Прикладний статистичний аналіз

Ігор Мірошниченко Юлія Хлевна

2025-04-04

Зміст

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 змоделюємо можливі результати для 30 студентів.

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

- Якщо монетка випаде орлом, студент оплачує доступ.
- Якщо монетка випаде решкою, студент відмовляється від курсу.
- Використаємо метод `integers()` 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.

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

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[2] 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 Бернуллі[3]. Параметр цього розподілу, а саме ймовірність успіху, нам невідомий.

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

де μ — ймовірність успіху.

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

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

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

Скорочено це записують як:

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

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

Тоді питання можна переформулювати так:

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

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

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

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

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

Лістинг 1.2 Функція щільності ймовірностей для біноміального розподілу

```
n = 30                                                                    ①
mu = 0.5                                                                  ②

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

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

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

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

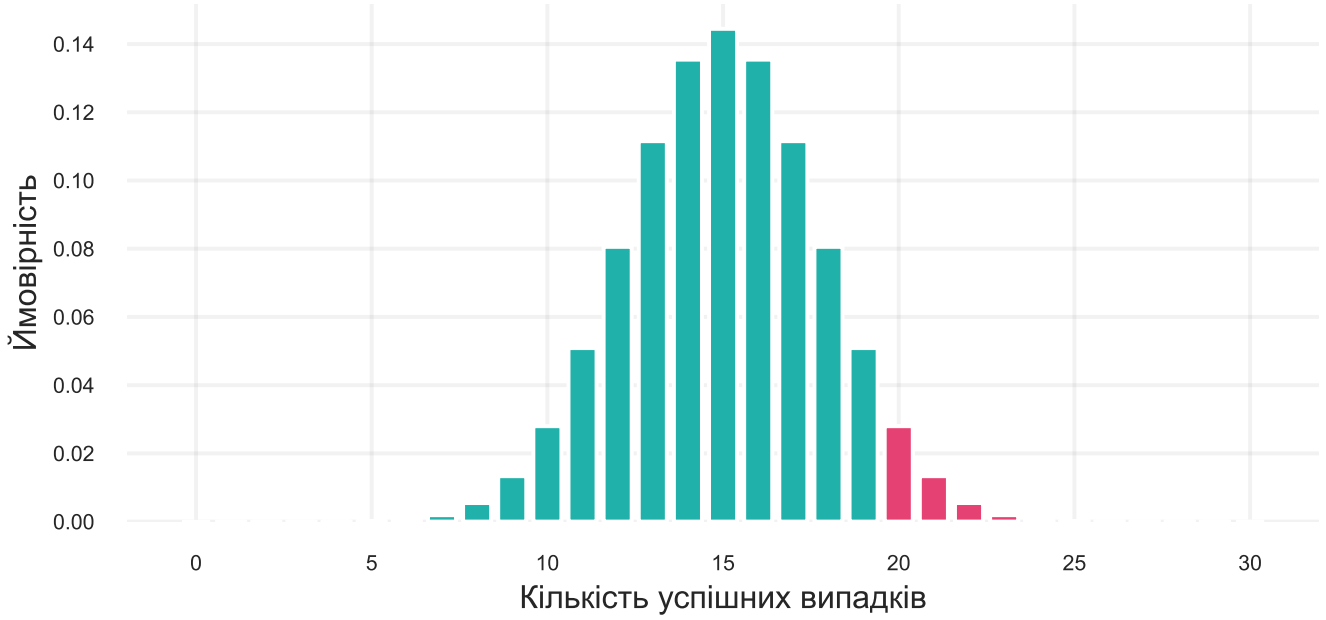


Рисунок 1.1: Візуалізація функції щільності ймовірностей для біноміального розподілу

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

1.2.2 Критерій

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

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

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

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

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

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

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

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

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

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.

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

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

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

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

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

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

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

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

Обчислимо FPR для $C = 21$

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

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

1.2.4 Обчислення FPR

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

```
x = np.arange(0, n + 1)
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()
```

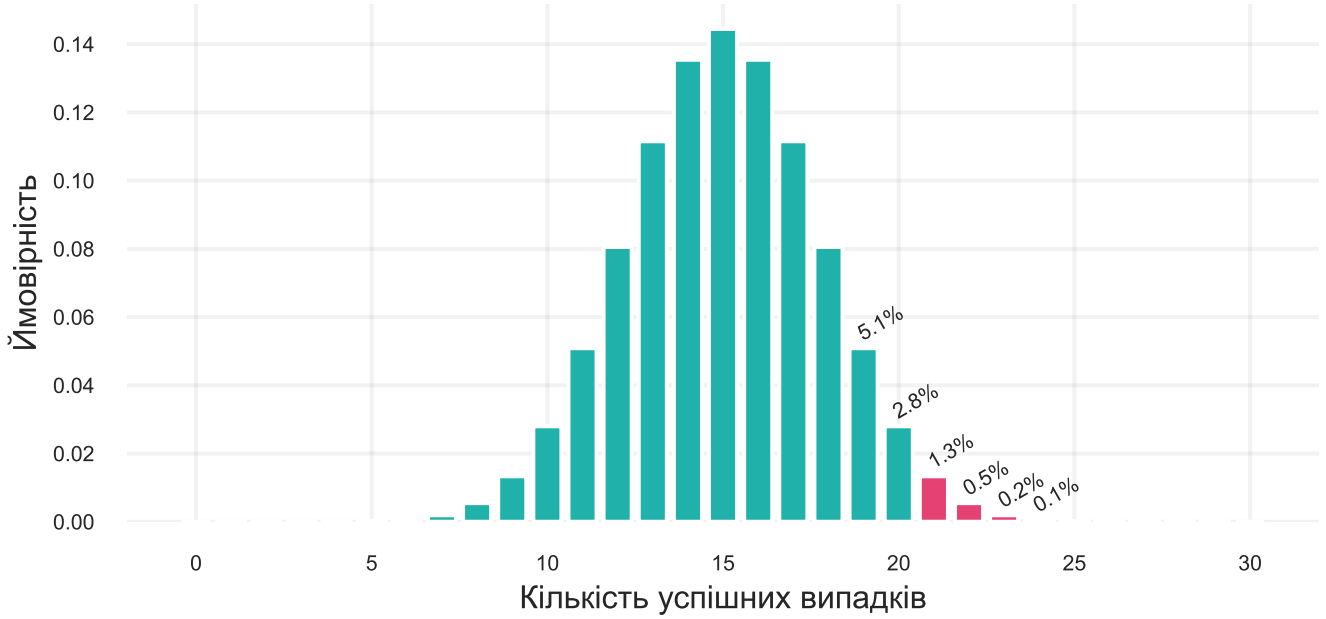


Рисунок 1.2: Ймовірність хибно відхилити H_0 за умови її вірності

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 α = 100% створити тривіально — достатньо завжди відхиляти H_0 — тому така постановка не має сенсу.

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

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

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

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

Якщо ж обрати 19, то FPR буде більше α:

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

1.3 Статистичні функції в Python

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

1.3.1 Біноміальний розподіл

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

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

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

Лістинг 1.3 Створення біноміального розподілу

