x_sum, kde=True, stat='density', color=turquoise, label='Емпіричний розподіл', ax
plt.plot(x_grid, check_sum.pdf(x_grid), color=red_pink, label='Теоретичний розподіл', alpha=0.
plt.xlabel('x')
plt.ylabel('Щільність')
plt.legend()
plt.show()
```

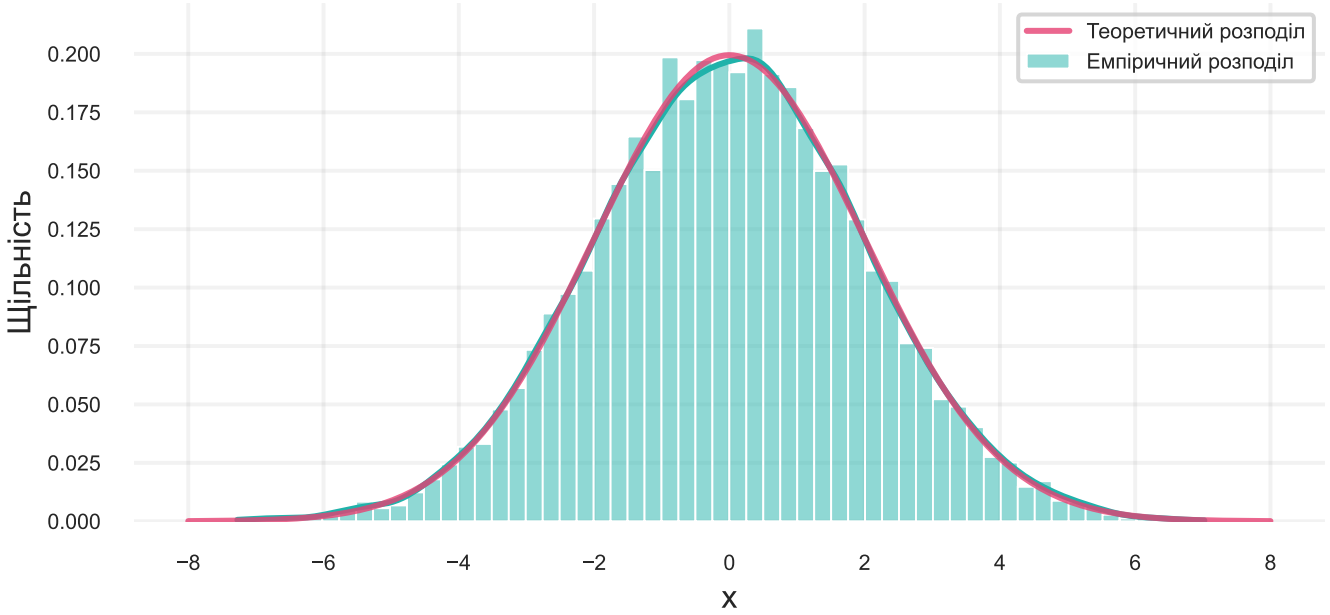
① Параметри μ_1 та μ_2.
② Параметри σ_1^2 та σ_2^2.
③ Кількість спостережень.
④ Генерація нормально розподілених випадкових величин ξ_1 та ξ_2.
⑤ Сума двох нормально розподілених випадкових величин.
⑥ Параметри суми $\xi_1 + \xi_2$.
⑦ Стандартне відхилення суми $\xi_1 + \xi_2$.
⑧ Емпіричний розподіл суми $\xi_1 + \xi_2$.
⑨ Теоретичний розподіл суми $\xi_1 + \xi_2$.



Лістинг 3.4 Візуалізація нормального розподілу множення нормально розподіленої випадкової величини на константу.

```
mean_one = 0                                                                                    ①
var_one = 1                                                                                     ②
a = 2                                                                                           ③
n = 10000                                                                                       ④
x1 = norm.rvs(loc=mean_one, scale=np.sqrt(var_one), size=n)                                     ⑤
x_mult = a * x1                                                                                 ⑥
check_mult = norm(loc=a * mean_one, scale=np.sqrt(a**2 * var_one))                              ⑦
x_grid = np.linspace(-8, 8, 1000)                                                               ⑧

fig, ax = plt.subplots()
sns.histplot(x_mult, kde=True, stat='density', color=turquoise, label='Емпіричний розподіл', a
plt.plot(x_grid, check_mult.pdf(x_grid), color=red_pink, label='Теоретичний розподіл', alpha=0
plt.xlabel('x')
plt.ylabel('Щільність')
plt.legend()
plt.show()
```

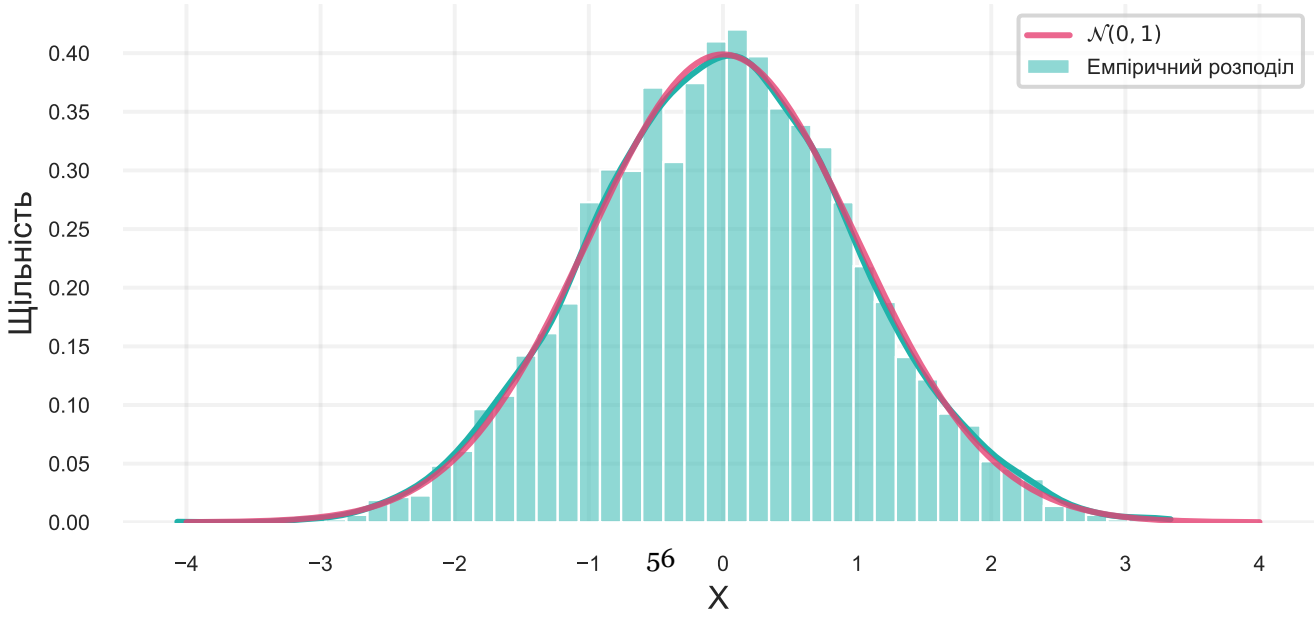
① Параметри μ_1 та σ_1^2.
② Параметри σ_1^2.
③ Константа a.
④ Кількість спостережень.
⑤ Генерація нормально розподіленої випадкової величини ξ_1.
⑥ Множення нормально розподіленої випадкової величини ξ_1 на константу a.
⑦ Параметри множення ξ_1 на константу a.
⑧ Стандартне відхилення множення ξ_1 на константу a.
⑨ Емпіричний та теоретичний розподіл множення ξ_1 на константу a.



Лістинг 3.5 Візуалізація ЦГТ при великій вибірці з біноміального розподілу.

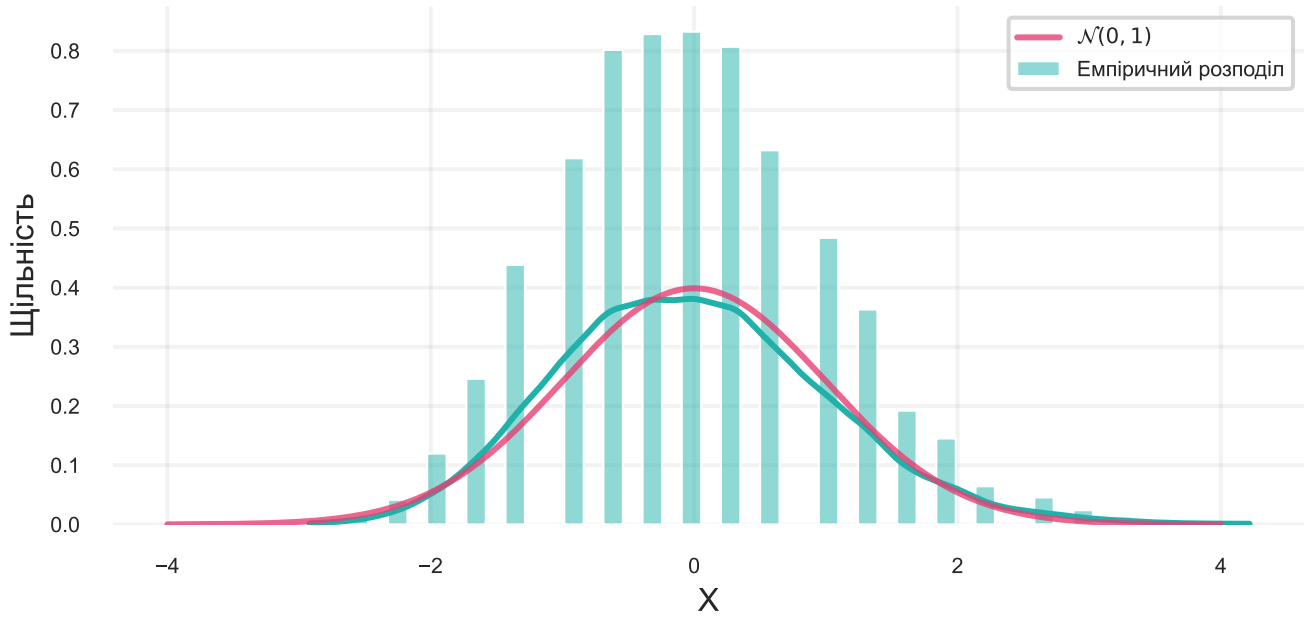
```
def visualize_CLT(sample_generator, expected_value, variance):
    np.random.seed(42)
    N = 5000                                                                    ①
    clt_sample = []                                                             ②
    for _ in range(N):
        sample = sample_generator()                                             ③
        sample_size = len(sample)                                               ④
        statistic = np.sqrt(sample_size) * (np.mean(sample) - expected_value) / np.sqrt(varian
        clt_sample.append(statistic)                                            ⑥

    x = np.linspace(-4, 4, 1000)                                                ⑦
    fig, ax = plt.subplots()
    sns.histplot(clt_sample, kde=True, stat='density', color=turquoise, label='Емпіричний розп
    ax.plot(x, norm().pdf(x), color=red_pink, label='$\mathcal{N}(0, 1)$', alpha=0.8)
    plt.legend()
    plt.xlabel('X')
    plt.ylabel('Щільність')
    plt.show()


p = 0.01
n = 20
size = 5000

visualize_CLT(lambda: np.random.binomial(n, p, size),                           ⑧
              expected_value = p * n,                                           ⑨
              variance = n * p * (1 - p)                                        ⑩
)
```

① Кількість вибірок.
② Пустий масив для зберігання статистик.
③ Генерація вибірки з M елементами.
④ Кількість елементів у вибірці.
⑤ Обчислення статистики.
⑥ Додавання статистики до масиву.
⑦ Візуалізація емпіричного розподілу та теоретичного розподілу.
⑧ Генерація вибірки з біноміального розподілу.
⑨ Математичне сподівання біноміального розподілу.
⑩ Дисперсія біноміального розподілу.



Лістинг 3.6 Візуалізація ЦГТ при малій вибірці з біноміального розподілу.

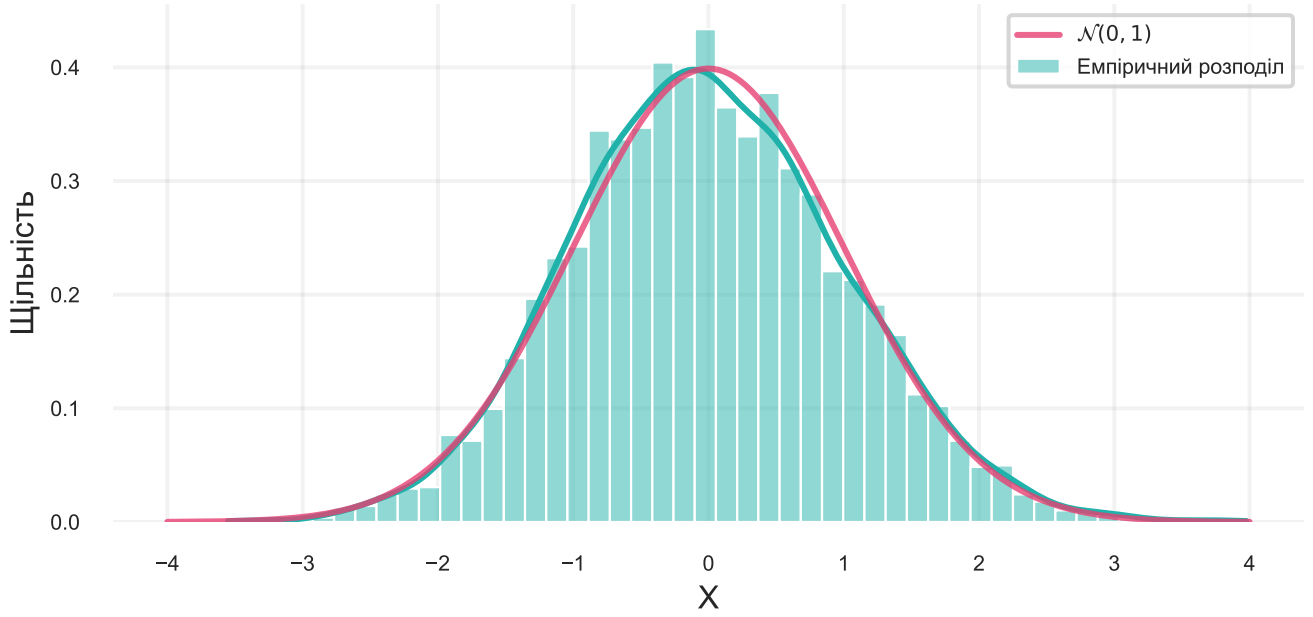
```
p = 0.05
n = 20
size = 10

visualize_CLT(lambda: np.random.binomial(n, p, size),
              expected_value = p * n,
              variance = n * p * (1 - p)
)
```



Лістинг 3.7 Візуалізація ЦГТ при великій вибірці з експоненціального розподілу.

```
p = 5                                                                    ①
size = 400                                                               ②
visualize_CLT(lambda: np.random.exponential(scale=1/p, size=size),       ③
              expected_value = 1/p,                                      ④
              variance = 1/(p**2)                                        ⑤
)
```

① Параметр λ експоненціального розподілу.
② Розмір вибірки.
③ Генерація вибірки з експоненціального розподілу.
④ Математичне сподівання експоненціального розподілу задається як $1/\lambda$.
⑤ Дисперсія експоненціального розподілу задається як $1/\lambda^2$.



Лістинг 3.8 Порівняння точного біноміального тесту та нормальної апроксимації при малій кількості спостережень.

```
def get_pvalue_by_normal_approx(t, n, mu_0):
    mu = n * mu_0                                                    ①
    sigma = np.sqrt(n * mu_0 * (1 - mu_0))                           ②
    return 1 - norm(loc=mu, scale=sigma).cdf(t)                      ③

n = 30                                                               ④
mu_0 = 0.5                                                           ⑤
t = 19                                                               ⑥

p_value = get_pvalue_by_normal_approx(t, n, mu_0)                    ⑦

print(f"p-значення за нормальною апроксимацією = {p_value:.4f}")      ⑧
print(f"p-значення за точним біноміальним тестом = {binomtest(t, n, mu_0, alternative='greater
```

① Математичне сподівання біноміального розподілу.
② Стандартне відхилення.
③ Обчислення p-значення.
④ Кількість спостережень.
⑤ Гіпотетичне значення параметра μ.
⑥ Реалізація статистики.
⑦ Обчислення p-значення.
⑧ Виведення p-значення.
⑨ Виведення точного p-значення.

