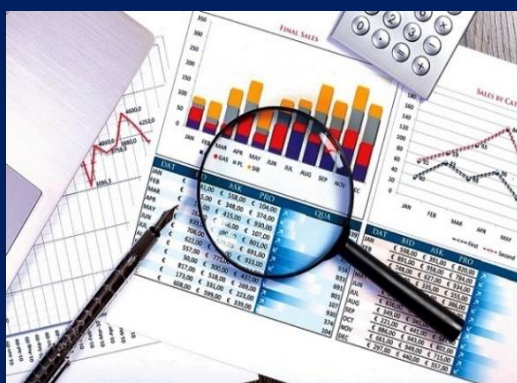


ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC SƯ PHẠM

NGUYỄN THẾ DŨNG

THỐNG KÊ ỨNG DỤNG VỚI PYTHON



NHÀ XUẤT BẢN N.T.D

Lời mở đầu

Ngày nay, khoa học dữ liệu đã và đang phát triển mạnh mẽ, có thể nói Thống kê là một trong những nền tảng cơ bản của khoa học dữ liệu. Tài liệu ***Thống kê ứng dụng với Python*** này nhằm hỗ trợ cho sinh viên trong việc học môn học Thống kê ứng dụng và môn học Cơ sở lập trình.

Khi học và làm việc với thống kê, chúng ta có những phần mềm hỗ trợ thống kê khá tốt như SPSS, SAS, STATA.... Với các phần mềm chuyên dụng, chúng ta có một phương tiện thuận lợi cho công việc. Tuy vậy, để có thể sáng tạo, linh hoạt trong công việc và mở rộng cho các công việc khác, chúng ta cần có những phương tiện hữu hiệu hơn đó là các ngôn ngữ lập trình. Trong dạy và học thống kê ngày nay, chúng ta không chỉ dừng lại ở lý thuyết suông mà cần mô phỏng, minh họa để tăng tính ứng dụng, nâng cao năng lực người học. Cùng với đó, Python được xem là ngôn ngữ lập trình phổ dụng và đảm bảo được một số tiêu chí lựa chọn ngôn ngữ cho dạy học nhập môn lập trình ở phổ thông và cho sinh viên, cũng như là ngôn ngữ đã và đang được phát triển cho thống kê và khoa học dữ liệu ([3], [4], [5]).

Trong tài liệu này, các kiến thức về thống kê được trình bày ở mức cơ bản để người đọc có thể hiểu được các minh họa ứng dụng Python mô phỏng các bài toán cơ bản của thống kê. Chúng tôi cũng dẫn nguồn đến các tài liệu tin cậy để người đọc tìm hiểu sâu thêm về thống kê.

Sử dụng tài liệu, người đọc sẽ tiếp cận việc học lập trình qua các ví dụ. Các bài toán cơ bản của thống kê sẽ được minh họa qua các ví dụ cụ thể được trình bày với các bước thực hiện. Sau các ví dụ là các chỉ dẫn để người đọc hiểu về các dòng mã lệnh chính yếu của Python để giải quyết bài toán, qua đó nâng cao khả năng ngôn ngữ lập trình và giải quyết vấn đề. Chúng tôi cũng sẽ dẫn nguồn đến các tài liệu khá đầy đủ để mô tả chi tiết cho các câu lệnh của Python, theo các trang mã nguồn mở gốc của Python như: <https://docs.python.org/>; <https://numpy.org/>; <https://pandas.pydata.org/>; <https://scipy.org/>; <https://matplotlib.org/>; ... để người học có thể hiểu tường tận, cũng như các cải biến qua các version của mã nguồn Python.

Tài liệu bao gồm các chương sau:

- *Giới thiệu về Python*
- *Xử lý dữ liệu cơ bản với Python*

- Sơ lược thống kê ứng dụng
- Lấy mẫu dữ liệu và lập bảng thống kê
- Trực quan hoá dữ liệu với biểu đồ
- Một số phân phối xác suất quan trọng trong thống kê
- Ước lượng và kiểm định
- Kiểm định phi tham số
- Tương quan và hồi quy

Các bài toán trên được minh hoạ với Python.

Tài liệu được biên soạn từ việc tham khảo, lược dịch từ nhiều nguồn tài liệu khác. Các tài liệu tham khảo được trích dẫn rõ ràng. Do sự học của sinh viên, rất mong các tác giả của các tài liệu được tham khảo bỏ quá, khi chúng tôi đã không kịp có lời với một số tác giả.

Tác giả xin bày tỏ lòng biết ơn đến các Thầy cô, bạn hữu đã mang lại những cơ duyên từ những năm 90 đến nay để tác giả được kết nối Thống kê với Lập trình.

Rất mong sự góp ý của người đọc xa gần để tài liệu được hoàn thiện hơn.

Trân trọng cảm ơn.

Huà đông Tân Sơn, 2021



N.T.D

N.T.D

MỤC LỤC

MỤC LỤC	4
DANH MỤC BIỂU ĐỒ VÀ HÌNH VẼ	9
CHƯƠNG 1. GIỚI THIỆU	11
1.1. Giới thiệu về Python	11
1.2. Các thư viện quan trọng sử dụng trong thống kê	12
Tóm tắt chương 1	13
Câu hỏi ôn tập và bài tập chương 1	13
CHƯƠNG 2. XỬ LÝ DỮ LIỆU CƠ BẢN VỚI PYTHON	14
2.1. Thư viện Numpy	14
2.1.1 Tạo mảng	14
2.1.2 Kiểm tra mảng.....	15
2.1.3 Định lại hàng cột.....	15
2.1.4. Mảng ở dạng stack	17
2.1.5. Chọn 1 phần tử của mảng	17
2.2. Thư viện Pandas	18
2.2.1. Cấu trúc dữ liệu.....	18
2.2.3. Bảng pivoting	20
2.2.4. Thống kê mô tả cơ bản các thông tin của dataframe	21
2.2.5. Sắp xếp.....	23
2.2.6. Thống kê mô tả cho dataframe	25
2.2.7. Kiểm tra lại dữ liệu	27
2.3. Xử lý dữ liệu ngoại lệ.....	29
2.4. Vào ra file	30
2.4.1. File dạng csv	30
2.4.2. Xử lý file Excel	32
2.4.3. Đọc dữ liệu từ file dữ liệu của SQL (SQLite)	35
Tóm tắt chương 2	36
Câu hỏi ôn tập và bài tập chương 2.....	36
CHƯƠNG 3. SƠ LƯỢC THỐNG KÊ ỨNG DỤNG.....	38
3.1. Sơ lược về thống kê.....	38
3.2. Một số khái niệm cơ bản	42

3.2.1. Dân số và Mẫu	42
3.2.2. Giá trị ngoại lệ	42
3.3. Một số đại lượng biểu thị xu hướng trung tâm	42
3.3.1. Giá trị trung bình.....	42
3.3.2. Trung bình có trọng số.....	43
3.3.3. Median	43
3.3.4. Mode	44
3.4. Một số đại lượng biểu thị xu hướng phân tán của dữ liệu	44
3.4.1. Phương sai.....	44
3.4.2. Độ lệch chuẩn	45
3.4.3. Sai số tiêu chuẩn (standard error)	45
3.4.4. Đại lượng Skewness & Kurtosis.....	46
3.4.5. Bách phân vị (Percentile).....	47
3.4.6. Phạm vi	48
3.5. Phương thức describe() cho thống kê mô tả đơn giản	49
3.6. Tính điểm Z.....	51
Tóm tắt chương 3	53
Câu hỏi ôn tập và bài tập chương 3.....	53
Chương 4. LẤY MẪU DỮ LIỆU VÀ LẬP BẢNG THỐNG KÊ	55
4.1. Lấy mẫu theo cụm.....	55
4.2. Lấy mẫu có hệ thống	58
4.3. Tạo bảng tần số và tần suất	60
4.4. Chuẩn hóa dữ liệu	63
4.5. Tính tần suất tương đối	65
Tóm tắt chương 4	66
Câu hỏi ôn tập và bài tập chương 4.....	66
Chương 5. TRỰC QUAN HOÁ DỮ LIỆU VỚI BIỂU ĐỒ.....	68
5. 1. Mẫu tổng quát để vẽ biểu đồ với Python	68
5.2. Biểu đồ histogram	69
5.3. Biểu đồ tròn (Pie)	72
5.4. Vẽ biểu đồ đường.....	76
5.5. Biểu đồ phân tán (Scatter).....	80
5.6. Biểu đồ thanh (bar chat)	83

5.7. Biểu đồ thanh ngang.....	86
5.8. Errore bar (thanh lỗi).....	90
5.9. Biểu đồ hình hộp (Boxplot).....	91
5.10. Biểu đồ Boxplot và Violin plot: 1 nhân tố	93
5.11. Boxplot và violin plot: hai nhân tố.....	94
5.12. Vẽ đồ thị phân bố xác suất và mật độ xác suất	95
5.13. Vẽ đồ thị trên cùng một trục	98
5.14. Lưu hình ảnh vừa vẽ.....	100
Tóm tắt chương 5	100
Câu hỏi ôn tập và bài tập chương 5.....	101
CHƯƠNG 6. MỘT SỐ PHÂN PHỐI XÁC SUẤT QUAN TRỌNG TRONG THỐNG KÊ.....	102
6.1. Hàm khối lượng xác suất và hàm phân phối tích lũy.....	102
6.2. Minh hoạ biến ngẫu nhiên có phân phối Bernoulli	105
6.3. Biến ngẫu nhiên nhị thức	107
6.4. Biến ngẫu nhiên hình học.....	109
6.5. Biến ngẫu nhiên Poisson	111
6.6. Hàm mật độ xác suất	113
6.7. Biến ngẫu nhiên có phân phối đều	114
6.8. Biến ngẫu nhiên có phân phối mũ.....	116
6.9. Biến ngẫu nhiên Gaussian.....	118
6.10. Sinh số ngẫu nhiên	123
6.11. Phân phối t.....	124
6.12. Phân phối Chi - bình phương	126
6.13. Phân phối F.....	128
6.14. Phân bố xác suất Weibull.....	129
Tóm tắt chương 6	129
Câu hỏi ôn tập và bài tập chương 6.....	129
CHƯƠNG 7. ƯỚC LƯỢNG VÀ KIỂM ĐỊNH.....	131
7.1. Giới thiệu bài toán kiểm định với công cụ máy tính.....	131
7.2. Giá trị P và Ý nghĩa Thống kê	134
7.3. Kiểm định One-Sample T-Test	137
7.4. T test 1 mẫu (One sample t-test)	138

7.5. Kiểm định t - hai mẫu (Two sample t test)	140
7.6. Kiểm định T với phân phối Welch (T-test Welch)	141
7.7. T-Test các mẫu được ghép nối (Paired Samples T-Test).....	143
7.8. Kiểm tra T-test, giá trị trung bình của hai mẫu độc lập	144
7.9. Giá trị tới hạn T	146
7.10. Z- test 1 mẫu và 2 mẫu (one sample z – test và two sample z – test)	148
7.11. Phép kiểm tra z-test một tỷ lệ và hai tỷ lệ	150
7.12. Giá trị tới hạn Z	153
7.13. Mức độ ý nghĩa	154
7.14. ANOVA một chiều	159
7.15. ANOVA hai chiều.....	160
7.16. Khoảng tin cậy (confidence interval).....	162
Tóm tắt chương 7	165
Câu hỏi ôn tập và bài tập chương 7.....	165
Chương 8. KIỂM ĐỊNH PHI THAM SỐ.....	167
8.1. Kiểm tra Wilcoxon	167
8.2. Kiểm tra tính độc lập Chi-Square.....	169
8.3. Giá trị tới hạn Chi-Square	170
8.4. Kiểm tra Mann-Whitney U	172
8.5. Kiểm tra mối liên quan giữa hai biến phân loại (kiểm tra Fisher)	173
8.6. Kiểm tra tính chuẩn của dữ liệu	175
KIỂM ĐỊNH TÍNH PHÙ HỢP	175
8.7. Kiểm tra Shapiro-Wilk	175
8.8. Kiểm tra Anderson-Darling.....	177
8.9. Kiểm tra Kolmogorov-Smirnov	179
8.10. Kiểm tra phân phối xác suất của biến phân loại bằng phép kiểm tra Chi-Square	181
8.11. Kiểm tra tỷ lệ (kiểm tra McNemar).....	182
8.12. Kiểm tra nhị thức.....	184
8.13. Kiểm tra Bartlett.....	185
8.14. Kiểm định F-test.....	187
8.15. Giá trị tới hạn F	188
8.16. Kiểm tra so sánh phương sai của 2 nhóm (Kiểm tra Levene).....	190

Tóm tắt chương 8	191
Câu hỏi ôn tập và bài tập chương 8.....	191
CHƯƠNG 9. TƯƠNG QUAN VÀ HỒI QUY	193
9.1. Hệ số tương quan	193
9.2. Tương quan thứ hạng	196
Hệ số tương quan Spearman	196
9.3. Hệ số tương quan Kendall.....	198
9.4. Tương quan bộ phận.....	199
9.5. Ma trận tương quan	201
9.6. Ma trận hiệp phương sai (Covariance Matrix).....	204
9.7. Hồi quy tuyến tính.....	207
9.8. Phần dư trong hồi quy tuyến tính.....	212
9.9. Hồi quy bậc hai	217
9.10. Hồi quy đa thức	218
Tóm tắt chương 9	220
Câu hỏi ôn tập và bài tập chương 9.....	220
TÀI LIỆU THAM KHẢO	221

DANH MỤC BIỂU ĐỒ VÀ HÌNH VẼ

Hình 1. 1. Hệ sinh thái Python cho khoa học dữ liệu ([2])	11
Hình 5. 1. Minh hoạ biểu đồ histogram.....	70
Hình 5. 2. Minh hoạ biểu đồ tròn	73
Hình 5. 3. Minh hoạ biểu đồ đường	78
Hình 5. 4. Minh hoạ biểu đồ phân tán (scatter).....	82
Hình 5. 5. Minh hoạ biểu đồ thanh (bar)	84
Hình 5. 6. Minh hoạ biểu đồ thang ngang	87
Hình 5. 7. Minh hoạ thanh lỗi.....	91
Hình 5. 8. Minh hoạ biểu đồ dạng hộp (box plot)	91
Hình 5. 9. Biểu đồ Boxplot và Violin plot một nhân tố	94
Hình 5. 10. Boxplot và violin plot hai nhân tố	95
Hình 5. 11. Minh hoạ đồ thị phân bố xác suất và mật độ xác suất.....	96
Hình 5. 12. Minh hoạ đồ thị phân bố xác suất và mật độ xác suất (Nguồn [3])	97
Hình 6. 1. Đồ thị PMF của biến ngẫu nhiên có phân phối Bernoulli	106
Hình 6. 2. Biểu đồ histogram của biến ngẫu nhiên có phân phối nhị thức	108
Hình 6. 3. Đồ thị PMF của biến nhị phân có phân phối nhị thức.....	108
Hình 6. 4. Đồ thị của hàm PMF và CDF của biến nhị thức với tham số n, p	109
Hình 6. 5. Biểu đồ histogram và đồ thị hàm PMF của biến ngẫu nhiên hình học (geometric) với $p = 0,5$	110
Hình 6. 6. Biểu đồ histogram và đồ thị hàm PMF của biến ngẫu nhiên có phân phối Poisson.....	112
Hình 6. 7. Xấp xỉ biến ngẫu nhiên nhị thức bởi biến ngẫu nhiên Poisson.....	113
Hình 6. 8. PDF và CDF của biến ngẫu nhiên phân phối đều	115
Hình 6. 9. Đồ thị của hàm PDF và CDF của biến ngẫu nhiên phân phối mũ	117
Hình 6. 10. Minh hoạ PDF and CDF của biến ngẫu nhiên phân phối chuẩn Gaussian	119
Hình 6. 11. Phân phối nhị thức và xấp xỉ phân phối chuẩn theo định lý giới hạn trung tâm	123
Hình 6. 12. Đồ thị hàm phối xác suất Chi-square với 4 bậc tự do	127
Hình 7. 1. Các bài toán kiểm định cơ bản	137
Hình 9. 1. Biểu đồ heatmap, minh hoạ mối tương quan giữa hai biến.....	203
Hình 9. 2. Biểu đồ heatmap, minh hoạ ma trận hiệp phương sai	206
Hình 9. 3. Biểu đồ scatter minh hoạ mối tương quan giữa hai biến.....	207
Hình 9. 4. Biểu đồ scatter minh hoạ mối tương quan giữa hai biến có tương quan tuyến tính	213

Hình 9. 5. Minh hoạ đường thẳng hồi quy tuyến tính (“đường thẳng phù hợp nhất”)	213
Hình 9. 6. Minh hoạ phần dư.....	214
Hình 9. 7. Minh hoạ hồi quy bậc hai	218
Hình 9. 8. Minh hoạ hồi quy với đường cong Lagrange	219

CHƯƠNG 1. GIỚI THIỆU

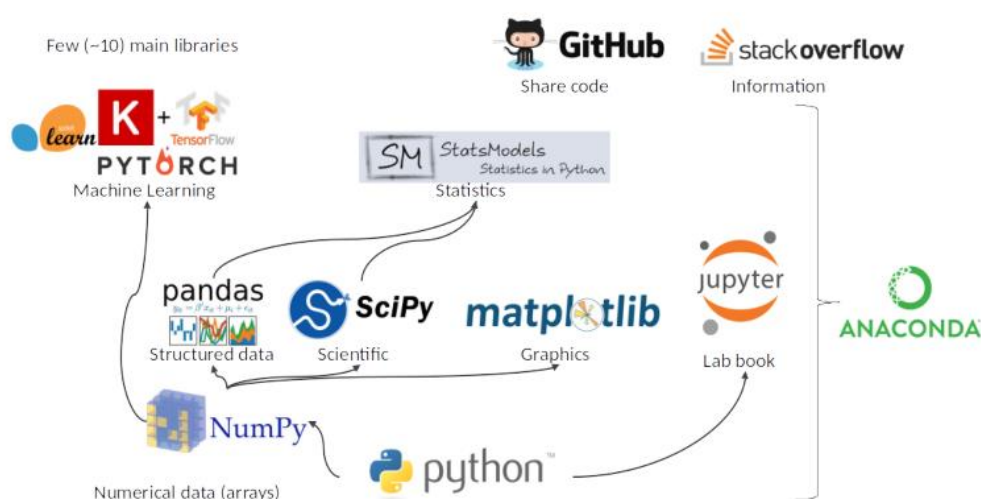
Mục đích

- Giới thiệu sơ lược về Python, cách cài đặt Python
- Giới thiệu các package cơ bản cho thống kê

Yêu cầu

- Cài đặt được python và IDE cho Python lên máy tính
- Chạy thử các ví dụ trong chương để làm quen với các câu lệnh cơ bản, cũng như import các thư viện cần thiết

1.1. Giới thiệu về Python



Hình 1. 1. Hệ sinh thái Python cho khoa học dữ liệu ([2])

Python là một ngôn ngữ lập trình bậc cao do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python vừa hướng thủ tục (procedural-oriented), vừa hướng đối tượng (object-oriented) đồng thời có thể nhúng vào ứng dụng như một giao tiếp kịch bản (scripting interface).

Thế mạnh của Python là rất gần gũi với ngôn ngữ tự nhiên (tiếng Anh), cấu trúc rõ ràng, dễ đọc, dễ học. Python hiện nay là ngôn ngữ lập trình phổ biến rộng rãi ở châu Âu, châu Mỹ và được coi như ngôn ngữ lập trình trường học.

Python được dùng để phát triển các ứng dụng web, game, khoa học dữ liệu (tính toán, phân tích, khai thác dữ liệu), máy học và trí tuệ nhân tạo, ...

Tải về từ <https://www.python.org/downloads/> và tiến hành cài đặt (chọn phiên bản 3.8 trở lên).

Sau khi hoàn tất cài đặt có thể kiểm tra:

- Nhấn phím Windows gõ cmd và Enter
- Gõ: python --version à Enter

Lúc này sẽ hiển thị phiên bản Python đã cài đặt trên máy tính.

Để lập trình theo một ngôn ngữ nào đó ta đều cần có chương trình cho phép gõ các câu lệnh và ra lệnh thực thi các câu lệnh đó. Trong các trường học, để lập trình với Pascal ta thường sử dụng FreePascal, với C ta thường dùng CodeBlock, ... Với Python, ta có nhiều lựa chọn.

a) Python Online Compiler

Search với từ khóa "Python Online Compiler" sẽ trả về kết quả với nhiều website khác nhau. Ví dụ <https://www.programiz.com/python-programming/online-compiler/> là một ứng dụng được cung cấp bởi Programiz.

b) Notepad++, Sublime Text, ...

Tải Notepad++ về tại đây: <https://notepad-plus-plus.org/downloads/>

Tải Sublime Text về tại đây: <https://www.sublimetext.com/3>

Đặc điểm: Đơn giản, dễ sử dụng.

Nhược điểm: Phải cài đặt plugin, thiết lập thêm mới có thể run python code.

c) Thonny

Tải về tại đây: <https://thonny.org/>

Thonny có giao diện đơn giản, cấu hình nhẹ (trên cùng một máy khởi động nhanh hơn nhiều so với PyCharm hay Spyder). Hỗ trợ debug trực quan giúp ta dễ theo dõi và hình dung quá trình thực thi chương trình. Sử dụng thư viện / module chuẩn của Python phát hành (không bổ sung hay import sẵn module).

d) PyCharm Educational Edition

Tải về tại đây: <https://www.jetbrains.com/pycharm-edu/>

PyCharm là môi trường phát triển tích hợp đa nền tảng (IDE) được phát triển bởi Jet Brains và được thiết kế đặc biệt cho Python. Tuy nhiên PyCharm khởi động khá nặng nề và yêu cầu làm việc với project. Như vậy, tùy nhu cầu sử dụng và kỹ năng lập trình mà chúng ta lựa chọn trình soạn thảo cho phù hợp. Đối với người mới bắt đầu học Python thì nên dùng Thonny để thực hành.

1.2. Các thư viện quan trọng sử dụng trong thống kê

numpy: dùng cho các kiểu dữ liệu vector và array

scipy: dùng cho các thuật toán cơ bản trong thống kê

matplotlib: dùng để vẽ các dạng đồ thị

seaborn: dùng để vẽ các dạng đồ thị và thực hiện thống kê nâng cao

pandas: Thao tác dữ liệu có cấu trúc (bảng). nhập / xuất tệp excel, v.v.

statsmodels: dùng để mô hình hóa thống kê và phân tích nâng cao, ví dụ như phân tích hồi quy và phân tích phương sai.

Hướng dẫn cài đặt các thư viện: vào Command Prompt của Window gõ lệnh:
pip install <tên gói>

Nếu chương trình không nhận biết được lệnh trên thì gõ lệnh

py -m pip install <tên gói>

Trong chương trình viết bởi Python ta thường gặp các câu lệnh có dạng:

import <tên thư viện> as <bí danh>

Câu lệnh nhằm thực hiện việc load một thư viện dưới 1 tên bí danh, để có thể thực hiện các phương thức, các hàm có trong thư viện.

Hoặc câu lệnh có dạng **from <tên thư viện> import <thư viện con>**

cũng nhằm thực hiện việc load một thư viện con.

Xét các ví dụ sau:

```
# 'generic import' of math module
import math
math.sqrt(25)
# import a function
from math import sqrt
sqrt(25) # no longer have to reference the module
# import multiple functions at once
from math import cos, floor
# import all functions in a module (generally discouraged)
# from os import *
# define an alias
import numpy as np
# show all functions in math module
content = dir(math)
```

Tóm tắt chương 1

Python cùng với các thư viện và cộng đồng người sử dụng Python đã và đang tạo ra hệ sinh thái xử lý dữ liệu – một ngành đang rất sôi động của Tin học hiện nay.

Cách cài đặt, thực hiện và các thư viện cơ bản đã được giới thiệu ở chương này.

Câu hỏi ôn tập và bài tập chương 1

Cài đặt và thực hiện các ví dụ trong chương này, đặc biệt lưu ý cách import các thư viện cơ bản của Python.

CHƯƠNG 2. XỬ LÝ DỮ LIỆU CƠ BẢN VỚI PYTHON

Mục đích

- Giới thiệu 2 thư viện cơ bản của python là numpy và pandas để lưu trữ và xử lý dữ liệu cơ bản, cho việc thử nghiệm các bài toán thống kê ở các chương sau.

Yêu cầu

- Cài đặt thực hiện được các ví dụ mẫu trong chương 2.
- Giải thích được các output của các ví dụ và các câu lệnh cơ bản của các đoạn lệnh minh họa.
- Vận dụng để thực hiện một số cấu trúc dữ liệu cơ bản.

2.1. Thư viện Numpy

NumPy là một phần mở rộng cho ngôn ngữ lập trình Python, bổ sung hỗ trợ cho các mảng và ma trận lớn, đa chiều (số), cùng với một thư viện lớn các hàm toán học cấp cao để hoạt động trên các mảng này.

(Các mã nguồn trong chương này được tham khảo từ [3])

2.1.1 Tạo mảng

Tạo ndarrays từ danh sách. lưu ý: mọi phần tử phải cùng loại.

```
import numpy as np
data1 = [1, 2, 3, 4, 5] # list
arr1 = np.array(data1) # 1d array
data2 = [range(1, 5), range(5, 9)] # list of lists
arr2 = np.array(data2) # 2d array
arr2.tolist() # convert array back to list
```

Out:

```
[[1, 2, 3, 4], [5, 6, 7, 8]]
```

create special arrays

```
np.zeros(10)
```

```
np.zeros((3, 6))
```

```
np.ones(10)
```

```
np.linspace(0, 1, 5) # 0 to 1 (inclusive) with 5 points
```

```
np.logspace(0, 3, 4) # 10^0 to 10^3 (inclusive) with 4 points
```

Out:

```
array([ 1., 10., 100., 1000.])
```

arange is like range, except it returns an array (not a list)

```
int_array = np.arange(5)
float_array = int_array.astype(float)
```

2.1.2 Kiểm tra mảng

```
arr1.dtype    # float64
arr2.ndim     # 2
arr2.shape    # (2, 4) - axis 0 is rows, axis 1 is columns
arr2.size     # 8 - total number of elements
len(arr2)     # 2 - size of first dimension (aka axis)
```

2.1.3 Định lại hàng cột

```
arr = np.arange(10, dtype=float).reshape((2, 5))

print(arr.shape)

print(arr.reshape(5, 2))
```

Out:

```
2, 5)

[0. 1.]
[2. 3.]
[4. 5.]
[6. 7.]
[8. 9.]]
```

Thêm vào 1 cột/hàng

```
a = np.array([0, 1])
a_col = a[:, np.newaxis]
print(a_col)
#or
a_col = a[:, None]
```

Out:

```
[[0]
 [1]]
```

Chuyển vị 1 mảng

```
print(a_col.T)
```

Out:

```
[[0 1]]
```

Ma trận được lưu trữ theo hàng

Đối với mảng 2 chiều, khi đọc dữ liệu thì Numpy mặc định di chuyển tuần tự như sau:

- lặp qua các hàng (axis 0)
- lặp qua các cột (axis 1)

Đối với mảng 3 chiều, di chuyển tuần tự như sau:

- lặp lại các chiều (axis 0)
- lặp qua các hàng (axis 1)
- lặp qua các cột (axis 2)

Ví dụ:

```
x = np.arange(2 * 3 * 4)
print(x)
Out:
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
 22 23]

# Reshape into 3D (axis 0, axis 1, axis 2)

x = x.reshape(2, 3, 4)
print(x)
Out:

[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]

# Selection get first plan
print(x[0, :, :])
Out:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

# Selection get first rows
print(x[:, 0, :])

Out:
[[ 0  1  2  3]
```



```
[12 13 14 15]]

# Selection get first columns

print(x[:, :, 0])
Out:

[[ 0  4  8]

 [12 16 20]]
```

2.1.4. Mảng ở dạng stack

```
a = np.array([0, 1])
b = np.array([2, 3])

#Stack ngang

np.hstack([a, b])

Out:
array([0, 1, 2, 3])

# Stack dọc (Vertical stacking)
np.vstack([a, b])
```

```
Out:
array([[0, 1],
       [2, 3]])

Lưu ý: mặc định là vstack

np.stack([a, b])
Out:
array([[0, 1],
       [2, 3]])
```

2.1.5. Chọn 1 phần tử của mảng

Chọn 1 phần tử của mảng

```
arr = np.arange(10, dtype=float).reshape((2, 5))
arr[0]          # 0th element (slices like a list)
arr[0, 3]       # row 0, column 3: returns 4
arr[0][3]       # alternative syntax
Out:
3.0
```

Cắt lát mảng 2 hay 3 chiều

Cú pháp: **start:stop:step**

với start (default 0) stop (default last) step (default 1)

Ví dụ:

```
arr[0, :]      # row 0: returns hàng 1 của mảng ([1, 2, 3, 4])
arr[:, 0]      # column 0: returns cột 1 của mảng ([1, 5])
arr[:, :2]     # columns strictly before index 2 (2 first
               # columns)
arr[:, 2:]     # columns after index 2 included

arr2 = arr[:, 1:4]  # columns between index 1 (included) and 4
                   # (excluded)
print(arr2)
Out:
[[1.  2.  3.]
 [6.  7.  8.]]
```

2.2. Thư viện Pandas

2.2.1. Cấu trúc dữ liệu

• Series (chuỗi) là mảng có nhãn một chiều có khả năng chứa bất kỳ kiểu dữ liệu nào (số nguyên, chuỗi, số dấu phẩy động, hay các đối tượng của Python, v.v.). Các nhãn trực được gọi chung là chỉ số. Phương pháp cơ bản để tạo chuỗi là gọi: `pd.series`.

Ví dụ `pd.series([1,3,5, np.nan, 6,8])`

• DataFrame là một cấu trúc dữ liệu có nhãn 2 chiều với các cột có nhiều kiểu khác nhau. DataFrame giống như một bảng tính hoặc bảng trong SQL.

Tạo DataFrame

```
columns = ['name', 'age', 'gender', 'job']
user1 = pd.DataFrame([[ 'alice', 19, "F", "student"],
                      [ 'john', 26, "M", "student"]],
                      columns=columns)
user2 = pd.DataFrame([[ 'eric', 22, "M", "student"],
                      [ 'paul', 58, "F", "manager"]],
                      columns=columns)
user3 = pd.DataFrame(dict(name=[ 'peter', 'julie'],
                           age=[33, 44], gender=[ 'M', 'F'],
                           job=[ 'engineer', 'scientist']))
print(user3)
```

Out:

	name	age	gender	job
0	peter	33	M	engineer
1	julie	44	F	scientist

Kết hợp các DataFrame

Kết hợp cột (axis = 1)

```
height = pd.DataFrame(dict(height=[1.65, 1.8]))
print(user1, "\n", height)
print(pd.concat([user1, height], axis=1))
```

Out:

	name	age	gender	job
0	alice	19	F	student
1	john	26	M	student

height

0	1.65
1	1.80

	name	age	gender	job	height
0	alice	19	F	student	1.65
1	john	26	M	student	1.80

Kết hợp hàng (default: axis = 0)

```
users = pd.concat([user1, user2, user3])
print(users)
```

Out:

	name	age	gender	job
0	alice	19	F	student
1	john	26	M	student
0	eric	22	M	student
1	paul	58	F	manager
0	peter	33	M	engineer
1	julie	44	F	scientist

Kết hợp hàng: append

```
user1.append(user2)
```

Nối các DataFrame

```
user4 = pd.DataFrame(dict(name=['alice', 'john', 'eric', 'julie'], height=[165, 180, 175, 171]))
```

Sử dụng phép giao các khoá của 2 frame (kết nối trong 2 bảng - innner joint)

```
merge_inter = pd.merge(users, user4)
print(merge_inter)
```

Out:

	name	age	gender	job	height
0	alice	19	F	student	165
1	john	26	M	student	180
2	eric	22	M	student	175
3	julie	44	F	scientist	171

Sử dụng hợp các khoá của 2 frame (kết nối ngoài 2 bảng - outer joint)

```
users = pd.merge(users, user4, on="name", how='outer')
print(users)
```

Out:

	name	age	gender	job	height
0	alice	19	F	student	165.0
1	john	26	M	student	180.0
2	eric	22	M	student	175.0
3	paul	58	F	manager	NaN
4	peter	33	M	engineer	NaN
5	julie	44	F	scientist	171.0

2.2.3. Bảng pivoting

"Unpivots" sẽ tạo ra một dataframe từ định dạng rộng sang định dạng dài (xếp chồng lên nhau).

```
staked = pd.melt(users, id_vars="name", var_name="variable",
value_name="value")
print(staked)
```

Out:

	name	variable	value
0	alice	age	19
1	john	age	26
2	eric	age	22
3	paul	age	58
4	peter	age	33
5	julie	age	44
6	alice	gender	F
7	john	gender	M
8	eric	gender	M
9	paul	gender	F
10	peter	gender	M
11	julie	gender	F
12	alice	job	student
13	john	job	student
14	eric	job	student
15	paul	job	manager
16	peter	job	engineer
17	julie	job	scientist

```
18  alice      height      165
19  john      height      180
20  eric      height      175
21  paul      height      NaN
22  peter     height      NaN
23  julie     height      171
```

"Pivot" sẽ tạo ra một DataFrame từ định dạng dài (xếp chồng lên nhau) sang định dạng rộng

```
print(staked.pivot(index='name', columns='variable',
values='value'))
```

Out:

variable	age	gender	height	job
name				
alice	19	F	165	student
eric	22	M	175	student
john	26	M	180	student
julie	44	F	171	scientist
paul	58	F	NaN	manager
peter	33	M	NaN	engineer

Tổng hợp các thông tin của dataframe

```
users          # print the first 30 and last 30 rows
type(users)    # DataFrame
users.head()   # print the first 5 rows
users.tail()   # print the last 5 rows
```

2.2.4. Thống kê mô tả cơ bản các thông tin của dataframe

```
users.describe(include="all")
```

Meta-information

```
users.index     # "Row names"
users.columns   # column names
users.dtypes    # data types of each column
users.values    # underlying numpy array
users.shape     # number of rows and columns
```

Ví dụ với user4, ta có:

```
import pandas as pd
```

N.T.D

```
user4 = pd.DataFrame(dict(name=['alice', 'john', 'eric',
'julie'], height=[165, 180, 175, 171]))
print(user4.describe(include="all"))
```

Out:

	name	height
count	4	4.000000
unique	4	NaN
top	alice	NaN
freq	1	NaN
mean	NaN	172.750000
std	NaN	6.344289
min	NaN	165.000000
25%	NaN	169.500000
50%	NaN	173.000000
75%	NaN	176.250000
max	NaN	180.000000

Chọn cột của 1 dataframe

```
users['gender'] # select one column
type(users['gender']) # Series
users.gender # select one column using the DataFrame
# select multiple columns
users[['age', 'gender']] # select two columns
my_cols = ['age', 'gender'] # or, create a list...
users[my_cols] # ...and use that list to select columns
type(users[my_cols]) # DataFrame
```

Chọn hàng của 1 dataframe

Sử dụng iloc để lựa chọn hàng dựa trên các chỉ số nguyên

```
df = users.copy()
df.iloc[0] # first row
df.iloc[0, :] # first row
df.iloc[0, 0] # first item of first row
```

```
df.iloc[0, 0] = 55
```

Sử dụng loc để lựa chọn dựa trên chỉ số nguyên lần một nhân (label)

```
df.loc[0]          # first row
df.loc[0, :]       # first row
df.loc[0, "age"]    # age item of first row
df.loc[0, "age"] = 55
```

Lựa chọn và lập chỉ mục

Ví dụ sau chỉ ra cách chọn các hàng từ dataframe user có gender là "F" để đưa vào DataFrame mới

```
df = users[users.gender == "F"]
print(df)
```

Out:

	name	age	gender	job	height
0	alice	19	F	student	165.0
3	paul	58	F	manager	NaN
5	julie	44	F	scientist	171.0

Lấy 2 hàng đầu tiên bằng cách sử dụng *iloc*

```
df.iloc[[0, 1], :] # Ok, but watch the index: 0, 3
```

Lập lại chỉ mục

```
df = df.reset_index(drop=True) # Watch the index
print(df)
print(df.loc[[0, 1], :])
```

Out:

	name	age	gender	job	height
0	alice	19	F	student	165.0
1	paul	58	F	manager	NaN
2	julie	44	F	scientist	171.0

	name	age	gender	job	height
0	alice	19	F	student	165.0
1	paul	58	F	manager	NaN

2.2.5. Sắp xếp

Sắp xếp có lặp lại các hàng

Phương thức iterrows():

- Trả về các cặp (chỉ mục, series).
- Truy xuất các trường có tên cột
- Không cho phép chỉnh sửa dữ liệu. Tùy thuộc vào loại dữ liệu, trình lặp trả về một bản sao chứ không phải một dạng xem và việc ghi vào nó sẽ không có hiệu lực.

```
for idx, row in df.iterrows():
    print(row["name"], row["age"])
```

```
Out:
alice 19
john 26
```

Phương thức itertuples ():

- Trả về các bộ (tuple) của các giá trị.
- Truy xuất các trường có chỉ số nguyên bắt đầu từ 0.

```
for tup in df.itertuples():
    print(tup[1], tup[2])
```

```
Out:
alice 19
john 26
```

Ta cũng có thể sử dụng `loc[i, ...]` để đọc và chỉnh sửa dữ liệu

```
for i in range(df.shape[0]):
    df.loc[i, "age"] *= 10      # df is modified
```

Trong 2 câu lệnh trên thì trường tuổi (age) của dataframe df được nhân thêm 10.