```
from scipy.stats import binom

n = 30                                                                    ①
mu = 0.5                                                                  ②

binom_dist = binom(n, mu)
```

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.

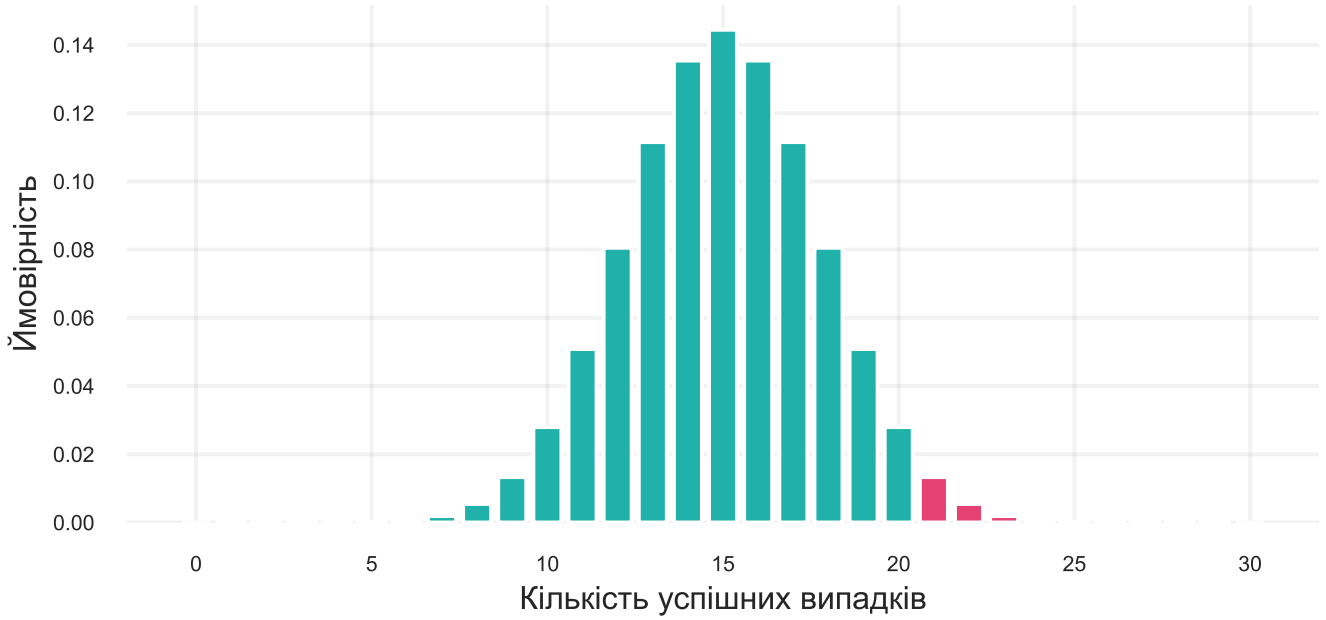


Рисунок 1.3: Функція щільності ймовірностей біноміального розподілу

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 $F_\xi(x) = P(\xi \leqslant x)$

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 $\geqslant 0.95$. А оскільки $P(\xi \leqslant 19) + P(\xi \geqslant 20) = 1$, 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 *квантилью*.

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

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

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

Лістинг 1.9 Обчислення рівня значущості критерію

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

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

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

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

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

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

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

У Python квантиль можна порахувати через 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 $P(Q \geqslant C) \leqslant \alpha$
2. Тобто потрібно $P(Q < C) \geqslant 1 - \alpha$
3. Q приймає тільки цілі значення: $P(Q \leqslant C - 1) \geqslant 1 - \alpha$, або $F(C - 1) \geqslant 1 - \alpha$
4. Отже, з визначення квантилі, $C - 1 = u_{1-\alpha}$
5. Значить $C = u_{1-\alpha} + 1$