```
p-значення за нормальною апроксимацією = 0.0721
p-значення за точним біноміальним тестом = 0.1002
```

Лістинг 3.9 Порівняння точного біноміального тесту та нормальної апроксимації при великій кількості спостережень.

```
n = 3000
mu_0 = 0.5
t = 1544

p_value = get_pvalue_by_normal_approx(t, n, mu_0)

print(f"p-значення за нормальною апроксимацією = {p_value:.4f}")
print(f"p-значення за точним біноміальним тестом = {binomtest(t, n, mu_0, alternative='greater
```

```
p-значення за нормальною апроксимацією = 0.0541
p-значення за точним біноміальним тестом = 0.0561
```

Лістинг 3.10 Реалізація Z-критерію Фішера в Python.

```
def z_test_pvalue(sample_mean, sample_size, population_mean, population_variance, alternative=
    z = (sample_mean - population_mean) / (np.sqrt(population_variance) / np.sqrt(sample_size)
    if alternative == 'two-sided':                                                              ②
        p_value = 2 * min(norm.cdf(z), 1 - norm.cdf(z))                                         ③
    elif alternative == 'greater':                                                              ④
        p_value = 1 - norm.cdf(z)                                                               ⑤
    elif alternative == 'less':                                                                 ⑥
        p_value = norm.cdf(z)                                                                   ⑦
    else:
        raise ValueError("Оберіть одну з альтернатив: ['two-sided', 'greater', 'less']")        ⑧
    return p_value                                                                              ⑨
```

① Обчислення статистики Z-критерію Фішера.
② Перевірка двосторонньої гіпотези.
③ Обчислення p-значення для двостороннього Z-критерію Фішера.
④ Перевірка правосторонньої гіпотези.
⑤ Обчислення p-значення для правостороннього Z-критерію Фішера.
⑥ Перевірка лівосторонньої гіпотези.
⑦ Обчислення p-значення для лівостороннього Z-критерію Фішера.
⑧ Виклик помилки, якщо альтернативна гіпотеза не відповідає жодній з можливих.
⑨ Повернення p-значення.

Лістинг 3.11 Порівняння p-значення за Z-критерієм Фішера, нормальною апроксимацією та точним біноміальним тестом.

```
n = 30
mu_0 = 0.5
t = 19
sample_mean = t / n                                                                             ①
population_variance = mu_0 * (1 - mu_0)                                                        ②

p_value = z_test_pvalue(sample_mean, n, mu_0, population_variance, alternative='greater')
print(f"p-значення за Z-критерієм Фішера = {p_value:.4f}")
print(f"p-значення за нормальною апроксимацією = {get_pvalue_by_normal_approx(t, n, mu_0):.4f}
print(f"p-значення за точним біноміальним тестом = {binomtest(t, n, mu_0, alternative='greater
```

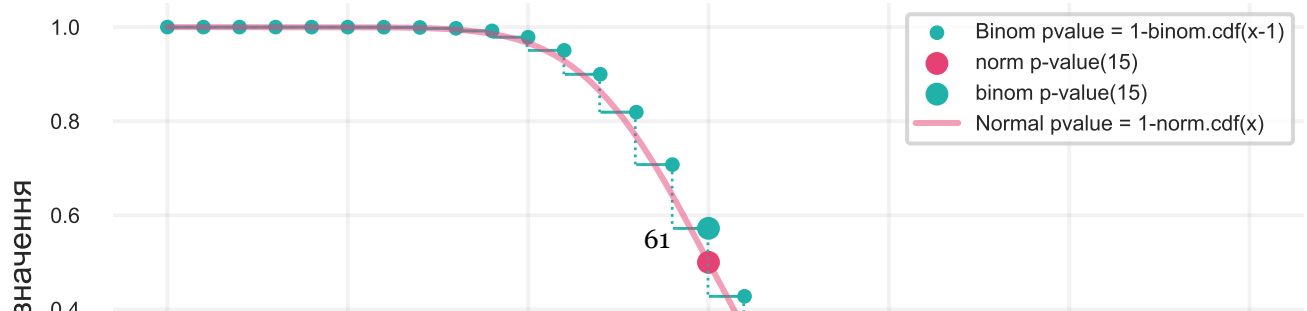
① Обчислення математичного сподівання вибірки.
② Дисперсія генеральної сукупності.

```
p-значення за Z-критерієм Фішера = 0.0721
p-значення за нормальною апроксимацією = 0.0721
p-значення за точним біноміальним тестом = 0.1002
```

Лістинг 3.12 Порівняння p-значення біноміального і нормального розподілів.

```
def cmp_pvalue_binom_and_norm(n, mu0, t, add_to_x=0):
    x_axis = np.linspace(0, n, 1000)
    dots_to_show = np.arange(0, n + 1, 1)

    add_str = "" if add_to_x == 0 else f"{add_to_x}"                                    ①

    sum_mu = n * mu0                                                                    ②
    sum_variance = n * mu0 * (1 - mu0)
    sum_std = np.sqrt(sum_variance) # 2>

    binom_dist = binom(n=n, p=mu0)                                                      ③
    norm_dist = norm(loc=sum_mu, scale=sum_std)

    plt.hlines(1 - binom_dist.cdf(x_axis[:-1]), x_axis[:-1], x_axis[1:], color=turquoise, line
    plt.vlines(x_axis[:-1], 1 - binom_dist.cdf(x_axis[:-1]), 1 - binom_dist.cdf(x_axis[1:]), c

    plt.scatter(dots_to_show, 1 - binom_dist.cdf(dots_to_show-1), color=turquoise,      ④
                alpha=1, linewidths=0.5, s=25,
                label=f'Binom pvalue = 1-binom.cdf(x-1)')

    plt.scatter(t, 1 - norm_dist.cdf(t + add_to_x), color=red_pink,                     ⑤
                alpha=1, marker='o', s=50, label=f'norm p-value({t})')

    plt.scatter(t, 1 - binom_dist.cdf(t - 1), color=turquoise, marker='o',              ⑥
                alpha=1, s=50, label=f'binom p-value({t})')

    plt.plot(x_axis, 1 - norm_dist.cdf(x_axis + add_to_x), color=red_pink, alpha=0.5,   ⑦
             label=f'Normal pvalue = 1-norm.cdf(x{add_str})')



    plt.legend()
    plt.xlabel('t')
    plt.ylabel('$p$-значення')
    plt.show()

n = 30
mu_0 = 0.5
t = 15

cmp_pvalue_binom_and_norm(n, mu_0, t)
```

① Додатковий доданок до x-координати (про нього ми поговоримо пізніше).
② Параметри нормального розподілу.
③ Створення біноміального та нормального розподілів.
④ p-значення біноміального розподілу.
⑤ p-значення нормального розподілу.
⑥ p-значення біноміального розподілу у точці t.
⑦ p-значення нормального розподілу у точці t.



Лістинг 3.13 Порівняння p-значення біноміального та нормального розподілів при малому $n = 20$ та $t = 10$.

```
n = 20
t = 10
mu_0 = 0.5

binom_pvalue = 1 - binom(n, mu_0).cdf(t - 1)                                    ①
norm_pvalue = 1 - norm(loc=n * mu_0, scale=np.sqrt(n * mu_0 * (1 - mu_0))).cdf(t)  ②
diff = binom_pvalue - norm_pvalue                                               ③

print(f"p-значення біноміального розподілу = {binom_pvalue:.4f}")
print(f"p-значення нормального розподілу = {norm_pvalue:.4f}")
print(f"Різниця між p-значеннями = {diff:.4f}")
```

① p-значення біноміального розподілу.
② p-значення нормального розподілу.
③ Різниця між p-значеннями.

```
p-значення біноміального розподілу = 0.5881
p-значення нормального розподілу = 0.5000
Різниця між p-значеннями = 0.0881
```

Лістинг 3.14 Порівняння p-значення біноміального та нормального розподілів при малому $n = 20$ та $t = 16$.

```
n = 20
t = 16
mu_0 = 0.5

binom_pvalue = 1 - binom(n, mu_0).cdf(t - 1)
norm_pvalue = 1 - norm(loc=n * mu_0, scale=np.sqrt(n * mu_0 * (1 - mu_0))).cdf(t)
diff = binom_pvalue - norm_pvalue

print(f"p-значення біноміального розподілу = {binom_pvalue:.4f}")
print(f"p-значення нормального розподілу = {norm_pvalue:.4f}")
print(f"Різниця між p-значеннями = {diff:.4f}")
```

```
p-значення біноміального розподілу = 0.0059
p-значення нормального розподілу = 0.0036
Різниця між p-значеннями = 0.0023
```

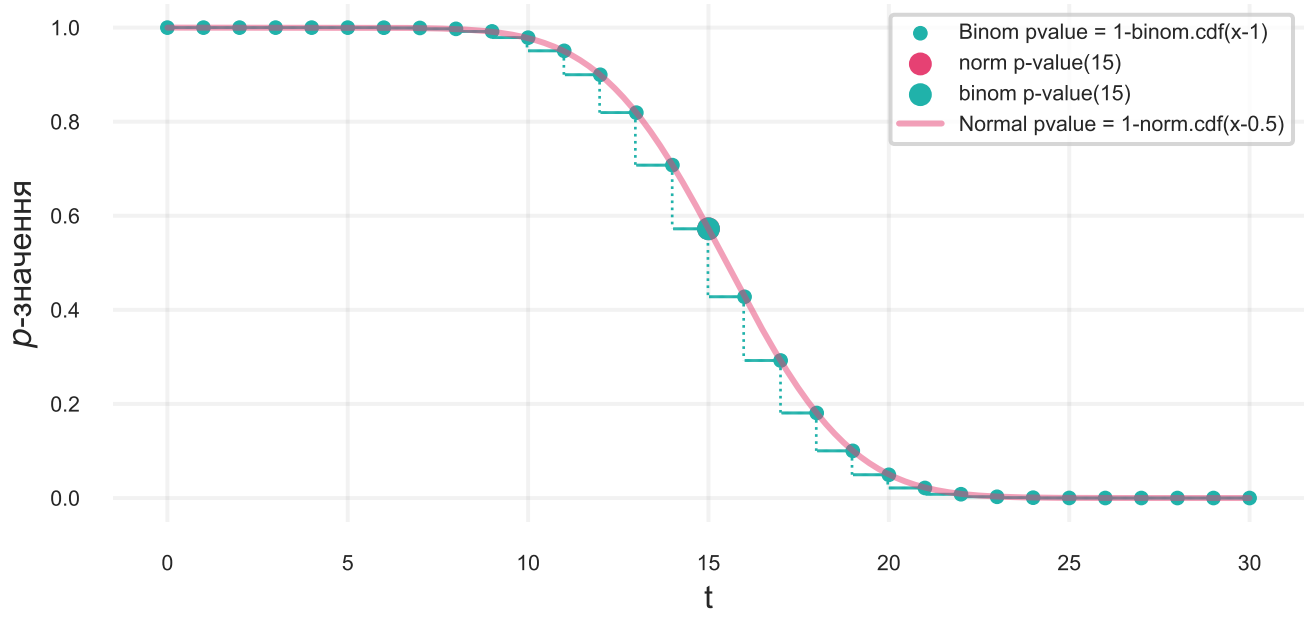
Лістинг 3.15 Порівняння p-значення біноміального та нормального розподілів при великому $n = 200$ та $t = 100$.

```
n = 200
t = 100
mu_0 = 0.5

binom_pvalue = 1 - binom(n, mu_0).cdf(t - 1)
norm_pvalue = 1 - norm(loc=n * mu_0, scale=np.sqrt(n * mu_0 * (1 - mu_0))).cdf(t)
diff = binom_pvalue - norm_pvalue

print(f"p-значення біноміального розподілу = {binom_pvalue:.4f}")
print(f"p-значення нормального розподілу = {norm_pvalue:.4f}")
print(f"Різниця між p-значеннями = {diff:.4f}")
```

```
p-значення біноміального розподілу = 0.5282
p-значення нормального розподілу = 0.5000
Різниця між p-значеннями = 0.0282
```

Лістинг 3.16 Порівняння p-значення біноміального та нормального розподілів з поправкою на неперервність.

```
cmp_pvalue_binom_and_norm(30, 0.5, 15, add_to_x=-0.5)
```



Лістинг 3.17 Порівняння p-значення біноміального та нормального розподілів з поправкою на неперервність при $n = 30$ та $t = 19$.

```
n = 30
t = 19
mu_0 = 0.5

binom_pvalue = 1 - binom(n, mu_0).cdf(t - 1)                                    ①
norm_pvalue = 1 - norm(loc=n * mu_0, scale=np.sqrt(n * mu_0 * (1 - mu_0))).cdf(t)   ②
norm_pvalue_correct = 1 - norm(loc=n * mu_0, scale=np.sqrt(n * mu_0 * (1 - mu_0))).cdf(t - 0.5

print(f"p-значення біноміального розподілу = {binom_pvalue:.4f}")
print(f"p-значення нормального розподілу = {norm_pvalue:.4f}")
print(f"p-значення нормального розподілу з поправкою на неперервність = {norm_pvalue_correct:.
```

① p-значення біноміального розподілу.
② p-значення нормального розподілу.
③ p-значення нормального розподілу з поправкою на неперервність.

```
p-значення біноміального розподілу = 0.1002
p-значення нормального розподілу = 0.0721
p-значення нормального розподілу з поправкою на неперервність = 0.1006
```

Лістинг 3.18 Функція Z-критерію Фішера з поправкою на неперервність.

```
def z_test_pvalue(sample_mean, sample_size, population_mean, population_variance, alternative=
    if continuity_correction:                                                              ①
        sample_mean = (sample_mean * sample_size - 1/2) / sample_size
    z = (sample_mean - population_mean) / (np.sqrt(population_variance) / np.sqrt(sample_size)
    if alternative == 'two-sided':
        p_value = 2 * min(norm.cdf(z), 1 - norm.cdf(z))
    elif alternative == 'greater':
        p_value = 1 - norm.cdf(z)
    elif alternative == 'less':
        p_value = norm.cdf(z)
    else:
        raise ValueError("Оберіть одну з альтернатив: ['two-sided', 'greater', 'less']")
    return p_value

n = 30
t = 19
mu0 = 0.5
variance = mu0 * (1 - mu0)

p_value = z_test_pvalue(t / n, n, mu0, variance, alternative='greater', continuity_correction=
print(f"p-значення за Z-критерієм Фішера з поправкою на неперервність = {p_value:.4f}") (14)
```

① Перевірка наявності поправки на неперервність.

```
p-значення за Z-критерієм Фішера з поправкою на неперервність = 0.1006
```

Chapter 4

t-критерій Стьюдента

4.1 Основні положення

Спробуємо розв'язати таке завдання.

Приклад 4.1.

> Менеджмент компанії розглядає новий підхід до планування щотижневих нарад, щоб зменшити втрати часу співробітників. Раніше середня тривалість таких нарад складала 70 хвилин. Ідея полягає в тому, щоб перейти до нової структури нарад, яка, за задумом, дозволить зменшити тривалість нарад до 60 хвилин.
>
> Протягом одного тижня провели 7 нарад у новому форматі й зафіксували їх тривалість. Якщо з'ясується, що нові наради тривають довше, ніж у середньому 70 хвилин, новий формат вважатимуть неефективним.
>
> Ваше завдання — перевірити, чи новий формат нарад дійсно ефективніший.
>
> Вийшла вибірка середньої тривалості нарад (в хвилинах): `[50, 55, 70, 45, 40, 70, 80]`.

Для початку переформулюємо умову мовою математики. Є вибірка:

- $X_1, X_2, ..., X_7$ — значення середньої тривалості нарад у новому форматі;
- Будемо вважати, що X з нормального розподілу, тобто $X \sim N(\mu, \sigma^2)$.

