

Прикладний статистичний аналіз

Посібник для студентів

Ігор Мірошніченко

Юлія Хлевна

2025-04-10

Зміст

Передмова	2
Вступ	3
1 Біноміальний критерій	4
1.1 Генеральна сукупність та вибірка	4
1.2 Статистичні гіпотези	5
1.3 Статистичні функції в Python	10
1.4 p -значення	14
1.5 Двосторонні критерії	15
1.6 Готові функції	20
1.7 Висновки	21
1.8 Питання для самоперевірки	21
2 Статистична потужність, ефект та довірчі інтервали	22
2.1 Статистична потужність	22
2.2 Потужність для біноміального розподілу	24
2.3 Мінімальна величина ефекту	28
2.4 Довірчі інтервали	29
2.5 Односторонні довірчі інтервали	31
2.6 Властивості довірчих інтервалів	33
3 Z-критерій Фішера	40
3.1 Нормальний розподіл	40
3.2 Нормальний розподіл у Python	41
3.3 Властивості нормального розподілу	43
3.4 Центральна гранична теорема	43
3.5 Нормальна апроксимація й застосування Z -критерію	44
3.6 Z -критерій Фішера в Python	46
3.7 Поправка на неперервність	46
4 t-критерій Стюдента	59
4.1 T -тест Стюдента	63
4.2 T -тест у Python	65
4.3 Довірчі інтервали	67
4.4 Довірчі інтервали у Python	68
4.5 T -тест та вимога нормальності	69
5 Монте-Карло в задачах статистики	70
Підсумки	71
Список літератури	72

Передмова

Вступ

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 змодельємо можливі результати для 30 студентів.

Давайте спростимо цю задачу до прикладу з підкиданням монетки та змодельємо 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.

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.

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

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

Лістинг 1.1 Підкидання монетки

```
rng = np.random.default_rng(seed=18) ①
n = 30 ②
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%

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 Бернуллі³. 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.

³Розподіл 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 розподілу⁴.

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

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 $\mu = 0.5$.

Лістинг 1.2 Функція щільності ймовірностей для біноміального розподілу

```
n = 30
mu = 0.5

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

plt.bar(x, y, color=turquoise)
plt.bar(x[x >= 20], y[x >= 20], color=red_pink)
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.

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

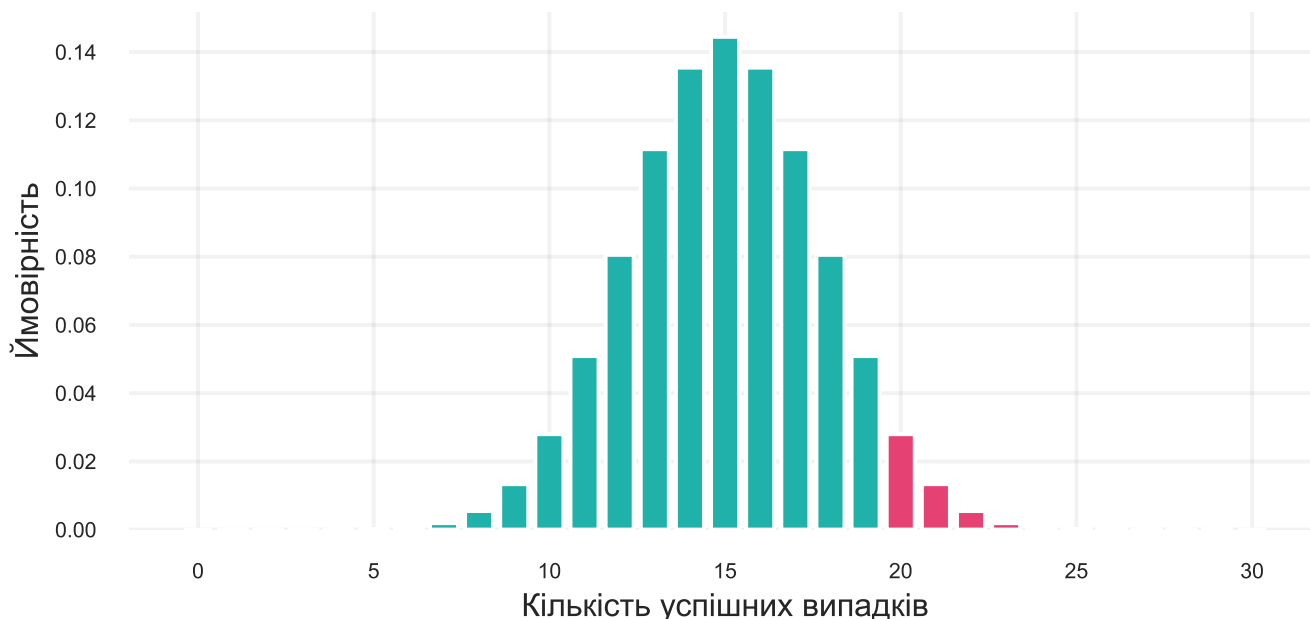


Рисунок 1.1: Візуалізація функції щільності ймовірностей для біноміального розподілу

Ціановим⁵ 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:

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

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

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

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

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

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.

⁵Англ. *cyan*, від грец. *κυανος* — “блакитний”, “лазуровий”.

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.

FP — H_0 відхиляється, коли вона вірна

FN — H_0 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 хибно позитивних 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 $Q \sim \text{Binom}(0.5, 30)$.

Обчислимо FPR для $C = 21$

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

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 Рисунок 1.2.

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

plt.bar(x, y, color=turquoise)
plt.bar(x[x >= crit_subs], y[x >= crit_subs], color=red_pink)
for i in range(crit_subs - 2, crit_subs + 4):
    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()
```

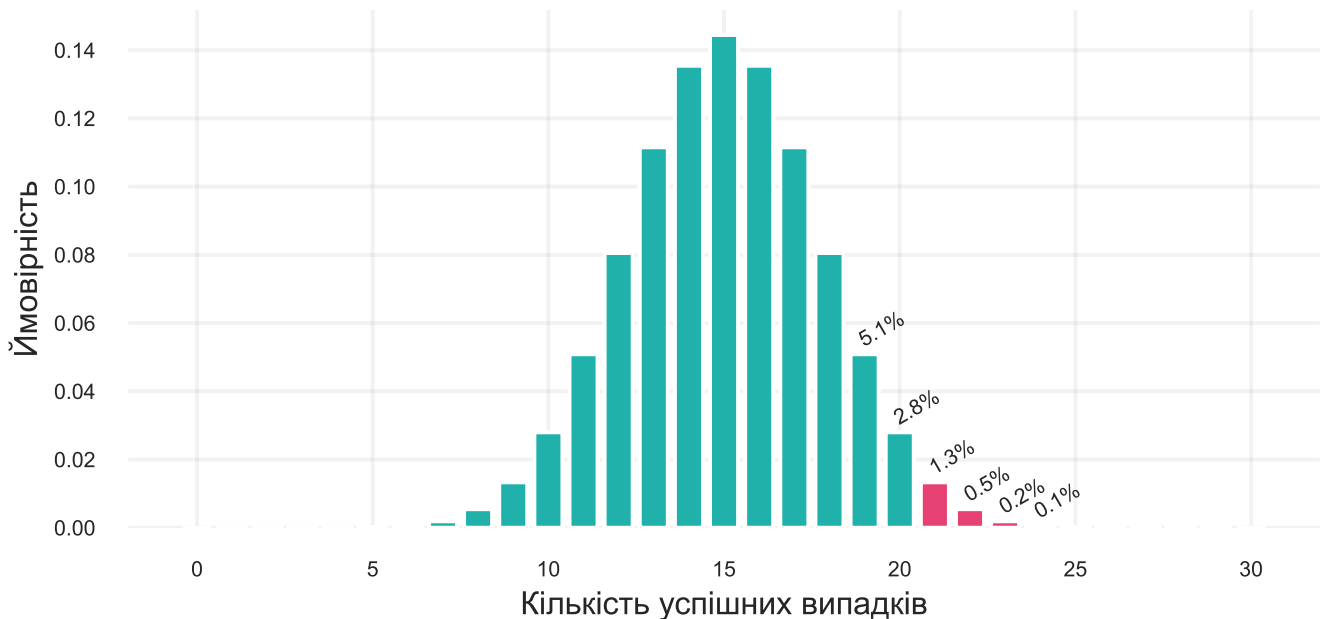


Рисунок 1.2: Ймовірність хибно відхилити H_0 за умови її вірності

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 $\alpha = 100\%$ 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 потрібно 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 $\text{Binom}(n, \mu)$ — 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`.

Лістинг 1.3 Створення біноміального розподілу

```
from scipy.stats import binom

n = 30
mu = 0.5

binom_dist = binom(n, mu)
```

①
②

1. Кількість спостережень.
2. Ймовірність успіху.

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).

Лістинг 1.4 Обчислення ймовірностей біноміального розподілу для кількості успіхів

```
binom_dist.pmf(20)
```

```
0.027981600724160654
```

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], color=red_pink, label="Критичне значення") ⑤
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.

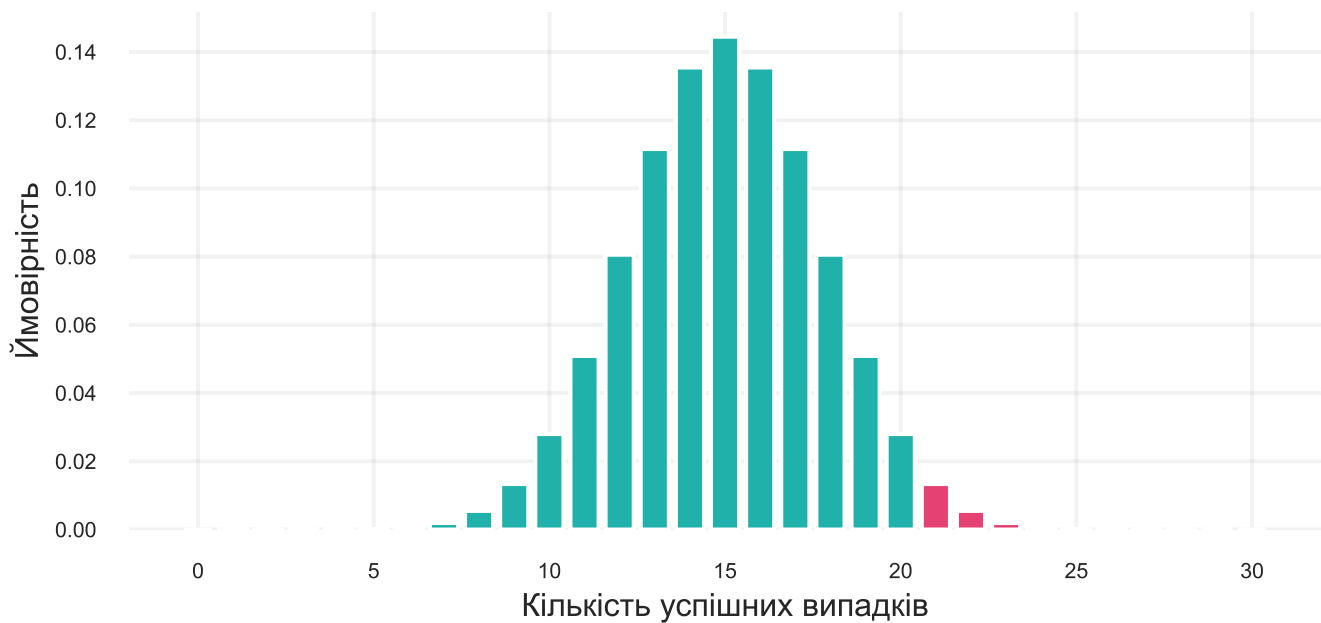


Рисунок 1.3: Функція щільності ймовірностей біноміального розподілу

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

Лістинг 1.5 Обчислення ймовірностей для критичної області

```
np.round(np.sum(y[crit_subs:]), 4)
```

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 $C = 19$?

Лістинг 1.6 Обчислення ймовірностей для критичної області при $C = 19$

```
crit_subs = 19
np.round(np.sum(y[crit_subs:]), 4)
```

0.1002

Тоді ймовірність помилки вже навіть більше 10%, що зовсім нам не підходить.
А якщо $C = 20$?

Лістинг 1.7 Обчислення ймовірностей для критичної області при $C = 20$

```
crit_subs = 20
np.round(np.sum(y[crit_subs:]), 4)
```

0.0494

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

1.3.3 Кумулятивна функція розподілу

Кумулятивна функція розподілу $F_\xi(x) = P(\xi \leq x)$

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

Лістинг 1.8 Обчислення кумулятивної функції розподілу біноміального розподілу

```
binom_dist.cdf(19)
```

①

0.9506314266473055

Ймовірність отримати 19 або менше успіхів у нашому завданні ≥ 0.95 . А оскільки $P(\xi \leq 19) + P(\xi \geq 20) = 1$,
можемо обчислити рівень значущості нашого критерію.

0.04936857335269451

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 *квантилем*.

$$u_p(\xi) = \{x \mid F_\xi(x) = p\}$$

Але при $p = 0.95$ й нашому біноміальному розподілу, такої точки немає. Ми з'ясували, що є точка,
праворуч від якої площа 0.494, а в наступній 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 $F_\xi(u_p) \geq p$.

$$u_p(\xi) = \min \{x \mid F_\xi(x) \geq p\}$$

Лістинг 1.9 Обчислення рівня значущості критерію