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

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

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

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

1.4 p-значення

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

1.4.1 Більш екстремальні значення

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

Визначимо поняття екстремальності формально:

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

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

1.4.2 p-значення

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

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

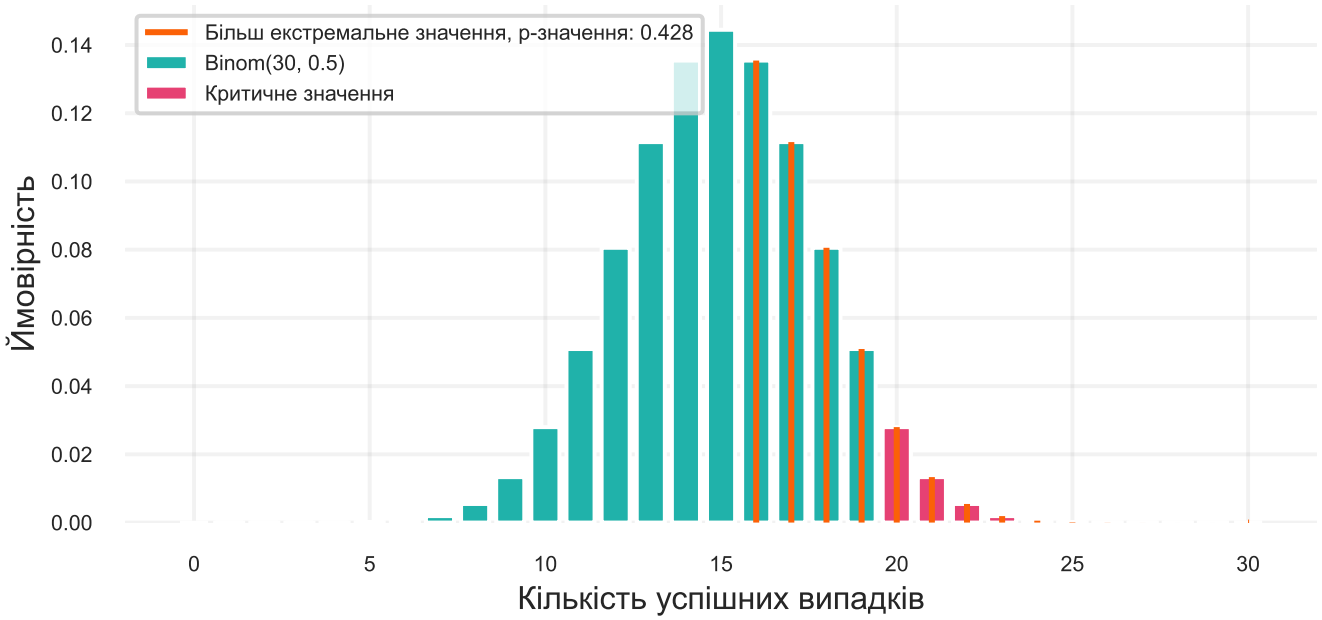


Рисунок 1.4: p-значення для критерію $Q = 15$

Тепер виведемо формулу через функції Python:

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

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

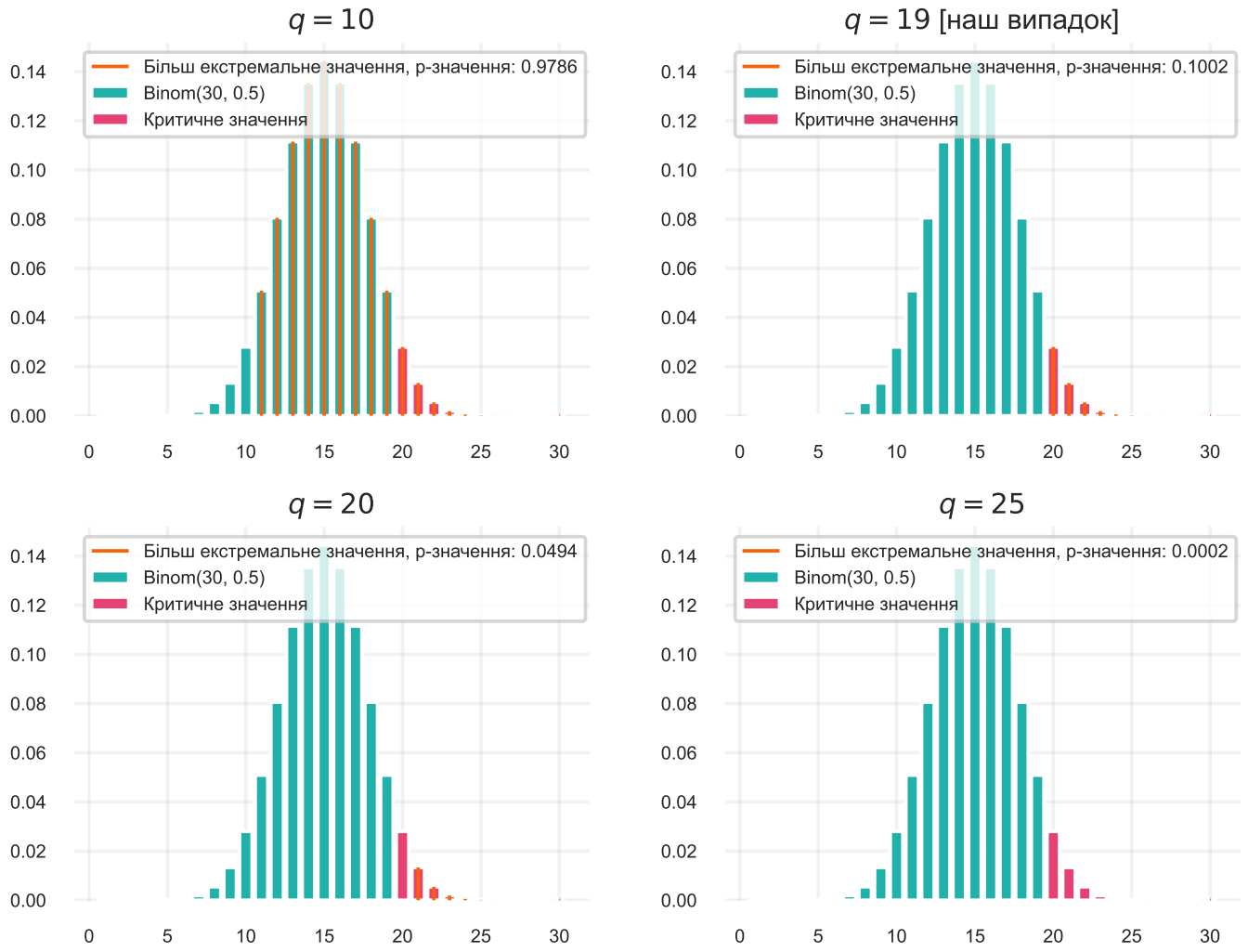


Рисунок 1.5: p-значення для критерію $Q = 10, 19, 20, 25$

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

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

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:

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

1.5.1 Як вибрати критичну область

Подивимося, який вигляд матиме критична область у такому разі.

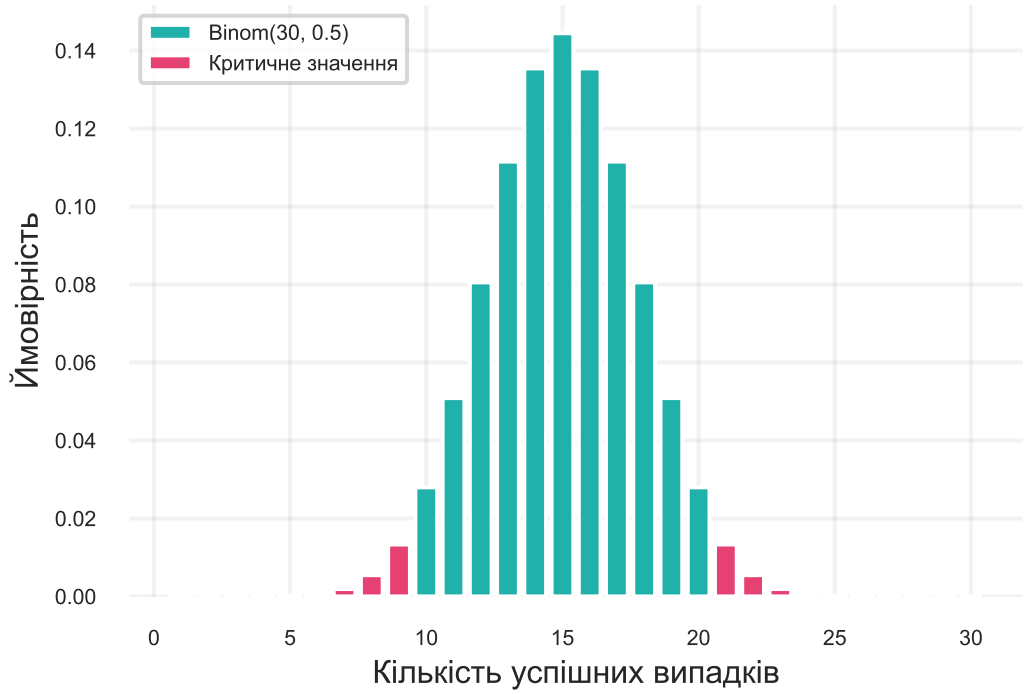


Рисунок 1.6: Двостороння критична область для критерію $= 6$

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

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

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

Реалізуємо функцію на Python.