Lựa chọn hàng theo phép lọc (filtering)

Phép lọc đơn giản trên các giá trị số

```
users[users.age < 20]
young_bool = users.age < 20
young = users[young_bool]
users[users.age < 20].job
print(young)
```

only show users with age < 20
or, create a Series of booleans...
...and use that Series to filter rows
select one column from the filtered results

Out:


```

      name age  gender  job  height
0    alice   19    F    student  165.0

```

Phép lọc đơn giản trên biến bất kỳ

```

users[users.job == 'student']
users[users.job.isin(['student', 'engineer'])]
users[users['job'].str.contains("stu|scient")]

```

Lọc với điều kiện kết hợp

```

users[users.age < 20][['age', 'job']] # select multiple
users[(users.age > 20) & (users.gender == 'M')] # use multiple
                                              conditions

```

Sắp xếp

```

df = users.copy()

df.age.sort_values() # only works for a
df.sort_values(by='age') # sort rows by a
                        specific column

df.sort_values(by='age', ascending=False) #
use descending order instead

df.sort_values(by=['job', 'age']) # sort by multiple
                                columns

df.sort_values(by=['job', 'age'], inplace=True) # modify df
print(df)

```

Dữ liệu của chúng ta sẽ được sắp theo job và age

Out:

```

      name age  gender  job  height
4    peter   33    M    engineer  NaN
3    paul   58    F    manager  NaN
5    julie   44    F    scientist  171.0
0    alice   19    F    student  165.0
2    eric   22    M    student  175.0
1    john   26    M    student  180.0

```

2.2.6. Thống kê mô tả cho dataframe

Thống kê cho các dữ liệu số

```
print(df.describe())
```

Out:

```

age          height
count  6.000000    4.000000
mean   33.666667   172.750000

```

```
std      14.895189  6.344289
min      19.000000 165.000000
25%      23.000000 169.500000
50%      29.500000 173.000000
75%      41.250000 176.250000
max      58.000000 180.000000
```

Thống kê tất cả các cột

```
print(df.describe(include='all'))
print(df.describe(include=['object'])) # limit to one (or more)
types
```

Out:

```

      name  age      gender  job  height
count      6  6.000000    6      6    4.000000
unique      6   NaN        2      4     NaN
top      peter   NaN        F      student NaN
freq       1   NaN        3      3     NaN
mean      NaN  33.666667   NaN    NaN  172.750000
std      NaN  14.895189   NaN    NaN   6.344289
min      NaN  19.000000   NaN    NaN  165.000000
25%      NaN  23.000000   NaN    NaN  169.500000
50%      NaN  29.500000   NaN    NaN  173.000000
75%      NaN  41.250000   NaN    NaN  176.250000
max      NaN  58.000000   NaN    NaN  180.000000

      name  gender  job
count      6      6      6
unique      6      2      4
top      peter  F      student
freq       1      3      3
```

Thống kê gộp nhóm (groupby)

```
print(df.groupby("job").mean())
print(df.groupby("job")["age"].mean())
print(df.groupby("job").describe(include='all'))
```

Các thống kê được gộp nhóm (group by) theo cột job.

Dưới đây là mô tả cho 2 câu lệnh đầu tiên của 3 câu lệnh trên.

```

                age      height
job
engineer      33.000000      NaN
manager       58.000000      NaN
scientist     44.000000    171.000000
student       22.333333    173.333333
job
engineer      33.000000
manager       58.000000
scientist     44.000000
student       22.333333
Name: age, dtype: float64

```

Gộp nhóm theo vòng lặp

```

for grp, data in df.groupby("job"):
    print(grp, data)

```

Out:

```

engineer  name  age  gender  job      height
4         peter   33     M      engineer   NaN
manager   name  age  gender  job      height
3         paul   58     F      manager   NaN
scientist  name  age  gender  job      height
5         julie   44     F      scientist  171.0
student   name  age  gender  job      height
0         alice   19     F      student   165.0
2         eric   22     M      student   175.0
1         john   26     M      student   180.0

```

2.2.7. Kiểm tra lại dữ liệu

Loại bỏ các hàng lặp

```
df = users.append(users.iloc[0], ignore_index=True)
```

```

print(df.duplicated())          # Series of booleans
# (True if a row is identical to a
# previous row)

df.duplicated().sum()           # count of duplicates
df[df.duplicated()]             # only show duplicates
df.age.duplicated()             # check a single column
                                # for duplicates

df.duplicated(['age', 'gender']).sum() #
specify columns for finding duplicates
df = df.drop_duplicates()       # drop duplicate rows

```

Dữ liệu thiếu

find missing values in a Series

```
df.height.isnull()      # True if NaN, False otherwise
df.height.notnull()     # False if NaN, True otherwise
df[df.height.notnull()]  # only show rows where age is not NaN
df.height.isnull().sum() # count the missing values
```

find missing values in a DataFrame

```
df.isnull()             # DataFrame of booleans
df.isnull().sum()        # calculate the sum of each column
```

Bỏ đi các dữ liệu thiếu

```
df.dropna()             # drop a row if ANY values are missing
df.dropna(how='all')     # drop a row only if ALL values are
                          # missing
```

Bù vào bằng các dữ liệu khác

```
df.height.mean()
df = users.copy()
df.loc[df.height.isnull(), "height"] = df["height"].mean()
print(df)
```

	name	age	gender	job	height
0	alice	19	F	student	165.00
1	john	26	M	student	180.00
2	eric	22	M	student	175.00
3	paul	58	F	manager	172.75
4	peter	33	M	engineer	172.75
5	julie	44	F	scientist	171.00

Các thay đổi trên dataframe

Đổi tên cột

```
df = users.copy()
df.rename(columns={'name': 'NAME'})
```

Đổi giá trị

```
df.job = df.job.map({'student': 'etudiant', 'manager':
'manager',
'engineer': 'ingenieur', 'scientist': 'scientific'})
```

2.3. Xử lý dữ liệu ngoại lệ

Giá trị **ngoại lệ** là một quan sát nằm cách xa bất thường so với các giá trị khác trong tập dữ liệu. Các yếu tố ngoại lai rất có thể là vấn đề, vì chúng có thể ảnh hưởng đến kết quả của một phân tích.

Cách xác định ngoại lệ

Trước khi có thể loại bỏ các ngoại lệ, trước tiên chúng ta phải quyết định xem những gì ta cho là ngoại lệ. Có hai cách phổ biến để làm như vậy:

a. Sử dụng khoảng tứ phân vị

Phạm vi liên phân tứ (IQR) là sự khác biệt giữa phân vị thứ 75 (Q3) và phân vị thứ 25 (Q1) trong một tập dữ liệu. Giá trị này nói lên mức chênh lệch của 50% giá trị ở giữa.

$$IQR = Q3 - Q1$$

Ta có thể xem một giá trị quan sát là một ngoại lệ, nếu nó lớn hơn 1,5 lần phạm vi phân vị thứ ba (Q3) hoặc bé hơn 1,5 lần phạm vi phân vị đầu tiên (Q1).

X là giá trị ngoại lệ, nếu:

$$X > Q3 + 0.5 \cdot IQR \text{ hoặc } X < Q1 - 0.5 \cdot IQR$$

```
import numpy as np
import pandas as pd
import scipy.stats as stats
#create dataframe with three columns 'A', 'B', 'C'
np.random.seed(10)
data = pd.DataFrame(np.random.randint(0, 10, size=(100, 3)),
columns=['A', 'B', 'C'])

#find Q1, Q3, and interquartile range for each column
Q1 = data.quantile(q=.25)
Q3 = data.quantile(q=.75)
# apply(stats.iqr) tính IQR của data
IQR = data.apply(stats.iqr)

#only keep rows in dataframe that have values within 1.5*IQR
of Q1 and Q3
data_clean = data[~((data < (Q1-0.5*IQR)) | (data >
(Q3+0.5*IQR))).any(axis=1)]

#find how many rows are left in the dataframe
data_clean.shape
(89, 3)
```

(Xem [6], [7])

b. Sử dụng điểm z (z-scores)

Ta cũng có thể xác định một quan sát là một ngoại lệ, nếu nó có điểm z nhỏ hơn -3 hoặc lớn hơn 3.

```
#find absolute value of z-score for each observation
z = np.abs(stats.zscore(data))

#only keep rows in dataframe with all z-scores less than
absolute value of 3
data_clean = data[(z<3).all(axis=1)]

#find how many rows are left in the dataframe
data_clean.shape
```

(Xem [6], [7])

Khi nào thì loại bỏ các ngoại lệ

Nếu một hoặc nhiều ngoại lệ xuất hiện trong dữ liệu, trước tiên ta nên đảm bảo rằng chúng không phải là kết quả của lỗi nhập dữ liệu. Đôi khi một ngoại lệ chỉ đơn giản là do nhập sai giá trị dữ liệu khi ghi dữ liệu.

Nếu giá trị ngoại lai là do lỗi nhập dữ liệu, ta có thể quyết định gán một giá trị mới cho nó, chẳng hạn như giá trị [trung bình](#) của tập dữ liệu.

Nếu giá trị là một giá trị ngoại lệ thực sự, ta có thể chọn loại bỏ nó nếu nó có tác động đáng kể đến phân tích tổng thể. Khi đó cần thông báo trong báo cáo hoặc phân tích tổng kết rằng ta đã loại bỏ một yếu tố ngoại lệ.

2.4. Vào ra file

2.4.1. File dạng csv

```
import tempfile, os.path
tmpdir = tempfile.gettempdir() # lấy thư mục của
file
csv_filename = os.path.join(tmpdir, "users.csv") # định danh file
cần xử lý là users.csv
users.to_csv(csv_filename, index=False) # đọc file csv ra
dataframe users
other = pd.read_csv(csv_filename) # đọc file csv ra
dataframe other
```

Đọc file csv từ một url

```
url =  
'https://github.com/duchesnay/pystatsml/raw/master/datasets/salary_table.csv'  
salary = pd.read_csv(url)      # đọc file csv ở url ở trên vào dataframe salary
```

Đọc từ csv file

Đọc dữ liệu từ file text vào DataFrame:

Ta có thể dễ dàng đọc vào một file **.csv** bằng cách sử dụng hàm **read_csv** và được trả về 1 dataframe.

Cũng có thể dùng hàm **read_csv** để đọc **1 file text** và cũng được trả về 1 dataframe.

Lưu ý một vài tham số của hàm **read_csv** như:

- **encoding**: chỉ định encoding của file đọc vào. Mặc định là utf-8.
- **sep**: thay đổi dấu ngăn cách giữa các cột. Mặc định là dấu phẩy (',')
- **header**: chỉ định file đọc vào có header (tiêu đề của các cột) hay không. Mặc định là infer.
- **index_col**: chỉ định chỉ số cột nào là cột chỉ số(số thứ tự). Mặc định là None.
- **n_rows**: chỉ định số bản ghi sẽ đọc vào. Mặc định là None – đọc toàn bộ.

Cú pháp chung:

```
pd.read_csv(link, header, index_col, sep, encoding)
```

Trong đó:

link: là đường dẫn tới file, có thể là một link trên internet.

header: thứ tự dòng làm tiêu đề.

index_col: thứ tự cột làm index.

sep: loại delimiter ngăn cách giữa các trường. Khoảng trắng là '\t', dấu phẩy là ','.

encoding: loại mã encoding để đọc dữ liệu.

Ví dụ: Đoạn mã dưới đây đọc file book1.csv vào trong dataframe là dataset.

```
import pandas as pd  
  
dataset = pd.read_csv('book1.csv', header = 0, index_col = 0,  
sep = ',')
```

```
dataset.head()  
print(dataset)
```

```
df = pd.read_csv("data.txt", sep=" ", header=None,  
names=["A", "B"])
```

đọc file data.txt có phân tách giữa các cột là dấu trắng “ ” ra dataframe df, với tên cột là A và B.

Với people.csv là 1 file dạng csv. Có thể in ra n bản ghi đầu tiên của dataframe sử dụng hàm **head**. Ngược lại của hàm **head** là hàm **tail**

```
NV_df = pd.read_csv('./people.csv')  
NV_df.head(5)
```

Kết quả in ra sẽ là 5 bản ghi đầu tiên có trong file people.csv này.

Khi không có chỉ định header, thì dòng đầu tiên của file csv sẽ là dòng header của file sẽ biến thành 1 bản ghi dữ liệu.

Có thể xem mô tả đầy đủ từng tham số của hàm **read_csv** của thư viện pandas tại đây (https://pandas.pydata.org/pandasdocs/stable/reference/api/pandas.read_csv.html).

2.4.2. Xử lý file Excel

```
xls_filename = os.path.join(tmpdir, "users.xlsx")  
users.to_excel(xls_filename, sheet_name='users', index=False)  
pd.read_excel(xls_filename, sheet_name='users')  
# Multiple sheets  
with pd.ExcelWriter(xls_filename) as writer:  
    users.to_excel(writer, sheet_name='users', index=False)  
    df.to_excel(writer, sheet_name='salary', index=False)  
pd.read_excel(xls_filename, sheet_name='users')  
pd.read_excel(xls_filename, sheet_name='salary')
```

Để xử lý file của Excel với pandas chúng ta cần cài thêm gói **xlrd**

```
import pandas as pd  
  
xl = pd.ExcelFile('example.xls')  
  
# get the first sheet as an object  
df = pd.read_excel(xl)
```


Đoạn lệnh trên tạo ra bảng df từ file excel example.xls

```
excel_file = 'movies.xls'

movies = pd.read_excel(excel_file)
```

Đoạn lệnh trên tạo ra bảng movies từ file excel movies.xls

Một số xử lý file Excel

Tập Excel là một trong những cách phổ biến nhất để lưu trữ dữ liệu. Hàm pandas [read_excel\(\)](#) cho phép bạn dễ dàng đọc trong các tập Excel.

Giả sử chúng ta có tập Excel sau:

	A	B	C	D	E	F
1	playerID	team	points			
2	1	Lakers	26			
3	2	Mavs	19			
4	3	Bucks	24			
5	4	Spurs	22			
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						

```
import pandas as pd

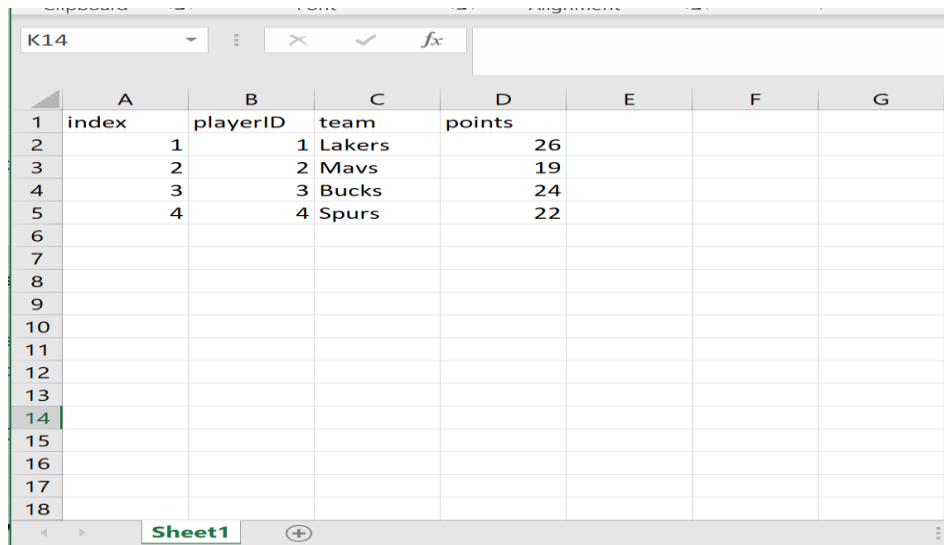
#import Excel file
df = pd.read_excel('data.xlsx')

#view DataFrame
df
```

	playerID	team	points
0	1	Lakers	26
1	2	Mavs	19
2	3	Bucks	24

Ví dụ 2: Đọc tệp Excel với cột chỉ mục

Ta cũng có thể có một tệp Excel trong đó một trong các cột là cột chỉ mục:



	A	B	C	D	E	F	G
1	index	playerID	team	points			
2		1	Lakers	26			
3		2	Mavs	19			
4		3	Bucks	24			
5		4	Spurs	22			
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							

Trong trường hợp này, có thể sử dụng `index_col` để chỉ ra cột nào sẽ sử dụng làm cột chỉ mục:

```
import pandas as pd

#import Excel file, specifying the index column
df = pd.read_excel('data.xlsx', index_col='index')

#view DataFrame
df
```

	playerID	team	points
index			
1	1	Lakers	26
2	2	Mavs	19
3	3	Bucks	24
4	4	Spurs	22

(Xem [6], [7])

Ví dụ: Đọc tệp Excel bằng tên trang tính

Chúng ta cũng có thể đọc tên trang tính cụ thể từ tệp Excel vào DataFrame

Để đọc một trang tính cụ thể dưới dạng DataFrame, có thể sử dụng đối số `sheet_name()` :

```
df = pd.read_excel('data.xlsx', sheet_name='second sheet')
```

Cài đặt xlrd

Khi cố gắng sử dụng hàm `read_excel()`, chúng ta có thể gặp phải lỗi sau:

ImportError: Install xlrd >= 1.0.0 for Excel support

Trong trường hợp này, trước tiên cần cài đặt xlrd:

```
pip install xlrd
```

Sau khi cài đặt này, ta có thể tiếp tục sử dụng hàm `read_excel()`.

Ghi dữ liệu vào tệp Excel

Việc xuất từ DataFrame sang file Excel, được thực hiện cách sử dụng hàm của pandas `to_excel()`

Để sử dụng chức năng này, trước tiên bạn cần cài đặt **openpyxl** để có thể ghi tệp vào Excel:

```
pip install openpyxl
```

Ví dụ :

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'points': [25, 12, 15, 14, 19],
                   'assists': [5, 7, 7, 9, 12],
                   'rebounds': [11, 8, 10, 6, 6]})
df.to_excel(r'C:\Users\Zach\Desktop\mydata.xlsx')
df.to_excel(r'C:\Users\Zach\Desktop\mydata.xlsx', index=False)
df.to_excel(r'C:\Users\Zach\Desktop\mydata.xlsx', index=False,
header=False)
df.to_excel(r'C:\Users\Zach\Desktop\mydata.xlsx',
sheet_name='this_data')
```

2.4.3. Đọc dữ liệu từ file dữ liệu của SQL (SQLite)

```
import pandas as pd
import sqlite3      # thư viện để có thể đọc dữ liệu của
SQLite
db_filename = os.path.join(tmpdir, "users.db")
```

Kết nối file data của SQLite

```
conn = sqlite3.connect(db_filename)
```

Tạo 1 bảng nhờ pandas

```
url = 'https://github.com/duchesnay/pystatsml/raw/master/datasets/salary_table.csv'
salary = pd.read_csv(url)
salary.to_sql("salary", conn, if_exists="replace")
```

Thêm vào một số thay đổi trên file từ SQLite

```
cur = conn.cursor()
values = (100, 14000, 5, 'Bachelor', 'N')
cur.execute("insert into salary values (?, ?, ?, ?, ?)", values)
conn.commit()
```

Đọc kết quả vào 1 dataframe

```
salary_sql = pd.read_sql_query("select * from salary;", conn)
print(salary_sql.head())
pd.read_sql_query("select * from salary;", conn).tail()
pd.read_sql_query('select * from salary where salary>25000;', conn)
pd.read_sql_query('select * from salary where experience=16;', conn)
pd.read_sql_query('select * from salary where education="Master";', conn)
```

Tóm tắt chương 2

Chương 2, giới thiệu hai module quan trọng của Python là Numpy và Pandas trong lưu trữ và xử lý dữ liệu thống kê. Trong đó, tập trung trình bày các ví dụ minh họa cho việc xử lý dữ liệu kiểu dãy (cho xử lý dữ liệu chuỗi thời gian trong thống kê) và dữ liệu dataframe (dạng bảng) cho bảng thống kê, cũng như cách đọc và ghi file trên một số định dạng file thông dụng như CSV và Excel hay SQLite

Câu hỏi ôn tập và bài tập chương 2

1. Thực hiện cài đặt các ví dụ minh họa trong chương, báo cáo kết quả thực hiện.
2. Giải thích các câu lệnh cơ bản để thực hiện trong các đoạn mã lệnh của các ví dụ
3. Thực hiện các ví dụ cho các dữ liệu có tính thực tiễn khác.
4. Kết nối các ví dụ để thực hiện tính toán các đại lượng thống kê cơ bản trên dataframe cho trong file **ngheSi.xlsx** sẵn có trên link sau: <https://docs.google.com/spreadsheets/d/1q7RVfoTKEuLP0Ud0a50mJu7e1-YJJbUe/edit?usp=sharing&ouid=106843557461060771783&rtpof=true&sd=true>

5. Lựa chọn 1 dataset trong địa chỉ sau: <https://www.kaggle.com/datasets> và thực hiện đọc dữ liệu trong dataset thành 1 dataframe, sau đó chuyển đổi từ file csv về file excel và ngược lại.

CHƯƠNG 3. SƠ LƯỢC THỐNG KÊ ỨNG DỤNG

Mục đích

- Giới thiệu sơ lược về thống kê ứng dụng
- Các bài toán cơ bản của thống kê ứng dụng cũng được trình bày
- Các đại lượng cơ bản của thống kê như xu hướng trung tâm, xu hướng phân tán của dữ liệu thống kê được trình bày cùng với một số minh họa tính toán với Python.

Yêu cầu

- Khái quát được các bài toán cơ bản của thống kê
- Cài đặt thực hiện được các ví dụ mẫu trong chương 3.
- Giải thích được các output của các ví dụ và các câu lệnh cơ bản của các đoạn lệnh minh họa.
- Vận dụng để thực hiện một số bài toán thực tế cơ bản.

3.1. Sơ lược về thống kê

Lĩnh vực **thống kê** liên quan đến việc thu thập, phân tích, giải thích và trình bày dữ liệu.

Khi công nghệ trở nên hiện diện nhiều hơn trong cuộc sống hàng ngày của chúng ta, ngày càng có nhiều dữ liệu được tạo ra và thu thập hơn bao giờ hết trong lịch sử loài người.

Thống kê là lĩnh vực có thể giúp chúng ta hiểu cách sử dụng dữ liệu để:

- Hiểu rõ hơn về thế giới xung quanh chúng ta.
- Đưa ra quyết định dựa trên cơ sở dữ liệu.
- Đưa ra dự đoán về tương lai dựa trên dữ liệu.

Hàng ngày chúng ta phải đối mặt với những tình huống có kết quả không chắc chắn và phải đưa ra quyết định dựa trên dữ liệu không đầy đủ, thống kê giúp làm rõ các câu hỏi sau:

- Xác định các biến số và độ đo của các biến số đó, để trả lời cho các quyết định.
- Xác định cỡ mẫu cần thiết phải thu thập để ra quyết định hợp lý
- Mô tả sự biến đổi của dữ liệu.
- Đưa ra các đánh giá định lượng về các tham số ước tính.

- Đưa ra dự đoán dựa trên dữ liệu thu thập được.

Có hai nhánh chính trong lĩnh vực thống kê:

Thống kê mô tả

Thống kê suy luận

Thống kê mô tả

Thống kê mô tả nhằm mục đích *mô tả* một phần dữ liệu thô bằng cách sử dụng thống kê tóm tắt, biểu đồ và bảng. Thống kê mô tả rất hữu ích vì chúng cho phép hiểu một nhóm dữ liệu nhanh chóng và dễ dàng hơn nhiều so với việc chỉ chăm chăm vào các hàng và các hàng giá trị dữ liệu thô.

Thống kê mô tả là mô tả và tóm tắt dữ liệu. Có ba dạng thống kê mô tả phổ biến:

1. Thống kê tổng hợp. Đây là những thống kê tóm tắt dữ liệu bằng một con số duy nhất. Có các loại thống kê tóm tắt phổ biến:

- Đánh giá xu hướng trung tâm : Đánh giá những đại lượng mô tả vị trí trung tâm của tập dữ liệu. Ví dụ: giá trị kỳ vọng (mean); trung vị (median).

- Đánh giá mức độ phân tán : Đánh giá những đại lượng mô tả mức độ phân tán của các giá trị trong tập dữ liệu. Ví dụ: phạm vi , tứ phân vị , độ lệch chuẩn, phương sai .

- Sự tương quan hoặc sự thay đổi chung cho biết về mối quan hệ giữa một cặp biến trong tập dữ liệu. Ví dụ: hiệp phương sai (covariance), hệ số tương quan...

2. Các đồ thị . Đồ thị giúp chúng ta trực quan hoá dữ liệu. Các loại biểu đồ phổ biến được sử dụng để trực quan hóa dữ liệu bao gồm boxplot , histogram , biểu đồ phân tán....

3. Các bảng . Các bảng giúp chúng ta hiểu cách mà dữ liệu được phân phối. Bảng phổ biến là bảng tần suất , nhằm chúng ta biết có bao nhiêu giá trị dữ liệu nằm trong một số phạm vi nhất định.

Ta có thể áp dụng thống kê mô tả cho một hoặc nhiều tập dữ liệu hoặc biến. Khi mô tả và tóm tắt một biến duy nhất, ta đang thực hiện phân tích đơn biến. Khi tìm kiếm các mối quan hệ thống kê giữa một cặp biến, ta đang thực hiện phân tích đa biến . Tương tự, một phân tích đa phương sai liên quan đến nhiều biến cùng một lúc.

Thống kê suy luận

Thống kê suy luận sử dụng một mẫu dữ liệu nhỏ để rút ra *suy luận* về dân số lớn hơn mà mẫu được lấy ra từ đó.

Tầm quan trọng của một mẫu đại diện

Để có thể tự tin vào khả năng sử dụng mẫu để rút ra các suy luận về quần thể, chúng ta cần đảm bảo rằng chúng ta có mẫu đại diện - tức là mẫu mà các đặc điểm của các cá thể trong mẫu phù hợp chặt chẽ với các đặc điểm của tổng thể.

Nếu mẫu của chúng ta không tương tự với tổng thể, thì chúng ta không thể tổng quát hóa các phát hiện từ mẫu thành tổng thể với độ tin cậy.

Cách lấy mẫu đại diện

Để tối đa hóa cơ hội để có được mẫu đại diện, chúng ta cần tập trung vào hai điều:

a. Đảm bảo rằng đã sử dụng phương pháp lấy mẫu ngẫu nhiên

Có một số **phương pháp lấy mẫu** ngẫu nhiên khác nhau, có thể sử dụng để tạo ra một mẫu đại diện, bao gồm:

- Một mẫu ngẫu nhiên đơn giản
- Một mẫu ngẫu nhiên có hệ thống
- Một mẫu ngẫu nhiên theo cụm
- Một mẫu ngẫu nhiên phân tầng

Phương pháp lấy mẫu ngẫu nhiên có xu hướng tạo ra các mẫu đại diện vì mọi thành viên của quần thể đều có cơ hội được đưa vào mẫu như nhau.

b. Đảm bảo kích thước mẫu đủ lớn

Cùng với việc sử dụng phương pháp lấy mẫu thích hợp, điều quan trọng là phải đảm bảo rằng mẫu đủ lớn để có đủ dữ liệu để tổng quát hóa cho tập hợp lớn hơn.

Để xác định mức độ lớn của mẫu, chúng ta phải xem xét quy mô dân số đang nghiên cứu, mức độ tin cậy muốn sử dụng và biên độ sai số được cho là có thể chấp nhận được.

Các dạng thống kê suy luận phổ biến

Có ba dạng thống kê suy luận phổ biến:

a. Kiểm tra giả thuyết

Chúng ta thường quan tâm đến việc trả lời các câu hỏi về dân số như:

- Tỷ lệ người dân ở Ohio ủng hộ ứng cử viên A có cao hơn 50% không?
- Chiều cao trung bình của một loại cây nào đó có bằng 14 inch không?
- Có sự khác biệt giữa chiều cao trung bình của học sinh Trường A so với Trường B không?

Để trả lời những câu hỏi này, chúng ta có thể thực hiện một **bài kiểm định giả thuyết**, cho phép chúng ta sử dụng dữ liệu từ một mẫu để đưa ra kết luận về quần thể.

b. Ước lượng và so sánh

Đôi khi chúng ta quan tâm đến việc ước tính một số giá trị cho một tập hợp. Ví dụ, chúng ta có thể quan tâm đến chiều cao trung bình của một loài thực vật nhất định ở Úc.

Thay vì đi khắp nơi và đo từng cây trong nước, chúng ta có thể thu thập một số mẫu cây nhỏ và đo từng cây một. Sau đó, chúng ta có thể sử dụng chiều cao trung bình của các cây trong mẫu để ước tính chiều cao trung bình của quần thể.

Tuy nhiên, mẫu của chúng ta không có khả năng cung cấp một ước tính hoàn hảo cho dân số. May mắn thay, chúng ta có thể giải thích cho sự không chắc chắn này bằng cách tạo **khoảng tin cậy**, cung cấp một phạm vi giá trị mà chúng ta tin rằng thông số dân số thực nằm trong đó.

Ví dụ: chúng ta có thể tạo ra khoảng tin cậy 95% là $[13,2, 14,8]$, cho biết chúng ta tin tưởng 95% rằng chiều cao trung bình thực sự của loài thực vật này là từ 13,2 inch đến 14,8 inch.

c. Hồi quy

Đôi khi chúng ta quan tâm đến việc hiểu mối quan hệ giữa hai biến trong một tập hợp.

Ví dụ, giả sử chúng ta muốn biết liệu số *giờ học mỗi tuần* có liên quan đến *điểm kiểm tra hay không*. Để trả lời câu hỏi này, chúng ta có thể thực hiện một kỹ thuật được gọi là **phân tích hồi quy**.

Vì vậy, chúng ta có thể quan sát số giờ học cùng với điểm kiểm tra của 100 sinh viên và thực hiện phân tích hồi quy để xem liệu có mối quan hệ đáng kể giữa hai biến hay không. Nếu **giá trị p của hồi quy là có ý nghĩa**, thì chúng ta có thể kết luận rằng có một mối quan hệ đáng kể giữa hai biến này trong tổng thể học sinh.

Sự khác biệt giữa thống kê mô tả và thống kê suy luận

Sự khác biệt giữa thống kê mô tả và thống kê suy luận có thể được mô tả như sau:

Thống kê mô tả sử dụng **thống kê** tóm tắt, đồ thị và bảng để *mô tả* một tập dữ liệu. Điều này rất hữu ích để giúp chúng ta hiểu nhanh chóng và dễ dàng về tập dữ liệu mà không cần xét tất cả các giá trị dữ liệu riêng lẻ.

Thống kê suy luận sử dụng các mẫu để rút ra *suy luận* về các quần thể lớn hơn. Tùy thuộc vào câu hỏi chúng ta muốn trả lời về một tập hợp, có thể quyết định sử dụng một hoặc nhiều phương pháp sau: kiểm định giả thuyết, ước lượng so sánh và phân tích hồi quy. Nếu chúng ta chọn sử dụng một trong những phương pháp này, hãy nhớ rằng **mẫu của chúng ta cần phải đại diện cho dân số đang nghiên cứu**, nếu không các kết luận rút ra sẽ không đáng tin cậy.

Do phạm vi của sách là tập trung minh họa các khả năng của Python trong thống kê, nên chúng tôi không làm rõ các khái niệm cơ bản của thống kê ứng dụng. Có thể tìm hiểu thêm trong các tài liệu sau: [1], [3], [4], [5]

3.2. Một số khái niệm cơ bản

3.2.1. Dân số và Mẫu

Trong thống kê, **dân số** là một tập hợp tất cả các yếu tố hoặc mục mà bạn quan tâm. Các quần thể thường rất lớn, điều này khiến chúng không thích hợp để thu thập và phân tích dữ liệu. Đó là lý do tại sao các nhà thống kê thường cố gắng đưa ra một số kết luận về một quần thể bằng cách chọn và kiểm tra một tập hợp con đại diện của quần thể đó.

Tập hợp con của tổng thể này được gọi là **mẫu**. Lý tưởng nhất là mẫu phải bảo toàn các đặc điểm thống kê thiết yếu của dân số ở mức độ thỏa đáng. Bằng cách đó, ta có thể sử dụng mẫu để thu thập các kết luận về dân số.

3.2.2. Giá trị ngoại lệ

Giá trị **ngoại lệ** là một điểm dữ liệu khác biệt đáng kể với phần lớn dữ liệu được lấy từ một mẫu hoặc tổng thể. Có nhiều nguyên nhân có thể gây ra ngoại lệ, nhưng sau đây là một số nguyên nhân giúp bạn bắt đầu:

- **Sự thay đổi tự nhiên** trong dữ liệu
- **Thay đổi** hành vi của hệ thống được quan sát
- **Lỗi** trong thu thập dữ liệu

Lỗi thu thập dữ liệu là một nguyên nhân đặc biệt nổi bật của các trường hợp ngoại lệ. Ví dụ, những hạn chế của các công cụ hoặc quy trình đo lường dẫn đến thu thập dữ liệu không chính xác. Các lỗi khác có thể do tính toán sai, lỗi do con người, v.v.

Không có một định nghĩa toán học chính xác về các giá trị ngoại lệ. Ta thường phải dựa vào kinh nghiệm, kiến thức về chủ đề đang quan tâm để xác định xem một điểm dữ liệu có phải là ngoại lệ hay không và có cách xử lý nó. Xem lại cách loại bỏ giá trị ngoại lệ ở chương 2.

3.3. Một số đại lượng biểu thị xu hướng trung tâm

3.3.1. Giá trị trung bình

Trung bình **mẫu**, còn được gọi là **trung bình số học mẫu** hoặc đơn giản là **trung bình**, là giá trị trung bình số học của tất cả các mục trong một tập dữ liệu. Bạn có thể

tính toán giá trị trung bình với Python thuần túy, bằng cách sử dụng `sum()` và `len()` không cần nhập thư viện:

```
x = [8.0, 1, 2.5, 4, 28.0]
mean = sum (x) / len (x)
print (mean)
```

Ta cũng có thể sử dụng các hàm thư viện NumPy:

```
import numpy dưới dạng np
x = [8.0, 1, 2.5, 4, 28.0]
mean = np.mean (x)
print (mean)
```

Lưu ý - Nếu có các giá trị `nan` trong dữ liệu (dữ liệu trống), thì giá trị đó `np.mean()` sẽ trả về dưới dạng `nan` ở đầu ra. Do đó ta cần phải xử lý nó. Chúng ta có thể sử dụng một hàm của NumPy như sau:

```
nanmean()
x_with_nan = [8.0, 1, 2.5, math.nan, 4, 28.0]
mean = np.nanmean(x_with_nan)
print (mean)
```

3.3.2. Trung bình có trọng số

Giá trị **trung bình có trọng số**, còn được gọi là **giá trị trung bình số học có trọng số** hoặc **trung bình có trọng số**, là một dạng tổng quát của giá trị trung bình số học cho phép xác định sự đóng góp tương đối của mỗi điểm dữ liệu vào kết quả.

Giá trị trung bình có trọng số rất hữu ích khi cần giá trị trung bình của một tập dữ liệu chứa các mục xuất hiện với tần số khác nhau.

Ví dụ: giả sử ta có một tập hợp trong đó 20% tổng số mục bằng 2, 50% số mục bằng 4 và 30% số mục còn lại bằng 8. Có thể tính giá trị trung bình của như sau:

```
x = [2, 4, 8]
w = [0, 2, 0, 5, 0, 3]
weighted_mean = np.average(x, weights = w)
print (weighted_mean)
```

3.3.3. Median

Trung **vị mẫu** (median) là phần tử giữa của tập dữ liệu đã sắp xếp.