```
1 - binom_dist.cdf(19)
```

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

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

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

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

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 функцію `ppf` (Percent Point Function).

Лістинг 1.10 Обчислення квантилю біноміального розподілу

```
binom_dist.ppf(0.95)
```

19.0

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

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

Лістинг 1.11 Знаходження критичного значення для критерію

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

```
find_crit_subs(30, 0.5, 0.05)
```

20.0

Критичне значення 20, отже підсумковий критерій має такий вигляд

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

$Q = 19$, 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 перебудувати 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:

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

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 .

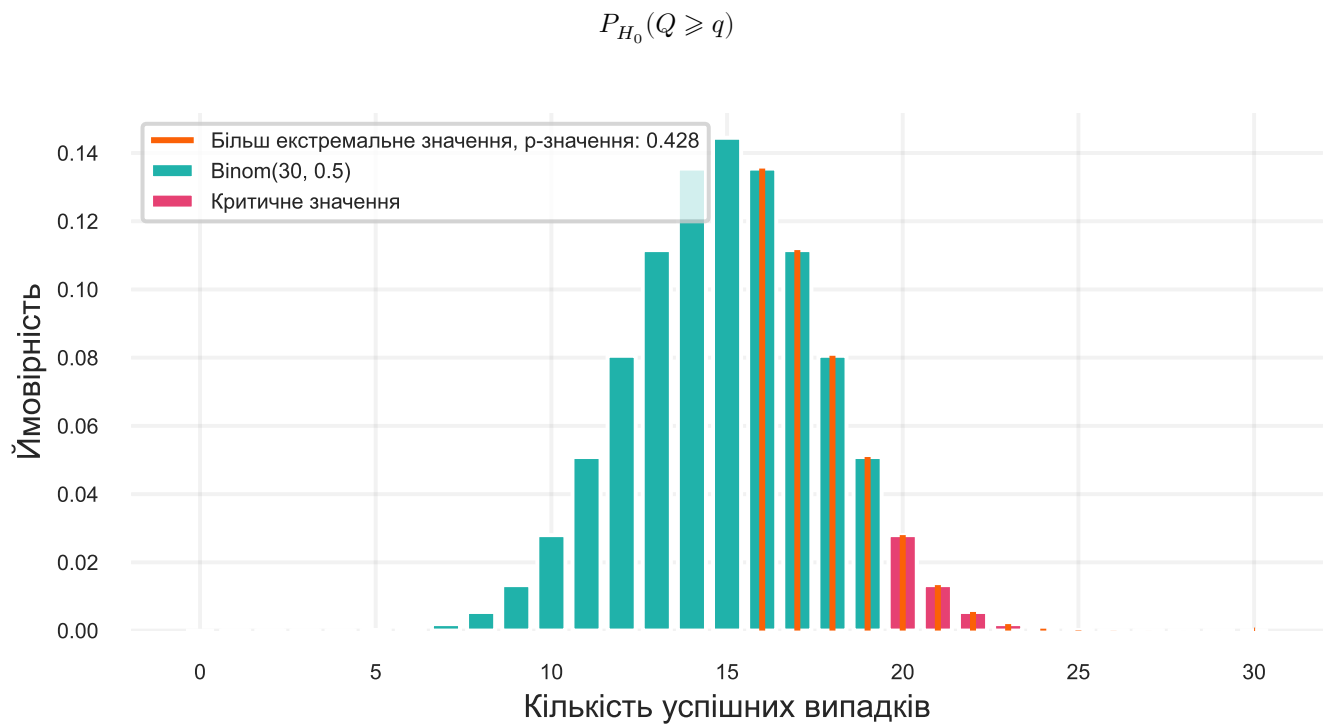


Рисунок 1.4: p -значення для критерію $Q = 15$

Тепер виведемо формулу через функції Python:

$$P_{H_0}(Q \geq q) = 1 - P_{H_0}(Q < q) = 1 - F(q)$$

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

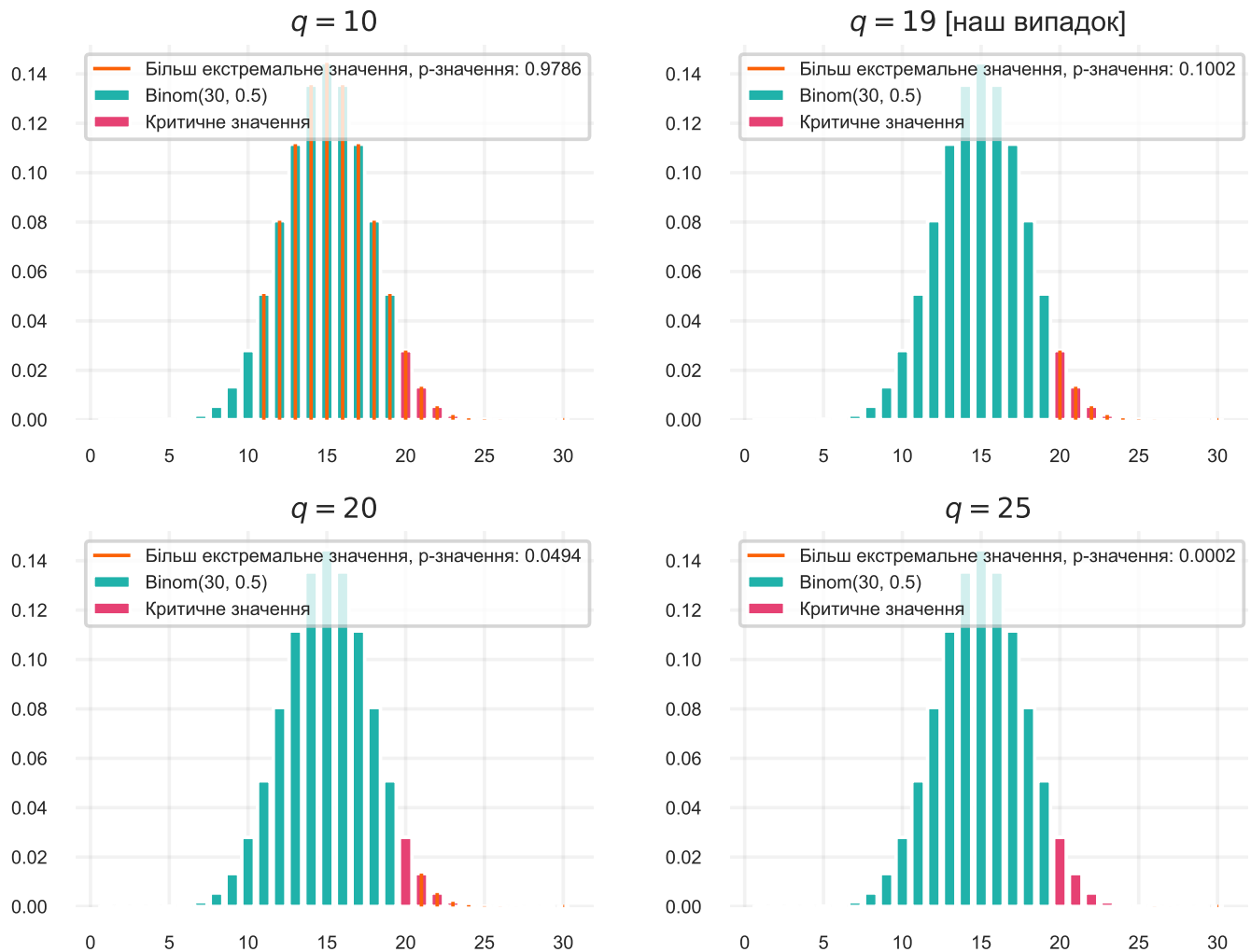


Рисунок 1.5: p -значення для критерію $Q = 10, 19, 20, 25$

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

$$H_0 \text{ відкидається} \Leftrightarrow p - \leq \alpha$$

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:

$$S = \{Q \geq C\} \cup \{Q \leq n - C\}$$

1.5.1 Як вибрати критичну область

Подивимося, який вигляд матиме критична область у такому разі.

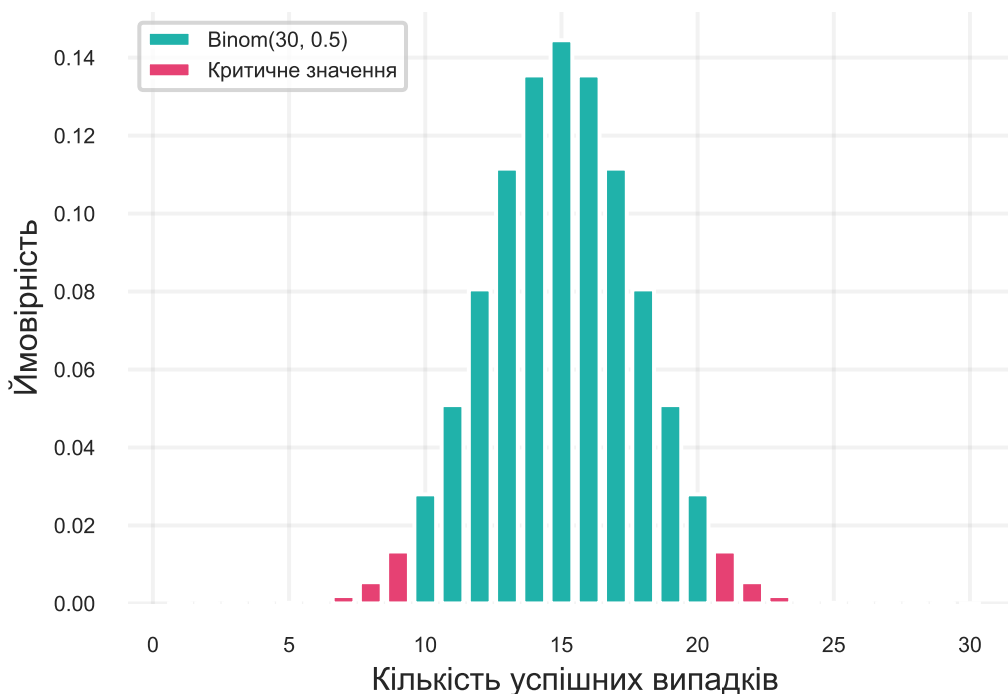


Рисунок 1.6: Двостороння критична область для критерію Q

З картинки видно, що якщо тепер відкидати відхилення за $Q \geq 20$, то необхідно відкидати й $Q \leq 10$, 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.

Лістинг 1.12 Знаходження критичного значення для двостороннього критерію

```
def find_crit_subs_two_sided(n, mu, alpha):  
    binom_dist = binom(n, mu)  
    return n / 2 - binom_dist.ppf(alpha / 2) + 1  
  
find_crit_subs_two_sided(30, 0.5, 0.05)
```

1.5.2 Як знайти p -значення

Критерій має вигляд

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

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

$$S = \{\Delta(\xi) \geq C\}$$

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

Лістинг 1.13 Обчислення p -значення для двостороннього критерію

```
def pvalue_two_sided_sym(n, q):  
    binom_h0 = binom(n=n, p=0.5)  
    diff = np.abs(q - 15)  
    right_sq = 1 - binom_h0.cdf(15 + diff - 1)  
    left_sq = binom_h0.cdf(15 - diff)  
    return left_sq + right_sq  
  
pvalue_two_sided_sym(30, 21)
```

0.04277394525706769

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

Лістинг 1.14 Обчислення p -значення для двостороннього критерію (спрощено)

```
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  
  
pvalue_two_sided_sym_simple(30, 21)
```

0.04277394525706768

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)

probs = binom_h0_nonsym.pmf(np.arange(31))

plt.bar(np.arange(31), probs, color=turquoise, label="Binom(30, 0.8)")
plt.legend(fontsize=8)
plt.show()
```

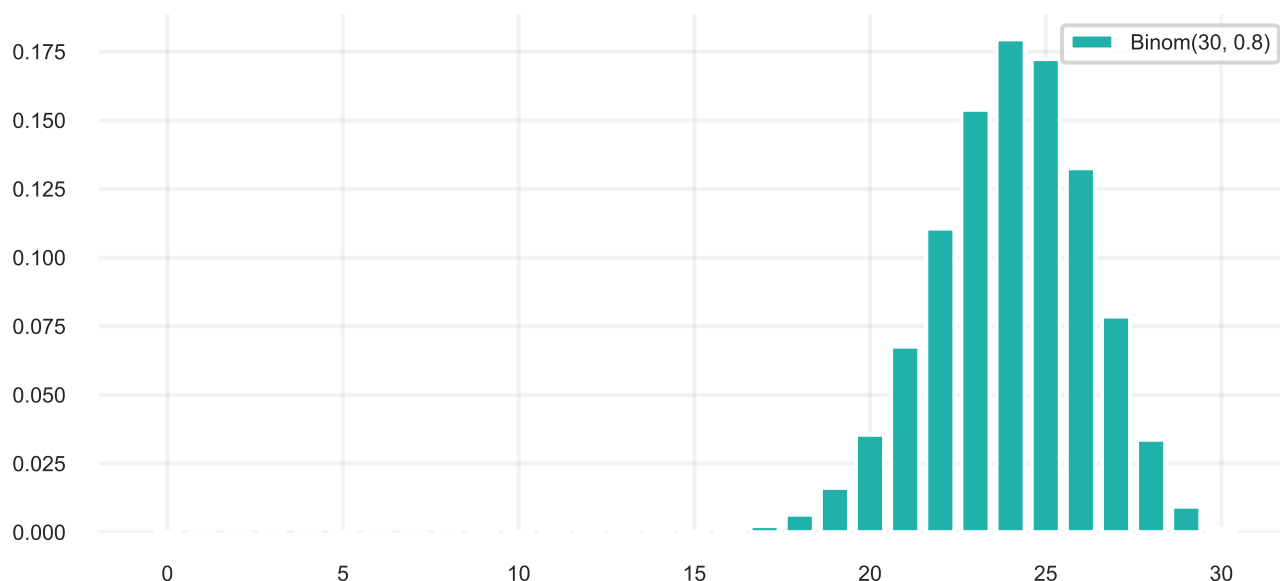


Рисунок 1.7: Біноміальний розподіл з імовірністю успіху 0.8

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

$$P(Q(\xi) \leq C) \leq \frac{\alpha}{2}$$

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

(18.0, 29.0)