```
6.0
```

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

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

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

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

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()
```

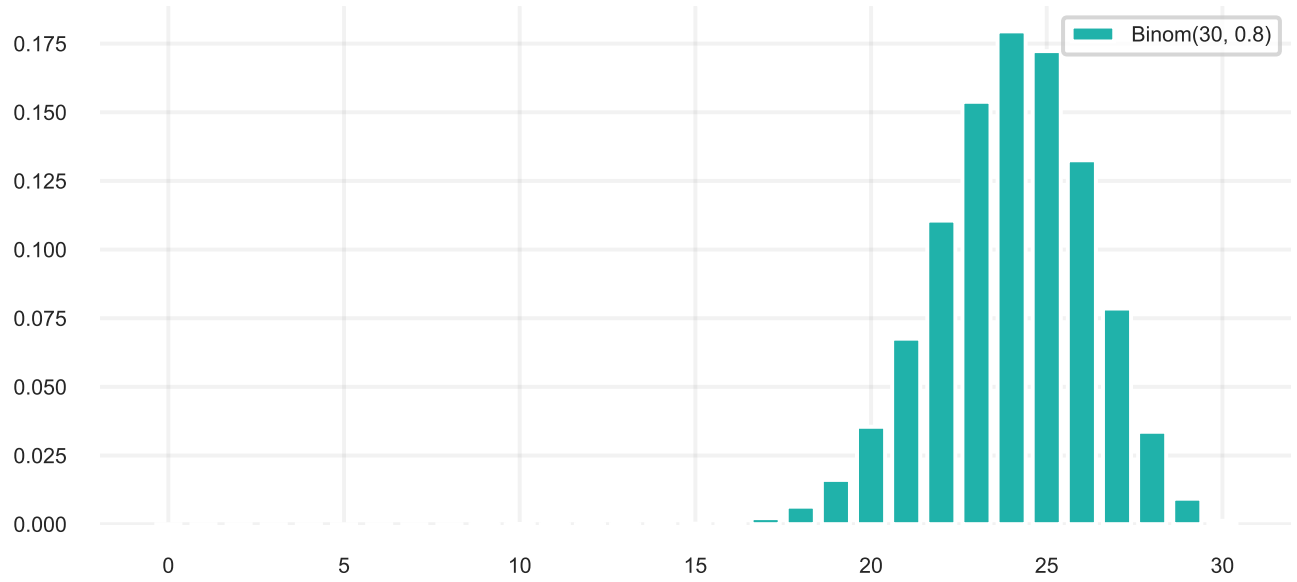


Рисунок 1.7: Біноміальний розподіл з імовірністю успіху 0.8

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

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

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

```
(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)
```

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

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

Зобразимо критичну область на графіку.

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

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()
```

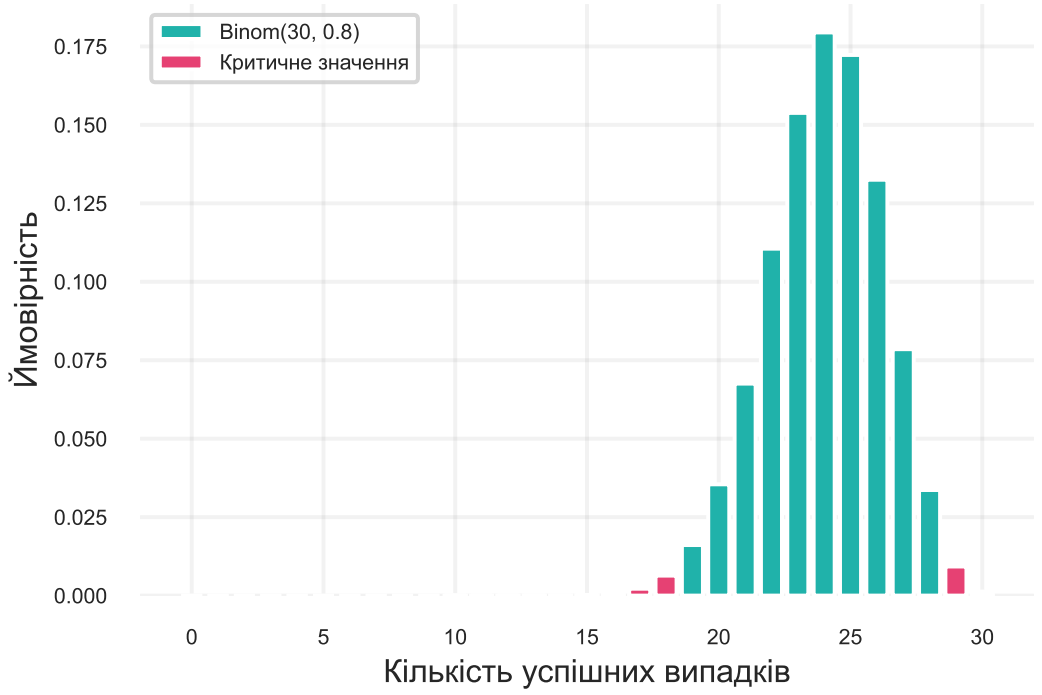


Рисунок 1.8: Двостороння критична область для критерію $C_1 = 18, C_2 = 29$

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

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

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.

$$H_0: \text{ на вулиці літо}$$

$$H_1: \text{ на вулиці не літо}$$

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

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

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

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).

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

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

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

Перший критерій

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

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

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

Другий критерій

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

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

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

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

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

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

2.2 Потужність для біноміального розподілу

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

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

Згадаймо критерій, за яким ми приймаємо рішення:

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

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

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

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

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

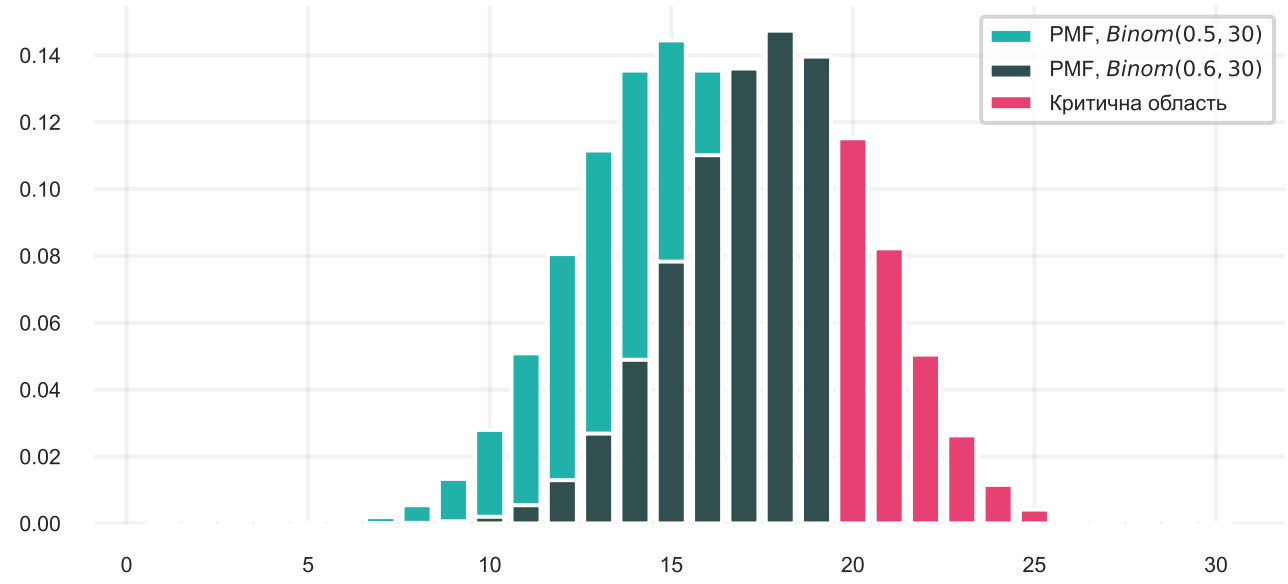


Рисунок 2.1: Потужність критерію для $\mu = 0.6$

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.

Подивимося, 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('Потжність')

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()
```