```
x = [8.0, 1, 2.5, 4, 28.0]
median = np.median(x)
print (median)
```

Cũng giống như giá trị trung bình, bạn cũng có thể tìm ra nangiá trị trung bình cho các giá trị trong số dữ liệu của mình.

```
x_with_nan = [8.0, 1, 2.5, math.nan, 4, 28.0]
median = np.nanmedian (x_with_nan)
print (median)
```

3.3.4. Mode

Mode là giá trị trong tập dữ liệu xảy ra thường xuyên nhất. Để tìm **mode**, chúng ta cần sử dụng chức năng thư viện Staticstíc trong Python.

```
import statistics

z = [2, 3, 2, 8, 12]

mode = statistics.mode(z)

print(mode)
```

3.4. Một số đại lượng biểu thị xu hướng phân tán của dữ liệu

Các thước đo của xu hướng trung tâm không đủ để mô tả dữ liệu. Do đó ta cần các **đại lượng đo về sự thay đổi** để định lượng mức độ phân tán của các điểm dữ liệu.

3.4.1. Phương sai

Phương **sai mẫu** định lượng mức độ phân tán của dữ liệu. Phương sai biểu thị mức độ phân tán của các điểm dữ liệu quanh giá trị trung bình.

$$var = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n} \quad \text{Phương sai tổng thể}$$

$$var = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1} \quad \text{Phương sai mẫu}$$

Với \bar{x} là giá trị trung bình

```
x = [8.0, 1, 2.5, 4, 28.0]
mean = statistics.mean (x)
variance = statistics.variance (x, mean)
print (mean)
```

```
print(variance)

8.7
123.2
```

3.4.2. Độ lệch chuẩn

Độ lệch chuẩn được tính theo công thức sau:

$$s = \sqrt{var}$$

```
x = [8.0, 1, 2.5, 4, 28.0]
mean = statistics.mean(x)
variance = statistics.variance(x, mean)
standard_deversion = variance ** 0,5
in (standard_deversion)
```

Ta cũng có thể sử dụng hàm `statistics.stdev()` để tính:

```
x = [8.0, 1, 2.5, 4, 28.0]

standard_deviation = statistics.stdev(x)

print (standard_deviation)
```

3.4.3. Sai số tiêu chuẩn (standard error)

Sai số tiêu chuẩn là ước lượng độ lệch chuẩn của một hệ số. Chúng ta càng có nhiều điểm dữ liệu để ước tính giá trị trung bình, thì ước tính giá trị trung bình của chúng ta càng trở nên tốt hơn.

Công thức tính:

$$SEM = \frac{s}{\sqrt{n}}$$

Ở đây SEM: sai số tiêu chuẩn; S = Độ lệch chuẩn và n = độ lớn của mẫu.

Có thể tính SEM theo công thức dựa trên hàm tính độ lệch chuẩn `std()` với tham số `ddof=1` và của căn bậc 2 `sqrt()` numpy, hay sử dụng hàm `sem()` của `scipy`.

```
from scipy.stats import sem

#define dataset
data = [3, 4, 4, 5, 7, 8, 12, 14, 14, 15, 17, 19, 22, 24, 24,
24, 25, 28, 28, 29]

#calculate standard error of the mean
```

```
sem(data)
```

```
2.001447
```

Một số nhận xét về sai số tiêu chuẩn

Sai số tiêu chuẩn càng lớn, thì càng có nhiều giá trị phân tán rộng xung quanh giá trị trung bình trong tập dữ liệu.

Khi kích thước mẫu tăng lên, sai số chuẩn của giá trị trung bình có xu hướng giảm.

3.4.4. Đại lượng Skewness & Kurtosis

Trong thống kê, **Skewness** và **Kurtosis** hai cách để đo hình dạng của một phân phối.

Skewness là thước đo sự đối xứng của một phân phối. Giá trị này có thể dương hoặc âm.

- Một **Skewness** âm cho biết rằng phần đuôi nằm ở phía bên trái của phân phối, mở rộng về phía nhiều giá trị âm hơn.
- Một **Skewness** dương chỉ ra rằng phần đuôi nằm ở phía bên phải của phân phối, mở rộng về phía các giá trị tích cực hơn.
- Giá trị bằng 0 chỉ ra rằng không có độ lệch nào trong phân phối, có nghĩa là phân phối hoàn toàn đối xứng.

Kurtosis là một thước đo để đánh giá xem phân phối có đuôi nặng (heavy-tailed) hay đuôi nhẹ (light-tailed) so với **phân phối chuẩn hay không**.

- Hệ số kurtosis của một phân phối chuẩn là 3.
- Nếu một phân phối nhất định có kurtosis nhỏ hơn 3, nó được cho là *platykurtic*, có nghĩa là nó có xu hướng tạo ra ít hơn và ít ngoại lệ cực đoan hơn so với phân phối chuẩn.
- Nếu một phân phối nhất định có kurtosis lớn hơn 3, nó được cho là *leptokurtic*, có nghĩa là nó có xu hướng tạo ra nhiều giá trị ngoại lai hơn so với phân phối chuẩn.

Lưu ý: Một số công thức, sẽ trừ đi 3 từ giá trị kurtosis để dễ so sánh với phân phối chuẩn. Sử dụng định nghĩa này, một phân phối sẽ có kurtosis lớn hơn phân phối chuẩn nếu nó có giá trị kurtosis lớn hơn 0.

Để tính toán độ lệch mẫu và độ lệch mẫu của tập dữ liệu này, chúng ta có thể sử dụng các hàm **skew()** and **kurt()** của thư viện Scipy

- **skew(array of values, bias=False)**

- **kurt(array of values, bias=False)**

Sử dụng đối số **bias = False** để tính toán độ lệch mẫu và độ lệch mẫu trái ngược với độ lệch tổng thể và độ lệch kurtosis.

Giả sử chúng ta có tập dữ liệu sau:

```
data = [88, 85, 82, 97, 67, 77, 74, 86, 81, 95, 77, 88,
85, 76, 81]
#calculate sample skewness
skew(data, bias=False)

0.032697

#calculate sample kurtosis
kurtosis(data, bias=False)
```

3.4.5. Bách phân vị (Percentile)

Trong thống kê mô tả, việc sử dụng giá trị [bách phân vị là phương pháp để ước tính](#) tỷ lệ dữ liệu trong một tập số liệu rơi vào vùng cao hơn hoặc vùng thấp hơn một giá trị cho trước. Số phân vị P^{th} là một giá trị mà tại đó nhiều nhất có $P\%$ số trường hợp quan sát trong tập dữ liệu có giá trị thấp hơn giá trị này và nhiều nhất là $(100 - P)\%$ của trường hợp có giá trị lớn hơn giá trị này.

Bách phân vị 10^{th} là giá trị mà tại đó nhiều nhất là 10% số quan sát là kém hơn giá trị này

Bách phân vị 25^{th} là giá trị mà tại đó nhiều nhất là 25% số quan sát là kém hơn giá trị này. Bách phân vị này còn được gọi là phần tư thứ nhất $Q1$.

Bách phân vị 50^{th} là giá trị mà tại đó nhiều nhất là 50% số quan sát là kém hơn giá trị này

Bách phân vị 75^{th} là giá trị mà tại đó nhiều nhất là 75% số quan sát là kém hơn giá trị này. Bách phân vị này còn được gọi là phần tư thứ ba $Q3$.

Bách phân vị 90^{th} là giá trị mà tại đó nhiều nhất là 90% số quan sát là kém hơn giá trị này

Bách phân vị 50^{th} chính là trung vị (median)

Khoảng cách giữa $Q1$ và $Q3$ được gọi là khoảng tứ phân vị (IQR). $IQR = Q3 - Q1$

Median, $Q3$, $Q1$ chúng ta đã gặp trong phần biểu đồ boxplot.

Có thể tính bách phân vị trong Python bằng cách sử dụng hàm [numpy.percentile\(\)](#), sử dụng cú pháp sau:

numpy.percentile (a, q)

- **a:** Mảng giá trị
- **q:** Phần trăm hoặc mảng các phần trăm để tính toán, các giá trị q phải nằm trong khoảng từ 0 đến 100.

Cách tìm bách phân vị của một mảng (xem [6], [7])

```
import numpy as np

#make this example reproducible
np.random.seed(0)

#create array of 100 random integers distributed between 0 and
500
data = np.random.randint(0, 500, 100)

#find the 37th percentile of the array
np.percentile(data, 37)

173.26

#Find the quartiles (25th, 50th, and 75th percentiles) of the
array
np.percentile(data, [25, 50, 75])

array([116.5, 243.5, 371.5])
```

Cách tìm bách phân vị của một cột DataFrame:

```
import numpy as np
import pandas as pd

#create DataFrame
df = pd.DataFrame({'var1': [25, 12, 15, 14, 19, 23, 25, 29,
33, 35],
                    'var2': [5, 7, 7, 9, 12, 9, 9, 4, 14, 15],
                    'var3': [11, 8, 10, 6, 6, 5, 9, 12, 13,
16]})

#find 90th percentile of var1 column
np.percentile(df.var1, 95)
#find 95th percentile of each column
df.quantile(.95)
#find 95th percentile of just columns var1 and var2
df[['var1', 'var2']].quantile(.95)
```

3.4.6. Phạm vi

Phạm vi chỉ đơn giản là sự khác biệt giữa giá trị dữ liệu cao nhất và thấp nhất có trong mẫu. Có thể sử dụng phương thức ptp của numpy để tính: np.ptp()

```
ages =  
[5, 31, 43, 48, 50, 41, 7, 11, 15, 39, 80, 82, 32, 2, 8, 6, 25, 36, 27, 61, 31]  
x = np.ptp(ages)  
print(x)
```

Sử dụng thư viện SciPy và Pandas chúng ta có thể thực hiện các số liệu thống kê mô tả cơ bản trên khá đơn giản. Chẳng hạn với scipy, ta chỉ cần gọi thực hiện scipy.stats.describe () như sau:

```
x =  
[5, 31, 43, 48, 50, 41, 7, 11, 15, 39, 80, 82, 32, 2, 8, 6, 25, 36, 27, 61, 31]  
result = scipy.stats.describe(x, ddof=1, bias=False)  
print(result)  
  
Output:  
DescribeResult(nobs=21, minmax=(2, 82),  
mean=32.38095238095238, variance=540.047619047619,  
skewness=0.6679020341687003, kurtosis=-  
0.029954628545992623)
```

3.5. Phương thức describe() cho thống kê mô tả đơn giản

Nhìn chung phương thức describe() của scipy trả về một đối tượng chứa các thống kê mô tả sau:

nobs: số lượng quan sát hoặc phần tử trong tập dữ liệu
minmax: bộ dữ liệu với các giá trị tối thiểu và tối đa của tập dữ liệu
mean: trung bình của tập dữ liệu
variance: phương sai của tập dữ liệu
skewness: độ lệch skewness của tập dữ liệu
kurtosis: kurtosis của tập dữ liệu

Chúng ta cũng có thể truy cập từng giá trị cụ thể trên như sau:

```
x =  
[5, 31, 43, 48, 50, 41, 7, 11, 15, 39, 80, 82, 32, 2, 8, 6, 25, 36, 27, 61, 31]  
result = scipy.stats.describe (x, ddof = 1, bias = False)
```

```
print (result.nobs)
print (result.minmax [0])
print (result.minmax [1])
print (result.variance)
print (result.skewness)
print (result.kurtosis)
```

Out:

```
21
2
82
540.047619047619
0,6679020341687003
-0,029954628545992623
```

Chúng ta cũng có thể sử dụng thư viện pandas để tính các giá trị trên, ví dụ:

```
x =
[5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]
z = pd.series(x)
result = z.describe()
print(result)
```

Output:

```
count      21.000000
mean       32.380952
std        23.238925
min         2.000000
25%        11.000000
50%        31.000000
75%        43.000000
max        82.000000
dtype: float64
```

Kết quả của phương thức describe() trả về một series mới gồm:

count: số phần tử trong tập dữ liệu

mean: trung bình của tập dữ liệu

std: độ lệch chuẩn của tập dữ liệu

min và max: giá trị tối thiểu và tối đa của tập dữ liệu

25%, 50% và 75%: các phân vị phần tư của tập dữ liệu

3.6. Tính điểm Z

Điểm Z và điểm T được sử dụng trong thống kê và được gọi là điểm tiêu chuẩn

Z-score là gì ?

Trong thống kê, Z-score (điểm số Z hay điểm số chuẩn) của một quan sát là số độ lệch chuẩn mà nó nằm trên hoặc dưới mức trung bình của tổng thể.

<https://www.pydev.vn/d/43-phan-phoi-t-la-gi-dinh-nghia-phan-phoi-t>

Trong thống kê, **điểm số z** cho chúng ta biết có bao nhiêu độ lệch chuẩn so với giá trị so **với** giá trị **trung bình**. Công thức tính điểm z:

$$z = (X - \mu) / \sigma$$

ở đây:

- X là một giá trị dữ liệu thô
- μ là giá trị trung bình
- σ là độ lệch chuẩn

Minh hoạ cách tính điểm Z với Python

Chúng ta có **thể** tính điểm z trong Python bằng cách sử dụng **scipy.stats.zscore**, sử dụng cú pháp sau:

scipy.stats.zscore(a, axis=0, ddof=0, nan_policy='propagate')

Ở đây:

- **a**: mảng chứa dữ liệu
- **axis**: chỉ ra chiều của mảng để tính z - score. Mặc định là 0, để tính theo hàng.
- **ddof**: Bậc tự do trong tính toán độ lệch chuẩn. Mặc định là 0.
- **nan_policy**: Cho phép bỏ qua các dữ liệu thiếu.

Minh hoạ cách tính cho mảng một chiều tạo bởi thư viện Numpy

Bước 1: Tạo một mảng giá trị.

```
data = np.array ([6, 7, 7, 12, 13, 13, 15, 16, 19, 22])
```

Bước 2: Tính điểm z cho mỗi giá trị trong mảng.

```
stats.zscore(data)
```

```
[-1.394, -1.195, -1.195, -0.199, 0, 0, 0.398, 0.598,
 1.195, 1.793]
```

Trong ví dụ trên:

- Giá trị đầu tiên là z – score của “6” trong mảng bằng **1.394** độ lệch chuẩn *thấp hơn* giá trị trung bình.
- Giá trị thứ năm là z – score của “13” trong mảng bằng **0** độ lệch chuẩn so với giá trị trung bình, tức là nó bằng với giá trị trung bình.
- Giá trị cuối cùng là z – score của “22” trong mảng là **1,793** độ lệch chuẩn lớn hơn so với giá trị trung bình.

Tính z score cho mảng đa chiều

Nếu chúng ta có một mảng nhiều chiều, chúng ta có thể sử dụng tham số **trục** để chỉ định rằng chúng ta muốn tính từng điểm z liên quan đến mảng của chính nó. Ví dụ: giả sử chúng ta có mảng đa chiều sau:

```
data = np.array ([[5, 6, 7, 7, 8],
                  [8, 8, 8, 9, 9],
                  [2, 2, 4, 4, 5]])
```

Chúng ta có thể sử dụng cú pháp sau để tính điểm z cho mỗi mảng:

```
stats.zscore(data, axis=1)
```

```
[[-1.569 -0.588 0.392 0.392 1.373]
 [-0.816 -0.816 -0.816 1.225 1.225]
 [-1.167 -1.167 0.5 0.5 1.333]]
```

- Giá trị đầu tiên của “5” trong mảng đầu tiên là **1,159** độ lệch chuẩn *thấp hơn* giá trị trung bình của mảng.
- Giá trị đầu tiên của “8” trong mảng thứ hai là **0,816** độ lệch chuẩn *thấp hơn* giá trị trung bình của mảng.
- Giá trị đầu tiên của “2” trong mảng thứ ba là **1,167** độ lệch chuẩn *thấp hơn* giá trị trung bình của mảng.

Với 1 dataframe

Chúng ta áp dụng phương thức `apply()` theo mẫu sau:

```
data.apply(stats.zscore)
```

```
data = pd.DataFrame(np.random.randint(0, 10, size=(5, 3)), columns=['A', 'B', 'C'])
```

	A	B	C
0	8	0	9
1	4	0	7
2	9	6	8
3	1	8	1
4	8	0	8


```
data.apply(stats.zscore) # tính z - score cho tất cả các cột của dataframe data
```

	A	B	C
0	0.659380	-0.802955	0.836080
1	-0.659380	-0.802955	0.139347
2	0.989071	0.917663	0.487713
3	-1.648451	1.491202	-1.950852
4	0.659380	-0.802955	0.487713

Tóm tắt chương 3

Chương 3, giới thiệu sơ lược các bài toán cơ bản của thống kê ứng dụng. Các đại lượng cơ bản của thống kê như xu hướng trung tâm, xu hướng phân tán của dữ liệu thống kê được trình bày cùng với một số minh họa tính toán với Python.

Câu hỏi ôn tập và bài tập chương 3

1. Hãy làm rõ đối tượng nghiên cứu của thống kê học
2. So sánh thống kê mô tả và thống kê suy luận
3. Thực hiện cài đặt các ví dụ minh họa trong chương, báo cáo kết quả thực hiện.
4. Giải thích các câu lệnh cơ bản để thực hiện trong các đoạn mã lệnh của các ví dụ
5. Tìm hiểu các công thức tính của các đại lượng thống kê, để cài đặt tính toán các đại lượng đó, mà không sử dụng các hàm có sẵn của các thư viện.
6. Thực hiện các ví dụ cho các dữ liệu có tính thực tiễn khác.
7. Kết nối các ví dụ để thực hiện tính toán các đại lượng thống kê cơ bản trên dataframe cho trong file excel `ngheesi.xlsx` ở phần phụ lục và sẵn có trên link sau:

<https://docs.google.com/spreadsheets/d/1q7RVfoTKEuLP0Ud0a50mJu7e1-YJJbUe/edit?usp=sharing&ouid=106843557461060771783&rtpof=true&sd=true>

Chương 4. LẤY MẪU DỮ LIỆU VÀ LẬP BẢNG THỐNG KÊ

Mục đích

Chương này nhằm trình bày các bước cơ bản về lấy mẫu dữ liệu, phân tổ và lập bảng thống kê. Đây là các bước cơ bản của thống kê mô tả.

Yêu cầu

So sánh các cách lấy mẫu từ tổng thể dân số.

Trình bày ý nghĩa của việc lập bảng thống kê

Cài đặt thực hiện được các ví dụ minh họa về lấy mẫu và lập bảng phân tổ

4.1. Lấy mẫu theo cụm

Các nhà nghiên cứu thường lấy **mẫu** từ một quần thể và sử dụng dữ liệu từ mẫu để đưa ra kết luận về tổng thể. Một phương pháp lấy mẫu thường được sử dụng là lấy mẫu theo **cụm**, trong đó một quần thể được chia thành các cụm và tất cả các thành viên của *một số* cụm được chọn để đưa vào mẫu.

Ví dụ:

Giả sử một công ty cung cấp các chuyến tham quan thành phố, muốn khảo sát khách hàng của mình. Trong số mười chuyến du lịch mà họ đưa ra, chọn ngẫu nhiên bốn chuyến du lịch và yêu cầu mọi khách hàng đánh giá trải nghiệm của họ trên thang điểm từ 1 đến 10. (xem [6], [7])

```
import pandas as pd
import numpy as np

#make this example reproducible
np.random.seed(0)

#create DataFrame
df = pd.DataFrame({'tour': np.repeat(np.arange(1,11), 20),
                  'experience': np.random.normal(loc=7,
                  scale=1, size=200)})
```

Đoạn mã sau đây cho thấy cách có được một mẫu khách hàng bằng cách chọn ngẫu nhiên bốn chuyến tham quan và đưa mọi thành viên trong các chuyến tham quan đó vào mẫu:

```
#randomly choose 4 tour groups out of the 10
clusters = np.random.choice(np.arange(1,11), size=4,
replace=False)

#define sample as all members who belong to one of the 4 tour
groups
cluster_sample = df[df['tour'].isin(clusters)]

#find how many observations came from each tour group
cluster_sample['tour'].value_counts()

[ 1   6  10   8] # 4 cluster được chọn là 1, 6, 10, 8
1      20
6      20
8      20
10     20
Name: tour, dtype: int64
```

- 20 khách hàng từ nhóm du lịch số 1 đã được đưa vào mẫu.
- 20 khách hàng từ nhóm du lịch số 6 đã được đưa vào mẫu.
- 20 khách hàng từ nhóm du lịch số 8 đã được đưa vào mẫu.
- 20 khách hàng từ nhóm du lịch số 10 đã được đưa vào mẫu.

Phương thức `isin(clusters)` để lấy các hàng của dataframe thuộc về cluster.

Như vậy, mẫu này bao gồm tổng số 80 khách hàng đến từ 4 nhóm du lịch khác nhau.

Lấy mẫu phân tầng

Một phương pháp lấy mẫu thường được sử dụng là lấy **mẫu ngẫu nhiên phân tầng**, trong đó một quần thể được chia thành các nhóm và một số thành viên nhất định từ mỗi nhóm được chọn ngẫu nhiên để đưa vào mẫu.

Ví dụ 1: Lấy mẫu phân tầng sử dụng số lượng (xem [6], [7])

Giả sử chúng ta có DataFrame sau đây chứa dữ liệu về 8 cầu thủ bóng rổ ở 2 đội khác nhau:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'team': ['A', 'A', 'A', 'A', 'B', 'B', 'B',
'B'],
                  'position': ['G', 'G', 'F', 'G', 'F', 'F', 'C',
'C'],
                  'assists': [5, 7, 7, 8, 5, 7, 6, 9],
                  'rebounds': [11, 8, 10, 6, 6, 9, 6, 10]})

#view DataFrame
df
```

team	position	assists	rebounds
A	G	5	11
A	G	7	8
A	F	7	10
A	G	8	6
B	F	5	6
B	F	7	9
B	F	6	6
B	C	9	10

0	A	G	5	11
1	A	G	7	8
2	A	F	7	10
3	A	G	8	6
4	B	F	5	6
5	B	F	7	9
6	B	C	6	6
7	B	C	9	10

Đoạn mã sau thực hiện lấy mẫu ngẫu nhiên phân tầng bằng cách chọn ngẫu nhiên 2 người chơi từ mỗi đội để đưa vào mẫu:

```
df.groupby('team', group_keys=False).apply(lambda x:
x.sample(2))
```

	team	position	assists	rebounds
0	A	G	5	11
3	A	G	8	6
6	B	C	6	6
5	B	F	7	9

Trong ví dụ trên chúng ta đã gộp nhóm theo đội và mỗi đội đã áp dụng phép lấy mẫu (sample) mỗi mẫu lấy ngẫu nhiên 2 phần tử.

Ví dụ 2: Lấy mẫu phân tầng sử dụng tỷ lệ

Cũng với DataFrame tương tự ở ví dụ 1, chứa dữ liệu về 8 cầu thủ bóng rổ ở 2 đội khác nhau:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'team': ['A', 'A', 'B', 'B', 'B', 'B', 'B',
'B'],
                  'position': ['G', 'G', 'F', 'G', 'F', 'F', 'C',
'C'],
                  'assists': [5, 7, 7, 8, 5, 7, 6, 9],
                  'rebounds': [11, 8, 10, 6, 6, 9, 6, 10]})

#view DataFrame
df
```

	team	position	assists	rebounds
0	A	G	5	11
1	A	G	7	8
2	A	F	7	10
3	A	G	8	6
4	B	F	5	6
5	B	F	7	9
6	B	C	6	6
7	B	C	9	10

Lưu ý rằng có 6 trong số 8 người chơi (75%) trong DataFrame thuộc đội A và 2 trong số 8 người chơi (25%) thuộc đội B.

Đoạn mã thực hiện lấy mẫu ngẫu nhiên phân tầng sao cho tỷ lệ người chơi trong mẫu từ mỗi đội khớp với tỷ lệ người chơi từ mỗi đội trong DataFrame đã cho: (xem [6], [7])

```
import numpy as np

#define total sample size desired
N = 4

#perform stratified random sampling
df.groupby('team', group_keys=False).apply(lambda x:
x.sample(int(np rint(N*len(x)/len(df))))).sample(frac=1).reset_inde
x(drop=True)
```

	team	position	assists	rebounds
0	B	F	7	9
1	B	G	8	6
2	B	C	6	6
3	A	G	7	8

Lưu ý rằng tỷ lệ người chơi từ đội A trong mẫu phân tầng (25%) khớp với tỷ lệ người chơi từ đội A và tỷ lệ người chơi từ đội B trong mẫu phân tầng (75%) khớp với tỷ lệ người chơi từ đội B trong DataFrame ban đầu.

4.2. Lấy mẫu có hệ thống

Một phương pháp lấy mẫu thường được sử dụng là lấy mẫu theo **hệ thống**, được thực hiện với quy trình hai bước đơn giản:

1. Mỗi thành viên của một quần thể được gán một số thứ tự.
2. Chọn một điểm xuất phát ngẫu nhiên và chọn mọi thành viên thứ n để đưa vào mẫu.

Ví dụ: Lấy mẫu có hệ thống

Giả sử một giáo viên muốn lấy mẫu gồm 100 học sinh từ một trường có tổng số học sinh là 500. Giáo viên chọn sử dụng phương pháp lấy mẫu có hệ thống, trong đó từng học sinh được xếp theo thứ tự bảng chữ cái theo họ của học sinh, chọn ngẫu nhiên một điểm xuất phát và chọn mọi học sinh thứ 5 vào mẫu. (xem [6], [7])

```
import pandas as pd
import numpy as np
import string
import random
```

```
#make this example reproducible
np.random.seed(0)

#create simple function to generate random last names
def randomNames(size=6, chars=string.ascii_uppercase):
    return ''.join(random.choice(chars) for _ in range(size))

#create DataFrame
df = pd.DataFrame({'last_name': [randomNames() for _ in
range(500)],
                    'GPA': np.random.normal(loc=85, scale=3,
size=500)})

#view first six rows of DataFrame
df.head()
```

	last_name	GPA
0	PXGPIV	86.667888
1	JKRRQI	87.677422
2	TRIZTC	83.733056
3	YHUGIN	85.314142
4	ZVUNVK	85.684160

đoạn mã sau đây chỉ ra cách lấy mẫu gồm 100 sinh viên thông qua lấy mẫu có hệ thống:

```
#obtain systematic sample by selecting every 5th row
sys_sample_df = df.iloc[::5]

#view first six rows of DataFrame
sys_sample_df.head()
```

	last_name	gpa
3	ORJFW	88.78065
8	RWPSB	81.96988
13	RACZU	79.21433
18	ZOHKA	80.47246
23	QJETK	87.09991
28	JTHWB	83.87300

```
#view dimensions of data frame
sys_sample_df.shape

(100, 2)
```

Trong đoạn mã trên lưu ý đến câu lệnh:

```
sys_sample_df = df.iloc[::5]
```

Câu lệnh này sẽ chọn từ phần tử đầu tiên, tiếp đó cứ 5 phần tử chọn 1 phần tử để đưa vào dataframe **sys_sample_df**. Do đó từ 500 phần tử của df ta thu được 100 phần tử được chọn đưa vào mẫu **sys_sample_df**

Sử dụng **shape ()**, chúng ta có thể thấy rằng mẫu hệ thống mà chúng ta thu được là một khung dữ liệu với 100 hàng và 2 cột.

4.3. Tạo bảng tần số và tần suất

Bảng phân phối tần số & Tần suất

Trong nhiều trường hợp, giá trị của một biến nào đó có sự lặp lại. Mặt khác khi ta quan tâm không chỉ là phần tử nào có giá trị là bao nhiêu mà ta còn muốn tìm hiểu có bao nhiêu phần tử có giá trị đã cho (phân phối của biến). Trong trường hợp đó bảng phân phối tần số và/hay tần suất có thể được sử dụng.

Bảng **tần số** là một bảng hiển thị tần số của các danh mục khác nhau. Bảng đặc biệt hữu ích để thấy được sự phân bố của các giá trị trong một tập dữ liệu.

Để tìm tần số của các giá trị riêng lẻ của 1 series, có thể sử dụng hàm **value_counts()** của thư viện pandas:

```
import pandas as pd

#define Series
data = pd.Series([1, 1, 1, 2, 3, 3, 3, 3, 4, 4, 5])

#find frequencies of each value
print(data.value_counts())
3      4
1      3
4      2
5      1
2      1
```

Bảng tần suất một chiều cho DataFrame

Để tìm tần số của DataFrame, bạn có thể sử dụng hàm **crosstab()** của thư viện pandas, với cú pháp sau:

crosstab(index, columns)

Ở đây:

- **index:** tên của cột được gộp nhóm
- **columns:** tên để đặt cho cột tần số

Ví dụ: giả sử chúng ta có DataFrame với thông tin về xếp loại (Grade) , tuổi (Age) và giới tính (Gender) của 10 học sinh khác nhau trong một lớp học. Dưới đây là cách tìm tần số cho từng loại lớp:

```
#create data
df = pd.DataFrame({'Grade': ['A','A','A','B','B','B','B','C','D','D'],
                   'Age': [18, 18, 18, 19, 19, 20, 18, 18, 19, 19],
                   'Gender': ['M','M','F','F','F','M','M','F','M','F']})
#find frequency of each letter grade
#find frequency of each letter grade
pd.crosstab(index=df['Grade'], columns='count')
```

col_0	count
Grade	
A	3
B	4
C	1
D	2

- 3 học sinh nhận được loại 'A' trong lớp.
- 4 học sinh nhận được loại 'B' trong lớp.
- 1 học sinh nhận được loại 'C' trong lớp.
- 2 học sinh nhận được loại 'D' trong lớp.