```
meeting_time = np.array([50, 55, 70, 45, 40, 70, 80])
print(f"Середнє значення: {np.mean(meeting_time):.2f} хвилин")
```

```
Середнє значення: 58.57 хвилин
```

Наша гіпотеза звучить так:

$$H_0 : E\overline{X} \geq 70 \text{ проти } H_1 : E\overline{X} < 70$$

Здається, що ми таке вже вміємо вирішувати: згадаємо про Z-критерій:

$$H_0 : \mu \leq \mu_0 \text{ проти } H_1 : \mu > \mu_0$$

- Статистика $Z(X) = \sqrt{n}\dfrac{\overline{X} - \mu_0}{\sqrt{\sigma^2}}$
- За досить великого розміру вибірки $Z(X) \overset{H_0}{\sim} \mathcal{N}(0,1)$ (за ЦГТ)
- Односторонній критерій: $\{Z(X) \geq z_{1-\alpha}\}$
  - p-значення = $1 - \Phi(z)$, де z — реалізація статистики $Z(X)$, $\Phi(z)$ — функція розподілу $\mathcal{N}(0,1)$

- Двосторонній критерій: $\left\{Z(X) \geq z_{1-\frac{\alpha}{2}}\right\} \bigcup \left\{Z(X) \leq -z_{1-\frac{\alpha}{2}}\right\}$
  - p-значення = $2 \cdot \min\left[\Phi(z), 1 - \Phi(z)\right]$, де z — реалізація статистики $Z(X)$

Тоді треба лише порахувати таку статистику: $\sqrt{n}\dfrac{\overline{X} - 70}{\sqrt{\sigma^2}} \overset{H_0}{\sim} \mathcal{N}(0, 1)$.

Але є суттєва проблема: **ми не знаємо** σ^2! Тому ми не можемо використовувати Z-критерій.

Давайте спробуємо оцінити σ^2 за допомогою вибірки. Для цього скористаємося формулою:

$$\hat{\sigma}^2 = S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2 \tag{4.1}$$

Вона називається **вибірковою дисперсією**. Вибіркова дисперсія є *незміщеною* та *консистентною* оцінкою дисперсії генеральної сукупності.

Вибіркова дисперсія є **незміщеною**, оскільки ми ділимо на $n - 1$, а не на n. Це робиться для того, щоб уникнути систематичної помилки в оцінці дисперсії. **Консистентність** пояснюється тим, що з ростом вибірки n ми все ближче підходимо до істинної дисперсії генеральної сукупності.

Для розрахунку вибіркової дисперсії в Python можна скористатися функцією `np.var` з параметром `ddof=1`, що означає, що ми ділимо на $n - 1$.

```
meeting_time_var = np.var(meeting_time, ddof=1)
print(f"Вибіркова дисперсія: {meeting_time_var:.2f} хвилин")
```

```
Вибіркова дисперсія: 222.62 хвилин
```

Давайте введемо новий критерій t'-тест, у якому ми підставимо:

- $t(X) := \sqrt{n}\dfrac{\overline{X} - \mu_0}{\sqrt{S^2}}$
- $t(X) \overset{H_0}{\sim} \mathcal{N}(0, 1)$

Залишилося перевірити: **Чи правда, що при** H_0 **розподіл** t**-статистики — стандартний нормальний?**

Для цього пропонується подивитися, як насправді буде розподілена статистика $t(X) = \sqrt{n}\dfrac{\overline{X} - \mu_0}{\sqrt{S^2}}$ у завданні, яке було поставлено від початку.

Для цього будемо вважати, що вибірка X складається з 7 елементів й $X \sim \mathcal{N}$.

- Ми M раз згенеруємо вибірку X та порахуємо щоразу статистику $t(X)$.
- У підсумку ми отримаємо вибірку розміру M для $t(X)$ й зможемо побудувати гістограму розподілу. Окремо побудуємо розподіл $\mathcal{N}(0, 1)$. Якщо емпіричний розподіл візуально збіжиться з теоретичним нормальним, значить, усе добре. А якщо ні, то так просто ми не можемо замінити σ^2 на S^2.
  - Додатково подивимося, що буде, якщо замінити $t(X)$ на $Z(X)$. Добре, що на штучному прикладі ми знаємо дисперсію.

Для цього ми напишемо функцію `sample_statistics`, яка зможе побудувати розподіл для будь-якої статистики, а не тільки для $t(X), Z(X)$. Вона приймає на вхід:

- `number_of_experiments` — кількість експериментів, які ми хочемо провести;
- `statistic_function` — функція, яка обчислює статистику;
- `sample_size` — розмір вибірки;
- `sample_distr` — розподіл, з якого ми генеруємо вибірку.

```
def sample_statistics(number_of_experiments, statistic_function, sample_size, sample_distr):
    statistic_sample = []
    for _ in range(number_of_experiments):
```

```
        sample = sample_distr.rvs(sample_size)
        statistic = statistic_function(sample)
        statistic_sample.append(statistic)
    return statistic_sample
```

Тепер перевіримо, чи дійсно $t(X)$ розподілена нормально. Для цього скористаємося функцією `sample_statistics` та побудуємо гістограму для $t(X)$. Генерувати вибірку будемо з нормального розподілу $\mathcal{N}(5, 3^2)$.

```
sample_size = 7
M = 100000
sample_distr = norm(loc=5, scale=3)

T_X = lambda sample: np.sqrt(sample_size) * (np.mean(sample) - sample_distr.mean()) / np.sqrt(
Z_X = lambda sample: np.sqrt(sample_size) * (np.mean(sample) - sample_distr.mean()) / sample_d

samples = {
    "T(X)": sample_statistics(
    number_of_experiments=M, statistic_function=T_X,
    sample_size=sample_size, sample_distr=sample_distr),

    "Z(X)": sample_statistics(
    number_of_experiments=M, statistic_function=Z_X,
    sample_size=sample_size, sample_distr=sample_distr)
}

for i, name in enumerate(["T(X)", "Z(X)"]):
    plt.subplot(1, 2, i + 1)
    current_sample = samples[name]
    l_bound, r_bound = np.quantile(current_sample, [0.001, 0.999])

    x = np.linspace(l_bound, r_bound, 1000)
    sns.distplot(current_sample, label='Емпіричний розподіл', color=turquoise)
    plt.plot(x, norm(0, 1).pdf(x), label='$\mathcal{N}(0, 1)$', color=red_pink)
    plt.legend(loc='upper left')
    plt.xlabel(f'{name}')
    plt.xlim((l_bound, r_bound))
    plt.ylabel('Щільність')
    plt.grid(linewidth=0.2)

plt.show()
```

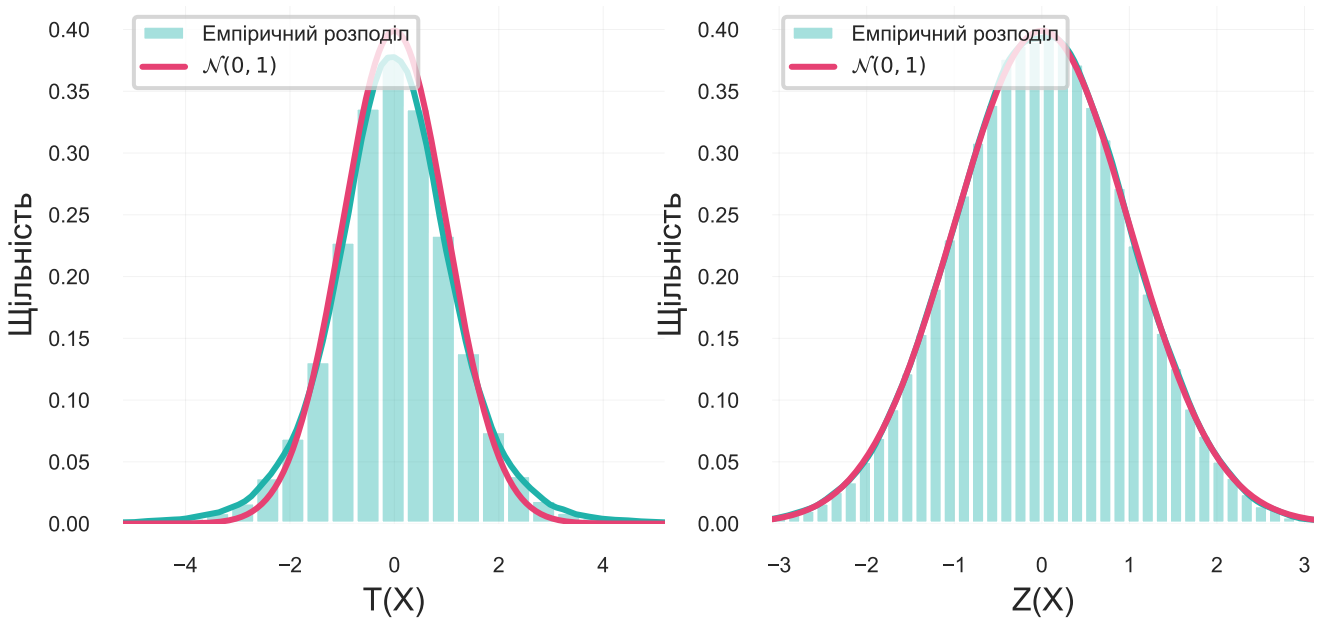


Рисунок 4.1: Симуляція розподілу $t(X)$ та $Z(X)$

Ми бачимо, що:

- Z-тест тут працює: $\sqrt{n}\frac{\overline{X} - \mu_0}{\sqrt{\sigma^2}} \sim \mathcal{N}(0, 1)$.
- Але ось для $t(X)$ це не так! **Вони відрізняються! А значить t'-критерій не підходить для початкової задачі!

Для того щоб стало зрозуміло, чому так сталося, розглянемо $t(X)$ у деталях. При створенні критерію є два кроки:

1. Придумати статистику для критерію
   - Із цим ми успішно впоралися, придумавши $t(X)$.
2. Зрозуміти розподіл статистики.
   - І ось це найскладніший крок, який не дозволяє використовувати будь-яку придуману статистику. Потрібно також розуміти її розподіл.
   - І з цим, як ми побачили, ми провалилися для $t(X)$. Нормальний розподіл не підійшов.

Але чому $t(X) = \sqrt{n}\frac{\overline{X} - \mu}{\sqrt{S^2}}$ не розподілена нормально, хоча $\sqrt{n}\frac{\overline{X} - \mu}{\sqrt{\sigma^2}} \overset{H_0}{\sim} \mathcal{N}(0, 1)$$? Чому при заміні σ^2 на S^2 усе зіпсувалося?

Справа в тому, що S^2 — це випадкова величина! Згадаймо, як ми виводили Z-критерій:

1. Ми порахували, що $\overline{X} \sim \mathcal{N}(\mu, \sigma^2)$. З ЦГТ або, у випадку вище, з властивостей нормального розподілу.
2. Далі, все також із властивостей цього розподілу випливає, що якщо ми віднімемо константу або поділимо на константу, то нормальний розподіл не перетвориться на інший: тому $\sqrt{n}\frac{\overline{X} - \mu_0}{\sqrt{\sigma^2}} \sim \mathcal{N}(0, 1)$.

Але ми нічого не знаємо про $\frac{\overline{X}}{\sqrt{\eta}}$, де $\overline{X} \sim \mathcal{N}, S^2 := \eta \sim P$, де P невідомо. Ми не знаємо поки що жодних теорем, які б хоч якось доводили, що тут також залишиться нормальний розподіл.

Давайте подивимося на розподіл $\sqrt{S^2}$ на все тому ж нормальному розподілі.

```
S2 = lambda sample: np.std(sample, ddof=1)
S2_sample = sample_statistics(
```

```
    number_of_experiments=M, statistic_function=S2,
    sample_size=sample_size, sample_distr=sample_distr
)

sns.distplot(S2_sample, label='Емпіричний розподіл', color=turquoise)
plt.legend()
plt.xlabel('$\sqrt{S^2}$')
plt.ylabel('Щільність')
plt.grid(linewidth=0.2)
plt.show()
```

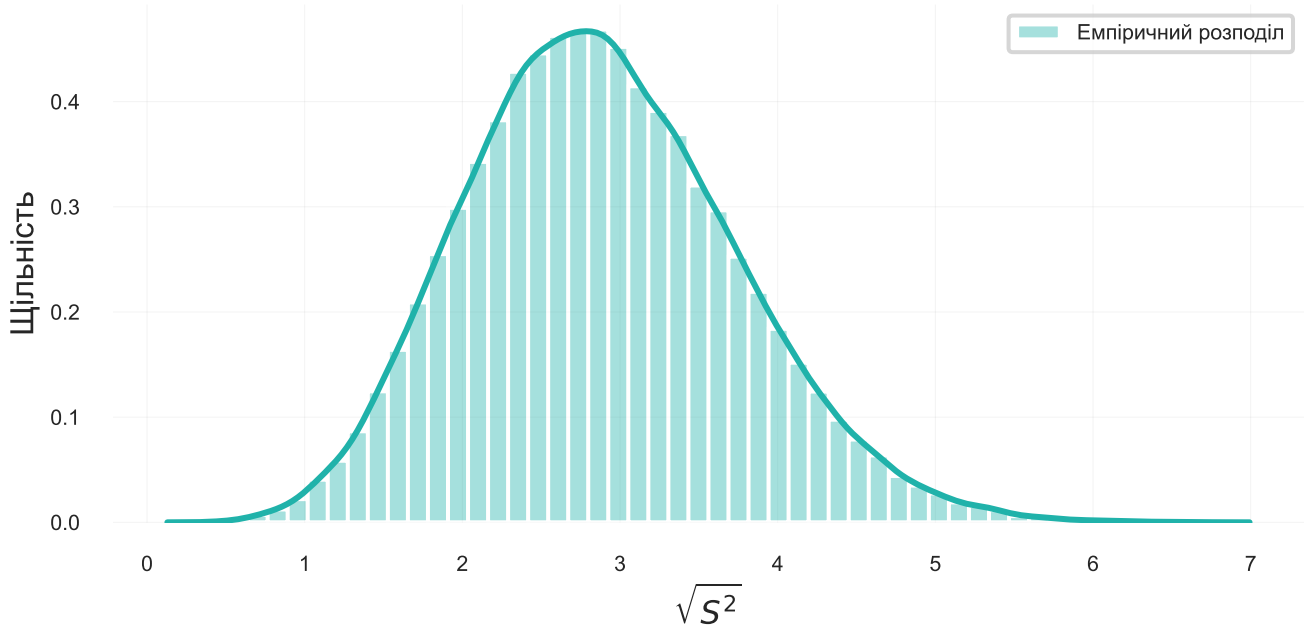


Рисунок 4.2: Розподіл $\sqrt{S^2}$

Розподіл $\sqrt{S^2}$ несиметричний й незрозуміло як розподілений. Тому, коли ми якусь величину з нормального розподілу ділимо на несиметричний незрозумілий розподіл, ми й отримуємо, що наша статистика t не з нормального розподілу.

Тож давайте виведемо критерій, який допоможе розв'язати початкову задачу!

4.2 t-тест Стьюдента

Для того щоб вивести t-тест, нам потрібно зрозуміти, як розподіляється статистика $t(X) = \sqrt{n}\frac{\overline{X} - \mu_0}{\sqrt{S^2}}$. Для того, щоб це дізнатися, нам знадобиться кілька фактів:

1. Нехай $X_1 \dots X_n \sim \mathcal{N}(\mu, \sigma^2)$
2. Нехай $\xi_1 \dots \xi_n \sim \mathcal{N}(0, 1)$. Тоді $\eta = \xi_1^2 + \dots + \xi_n^2 \sim \chi_n^2$[1].
    - Тоді $\sum_{i=1}^{n} \left(\xi_i - \overline{\xi}\right)^2 \sim \chi_{n-1}^2$ [2].

[1] Розподіл χ^2 — це розподіл суми квадратів k незалежних нормальних випадкових величин з нульовим математичним сподіванням. Тобто, якщо $X_1, X_2, \dots, X_k \sim \mathcal{N}(0, 1)$, то $Y = X_1^2 + X_2^2 + \dots + X_k^2 \sim \chi_k^2$.

[2] Доведення Cochran (1934).

- $S_X^2 = \dfrac{1}{n-1}\sum\limits_{i=1}^{n}(X_i - \overline{X})^2$
- $\xi_i := \dfrac{X_i - \mu}{\sigma} \sim \mathcal{N}(0,1)$. Тоді

$$S_\xi^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(\xi_i - \overline{\xi}\right)^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{X_i-\mu}{\sigma} - \sum_{i=1}^{n}\left[\frac{X_i-\mu}{n\sigma}\right]\right)^2 =$$
$$= \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{X_i}{\sigma} - \frac{\mu}{\sigma} - \sum_{i=1}^{n}\left[\frac{X_i}{n\sigma}\right] + \frac{n\mu}{n\sigma}\right)^2 =$$
$$= \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{X_i}{\sigma} - \frac{\overline{X}_i}{\sigma}\right)^2 = \frac{1}{\sigma\cdot(n-1)}\sum_{i=1}^{n}\left(X_i - \overline{X}_i\right)^2 = \frac{1}{\sigma}S_X^2$$

- А значить $\dfrac{(n-1)\cdot S_X^2}{\sigma^2} = \sum\limits_{i=1}^{n}\left(\xi_i - \overline{\xi}\right)^2 \sim \chi^2_{n-1}$

3. Нехай $\xi \sim \mathcal{N}(0,1), \eta \sim \chi^2_k$ і ξ з η незалежні. Тоді статистика $\zeta = \dfrac{\xi}{\sqrt{\eta/k}} \sim t_k$ — з розподілу Стьюдента[3] з k ступенями свободи.
   - $\xi := \sqrt{n}\dfrac{\overline{X} - \mu_0}{\sigma} \sim \mathcal{N}(0,1)$
   - $\eta := \dfrac{(n-1)\cdot S_X^2}{\sigma^2} \sim \chi^2_{n-1}$
   - ξ и η незалежні[4].
   - Тоді

$$t = \sqrt{n}\frac{\overline{X}-\mu_0}{\sqrt{S^2}} = \frac{\sqrt{n}\dfrac{\overline{X}-\mu_0}{\sigma}}{\sqrt{\dfrac{(n-1)\cdot S_X^2}{(n-1)\sigma^2}}} = \frac{\xi}{\sqrt{\dfrac{\eta}{n-1}}} \sim t_{n-1}$$

У підсумку, статистика $t = \sqrt{n}\dfrac{\overline{X}-\mu_0}{\sqrt{S^2}} \sim t_{n-1}$ — взята з розподілу Стьюдента з $n-1$ ступенем свободи. **Але тільки в разі, якщо початкова вибірка з нормального розподілу!**

Тепер нам достатньо даних, щоб побудувати t-тест:

$$H_0 : \mu = \mu_0,\ X \sim \mathcal{N}\ \ H_1 : \mu > \mu_0 \tag{4.2}$$

Статистика $t(X)$ буде виглядати так:

$$t(X) = \sqrt{n}\frac{\overline{X}-\mu_0}{\sqrt{S^2}} \sim t_{n-1} \tag{4.3}$$

Тоді односторонній критерій набуває вигляду:

$$\left\{t(X) \geq t_{n-1,1-\alpha}\right\} \tag{4.4}$$

А p-значення для одностороннього критерію можна обчислити так:

$$p\text{-значення} = 1 - \tau_{n-1}(z), \tag{4.5}$$

[3] Розподіл Стьюдента — це розподіл, який виникає при нормальному розподілі з невідомою дисперсією. Якщо $X_1, X_2, \ldots, X_n \sim \mathcal{N}(\mu, \sigma^2)$, то $t = \dfrac{\overline{X}-\mu}{S/\sqrt{n}} \sim t_{n-1}$, де $\overline{X}$ — вибіркове середнє, S — вибіркова стандартна девіація.

[4] Доведення Basu (1955).

де z — реалізація статистики $t(X)$, $\tau_{n-1}(z)$ — функція розподілу t_{n-1}

Двосторонній критерій буде виглядати так:

$$\left\{t(X) \geq t_{n-1,1-\frac{\alpha}{2}}\right\} \bigcup \left\{t(X) \leq -t_{n-1,1-\frac{\alpha}{2}}\right\} \tag{4.6}$$

При цьому p-значення для двостороннього критерію можна обчислити так:

$$p\text{-значення} = 2 \cdot \min\left[\tau_{n-1}(z), 1 - \tau_{n-1}(z)\right], \tag{4.7}$$

де z — реалізація статистики $t(X)$, $\tau_{n-1}(z)$ — функція розподілу t_{n-1}.

4.3 t-тест у Python

Давайте тепер протестуємо всі наші теоретичні дослідження на практиці. Для цього нам знадобляться наступні бібліотеки функції:

- `scipy.stats.chii2` — для розподілу χ^2;
- `scipy.stats.t` — для t розподілу Стюдента;
- `scipy.stats.ttest_1samp` — для t-тесту.

Подивимось на розподіл χ^2 та розподіл η.