Отже, наш критерій для перевірки гіпотези

$$H_0 : \mu = 0.8$$

$$H_1 : \mu \neq 0.8$$

має вигляд

Лістинг 1.15 Знаходження критичного значення для двостороннього критерію

```
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  
  
two_sided_criterion_nonsym(30, 0.8, 0.05)
```

$$S = \{Q(\xi) \leq 18\} \cup \{Q(\xi) \geq 29\}$$

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)  
  
plt.figure(figsize=(6, 4))  
plt.bar(np.arange(31), probs, color=turquoise, label="Binom(30, 0.8)")  
plt.bar(np.arange(31)[np.arange(31) <= C1], probs[np.arange(31) <= C1], color=red_pink, label=  
plt.bar(np.arange(31)[np.arange(31) >= C2], probs[np.arange(31) >= C2], color=red_pink)  
plt.xlabel("Кількість успішних випадків")  
plt.ylabel("Ймовірність")  
plt.legend(fontsize = '8', loc = 'upper left')  
plt.show()
```

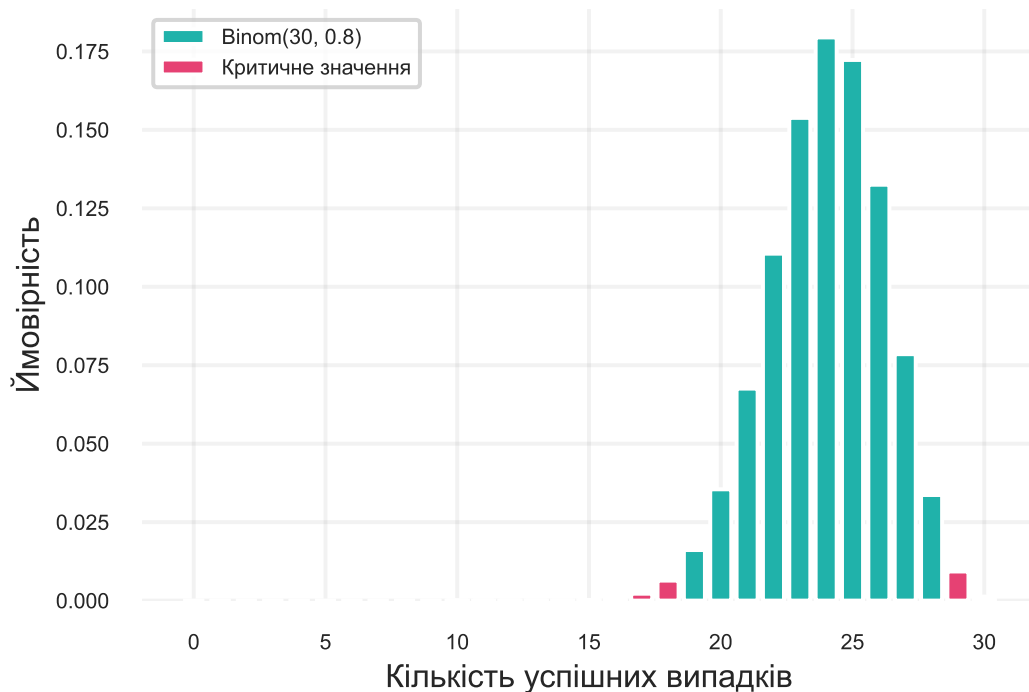


Рисунок 1.8: Двостороння критична область для критерію $C_1 = 18, C_2 = 29$

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 $p_1 \leq \frac{\alpha}{2}$, другий при $p_2 \leq \frac{\alpha}{2}$. А наш об'єднаний, коли виконано одну з цих умов, тобто

$$2p_1 \leq \alpha \vee 2p_2 \leq \alpha \Leftrightarrow 2 \cdot \min(p_1, p_2) \leq \alpha$$

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.

Лістинг 1.16 Обчислення p -значення для двостороннього критерію з несиметричним розподілом

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

```
pvalue_two_sided(30, 28, 0.8)
```

```
0.08835797030399428
```

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.

Лістинг 1.17 Обчислення p -значення для двостороннього критерію з симетричним розподілом

```
pvalue_two_sided(n=30, q=20, mu=0.5)
```

```
0.09873714670538902
```

Лістинг 1.18 Знаходження p -значення для двостороннього критерію

```
pvalue_two_sided_sym(n=30, q=20)
```

```
0.09873714670538904
```

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

Лістинг 1.19

```
from scipy.stats import binomtest

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

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

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

1.7 Висновки

Ми розглянули, як можна використовувати біноміальний розподіл для перевірки гіпотези про ймовірність успіху. Для цього ми визначили критерій, критичну область, p -значення. Показали, як можна використовувати ці поняття для різних видів гіпотез: односторонніх, двосторонніх, симетричних та несиметричних.

1.8 Питання для самоперевірки

1. Які гіпотези можна перевірити за допомогою біноміального розподілу?
2. Як визначити критичну область для критерію?
3. Як визначити p -значення для критерію?
4. Як визначити критичну область для двостороннього критерію?
5. Як визначити p -значення для двостороннього критерію?
6. Як визначити критичну область для несиметричного розподілу?
7. Як визначити p -значення для несиметричного розподілу?

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 : на вулиці літо

H_1 : 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.

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

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.

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

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.

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

Обчислимо FNR.

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

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).

$$\text{Power}_S = 1 - FNR \quad (2.1)$$

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.

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

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 $FPR \leq \alpha$. 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 відповідь формула потужності.

Згадаймо критерій, за яким ми приймаємо рішення:

$$Q(\xi) = \sum_{i=1}^n \xi_i - \text{кількість підписок}$$
$$S = \{Q \geq 20\}$$

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 $\text{Binom}(30, 0.6)$.

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

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

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

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

plt.legend(fontsize=8)
plt.show()
```

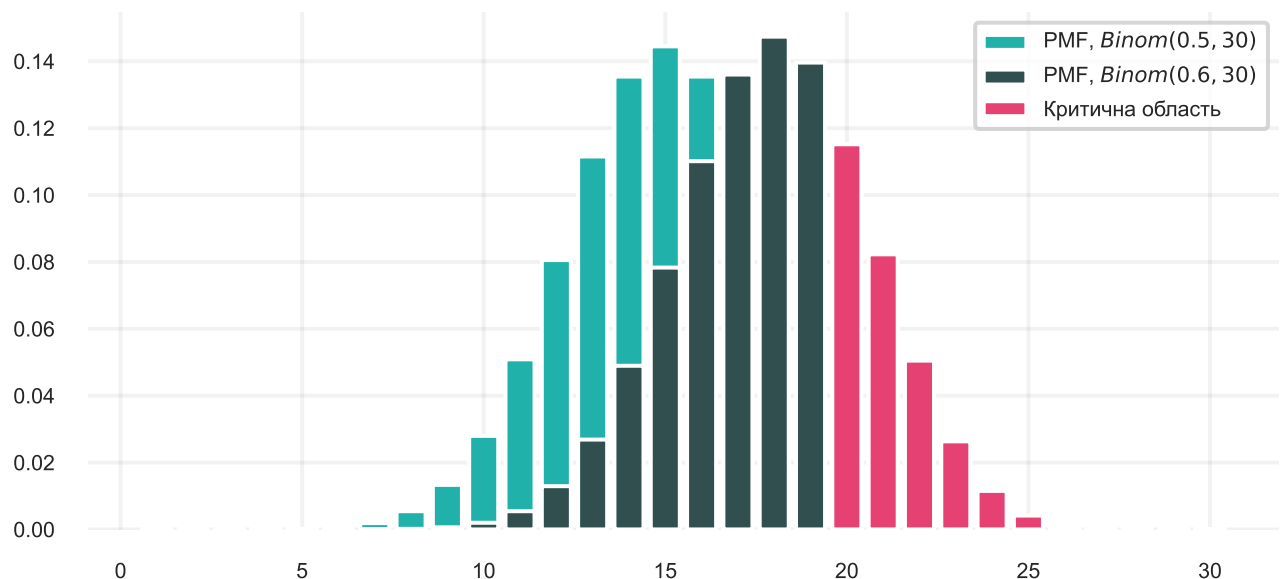


Рисунок 2.1: Потужність критерію для $\mu = 0.6$

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

Лістинг 2.1 Обчислення потужності критерію

```
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.

Лістинг 2.2 Обчислення потужності критерію для загального випадку

```
def get_stat_power(N, mu_h0, mu_alternative, alpha):
    '''Обчислює статистичну потужність критерію для біноміального розподілу

    Параметри:
        N - кількість бернуллієвських експериментів (розмір вибірки)
        mu_h0 - імовірність успіху в нульовій гіпотезі
        mu_alternative - передбачувана ймовірність успіху в експерименті
        alpha - рівень значущості критерію
    '''
    binom_h0 = binom(n=N, p=mu_h0)
    binom_alternative = binom(n=N, p=mu_alternative)
    critical_value = binom_h0.ppf(1 - alpha) + 1
    return 1 - binom_alternative.cdf(critical_value - 1)

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

0.2914718612234968

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.

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

0.9655326717180749

Лістинг 2.3 Обчислення потужності критерію для 300 клієнтів

```
get_stat_power(300, 0.5, 0.6, alpha=0.05)
```

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)
power = get_stat_power(n_grid, 0.5, 0.6, alpha=0.05)

plt.xlabel('Кількість пробних занять')
plt.ylabel('Потужність')

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

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

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

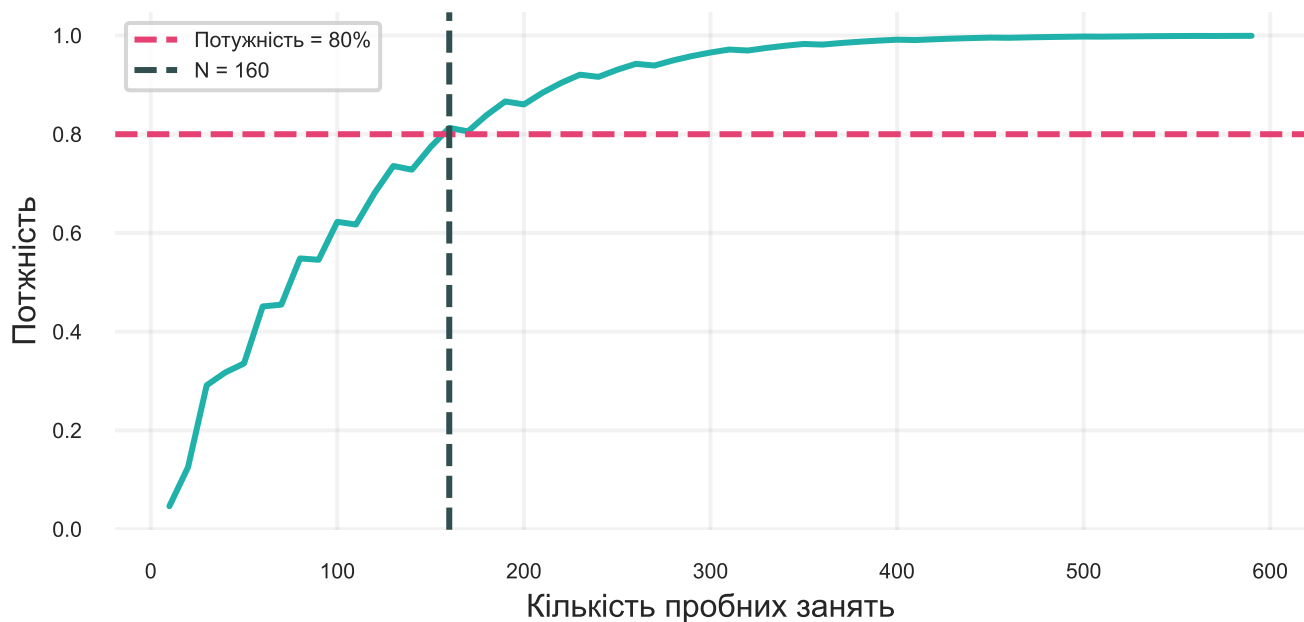


Рисунок 2.2: Залежність потужності від розміру вибірки для $\mu = 0.6$

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)
power = get_stat_power(n_grid, 0.5, 0.51, alpha=0.05)

plt.xlabel('Кількість пробних занять', fontsize=8)
plt.ylabel('Потужність', fontsize=8)

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

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

plt.legend()
plt.show()