```
pd.crosstab(index=df['Age'], columns='count')
```

col_0	count
Age	
18	5
19	4
20	1

- 5 học sinh 18 tuổi.
- 4 học sinh 19 tuổi.
- 1 sinh viên 20 tuổi.

Ta cũng có thể dễ dàng hiển thị các tần suất dưới dạng tỷ lệ của toàn bộ tập dữ liệu, bằng cách lấy tần số chia cho tổng số phần tử:

```
#define crosstab
tab = pd.crosstab(index=df['Age'], columns='count')

#find proportions
tab/tab.sum()
col_0 count
```

Age	
18	0.5
19	0.4
20	0.1

- 50% học sinh 18 tuổi.
- 40% sinh viên 19 tuổi.
- 10% sinh viên 20 tuổi.

Bảng tần số hai chiều cho dataframe

Cũng có thể tạo bảng tần số hai chiều để hiển thị tần số cho hai biến khác nhau trong tập dữ liệu.

Ví dụ: Tạo bảng tần số hai chiều cho các biến Tuổi và Xếp loại

```
pd.crosstab(index=df['Age'], columns=df['Grade'])
```

Grade	A	B	C	D
Age				
18	3	1	1	0
19	0	2	0	2
20	0	1	0	0

- Có **3** học sinh 18 tuổi được loại A trong lớp.
- Có **1** học sinh 18 tuổi đạt loại 'B' trong lớp.
- Có **1** học sinh 18 tuổi đạt loại 'C' trong lớp.
- Có **0** học sinh 18 tuổi và được loại 'D' trong lớp.

Có thể tìm thấy những tài liệu đầy đủ cho **crosstab ()** tại:

<https://pandas.pydata.org/pandasdocs/stable/reference/api/pandas.crosstab.html>

Bảng tham chiếu chéo (crosstab hay contingency table) thường được dùng trong khảo sát các biến định tính hoặc định lượng nhưng đã được phân nhóm, hoặc thuộc loại rời rạc. Trong bảng tham chiếu chéo (Bảng 1) :

- cột tiêu đề ghi các giá trị A_i của biến A,
- dòng tiêu đề ghi các giá trị B_j của biến B,
- ô ij ghi số phân tử f_{ij} có giá trị của biến A là A_i và biến B là B_j

Bảng 1 Cấu trúc bảng tham chiếu chéo					
	Biến B				
	B_1	B_2	...	B_j	...

Biến A	A ₁	f ₁₁	f ₁₂	...	f _{1j}	...
	A ₂	f ₂₁	f ₂₂	...	f _{2j}	...

	A _i	f _{i1}	f _{i2}	...	f _{ij}	...

4.4. Chuẩn hóa dữ liệu

Trong thống kê và học máy, chúng ta cần **chuẩn hóa** các biến sao cho phạm vi giá trị nằm trong khoảng từ 0 đến 1.

Lý do phổ biến nhất để chuẩn hóa các biến là khi chúng ta tiến hành một số loại phân tích đa biến (khi chúng ta muốn hiểu mối quan hệ giữa một số biến dự báo và một biến phản hồi) và chúng ta muốn mỗi biến đóng góp như nhau vào phân tích.

Khi các biến được đo lường ở các thang đo khác nhau, chúng thường không đóng góp như nhau vào phân tích. Ví dụ: nếu giá trị của một biến nằm trong khoảng từ 0 đến 100000 và giá trị của một biến khác nằm trong khoảng từ 0 đến 100, thì biến có phạm vi lớn hơn sẽ có trọng số lớn hơn trong phân tích.

Bằng cách chuẩn hóa các biến, chúng ta có thể chắc chắn rằng mỗi biến đều đóng góp như nhau vào phân tích.

Để chuẩn hóa các giá trị từ 0 đến 1, chúng ta có thể sử dụng công thức sau:

$$X_{\text{norm}} = (X_i - X_{\text{min}}) / (X_{\text{max}} - X_{\text{min}})$$

ở đây:

- **X_{norm}**: giá trị chuẩn hoá của biến thứ i^{th}
- **x_i**: Giá trị của biến thứ i^{th}
- **X_{max}**: Giá trị nhỏ nhất trong dữ liệu
- **X_{min}**: Giá trị lớn nhất trong dữ liệu

Ví dụ 1: Chuẩn hóa một mảng

```
import numpy as np

#create NumPy array
data = np.array([[13, 16, 19, 22, 23, 38, 47, 56, 58, 63, 65,
70, 71]])

#normalize all values in array
data_norm = (data - data.min()) / (data.max() - data.min())
```

```
#view normalized values
data_norm

array([[0.          , 0.05172414, 0.10344828, 0.15517241,
        0.17241379,
        0.43103448, 0.5862069 , 0.74137931, 0.77586207,
        0.86206897,
        0.89655172, 0.98275862, 1.          ]])
```

Tương tự ví dụ sau nhằm chuẩn hoá dữ liệu trong 1 dataframe.

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'points': [25, 12, 15, 14, 19, 23, 25, 29],
                   'assists': [5, 7, 7, 9, 12, 9, 9, 4],
                   'rebounds': [11, 8, 10, 6, 6, 5, 9, 12]})

#normalize values in every column
df_norm = (df-df.min())/ (df.max() - df.min())
```

Ví dụ 2: Chuẩn hóa các biến cụ thể trong DataFrame (xem [6], [7])

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'points': [25, 12, 15, 14, 19, 23, 25, 29],
                   'assists': [5, 7, 7, 9, 12, 9, 9, 4],
                   'rebounds': [11, 8, 10, 6, 6, 5, 9, 12]})

df_norm = (df-df.min())/ (df.max() - df.min())

define columns to normalize
x = df.iloc[:,0:2]

#normalize values in first two columns only
df.iloc[:,0:2] = (x-x.min())/ (x.max() - x.min())

#view normalized DataFrame
df
```

	points	assists	rebounds
0	0.764706	0.125	11
1	0.000000	0.375	8
2	0.176471	0.375	10
3	0.117647	0.625	6
4	0.411765	1.000	6
5	0.647059	0.625	5
6	0.764706	0.625	9

7	1.000000	0.000	12
---	----------	-------	----

4.5. Tính tần suất tương đối

Tần suất tương đối nhằm đo **tần suất** một giá trị nhất định xuất hiện trong tập dữ liệu *so* với tổng số giá trị trong tập dữ liệu.

Ta có thể sử dụng hàm tự xây dựng sau trong Python để tính tần số tương đối:

```
def rel_freq(x):
    freqs = [(value, x.count(value) / len(x)) for value in
set(x)]
    return freqs
```

Ví dụ 1: Tần suất tương đối cho một danh sách các số

Đoạn mã sau đây cho thấy cách sử dụng hàm này để tính tần số tương đối cho một danh sách các số:

```
#define data
data = [1, 1, 1, 2, 3, 4, 4]

#calculate relative frequencies for each value in list
rel_freq(data)

[(1, 0.42857142857142855),
 (2, 0.14285714285714285),
 (3, 0.14285714285714285),
 (4, 0.2857142857142857)]
```

- Giá trị “1” có tần suất tương đối là **0,42857** trong tập dữ liệu.
- Giá trị “2” có tần suất tương đối là **0,142857** trong tập dữ liệu.
- Giá trị “3” có tần suất tương đối là **0,142857** trong tập dữ liệu.
- Giá trị “4” có tần suất tương đối là **0,28571** trong tập dữ liệu.

Ví dụ 2: Tần suất tương đối cho một danh sách các ký tự

Đoạn mã sau chỉ ra cách sử dụng hàm **rel_freq()** để tính tần số tương đối cho danh sách các ký tự:

```
#define data
data = ['a', 'a', 'b', 'b', 'c'] # tính tần số tương đối cho
từng giá trị trong danh sách
rel_freq (data)
```

```
[('a', 0.4), ('b', 0.4), ('c', 0.2)]
```

- Giá trị “a” có tần suất tương đối là **0,4** trong tập dữ liệu.
- Giá trị “b” có tần suất tương đối là **0,4** trong tập dữ liệu.
- Giá trị “c” có tần suất tương đối là **0,2** trong tập dữ liệu.

Ví dụ 3: Tần suất tương đối cho một cột trong DataFrame

Đoạn mã sau đây cho thấy cách sử dụng hàm **rel_freq()** ở trên để tính tần số tương đối cho một cột cụ thể trong DataFrame

```
import pandas as pd

#define data
data = pd.DataFrame({'A': [25, 15, 15, 14, 19],
                      'B': [5, 7, 7, 9, 12],
                      'C': [11, 8, 10, 6, 6]})

#calculate relative frequencies of values in column 'A'
rel_freq(list(data['A']))

[(25, 0.2), (19, 0.2), (14, 0.2), (15, 0.4)]
```

- Giá trị “25” có tần suất tương đối trong cột là **0,2**.
- Giá trị “19” có tần suất tương đối trong cột là **0,2**.
- Giá trị “14” có tần suất tương đối trong cột là **0,2**.
- Giá trị “15” có tần suất tương đối là **0,4** trong cột.

Tóm tắt chương 4

Chương này trình bày các vấn đề liên quan đến lấy mẫu dữ liệu, phân tổ và lập bảng thống kê là các bước cơ bản của thống kê mô tả và là cơ sở cho thống kê suy luận, hay các bài toán kiểm định thống kê.

Câu hỏi ôn tập và bài tập chương 4

1. Hãy làm rõ ý nghĩa của các bảng thống kê
2. So sánh các cách lấy mẫu
3. Thực hiện cài đặt các ví dụ minh họa trong chương, báo cáo kết quả thực hiện.

4. Giải thích các câu lệnh cơ bản để thực hiện trong các đoạn mã lệnh của các ví dụ

5. Thực hiện các ví dụ cho các dữ liệu có tính thực tiễn khác.

6. Kết nối các ví dụ để thực hiện tính toán các đại lượng thống kê cơ bản, cũng như lập các bảng thống kê trên dataframe cho trong file excel nghesi.xlsx sẵn có trên link sau:

<https://docs.google.com/spreadsheets/d/1q7RVfoTKEuLP0Ud0a50mJu7e1-YJJbUe/edit?usp=sharing&ouid=106843557461060771783&rtpof=true&sd=true>

Chương 5. TRỰC QUAN HOÁ DỮ LIỆU VỚI BIỂU ĐỒ

Mục đích

Biểu đồ là một công cụ mô tả dữ liệu thống kê hiệu quả, với sức mạnh của máy tính, việc vẽ biểu đồ trở nên dễ dàng hơn. Chương này trình bày cách vẽ các biểu đồ cơ bản trong thống kê ứng dụng với Python.

Yêu cầu

- Khái quát và giải thích được các câu lệnh cơ bản của các ví dụ minh họa cách vẽ các biểu đồ.
- Giải thích được các mẫu cơ bản cho vẽ biểu đồ với Python
- Cài đặt thực hiện được các ví dụ minh họa vẽ biểu đồ với 2 thư viện cơ bản là *matplotlib* và *seaborn* của Python.
- Trình bày được ý nghĩa và giải thích được các loại biểu đồ cơ bản.

5. 1. Mẫu tổng quát để vẽ biểu đồ với Python

Để vẽ biểu đồ có thể sử dụng mẫu tổng quát sau:

Import the required packages

```
import matplotlib.pyplot as plt
import numpy as np
```

Generate the data

```
x = np.arange(0, 10, 0.2)
y = np.sin(x)
z = np.cos(x)
```

Generate the figure and the axes

```
fig, axs = plt.subplots(nrows=2, ncols=1)
# On the first axis, plot the sine and label the ordinate
axs[0].plot(x, y)
axs[0].set_ylabel('Sine')
```

On the second axis, plot the cosine

```
axs[1].plot(x, z)
axs[1].set_ylabel('Cosine')
```

Display the resulting plot

```
plt.show()
```

Cụ thể ta có cú pháp chung để sử dụng để tạo một số biểu đồ cơ bản với *matplotlib*

<https://datatofish.com/python-tutorials/>

Scatter plot

```
import matplotlib.pyplot as plt  
plt.scatter(xAxis,yAxis)  
plt.title('title name')  
plt.xlabel('xAxis name')  
plt.ylabel('yAxis name')  
plt.show()
```

Line chart

```
import matplotlib.pyplot as plt  
plt.plot(xAxis,yAxis)  
plt.title('title name')  
plt.xlabel('xAxis name')  
plt.ylabel('yAxis name')  
plt.show()
```

Bar chart

```
import matplotlib.pyplot as plt  
xAxis = [i + 0.5 for i, _ in enumerate(xAxis)]  
plt.bar(xAxis,yAxis)  
plt.title('title name')  
plt.xlabel('xAxis name')  
plt.ylabel('yAxis name')  
plt.xticks([i + 0.5 for i, _ in enumerate(xAxis)], xAxis)  
plt.show()
```

5.2. Biểu đồ histogram

Bước 1. Thu thập dữ liệu

Bước 2: Xác định số lượng bin

Tiếp theo, xác định số lượng bin sẽ được sử dụng cho biểu đồ.

Để đơn giản hơn, hãy đặt số lượng bin là 10. Ở phần dưới, chúng ta sẽ làm rõ cách để định ra số bin.

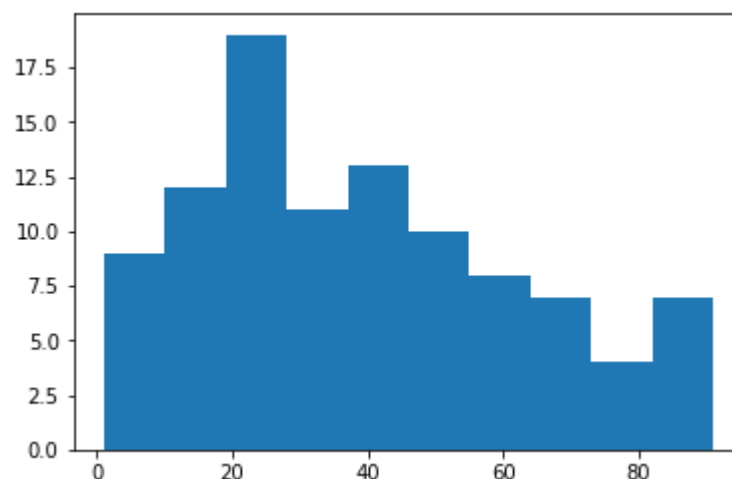
Bước 3: Vẽ biểu đồ

```
import matplotlib.pyplot as plt

x = [1,1,2,3,3,5,7,8,9,10,
     10,11,11,13,13,15,16,17,18,18,
     18,19,20,21,21,23,24,24,25,25,
     25,25,26,26,26,27,27,27,27,27,
     29,30,30,31,33,34,34,34,35,36,
     36,37,37,38,38,39,40,41,41,42,
     43,44,45,45,46,47,48,48,49,50,
     51,52,53,54,55,55,56,57,58,60,
     61,63,64,65,66,68,70,71,72,74,
     75,77,81,83,84,87,89,90,90,91
    ]

plt.hist(x, bins=10)

plt.show()
```



Hình 5. 1. Minh họa biểu đồ histogram

Cách bổ sung để xác định số lượng bin:

Chúng ta có thể tính số lượng bin bằng cách sử dụng các công thức sau:

- n = số quan sát

- **Phạm vi** = giá trị lớn nhất - giá trị nhỏ nhất
- **# khoảng** = \sqrt{n}
- **Chiều rộng của khoảng** = Phạm vi / (# khoảng)

Công thức này có thể được sử dụng để tạo bảng tần suất cho biểu đồ histogram.

Giả sử với dữ liệu tuổi là:

Age

```
1, 1, 2, 3, 3, 5, 7, 8, 9, 10,
10, 11, 11, 13, 13, 15, 16, 17, 18, 18,
18, 19, 20, 21, 21, 23, 24, 24, 25, 25,
25, 25, 26, 26, 26, 27, 27, 27, 27, 27,
29, 30, 30, 31, 33, 34, 34, 34, 35, 36,
36, 37, 37, 38, 38, 39, 40, 41, 41, 42,
43, 44, 45, 45, 46, 47, 48, 48, 49, 50,
51, 52, 53, 54, 55, 55, 56, 57, 58, 60,
61, 63, 64, 65, 66, 68, 70, 71, 72, 74,
75, 77, 81, 83, 84, 87, 89, 90, 90, 91
```

Theo công thức trên ta có bảng tần suất sau:

Intervals (bins)	Frequency
0-9	9
10-19	13
20-29	19
30-39	15
40-49	13
50-59	10
60-69	7
70-79	6
80-89	5
90-99	3

```
import matplotlib.pyplot as plt

x = [1, 1, 2, 3, 3, 5, 7, 8, 9, 10,
     10, 11, 11, 13, 13, 15, 16, 17, 18, 18,
     18, 19, 20, 21, 21, 23, 24, 24, 25, 25,
```

```
25, 25, 26, 26, 26, 27, 27, 27, 27, 27,
29, 30, 30, 31, 33, 34, 34, 34, 35, 36,
36, 37, 37, 38, 38, 39, 40, 41, 41, 42,
43, 44, 45, 45, 46, 47, 48, 48, 49, 50,
51, 52, 53, 54, 55, 55, 56, 57, 58, 60,
61, 63, 64, 65, 66, 68, 70, 71, 72, 74,
75, 77, 81, 83, 84, 87, 89, 90, 90, 91
]

plt.hist(x, bins=[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 99])
plt.show()
```

5.3. Biểu đồ tròn (Pie)

Để tạo biểu đồ hình tròn bằng matplotlib, chúng ta có thể sử dụng mẫu sau để tạo biểu đồ hình tròn:

```
import matplotlib.pyplot as plt
my_data = [value1, value2, value3, ...]
my_labels = 'label1', 'label2', 'label3', ...
plt.pie(my_data, labels=my_labels, autopct='%1.1f%%')
plt.title('My Title')
plt.axis('equal')
plt.show()
```

Các bước tạo biểu đồ hình tròn bằng Matplotlib

Bước 1: Thu thập dữ liệu

Ví dụ với dữ liệu sau:

Tasks Pending	300
Tasks Ongoing	500
Tasks Completed	700

Bước 2: Vẽ biểu đồ hình tròn

Ta có thể sử dụng mẫu sau để hỗ trợ vẽ biểu đồ.


```
import matplotlib.pyplot as plt

Tasks = [300,500,700]

my_labels = 'Tasks Pending','Tasks Ongoing','Tasks Completed'

plt.pie(Tasks,labels=my_labels,autopct='%1.1f%%')

plt.title('My Tasks')

plt.axis('equal')

plt.show()
```



Hình 5. 2. Minh hoạ biểu đồ tròn

Bước 3: Tạo kiểu cho biểu đồ

Bạn có thể tạo kiểu thêm cho biểu đồ hình tròn bằng cách thêm:

- Start angle
- Shadow
- Colors
- Explode component

```
import matplotlib.pyplot as plt

Tasks = [300,500,700]

my_labels = 'Tasks Pending','Tasks Ongoing','Tasks Completed'

my_colors = ['lightblue','lightsteelblue','silver']

my_explode = (0, 0.1, 0)
```

```
plt.pie(Tasks, labels=my_labels, autopct='%1.1f%%',
startangle=15, shadow = True, colors=my_colors,
explode=my_explode)

plt.title('My Tasks')

plt.axis('equal')

plt.show()
```



Tạo biểu đồ dựa trên Pandas DataFrame

Chúng ta cũng có thể tạo biểu đồ hình tròn dựa trên pandas. Với ví dụ trên, cách vẽ biểu đồ dựa trên pandas như sau:

```
from pandas import DataFrame
import matplotlib.pyplot as plt

Data = {'Tasks': [300,500,700]}

df = DataFrame(Data,columns=['Tasks'])

my_labels = 'Tasks Pending','Tasks Ongoing','Tasks
Completed'

plt.pie(df,labels=my_labels,autopct='%1.1f%%')

plt.title('My Tasks')

plt.axis('equal')

plt.show()
```

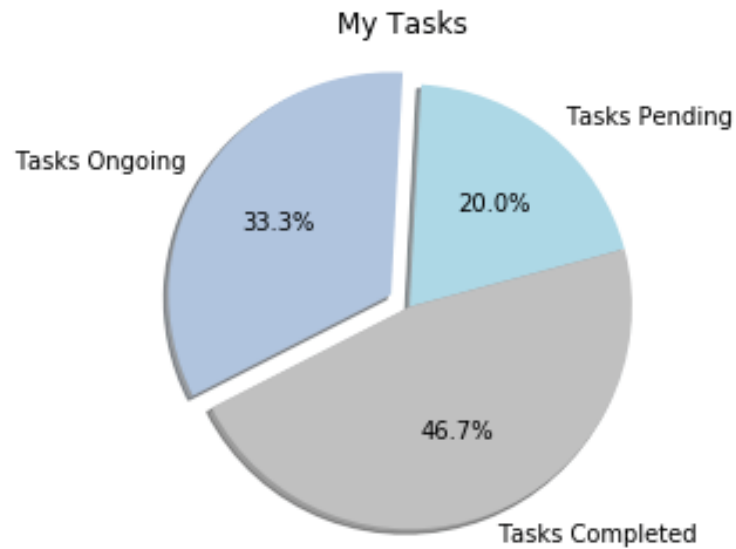


Sau đó, có thể chọn tạo kiểu cho biểu đồ bằng cú pháp sau:

```
from pandas import DataFrame
import matplotlib.pyplot as plt
Data = {'Tasks': [300,500,700]}
df = DataFrame(Data,columns=['Tasks'])
my_labels = 'Tasks Pending','Tasks Ongoing','Tasks Completed'
my_colors = ['lightblue','lightsteelblue','silver']
my_explode = (0, 0.1, 0)

plt.pie(df, labels=my_labels, autopct='%1.1f%%',
startangle=15, shadow = True, colors=my_colors,
explode=my_explode)

plt.title('My Tasks')
plt.axis('equal')
plt.show()
```



5.4. Vẽ biểu đồ đường

Dưới đây là một mẫu có thể sử dụng để vẽ biểu đồ đường:

```
import matplotlib.pyplot as plt
plt.plot(xAxis, yAxis)
plt.title('title name')
plt.xlabel('xAxis name')
plt.ylabel('yAxis name')
plt.show()
```

Bước 1: Thu thập dữ liệu

Ví dụ: Dữ liệu sau về hai biến số của một nền kinh tế:

- Năm
- Tỷ lệ thất nghiệp

Year	Unemployment_Rate
1920	9.8
1930	12
1940	8
1950	7.2

1960	6.9
1970	7
1980	6.5
1990	6.2
2000	5.5
2010	6.3

Bước 2: Đưa dữ liệu vào list hay dataframe

Chẳng hạn đưa dữ liệu trên bằng cách sử dụng hai List sau:

```
Year =
[1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010]

Unemployment_Rate =
[9.8, 12, 8, 7.2, 6.9, 7, 6.5, 6.2, 5.5, 6.3]
```

Bước 3: Vẽ biểu đồ dạng đường

```
import matplotlib.pyplot as plt

Year =
[1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010]

Unemployment_Rate = [9.8, 12, 8, 7.2, 6.9, 7, 6.5, 6.2, 5.5, 6.3]

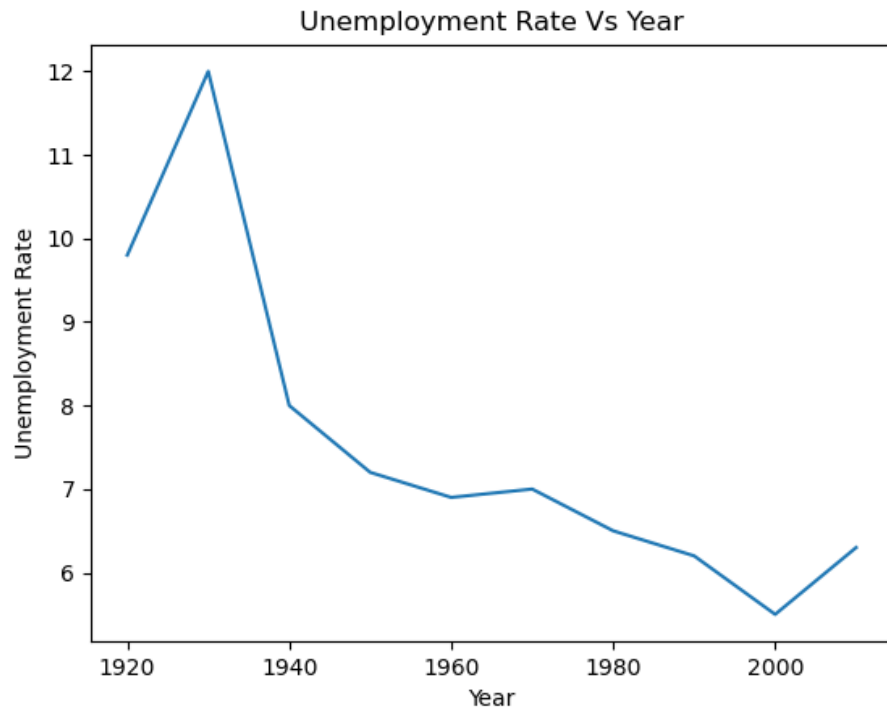
plt.plot(Year, Unemployment_Rate)

plt.title('Unemployment Rate Vs Year')

plt.xlabel('Year')

plt.ylabel('Unemployment Rate')

plt.show()
```



Hình 5. 3. Minh họa biểu đồ đường

Ta có thể tạo *kiểu* thêm cho biểu đồ đường bằng cách sử dụng đoạn mã sau:

```
import matplotlib.pyplot as plt

Year =
[1920,1930,1940,1950,1960,1970,1980,1990,2000,2010]
Unemployment_Rate = [9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3]

plt.plot(Year, Unemployment_Rate, color='red',
marker='o')

plt.title('Unemployment Rate Vs Year', fontsize=14)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Unemployment Rate', fontsize=14)
plt.grid(True)
plt.show()
```



Cách tạo biểu đồ đường với Pandas DataFrame

Ngoài ra, ta cũng có thể đưa dữ liệu và dataframe, sau đó vẽ biểu đồ.

Xét đoạn mã sau:

```
import pandas as pd
import matplotlib.pyplot as plt

Data = {'Year':
[1920,1930,1940,1950,1960,1970,1980,1990,2000,2010],

'Unemployment_Rate':
[9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3]

}

df =
pd.DataFrame(Data,columns=['Year','Unemployment_Rate'])

plt.plot(df['Year'], df['Unemployment_Rate'],
color='red', marker='o')

plt.title('Unemployment Rate Vs Year', fontsize=14)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Unemployment Rate', fontsize=14)
plt.grid(True)
plt.show()
```



5.5. Biểu đồ phân tán (Scatter)

Có thể sử dụng mẫu sau để tạo biểu đồ phân tán với Matplotlib:

```
import matplotlib.pyplot as plt
plt.scatter(xAxis, yAxis)
plt.title('title name')
plt.xlabel('xAxis name')
plt.ylabel('yAxis name')
plt.show()
```

Bước 1: Thu thập dữ liệu

Ví dụ: giả sử có dữ liệu sau về một nền kinh tế:

Unemployment_Rate	Stock_Index_Price
6.1	1500
5.8	1520
5.7	1525
5.7	1523

5.8	1515
5.6	1540
5.5	1545
5.3	1560
5.2	1555
5.2	1565

Ta cần mô tả *mối quan hệ* giữa Unemployment_Rate và Stock_Index_Price.

Bước 2: Đưa dữ liệu vào trong 1 list

```
Unemployment_Rate =
[6.1, 5.8, 5.7, 5.7, 5.8, 5.6, 5.5, 5.3, 5.2, 5.2]

Stock_Index_Price =
[1500, 1520, 1525, 1523, 1515, 1540, 1545, 1560, 1555, 1565]

print(Unemployment_Rate)
print(Stock_Index_Price)
```

Bước 3: Tạo biểu đồ phân tán với Matplotlib

Ta có thể sử dụng mẫu sau để tạo sơ đồ phân tán trong Python.

Trong ví dụ sau:

- xAxis là Unemployment_Rate
- yAxis là Stock_Index_Price
- title name là 'Unemployment Rate Vs Stock Index Price'
- xAxis name là 'Unemployment Rate'
- yAxis name là 'Stock Index Price'

```
import matplotlib.pyplot as plt

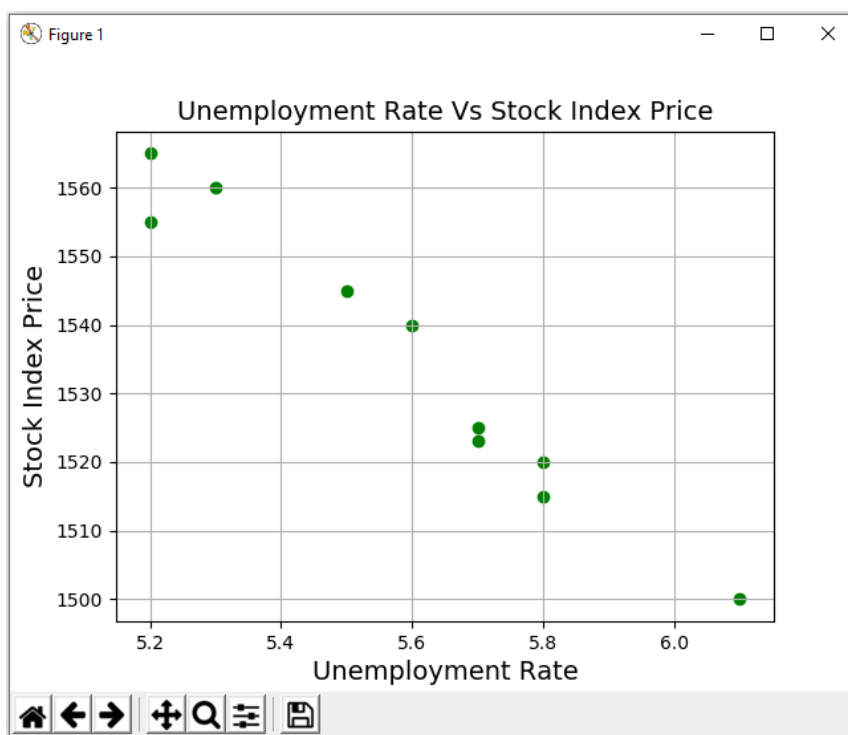
Unemployment_Rate =
[6.1, 5.8, 5.7, 5.7, 5.8, 5.6, 5.5, 5.3, 5.2, 5.2]

Stock_Index_Price =
[1500, 1520, 1525, 1523, 1515, 1540, 1545, 1560, 1555, 1565]

plt.scatter(Unemployment_Rate, Stock_Index_Price,
            color='green')
```

```
plt.title('Unemployment Rate Vs Stock Index Price',
          fontsize=14)

plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()
```



Hình 5. 4. Minh hoạ biểu đồ phân tán (scatter)

Qua biểu đồ trên ta có thể thấy mối quan hệ giữa Unemployment_Rate và Stock_Index_Price. Khi Unemployment Rate tăng, Stock Index Price *giảm*.

Biểu đồ phân tán đặc biệt hữu ích khi áp dụng [mô hình hồi quy tuyến tính](#). Biểu đồ phân tán giúp ta xác định xem có mối quan hệ tuyến tính giữa các biến hay không.

Tạo biểu đồ phân tán bằng Pandas DataFrame

Tương tự như bước 1 của ví dụ trên, ta đưa danh dữ liệu vào 1 dataframe df, sau đó vẽ biểu đồ như sau:

```
import pandas as pd
import matplotlib.pyplot as plt

data = {'Unemployment_Rate':
[6.1,5.8,5.7,5.7,5.8,5.6,5.5,5.3,5.2,5.2],
        'Stock_Index_Price':
[1500,1520,1525,1523,1515,1540,1545,1560,1555,1565]}


```

```
df =
pd.DataFrame(data, columns=['Unemployment_Rate', 'Stock_Index_Price']
)
plt.scatter(df['Unemployment_Rate'], df['Stock_Index_Price'],
color='green')
plt.title('Unemployment Rate Vs Stock Index Price', fontsize=14)
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()
```

5.6. Biểu đồ thanh (bar chart)

Bước 1: Thu thập dữ liệu cho biểu đồ thanh

Xét dữ liệu sau:

Country	GDP_Per_Capita
USA	45000
Canada	42000
Germany	52000
UK	49000
France	47000

Bước 2: Đưa dữ liệu vào 1 list hay dataframe

```
Country = ['USA', 'Canada', 'Germany', 'UK', 'France']
```

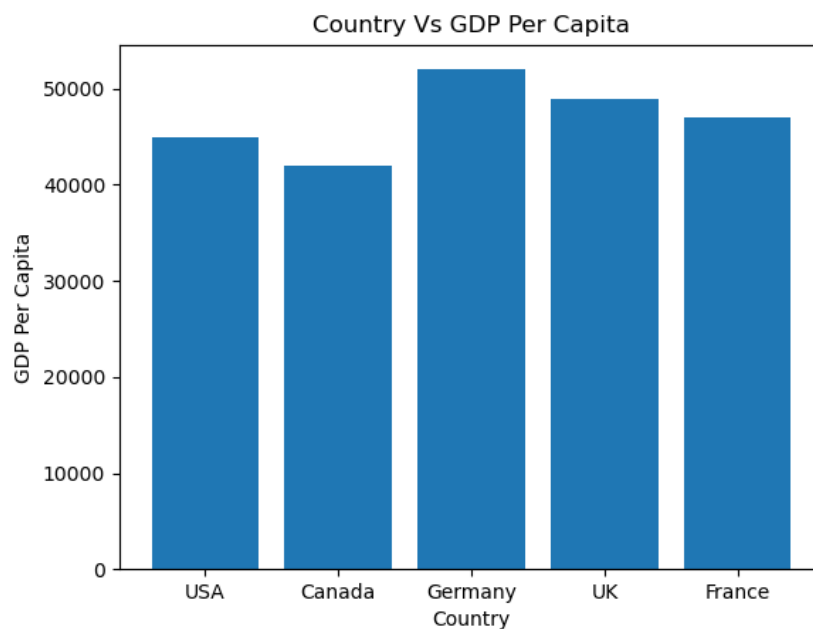
```
GDP_Per_Capita = [45000, 42000, 52000, 49000, 47000]
```

Bước 3: Tạo biểu đồ thanh

Có thể sử dụng mẫu bên dưới để tạo biểu đồ thanh:

```
import matplotlib.pyplot as plt
plt.bar(xAxis, yAxis)
plt.title('title name')
plt.xlabel('xAxis name')
plt.ylabel('yAxis name')
plt.show()
import matplotlib.pyplot as plt
```

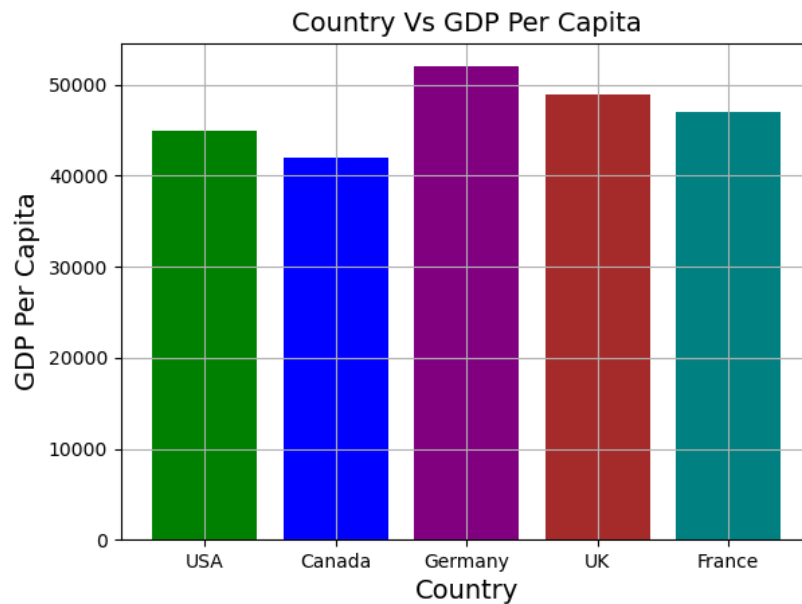
```
Country = ['USA', 'Canada', 'Germany', 'UK', 'France']
GDP_Per_Capita = [45000, 42000, 52000, 49000, 47000]
plt.bar(Country, GDP_Per_Capita)
plt.title('Country Vs GDP Per Capita')
plt.xlabel('Country')
plt.ylabel('GDP Per Capita')
plt.show()
```



Hình 5. 5. Minh họa biểu đồ thanh (bar)

Thay đổi kiểu dáng của biểu đồ

```
import matplotlib.pyplot as plt
Country = ['USA', 'Canada', 'Germany', 'UK', 'France']
GDP_Per_Capita = [45000, 42000, 52000, 49000, 47000]
New_Colors = ['green', 'blue', 'purple', 'brown', 'teal']
plt.bar(Country, GDP_Per_Capita, color=New_Colors)
plt.title('Country Vs GDP Per Capita', fontsize=14)
plt.xlabel('Country', fontsize=14)
plt.ylabel('GDP Per Capita', fontsize=14)
plt.grid(True)
plt.show()
```



Tạo biểu đồ thanh với Pandas DataFrame

Đây là mã có thể tham khảo:

```
import matplotlib.pyplot as plt
import pandas as pd

Data = {'Country': ['USA', 'Canada', 'Germany', 'UK', 'France'],
        'GDP_Per_Capita': [45000, 42000, 52000, 49000, 47000]}

df = pd.DataFrame(Data, columns=['Country', 'GDP_Per_Capita'])
New_Colors = ['green', 'blue', 'purple', 'brown', 'teal']
plt.bar(df['Country'], df['GDP_Per_Capita'],
        color=New_Colors)
plt.title('Country Vs GDP Per Capita', fontsize=14)
plt.xlabel('Country', fontsize=14)
plt.ylabel('GDP Per Capita', fontsize=14)
plt.grid(True)
plt.show()
```

5.7. Biểu đồ thanh ngang

Bước 1: Thu thập dữ liệu cho biểu đồ

Ví dụ:

Product	Quantity
Computer	320
Monitor	450
Laptop	300
Printer	120
Tablet	280

Đưa dữ liệu vào trong các list

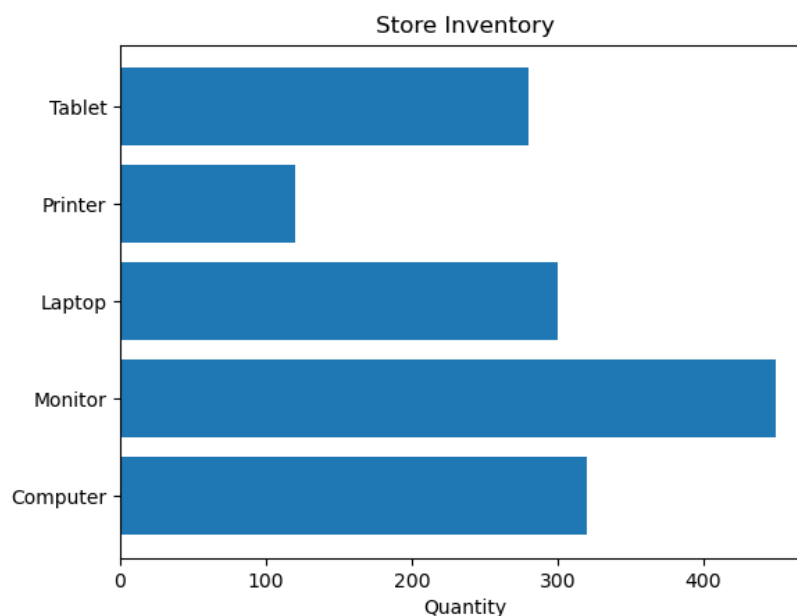
```
Product = ['Computer', 'Monitor', 'Laptop', 'Printer', 'Tablet']
Quantity = [320, 450, 300, 120, 280]
```

Bước 2: Vẽ biểu đồ thanh ngang

Có thể tham khảo đoạn mã sau:

```
import matplotlib.pyplot as plt
Product =
['Computer', 'Monitor', 'Laptop', 'Printer', 'Tablet']
Quantity = [320, 450, 300, 120, 280]
plt.barh(Product, Quantity)      # lưu ý: barh
plt.title('Store Inventory')
plt.ylabel('Product')
plt.xlabel('Quantity')
plt.show()
```

Lưu ý rằng '**Sản phẩm**' được hiển thị trên y_axis, trong khi '**Số lượng**' được hiển thị trên x_axis:



Hình 5. 6. Minh hoạ biểu đồ thang ngang

Bước 3 (tùy chọn): Tạo kiểu cho biểu đồ

Có thể tạo kiểu thêm cho biểu đồ:

```
plt.style.use('ggplot')
```

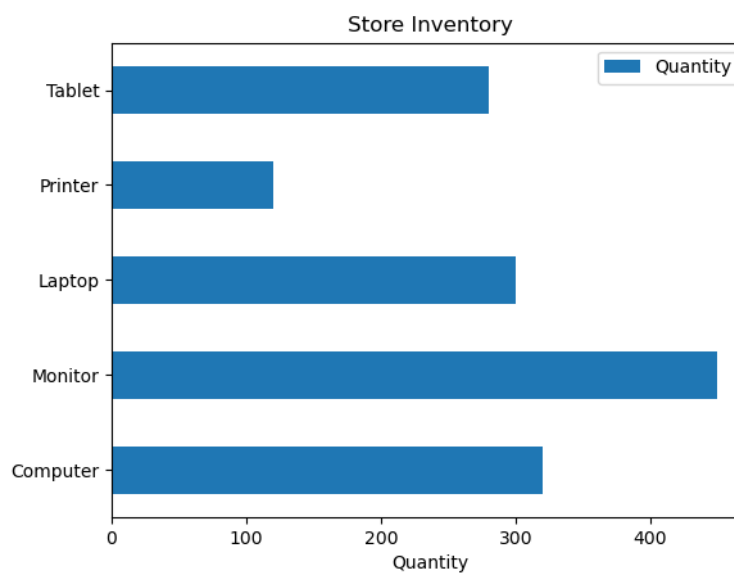
Đoạn mã trên được tinh chỉnh lại:

```
import matplotlib.pyplot as plt
Product = ['Computer', 'Monitor', 'Laptop', 'Printer', 'Tablet']
Quantity = [320, 450, 300, 120, 280]
plt.style.use('ggplot')
plt.barh(Product, Quantity)
plt.title('Store Inventory')
plt.ylabel('Product')
plt.xlabel('Quantity')
plt.show()
```



Vẽ biểu đồ thanh ngang với pandas

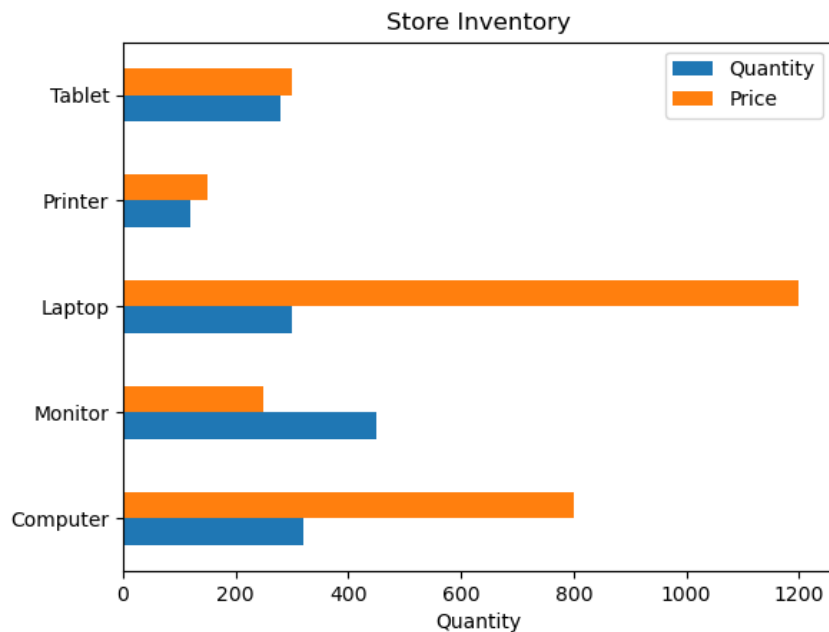
```
import matplotlib.pyplot as plt
import pandas as pd
data = {'Quantity': [320,450,300,120,280]}
df = pd.DataFrame(data,columns=['Quantity'], index =
['Computer','Monitor','Laptop','Printer','Tablet'])
df.plot.barh()
plt.title('Store Inventory')
plt.ylabel('Product')
plt.xlabel('Quantity')
plt.show()
```



Giả sử ta cần vẽ biểu đồ thanh ngang biểu thị cả 'Price' và "Quantity" của từng loại sản phẩm.

Có thể tham khảo đoạn mã sau:

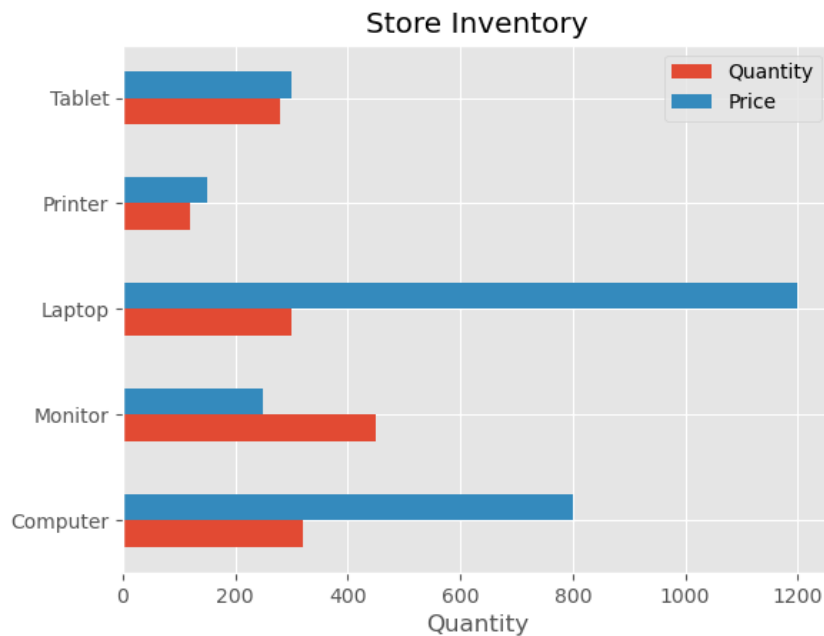
```
import matplotlib.pyplot as plt
import pandas as pd
data = {'Quantity': [320,450,300,120,280],
        'Price': [800,250,1200,150,300]}
df = pd.DataFrame(data,columns=['Quantity','Price'], index =
['Computer','Monitor','Laptop','Printer','Tablet'])
df.plot.barh()
plt.title('Store Inventory')
plt.ylabel('Product')
plt.xlabel('Quantity')
plt.show()
```



Thêm kiểu dáng cho biểu đồ:

```
import matplotlib.pyplot as plt
import pandas as pd
data = {'Quantity': [320,450,300,120,280],
        'Price': [800,250,1200,150,300]}
df = pd.DataFrame(data,columns=['Quantity','Price'], index =
['Computer','Monitor','Laptop','Printer','Tablet'])
plt.style.use('ggplot')
df.plot.barh()
plt.title('Store Inventory')
plt.ylabel('Product')
plt.xlabel('Quantity')
```

```
plt.show()
```

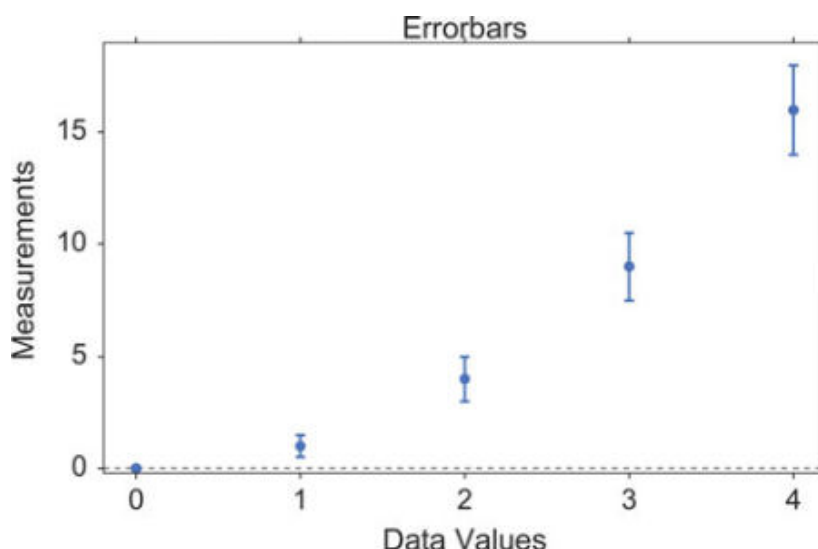


5.8. Errore bar (thanh lỗi)

Thanh lỗi là một cách phổ biến để hiển thị giá trị trung bình và sự thay đổi khi so sánh các giá trị đo lường. Cần nêu rõ ràng thanh lỗi tương ứng với độ lệch chuẩn hoặc với sai số tiêu chuẩn của dữ liệu. Khi sử dụng sai số tiêu chuẩn, ta có một tính chất rất hay là: Khi các thanh lỗi cho các sai số tiêu chuẩn của hai nhóm chồng lên nhau, người ta có thể chắc chắn rằng sự khác biệt giữa hai giá trị trung bình của hai nhóm là không có ý nghĩa thống kê ($p > 0.05$).

```
index = np.arange(5)
y = index**2
errorBar = index/2

plt.errorbar(index, y, yerr=errorBar, fmt='o', capsize=5,
             capthick=3)
```



Hình 5. 7. Minh họa thanh lỗi

5.9. Biểu đồ hình hộp (Boxplot)

Biểu đồ hình hộp (đôi khi được gọi là biểu đồ hình hộp và râu) là một biểu đồ hiển thị tóm tắt năm giá trị quan trọng của một tập dữ liệu.

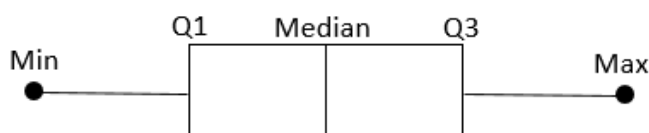
Bao gồm:

- Giá trị nhỏ nhất (minimum)
- Phân vị thứ nhất (The first quartile – bách phân vị 25%) – Q1
- Median
- Phân vị thứ ba (The third quartile – bách phân vị 75%) – Q3
- Giá trị lớn nhất (maximum)

Biểu đồ hình hộp cho phép chúng ta dễ dàng hình dung sự phân bố của các giá trị trong một tập dữ liệu bằng cách sử dụng một biểu đồ đơn giản.
<https://www.statology.org/boxplots/>

Cách tạo Boxplot

Để tạo một Boxplot, chúng ta vẽ một ô từ Q1 đến Q3. Sau đó, chúng ta vẽ một đường thẳng đứng ở trung tuyến Q1 và Q3. Cuối cùng, chúng ta vẽ "râu" từ Q1, Q3 đến giá trị min và max.



Hình 5. 8. Minh họa biểu đồ dạng hộp (box plot)

Giả sử chúng ta có tập dữ liệu sau cho biết chiều cao của mười cây trồng:

Plant height (inches)
14
16
12
11
24
19
13
12
20
10

Để tạo một boxplot, chúng ta cần tìm min, Q1, median, Q3 và max.

Bước 1: Sắp xếp dữ liệu từ nhỏ nhất đến lớn nhất.

10, 11, 12, 12, 13, 14, 16, 19, 20, 24

Bước 2: Tìm median (trung vị)

Trong trường hợp này, đó là giá trị trung bình của hai số ở giữa:

10, 11, 12, 12, **13, 14**, 16, 19, 20, 24

Trung vị = $(13 + 14) / 2 = 13,5$

Bước 3: Tìm Q1 và Q3

Q1 là trung vị của các số ở bên trái của trung vị. Trong trường hợp này, nó là 12.

10, 11, 12, 12, 13, 14, 16, 19, 20, 24

Q3 là trung vị của các số ở bên phải của trung vị. Trong trường hợp này, nó là

19.

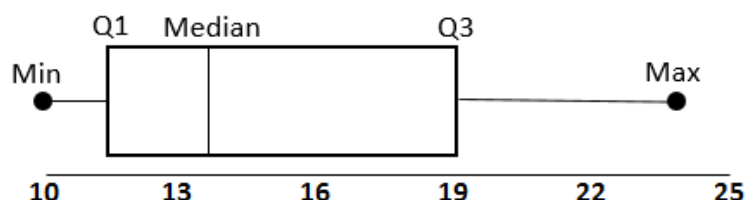
10, 11, 12, 12, 13, **14, 16, 19**, 20, 24

Bước 4: Tìm giá trị nhỏ nhất và lớn nhất

Min là 10 và Max là 24.

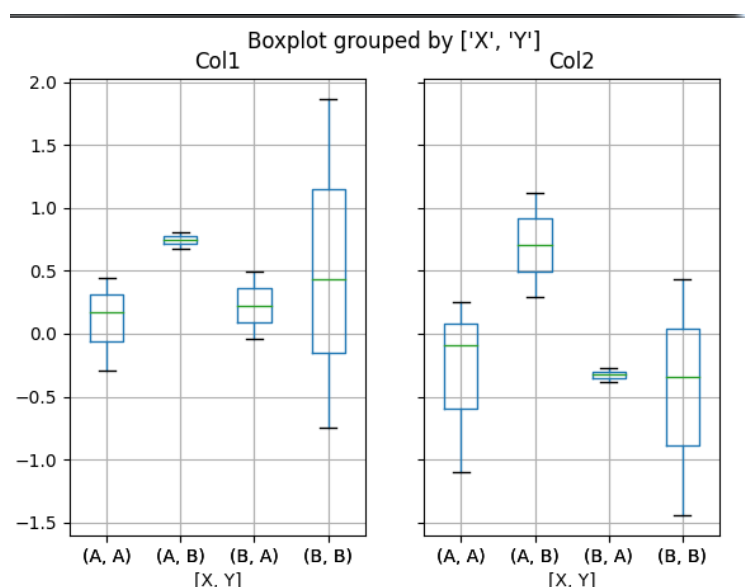
10, 11, 12, 12, 13, 14, 16, 19, 20, **24**

Bước 5: Vẽ Boxplot bằng cách sử dụng 5 giá trị trên



Xét ví dụ sau:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df = pd.DataFrame(np.random.randn(10, 3),
                  columns=['Col1', 'Col2', 'Col3'])
df['X'] = pd.Series(['A', 'A', 'A', 'A', 'A',
                    'B', 'B', 'B', 'B', 'B'])
df['Y'] = pd.Series(['A', 'B', 'A', 'B', 'A',
                    'B', 'A', 'B', 'A', 'B'])
boxplot = df.boxplot(column=['Col1', 'Col2'], by=['X', 'Y'])
plt.show()
```



<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.boxplot.html>

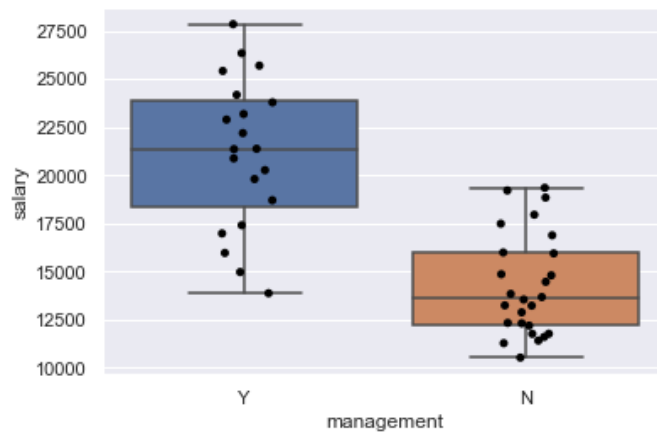
5.10. Biểu đồ Boxplot và Violin plot: 1 nhân tố

Các biểu đồ dạng hộp (boxplot) là phi tham số: chúng hiển thị sự thay đổi trong các mẫu của một tổng thể thống kê mà không đưa ra bất kỳ giả định nào về phân phối thống kê cơ bản.

Với dữ liệu trong file salary, chúng ta vẽ biểu đồ box và violin nhờ sự hỗ trợ của seaborn như sau: (xem [3])

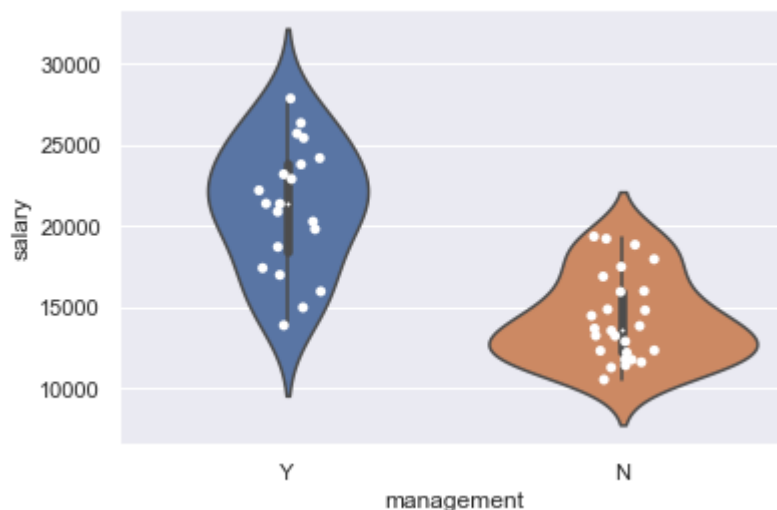
```
ax = sns.boxplot(x="management", y="salary", data=salary)
ax = sns.stripplot(x="management", y="salary",
data=salary, jitter=True, color="black")
```

Ta vẽ biểu đồ với dữ liệu của x là trường management và dữ liệu của trục y là salary.



Hình 5. 9. Biểu đồ Boxplot và Violin plot một nhân tố

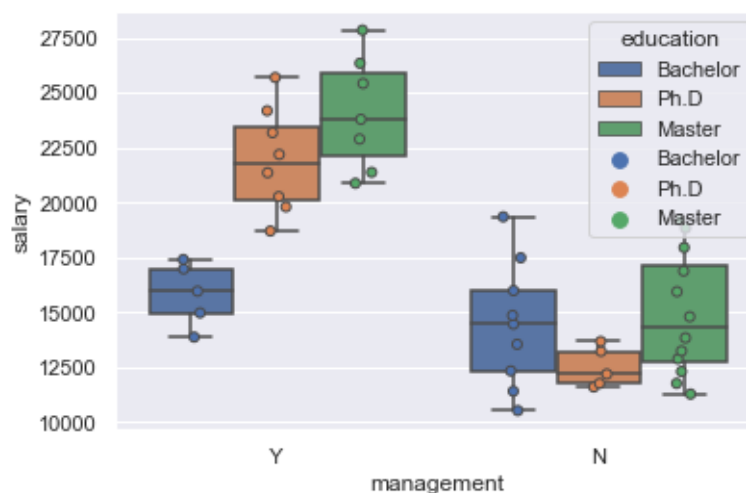
```
ax = sns.violinplot(x="management", y="salary", data=salary)
ax = sns.stripplot(x="management", y="salary", data=salary, jitter=True,
color="white")
```



5.11. Boxplot và violin plot: hai nhân tố

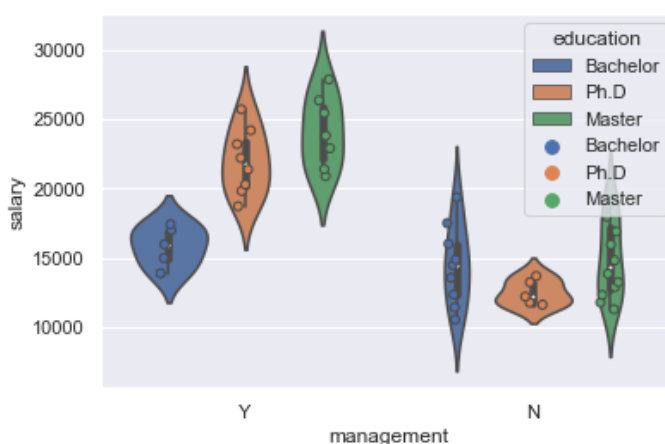
```
ax = sns.boxplot(x="management", y="salary",
hue="education", data=salary)
ax = sns.stripplot(x="management", y="salary",
```

```
hue="education", data=salary, jitter=True, dodge=True,
linewidth=1)
```



Hình 5. 10. Boxplot và violin plot hai nhân tố

```
ax = sns.violinplot(x="management", y="salary",
hue="education", data=salary)
ax = sns.stripplot(x="management", y="salary",
hue="education", data=salary, jitter=True, dodge=True,
linewidth=1)
```

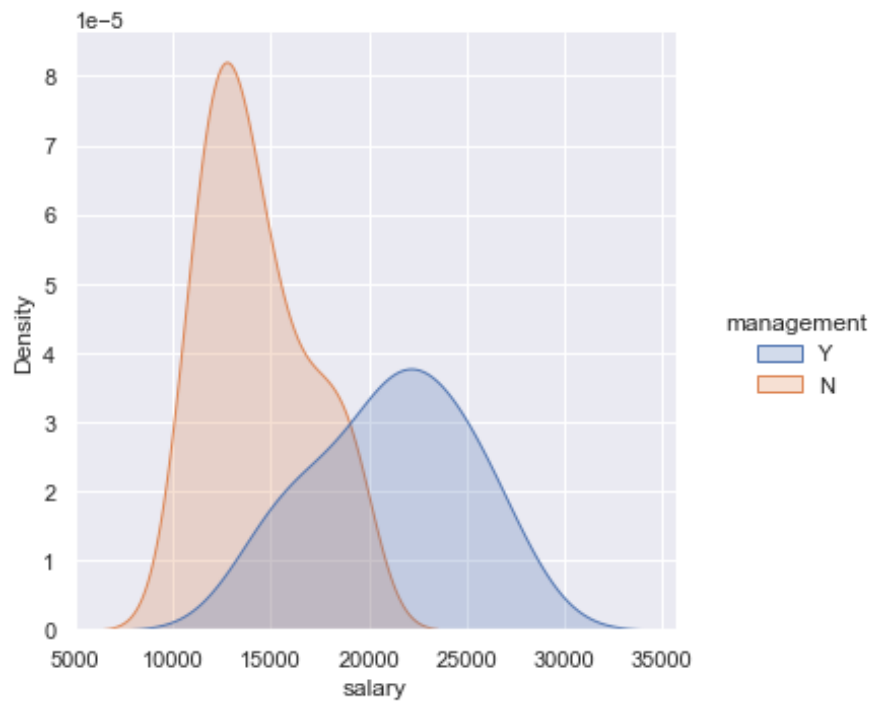


5.12. Vẽ đồ thị phân bố xác suất và mật độ xác suất

Trong thư viện seaborn ta sử dụng mẫu câu lệnh sau:

Với tham số `kind = 'kde'` để đồ thị hàm phân bố xác suất.

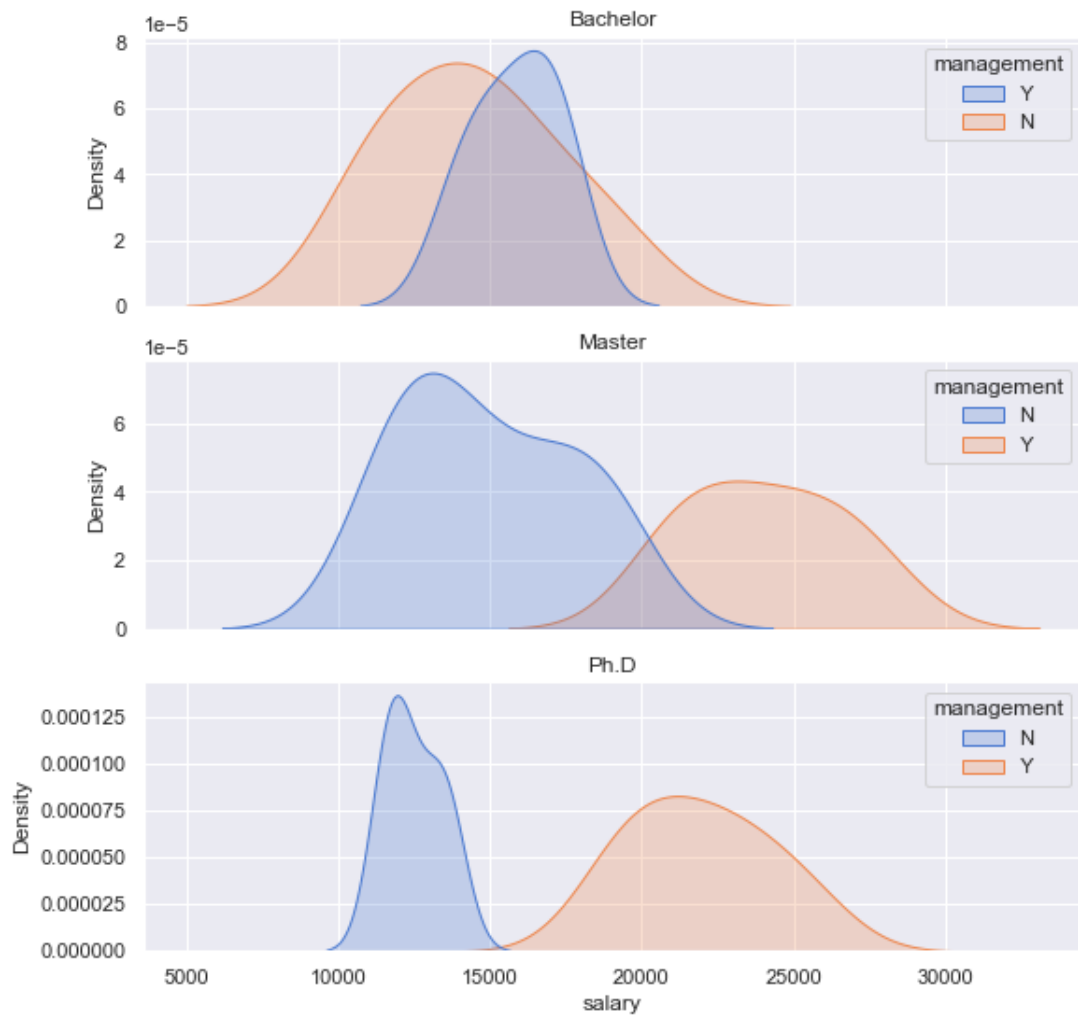
```
ax = sns.displot(x="salary", hue="management", kind="kde",
data=salary, fill=True)
```



Hình 5. 11. Minh hoạ đồ thị phân bố xác suất và mật độ xác suất

Thể hiện đồ thị phân phối và mật độ cho các nhóm dữ liệu

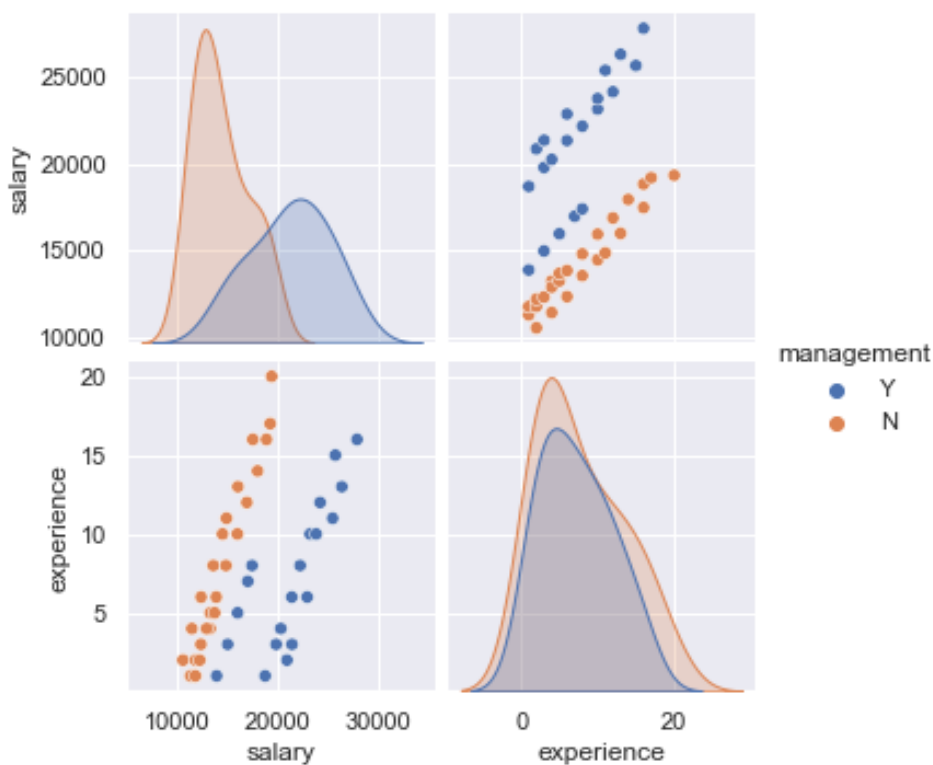
```
fig, axes = plt.subplots(3, 1, figsize=(9, 9), sharex=True)
i = 0
for edu, d in salary.groupby(['education']):
    sns.kdeplot(x="salary", hue="management", data=d,
                fill=True, ax=axes[i], palette="muted")
    axes[i].set_title(edu)
    i += 1
```

Hình 5. 12. Minh họa đồ thị phân bố xác suất và mật độ xác suất (Nguồn [3])

Vẽ biểu đồ phân tán theo cặp

```
ax = sns.pairplot(salary, hue="management")
```



(Nguồn [3])

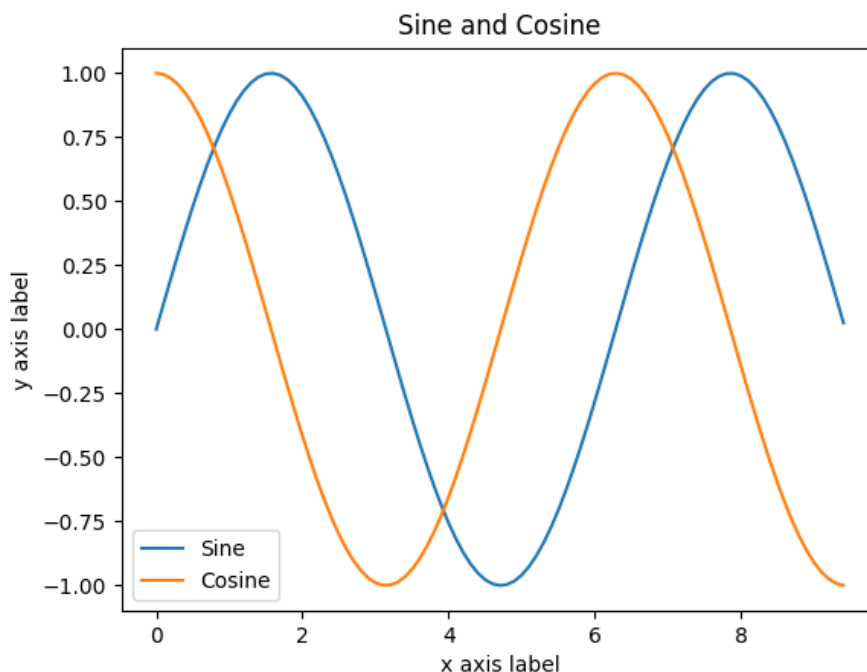
5.13. Vẽ đồ thị trên cùng một trục

```
import numpy as np
import matplotlib.pyplot as plt

# Tính toán x và y để lấy 1 số cặp điểm trên đồ thị hình
sin và cosin
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Vẽ đồ thị theo các điểm dữ liệu dùng matplotlib

plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label') # nhãn trục x
plt.ylabel('y axis label') # nhãn trục y
plt.title('Sine and Cosine') # set tiêu đề
plt.legend(['Sine', 'Cosine']) # Hiển thị chú thích của
đồ thị
plt.show()
```



Hình 5. 13. Minh họa vẽ biểu đồ trên cùng 1 trục

Ta có thể vẽ các đồ thị khác nhau trên cùng một hình, bằng cách sử dụng hàm `subplot`.