```
sample_size = 7
sample_distr = norm(loc=5, scale=3)
sample = sample_distr.rvs(sample_size)
M = 10000

eta_statistic = lambda sample: np.var(sample, ddof=1) * (sample_size - 1) / sample_distr.var()
eta_sample = sample_statistics(
    number_of_experiments=M, statistic_function=eta_statistic,
    sample_size=sample_size, sample_distr=sample_distr
)

chi2_dist = chi2(df=sample_size-1)

l_bound, r_bound = np.quantile(eta_sample, [0.001, 0.999])
x = np.linspace(l_bound, r_bound, 1000)

sns.distplot(eta_sample, label='Емпіричний розподіл', color=turquoise)
plt.plot(x, chi2_dist.pdf(x), label='$\chi^2$', color=red_pink)
plt.legend()
plt.xlabel('$\eta$')
plt.ylabel('Щільність')
plt.grid(linewidth=0.2)
plt.show()
```

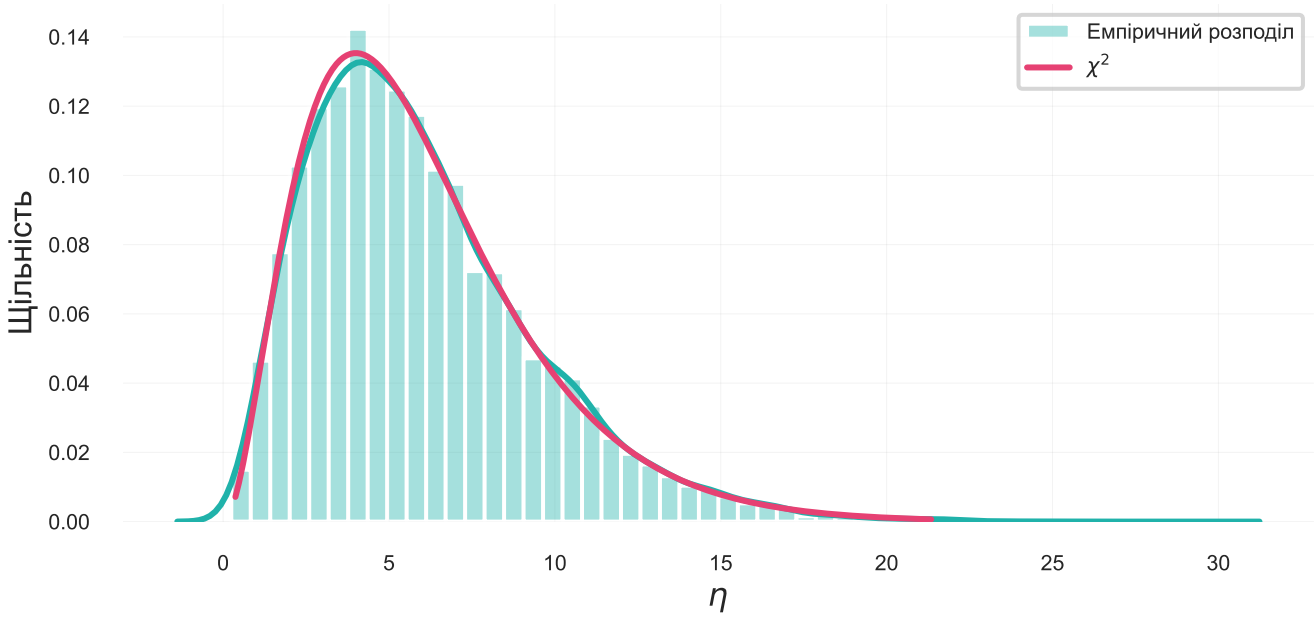


Рисунок 4.3: Розподіл емпіричного η та теоретичного χ^2.

Ми бачимо, що емпіричний розподіл η та теоретичний χ^2 збігаються. Це означає, що ми можемо використовувати t-тест для перевірки гіпотези.

Тепер перевіримо, чи дійсно $t(X)$ описується розподілом Стьюдента. Для цього скористаємося функцією `sample_statistics` та побудуємо гістограму для $t(X)$. Генерувати вибірку будемо з нормального розподілу $\mathcal{N}(5, 3^2)$.

```
sample_size = 7
sample_distr = norm(loc=5, scale=3)
sample = sample_distr.rvs(sample_size)
M = 10000

T_X = lambda sample: np.sqrt(sample_size) * (np.mean(sample) - sample_distr.mean()) / np.std(s
T_sample = sample_statistics(
    number_of_experiments=M, statistic_function=T_X,
    sample_size=sample_size, sample_distr=sample_distr
)


T_dist = t(df=sample_size-1)

l_bound, r_bound = np.quantile(T_sample, [0.001, 0.999])
x = np.linspace(l_bound, r_bound, 1000)

sns.distplot(T_sample, color=turquoise, label='Емпіричний розподіл')
plt.plot(x, T_dist.pdf(x), c=red_pink, label='$t_{n-1}$')
plt.plot(x, norm(0, 1).pdf(x), c=slate, linestyle='--', label='$\mathcal{N}(0, 1)$')
plt.legend()
plt.xlabel('$t(X)$')
plt.ylabel('Щільність')
plt.xlim((l_bound, r_bound))
plt.grid(linewidth=0.2)
```

```
plt.show()
```

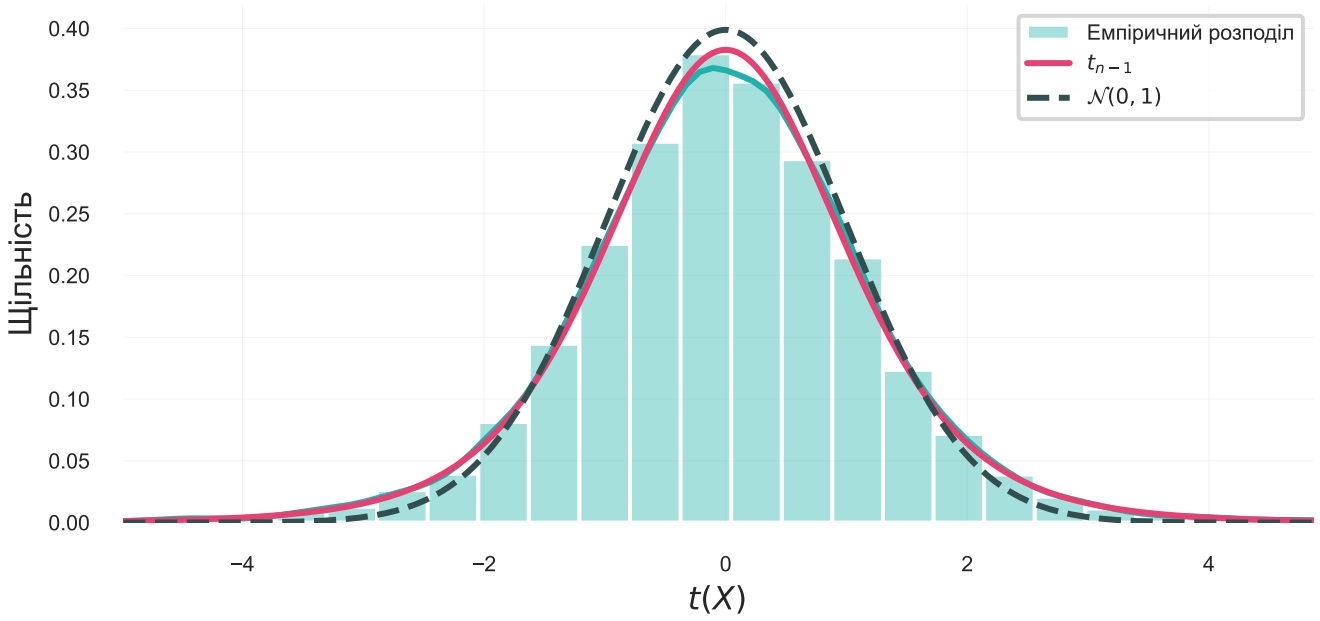


Рисунок 4.4: Розподіл $t(X)$

Розподіл Стьюдента практично ідеально описує дані, тоді як нормальний розподіл більш “центрований”.

Тепер, як викликати вбудований t-тест у Python? Для цього скористаємося функцією `scipy.stats.ttest_1samp`. Вона приймає на вхід:

- `a` — вибірка;
- `popmean` — середнє значення генеральної сукупності, яке ми хочемо перевірити;
- `axis` — вздовж якої осі обчислювати тест. За замовчуванням `0`;
- `nan_policy` — як обробляти NaN. Може приймати значення `propagate`, `raise`, `omit`. За замовчуванням `propagate`;
- `alternative` — альтернативна гіпотеза. Може приймати значення `two-sided`, `less`, `greater`. За замовчуванням `two-sided`.

```
meeting_time = np.array([50, 55, 70, 45, 40, 70, 80])

ttest_result = ttest_1samp(meeting_time, 70, alternative='less')
print(f"Статистика: {ttest_result.statistic:.2f}")
print(f"p-значення: {ttest_result.pvalue:.2f}")
```

```
Статистика: -2.03
p-значення: 0.04
```

Оскільки p-значення менше 0.05, то ми відхиляємо нульову гіпотезу. Це означає, що середня тривалість нарад у новому форматі триває менше 70 хвилин. Відповідно до t-тесту, ми можемо стверджувати, що новий формат нарад дійсно скорочує їх тривалість.

4.4 Довірчі інтервали

Давайте тепер розглянемо, як можна оцінити параметри генеральної сукупності за допомогою t-тесту. Розглянемо два виведення довірчого інтервалу.

4.4.1 Перший метод

Нехай є статистика Q та критерій $\psi(Q)$ для перевірки гіпотези $H_0 : \theta = m$ рівня значущості α.

Тоді довірчий інтервал для θ рівня довіри $1-\alpha$: множина таких m, що критерій $\psi(Q)$ не відкидає для них H_0.

Нехай μ — істинне середнє вибірки. Ми також знаємо, що за $H_0 : \sqrt{n}\dfrac{\overline{X}-m}{\sqrt{S^2}} \sim t_{n-1}$.

Нас цікавлять такі m, що: $\left\{-t_{n-1,1-\frac{\alpha}{2}} < \sqrt{n}\dfrac{\overline{X}-m}{\sqrt{S^2}} < t_{n-1,1-\frac{\alpha}{2}}\right\}$, у цьому разі критерій не відкинеться.

Розпишемо, щоб у центрі залишилося тільки m: $\left\{\overline{X} - \dfrac{t_{n-1,1-\alpha/2}\sqrt{S^2}}{\sqrt{n}} < m < \overline{X} + \dfrac{t_{n-1,1-\alpha/2}\sqrt{S^2}}{\sqrt{n}}\right\}$. А отже, наш довірчий інтервал:

$$CI_\mu = \left(\overline{X} \pm \frac{t_{n-1,1-\alpha/2}\sqrt{S^2}}{\sqrt{n}}\right), \tag{4.8}$$

де $S^2 = \dfrac{1}{n-1}\sum\limits_{i=1}^{n}(X_i - \overline{X})^2$

4.4.2 Другий метод

Довірчим інтервалом для параметра θ рівня довіри $1-\alpha$ є пара статистик $L(X), R(X)$, таких, що $P(L(X) < \theta < R(X)) = 1-\alpha$.

Це класичне визначення довірчого інтервалу. Тобто, ми повинні знайти такі $L(X)$ та $R(X)$, що $P(L(X) < \mu < R(X)) = 1-\alpha$.

$$\begin{aligned}
&t(X) = \sqrt{n}\frac{\overline{X}-\mu}{\sqrt{S^2}} \sim t_{n-1} \Rightarrow \\
&P\left(-t_{n-1,1-\alpha/2} < \sqrt{n}\frac{\overline{X}-\mu}{\sqrt{S^2}} < t_{n-1,1-\alpha/2}\right) = 1-\alpha \Leftrightarrow \\
&P\left(\overline{X} - \frac{t_{n-1,1-\alpha/2}\sqrt{S^2}}{\sqrt{n}} < \mu < \overline{X} + \frac{t_{n-1,1-\alpha/2}\sqrt{S^2}}{\sqrt{n}}\right) = 1-\alpha
\end{aligned} \tag{4.9}$$

Тоді

$$CI_\mu = \left(\overline{X} \pm \frac{t_{n-1,1-\alpha/2}\sqrt{S^2}}{\sqrt{n}}\right) \tag{4.10}$$

Цей довірчий інтервал збігається з попереднім. Тобто, ми можемо використовувати обидва методи для побудови довірчого інтервалу.

4.5 Довірчі інтервали у Python

Давайте тепер побудуємо довірчий інтервал для середнього значення тривалості нарад у новому форматі. Для цього скористаємося функцією `scipy.stats.t.interval`. Вона приймає на вхід:

- `confidence` — рівень значущості;
- `df` — кількість ступенів свободи;
- `loc` — середнє значення, за замовчуванням `0`;
- `scale` — стандартна девіація, за замовчуванням `1`.

Для побудови лівостороннього довірчого інтервалу візьмемо `confidence` на рівні 90%, оскільки ми хочемо перевірити, чи тривалість нарад у новому форматі *менша* 70 хвилин.