```

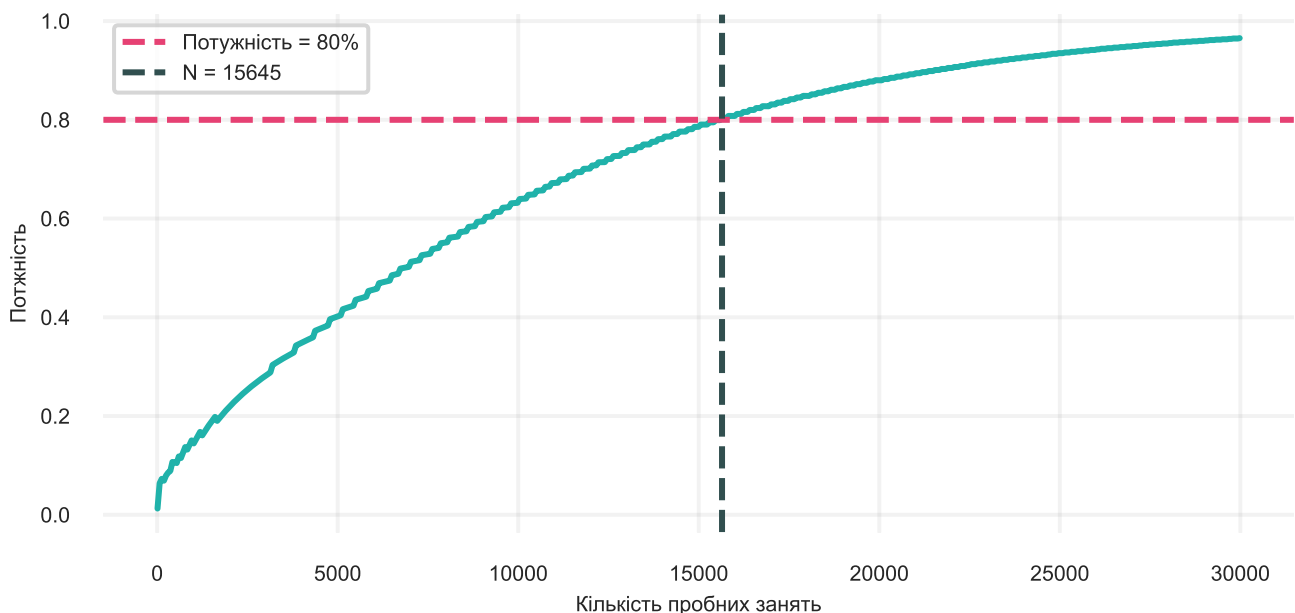


Рисунок 2.3: Залежність потужності від розміру вибірки для $\mu = 0.51$

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)
power = get_stat_power(30, 0.5, mu_grid, alpha=0.05)

```

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

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

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

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

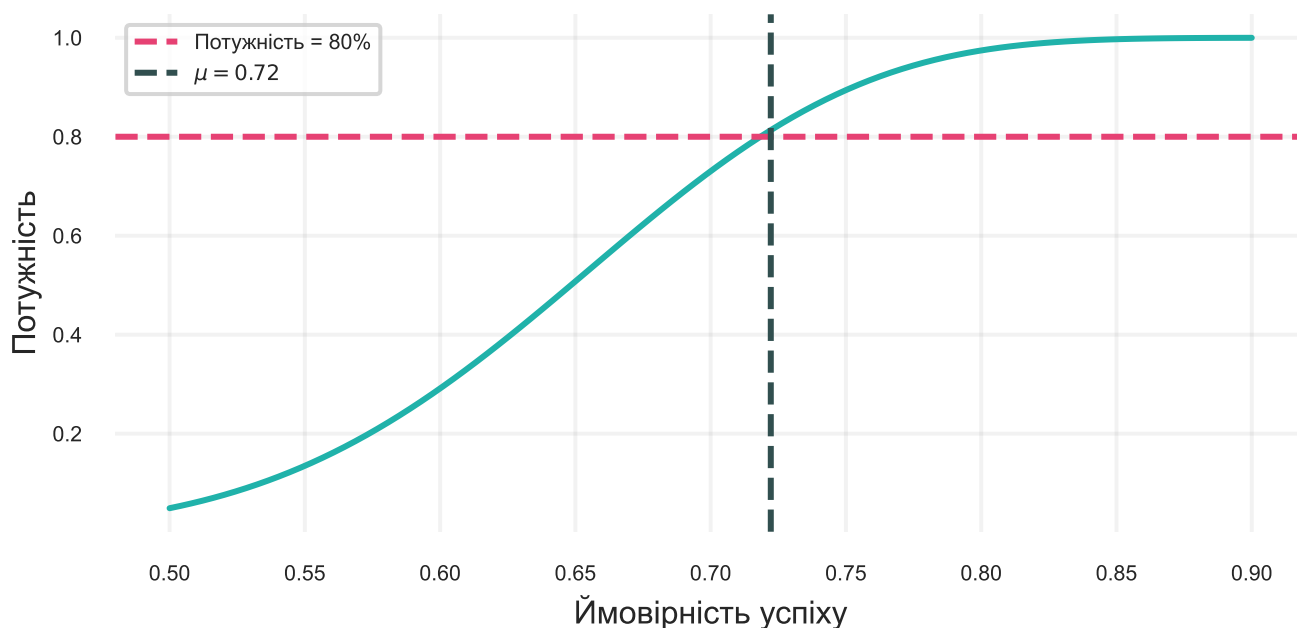


Рисунок 2.4: Залежність потужності від параметра μ

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.

```
0.21843687374749496
```

Лістинг 2.4 Обчислення MDE

```
def binom_test_mde_one_sided(N, mu0, alpha=0.05, min_power=0.8):
    delta_grid = np.linspace(0, 1 - mu0, 500)
    power = get_stat_power(N, mu0, mu0 + delta_grid, alpha=alpha)
    fit_delta = delta_grid[power >= min_power]
    return fit_delta[0]

binom_test_mde_one_sided(30, 0.5)
```

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 зворотнє 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.

95% довірчий інтервал: [0.439 - 0.8]

1. Функція, що обчислює критичні значення для двостороннього критерію.
2. Кількість успішних підписок.
3. Сітка значень μ .
4. Список значень μ , для яких гіпотеза не відкидається.
5. Перебір значень μ .

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 $\frac{19}{30}$.

Подивимось, 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.

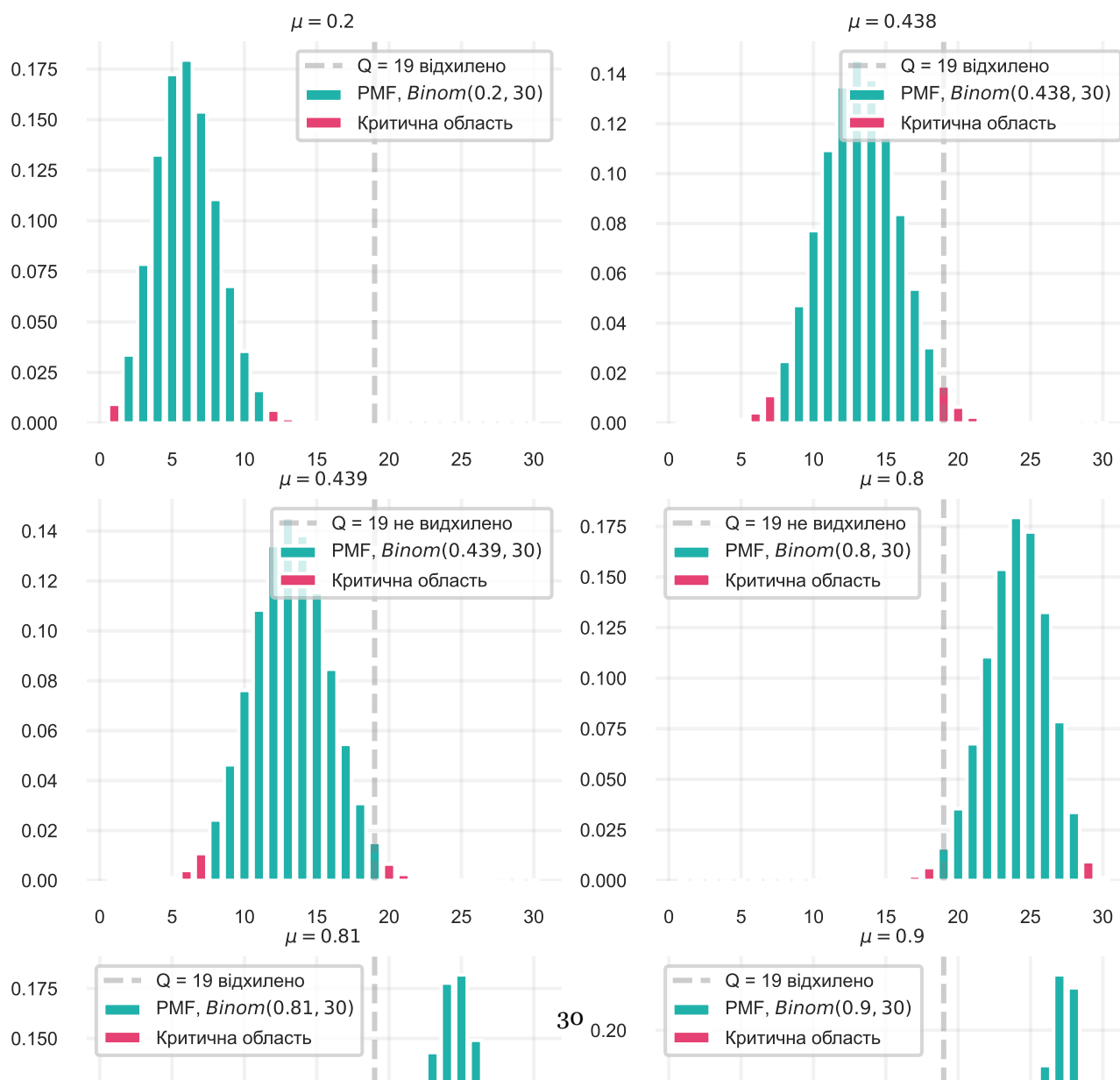
Лістинг 2.5 Довірчі інтервали для біноміального розподілу

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

success_cnt = 19 ②
mu_grid = np.arange(0, 1, 0.001) ③
mu_no_rejection = [] ④

for mu_h0 in mu_grid: ⑤
    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)}]')
```



Лістинг 2.6 Довірчі інтервали для біноміального розподілу

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

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

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

    ax.bar(x_grid, probs, color=turquoise, label=f'PMF, $Binom(\{mu\_h0\}, 30)$')
    c1, c2 = two_sided_criterion_nonsym(30, mu_h0, alpha=0.05)
    crit_reg = (x_grid <= c1) | (x_grid >= c2)
    ax.bar(x_grid[crit_reg], probs[crit_reg], color=red_pink, label='Критична область')

    is_rejection = success_cnt <= c1 or success_cnt >= c2
    ax.axvline(success_cnt, ls='--', label=f'Q = {success_cnt} ' + ('відхилено' if is_rejection))

    rejection_prob = probs[crit_reg].sum()
    ax.set_title(f'$\mu = \{mu\_h0\}$', fontsize=8)
    ax.legend()
```

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.

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.

Двосторонній 95% довірчий інтервал: [0.542 - 0.877]

Односторонній 95% довірчий інтервал: [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.

Лістинг 2.7 Односторонні довірчі інтервали для біноміального розподілу

```
def make_binom_criterion(n, mu=0.5, alpha=0.05):
    binom_h0 = binom(n=n, p=mu)
    q = binom_h0.ppf(1 - alpha)
    return q + 1

success_cnt = 19
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)