Dưới đây là một ví dụ:

```
import numpy as np
import matplotlib.pyplot as plt

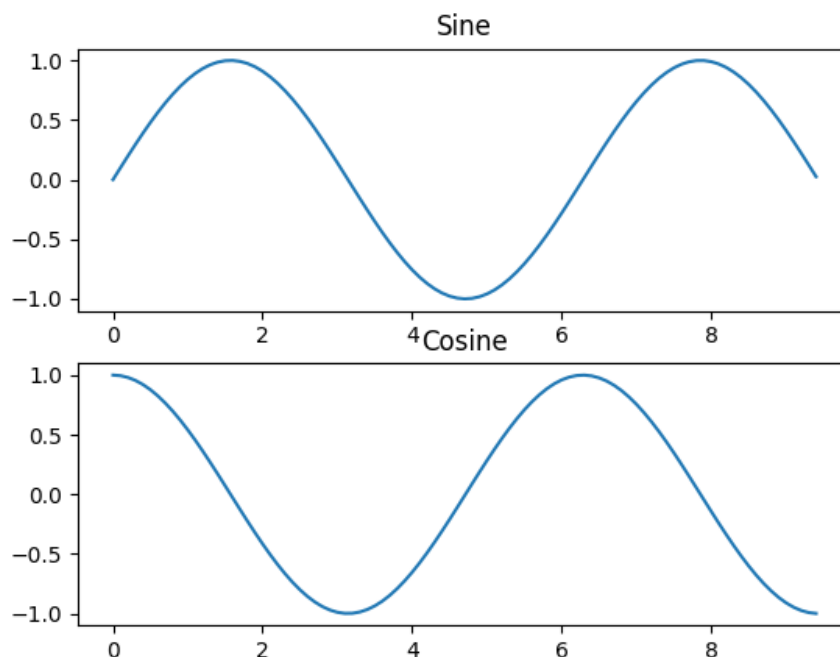
# Tính toán x và y để lấy 1 số cặp điểm trên đồ thị hình
sin và cosin
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Thiết lập lưới các đồ thị chia làm 2 đồ thị con theo
chiều cao.
# và set đồ thị con đầu tiên active.
# active nghĩa là bảo code rằng tao đang làm việc với nó.
plt.subplot(2, 1, 1)

# Vẽ đồ thị đầu tiên
plt.plot(x, y_sin)
plt.title('Sine')

# Set đồ thị thứ 2 active, và vẽ.
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')
```

```
# Hiển thị.  
plt.show()
```



Hình 5. 14. Minh hoạ vẽ biểu đồ trên cùng 1 hình

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.density.html>

5.14. Lưu hình ảnh vừa vẽ

```
### bitmap format  
plt.plot(x, sinus)  
plt.savefig("sinus.png")  
plt.close()  
# Prefer vectorial format (SVG: Scalable Vector Graphics)  
# can be edited with  
# Inkscape, Adobe Illustrator, Blender, etc.  
plt.plot(x, sinus)  
plt.savefig("sinus.svg")  
plt.close()  
# Or pdf  
plt.plot(x, sinus)  
plt.savefig("sinus.pdf")  
plt.close()
```

Tóm tắt chương 5

Chương này trình bày cách vẽ các biểu đồ cơ bản của thống kê ứng dụng với Python.

Bắt đầu là các mẫu trong module Matplotlib để vẽ các biểu đồ cơ bản của thống kê ứng dụng. Tiếp theo chúng ta minh họa qua các ví dụ cho các biểu đồ histogram, thanh, boxplot, biểu đồ đường, biểu đồ tròn....

Câu hỏi ôn tập và bài tập chương 5

1. Thực hiện các mẫu cơ bản để biểu đồ, lưu lại để thực hiện ở các bài tập 4 và 5.
2. Thực hiện cài đặt các ví dụ minh họa trong chương, báo cáo kết quả thực hiện.
3. Giải thích các câu lệnh cơ bản để thực hiện trong các đoạn mã lệnh của các ví dụ
4. Thực hiện các ví dụ cho các dữ liệu có tính thực tiễn khác.
5. Đọc dữ liệu ở file `nghesi.xlsx`. theo link sau:
<https://docs.google.com/spreadsheets/d/1q7RVfoTKEuLP0Ud0a50mJu7e1-YJJbUe/edit?usp=sharing&ouid=106843557461060771783&rtpof=true&sd=true>
 - 5.1. Hiển thị dữ liệu trong file theo các dạng biểu đồ thích hợp và đưa ra các bảng thống kê tần suất, bảng phân tổ của dữ liệu trong file.
 - 5.2. Xét xem dữ liệu có phân phối chuẩn hay không (tức biểu đồ histogram có dạng hình quả chuông úp ngược)?

CHƯƠNG 6. MỘT SỐ PHÂN PHỐI XÁC SUẤT QUAN TRỌNG TRONG THỐNG KÊ

Mục đích

Trình bày một số biến ngẫu nhiên với phân phối xác suất quan trọng trong thống kê ứng dụng.

(Một số mã nguồn trong chương này tham khảo từ [4], [6])

Yêu cầu

- Giải thích được ý nghĩa của hàm PMF, CDF cho biến ngẫu nhiên rời rạc và ý nghĩa của hàm PDF cho biến ngẫu nhiên liên tục
- Cài đặt thực hiện các ví dụ minh họa cho các biến ngẫu nhiên có phân phối xác suất quan trọng trong thống kê.
- Giải thích được ý nghĩa của các cơ bản trong các ví dụ minh họa về hàm PDF, CDF, PMF của các biến ngẫu nhiên trong các ví dụ minh họa.

6.1. Hàm khối lượng xác suất và hàm phân phối tích lũy

Hàm khối lượng xác suất (PMF) của một biến ngẫu nhiên X rời rạc là một hàm xác định xác suất để X nhận giá trị x : $P(X=x)$.

PMF được xác định bởi công thức sau:

$$p_X(x) = P[X = x]$$

Tập hợp tất cả các trạng thái có thể có của X được ký hiệu là Ω .

Ví dụ:

Tung một đồng xu 2 lần, khi đó không gian mẫu là: $\Omega = \{SS, SN, NS, NN\}$.

Xét biến ngẫu nhiên X = số mặt sấp (mặt S) xảy ra. Khi đó, ta có:

$$X(\text{"SS"}) = 2, X(\text{"NS"}) = 1, X(\text{"SN"}) = 1, X(\text{"NN"}) = 0.$$

Biến ngẫu nhiên X nhận 3 giá trị: 0, 1, 2. Khi đó hàm khối lượng xác suất PMF của X là:

$$\begin{aligned} p_X(0) &= P[X = 0] = P[\{\text{"NN"}\}] = 1/4, \\ p_X(1) &= P[X = 1] = P[\{\text{"NS"}, \text{"SN"}\}] = 1/2, \\ p_X(2) &= P[X = 2] = P[\{\text{"SS"}\}] = 1/4 \end{aligned}$$

Biểu đồ histogram là biểu đồ thể hiện tần suất của các trạng thái của biến ngẫu nhiên. Trục x là tập hợp các trạng thái, trong khi trục y là tần số. Vì vậy, PMF có mối quan hệ chặt chẽ biểu đồ histogram của biến ngẫu nhiên rời rạc.

Cho biến ngẫu nhiên rời rạc X với không gian mẫu: $\Omega = \{x_1, x_2, \dots\}$.

Hàm phân phối tích lũy của X (**cumulative distribution function - CDF**)

$$F_X(x_k) \stackrel{\text{def}}{=} \mathbb{P}[X \leq x_k] = \sum_{\ell=1}^k p_X(x_\ell).$$

Nếu $\Omega = \{\dots, -1, 0, 1, 2, \dots\}$, khi đó CDF của X là:

$$F_X(k) \stackrel{\text{def}}{=} \mathbb{P}[X \leq k] = \sum_{\ell=-\infty}^k p_X(\ell).$$

Đoạn mã sau minh họa việc vẽ đồ thị của hàm PMF và CDF của biến ngẫu nhiên X có không gian mẫu là: $[0, 1, 4]$, với xác suất nhận được trạng thái tương ứng là: $0.25, 0.5, 0.25$. Tức $P(X=0) = P(X=4) = 0.25$ và $P(X=1) = 0.5$ và

```
import numpy as np
import matplotlib.pyplot as plt
p = np.array([0.25, 0.5, 0.25])
x = np.array([0, 1, 4])
F = np.cumsum(p)
plt.stem(x, p, use_line_collection=True); plt.show()
plt.step(x, F); plt.show()
```

<https://numpy.org/doc/stable/reference/generated/numpy.cumsum.html>

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.step.html

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.stem.html

Trong đoạn mã trên, chúng ta sử dụng hàm `cumsum()` của `numpy` để tính CDF.

Minh họa về giá trị kỳ vọng

```
# Python code to compute the mean of a dataset
import numpy as np
X = np.random.rand(10000)
mX = np.mean(X)

# Python code to compute the expectation
```

```
import numpy as np

p = np.array([0.25, 0.5, 0.25])
x = np.array([0, 1, 2])
EX = np.sum(p*x)
```

Ví dụ: Cho biến ngẫu nhiên X với hàm PMF là: $p_X(k) = 1/(2^k)$, với $k = 1, 2, 3, \dots$

Khi đó giá trị kỳ vọng của X :

$$\begin{aligned}\mathbb{E}[X] &= \sum_{k=1}^{\infty} k p_X(k) = \sum_{k=1}^{\infty} k \cdot \frac{1}{2^k} \\ &= \frac{1}{2} \sum_{k=1}^{\infty} k \cdot \frac{1}{2^{k-1}} = \frac{1}{2} \cdot \frac{1}{(1 - \frac{1}{2})^2} = 2.\end{aligned}$$

Được minh họa cách tính qua đoạn mã sau với Python, Ở đây, chúng ta xấp xỉ tổng vô hạn bằng một tổng hữu hạn của $k = 1, \dots, 100$.

```
import numpy as np

k = np.arange(100)
p = np.power(0.5, k)
EX = np.sum(p*k)

print(EX)
```

Minh họa về phương sai

```
% Python code to compute the variance

import numpy as np

X = np.random.rand(10000)

vX = np.var(X)

sX = np.std(X)
```

<https://numpy.org/doc/stable/reference/generated/numpy.std.html>

<https://numpy.org/doc/stable/reference/generated/numpy.var.html>

6.2. Minh hoạ biến ngẫu nhiên có phân phối Bernoulli

Biến ngẫu nhiên Bernoulli

Biến ngẫu nhiên có hai trạng thái: 1 hoặc 0. Xác suất nhận 1 là p và xác suất nhận 0 là $1 - p$.

Hàm phân phối khối: PMF của X là:

$$p_X(0) = 1 - p, p_X(1) = p,$$

Ở đây $0 < p < 1$ được gọi là tham số Bernoulli.

Trong Python, việc tạo biến ngẫu nhiên Bernoulli có thể được thực hiện bằng cách gọi trình tạo số ngẫu nhiên nhị thức `np.random.binomial`. Khi tham số n bằng 1, biến ngẫu nhiên nhị thức tương đương với biến ngẫu nhiên Bernoulli. Dưới đây là đoạn mã Python để minh hoạ biến ngẫu nhiên Bernoulli. (xem [4], [6])

```
# Python code to generate 1000 Bernoulli random variables
import numpy as np
import matplotlib.pyplot as plt

p = 0.5
n = 1
X = np.random.binomial(n,p,size=1000)
plt.hist(X,bins='auto')
plt.show()
```

Trong Python có thể gọi phương thức `stats.bernoulli.rvs()` của thư viện `scipy`, để tạo dãy Bernoulli ngẫu nhiên.

```
# Python code to call scipy.stats library
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

p = 0.5
X = stats.bernoulli.rvs(p,size=1000)
plt.hist(X,bins='auto')
plt.show()
```

Trong Python, ta có thể tạo một đối tượng `rv`. Sau đó chúng ta có thể gọi các thuộc tính của `rv` để xác định giá trị trung bình, phương sai, v.v.

```
# Python code to generate a Bernoulli rv object
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

p = 0.5

rv = stats.bernoulli(p)

mean, var = rv.stats(moments='mv')

print(mean, var)
```

Với `rv`, chúng ta cũng có thể gọi thực hiện tính hàm khối xác suất PMF của nó là `rv.pmf()`:

```
# Python code to plot the PMF of a Bernoulli
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

p = 0.3

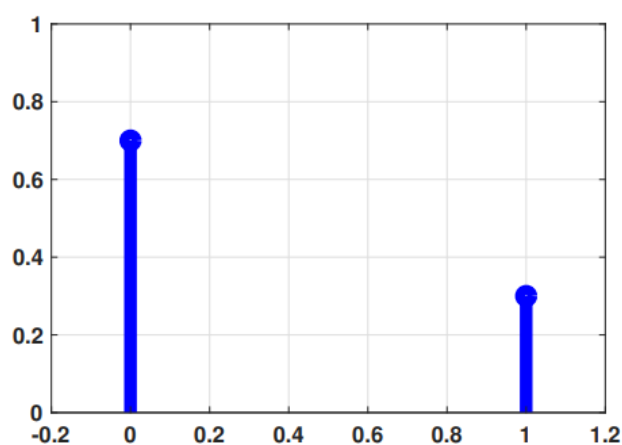
rv = stats.bernoulli(p)

x = np.linspace(0, 1, 2)

f = rv.pmf(x)

plt.plot(x, f, 'bo', ms=10)

plt.vlines(x, 0, f, colors='b', lw=5, alpha=0.5)
```



Hình 6. 1. Đồ thị PMF của biến ngẫu nhiên có phân phối Bernoulli

6.3. Biến ngẫu nhiên nhị thức

Giả sử chúng ta tung đồng xu n lần và đếm số lần xuất hiện mặt sấp. Vì mỗi lần lật xu là một biến ngẫu nhiên (Bernoulli), nên tổng của chúng cũng là một biến ngẫu nhiên. Biến ngẫu nhiên mới này được gọi là biến ngẫu nhiên nhị thức.

Cho X là một biến ngẫu nhiên nhị thức. Khi đó, PMF của X là:

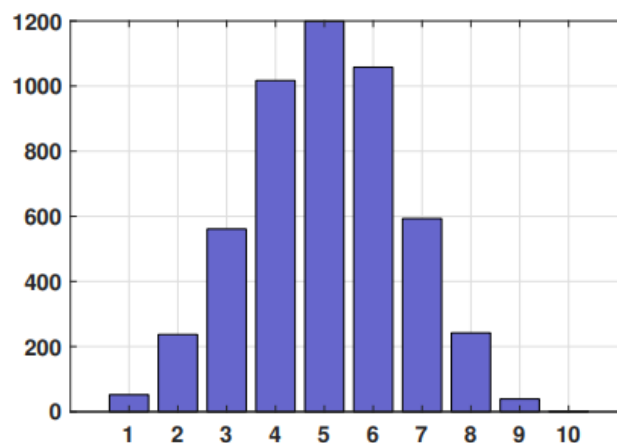
$$p_X(k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, \dots, n,$$

trong đó $0 < p < 1$ là tham số nhị thức và n là tổng số trạng thái. Ta viết: $X \sim \text{Binomial}(n, p)$, để nói rằng X được rút ra từ phân phối nhị thức với tham số p có kích thước n .

Biểu đồ histogram của biến ngẫu nhiên nhị thức được thể hiện như hình dưới đây. Ở đây, chúng ta xem xét ví dụ trong đó $n = 10$ và $p = 0,5$. Để tạo biểu đồ, chúng tôi sử dụng 5000 mẫu.

Trong Python, việc tạo biến ngẫu nhiên nhị thức như trong hình dưới đây có thể được thực hiện bằng cách gọi `binornd` và `np.random.binomial`.

```
# Python code to generate 5000 Binomial random variables
import numpy as np
import matplotlib.pyplot as plt
p = 0.5
n = 10
X = np.random.binomial(n, p, size=5000)
plt.hist(X, bins='auto')
```



(a) Histogram based on 5000 samples

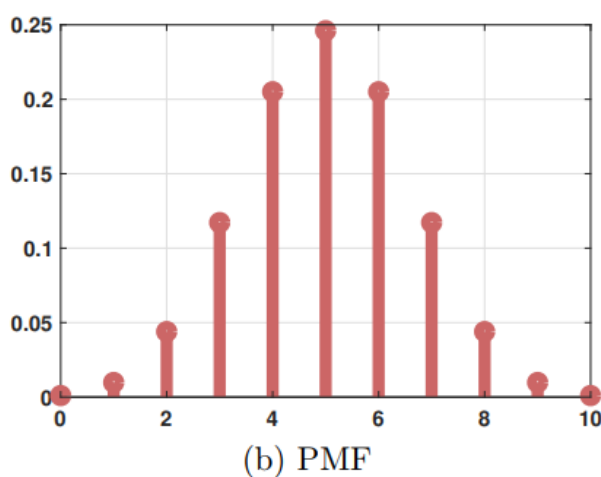
Hình 6. 2. Biểu đồ histogram của biến ngẫu nhiên có phân phối nhị thức

```
# Python code to generate a binomial PMF
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

p = 0.5
n = 10

rv = stats.binom(n,p)
x = np.arange(11)
f = rv.pmf(x)

plt.plot(x, f, 'bo', ms=10)
plt.vlines(x, 0, f, colors='b', lw=5, alpha=0.5)
```



Hình 6. 3. Đồ thị PMF của biến nhị phân có phân phối nhị thức

Trong Python, chúng ta tạo ra đối tượng `rv = stats.binom (n, p)` sau đó gọi `rv.stats` để tính mean và phương sai của biến nhị thức.

```
# Python code to compute the mean and var of a binomial rv
import scipy.stats as stats

p = 0.5
n = 10

rv = stats.binom(n,p)

M, V = rv.stats(moments='mv') # lần lượt tính mean và variance

print(M, V)
```

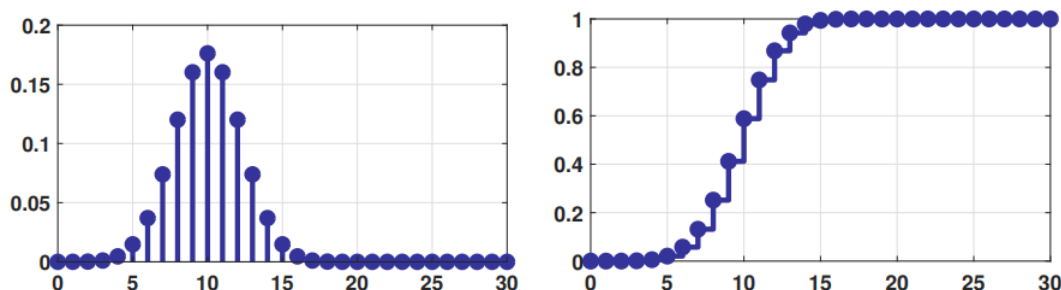
Để tính toán các giá trị của hàm cdf. Trong Python, lệnh tương ứng là `rv = stats.binom(n, p)` tiếp theo là `rv.cdf`.

```
# # Python code to plot CDF of a binomial random variable
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

p = 0.5
n = 10

rv = stats.binom(n,p)
x = np.arange(11)
F = rv.cdf(x)

plt.plot(x, F, 'bo', ms=10)
plt.vlines(x, 0, F, colors='b', lw=5, alpha=0.5)
```



Hình 6. 4. Đồ thị của hàm PMF và CDF của biến nhị thức với tham số n, p

6.4. Biến ngẫu nhiên hình học

Trong một số ứng dụng, chúng ta cần quan tâm đến việc thử một thử nghiệm nhị phân (phép thử chỉ có 2 trạng thái đúng/sai, có/không, 0/1...) cho đến khi chúng ta nhận được 1 trạng thái mong đợi. Ví dụ: chúng ta muốn tiếp tục gọi điện thoại cho ai đó cho đến khi người đó bắt máy. Trong trường hợp này, biến ngẫu nhiên để mô tả số lần cần thực hiện cho đó khi nhận được thành công. Biến ngẫu nhiên như vậy được gọi là biến ngẫu nhiên hình học.

Cho X là một biến ngẫu nhiên hình học. Khi đó, PMF của X là:

$$p_X(k) = (1 - p)^{k-1}p, \quad k = 1, 2, \dots,$$

trong đó: $0 < p < 1$ được gọi là tham số hình học. Chúng ta viết: $X \sim \text{Geometric}(p)$

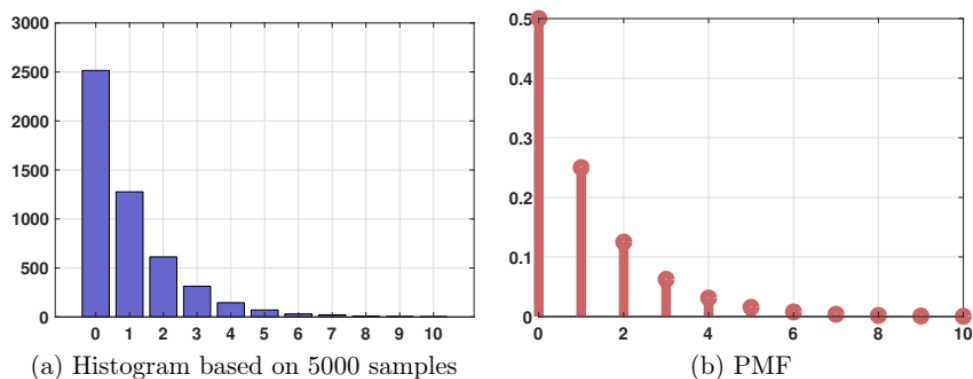
để nói rằng X có phân phối hình học với tham số p .

Trong Python, việc tạo các biến ngẫu nhiên hình học có thể được thực hiện bởi `np.random.geometric`.

```
# Python code to generate 1000 geometric random variables
import numpy as np
import matplotlib.pyplot as plt
p = 0.5
X = np.random.geometric(p, size=1000)
plt.hist(X, bins='auto')
```

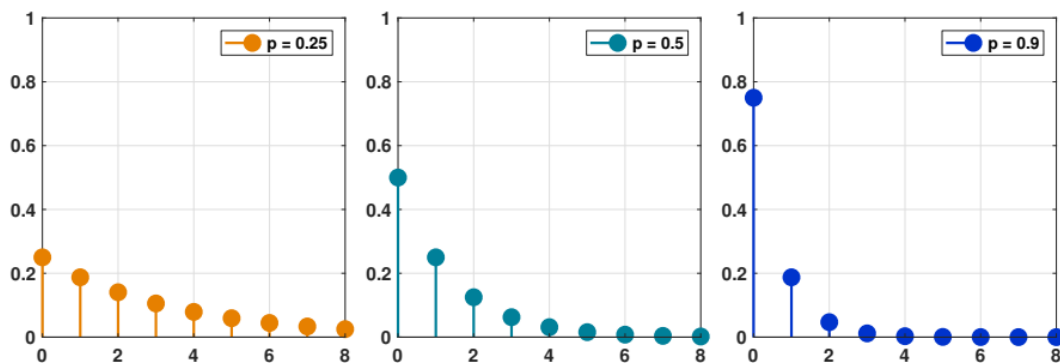
Để tính toán hàm PMF, trong Python, chúng ta gọi `rv = stats.geom` của thư viện `scipy` theo sau là `rv.pmf`.

```
# Python code to generate geometric PMF
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
x = np.arange(1, 11)
p = 0.5
rv = stats.geom(p)
f = rv.pmf(x)
plt.plot(x, f, 'bo', ms=8, label='geom pmf')
plt.vlines(x, 0, f, colors='b', lw=5, alpha=0.5)
```



Hình 6. 5. Biểu đồ histogram và đồ thị hàm PMF của biến ngẫu nhiên hình học (geometric) với $p = 0,5$

Thử so sánh hình dạng của PMF với các giá trị khác nhau của p . Trong hình dưới đây chúng ta thay đổi tham số p lần lượt là 0,25; 0,5; 0,9. Đối với p nhỏ, hàm PMF bắt đầu với giá trị thấp và phân rã với tốc độ chậm. Điều ngược lại xảy ra đối với p lớn, khi đó hàm PMF bắt đầu với giá trị cao và giảm nhanh chóng.



6.5. Biến ngẫu nhiên Poisson

Trong nhiều hệ thống vật lý, sự xuất hiện của các sự kiện thường được mô hình hóa dưới dạng biến ngẫu nhiên Poisson, ví dụ, lượt đến của photon, sự phát xạ điện tử và lượt đến của cuộc gọi điện thoại. Trong mạng xã hội, số lượng cuộc trò chuyện trên mỗi người dùng cũng có thể được mô hình hóa dưới dạng Poisson. Trong thương mại điện tử, số lượng giao dịch trên mỗi người dùng trả tiền, được mô hình hoá bằng mô hình Poisson.

Cho X là một biến ngẫu nhiên Poisson. Khi đó, PMF của X là:

$$p_X(k) = \frac{\lambda^k}{k!} e^{-\lambda}, k = 0, 1, 2$$

trong đó $\lambda > 0$ gọi là tham số Poisson. Chúng ta viết: $X \sim \text{Poisson}(\lambda)$

để nói rằng X có phân phối Poisson với tham số λ .

Trong định nghĩa này, tham số λ xác định tốc độ đến.

```
# Python code to generate 5000 Poisson random variables
and it's hitogram

import numpy as np

import matplotlib.pyplot as plt

lamdb = 1

X = np.random.poisson(lamdb, size=5000)

plt.hist(X, bins='auto')
```

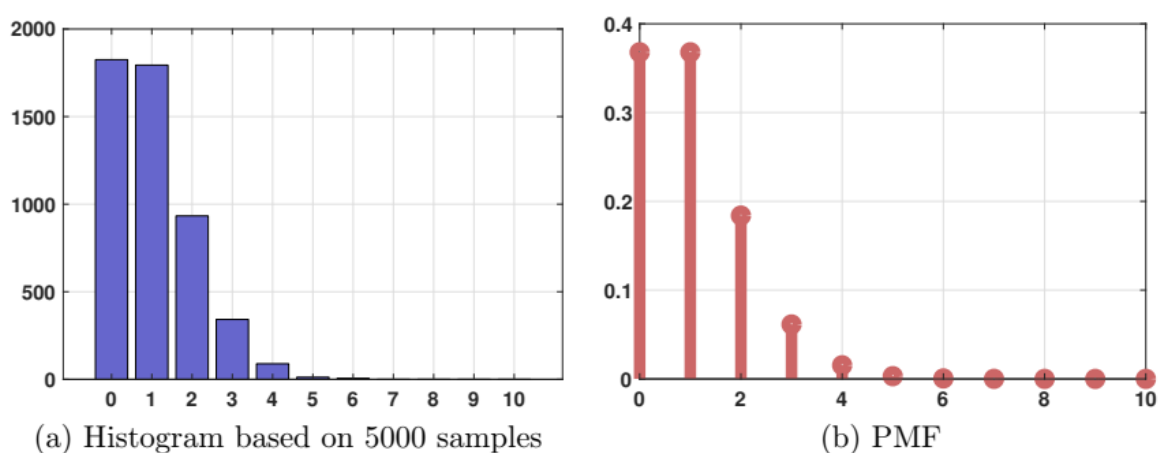
Đối với PMF, trong Python, chúng ta có thể gọi `rv.pmf` với `rv = stats.poisson`, trong thư viện `scipy`.

```
# Python code to plot the Poisson PMF

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

x = np.arange(0,11)
rv = stats.poisson(lambd)
f = rv.pmf(x)

plt.plot(x, f, 'bo', ms=8, label='geom pmf')
plt.vlines(x, 0, f, colors='b', lw=5, alpha=0.5)
```



Hình 6. 6. Biểu đồ histogram và đồ thị hàm PMF của biến ngẫu nhiên có phân phối Poisson

Biểu đồ histogram và hàm PMF của biến ngẫu nhiên phân phối Poisson với tham số $\lambda=1$

Hàm CDF của biến ngẫu nhiên Poisson là:

$$F_X(k) = \mathbb{P}[X \leq k] = \sum_{\ell=0}^k \frac{\lambda^\ell}{\ell!} e^{-\lambda}.$$

Để tính giá trị trung bình và phương sai của một biến ngẫu nhiên Poisson, chúng ta có thể gọi `rv.stats (moment = 'mv')` với thư viện `scipy`.

```
# Python code to compute Poisson statistics

import scipy.stats as stats

lambd = 1

rv = stats.poisson(lambd)

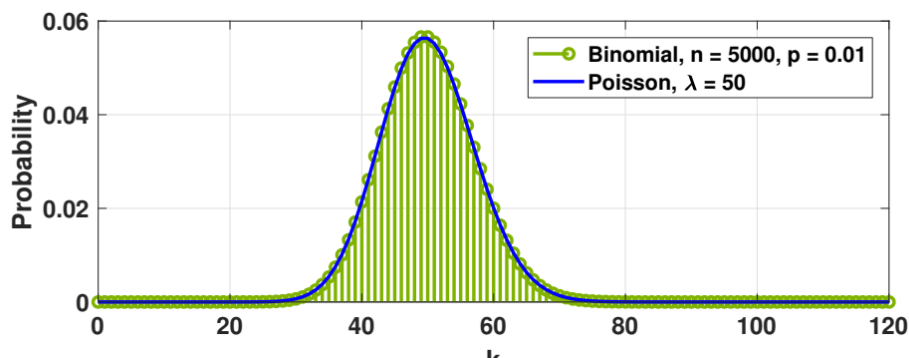
M, V = rv.stats(moments='mv')
```


Định lý: Với p đủ nhỏ và n đủ lớn ta có:

$$\binom{n}{k} p^k (1-p)^{n-k} \approx \frac{\lambda^k}{k!} e^{-\lambda}$$

Ở đây: $\lambda = n.p$

```
# Python code to approximate binomial using Poisson
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
n = 1000; p = 0.05
rv1 = stats.binom(n,p)
X = rv1.rvs(size=10000)
plt.figure(1); plt.hist(X, bins=np.arange(0,100));
rv2 = stats.poisson(n*p)
f = rv2.pmf(bin)
plt.figure(2); plt.plot(f)
```



Hình 6. 7. Xấp xỉ biến ngẫu nhiên nhị thức bởi biến ngẫu nhiên Poisson

BIẾN NGẪU NHIÊN LIÊN TỤC

6.6. Hàm mật độ xác suất

Đối với biến ngẫu nhiên liên tục, ta có khái niệm quan trọng hàm mật độ xác suất (probability density function – hàm PDF).

Định nghĩa: Hàm mật độ xác suất f_X của biến ngẫu nhiên X là ánh xạ $f_X : \Omega \rightarrow \mathbb{R}$, thoả tính chất:

$$\mathbb{P}[\{x \in A\}] = \int_A f_X(x) dx$$

^ **Không âm:** $f_X(x) \geq 0$ với mọi $x \in \Omega$

^ **Đơn vị:** $\int_{\Omega} f_X(x) dx = 1$

Ở đây: Ω là không gian mẫu.

Cho X là biến ngẫu nhiên liên tục. Hàm mật độ xác suất (PDF) của X là một hàm $f_X : \Omega \rightarrow \mathbb{R}$ khi đó ta có:

$$\mathbb{P}[a \leq X \leq b] = \int_a^b f_X(x) dx.$$

Hàm mật độ xem như là một mở rộng của hàm khối xác suất (PMF) cho biến ngẫu nhiên liên tục.

Chúng ta tích phân của hàm PDF để tính xác suất, thay cho tổng trong biến ngẫu nhiên rời rạc.

Hàm phân phối lũy tích (CDF)

Cho X là biến ngẫu nhiên liên tục trên không gian mẫu $\Omega = \mathbb{R}$.

Khi đó hàm phân phối lũy tích (CDF) của X là:

$$F_X(x) \stackrel{\text{def}}{=} \mathbb{P}[X \leq x] = \int_{-\infty}^x f_X(x') dx'.$$

Các khái niệm quan trọng của biến ngẫu nhiên liên tục như giá trị kỳ vọng, moment cấp k ; phương sai, mode... được tính toán dựa trên hàm PDF và CDF có thể tìm đọc ở các tài liệu xác suất thống kê chuyên ngành như [3], [4].

Mối quan hệ giữa PDF, CDF xem [3], trang 193

Hàm mật độ xác suất (PDF) là đạo hàm của hàm phân phối tích lũy (CDF. Hơn nữa, diện tích dưới đường cong của PDF giữa âm vô cùng và x bằng giá trị của x của CDF.

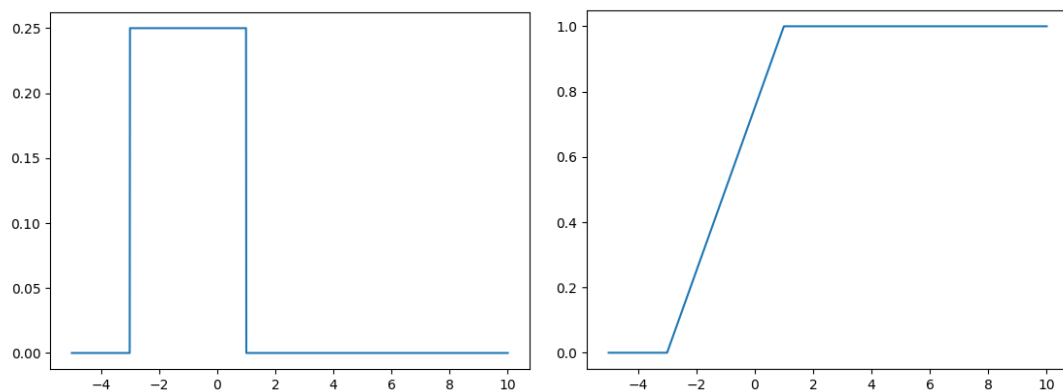
6.7. Biến ngẫu nhiên có phân phối đều

Phân phối đều là một dạng phân phối xác suất thống kê trong đó tất cả các kết quả đều có khả năng xảy ra như nhau, mỗi biến có cùng một xác suất để cho ra một kết quả.

```
# Python code to generate the PDF and CDF of uniform
varibale

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

x = np.linspace(-5,10,1500)
f = stats.uniform.pdf(x,-3,4)
F = stats.uniform.cdf(x,-3,4)
plt.plot(x,f); plt.show()
plt.plot(x,F); plt.show()
```



Hình 6. 8. PDF và CDF của biến ngẫu nhiên phân phối đều

```
# Python code to compute empirical mean, var, median, mode
of uniform varibale

X = stats.uniform.rvs(a,b,size=1000)
M = np.mean(X)
V = np.var(X)
Med = np.median(X)
Mod = stats.mode(X)

# Python code to compute the probability  $P(0.2 < X < 0.3)$ 
of uniform varibale

a = 0; b = 1;
rv = stats.uniform(a,b)
F = rv.cdf(0.3)-rv.cdf(0.2)
```

6.8. Biến ngẫu nhiên có phân phối mũ

Phân phối biểu diễn xác suất thời gian giữa các lần một sự kiện xảy ra. Biến ngẫu nhiên X tuân theo phân phối mũ $X \sim \text{Exp}(\lambda)$ với tham số λ là tỉ lệ xảy ra của sự kiện A .

Hàm phân phối xác suất của X : $f_X(x) = \lambda e^{-\lambda x}$ với $x \geq 0$, ngược lại $f_X(x) = 0$.