```
meeting_time = np.array([50, 55, 70, 45, 40, 70, 80])

confidence = 0.90
df = len(meeting_time) - 1
loc = np.mean(meeting_time)
scale = np.std(meeting_time, ddof=1) / np.sqrt(len(meeting_time))

interval = t.interval(confidence, df, loc, scale)
print(f"Довірчий інтервал: {np.round(interval, 2)}")
```

```
Довірчий інтервал: [47.61 69.53]
```

4.6 t-тест та вимога нормальності

Ми навчилися розв'язувати задачу оцінки середнього вибірки, коли дисперсія невідома, але вибірка з нормального розподілу. Тепер розглянемо, що буде, якщо вибірка не з нормального розподілу.

Приклад 4.2.

> Ви запускаєте онлайн-платформу з курсами програмування. Ви плануєте надавати доступ до курсів за фіксовану плату, але також інвестуєте в маркетинг та підтримку студентів. У середньому, прибуток від одного користувача (після вирахування витрат на платформу, рекламу тощо) становить X грн., але витрати на залучення кожного нового студента — 1000 грн.
>
> Студенти можуть скористатися гарантією повернення грошей протягом 14 днів. Ви хочете перевірити, чи є прибуток від нових користувачів більшим за 0 грн. (тобто, чи є прибуток від нових користувачів більшим за витрати на залучення нових студентів). Тому іноді прибуток від користувача — позитивне число, а іноді — негативне.
>
> Інвестори готові профінансувати вашу платформу, якщо ви доведете, що вона буде прибутковою. У вас є дані про чистий прибуток або збиток від кожного користувача, який вже зареєструвався.

Згенеруємо штучні дані для цієї задачі. Для цього змішаємо логнормальний розподіл для позитивних значень (прибуток) та від'ємний χ^2 для від'ємних значень (збиток).

```
n = 5000
p_positive = 0.6

n_pos = int(n * p_positive)
profits = np.random.lognormal(mean=2, sigma=0.8, size=n_pos) * 100

n_neg = n - n_pos
losses = -np.random.chisquare(df=2, size=n_neg) * 100

profits = np.concatenate([profits, losses])
```

```
np.random.shuffle(profits)

sns.histplot(profits, bins=100, kde=True, color=turquoise)

plt.xlabel('Прибуток або збиток')
plt.ylabel('Кількість користувачів')
plt.grid(True)
plt.show()
```

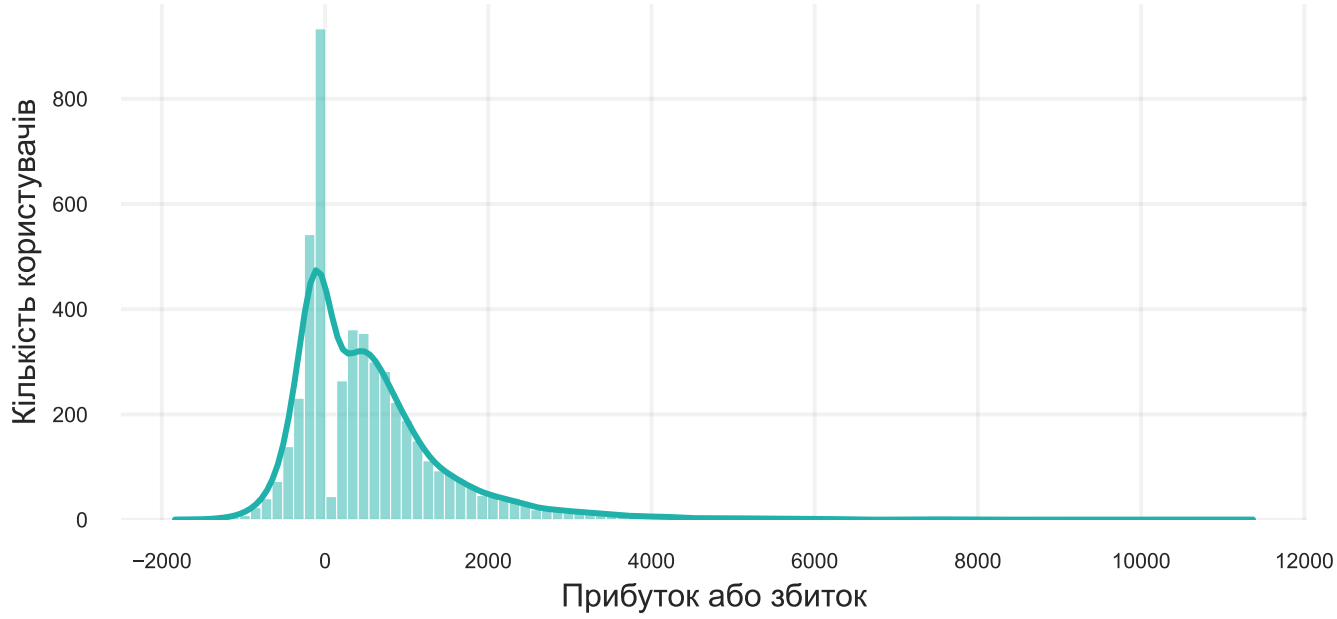


Рисунок 4.5: Візуалізація штучних до задачі.

Порахуємо середній прибуток.

```
print(f"Середній прибуток: {profits.mean():.2f}")
```

```
Середній прибуток: 547.45
```

На відміну від попереднього завдання тут 2 відмінності:

- Початкова вибірка не з нормального розподілу
- Вибірка досить велика: не 7 елементів, а вже 5000.

4.6.1 t'-тест

Згадаймо, що в нас від початку була ідея в Z-тесті замість статистики Z, у якій дисперсія відома, використовувати критерій t, де дисперсія оцінена на даних. І використовувати нормальний розподіл. Тільки в першому завданні цей критерій нам не допоміг. Але що, якби вибірка була великою? Чи могли б ми використовувати нормальний розподіл для наближення?

1. Будемо розглядати ту саму статистику $t = \sqrt{n}\dfrac{\overline{X} - \mu_0}{\sqrt{S^2}}$
2. $\xi := \sqrt{n}\dfrac{\overline{X} - \mu_0}{\sqrt{\sigma^2}} \xrightarrow{d} \mathcal{N}(0,1)$. За ЦГТ збіжність є тільки за розподілом.
3. тоді $t = \sqrt{n}\dfrac{\overline{X} - \mu_0}{\sqrt{S^2}} = \xi \cdot \sqrt{\dfrac{\sigma^2}{S^2}}$. Позначимо $\phi := \sqrt{\dfrac{\sigma^2}{S^2}}$

- Пам'ятаєте, раніше було сказано, що S^2 — найкраща оцінка для дисперсії? Річ у тім, що вона є консистентною оцінкою для σ^2. Тобто S^2 збігається за ймовірністю до σ^2. Тобто $S^2 \xrightarrow{p} \sigma^2$.
- А в цьому випадку існує теорема, яка стверджує, що $\phi = \dfrac{\sigma^2}{S^2} \xrightarrow{p} 1$.

4. $t = \xi \cdot \phi$.
   - $\xi \xrightarrow{d} \mathcal{N}(0,1)$
   - $\phi \xrightarrow{p} 1$
   - І тут набуває чинності ще одна теорема: $t = \xi \cdot \phi \xrightarrow{d} 1 \cdot \mathcal{N}(0,1)$. Та сама збіжність, що й у ЦПТ!
   - Тобто статистика t так само буде з нормального розподілу.

Отже, якщо вибірка велика, то ми можемо вважати, що $t(X) \overset{H_0}{\sim} \mathcal{N}(0,1)$.

> **Примітка**
>
> Зауважимо, що у випадку "нормальний розподіл, велика вибірка" працюють одразу 2 критерії: t-тест та t'-тест. Це означає, що якщо $t(X) \overset{H_0}{\sim} t_{n-1}$ та $t(X) \overset{H_0}{\sim} \mathcal{N}(0,1)$, то $t_{n-1} \approx \mathcal{N}(0,1)$.
> Формально ж, якщо ступінь свободи в t-розподілі дорівнює нескінченності, то це нормальний розподіл!
> $\lim_{n\to\infty} t_n = \mathcal{N}(0,1)$
> А якщо $t_{n-1} \approx \mathcal{N}(0,1)$, то ми замість t'-критерію ми можемо використовувати t-критерій!

В такому випадку критерій t-тесту буде виглядати так:

$$H_0 : \mu = \mu_0 \;\; H_1 : \mu > \mu_0 \tag{4.11}$$

Статистика $t(X)$ буде виглядати так:

$$t(X) = \sqrt{n}\frac{\overline{X} - \mu_0}{\sqrt{S^2}} \tag{4.12}$$

При достатньо великій вибірці $t(X) \sim \mathcal{N}(0,1)$.

Тоді односторонній критерій набуває вигляду:

$$\{t(X) \geq z_{1-\alpha}\} \tag{4.13}$$

А p-значення для одностороннього критерію можна обчислити так:

$$p\text{-значення} = 1 - \Phi(z), \tag{4.14}$$

де z — реалізація статистики $t(X)$, $\Phi(z)$ — функція розподілу $\mathcal{N}(0,1)$.

Двосторонній критерій буде виглядати так:

$$\left\{t(X) \geq z_{1-\frac{\alpha}{2}}\right\} \bigcup \left\{t(X) \leq -z_{1-\frac{\alpha}{2}}\right\} \tag{4.15}$$

При цьому p-значення для двостороннього критерію можна обчислити так:

$$p\text{-значення} = 2 \cdot \min\left[\Phi(z), 1 - \Phi(z)\right], \tag{4.16}$$

де z — реалізація статистики $t(X)$, $\Phi(z)$ — функція розподілу $\mathcal{N}(0,1)$.

Перевіримо наш критерій на великій вибірці. Для цього згеренуємо вибірку з експоненційного розподілу $\mathcal{E}(300)$, де $X \sim \mathcal{E}(\lambda)$, $\lambda = 1/\mu$. Вибірка буде згенерована з параметром $\lambda = 1/300$. Тобто, середнє значення вибірки буде 300.

```
sample_size=2000
M = 10000
sample_distr = expon(loc=5, scale=300)

T_X = lambda sample: np.sqrt(sample_size) * (np.mean(sample) - sample_distr.mean()) / np.std(s

T_sample = sample_statistics(
    number_of_experiments=M, statistic_function=T_X,
    sample_size=sample_size, sample_distr=sample_distr)

l_bound, r_bound = np.quantile(T_sample, [0.001, 0.999])

x = np.linspace(l_bound, r_bound, 1000)
sns.distplot(T_sample, label='Емпіричний розподіл', color=turquoise)
plt.plot(x, norm(0, 1).pdf(x), label='Експоненціальний розподіл', color=red_pink)
plt.legend()
plt.xlabel(f'{name}')
plt.xlim((l_bound, r_bound))
plt.ylabel('Щільність')
plt.grid(linewidth=0.2)
plt.show()
```

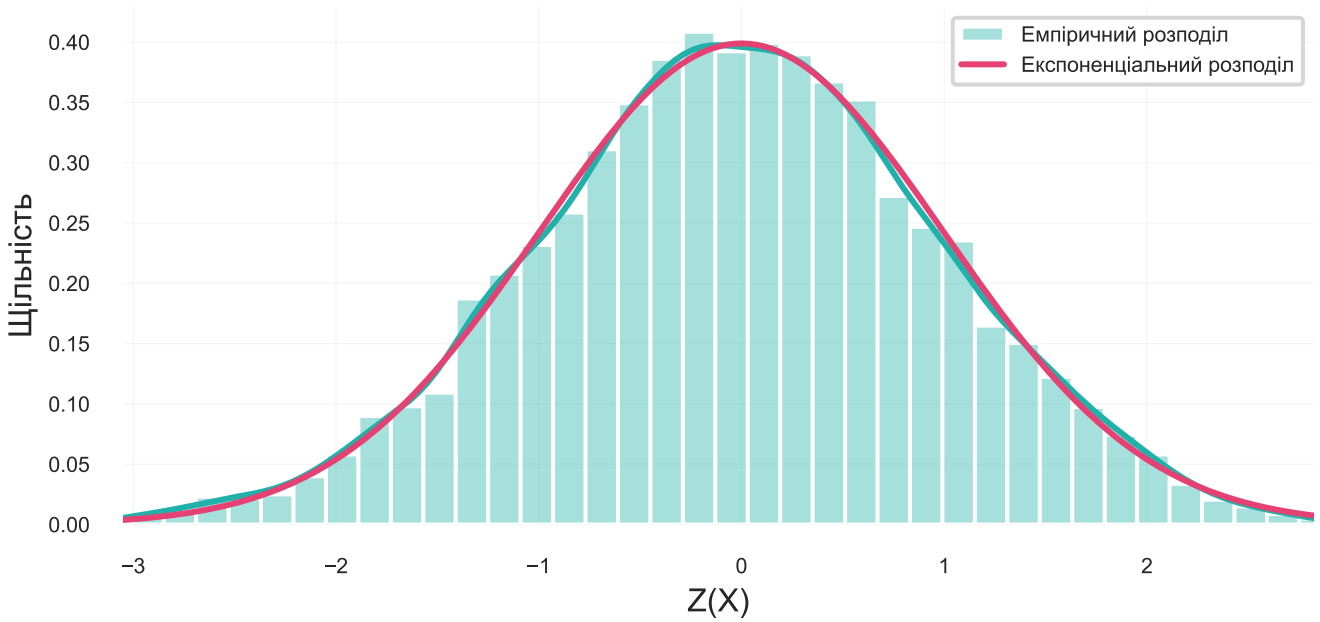


Рисунок 4.6: Розподіл $t'(X)$ для великої вибірки.

Ми бачимо, що емпіричний розподіл $t'(X)$ та теоретичний $\mathcal{N}(0,1)$ збігаються. Це означає, що ми можемо використовувати t'-тест для перевірки гіпотези. Якщо ж перевірити на великих вибірках з нормального розподілу, то t-тест та t'-тест будуть давати доволі схожі результати.