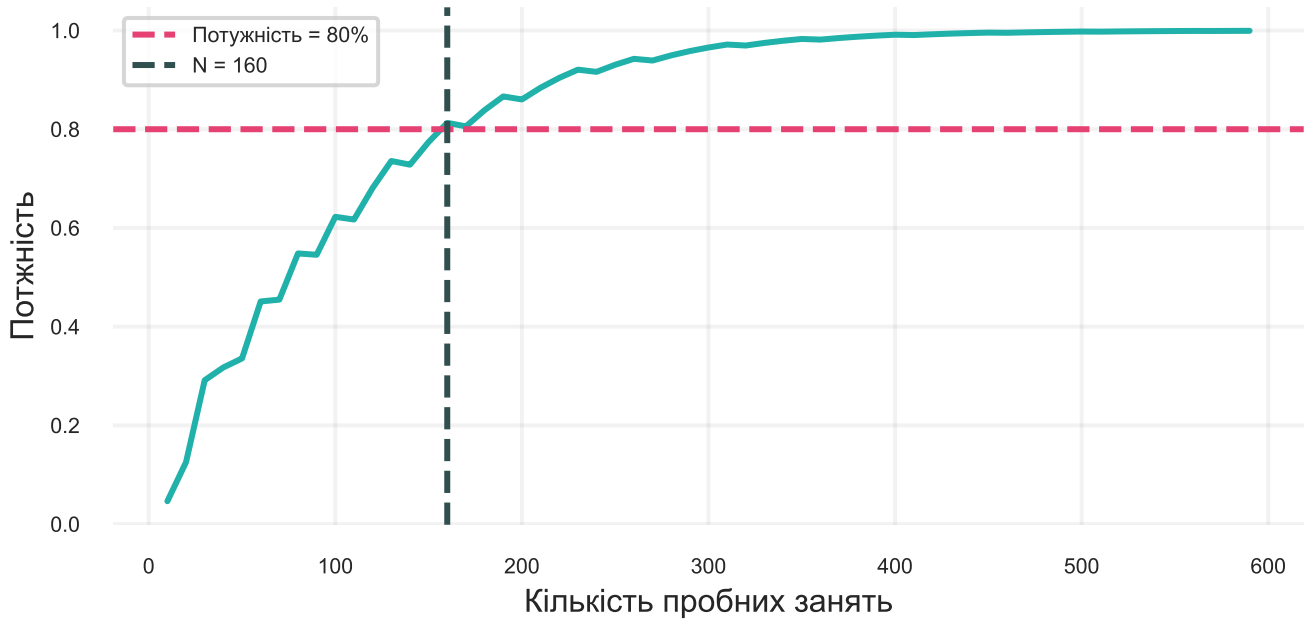


Рисунок 2.2: Залежність потужності від розміру вибірки для $\mu = 0.6$

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.

```python
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('Потжність', 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()
```

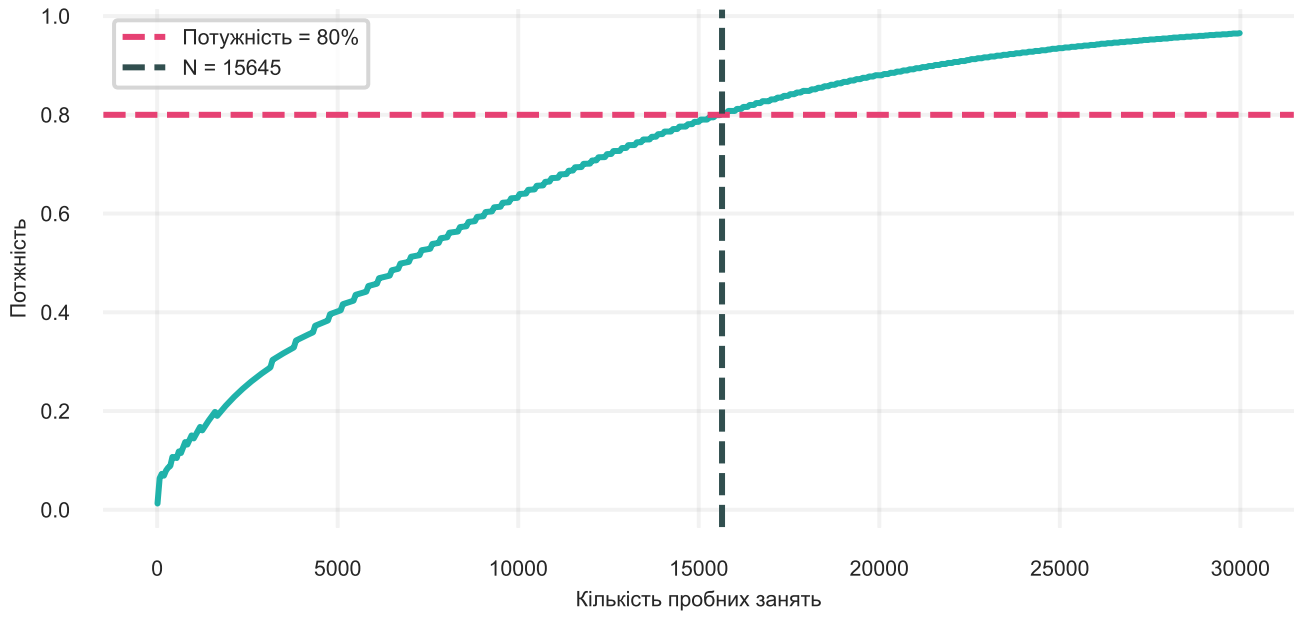


Рисунок 2.3: Залежність потужності від розміру вибірки для $\mu = 0.51$

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.

```python
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()
```