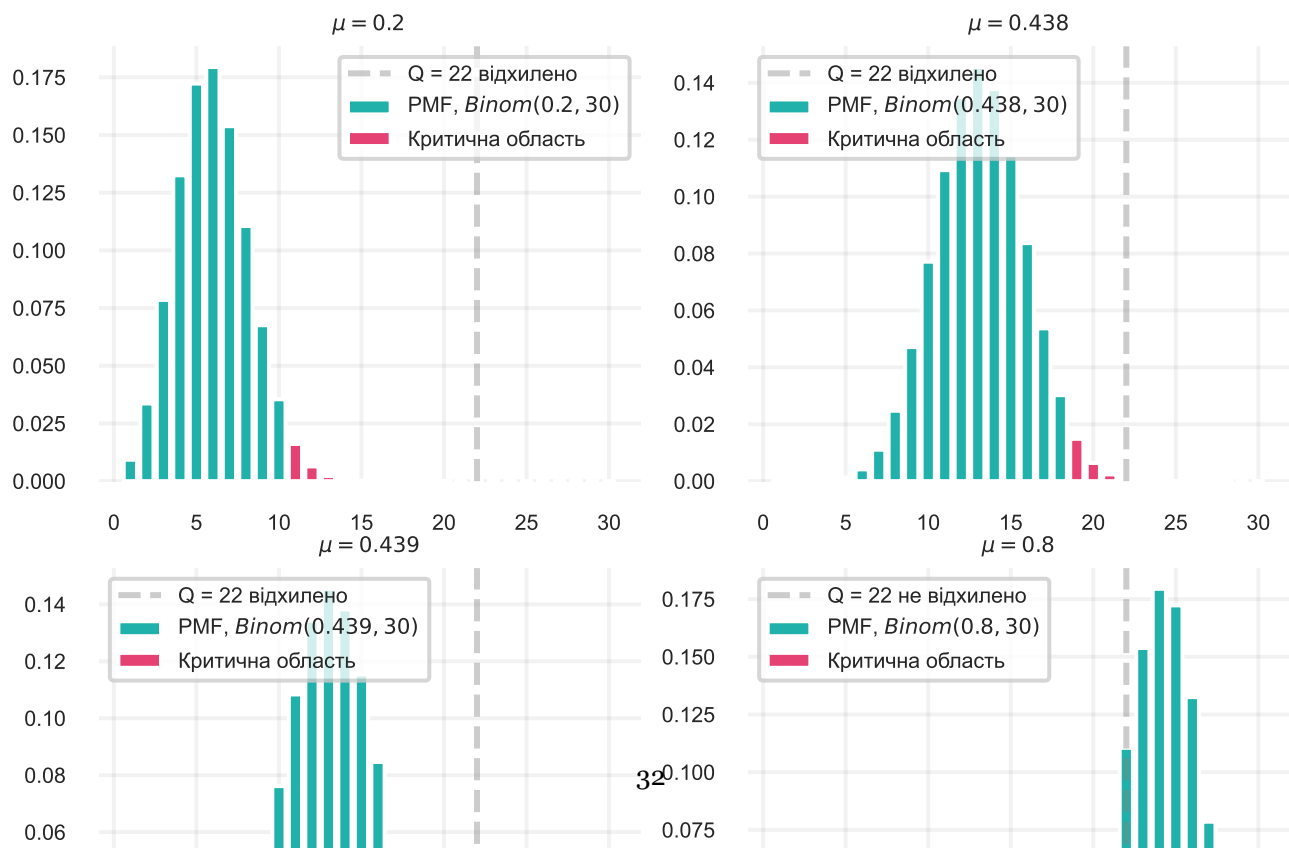
print(f'95% довірчий інтервал: [{min(mu_no_rejection)} - {max(mu_no_rejection)}]')
```

Лістинг 2.8 Двосторонній довірчий інтервал для 22 успіхів

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



Лістинг 2.9 Односторонній довірчий інтервал для 22 успіхів

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

Лістинг 2.10 Односторонній довірчий інтервал для 22 успіхів

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

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

    ax.bar(x_grid, probs, color=turquoise, label=f'PMF, $Binom(\{mu\_h0\}, 30)\$')
    c = make_binom_criterion(30, mu_h0, alpha=0.05)
    crit_reg = (x_grid >= c)
    ax.bar(x_grid[crit_reg], probs[crit_reg], color=red_pink, label='Критична область')

    is_rejection = success_cnt >= c
    ax.axvline(success_cnt, ls='--', label=f'Q = {success_cnt} ' + ('відхилено' if is_rejection))

    rejection_prob = probs[crit_reg].sum()
    ax.set_title(f'$\mu = \{mu\_h0\}$', fontsize=8)
    ax.legend()
```

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 $\alpha = 0.1$.

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 $S = \{Q(\xi) \leq C\}$ 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 $H_0 : \mu = \mu_0$.

Лістинг 2.11 Двосторонній довірчий інтервал для 22 успіхів з $\alpha = 0.1$

```
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}']')
```

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.

$$P(\mathcal{L}(Q) < \mu_0 < \mathcal{R}(Q)) \geq 1 - \alpha \quad (2.2)$$

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%

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

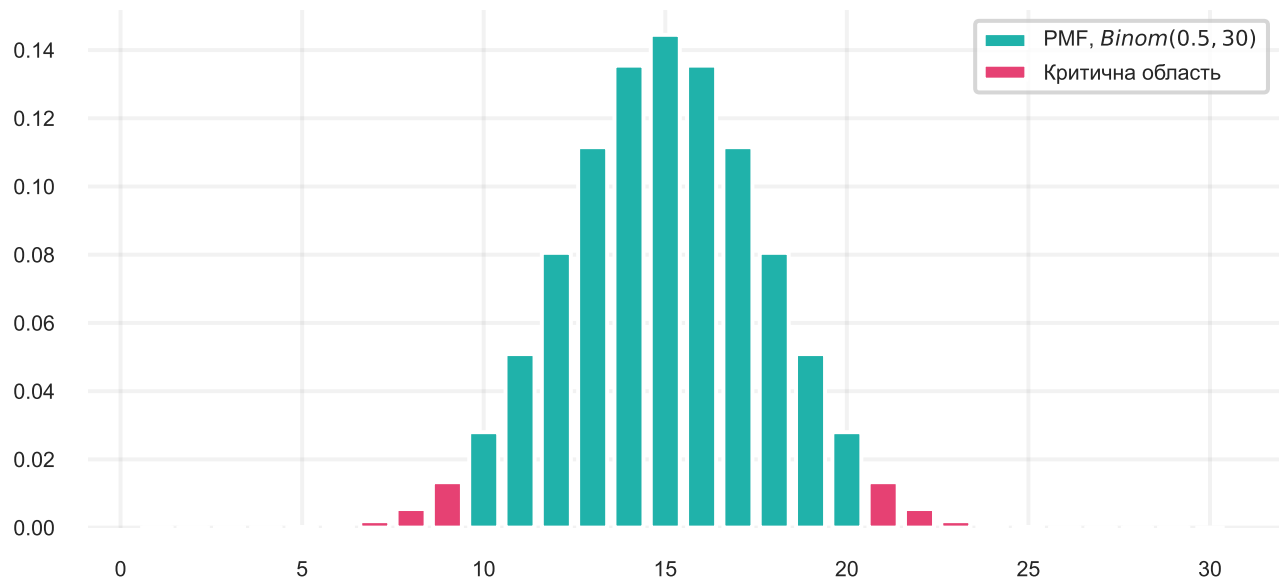


Рисунок 2.7: Розподіл статистики при істинній ймовірності успіху

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 $\leq \alpha$.

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

Часто так й вводять визначення довірчого інтервалу. Для вибірки ξ_1, \dots, ξ_n — це така пара статистик $\mathcal{L}(\xi)$ і $\mathcal{R}(\xi)$, що хоч яким би не було μ_0 ,

$$P(\mathcal{L}(\xi) < \mu_0 < \mathcal{R}(\xi)) \geq 1 - \alpha \quad (2.3)$$

де $\mathcal{L}(\xi)$ і $\mathcal{R}(\xi)$ — 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.

Відсоток успішних випадків: 96.6%

Час виконання: 0.0771 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.

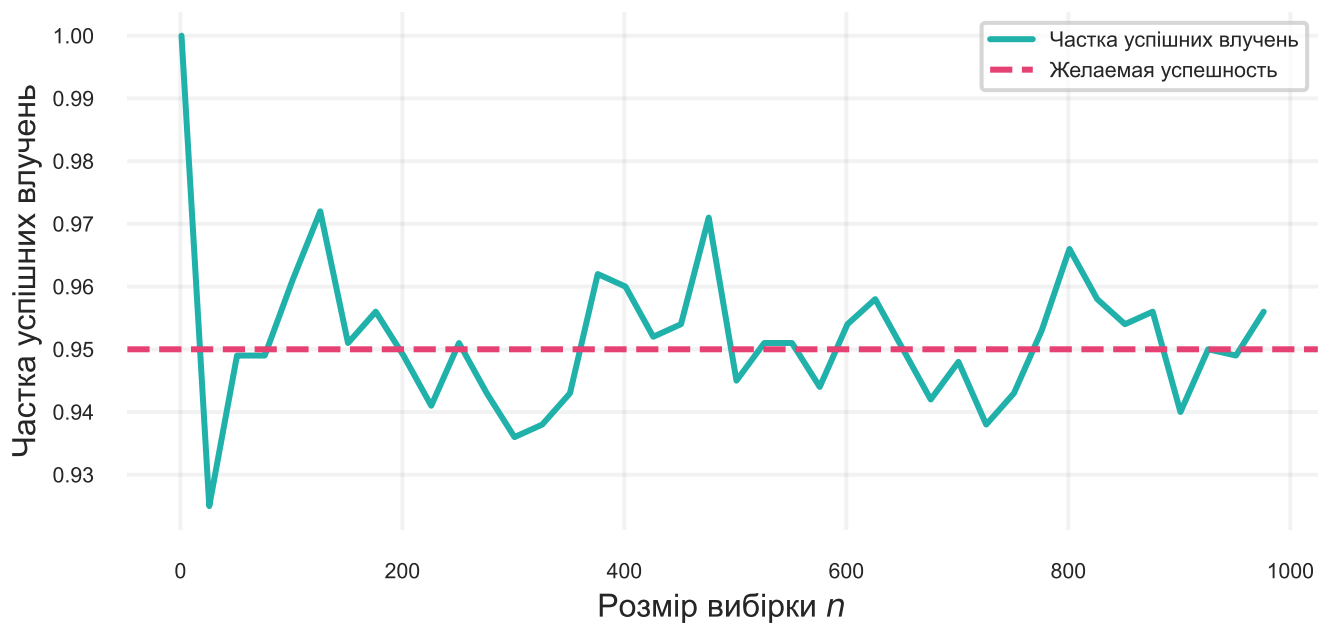


Рисунок 2.8: Залежність частки успішних влучень μ у довірчий інтервал від розміру вибірки

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%.

Лістинг 2.12 Перевірка властивості довірчого інтервалу.

```
import time

start_time = time.time()

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

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

    return min(mu_no_rejection), max(mu_no_rejection)

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 = my_binomial_confint(n=SAMPLE_SIZE, alpha=0.05, q=q)
    if L < latent_mu < R:
        pass
    else:
        confint_fail_cases += 1

success_cases = round(100 * (N_EXPERIMENTS - confint_fail_cases) / N_EXPERIMENTS, 2)

print(f"Відсоток успішних випадків: {success_cases}%")

end_time = time.time()
print(f"Час виконання: {end_time - start_time:.4f} секунди")
```

Відсоток успішних випадків: 61.4%
Час виконання: 3.7842 секунди

Лістинг 2.13 Розподіл статистики при істинній ймовірності успіху

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

plt.bar(x_grid, probs, color=turquoise, label=f'PMF, $Binom(\{\mu_0\}, 30)$')
c1, c2 = two_sided_criterion_nonsym(30, mu0, alpha=0.05)
crit_reg = (x_grid >= c2) | (x_grid <= c1)
plt.bar(x_grid[crit_reg], probs[crit_reg], color=red_pink, label='Критична область')
plt.legend()
plt.show()
```

Лістинг 2.14 Довірчий інтервал Вілсона

```
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 = round(100 * (N_EXPERIMENTS - confint_fail_cases) / N_EXPERIMENTS, 2)
print(f"Відсоток успішних випадків: {success_cases}%")
end_time = time.time()
print(f"Час виконання: {end_time - start_time:.4f} секунди")
```

Лістинг 2.15 Залежність частки успішних влучень μ у довірчий інтервал від розміру вибірки

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

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

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

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

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

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).

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

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()
```

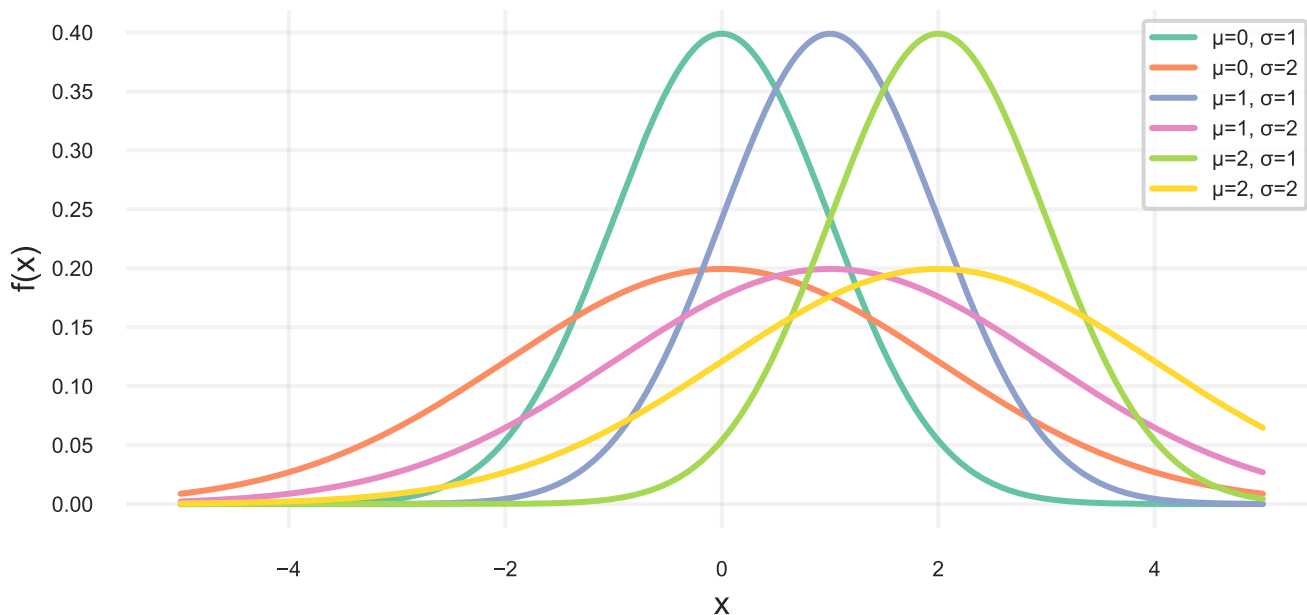


Рисунок 3.1: Нормальний розподіл з різними параметрами

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.

¹Документація 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.