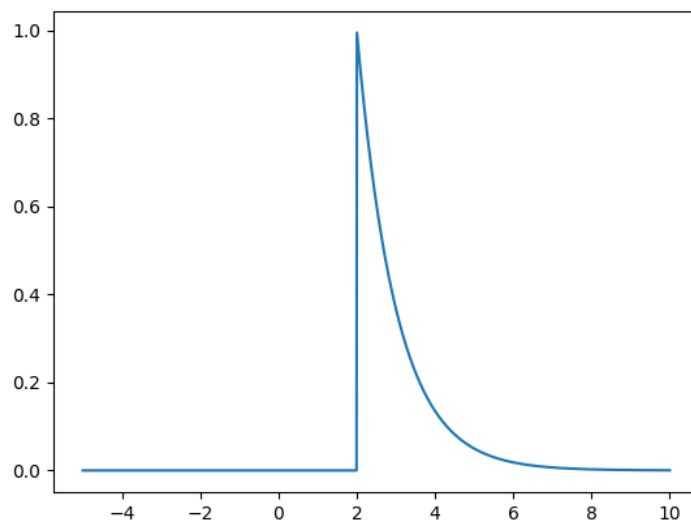
Hàm phân phối tích lũy (CDF) của X là $F(x; \lambda) = 1 - e^{-\lambda x}$

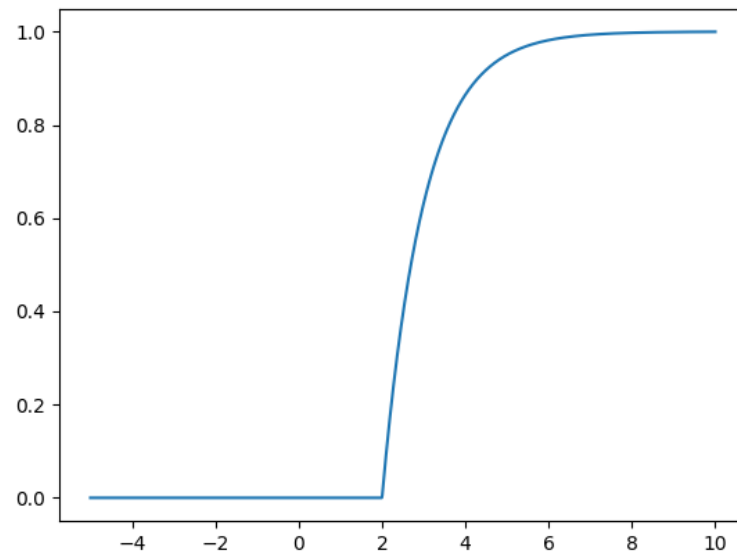
Kỳ vọng $E[X] = 1/\lambda$

Phương sai: $\text{Var}(X) = 1/\lambda^2$

```
# Python code to generate the PDF and CDF of Exponential
random variable
```

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
x = np.linspace(-5, 10, 1500)
f = stats.expon.pdf(x, 2)
F = stats.expon.cdf(x, 2)
plt.plot(x, f); plt.show()
plt.plot(x, F); plt.show()
```

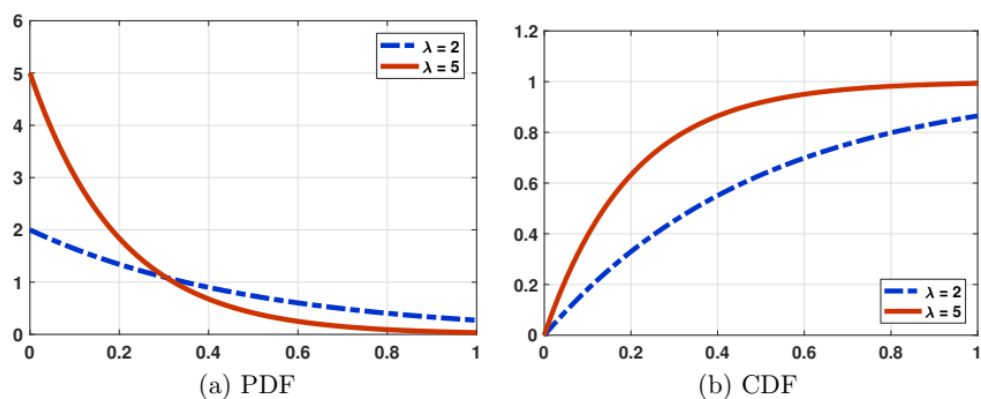




Hình 6. 9. Đồ thị của hàm PDF và CDF của biến ngẫu nhiên phân phối mũ

```
# Python code to plot the exponential PDF
lamdb1 = 1/2
lamdb2 = 1/5
x = np.linspace(0,1,1000)
f1 = stats.expon.pdf(x,scale=lamdb1)
f2 = stats.expon.pdf(x,scale=lamdb2)
plt.plot(x, f1)
plt.plot(x, f2)
```

```
# Python code to plot the exponential CDF
F = stats.expon.cdf(x,scale=lamdb1)
plt.plot(x, F)
```



6.9. Biến ngẫu nhiên Gaussian

Phân phối chuẩn - Normal distribution

Phân phối chuẩn hay còn được gọi là phân phối *Gauss* là một trong những phân phối quan trọng nhất và được ứng dụng rất rộng rãi trong thực tế.

Biến ngẫu nhiên X tuân theo phân phối chuẩn $X \sim N(\mu, \sigma^2)$ với tham số kỳ vọng μ và phương sai σ^2 , ta có:

Hàm phân phối xác suất của X :

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\}$$

Hàm phân phối tích lũy của X :

$$F_X(x) = \Phi \left(\frac{x - \mu}{\sigma} \right)$$

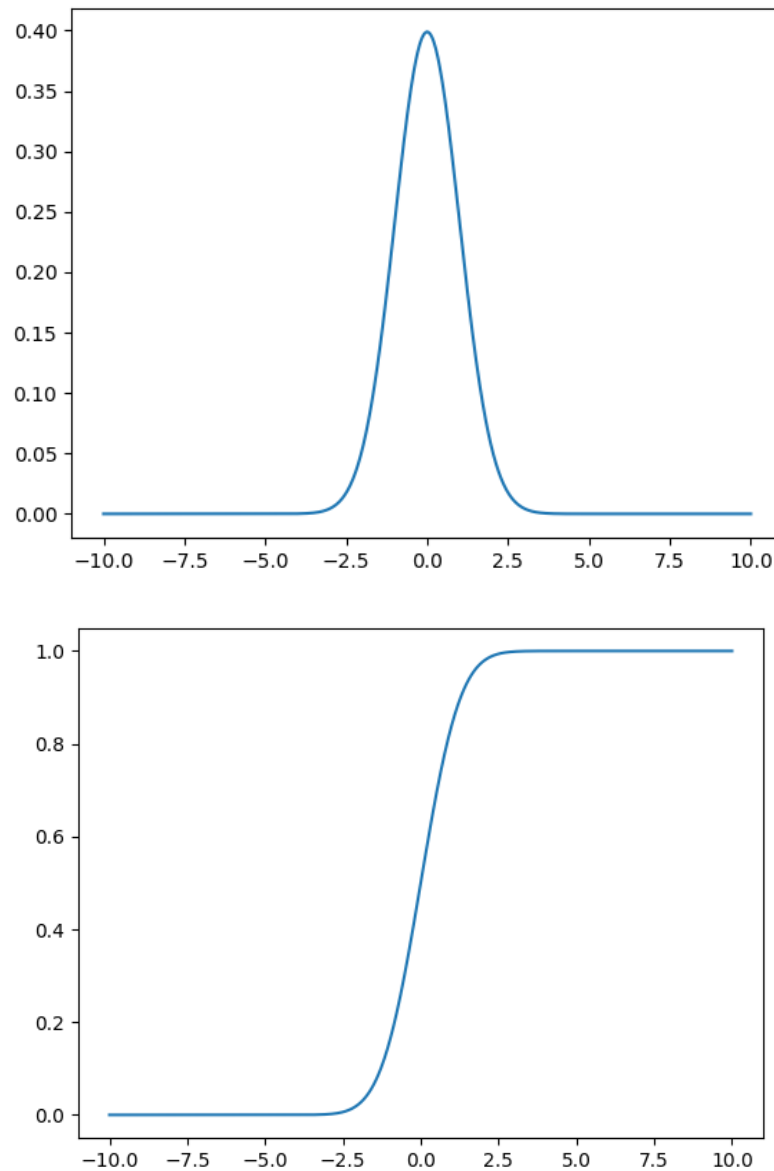
Ở đây, hàm Φ được tính dựa trên hàm lỗi (error function) được định nghĩa như sau:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

và ta có:

$$\Phi(x) = \frac{1}{2} \left[1 + \text{erf} \left(\frac{x}{\sqrt{2}} \right) \right]$$

```
# Python code to generate standard Gaussian PDF and CDF
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
x = np.linspace(-10,10,1000)
f = stats.norm.pdf(x)
F = stats.norm.cdf(x)
plt.plot(x,f); plt.show()
plt.plot(x,F); plt.show()
```



Hình 6. 10. Minh họa PDF and CDF của biến ngẫu nhiên phân phối chuẩn Gaussian

```
# Python code to verify standardized Gaussian
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
x = np.linspace(-5,5,1000)
mu = 3; sigma = 2;
f1 = stats.norm.pdf((x-mu)/sigma,0,1) # standardized
f2 = stats.norm.cdf(x,mu,sigma) # raw
```

Đối với một biến ngẫu nhiên X với hàm mật độ xác suất là $f_X(x)$, các khái niệm sau mô tả mức độ hướng trung tâm (**central moments**) của dữ liệu:

$$mean = \mathbb{E}[X] \stackrel{\text{def}}{=} \mu$$

$$variance = \mathbb{E}[(X - \mu)^2] \stackrel{\text{def}}{=} \sigma^2$$

$$skewness = \mathbb{E}\left[\left(\frac{X - \mu}{\sigma}\right)^3\right] \stackrel{\text{def}}{=} \gamma$$

$$kurtois = \mathbb{E}\left[\left(\frac{X - \mu}{\sigma}\right)^4\right] \stackrel{\text{def}}{=} \kappa, \text{ ngoài ra có thể } kurtois \stackrel{\text{def}}{=} \kappa - 3$$

Ví dụ về tính toán skewness and kurtosis của biến ngẫu nhiên phân phối gamma distribution với thư viện scipy.

```
# Python code to compute skewness and kurtosis of a gamma distribution
```

```
import scipy.stats as stats
X = stats.gamma.rvs(3, 5, size=10000)
s = stats.skew(X)
k = stats.kurtosis(X)
```

Định lý giới hạn trung tâm

Cho $X_1, X_2, \dots, X_n, \dots$ là một dãy các biến ngẫu nhiên độc lập cùng tuân theo một quy luật phân phối xác suất với kỳ vọng toán μ và phương sai hữu hạn σ^2 , thì quy luật phân phối xác suất của biến ngẫu nhiên

$$U_n = \frac{S_n - E(S_n)}{\sqrt{V(S_n)}} = \frac{S_n - n\mu}{\sqrt{n\sigma^2}} \text{ với } S_n = \sum_{k=1}^n X_k$$

sẽ hội tụ tới biến ngẫu nhiên có phân phối chuẩn tắc $N(0,1)$ khi $n \rightarrow \infty$

Nói cách khác, một biến ngẫu nhiên được định nghĩa như trung bình của một số lớn của các biến ngẫu nhiên độc lập và phân phối đồng nhất là xấp xỉ phân phối chuẩn.

Cụ thể, nếu x_1, x_2, \dots, x_n là các biến ngẫu nhiên với μ, σ và số n đủ lớn thì:

$$\frac{1}{n} \cdot (x_1 + x_2 + \dots + x_n)$$

là giá trị xấp xỉ phân phối chuẩn với mean μ và độ lệch chuẩn là σ/\sqrt{n}

Với $\mu = 0$ và $\sigma = 1$, ta có:

$$\frac{(x_1 + \dots + x_n) - \mu n}{\sigma\sqrt{n}}$$

là giá trị xấp xỉ phân phối chuẩn tắc.

Tầm quan trọng thực tiễn của định lý giới hạn trung tâm là định lý chỉ ra phân phối chuẩn có thể được sử dụng như là một xấp xỉ cho một số dạng phân phối khác. Dưới đây sẽ minh họa cho điều này là các biến ngẫu nhiên nhị thức (binomial) với hai tham số là số lớn n và xác suất p .

Một biến ngẫu nhiên nhị thức $\text{Binomial}(n, p)$ là tổng của n biến ngẫu nhiên Bernoulli(p) độc lập. Mỗi biến ngẫu nhiên Bernoulli(p) sẽ nhận giá trị 1 nếu xác suất là p và nhận giá trị 0 nếu xác suất là $1-p$.

Biến ngẫu nhiên nhị thức $\text{Binomial}(n, p)$ và biến ngẫu nhiên Bernoulli(p) có thể được cài đặt bằng Python như sau:

```
def bernoulli_trial(p):  
    return 1 if random.random() < p else 0  
def binomial(n, p):  
    return sum(bernoulli_trial(p) for _ in range(n))
```

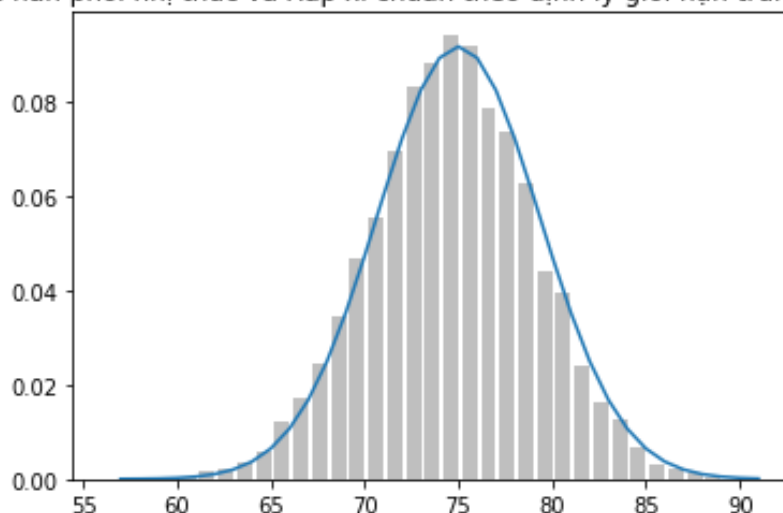
Giá trị μ của biến Bernoulli(p) là p và σ là $\sqrt{p(1-p)}$ vì vậy, áp dụng định lý giới hạn trung tâm khi n đủ lớn, một biến $\text{Binomial}(n, p)$ là xấp xỉ của một biến ngẫu nhiên chuẩn với μ là np và σ là: $\sqrt{np(1-p)}$.

Chúng ta có thể thấy kết quả trên một cách trực quan bằng đoạn mã Python hoàn chỉnh sau:

```
import random  
import matplotlib.pyplot as plt  
import math  
from collections import Counter
```

```
def normal_cdf(x, mu=0, sigma=1):  
    return (1 + math.erf((x - mu) / math.sqrt(2) / sigma)) / 2  
  
def bernoulli_trial(p):  
    return 1 if random.random() < p else 0  
  
def binomial(n, p):  
    return sum(bernoulli_trial(p) for _ in range(n))  
  
def make_hist(p, n, num_points):  
    data = [binomial(n, p) for _ in range(num_points)]  
    # các biểu đồ cột là các nhị thức  
    histogram = Counter(data)  
    plt.bar([x - 0.4 for x in histogram.keys()],  
            [v / num_points for v in  
histogram.values()], 0.8, color='0.75')  
    mu = p * n  
    sigma = math.sqrt(n * p * (1 - p))  
    # đường cong là xấp xỉ chuẩn  
    xs = range(min(data), max(data) + 1)  
    ys = [normal_cdf(i + 0.5, mu, sigma) - normal_cdf(i - 0.5,  
mu, sigma) for i in xs]  
    plt.plot(xs, ys)  
    plt.title("Phân phối nhị thức và Xấp xỉ chuẩn theo định lý  
giới hạn trung tâm")  
    plt.show()  
make_hist(0.75, 100, 10000)
```

Phân phối nhị thức và Xấp xỉ chuẩn theo định lý giới hạn trung tâm



Hình 6. 11. Phân phối nhị thức và xấp xỉ phân phối chuẩn theo định lý giới hạn trung tâm

6.10. Sinh số ngẫu nhiên

Hầu hết các phần mềm máy tính ngày nay đều có bộ tạo số ngẫu nhiên được tích hợp sẵn. Đối với các loại biến ngẫu nhiên phổ biến, ví dụ, phân phối Gauss hoặc hàm mũ, các chương trình tạo số ngẫu nhiên này có thể dễ dàng tạo ra các số theo phân phối đã chọn. Tuy nhiên, nếu chúng ta được cho một phân phối PDF bất kỳ (hoặc PMF) không nằm trong danh sách các phân phối cho trước, làm thế nào để có thể tạo các số ngẫu nhiên theo PDF hoặc PMF mà chúng ta muốn?

Để tạo số ngẫu nhiên có phân phối tùy ý F_X

- Bước 1: Tạo một số ngẫu nhiên $U \sim \text{Uniform}(0, 1)$.

- Bước 2: Cho $Y = F_X^{-1}(U)$.

Khi đó phân phối của Y là F_X .

```
# Python code to generate Gaussian from uniform
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

mu = 3
sigma = 2

U = stats.uniform.rvs(0,1,size=10000)
gU = sigma*stats.norm.ppf(U)+mu
```

```
plt.hist(U); plt.show()

plt.hist(gU); plt.show()

# Python code to generate exponential random variables
import numpy as np
import scipy.stats as stats
lamdb = 1;
U = stats.uniform.rvs(0,1,size=10000)
gU = -(1/lamdb)*np.log(1-U)

# Python code to generate the desired random variables
import numpy as np
import scipy.stats as stats
U = stats.uniform.rvs(0,1,size=10000)
gU = np.zeros(10000)
gU[np.logical_and(U >= 0.0, U <= 0.1)] = 1
gU[np.logical_and(U > 0.1, U <= 0.6)] = 2
gU[np.logical_and(U > 0.6, U <= 0.9)] = 3
gU[np.logical_and(U > 0.9, U <= 1)] = 4
```

6.11. Phân phối t

6.11.1. Bậc tự do

Bậc tự do (tiếng Anh: Degrees Of Freedom) là thuật ngữ sử dụng trong thống kê chỉ số lượng các giá trị có thể tự do thay đổi trong một mẫu dữ liệu.

Ví dụ: Xét một mẫu dữ liệu đơn giản bao gồm năm số nguyên dương. Các giá trị có thể là bất kỳ số nào sao cho không có mối quan hệ nào giữa chúng. Về mặt lý thuyết, mẫu dữ liệu này sẽ có năm bậc tự do.

Giả sử biết bốn số trong mẫu là {3, 8, 5, 4} và giá trị trung bình mẫu dữ liệu là 6.

Điều này có nghĩa là số thứ năm phải bằng 10 mà không thể là số nào khác, nó được gọi là không thể tự do thay đổi. Vì vậy, bậc tự do cho mẫu dữ liệu này là 4.

Công thức tính bậc tự do

$$Df = N - 1$$

Trong đó:

- Df là bậc tự do
- N là kích thước mẫu

Bậc tự do thường được sử dụng trong nhiều phương thức kiểm định giả thuyết trong thống kê

6.11.2. Định nghĩa phân phối t

Phân phối t hay còn gọi là **phân phối student** là phân phối mẫu lí thuyết gần đúng với phân phối chuẩn. Phân phối t được sử dụng để thiết lập khoảng tin cậy khi dùng các mẫu nhỏ để ước lượng giá trị trung bình của tổng thể.

Nếu \bar{X} là giá trị trung bình của mẫu và s là độ lệch chuẩn của mẫu, thì khi đó thống kê:

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}} = \frac{\bar{x} - \mu}{SE}$$

với SE là sai số tiêu chuẩn, sẽ là biến ngẫu nhiên có phân phối t.

Chẳng hạn, nếu khoảng tin cậy 95% được tính từ phân phối t cho một mẫu nhỏ, sẽ tạo ra khoảng tin cậy rộng hơn trường hợp khoảng tin cậy 95% tính cho mẫu lớn. Phân phối t thường được sử dụng để xác định mức ý nghĩa cho quá trình kiểm định giả thuyết thống kê.

Nếu phân phối chuẩn tắc có dạng $Z \sim N(0,1)$ thì phân phối t có dạng $T \sim N(0, \nu/(\nu-2))$, trong đó ν là bậc tự do và $\nu/(\nu-2)$ là phương sai. Như vậy khi ν lớn (>30) thì $\nu/(\nu-2)$ gần bằng 1 và T có phân phối chuẩn tắc.

Chúng ta có thể sử dụng hàm **t.rvs(df, size)** để tạo các giá trị ngẫu nhiên từ phân phối t với bậc tự do và cỡ mẫu cụ thể:

```
from scipy.stats import t

#generate random values from t distribution with df=6 and sample
size=10
t.rvs(df=6, size=10)
```

```
array([-3.95799716, -0.01099963, -0.55953846, -1.53420055, -
1.41775611,
      -0.45384974, -0.2767931 , -0.40177789, -0.3602592 ,
0.38262431])
```

Kết quả là một mảng gồm 10 giá trị tuân theo phân phối t với 6 bậc tự do.

6.12. Phân phối Chi - bình phương

Nếu biến ngẫu nhiên X có phân phối chuẩn tắc ($X \in N(0,1)$), khi đó X^2 có phân phối Chi-square χ^2 với 1 bậc tự do. Tổng của n biến ngẫu nhiên có phân phối chuẩn tắc, sẽ là một biến ngẫu nhiên có phân phối Chi-square với n bậc tự do.

Phân phối Chi – square thỏa:

- Giá trị trung bình của phân phối chính là bậc tự do n ($\mu = n$).
- Phương sai bằng 2 lần bậc tự do: $\sigma^2 = 2n$
- Trị cực đại (mode) khi $x^2 = n-2$ (với $n \geq 2$).
- Khi bậc tự do n tăng, đường cong Chi - square tiến dần về đường cong chuẩn.

Để vẽ biểu đồ của phân phối Chi-Square trong Python, ta có thể sử dụng cú pháp sau:

```
#x-axis ranges from 0 to 20 with .001 steps
x = np.arange(0, 20, 0.001)

#plot Chi-square distribution with 4 degrees of freedom
plt.plot(x, chi2.pdf(x, df=4))
```

x -axis xác định phạm vi cho trục x và **plt.plot()** tạo đường cong cho phân phối Chi-square với tham số bậc tự do.

Các ví dụ sau đây cho thấy cách sử dụng các hàm này.

Ví dụ 1:

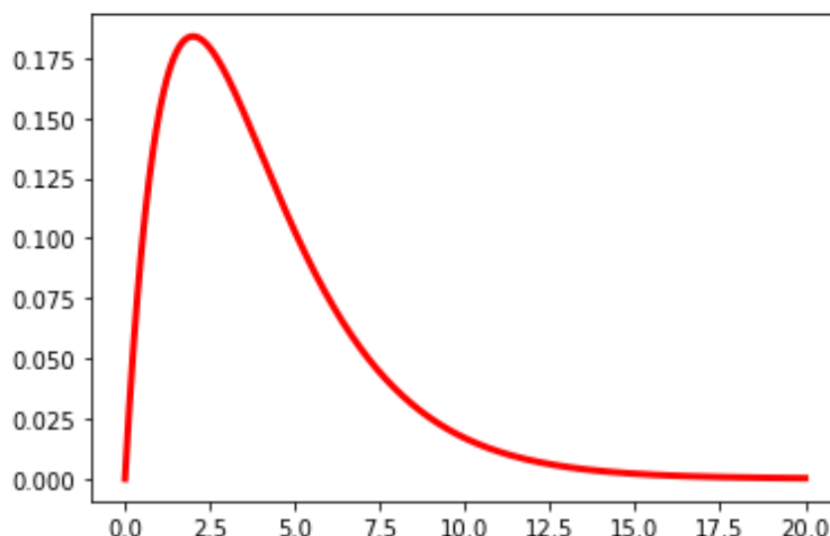
Đoạn mã sau minh họa cách vẽ một đường cong phân phối Chi-square với 4 bậc tự do

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2

#x-axis ranges from 0 to 20 with .001 steps
x = np.arange(0, 20, 0.001)

#plot Chi-square distribution with 4 degrees of freedom
```

```
plt.plot(x, chi2.pdf(x, df=4), color='red', linewidth=3)
plt.show()
```



Hình 6. 12. Đồ thị hàm phối xác suất Chi-square với 4 bậc tự do

Ví dụ 2:

Cách vẽ nhiều đường cong phân phối Chi-square với các bậc tự do khác nhau:
(xem [6], [7])

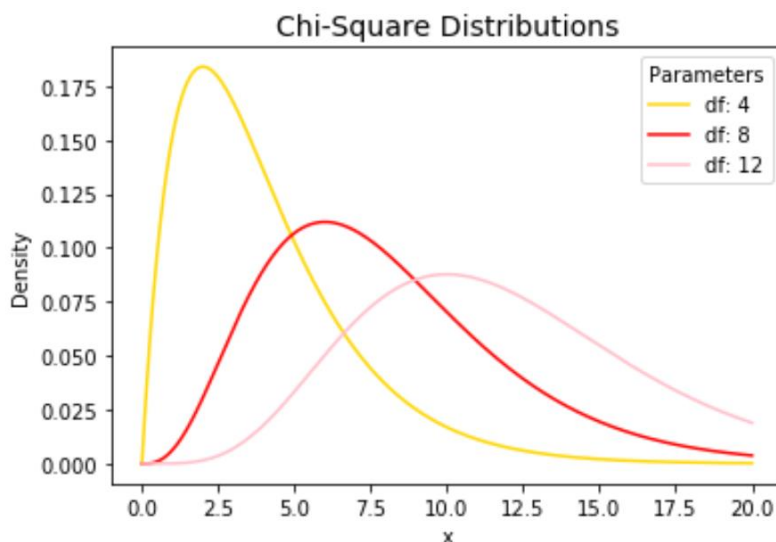
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2

#x-axis ranges from 0 to 20 with .001 steps
x = np.arange(0, 20, 0.001)

#define multiple Chi-square distributions
plt.plot(x, chi2.pdf(x, df=4), label='df: 4',
color='gold')
plt.plot(x, chi2.pdf(x, df=8), label='df: 8',
color='red')
plt.plot(x, chi2.pdf(x, df=12), label='df: 12',
color='pink')

#add legend to plot
plt.legend(title='Parameters')