```
sample_size=2000
M = 30000
sample_distr = norm(loc=5, scale=300)

T_X = lambda sample: np.sqrt(sample_size) * (np.mean(sample) - sample_distr.mean()) / np.std(s
T_sample = sample_statistics(
```

```
    number_of_experiments=M, statistic_function=T_X,
    sample_size=sample_size, sample_distr=sample_distr)

l_bound, r_bound = np.quantile(T_sample, [0.001, 0.999])

x = np.linspace(l_bound, r_bound, 1000)
sns.distplot(T_sample, label='Емпіричний розподіл', color=turquoise)
plt.plot(x, norm(0, 1).pdf(x), label='$\mathcal{N}(0, 1)$', color=red_pink)
plt.legend()
plt.xlabel(f'{name}')
plt.xlim((l_bound, r_bound))
plt.ylabel('Щільність')
plt.grid(linewidth=0.2)
plt.show()
```

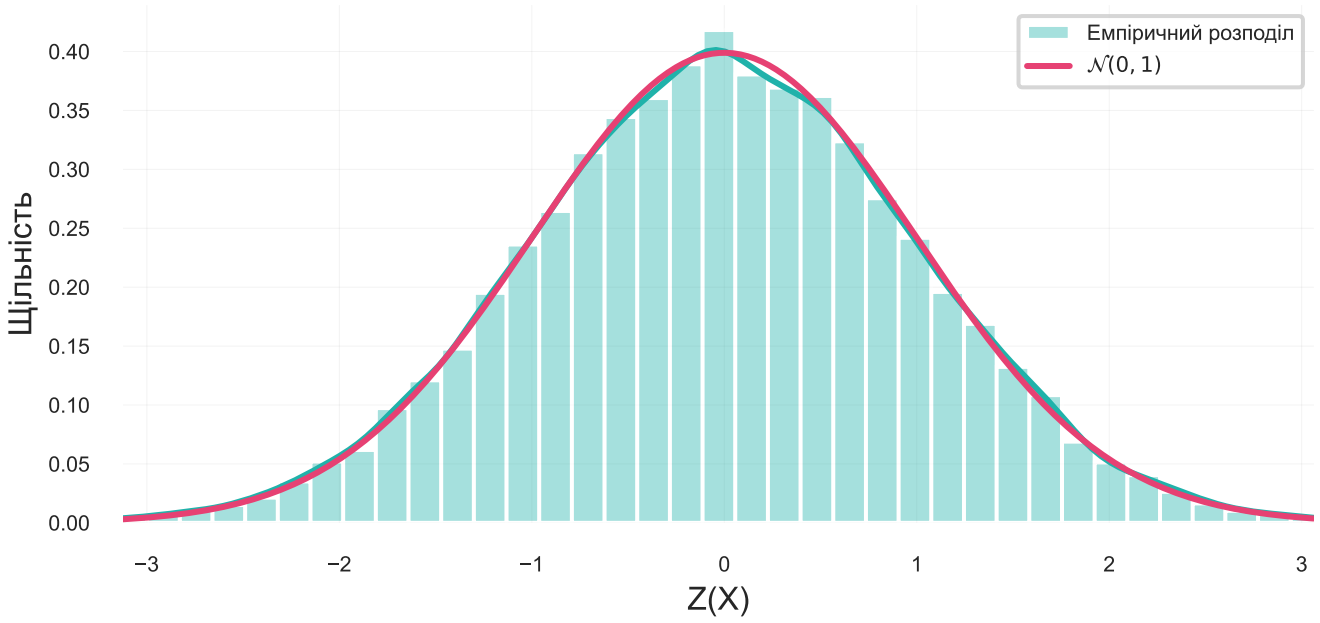


Рисунок 4.7: Розподіл $t(X)$ для великої вибірки.

Виходить, що статистика $t'(X)$ при великій вибірці з нормального розподілу також буде з нормального розподілу.

4.7 Довірчий інтервал

Довірчий інтревал виводиться аналогічно до t-тесту.

$$
\begin{aligned}
&t'(X) = \sqrt{n}\frac{\overline{X} - \mu}{\sqrt{S^2}} \sim \mathcal{N}(0,1) \Rightarrow \\
&P\left(-z_{1-\frac{\alpha}{2}} < \sqrt{n}\frac{\overline{X} - \mu}{\sqrt{S^2}} < z_{1-\frac{\alpha}{2}}\right) = 1 - \alpha \Leftrightarrow \\
&P\left(\overline{X} - \frac{z_{1-\frac{\alpha}{2}} S^2}{\sqrt{n}} < \mu < \overline{X} + \frac{z_{1-\frac{\alpha}{2}} S^2}{\sqrt{n}}\right) = 1 - \alpha
\end{aligned}
\tag{4.17}
$$

Переглнемо на практиці, як це виглядає у Python.

```
sample = expon(scale=300).rvs(2000)
ci = norm.interval(confidence=0.95, loc=np.mean(sample), scale=sem(sample))
print(f"CI = {np.round(ci, 2)}")
```

```
CI = [293.89 319.75]
```

Тепер ми можемо повернутися до нашої задачі з прибутком.

```
ci_profit = norm.interval(confidence=0.95, loc=np.mean(profits), scale=sem(profits))
print(f"CI = {np.round(ci_profit, 2)}")
```

```
CI = [520.19 574.72]
```

Це означає, що ми можемо стверджувати, що прибуток від нових користувачів більший за 0 грн.

4.8 Вибір критерію

Для початку визначимося, коли який критерій краще використовувати?

1. Якщо вибірка розміру 60, то вже $t_{59} \approx \mathcal{N}(0, 1)$.
    - Подивимося на розподіли Стьюдента і нормального:

```
df = 59
t_dist = t(df=df)
z_dist = norm(loc=0, scale=1)

x = np.linspace(-3, 3, 100)

plt.plot(x, z_dist.pdf(x), label='$\mathcal{N}(0, 1)$', color=red_pink)
plt.plot(x, t_dist.pdf(x), label='$t_{59}$', color=turquoise)
plt.legend()
plt.grid(linewidth=0.2)
plt.show()
```

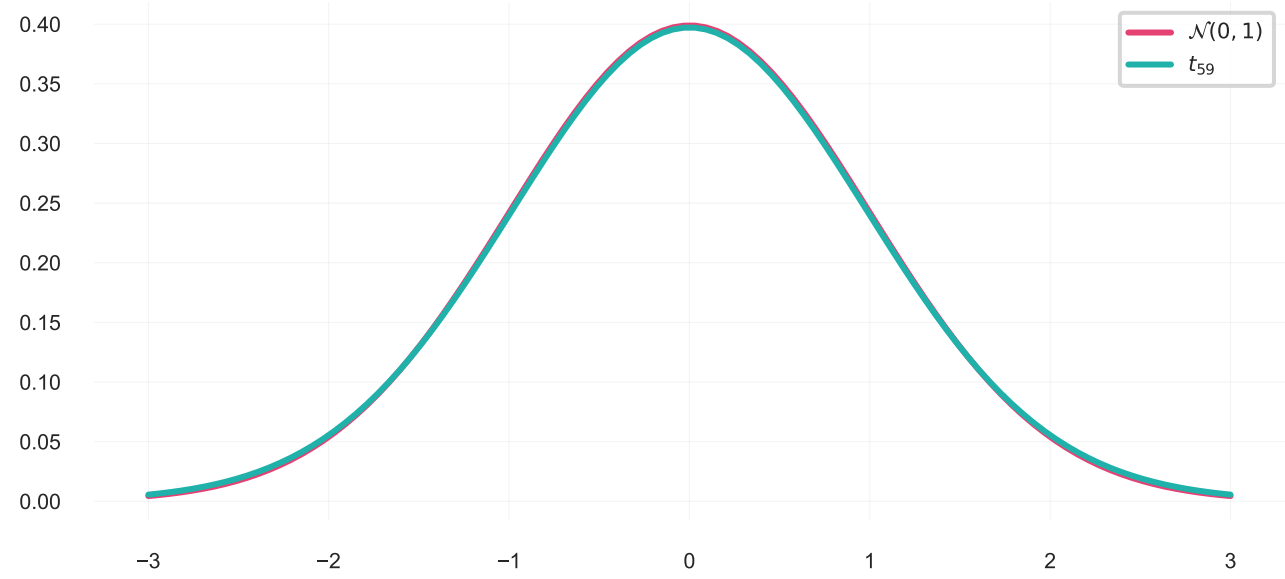


Рисунок 4.8: Розподіл $t(X)$ та $N(0, 1)$.

- Ми бачимо, що ці два розподіли візуально повністю збігаються, тому неважливо, як порахувати: статистика $t \sim \mathcal{N}(0, 1)$ або $t \sim t_n$.
- Але це не означає, що з $N = 60$ t-тест або t'-тест працюють коректно! Якщо вибірка не з нормального розподілу, вони обидва можуть усе ще помилятися.

2. Якщо вибірка менше 60, то безпечніше використовувати t-тест, ніж t'-тест.
   - **У t-тест FPR завжди буде меншим, ніж у t'-тест**.
     - На FPR впливає відсоток випадків `pvalue < alpha`. У t-тест p-значення $\geq$ t'-тест p-значення.
     - `pvalue = t_distr.cdf(x)` або `pvalue = norm_dist.cdf(x)`. Тож чим важчий хвіст у розподілу, тим більше p-значення.

Подивимось на прикладі.

```
df_array = [2, 5, 10, 20]
x = np.linspace(-3, 3, 100)

for df in df_array:
    t_dist = t(df=df)
    plt.plot(x, t_dist.cdf(x), label=f't(df={df})')

z_dist = norm(loc=0, scale=1)
plt.plot(x, z_dist.cdf(x), c=red_pink, label='$\mathcal{N}(0, 1)$')
plt.legend()
plt.xlabel('X')
plt.grid(linewidth=0.2)
plt.show()
```

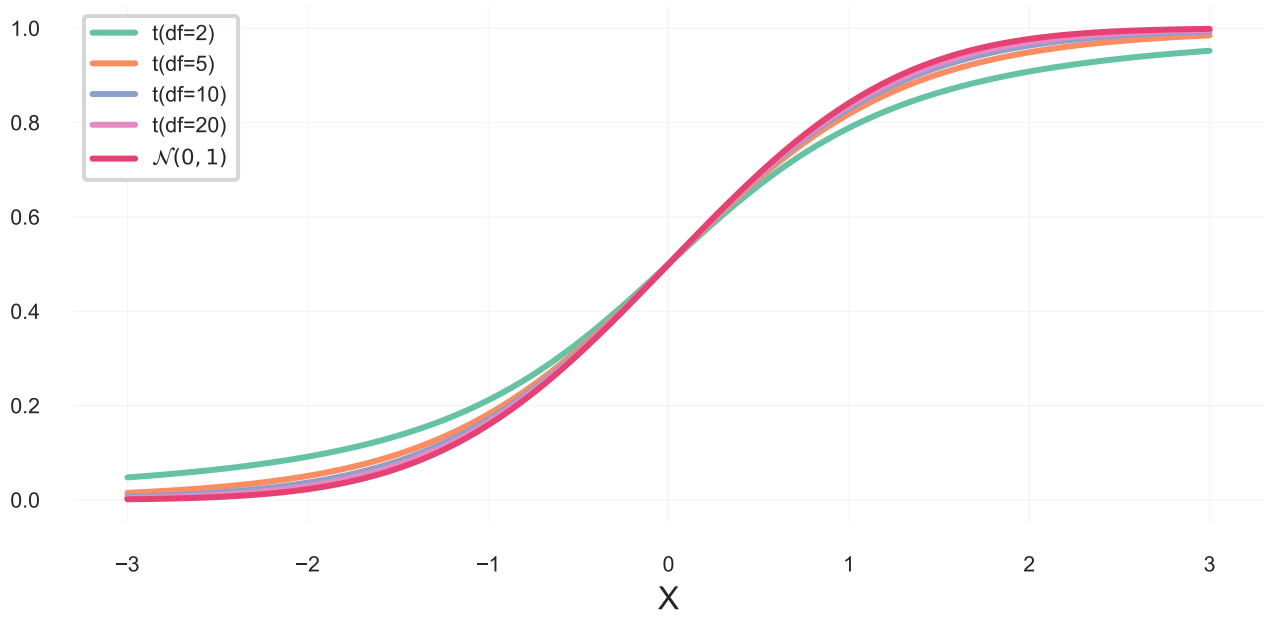


Рисунок 4.9: Куртки розподілів t та $N(0, 1)$.

Як видно на графіку, що менший ступінь свободи, то вищою є лінія на графіку (за x < 0), а отже $P(X < x)$ буде більшим, ніж у нормальному розподілі. Тобто, t-тест буде давати менше p-значення, ніж t'-тест. А отже, t-тест буде відкидати нульову гіпотезу частіше, ніж t'-тест.

> **Примітка**
>
> Розподіл Стьюдента з нескінченністю ступенів свободи — це нормальний розподіл: $t_\infty = \mathcal{N}(0, 1)$. Тому `norm(0, 1).cdf(x) = t_distr(df=infinity).cdf(x) < t_distr(df=N).cdf(x)`.

Тому, якщо вибірка невелика, безпечніше використовувати t-тест. Але все ще не факт, що ваш критерій буде валідний!

Ми бачимо, що ми скрізь можемо використовувати t-тест (а t'-тест не завжди), і в разі маленьких вибірок він безпечніший. **Тому t-тест і став набагато популярнішим, ніж t'-тест**. Але t'-тест на практиці може бути теж корисний:

- Не треба думати під час реалізації про ступені свободи.
- Написати такий критерій на SQL буде набагато простіше: ви можете використовувати табличні значення в коді, щоб зрозуміти, чи відкинувся критерій.
- Робити різні теоретичні обчислення простіше.
- У ньому складніше помилитися під час реалізації.

4.9 Мінімальний ефект

Повернемося до завдання зі стартапом. Уявімо, що ми хочемо запустити наш стартап на новому ринку, наприклад у іншій країні. **Питання: чи можемо ми зменшити розмір вибірки?**

Що взагалі нам заважає взяти занадто маленьку вибірку? Наприклад, якщо ми перевіряємо наш стартап на 1-2 користувачів, то ми нічого не можемо сказати про наш істинний ефект, він може бути як більшим за 0, так і меншим. Буде занадто широкий довірчий інтервал (через велику дисперсію у вибірці), і нам потрібен величезний ефект, щоб його виявити.

Ще, можливо, ми не можемо використовувати критерій на такій маленькій вибірці. А якщо вибірка складалася б із нескінченної кількості користувачів, то ми могли б абсолютно точно сказати справжній прибуток від користувача, навіть якщо він дорівнює 1 копійці. При цьому обидва випадки нас не влаштовують. У першому — ми не зможемо запустити стартап через занадто великий шум, а в другому — нам потрібна вічність, щоб перевірити нашу гіпотезу.

І тут нам допоможе MDE (minimum detectable effect). Це таке істинне значення ефекту, що наш шанс його виявити дорівнює $1 - \beta$ при використанні нашого критерію.

Ми можемо подивитись, який ефект ми зможемо зафіксувати під меншої кількості користувачів, і від цього вирішити, чи підходить нам така вибірка, чи ні. Наприклад:

- Ми бачимо, що MDE 100 гривень. Тобто з ймовірністю $1 - \beta$ (на практиці 80%) ми його виявимо, **якщо такий ефект буде**. І з імовірністю 80% стартап запуститься на новому ринку. Чудово, це нас влаштовує, ми перевіряємо гіпотезу на меншій вибірці.
- Ми бачимо, що MDE 10000 гривень. Це, навпаки, занадто багато: у нас 99% послуг коштують менше 1000 гривень. Ми не наберемо такого прибутку, стартап невиграшний, потрібно брати вибірку більшого розміру.

Тому слід чітко визначити **від чого залежить MDE**. Це може бути:

- *Помилка першого роду*, або α. Наприклад, за $\alpha = 1$ ми знайдемо ефект і за розміру вибірки, що дорівнює одиниці (ми просто завжди відкидатимемо H_0). А за $\alpha = 0$ ми ніколи не зафіксуємо ефект.
- *Потужність*, або $1 - \beta$. Випливає із самого визначення.
- Від *шуму в даних*, або від *дисперсії*. Що більш шумні дані, як ми знаємо, то ширший довірчий інтервал. А отже, складніше точно передбачити межі для істинного ефекту, тому й MDE буде більшим.
- Від *розміру вибірки*. Нас цікавить не просто дисперсія в даних, а дисперсія середнього значення: вона за тією самою логікою має бути якомога меншою. А що таке дисперсія середнього? Це $\frac{\sigma^2}{N}$, тому MDE також залежить від розміру вибірки.

Тепер давайте виведемо формулу виходячи з того, що ми знаємо всі ці чотири параметри. Для початку визначимося з гіпотезою, що перевіряється:

$$H_0 : \mu_0 = 0 \; . \; H_1 : \mu_0 > 0 \tag{4.18}$$

Позначимо оцінку дисперсії середнього значення:

$$S_\mu^2 := \frac{S^2}{N} \tag{4.19}$$

А також стандартне відхилення середнього значення:

$$S_\mu = \sqrt{\frac{S^2}{N}} \tag{4.20}$$

Тепер ми знаємо, що

$$\overline{X} \sim \mathcal{N}(\mu, S_\mu^2) \tag{4.21}$$

Нам треба знайти MDE $= m$, таке, що:

- якщо $\overline{X} \sim \mathcal{N}(m, S_\mu^2)$, то в $1-\beta$ відсотку випадків для нього відкинеться критерій. Перевіряємо потужність (ціанова площа на графіку).
- якщо $\overline{X} \sim \mathcal{N}(0, S_\mu^2)$, то критерій відкинеться для нього в α відсотків випадків. Перевіряємо FPR (рожева площа на графіку).

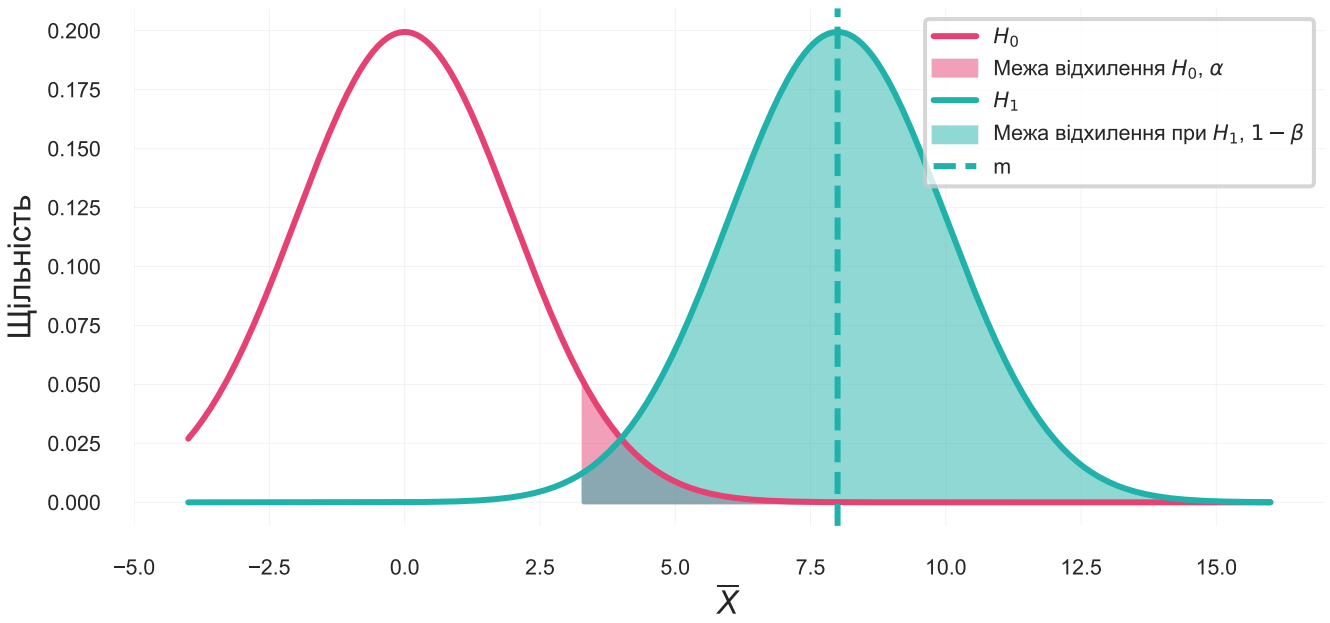


Рисунок 4.10: Графік MDE.

Нехай

$$B(X) : P_{H_0}(\overline{X} > B(X)) = \alpha, \tag{4.22}$$

де $B(X)$ — межа відхилення нульової гіпотези.

Тоді

$$P_{H_1}(\overline{X} > B(X)) = 1 - \beta \tag{4.23}$$

Або

$$P_{H_1}(\overline{X} - m > B(X) - m) = 1 - \beta \tag{4.24}$$

Позначимо $\xi := \overline{X} - m$. Тоді

$$P_{H_0}(\xi > B(X) - m) = 1 - \beta, \tag{4.25}$$

де $B(X) - m$ — межа відхилення нульової гіпотези з урахуванням істинного ефекту.

Треба розв'язати ці 2 рівняння й ми отримаємо вираз m через усі чотири параметри.

За H_0 наш критерій має такий вигляд:

$$\{T(X) \geq z_{1-\alpha}\} \Leftrightarrow \left\{\sqrt{N}\frac{\overline{X}}{\sqrt{S^2}} \geq z_{1-\alpha}\right\} \Leftrightarrow B(X) = z_{1-\alpha}\sqrt{\frac{S^2}{N}} = z_{1-\alpha}S_\mu \tag{4.26}$$

Тоді

$$P_{H_0}(\xi > z_{1-\alpha}S_\mu - m) = 1 - \beta \tag{4.27}$$

Але працювати з розподілом $\mathcal{N}(0, S_\mu^2)$ не дуже зручно, набагато простіше з $\mathcal{N}(0,1)$. Для цього переходу достатньо перейти від $\xi \to \frac{\xi}{S_\mu}$ за властивостями нормального розподілу.

Позначимо $\eta := \frac{\xi}{S_\mu}$. Тоді

$$\begin{aligned} &P_{H_0}(\xi > z_{1-\alpha}S_\mu - m) = \\ &P_{H_0}(\frac{\xi}{S_\mu} > \frac{z_{1-\alpha}S_\mu - m}{S_\mu}) = \\ &P_{\mathcal{N}(0,1)}(\eta > z_{1-\alpha} - \frac{m}{S_\mu}) = 1 - \beta \end{aligned} \tag{4.28}$$

За умови $\Phi(C) = P(\eta < C)$, тоді

$$\begin{aligned} &1 - \Phi\left(z_{1-\alpha} - \frac{m}{S_\mu}\right) = 1 - \beta \Leftrightarrow \\ &z_{1-\alpha} - \frac{m}{S_\mu} = z_\beta, \end{aligned} \tag{4.29}$$

де $z_\beta = \Phi^{-1}(\beta)$ — квантиль β нормального розподілу.

Тепер згадаємо, що $\eta \sim \mathcal{N}(0,1)$, тоді

$$m = (z_{1-\alpha} - z_\beta) \cdot S_\mu = (z_{1-\alpha} + z_{1-\beta}) \cdot \sqrt{\frac{S^2}{N}} \tag{4.30}$$

Отже, ми отримали формулу для MDE:

$$\text{MDE} = (z_{1-\alpha} + z_{1-\beta}) \cdot \sqrt{\frac{S^2}{N}} \tag{4.31}$$

де $z_{1-\alpha}$ — квантиль α нормального розподілу, $z_{1-\beta}$ — квантиль β нормального розподілу, S^2 — оцінка дисперсії, N — розмір вибірки.

Повернемося до стартапу. Припустимо, що $N = 1000$, $\alpha = 5\%$, $1 - \beta = 80\%$, а як дізнатися S^2?

На практиці є 3 способи:

- Оцінити на історичних даних. У цьому випадку це не підходить, тому що раніше стартапу на новому ринку не було.
- Оцінити за схожими даними. Наприклад, у нашому випадку, оцінити дисперисію на початковому ринку.
- Якось теоретично оцінити. Найгірший спосіб, який працює, якщо перші два не допомагають.

Подивимося тепер MDE у нашому завданні.