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

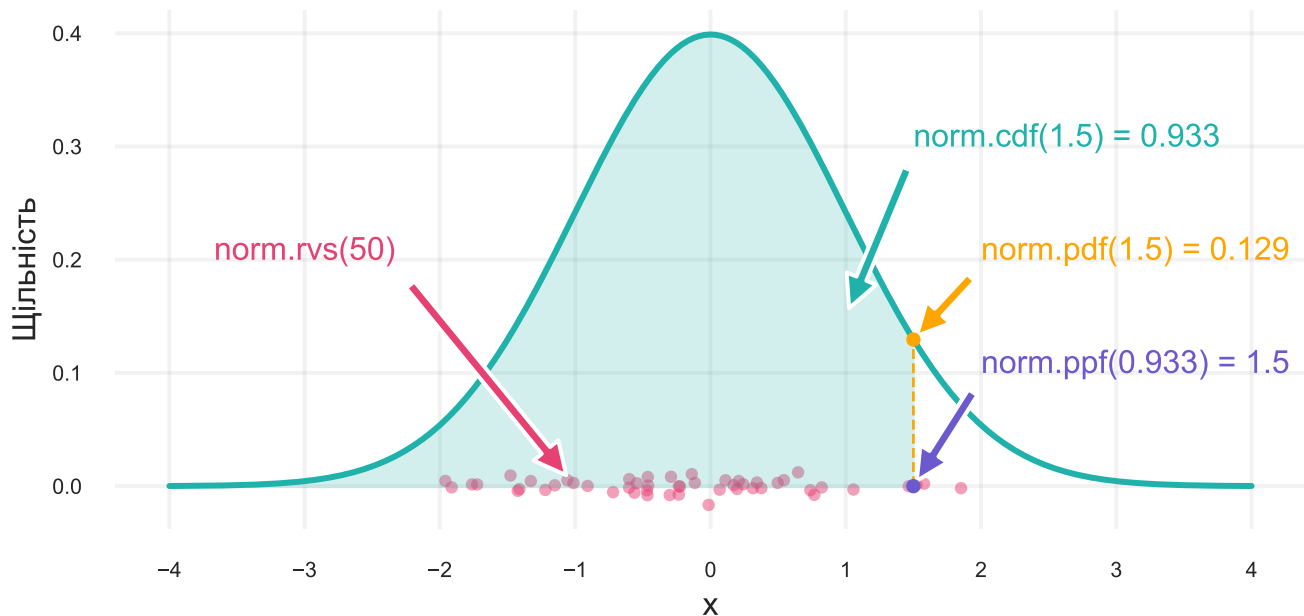


Рисунок 3.2: Демонстрація методів класу `norm`

3.3 Властивості нормального розподілу

Нормальний розподіл має кілька важливих властивостей²:

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 ξ_1, \dots, ξ_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 $\sqrt{n} \frac{\bar{\xi} - \mu}{\sqrt{\sigma^2}}$ збігається за розподілом³ 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

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

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

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

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

i 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.

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

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 $X_1, X_2, \dots, X_n, X_i \sim \text{Bernoulli}(\mu)$ 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:

- $T(X^n) = \sum_{i=1}^n X_i$, $T \stackrel{H_0}{\sim} \text{Binom}(n, \mu_0)$
- 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 (1.19).

А тепер подивимось, що нам говорить ЦГТ:

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

При цьому цього разу ми дивимось 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 більшою кількістю 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:

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

де μ_0 — гіпотетичне значення параметра μ .

Статистика Z-критерію Фішера має вигляд:

$$Z = \frac{\bar{X} - \mu_0}{\sigma / \sqrt{n}} \quad (3.7)$$

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

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

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

Двосторонній критерій набуває вигляду:

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

де α — рівень значущості.

А p -значення для двостороннього критерію розраховується за формулою:

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

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

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

Тоді односторонній Z -критерій Фішера має вигляд:

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

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 — неціле 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.

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

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:

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

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 Стюдента, 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

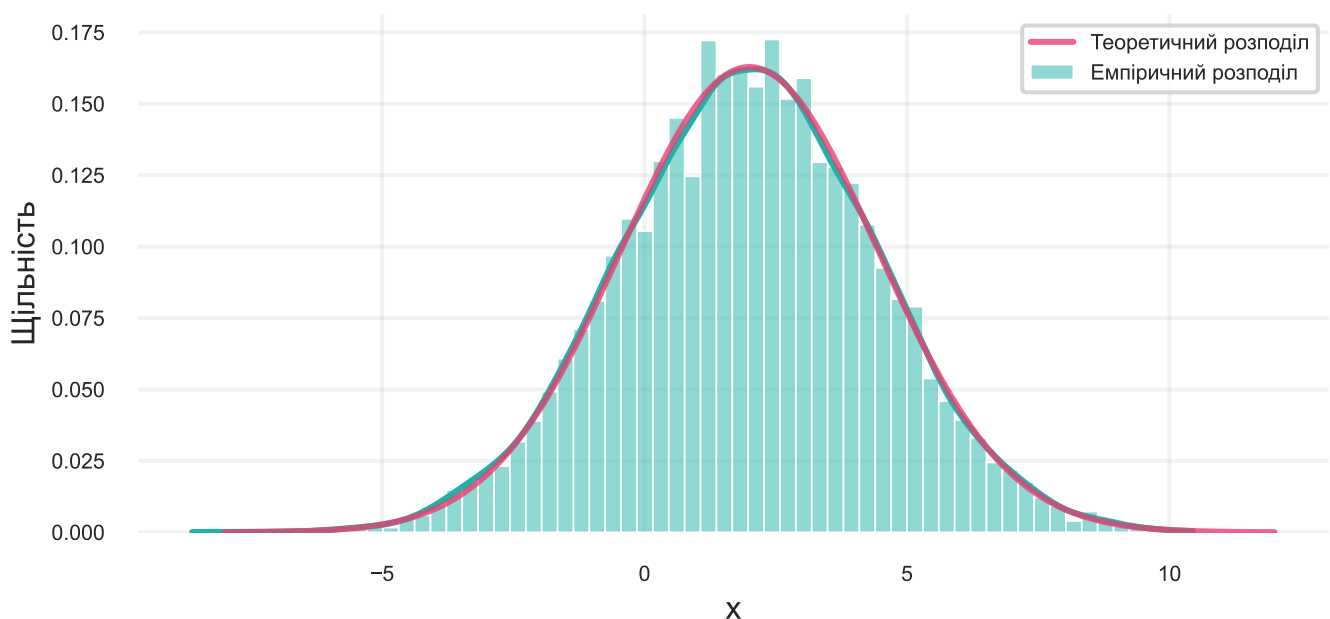
x1 = norm.rvs(loc=mean_one, scale=np.sqrt(var_one), size=n) ④
x2 = norm.rvs(loc=mean_two, scale=np.sqrt(var_two), size=n)

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=ax)
plt.plot(x_grid, check_sum.pdf(x_grid), color=red_pink, label='Теоретичний розподіл', alpha=0.5)
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 $\xi_1 + \xi_2$.

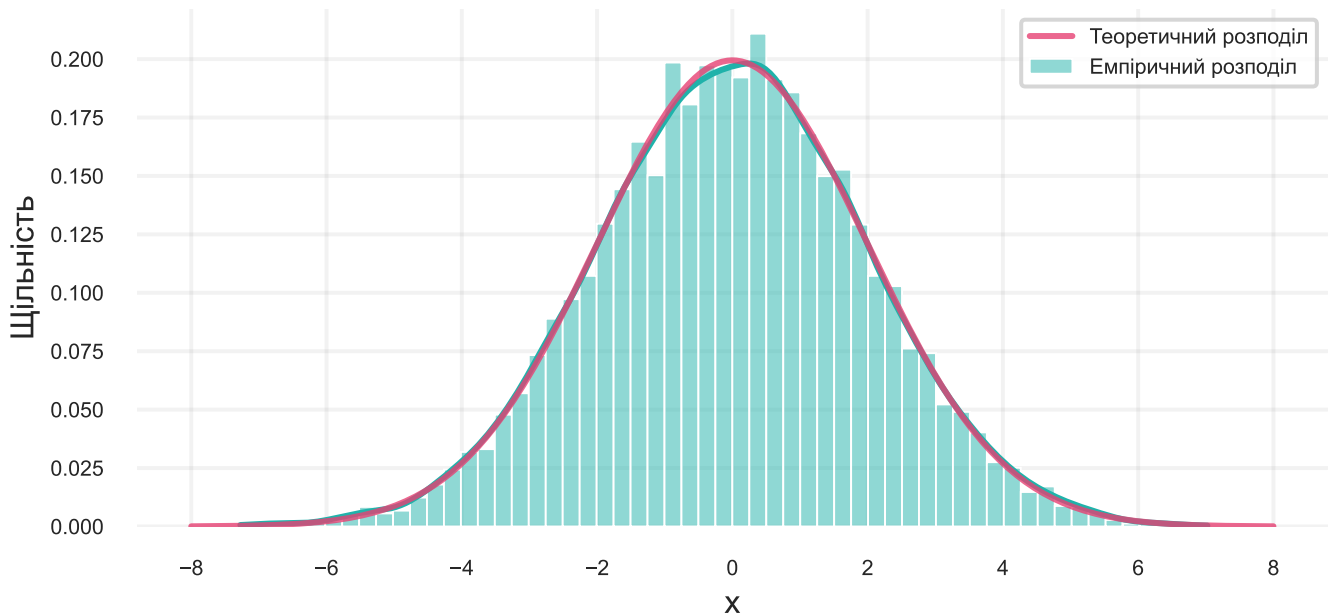


Лістинг 3.4 Візуалізація нормального розподілу множення нормально розподіленої випадкової величини на константу.

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

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()
```

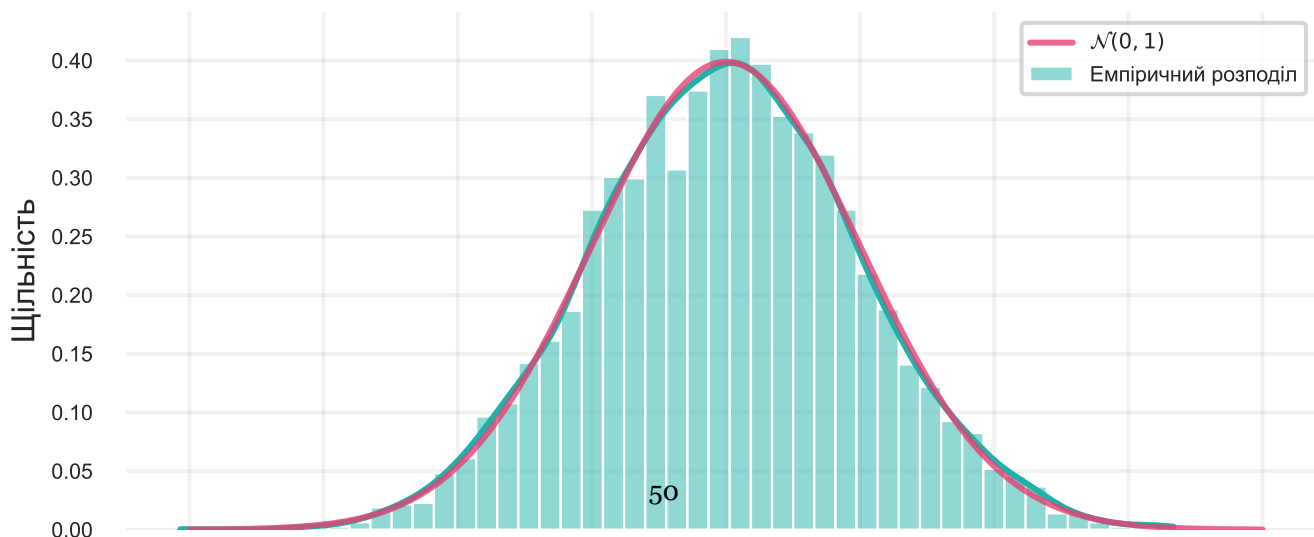
- ① Параметри μ_1 та σ_1^2 .
- ② Параметри σ_1^2 .
- ③ Константа a .
- ④ Кількість спостережень.
- ⑤ Генерація нормально розподіленої випадкової величини ξ_1 .
- ⑥ Множення нормально розподіленої випадкової величини ξ_1 на константу a .
- ⑦ Параметри множення ξ_1 на константу a .
- ⑧ Стандартне відхилення 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  
    clt_sample = []  
    for _ in range(N):  
        sample = sample_generator()  
        sample_size = len(sample)  
        statistic = np.sqrt(sample_size) * (np.mean(sample) - expected_value) / np.sqrt(variance)  
        clt_sample.append(statistic)  
  
    x = np.linspace(-4, 4, 1000)  
    fig, ax = plt.subplots()  
    sns.histplot(clt_sample, kde=True, stat='density', color=turquoise, label='Емпіричний розподіл')  
    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),  
              expected_value = p * n,  
              variance = n * p * (1 - p))
```