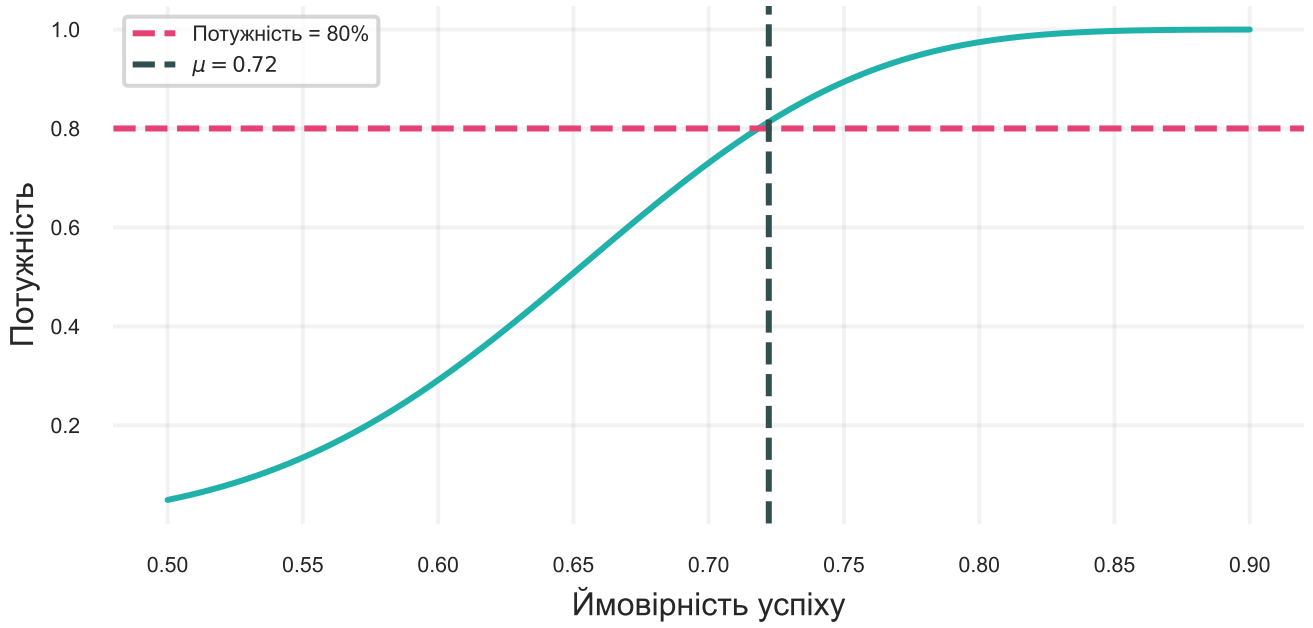


Рисунок 2.4: Залежність потужності від параметра μ

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 зворотне завдання, За необхідним MDE визначити обсяг вибірки. Якщо ми говоримо, що хочемо детектувати +10 в.п., тобто 60% успішних підписок, то потрібно знайти 160 тестових клієнтів, це видно з попередніх графіків. Якщо 30 осіб нам, наприклад, шукати місяць, такий тест може затягнутися майже на півроку. Тому варто подумати про те, щоб виділити додаткові ресурси на пошук клієнтів, наприклад, залучити маркетологів.

2.4 Довірчі інтервали

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

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

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

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

```
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.

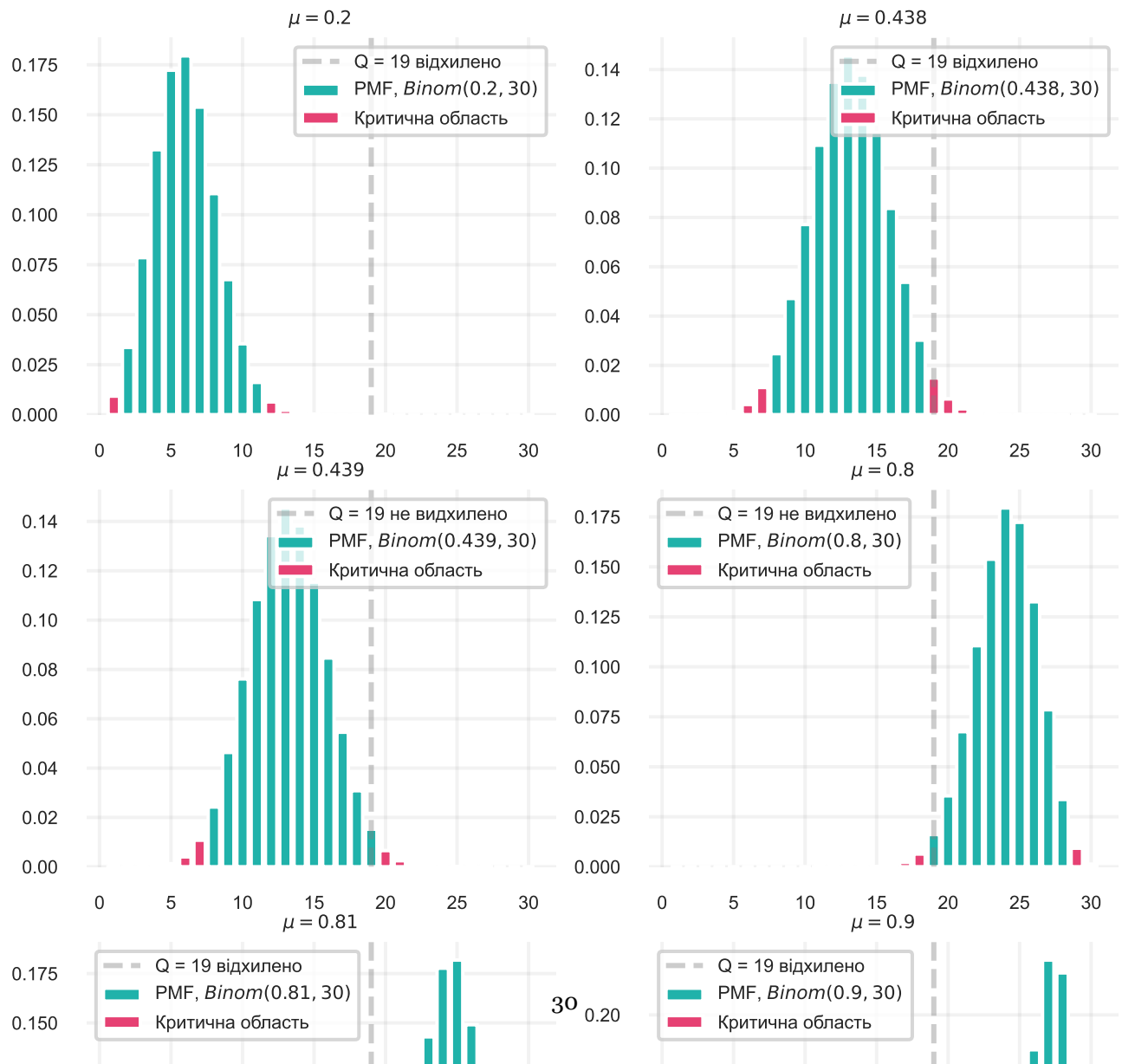
Отримавши такий інтервал, ми відразу можемо зробити висновок, що гіпотеза $H_0 : \mu = 0.5$ не відкидається, оскільки 0.5 лежить у довірчому інтервалі. Але при цьому відразу зрозуміло, що $\mu \neq 0.4$ на рівні значущості α.

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 $\frac{19}{30}$.

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

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_rejectio

    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.

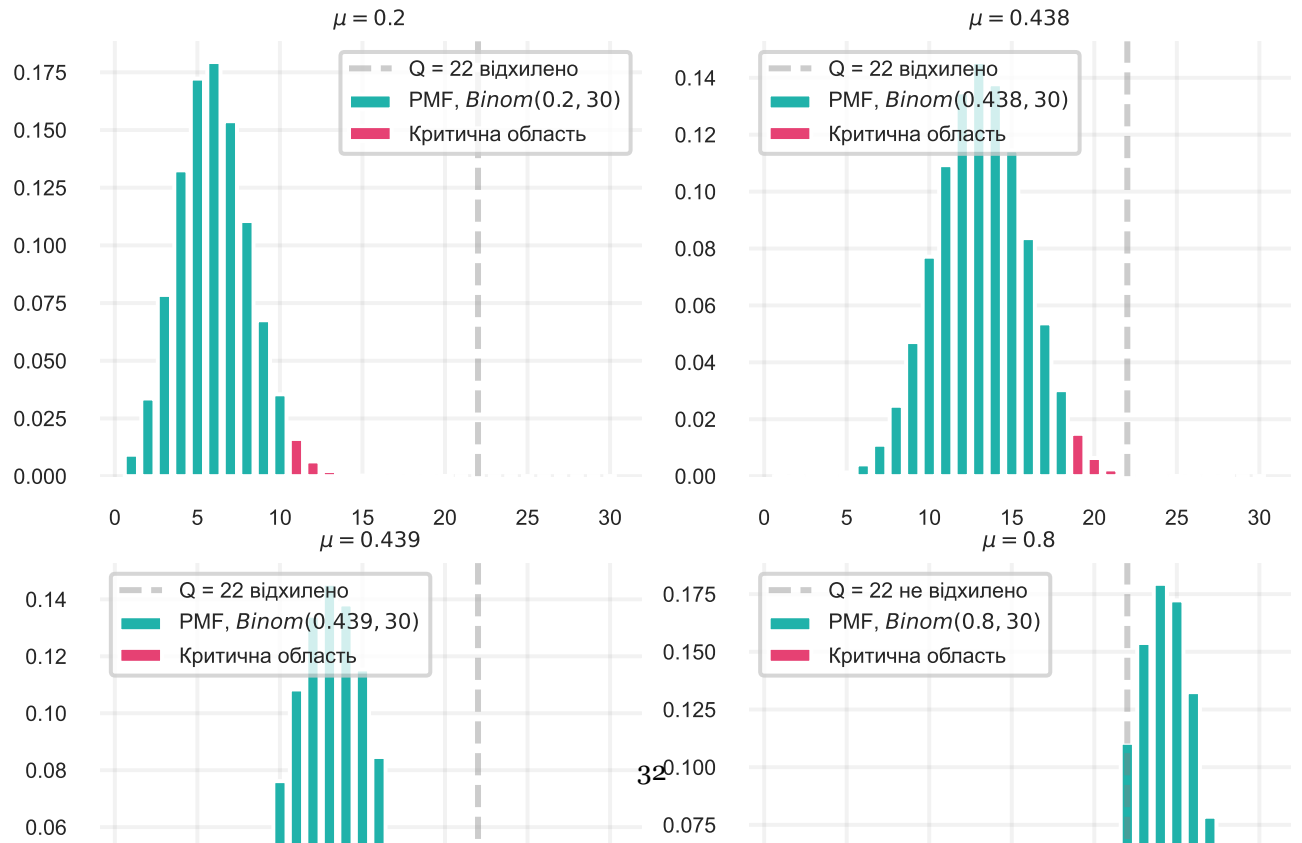
Навіщо ж ми тоді взагалі використовуємо двосторонній інтервал? Щоб це зрозуміти, подивимося, як виглядають візуально межі для одностороннього інтервалу.

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
```