```
N = 1000
S2 = np.var(profits)
alpha = 0.05
beta = 1 - 0.8

MDE = (norm().ppf(1-alpha) + norm().ppf(1 - beta)) * np.sqrt(S2/N)
print(f"MDE при N = {N}: {np.round(MDE, 2)}")
```

```
MDE при N = 1000: 77.33
```

А отже, ми можемо розраховувати на точність лише в 77.33 грн. Але дня нас це може бути занадто великий MDE: хочеться, щоб він був $\leq$ 50 грн, ми припускаємо, що це ймовірніший істинний ефект, виходячи з попереднього досвіду.

Давайте тепер розв'яжемо зворотну задачу: Ми знаємо $MDE = 50$ грн., $\alpha = 5\%$, $1-\beta = 80\%$, S^2, чому дорівнює N? Виведемо його з формули MDE:

$$N = \left(\frac{z_{1-\alpha} + z_{1-\beta}}{\text{MDE}}\right)^2 S^2 \tag{4.32}$$

```
S2 = np.var(profits)
alpha = 0.05
beta = 1 - 0.8
mde = 50

N = ((norm().ppf(1-alpha) + norm().ppf(1 - beta)) / mde)**2 * S2
N = int(N) + 1
print(f"Мінімальний розмір вибірки: {N}")
```

```
Мінімальний розмір вибірки: 2392
```

Тепер ми знаємо, що нам потрібно 2392 користувачів, щоб перевірити нашу гіпотезу з MDE 50 грн.

4.10 Двовибірковий t-тест

Приклад 4.3.

> На нашому онлайн-сервісі з розміщення оголошень є платні послуги просування. Ми плануємо запровадити знижки на ці послуги, щоб залучити більше користувачів і збільшити дохід. Для перевірки ефективності було вирішено провести A/B тест: одній половині нових користувачів ми не надавали знижки, а другій половині — надавали знижки на просування. Потрібно визначити, чи призвело це до зростання доходу.

Для вирішення цього завдання ми не можемо використовувати одновибірковий t-тест. Цього разу в нас дві вибірки A — контроль, та B — тест.

Наша гіпотеза звучить так:

$$H_0 : EA = EB \;\; H_1 : EA < EB$$

Далі в нас може бути кілька варіантів:

1. **Обидві вибірки нормальні**.

Тоді у випадку рівних дисперсій $\sigma_A^2 = \sigma_B^2$, спільна дисперсія S_{pooled}^2 обчислюється за формулою:

$$S_{pooled}^2 = \frac{(N-1)S_A^2 + (M-1)S_B^2}{N+M-2}, \tag{4.33}$$

де N, M — розмір контролю і тесту відповідно.

А критерій має такий вигляд:

$$t(A,B) = \frac{\overline{A} - \overline{B}}{S_{pooled}^2\sqrt{1/N + 1/M}} \overset{H_0}{\sim} t_{n+m-2} \tag{4.34}$$

де $n+m-2$ — ступені свободи.

У випадку, якщо дисперсії не рівні $\sigma_A^2 \neq \sigma_B^2$[5], то:

$$t(A,B) = \frac{\overline{A} - \overline{B}}{\sqrt{S_A^2/N + S_B^2/M}} \overset{H_0}{\sim} t_v \tag{4.35}$$

де v — ступені свободи, які обчислюються за формулою:

$$v = \frac{\left(\frac{S_A^2}{N} + \frac{S_B^2}{M}\right)^2}{\left(\frac{(S_A^2)^2}{N^2(N-1)} + \frac{(S_B^2)^2}{M^2(M-1)}\right)} \tag{4.36}$$

2. **Хоча б одна вибірка не нормальна**.

Тоді ми можемо використовувати нормальну апроксимація при великій вибірці:

$$t(A,B) = \frac{\overline{A} - \overline{B}}{\sqrt{S_A^2/N + S_B^2/M}} \overset{H_0}{\sim} \mathcal{N}(0,1) \tag{4.37}$$

де S_A^2, S_B^2 — вибіркові дисперсії.

4.10.1 Двовибірковий t-тест у Python

Для реалізації двовибіркового t-тесту в Python ми можемо використовувати `ttest_ind`[6] з бібліотеки `scipy.stats`.

Подивимось на приклад з двома вибірками, перша вибірка — з експоненційного розподілу, а друга — з нормального розподілу.

[5] Такий підхід називається t-тестом Уелча.
[6] Документація: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.htm

```python
X = expon(scale=1100).rvs(1000)
Y = norm(loc=980, scale=30).rvs(1000)

t_results = ttest_ind(X, Y, equal_var=False, alternative='greater')
ci_t_results = t_results.confidence_interval(confidence_level=0.95)

print(f"t-статистика = {np.round(t_results.statistic, 2)}")
print(f"p-значення = {np.round(t_results.pvalue, 5)}")
print(f"Довірчий інтервал = {np.round(ci_t_results, 2)}")
```

```
t-статистика = 3.62
p-значення = 0.00015
Довірчий інтервал = [68.38   inf]
```

Попередження

При використанні t-тесту, як одновибіркового, так і двовибіркового, важливо пам'ятати, що:

- Елементи вибірок мають бути незалежні.
  - Наприклад, ваша вибірка не може містити кілька замовлень одного користувача. Вони мають бути агреговані, інакше критерії будуть невалідні!
- У двовибірковому критерії **вибірки тесту і контролю повинні бути незалежні!**.
  - Інакше критерії так само будуть невалідними.

4.11 Контрольні питання

Chapter 5

Монте-Карло в задачах статистики

У цій частині ми розглянемо метод Монте-Карло, який є потужним інструментом для чисельного моделювання та статистичного аналізу. Метод Монте-Карло дозволяє оцінювати ймовірності, інтеграли та інші статистичні характеристики шляхом випадкового вибору з певного розподілу. Ключовим моментом, на котрому ми зупинимось, це пошук відповідей на питання:

- Як перевірити наш критерій?
- Чи можна використовувати критерій на практиці?
- Якщо у нас є дві або більше альтернатив, як обрати найкращу?

5.1 Перевірка критерію

За допомогою методу Монте-Карло ми в *загальному випадку* зможемо відповісти на запитання:

- Чи можна використовувати цей критерій для нашого завдання?
- Чи правильно взагалі реалізовано критерій?

Увесь цей розділ насамперед буде присвячено AB-тестам і як можна перевіряти критерії для них. Основним критерієм для перевірки в цьому розділі стане t-тест, оскільки навколо нього обертається доволі багато міфів та непорозумінь. Ми з вами:

- Покажемо на практиці, що t-тест працює для вибірок не тільки з нормального розподілу.
- Подивимося, як визначити, з якого розміру вибірки можна застосовувати t-тест.

Як ми пам'ятаємо з минулої глави (див. 4), t-тест працює теоретично для вибірок з будь-якого розподілу, якщо вибірка досить велика. Але що значить, що критерій "коректний"? Давайте підемо від визначення:

- Критерій рівня значущості α означає, що ймовірність невірно відкинути нульову гіпотезу $\leq \alpha$.
- А це зі свого боку означає, що якщо нескінченно багато разів повторити один й той самий експеримент, у якому правильна нульова гіпотеза, генеруючи наново експеримент, то кількість хибнопозитивних спрацьовувань буде меншою за α відсотків.

Ці визначення дозволяють нам визначити процедуру перевірки критерію:

1. Створюємо код критерію, який ми будемо перевіряти.
2. Генеруємо якомога більше експериментів, де вірна H_0.
3. Досліджуємо на них придуманий критерій.
4. Перевіряємо, чи правда, що тільки в α відсотків випадків критерій відкидається?

Тепер давайте розглянемо процедуру більш детально:

1. Насамперед треба вибрати розподіл, який буде описувати наші дані. Наприклад, якщо у нас метрика конверсії, то це розподіл Бернуллі, а якщо метрика — виторг, то краще використовувати експоненціальний розподіл як найпростіше наближення.

2. Завести лічильник `bad_cnt`, який буде рахувати кількість разів, коли критерій помилився. Ініціалізувати його нулем.
3. Далі в циклі розміру N, де N — натуральне число від 1000 до нескінченності (чим воно більше, тим краще):
   - Симулюємо створення вибірки з розподілу, обраного на першому кроці. Так, щоб вірною була H_0. У випадку AB-тесту симулювати треба не одну вибірку, а дві: для тесту і контролю.
   - Досліджуємо на згенерованих даних критерій, що перевіряється.
   - Далі перевірити, чи критерій відкинув нульову гіпотезу. Якщо так, то збільшуємо `bad_cnt` на одиницю.
4. Порахувати частку помилок. Це буде ймовірність того, що критерій помиляється.
   - Якщо вона приблизно збігається з α, то все добре.
   - Якщо вона менша за α, то в принципі це адекватний критерій на практиці, просто він буде менш потужний, ніж критерій, що помиляється рівно у α відсотку випадків. Але на практиці варто перевірити: а теоретично така ситуація можлива? Чи це помилка в коді критерію?
   - Якщо критерій помиляється більше, ніж у α, то значить він некоректний і ним не можна користуватися. Використовуючи такий критерій, ви будете помилятися частіше, ніж треба, і це може призвести до серйозних помилок у бізнес-рішеннях.

Розглянемо процедуру на прикладі: перевіримо, чи можна використовувати t-тест для вибірок із нормального розподілу?