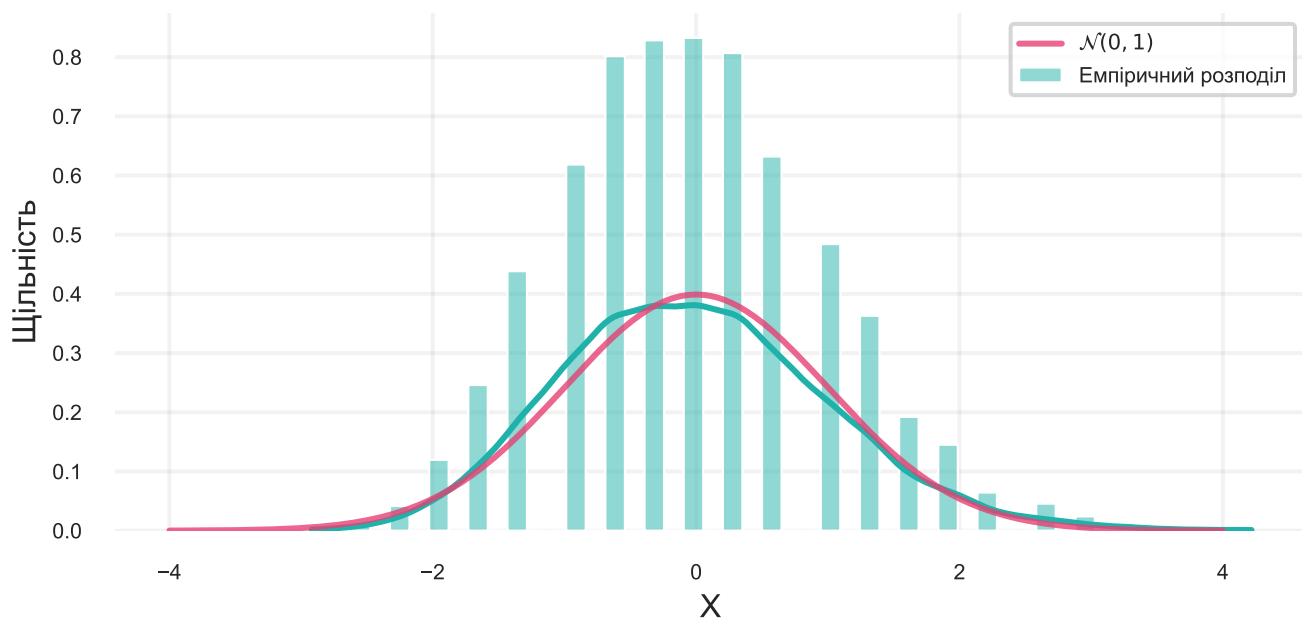
- ① Кількість вибірок.
- ② Пустий масив для зберігання статистик.
- ③ Генерація вибірки з M елементами.
- ④ Кількість елементів у вибірці.
- ⑤ Обчислення статистики.
- ⑥ Додавання статистики до масиву.
- ⑦ Візуалізація емпіричного розподілу та теоретичного розподілу.
- ⑧ Генерація вибірки з біноміального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

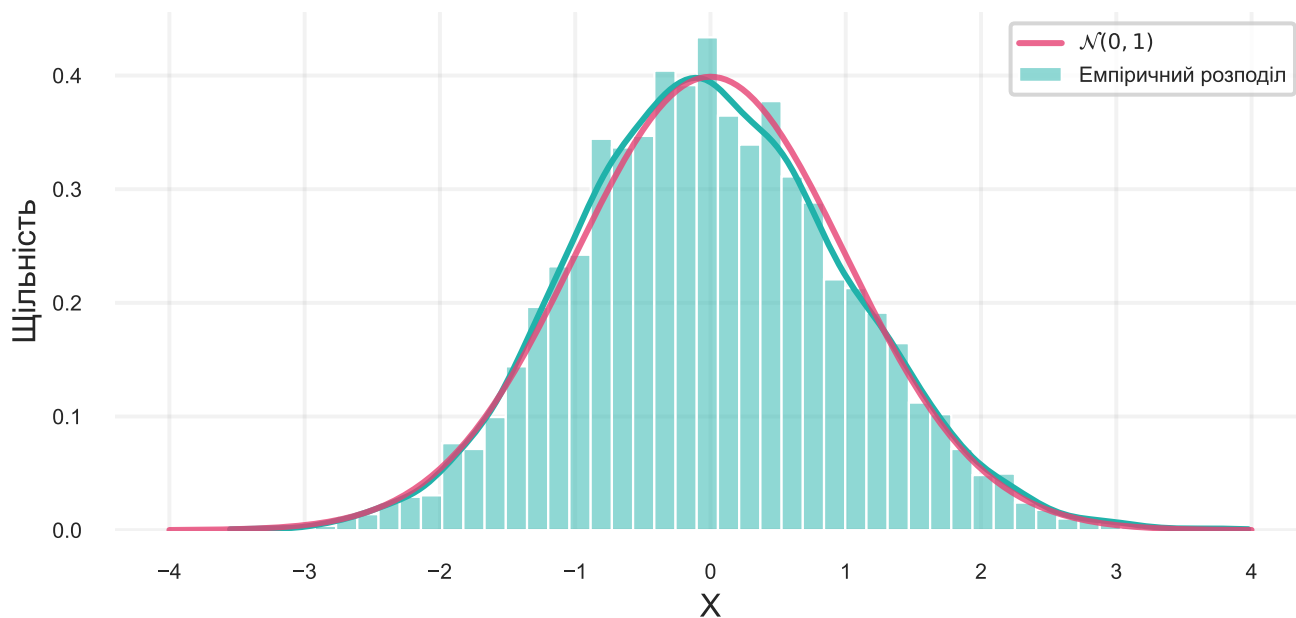
visualize_CLT(lambda: np.random.binomial(n, p, size),
              expected_value = p * n,
              variance = n * p * (1 - p)
)
```



Лістинг 3.7 Візуалізація ЦГТ при великій вибірці з експоненціального розподілу.

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

- 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  
    sigma = np.sqrt(n * mu_0 * (1 - mu_0))  
    return 1 - norm(loc=mu, scale=sigma).cdf(t)  
  
n = 30  
mu_0 = 0.5  
t = 19  
  
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")
```

- 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='two-sided'):
    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
```

- 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
population_variance = mu_0 * (1 - mu_0)

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'):.4f}")
```

- 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})')

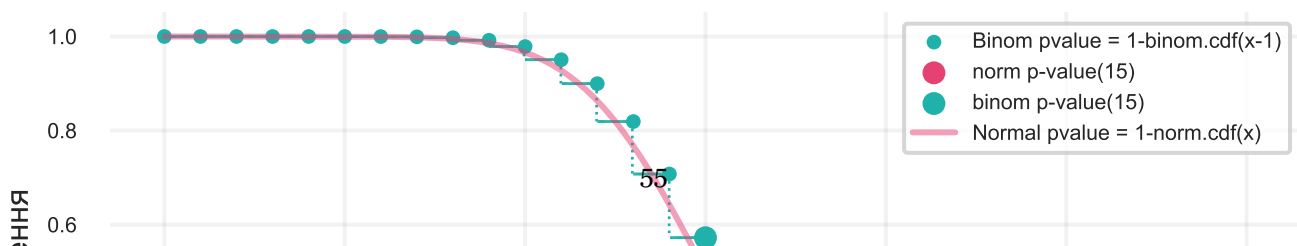
    plt.plot(x_axis, 1 - norm_dist.cdf(x_axis + add_to_x), color=red_pink, alpha=0.5, ⑦
            label=f'Normal pvalue = 1-norm.cdf(x{add_str})')

    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)
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}")
```

- 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

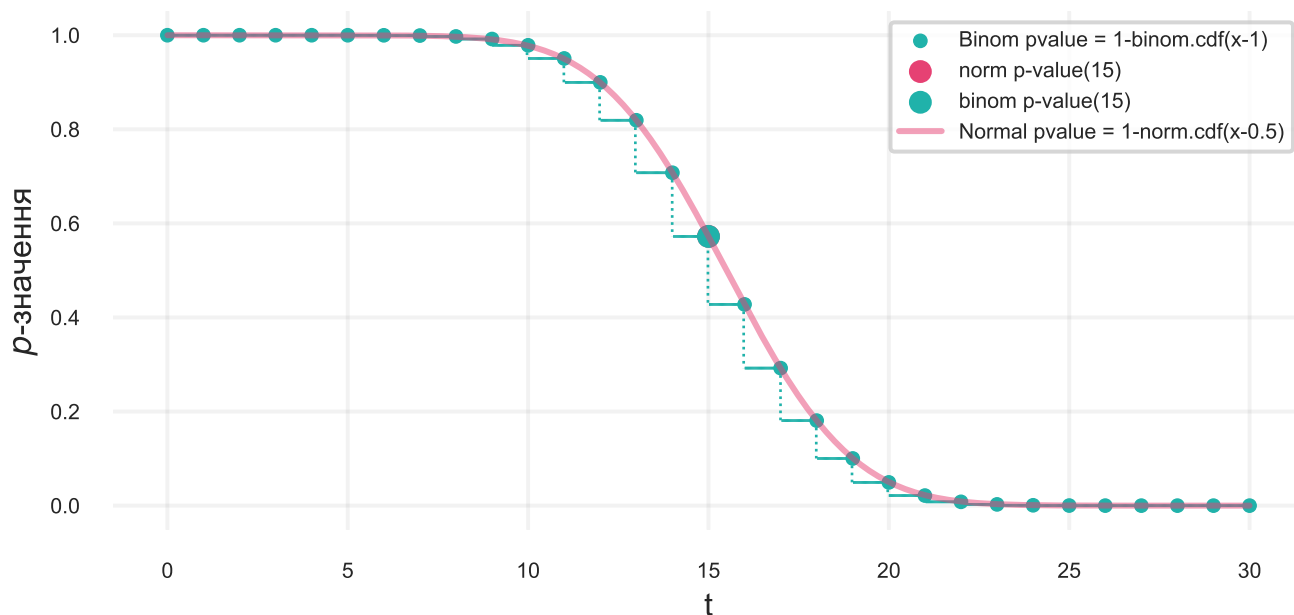
binom_pvalue = 1 - binom(n, mu_0).cdf(t - 1)
norm_pvalue = 1 - norm(loc=n * mu_0, scale=np.sqrt(n * mu_0 * (1 - mu_0))).cdf(t)
diff = binom_pvalue - norm_pvalue

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:.4f}")
```

- 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='greater', continuity_correction=True): ①
    if continuity_correction:
        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=True)
print(f"p-значення за Z-критерієм Фішера з поправкою на неперервність = {p_value:.4f}") ⑭
```

- ⑭ Перевірка наявності поправки на неперервність.

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

Chapter 4

t -критерій Стьюдента

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, \dots, 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:

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

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 $Z(X) = \sqrt{n} \frac{\bar{X} - \mu_0}{\sqrt{\sigma^2}}$
- За досить великого розміру вибірки $Z(X) \stackrel{H_0}{\sim} \mathcal{N}(0, 1)$ (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: $\{Z(X) \geq z_{1-\frac{\alpha}{2}}\} \cup \{Z(X) \leq -z_{1-\frac{\alpha}{2}}\}$
 - p -значення = $2 \cdot \min[\Phi(z), 1 - \Phi(z)]$, де z — реалізація статистики $Z(X)$

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

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:

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

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:

- $T(X) := \sqrt{n} \frac{\bar{X} - \mu_0}{\sqrt{S^2}}$
- $T(X) \stackrel{H_0}{\sim} \mathcal{N}(0, 1)$

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

Для цього пропонується подивитися, як насправді буде розподілена статистика $T(X) = \sqrt{n} \frac{\bar{X} - \mu_0}{\sqrt{S^2}}$ 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 подивимось, 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()
```

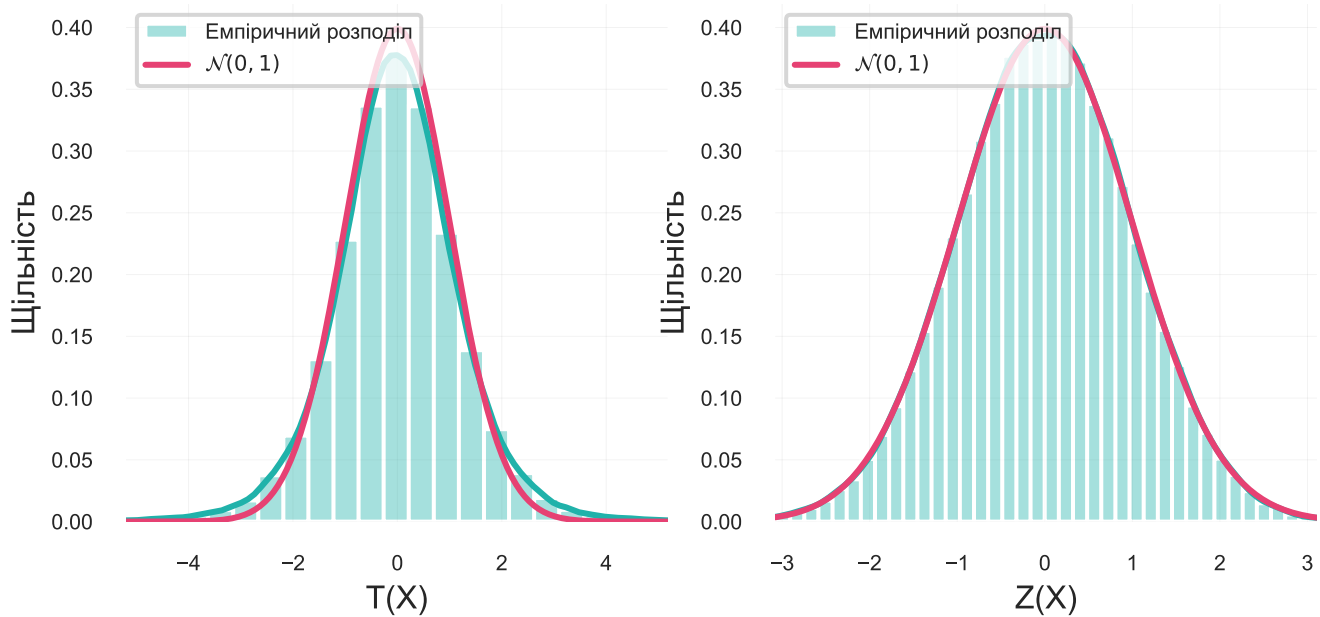


Рисунок 4.1: Симуляція розподілу $T(X)$ та $Z(X)$

Ми бачимо, що:

- Z -тест тут працює: $\sqrt{n} \frac{\bar{X} - \mu_0}{\sqrt{\sigma^2}} \sim \mathcal{N}(0, 1)$.
- 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 $T(X) = \sqrt{n} \frac{\bar{X} - \mu}{\sqrt{S^2}}$ не розподілена нормально, хоча $\sqrt{n} \frac{\bar{X} - \mu}{\sqrt{\sigma^2}} \stackrel{H_0}{\sim} \mathcal{N}(0, 1)$? 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 $\bar{X} \sim \mathcal{N}(\mu, \sigma^2)$. 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 $\sqrt{n} \frac{\bar{X} - \mu_0}{\sqrt{\sigma^2}} \sim \mathcal{N}(0, 1)$.