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_rejectio
```

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_rejectio

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

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.

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

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

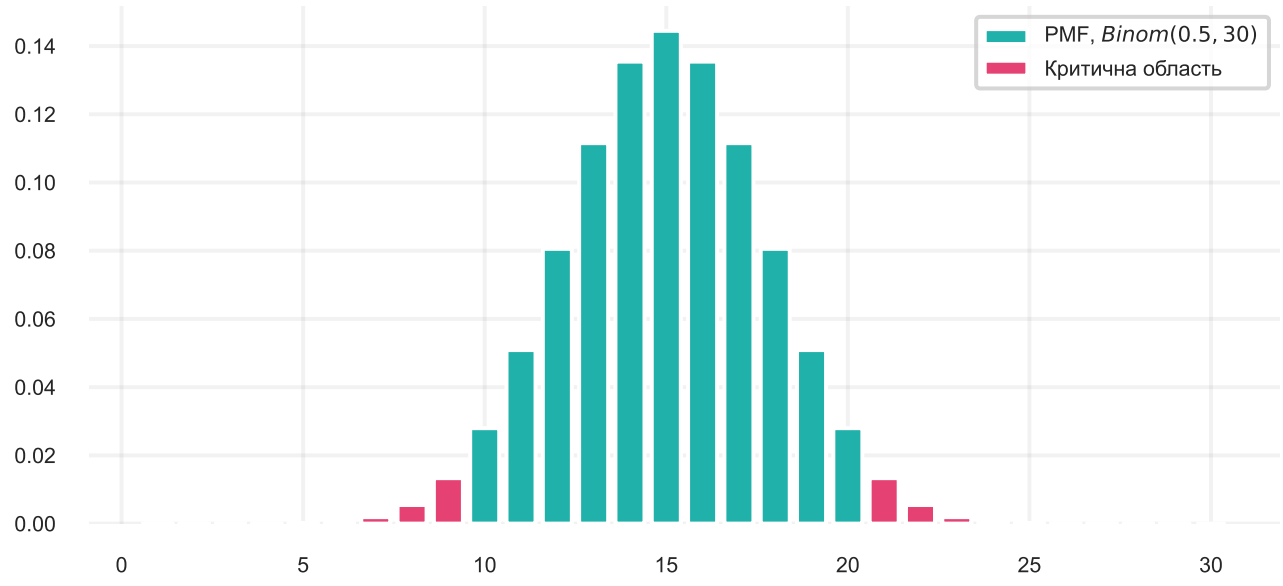


Рисунок 2.7: Розподіл статистики при істинній ймовірності успіху

Нехай реалізувалося значення статистики q. За такою вибіркою можна побудувати довірчий інтервал на μ. Він буде якось розташований, але зараз нас цікавить, чи потрапить у нього μ_0. За визначенням потрапляння в інтервал відбудеться, якщо не відкидається гіпотеза $H_0 : \mu = \mu_0$. Але тоді за справедливості H_0 статистика має той розподіл, що і на малюнку. І гіпотеза відкидається тільки в разі потрапляння в критичну область, а це трапляється з ймовірністю $\leqslant \alpha$.

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

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

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

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

2.6.2 Довірчий інтервал Вілсона

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

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

```
Відсоток успішних випадків: 96.6%
Час виконання: 0.0736 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.

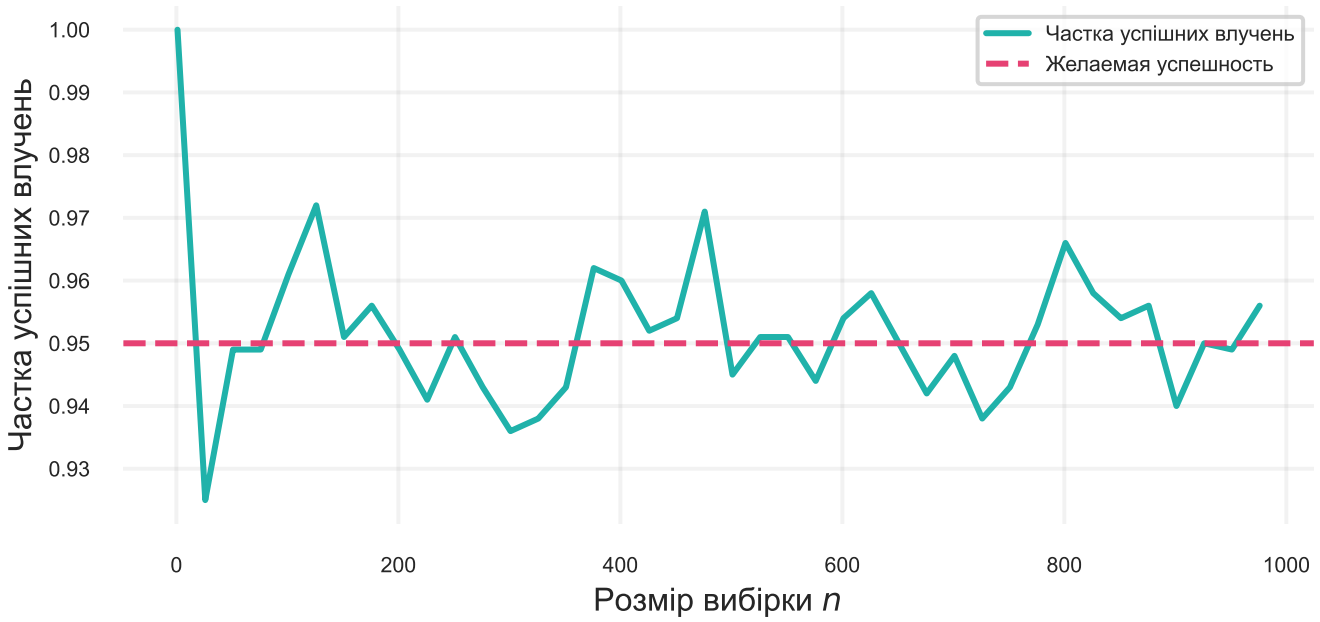


Рисунок 2.8: Залежність частки успішних влучень μ у довірчий інтервал від розміру вибірки

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: 3.7781 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, $Binom({mu0}, 30)$')
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).