```
np.random.seed(42)

bad_cnt = 0
N = 10000
alpha = 0.05

sample_dist = norm(loc=2, scale=3)
mu0=sample_dist.expect()
for i in range(N):
    test    = sample_dist.rvs(5)
    control = sample_dist.rvs(5)
    pvalue = ttest_ind(test, control, alternative='two-sided').pvalue
    bad_cnt += (pvalue < alpha)

print(f"FPR: {bad_cnt/N:.3f}")
```

```
FPR: 0.052
```

Зверніть увагу, що $FPR = 0.05$, хоча він мав дорівнювати 5%. Чи правда, що критерій некоректний? Ні, ми просто не врахували шум: ми навряд чи зможемо отримати на кінцевому числі експериментів точну рівність $\text{FPR} = \alpha$.

Тому пункт 4 процедури перевірки критерію можна уточнити:

4. Порахувати частку помилок й *побудувати довірчий інтервал для нього*. Якщо α лежить у ньому, значить усе добре, а інакше розбираємося, що пішло не так.

Довірчий інтервал можна побудувати різними способами (див. 2.4). Але можна зробити простіше: у Python є функція, яка будує довірчий інтервал Вілсона[1]: він не такий точний, як ми виводили раніше, зате він швидший й працює швидше. Давайте спробуємо його реалізувати:

```
ci = proportion_confint(count = bad_cnt, nobs = N, alpha=0.05, method='wilson')
print(f"FPR: {bad_cnt/N:.3f}\nДовірчий інтервал: ({ci[0]:.3f}, {ci[1]:.3f})")
```

```
FPR: 0.052
Довірчий інтервал: (0.048, 0.056)
```

[1] Wilson (1927)

Як бачимо, що 5% потрапили в довірчий інтервал, а отже, ми можемо вважати, що критерій є валідним для нашого завдання.

А що, якби розподіл був складнішим?

Розглянемо приклад, коли магматичне сподівання в тесті й контролі рівні, але вибірки з різних розподілів. Тобто H_0 правильна, але розподіли різні. Для цього ми можемо взяти два експоненціальних розподіли з різними параметрами. Наприклад, раніше в середньому виручка від користувача була приблизно 10 гривень, а після введення ефекту впливу (нове ціноутворення) частина користувачів стала менше платити, але середній чек залишився таким самим: 10 гривень.

```
np.random.seed(42)

test_dist = expon(scale = 10)
control_dist = expon(loc=5, scale = 5)

x = np.linspace(0, 100, 1000)

plt.plot(x, test_dist.pdf(x), label='Тест', color=turquoise)
plt.plot(x, control_dist.pdf(x), label='Контроль', color=slate)
plt.xlabel('x')
plt.ylabel('Щільність')
plt.legend()
plt.grid(linewidth=0.2)
plt.show()
```

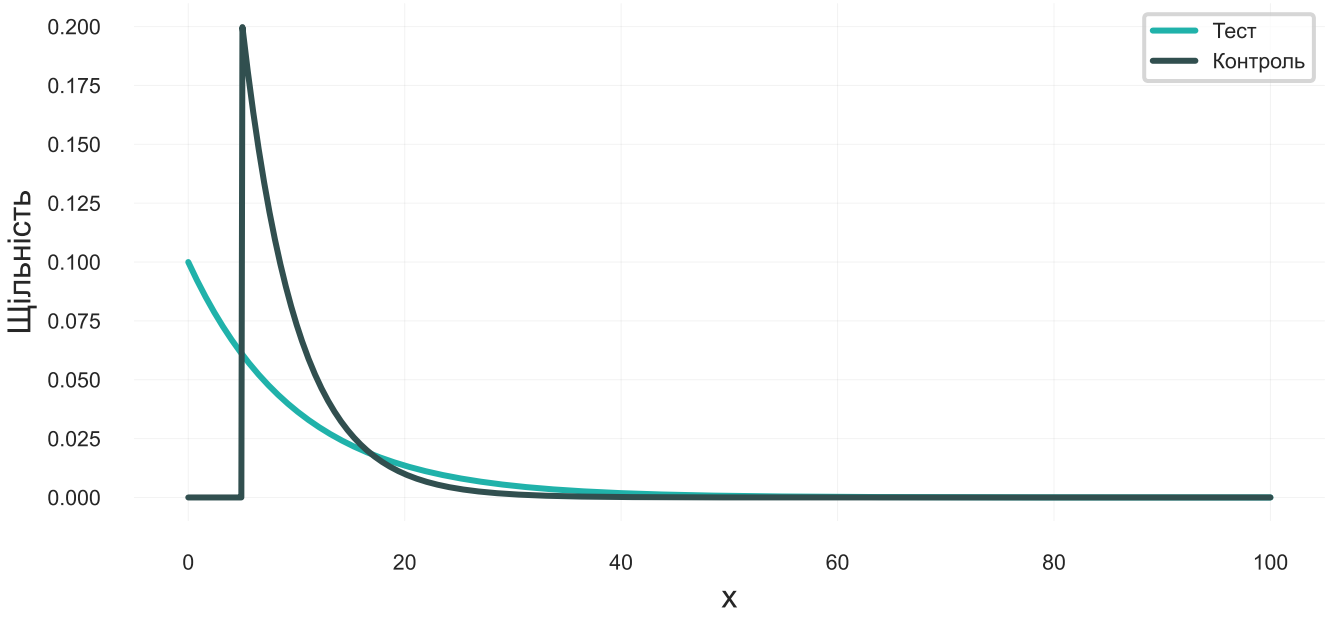


Рисунок 5.1: Приклад, коли H_0 правильна, але розподіли різні

Напишемо функцію `check_criterion()`, яка буде перевіряти критерій на коректність. Вона приймає на вхід розподіли для тесту й контролю, розмір вибірки, кількість експериментів, а також вміє виводити довірчий інтервал.

```
def check_criterion(test_dist, control_dist, sample_size, N_exps=10000, to_print=True):
    np.random.seed(35)
    bad_cnt=0
```

```
    alpha=0.05

    for i in range(N_exps):
        test    = test_dist.rvs(sample_size)
        control = control_dist.rvs(sample_size)
        pvalue = ttest_ind(test, control, equal_var=False, alternative='two-sided').pvalue
        bad_cnt += (pvalue < alpha)

    ci = proportion_confint(count = bad_cnt, nobs = N_exps, alpha=0.05, method='wilson')

    if to_print:
        print(f"FPR: {bad_cnt/N_exps:.3f}\nДовірчий інтервал: ({ci[0]:.3f}, {ci[1]:.3f})")
    else:
        return ci
```

Тепер перевіримо, чи працює t-тест для вибірок з різних експоненціальних розподілів. Одразу спробуємо перевірити відомий міф про достатність вибірки 30[2]. Чи дійсно t-тест працює, якщо вибірка більша за 30?

```
check_criterion(test_dist, control_dist, sample_size=40)
```

```
FPR: 0.061
Довірчий інтервал: (0.057, 0.066)
```

Як бачимо, t-тест не спрацював, хоча вибірка більша за 30. Істинне α не лежить у довірчому інтервалі. Але з якого розміру вибірки t-тест почне працювати правильно?

5.2 Визначення розміру вибірки

Визначити розмір вибірки, з якого t-тест почне працювати, можна за допомогою методу Монте-Карло. Для цього ми будемо перевіряти t-тест на вибірках різного розміру, поки не знайдемо такий розмір, при якому t-тест почне працювати. Для цього ми будемо перевіряти t-тест на вибірках від 20 до 100 з кроком 10.

```
scale = np.arange(20, 110, 10)
for N in scale:
    left, right = check_criterion(test_dist=test_dist, control_dist=control_dist, sample_size=
    if left < alpha < right:
        print(f"Розмір вибірки {N} достатній для t-тесту")
        break
```

```
Розмір вибірки 60 достатній для t-тесту
```

Як бачимо, t-тест починає працювати з вибірки розміром 60.

```
check_criterion(test_dist=test_dist, control_dist=control_dist, sample_size=60)
```

```
FPR: 0.053
Довірчий інтервал: (0.048, 0.057)
```

Але це доволі умовна межа, бо якщо у цьому випадку ми візьмемо вибірку розміром 61, то t-тест не спрацює.

```
check_criterion(test_dist=test_dist, control_dist=control_dist, sample_size=61)
```

```
FPR: 0.057
Довірчий інтервал: (0.052, 0.061)
```

Це може бути пов'язано з тим, що ми взяли недостатню велику кількість експериментів, або з дисперсією, або з обома факторами. Тому, якщо потрібна більша точність — необхідно проводити більше експериментів.

[2] Hogg, Tanis, and Zimmerman (2015)

5.3 Моделювання експерименту

Розберемо два сценарії моделювання експерименту:

1. Генерація тестової та контрольної групи через імітаційне моделювання. За допомогою різних розподілів можна спробувати наблизити реальний розподіл на даних. Наприклад:
   - Для генерації виручки використовувати експоненціальний розподіл. Чим більша виручка від користувача — тим менше таких людей.
   - Для генерації конверсійних вибірок (наприклад, клікне/не клінкет) використовувати бернуллііївську вибірку.
   - Іноді можна брати суміш розподілів: нехай 90% користувачів нашого сайту приносять нульову виручку. Тоді можна перемножити бернуллівський розподіл на експоненціальний для моделювання виручки від користувача.
   - Також для перевірки критерію рівності середніх не обов'язково мають збігатися розподіли в тесті та в контролі. Вони можуть бути різними, але математичне сподівання має збігатися.
2. Використати історичні дані компанії. У багатьох компаній є логування подій. Тоді ми зможемо прямо на реальних даних оцінити правильність критерію! І не потрапити в пастку того, що на штучних вибірках критерій валідний, а на реальних даних ні. Наприклад, у нас є дані про транзакції користувачів за кілька років. Це вже один готовий набір даних: ви ділите всіх користувачів на тестову та контрольну групи й отримуєте один "експеримент" для перевірки вашого критерію.

Залишилося зрозуміти, як з одного великого набору даних зробити N маленьких. Покажемо на прикладі сервісу з оголошень: наші користувачі розміщують оголошення, кожне оголошення відноситься тільки до однієї категорії товарів і розміщено тільки в одному регіоні. Звідси виникає нехитрий алгоритм:

- Розіб'ємо всі оголошення користувачів на чотири (або N у загальному випадку) категорії: автомобілі, спецтехніка, послуги та нерухомість. Тепер наш набір даних можна розбити на ці підкатегорії: наприклад, в одному наборі даних дивитися виручку користувача тільки в цій підкатегорії.
- Поділимо набір даних за місяцями: витрат користувача за листопад, за грудень тощо.
- Ще всі метрики можна поділити за адміністративно-територіальними одиницями: місто, громада, район, область тощо.
- Об'єднаємо всі три правила в одне. Наприклад: набір даних витрат користувача в сервісі оголошень за жовтень у Києві.
- Тепер у нас є велика кількість наборів даних і в кожному з них є користувачі. Поділимо користувачів випадково на тест і контроль й отримаємо фінальні набір даних для валідації придуманих статистичних критеріїв.

Якщо порівнювати два підходи, то головні переваги штучних даних у тому, що їх скільки завгодно, вони генеруються швидко, й ви повністю контролюєте розподіл. Можна створити нескінченно багато наборів даних й дуже точно оцінити помилку першого роду вашого критерію. На початкових етапах дослідження нового критерію штучні дані значно кращі за реальні. Головний мінус — ви отримали коректність вашого критерію тільки на штучних даних! На реальних же даних критерій може працювати некоректно.

У наборів даних, отриманих на справжніх даних, усе навпаки: зібрати їх велику кількість складно, та й не завжди нормально побудований процес їх збору. Але адекватна оцінка коректності критерію для перевірки гіпотез у вашій компанії можлива тільки в такий спосіб. Завжди можна реалізувати такий критерій, який буде правильно працювати на штучних даних. Але, зіткнувшись у реальності з більш шумними даними, він може почати помилятися частіше, ніж у 5% випадків. Тому важливо переконатися, що саме на справжніх даних метод працюватиме правильно.

5.4 Додаткові питання

5.4.1 t-тест та мала вибірка не з нормального розподілу

У попередньому розділі ми розглянули t-тест для вибірок з нормального розподілу. Розглянемо екстремальний випадок, коли вибірка мала, а розподіл не нормальний. Ми знову використаємо метод Монте-Карло, щоб перевірити, чи працює t-тест у цьому випадку.

```
test_dist    = expon(scale=20)
control_dist = expon(scale=20)

check_criterion(test_dist=test_dist, control_dist=control_dist, sample_size=10)
```

```
FPR: 0.040
Довірчий інтервал: (0.036, 0.044)
```

Тут FPR статистично значущо менше 5%, а отже, використовувати t-тест **можна**. Тільки треба бути готовим, що він буде не дуже потужним.

5.4.2 Як обрати критерій

Нехай у вас є два критерії, й обидва валідні на наших даних. Як зрозуміти на практиці, який із них кращий?

Правильна відповідь — треба порівняти потужність 2 критеріїв! Але як її дізнатися?

Пропонується повторити ту саму процедуру, що ми робили вище, тільки замість генерації експерименту, коли вірна H_0, генерувати експеримент, коли вірна альтернатива. У разі порівняння середніх — треба додати ефект до тесту. І замість FPR рахувати TPR — частка правильно відхилених нульових гіпотез. Чим більше — тим краще.

Перевіримо на прикладі t-тесту, як це працює.

```
rej_cnt = 0
N = 10000
alpha=0.05

sample_dist = norm(loc=2, scale=3)
mu=sample_dist.expect()

for i in range(N):
    test    = sample_dist.rvs(15)
    control = sample_dist.rvs(15) * 2
    pvalue = ttest_ind(test, control, equal_var=False, alternative='two-sided').pvalue
    rej_cnt += (pvalue < alpha)

print(f"TPR: {rej_cnt/N:.3f}")
```

```
TPR: 0.194
```

Бачимо, що потужність критерію в цьому випадку дорівнює 0.19. Якщо є другий критерій — треба запустити таку перевірку для 2го критерію й оцінити, який критерій кращий чи гірший, не забувши про статичну значущість.

Ще є питання: ви оцінили 2 критерії лише при додаванні одного ефекту, наприклад у випадку вище, коли $\mu_T = \mu_C * 2$. А якби була інша зміна, збереглися б результати, що цей критерій кращий? Не факт, тому треба підбирати такий ефект, який найчастіше зустрінеться на практиці. Ваше завдання ще правильно зімітувати ефект, схожий на справжній.

Логіка тут точно така сама, як і чому краще генерувати експерименти на історичних даних, а не на справжніх.

Тобто, ваше завдання для оцінки потужності критерію полягає в:

1. Створенні 1000 експериментів, на історичних даних, або на симульованих
2. Підборі ефекту, який буде найкраще імітувати істинний ефект, що перевіряється, в гіпотезі.

Список літератури

Basu, D. 1955. *On Statistics Independent of a Complete Sufficient Statistic. Sankhya*. Vol. 15.

Cochran, William G. 1934. *The Distribution of Quadratic Forms in a Normal System, with Applications to the Analysis of Covariance. Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 30. 3. Cambridge University Press. https://doi.org/10.1017/S0305004100016595.

Gnedenko, Boris V., and Alexander N. Kolmogorov. 2021. *Limit Distributions for Sums of Independent Random Variables*. Martino Fine Books.

Hogg, Robert V., Elliott A. Tanis, and Dale L. Zimmerman. 2015. *Probability and Statistical Inference*. 9th ed. Pearson.

Lemons, Don S. 2002. *An Introduction to Stochastic Processes in Physics*. The Johns Hopkins University Press.

Wilson, Edwin Bidwell. 1927. *Probable Inference, the Law of Succession, and Statistical Inference. Journal of the American Statistical Association*. Vol. 22. 158.