Але ми нічого не знаємо про $\frac{\bar{X}}{\sqrt{\eta}}$, де $\bar{X} \sim \mathcal{N}$, $S^2 := \eta \sim P$, 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()

```

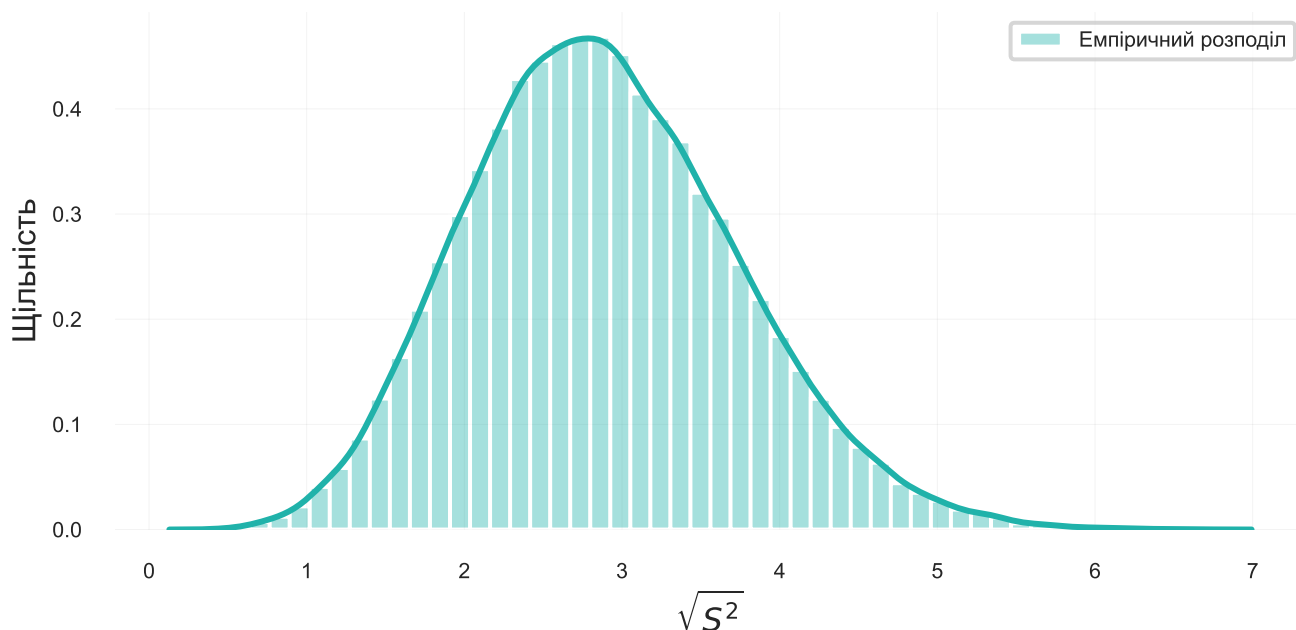


Рисунок 4.2: Розподіл $\sqrt{S^2}$

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 введемо критерій, який допоможе розв'язати початкову задачу!

4.1 T -тест Стюдента

Для того щоб вивести T -тест, нам потрібно зрозуміти, як розподіляється статистика $T(X) = \sqrt{n} \frac{\bar{X} - \mu_0}{\sqrt{S^2}}$. 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.

¹Розподіл 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 $Y = X_1^2 + X_2^2 + \dots + X_k^2 \sim \chi_k^2$.

- Тоді $\sum_{i=1}^n (\xi_i - \bar{\xi})^2 \sim \chi_{n-1}^2$ ².

- $S_X^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$

- $\xi_i := \frac{X_i - \mu}{\sigma} \sim \mathcal{N}(0, 1)$. Тоді

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

- А значить $\frac{(n-1) \cdot S_X^2}{\sigma^2} = \sum_{i=1}^n (\xi_i - \bar{\xi})^2 \sim \chi_{n-1}^2$

3. Нехай $\xi \sim \mathcal{N}(0, 1)$, $\eta \sim \chi_k^2$ і ξ з η незалежні. Тоді статистика $\zeta = \frac{\xi}{\sqrt{\eta/k}} \sim t_k$ — з розподілу Стюдента³ з k ступенями свободи.

- $\xi := \sqrt{n} \frac{\bar{X} - \mu_0}{\sigma} \sim \mathcal{N}(0, 1)$

- $\eta := \frac{(n-1) \cdot S_X^2}{\sigma^2} \sim \chi_{n-1}^2$

- ξ и η незалежні⁴.

- Тоді

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

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

Тепер нам достатньо даних, щоб побудувати T -тест:

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

Статистика $T(X)$ буде виглядати так:

$$T(X) = \sqrt{n} \frac{\bar{X} - \mu_0}{\sqrt{S^2}} \sim t_{n-1} \quad (4.3)$$

Тоді односторонній критерій набуває вигляду:

$$\{T(X) \geq t_{n-1, 1-\alpha}\} \quad (4.4)$$

А p -значення для одностороннього критерію можна обчислити так:

²Доведення Cochran (1934).

³Розподіл Стюдента — це розподіл, який виникає при нормальному розподілі з невідомою дисперсією. Якщо $X_1, X_2, \dots, X_n \sim \mathcal{N}(\mu, \sigma^2)$, то $T = \frac{\bar{X} - \mu}{S/\sqrt{n}} \sim t_{n-1}$, де \bar{X} — вибіркве середнє, S — вибірква стандартна девіація.

⁴Доведення Basu (1955).

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

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

Двосторонній критерій буде виглядати так:

$$\{T(X) \geq t_{n-1, 1-\frac{\alpha}{2}}\} \cup \{T(X) \leq -t_{n-1, 1-\frac{\alpha}{2}}\} \quad (4.6)$$

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

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

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

4.2 T -тест у Python

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

- `scipy.stats.chi2` — 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()
```

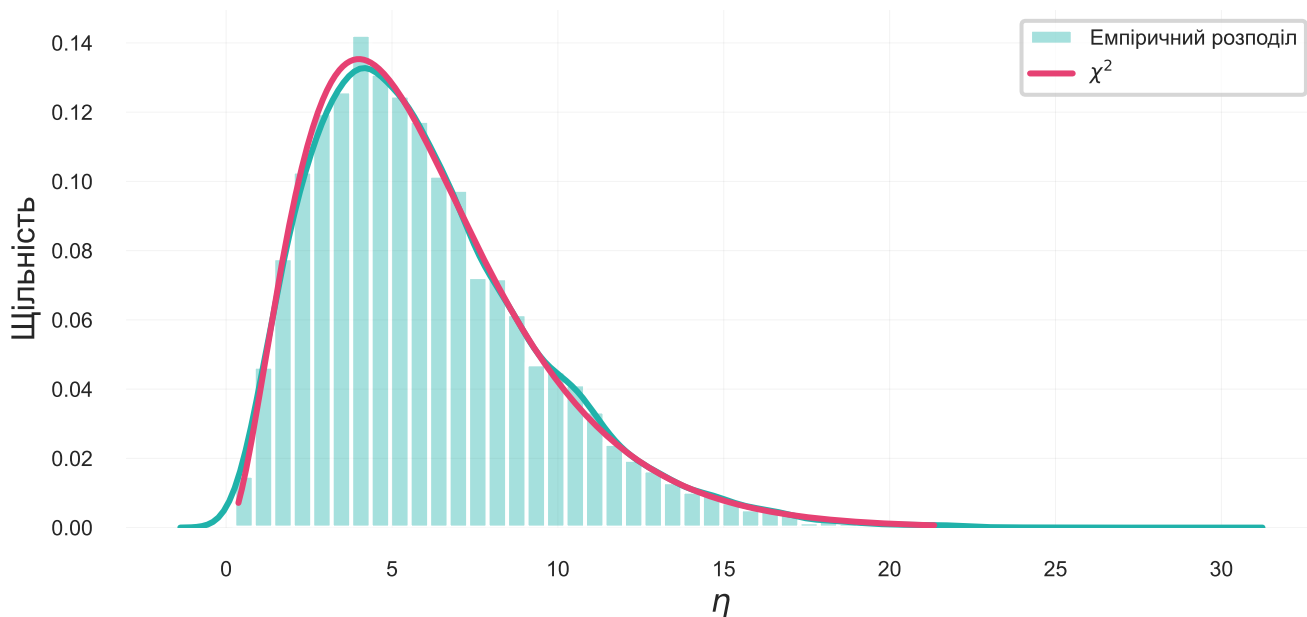


Рисунок 4.3: Розподіл емпіричного η та теоретичного χ^2 .

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 Стюдента. 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(sample)
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()
```

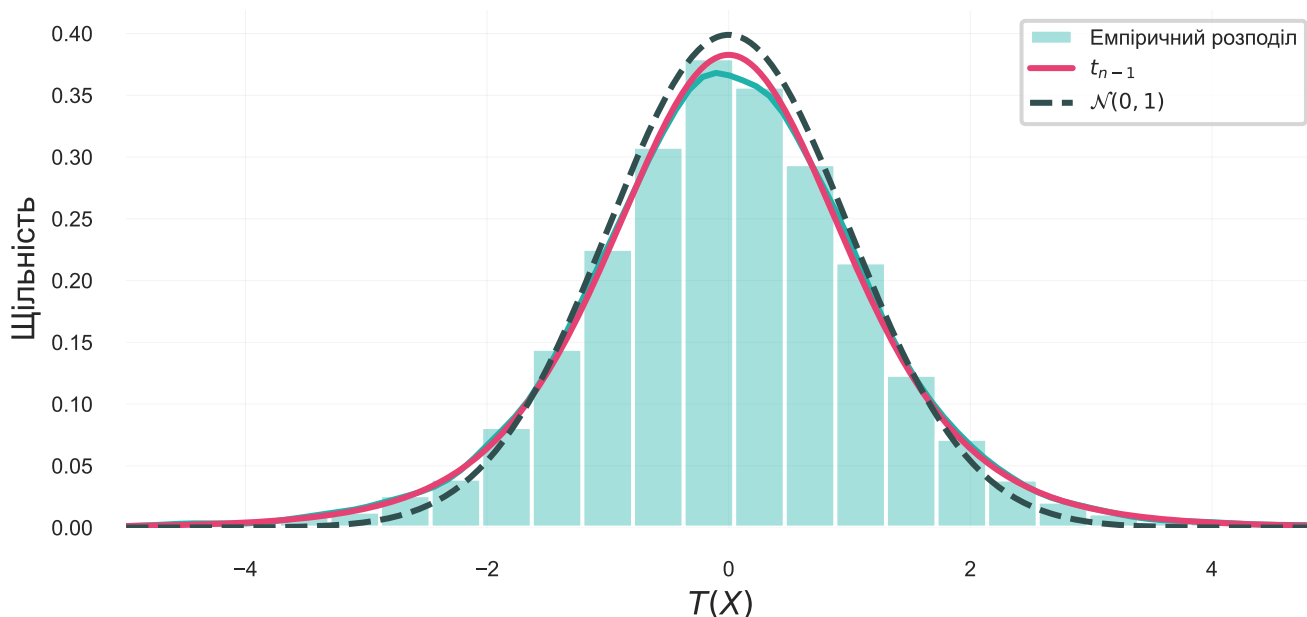


Рисунок 4.4: Розподіл $T(X)$

Розподіл Стюдента 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.

4.3 Довірчі інтервали

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

4.3.1 Перший метод

Нехай Q — статистика 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 $H_0 : \sqrt{n} \frac{\bar{X} - m}{\sqrt{S^2}} \sim t_{n-1}$.

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

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

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

$$\text{де } S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$$

4.3.2 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 $P(L(X) < \mu < R(X)) = 1 - \alpha$.

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

Тоді

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

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

4.4 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]

4.5 T -тест та вимога нормальності

Chapter 5

Монте-Карло в задачах статистики

Підсумки

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.
- Lemons, Don S. 2002. *An Introduction to Stochastic Processes in Physics*. The Johns Hopkins University Press.