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

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

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

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

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

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

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

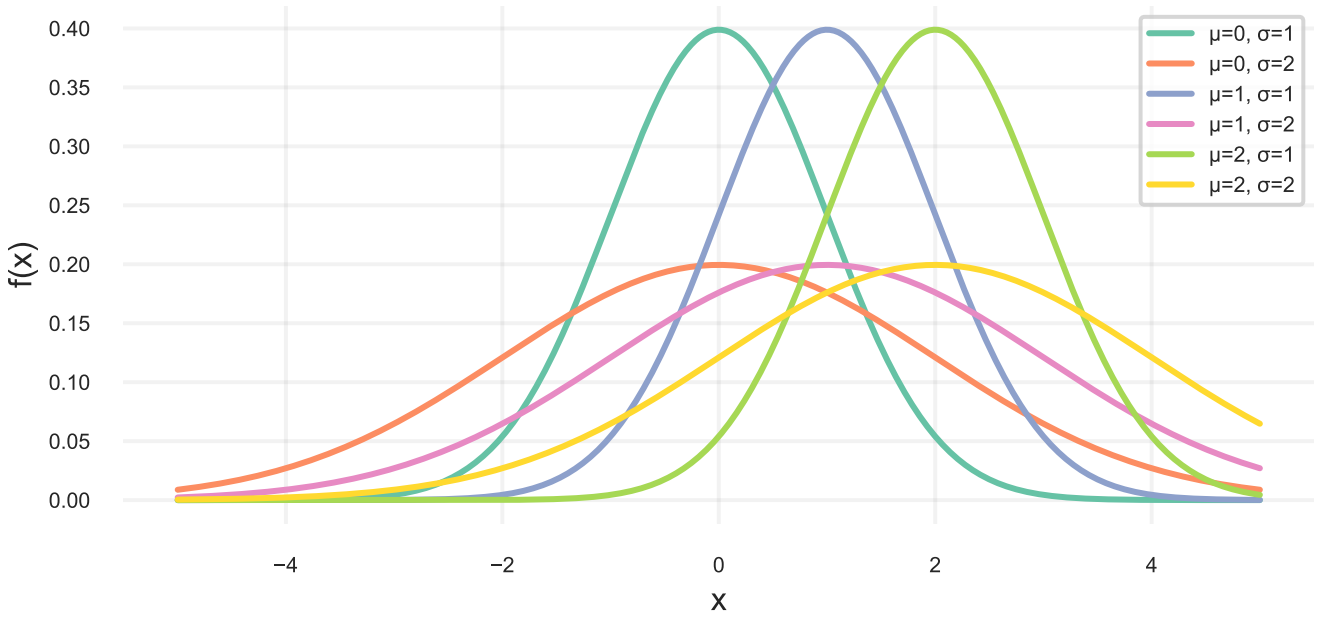


Рисунок 3.1: Нормальний розподіл з різними параметрами

3.2 Нормальний розподіл у Python

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

Параметри класу:

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

Методи класу:

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

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

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

Візуалізація методів класу `norm` показана на рисунку 3.2.

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

Лістинг 3.2 Нормальний розподіл у Python.

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

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

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

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

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

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

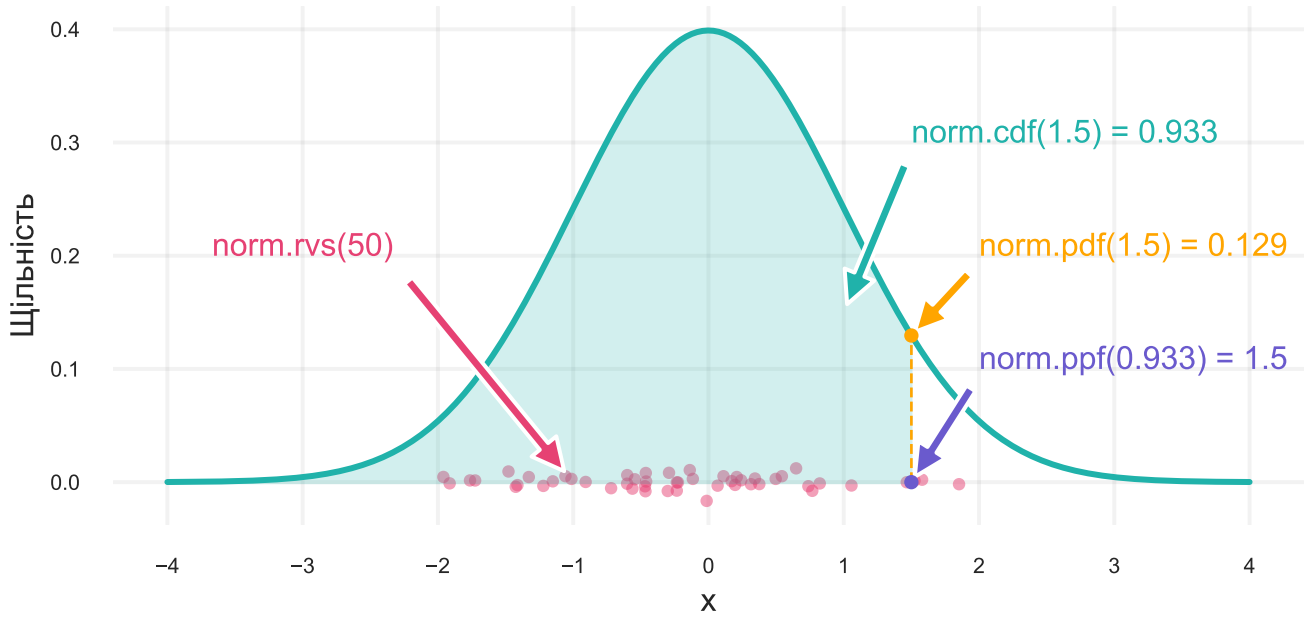


Рисунок 3.2: Демонстрація методів класу `norm`

3.3 Властивості нормального розподілу

Нормальний розподіл має кілька важливих властивостей:

1. **Сума двох незалежних нормально розподілених випадкових величин також має нормальний розподіл**[2]:

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

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.

[2] Доведення цієї властивості можна знайти в роботі Lemons (2002).

Chapter 4

t-критерій Стьюдента

Chapter 5

Монте-Карло в задачах статистики

Підсумки

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