#add axes labels and a title
plt.ylabel('Density')
plt.xlabel('x')
plt.title('Chi-Square Distributions', fontsize=14)
```



6.13. Phân phối F

Phân phối F thường được sử dụng trong việc xác định các giá trị quan trọng trong phân tích thống kê ANOVA.

Nếu chúng ta muốn điều tra xem hai nhóm mẫu có cùng phương sai hay không, chúng ta phải tính tỷ lệ bình phương độ lệch chuẩn mẫu:

$$F = \frac{S_x^2}{S_y^2}$$

Trong đó S_x là độ lệch chuẩn mẫu của mẫu đầu tiên và S_y là độ lệch chuẩn mẫu của mẫu thứ hai. Phân phối của thống kê này là phân phối F.

Phân phối F: Nếu gọi U_1 là phân phối Chi - square với bậc tự do d_1 và U_2 là phân phối Chi - square với bậc tự do d_2 .

Khi đó phân phối F là tỉ số giữa 2 phân phối Chi - square U_1 và U_2 . $F = U_1/U_2$.

F cũng là phân phối Chi - square có trị trung bình μ và phương sai σ^2 .

$$\mu = \frac{d_2}{d_2 - 2}$$

$$\sigma^2 = \frac{2d_2^2(d_1 + d_2) - 2}{d_1(d_2 - 2)^2(d_2 - 4)}$$

6.14. Phân bố xác suất Weibull

Phân bố xác suất Weibull (hay còn gọi là phân bố xác suất Rosin-Rammler) là một dạng thường dùng để mô tả thống kê sự xuất hiện của các đại lượng cực trị trong khí tượng, thủy văn và dự báo thời tiết như dòng chảy lũ, sóng, gió lớn nhất. Ngoài ra phân bố này cũng hay được dùng trong phân tích xác suất sống sót hoặc phá huỷ trong lý thuyết độ tin cậy, dùng trong lý thuyết cực trị; biểu diễn thời gian sản xuất và phân phối trong công nghiệp; sự phân tán tín hiệu radar và sự suy giảm tín hiệu trong liên lạc không dây.

Hàm phân phối xác suất:

$$f(x) = \frac{c}{b} \left(\frac{x-a}{b} \right)^{c-1} \exp \left[- \left(\frac{x-a}{b} \right)^c \right]$$

Với a: tham số vị trí; $b > 0$ tham số tỷ lệ và $c > 0$ tham số hình dạng.

Hàm phân phối tích lũy:

$$F(x) = P\{X \leq x\} = \int_{-\infty}^x f(x) dx = 1 - \exp \left[- \left(\frac{x-a}{b} \right)^c \right]$$

Trong trường hợp thông số hình dạng $c = 1$, phân bố Weibull trở thành phân bố hàm mũ với trị bình quân b.

Tóm tắt chương 6

Chương này đã trình bày một số biến ngẫu nhiên với phân phối xác suất quan trọng trong thống kê ứng dụng

Trong chương cần lưu ý đến các khái niệm cơ bản: hàm PMF, CDF cho biến ngẫu nhiên rời rạc và ý nghĩa của hàm PDF cho biến ngẫu nhiên liên tục

Người học cần giải thích được ý nghĩa của các cơ bản trong các ví dụ minh họa về hàm PDF, CDF, PMF của các biến ngẫu nhiên trong các ví dụ minh họa.

Câu hỏi ôn tập và bài tập chương 6

1. Hãy làm rõ ý nghĩa của các hàm PMF, CDF, PDF

2. Tìm hiểu ý nghĩa của các phân phối xác suất cơ bản của thống kê được nêu trong chương.
3. Thực hiện cài đặt các ví dụ minh họa trong chương, báo cáo kết quả thực hiện qua hình ảnh biểu đồ và kết quả thực hiện trên màn hình máy tính.
4. Giải thích các câu lệnh cơ bản để thực hiện trong các đoạn mã lệnh của các ví dụ
5. Thực hiện các ví dụ cho các dữ liệu có tính thực tiễn khác.

CHƯƠNG 7. ƯỚC LƯỢNG VÀ KIỂM ĐỊNH

Mục đích

Trình bày các vấn đề của bài toán ước lượng và kiểm định thống kê cơ bản

- *Kiểm định trung bình cho một tổng thể*
- *Kiểm định sai khác trung bình cho hai tổng thể độc lập*
- *Kiểm định sai khác trung bình cho mẫu cặp*
- + *Kiểm định giả thiết cho tỷ lệ*
- *Kiểm định tỷ lệ cho một tổng thể*
- + *Phân tích ANOVA*

(Trong chương này một số mã nguồn tham khảo từ [6], [7])

Yêu cầu

- *Nêu được ý nghĩa của giả thuyết không và giả thuyết thay thế trong bài toán kiểm định*
- *Trình bày và cho được ví dụ thực tiễn về các bước cơ bản để thực hiện bài toán kiểm định.*
- *Cài đặt thực hiện các ví dụ minh họa trong chương.*
- *Giải thích các câu lệnh cơ bản trong các ví dụ*
- *Vận dụng các ví dụ minh họa cho dữ liệu thực tiễn*

7.1. Giới thiệu bài toán kiểm định với công cụ máy tính

Khi phân tích thống kê dữ liệu thường chúng ta tiến hành các bài kiểm tra giả thuyết: hình thành giả thuyết; thu thập dữ liệu và sau đó chấp nhận hoặc bác bỏ giả thuyết. Các bài kiểm tra giả thuyết, tạo thành khung cơ bản cho hầu hết các phân tích trong y học và khoa học đời sống.

Ngày nay, việc phân tích dữ liệu thống kê là một quá trình tương tác cao: chúng ta xem xét dữ liệu và tạo ra các mô hình có thể giải thích dữ liệu. Sau đó, cần xác định các thông số phù hợp nhất cho các mô hình này và kiểm tra các mô hình này. Nếu không hài lòng với kết quả, ta cần sửa đổi mô hình để cải thiện sự tương ứng giữa các

mô hình và dữ liệu cho đến khi có kết quả mong đợi, tiếp đó ta cần tính toán khoảng tin cậy cho các thông số mô hình và đưa ra các cách diễn giải về kết quả phân tích dữ liệu dựa trên các giá trị này.

Kiểm định giả thuyết thống kê (statistical hypothesis test) là phương pháp ra quyết định sử dụng dữ liệu, hoặc từ thí nghiệm hoặc từ nghiên cứu quan sát (observational study) (không có kiểm soát). Trong thống kê (statistics), một kết quả được gọi là đủ độ tin cậy mang tính thống kê (statistically significant) nếu nó ít có khả năng diễn ra theo một ngưỡng xác suất cho trước (ví dụ 5% hay 10%). Cụm từ kiểm định độ tin cậy ("test of significance") được đưa ra bởi Ronald Fisher.

Kiểm định giả thuyết đôi khi được gọi là phân tích dữ liệu để khẳng định, để so sánh với phân tích dữ liệu để khám phá. Giả thuyết thống kê là một mệnh đề nhận định về tham số của tổng thể. Khi ta đồng nhất tổng thể với một biến ngẫu nhiên thì giả thuyết thống kê cũng có thể là nhận định về phân phối xác suất của biến ngẫu nhiên.

Ký hiệu H_0 là giả thuyết của tham số tổng thể, đi kèm với giả thuyết H_0 là mệnh đề đối lập được gọi là đối thuyết, ký hiệu là H_1 . Bài toán kiểm định giả thuyết thống kê gồm một cặp giả thuyết H_0 và đối thuyết H_1 . Dựa vào thông tin mẫu lấy được từ tổng thể ta phải đưa ra quyết định bác bỏ hay chấp nhận giả thuyết H_0 , việc chấp nhận giả thuyết H_0 tương đương với bác bỏ đối thuyết H_1 và ngược lại.

Các bước làm bài toán kiểm định

Để tiến hành kiểm định giả thuyết, thông thường người ta có thể sử dụng miền tiêu chuẩn, xác suất ý nghĩa hoặc ước lượng khoảng của các tiêu chuẩn hay tham số thống kê, với các bước thực hiện tương ứng.

- Sử dụng miền tiêu chuẩn. Để giải quyết một bài toán kiểm định giả thuyết thống kê thông qua việc sử dụng miền tiêu chuẩn, người ta thường thực hiện các bước sau:

Bước 1: Xác định tham số cần kiểm định, đặt giả thuyết và đối thuyết.

Bước 2: Xác định tiêu chuẩn thống kê và tính giá trị của tiêu chuẩn thống kê đối với giá trị mẫu đã cho.

Bước 3: Xác định miền bác bỏ W .

Bước 4: So sánh giá trị của tiêu chuẩn thống kê với miền bác bỏ W và kết luận bác bỏ hay chấp nhận giả thuyết H_0 .

- Sử dụng xác suất ý nghĩa (p -value)

Nếu ta bác bỏ giả thuyết H_0 khi thấy một giá trị cụ thể a của mẫu xuất hiện, thì ta cũng phải bác bỏ giả thuyết đó cho những giá trị khác của mẫu thuộc vào một miền xác định bởi a . Chẳng hạn với giả thuyết cần kiểm định là “Chi tiết máy được gia công có kích thước đạt tiêu chuẩn”, nếu ta bác bỏ giả thuyết khi đo thấy sản phẩm có kích thước lệch so với quy định 1 milimét thì ta cũng phải bác bỏ giả thuyết cho mọi sản phẩm khác đo được kích thước lệch so với quy định nhiều hơn 1 milimét.

Có thể về thực chất thì các sản phẩm đó đều có kích thước đạt tiêu chuẩn nhưng do những tác động ngẫu nhiên trong quá trình đo đạc mà ta có kết luận sai, dẫn đến việc phạm sai lầm với một xác suất nào đó. Tập hợp chứa các giá trị của mẫu phải bác bỏ khi đã bác bỏ một giá trị cụ thể cho trước của mẫu có một xác suất phạm sai lầm được gọi là *xác suất ý nghĩa* ứng với giá trị cụ thể đó.

Định nghĩa xác suất ý nghĩa (p -value):

Ứng với một giá trị mẫu cụ thể của tiêu chuẩn thống kê dùng kiểm định giả thuyết, *xác suất ý nghĩa* (p -value) là giá trị của xác suất phạm sai lầm nếu bác bỏ giả thuyết H_0 khi ta có giá trị mẫu cụ thể đó trong khi giả thuyết là đúng đối với mẫu đang xét. Ta thấy xác suất ý nghĩa chính là xác suất phạm sai lầm loại I đã trình bày ở phía trên.

Xác suất này nhỏ tương ứng với khả năng phạm sai lầm khi bác bỏ giả thuyết là nhỏ và ta có thể bác bỏ giả thuyết mà không e ngại có sai lầm. Ngược lại thì ta phải chấp nhận giả thuyết vì khả năng phạm sai lầm sẽ lớn. Như vậy ta có thể sử dụng xác suất ý nghĩa để giải quyết bài toán kiểm định theo thủ tục sau: Tiến hành các Bước 1 và 2 như trình bày ở trên và làm tiếp.

Bước 3': Tính xác suất ý nghĩa tương ứng với giá trị cụ thể của tiêu chuẩn thống kê đã có ở Bước 2:

Bước 4': So sánh xác suất ý nghĩa trên đây với mức ý nghĩa đã định trước (thường được cho bằng 5%, 1%, 0,5% hoặc 0,1%), nếu xác suất ý nghĩa nhỏ hơn hoặc bằng mức ý nghĩa thì bác bỏ giả thuyết, còn nếu ngược lại thì phải chấp nhận giả thuyết.

7.2. Giá trị P và ý nghĩa Thống kê

Trong thống kê, xác suất ý nghĩa (**giá trị p**, **p – giá trị, p - value**) thường được sử dụng trong kiểm định giả thuyết cho kiểm định t, kiểm định chi-bình phương, phân tích hồi quy, ANOVA và nhiều phương pháp thống kê khác.

Nếu giá trị p của kiểm định giả thuyết đủ nhỏ, chúng ta có thể bác bỏ giả thuyết rỗng. Cụ thể, khi chúng ta tiến hành kiểm định giả thuyết, ngay từ đầu chúng ta phải chọn một mức ý nghĩa. Các lựa chọn phổ biến cho mức ý nghĩa là 0,01, 0,05 và 0,10.

Nếu các giá trị p *nhỏ hơn* mức ý nghĩa của chúng ta, thì chúng ta có thể bác bỏ giả thuyết không.

Ngược lại, nếu giá trị p *bằng hoặc lớn hơn* mức ý nghĩa của chúng ta, thì chúng ta không thể bác bỏ giả thuyết không.

Một **giá trị p** là xác suất quan sát một thống kê mẫu, đây là xác suất cực tiểu để qua số liệu thống kê mẫu, có thể cho rằng giả thuyết không là đúng.

Ví dụ: giả sử một nhà máy tuyên bố rằng họ sản xuất lốp xe có trọng lượng trung bình là 200 pound. Một kiểm toán viên đưa ra giả thuyết rằng trọng lượng trung bình thực của lốp xe được sản xuất tại nhà máy này khác với 200 pound, vì vậy anh ta thực hiện một bài kiểm tra giả thuyết và nhận thấy rằng giá trị p của bài kiểm tra là 0,04. Dưới đây là cách diễn giải theo nghĩa của giá trị p:

Nếu nhà máy thực sự sản xuất lốp xe có trọng lượng trung bình là 200 pound, thì 4% trong số tất cả các cuộc đánh giá sẽ thu được hiệu quả quan sát được trong mẫu hoặc lớn hơn do lỗi mẫu ngẫu nhiên. Tức việc thu thập dữ liệu mẫu mà kiểm toán viên đã làm sẽ khó xảy ra, nếu thực sự nhà máy sản xuất lốp xe có trọng lượng trung bình là 200 pound.

Tùy thuộc vào mức ý nghĩa được sử dụng trong kiểm định giả thuyết này, kiểm toán viên có thể sẽ bác bỏ giả thuyết không rằng trọng lượng trung bình thực của lốp xe được sản xuất tại nhà máy này thực sự là 200 pound. Dữ liệu mẫu mà anh ta thu được từ cuộc kiểm toán không phù hợp lắm với giả thuyết không.

Một số lưu ý khi diễn giải giá trị P

Quan niệm sai lầm lớn nhất về giá trị p là xem nó tương đương với xác suất mắc lỗi khi bác bỏ giả thuyết không, một cách thực sự (được gọi là lỗi Loại I).

Có hai lý do chính khiến giá trị p không phải là tỷ lệ lỗi:

1. Các giá trị P được tính toán dựa trên giả định rằng giả thuyết không là đúng và sự khác biệt giữa dữ liệu mẫu và giả thuyết không đơn giản là do ngẫu nhiên gây ra. Do đó, giá trị p không thể cho ta biết xác suất giá trị null là đúng hay sai vì nó đúng 100% dựa trên quan điểm của các phép tính.

2. Mặc dù giá trị p thấp, để chỉ ra rằng dữ liệu mẫu của ta không chắc để giả định null là đúng, nhưng giá trị p vẫn không thể cho bạn biết trường hợp nào sau đây có nhiều khả năng xảy ra hơn:

- Giá trị null là sai
- Giá trị null là đúng nhưng ta thu được một mẫu chưa đủ

Xét lại ví dụ trên:

• **Giải thích đúng:** Giả sử nhà máy, sản xuất lốp xe có trọng lượng trung bình là 200 pound, ta sẽ nhận được sự khác biệt quan sát được mà ta *đã* nhận được trong mẫu của mình hoặc sự khác biệt lớn hơn trong 4% các cuộc đánh giá do lỗi lấy mẫu ngẫu nhiên.

• **Cách diễn giải sai:** Nếu ta bác bỏ giả thuyết không, có 4% khả năng bạn đang mắc sai lầm.

Các ví dụ sau đây minh họa các cách đúng để giải thích giá trị p trong bối cảnh kiểm định giả thuyết.

Ví dụ 1

Một công ty điện thoại tuyên bố rằng 90% khách hàng của họ hài lòng với dịch vụ của họ. Để kiểm tra tuyên bố này, một nhà nghiên cứu độc lập đã thu thập một mẫu ngẫu nhiên đơn giản gồm 200 khách hàng và hỏi họ có hài lòng với dịch vụ của họ hay không, 85% trả lời là có. Giá trị p được liên kết với dữ liệu mẫu này là 0,018.

Giải thích đúng về giá trị p: Giả sử rằng 90% khách hàng thực sự hài lòng với dịch vụ của họ, nhà nghiên cứu sẽ thu được sự khác biệt quan sát được mà anh ta *đã* thu được trong mẫu của mình hoặc sự khác biệt lớn hơn trong 1,8% các cuộc đánh giá do lỗi lấy mẫu ngẫu nhiên .

Ví dụ 2

Một công ty phát minh ra một loại pin mới cho điện thoại. Công ty tuyên bố rằng pin mới này sẽ hoạt động lâu hơn pin cũ ít nhất 10 phút. Để kiểm tra tuyên bố này, một nhà nghiên cứu lấy một mẫu ngẫu nhiên đơn giản gồm 80 pin mới và 80 pin cũ. Pin mới chạy trung bình 120 phút với độ lệch chuẩn là 12 phút và pin cũ chạy trung bình 115 phút với độ lệch chuẩn là 15 phút. Giá trị p thu được từ thử nghiệm cho sự khác biệt về trung bình tổng thể là 0,011.

Giải thích đúng về giá trị p: Giả sử rằng pin mới hoạt động trong cùng một khoảng thời gian hoặc ít hơn pin cũ, nhà nghiên cứu sẽ thu được sự khác biệt quan sát được hoặc sự khác biệt lớn hơn trong 1,1% nghiên cứu do lỗi lấy mẫu ngẫu nhiên.

Chú ý

Việc xác định miền bác bỏ có thể tiến hành thông qua việc tra bảng các giá trị tới hạn và có thể làm bằng tay. Trong khi đó việc tính toán xác suất ý nghĩa bằng cách tra bảng lại chưa được quen dùng. Do đó trong nhiều giáo trình chỉ trình bày thủ tục a) khi nói đến quy trình kiểm định giả thuyết. Tuy nhiên khi máy tính ngày càng phổ biến hơn và các phần mềm sẵn sàng cung cấp các tính toán liên quan đến xác suất ý nghĩa thì thủ tục b) tỏ ra rất thuận tiện.

Ngoài hai thủ tục trên, nhiều bài toán kiểm định có thể được tiến hành bằng cách sử dụng các ước lượng khoảng của các tham số hoặc các tiêu chuẩn thống kê, khá tiện dụng trong cả các tính toán bằng tay và cả khi có sự trợ giúp của máy tính.

- Sử dụng khoảng tin cậy (ước lượng khoảng) của tham số hoặc tiêu chuẩn thống kê Để tiến hành kiểm định bằng khoảng tin cậy, sau Bước 1 như đã nêu ở phần trên, ta tiếp tục tiến hành các bước sau:

Bước 2: Xác định tiêu chuẩn thống kê và tìm khoảng tin cậy (ước lượng khoảng) của tiêu chuẩn đó (hoặc của tham số cần quan tâm) ứng với mẫu đã có và độ tin cậy đã định trước.

Bước 3: So sánh khoảng tin cậy trên với một giá trị đã định, nếu khoảng tin cậy không chứa giá trị đó thì bác bỏ giả thuyết, còn nếu khoảng tin cậy chứa giá trị đó thì phải chấp nhận giả thuyết.

Các ví dụ

- Kiểm tra một tỷ lệ: Kiểm tra xem 1 đồng xu có bị thiên lệch không? Giả sử có 200 lần mặt sấp (head) đã được tìm thấy hơn 300 lần tung đồng xu, xét xem đồng xu có thiên lệch không?

- Kiểm tra sự liên kết giữa hai biến.

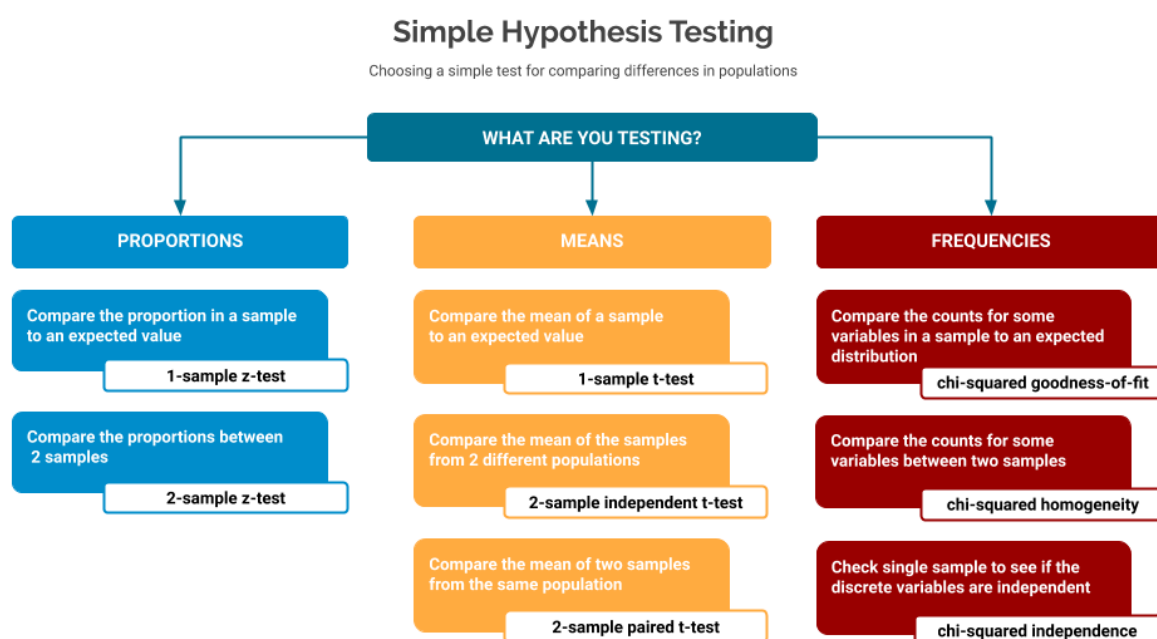
- Chiều cao và giới tính: Trong một mẫu 25 cá thể (15 nữ, 10 nam), chiều cao của nữ có khác với chiều cao của nam?

- Tuổi cao và tăng huyết áp động mạch: Trong một mẫu 25 người, chiều cao tuổi có tương quan với tăng huyết áp động mạch không?

Các bước thực hiện:

1. Lập mô hình dữ liệu
2. Xét tính phù hợp: ước tính các tham số của mô hình (tần số, trung bình, tương quan, hệ số hồi quy)
3. Tính toán thống kê thử nghiệm từ các tham số của mô hình.
4. Hình thành giả thuyết rằng: Thống kê kiểm định (phân phối của) sẽ là bao nhiêu nếu quan sát là kết quả của sự may rủi thuần túy.
5. Tính xác suất (p -value) để nhận được giá trị lớn hơn cho thống kê thử nghiệm (với giả thuyết không).

Trước khi đi vào các bài toán cơ bản của kiểm định, chúng ta xem xét hình ảnh sau để thấy được bức tranh tổng quan của vấn đề.



Hình 7. 1. Các bài toán kiểm định cơ bản

<https://sonalake.com/latest/hypothesis-testing-of-frequency-based-samples/>

7.3. Kiểm định One-Sample T-Test

Kiểm định One-Sample T-Test nhằm mục đích so sánh trung bình (mean) của tổng thể với một giá trị cụ thể nào đó.

Phương pháp kiểm định T-Test (kiểm định sự khác biệt) được sử dụng trong kiểm định sự khác biệt về giá trị trung bình của tổng thể với một giá trị cho trước, hoặc kiểm định sự khác biệt về giá trị trung bình giữa hai tổng thể.

3 loại T-Test trong thống kê và mục đích sử dụng của từng loại

- **One-Sample T-Test:** Dùng để so sánh giá trị trung bình của một tổng thể với một giá trị cụ thể nào đó. Chẳng hạn như kiểm tra xem chiều cao trung bình của đội tuyển bóng đá nam U22 Việt Nam là cao hơn, thấp hơn hay bằng với mức 1,8 mét; Điểm trung bình môn Tin học của sinh viên trong lớp là cao hơn, thấp hơn hay bằng 7 điểm...

- **Independent Samples T-Test:** là một thử nghiệm thống kê kiểm định xem có sự khác biệt có ý nghĩa thống kê giữa các phương tiện trong hai nhóm thống kê không liên quan hay không. Ví dụ, ta có 2 nhóm giá trị là nhóm độ tuổi (dưới 30 tuổi; trên 30 tuổi) và biến mức độ hài lòng. Để biết được giữa hai nhóm này, nhóm nào có mức độ hài lòng cao hơn ta sẽ dùng phương pháp kiểm định Independent Samples T-Test.

- **Paired Sample T-Test:** Phương pháp paired samples t-test được sử dụng cho mục đích so sánh sự biến đổi từng cặp giá trị trước khi và sau khi có một tác động gì đó (so sánh xem trước và sau có sự khác biệt hay không). Một ví dụ minh họa cho kiểm định này là: Một công ty áp dụng mức KPI (chỉ số đo lường và đánh giá hiệu quả hoạt động) cho một bộ phận trong công ty để thử nghiệm sự khác biệt mức độ hài lòng của nhân viên giữa chính sách mới và chính sách cũ.

7.4. T test 1 mẫu (One sample t-test)

Ví dụ 1:

Giả sử ta muốn biết chiều cao trung bình của một loài thực vật nhất định có bằng 15 inch hay không. Chúng ta thu thập một mẫu ngẫu nhiên của 12 cây và ghi lại chiều cao của mỗi cây.

Sử dụng các bước sau để tiến hành kiểm định t – test một mẫu để xác định xem chiều cao trung bình của loài cây này có thực sự bằng 15 inch hay không.

Bước 1: Tạo dữ liệu

Đầu tiên, chúng ta sẽ tạo một mảng để chứa các số đo của 12 cây:

```
data = [14, 14, 16, 13, 12, 17, 15, 14, 15, 13, 15, 14]
```

Bước 2: Tiến hành kiểm tra t một mẫu.

Tiếp theo, chúng ta sẽ sử dụng hàm `ttest_1samp()` từ thư viện `scipy.stats` để tiến hành một thử nghiệm t mẫu, sử dụng cú pháp sau:

`ttest_1samp(a, popmean)`

ở đây:

- **a:** một dãy các quan sát mẫu
- **popmean:** trung bình dân số dự kiến

Dưới đây là cách sử dụng hàm này

```
import scipy.stats as stats

#perform one sample t-test
stats.ttest_1samp(a=data, popmean=15)

(statistic=-1.6848, pvalue=0.1201)
```

Thống kê kiểm định t là **-1,6848** và xác suất ý nghĩa (*p-value/p – giá trị/ giá trị p*) hai phía tương ứng là **0,1201**.

Bước 3: Diễn giải kết quả.

Giả thuyết cho kiểm định t ở trên như sau:

H_0 : $\mu = 15$ (chiều cao trung bình của loài thực vật này là 15 inch)

H_A : $\mu \neq 15$ (chiều cao trung bình *không phải* là 15 inch)

Bởi vì xác suất ý nghĩa của thử nghiệm của chúng ta (**0,1201**) lớn hơn $\alpha = 0,05$, chúng ta không thể bác bỏ giả thuyết không H_0 của thử nghiệm. Tức không có đủ bằng chứng để nói rằng chiều cao trung bình của loài thực vật đặc biệt này là khác 15 inch.

7.5. Kiểm định t - hai mẫu (Two sample t test)

T - test hai mẫu được sử dụng để kiểm tra xem liệu giá trị trung bình của hai quần thể có bằng nhau hay không.

Ví dụ 2:

Các nhà nghiên cứu muốn biết liệu hai loài thực vật khác nhau có cùng chiều cao trung bình hay không. Để kiểm tra điều này, họ thu thập một mẫu ngẫu nhiên đơn giản gồm 20 cây từ mỗi loài.

Sử dụng các bước sau để tiến hành một phép thử t hai mẫu để xác định xem hai loài cây có cùng chiều cao hay không.

Bước 1: Tạo dữ liệu.

Đầu tiên, chúng ta sẽ tạo hai mảng để chứa các số đo của mỗi nhóm 20 cây:

```
import numpy as np

group1 = np.array([14, 15, 15, 16, 13, 8, 14, 17, 16, 14, 19, 20,
21, 15, 15, 16, 16, 13, 14, 12])
group2 = np.array([15, 17, 14, 17, 14, 8, 12, 19, 19, 14, 17, 22,
24, 16, 13, 16, 13, 18, 15, 13])
```

Bước 2: Tiến hành kiểm tra t-test hai mẫu.

Chúng ta sẽ sử dụng hàm `ttest_ind()` từ thư viện `scipy.stats` để tiến hành hai thử nghiệm t mẫu, sử dụng cú pháp sau:

`ttest_ind(a, b, equal_var = True)`

ở đây:

- **a:** một dãy (array) các quan sát mẫu cho nhóm 1
- **b:** một dãy các quan sát mẫu cho nhóm 2
- **equal_var:** Nếu nhận giá trị True, hàm sẽ thực hiện kiểm định t-test 2 mẫu độc lập với giả định các phương sai tổng thể bằng nhau. Nếu tham số này nhận giá trị Sai, hàm sẽ thực hiện kiểm định t - Welch, kiểm định này không giả định các phương sai tổng thể bằng nhau. Mặc định là giá trị True.

Trước khi thực hiện kiểm tra, chúng tôi cần quyết định xem liệu chúng ta có giả sử hai quần thể có phương sai bằng nhau hay không. Theo nguyên tắc chung, chúng ta có thể giả sử các quần thể có phương sai bằng nhau nếu tỷ lệ của phương sai của mẫu lớn hơn với phương sai mẫu nhỏ hơn, có tỷ lệ nhỏ hơn 4: 1.

```
#find variance for each group
print(np.var(group1) , np.var(group2))
```

Tỷ lệ của phương sai mẫu lớn hơn với phương sai mẫu nhỏ hơn là $12,26 / 7,73 = 1,586$, nhỏ hơn 4. Điều này có nghĩa là chúng ta có thể giả định rằng các phương sai tổng thể là bằng nhau.

Do đó, chúng ta có thể tiến hành thực hiện hai phép thử t mẫu với các phương sai bằng nhau:

```
import scipy.stats as stats

#perform two sample t-test with equal variances
stats.ttest_ind(a=group1, b=group2, equal_var=True)

(statistic=-0.6337, pvalue=0.53005)
```

Thông kê kiểm định t là **-0,6337** và xác suất ý nghĩa hai phía tương ứng là **0,53005**.

Bước 3: Diễn giải kết quả.

Hai giả thuyết cho hai phép thử t mẫu cụ thể này như sau:

H_0 : $\mu_1 = \mu_2$ (trung bình hai tổng thể bằng nhau)

H_A : $\mu_1 \neq \mu_2$ (hai trung bình tổng thể *không* bằng nhau)

Vì xác suất ý nghĩa của thử nghiệm của chúng ta (**0,53005**) lớn hơn $\alpha = 0,05$, chúng ta không thể bác bỏ giả thuyết vô hiệu của thử nghiệm. Tức không có đủ bằng chứng để nói rằng chiều cao trung bình của thực vật giữa hai quần thể là khác nhau.

7.6. Kiểm định T với phân phối Welch (T-test Welch)

Cách phổ biến nhất để so sánh giá trị trung bình (mean) giữa **hai nhóm độc lập** là sử dụng **thử nghiệm t hai mẫu**. Tuy nhiên, thử nghiệm này giả định rằng phương sai giữa hai nhóm là bằng nhau.

Nếu nghi ngờ rằng phương sai giữa hai nhóm *không* bằng nhau, thì thay vào đó, ta có thể sử dụng **phép thử t Welch's**, là **phép thử** tương đương không tham số của phép thử t hai mẫu.

Để thực hiện phép thử t test Welch trong Python, chúng ta có thể sử dụng hàm `ttest_ind()` từ thư viện **SciPy**, sử dụng cú pháp sau:

`ttest_ind(a, b, equal_var = False)`

ở đây:

- **a**: Mảng giá trị dữ liệu đầu tiên
- **b**: Mảng giá trị dữ liệu thứ hai
- **equal_var**: Chỉ định False để có giả định các phương sai là không bằng nhau giữa hai mẫu thống kê

Ví dụ 3:

Giả sử chúng ta muốn so sánh điểm thi của 12 học sinh đã sử dụng một tài liệu luyện thi để chuẩn bị cho một kỳ thi nào đó so với 12 học sinh không sử dụng.

```
#import ttest_ind() function
from scipy import stats

#define two arrays of data
booklet = [90, 85, 88, 89, 94, 91, 79, 83, 87, 88, 91, 90]
no_booklet = [67, 90, 71, 95, 88, 83, 72, 66, 75, 86, 93, 84]

#perform Welch's t-test
stats.ttest_ind(booklet, no_booklet, equal_var = False)

Ttest_indResult(statistic=2.23606797749, pvalue=0.04170979503207)
```

Kết quả kiểm định cho kết quả là **2,2361** và xác suất ý nghĩa tương ứng là **0,0417**.

Vì xác suất ý nghĩa này nhỏ hơn 0,05, chúng ta có thể bác bỏ giả thuyết không của kiểm định và kết luận rằng có sự khác biệt có ý nghĩa thống kê về điểm thi trung bình giữa hai nhóm.

Lưu ý rằng hai cỡ mẫu trong ví dụ này là bằng nhau, nhưng thử nghiệm t của Welch vẫn hoạt động ngay cả khi hai cỡ mẫu không bằng nhau.

7.7. T-Test các mẫu được ghép nối (Paired Samples T-Test)

Phép **thử t từng mẫu được ghép nối** được sử dụng để so sánh giá trị trung bình của hai mẫu khi mỗi quan sát trong một mẫu có thể được ghép nối với một quan sát trong mẫu kia.

Ví dụ 4:

Giả sử chúng ta muốn biết liệu một chương trình học nhất định có ảnh hưởng đáng kể đến kết quả học tập của học sinh trong một kỳ thi cụ thể hay không. Để kiểm tra điều này, chúng ta có 15 học sinh trong một lớp làm bài kiểm tra trước. Sau đó, chúng ta cho mỗi học sinh tham gia vào chương trình học trong hai tuần. Sau đó, học sinh làm lại một bài kiểm tra có độ khó tương tự.

Để so sánh sự khác biệt giữa điểm trung bình trong bài kiểm tra thứ nhất và thứ hai, chúng ta sử dụng bài kiểm tra t mẫu ghép nối (Paired Samples T-Test) vì đối với mỗi học sinh, điểm kiểm tra đầu tiên của họ có thể được ghép nối với điểm kiểm tra thứ hai của họ.

Bước 1: Tạo dữ liệu.

Đầu tiên, chúng ta sẽ tạo hai mảng để lưu điểm của bài kiểm tra trước và bài kiểm tra sau:

```
pre = [88, 82, 84, 93, 75, 78, 84, 87, 95, 91, 83, 89, 77, 68, 91]
post = [91, 84, 88, 90, 79, 80, 88, 90, 90, 96, 88, 89, 81, 74, 92]
```

Bước 2: Tiến hành kiểm tra các mẫu đã được ghép đôi.

Tiếp theo, chúng ta sẽ sử dụng hàm **ttest_rel()** từ thư viện `scipy.stats` để tiến hành kiểm tra t-test mẫu được ghép nối, sử dụng cú pháp sau:

ttest_rel(a, b)

ở đây:

- **a**: Dãy các quan sát mẫu từ nhóm 1

- **b:** Dãy các quan sát mẫu từ nhóm 2

Dưới đây là cách sử dụng hàm này trong ví dụ trên

```
import scipy.stats as stats

#perform the paired samples t-test
stats.ttest_rel(pre, post)

(statistic=-2.9732, pvalue=0.0101)
```

Kết quả thống kê là **-2,9732** và tương ứng xác suất ý nghĩa hai phía là **0,0101**.

Bước 3: Diễn giải kết quả

Trong ví dụ này, t-test các mẫu được ghép nối, đã sử dụng giả thuyết rỗng và giả thuyết thay thế như sau:

H₀ : Điểm trung bình trước khi kiểm tra và sau khi kiểm tra bằng nhau

H_A : Điểm trung bình trước khi kiểm tra và sau khi kiểm tra *không* bằng nhau

Vì xác suất ý nghĩa (**0,0101**) nhỏ hơn 0,05, chúng ta bác bỏ giả thuyết không. Tức có đầy đủ bằng chứng để nói rằng điểm trung bình thực sự của sinh viên trước và sau khi tham gia chương trình học là khác nhau.

7.8. Kiểm tra T-test, giá trị trung bình của hai mẫu độc lập

Đây là phép kiểm tra hai phía đối với giả thuyết không, cho rằng hai mẫu độc lập có giá trị trung bình (kỳ vọng) là bằng nhau.

```
scipy.stats.ttest_ind_from_stats(mean1, std1, nobs1, mean2, std2, nobs2, equal_var=True, alternative='two-sided')
```

Tham số

mean1: mảng lưu các giá trị trung bình của mẫu 1;

std1: mảng chứa độ lệch chuẩn của mẫu 1;

nobs1: mảng các phần tử của mẫu 1.

Mean2: mảng lưu các giá trị trung bình của mẫu 2;

Std2: mảng chứa độ lệch chuẩn của mẫu 2;

Nobs2: mảng các phần tử của mẫu 2.

Equal: tham số tùy chọn để chỉ ra giả định 2 mẫu có phương sai mẫu bằng nhau hay không.

alternative{'two-sided', 'less', 'greater'}: tham số tùy chọn chỉ ra việc kiểm định 1 hay 2 phía.

Hàm trả về:

Statistic: mảng kiểu thực tính giá trị kiểm định t-test và p – giá trị cho kiểm tra 2 phía (two-tailed p-value).

Tương tự ta có hàm:

```
scipy.stats.ttest_ind(a, b, axis=0, equal_var=True, nan_policy='propagate', permutations=None, random_state=None, alternative='two-sided', trim=0)
```

Nhằm kiểm định T-test giá trị kỳ vọng của 2 mẫu độc lập.

Đây là phép kiểm tra hai phía đối với giả thuyết không: 2 mẫu độc lập có giá trị trung bình (kỳ vọng) bằng nhau. Thử nghiệm này giả định rằng các quần thể có các phương sai giống nhau theo mặc định.

Tham số:

a, b: các mảng có độ dài như nhau lưu các phần tử của mẫu

axis: tham số tùy chọn để chỉ ra chiều của mảng .a, b cần tính toán

equal_var: tham số tùy chọn để chỉ ra 2 mẫu kiểm định có cùng phương sai hay không

nan_policy{'propagate', 'raise', 'omit'}: tham số tùy chọn để cho pgesp bỏ qua hay không các giá trị trống.

Permutations: tham số nhận giá trị kiểu nguyên không âm để chỉ ra có hay không việc sử dụng phân phối t để tính p – giá trị.

random_state: tham số tùy chọn, chỉ ra việc sử dụng hàm sinh số ngẫu nhiên của numpy hay không khi sử dụng để tạo mẫu.

alternative{'two-sided', 'less', 'greater'}: tham số tùy chọn để chỉ ra sử dụng kiểm định 1 hay 2 phía.

Hàm trả về:

Statistic: mảng kiểu thực lưu giá trị thống kê t – test và p – giá trị 2 phía.

https://maths.uel.edu.vn/Resources/Docs/SubDomain/maths/TaiLieuHocTap/ToanUngDung/phn_tch_anova.html

7.9. Giá trị tới hạn T

Giá trị tới hạn (Critical value) là gì?

Critical value được hiểu là giá trị tới hạn. Đặc biệt, điểm đầu và điểm cuối của các khoảng tin cậy cũng có thể gọi là giá trị tới hạn.

Giá trị tới hạn (*critical value*) $\alpha\%$ của một phân phối xác suất là giá trị mà chỉ có $\alpha\%$ xác suất nằm trên (hoặc nằm dưới) nó. Chẳng hạn chỉ có xác suất 5% là một biến chuẩn chuẩn hóa sẽ nhận các giá trị nằm trên giá trị gần bằng 5% của phân phối chuẩn chuẩn hóa (trong trường hợp này là 1.65).

Giá trị tới hạn thường được sử dụng để xác định tiêu chuẩn chấp nhận trong kiểm định giả thuyết thống kê và tính toán khoảng tin cậy

Khi tiến hành kiểm định t, ta sẽ nhận được một kết quả thống kê kiểm tra. Để xác định xem kết quả của t-test là có ý nghĩa thống kê, ta cần so sánh các số liệu thống kê kiểm tra với **giá trị tới hạn T**. Nếu giá trị tuyệt đối của thống kê thử nghiệm lớn hơn giá trị tới hạn T thì kết quả của thử nghiệm có ý nghĩa thống kê.

Giá trị tới hạn T có thể được tìm thấy bằng cách sử dụng [bảng phân phối t](#) hoặc bằng cách sử dụng phần mềm thống kê.

Để tìm giá trị tới hạn T, ta cần chỉ định:

- Mức ý nghĩa (các lựa chọn phổ biến là 0,01, 0,05 và 0,10)
- Bậc tự do

Sử dụng hai giá trị này, chúng ta có thể xác định giá trị quan trọng T được so sánh với thống kê thử nghiệm.

Cách tìm giá trị tới hạn T

Để tìm giá trị quan trọng T trong Python, có thể sử dụng hàm [scipy.stats.t.ppf](#), sử dụng cú pháp sau:

scipy.stats.t.ppf (q, df)

ở đây:

- **q**: Mức ý nghĩa sử dụng
- **df**: Bậc tự do

Các ví dụ sau đây minh họa cách tìm giá trị tới hạn T cho bài kiểm định bên trái, bài kiểm định bên phải và bài kiểm định hai bên.

Kiểm tra bên trái

Giả sử chúng ta muốn tìm giá trị tới hạn T cho phép thử bên trái với mức ý nghĩa là 0,05 và bậc tự do = 22:

```
import scipy.stats

#find T critical value
scipy.stats.t.ppf(q=1-.05,df=22)

1.7171
```

Giá trị tới hạn T là **-1,7171**. Do đó, nếu thống kê thử nghiệm nhỏ hơn giá trị này, kết quả của thử nghiệm có ý nghĩa thống kê.

Kiểm tra bên phải

Giả sử chúng ta muốn tìm giá trị tới hạn T cho phép thử bên phải với mức ý nghĩa là 0,05 và bậc tự do = 22:

```
import scipy.stats

#find T critical value
scipy.stats.t.ppf(q=1-.05,df=22)

1.7171
```

Giá trị tới hạn T là **1.7171**. Như vậy, nếu thống kê kiểm định lớn hơn giá trị này thì kết quả kiểm định có ý nghĩa thống kê.

Kiểm định hai phía

Giả sử chúng ta muốn tìm giá trị tới hạn T cho phép thử hai phía với mức ý nghĩa là 0,05 và bậc tự do = 22:

```
import scipy.stats

#find T critical value
scipy.stats.t.ppf(q=1-.05/2,df=22)

2.0739
```

Khi thực hiện kiểm tra hai phía, sẽ có hai giá trị quan trọng. Trong trường hợp này, các giá trị tới hạn T là **2,0739** và **-2,0739**. Như vậy, nếu thống kê kiểm định nhỏ hơn -2.0739 hoặc lớn hơn 2.0739 thì kết quả kiểm định có ý nghĩa thống kê.

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.t.html>

7.10. Z- test 1 mẫu và 2 mẫu (one sample z – test và two sample z – test)

Z-Test là một kiểm định thống kê được dùng để xác định xem liệu giá trị kỳ vọng của hai tổng thể có khác nhau khi biết phương sai và kích thước mẫu đủ lớn. Để kiểm định này được thực hiện một cách chính xác, giả định đưa ra là có phân phối chuẩn và những sai số như độ lệch chuẩn là đo đạc được.

Kiểm định giá trị trung bình của 1 tổng thể (one-sample t test), kiểm định giá trị trung bình của 2 tổng thể (two-sample t test), kiểm định trung bình của hai mẫu phụ thuộc hay mẫu phân phối từng cặp (paired sample t test), ước lượng hợp lý cực đại (maximum-likelihood estimate) là những dẫn chứng về các kiểm định có thể được tiến hành như kiểm định Z. Kiểm định Z có liên quan chặt chẽ đến kiểm định T, nhưng kiểm định T hiệu quả nhất khi mẫu kiểm nghiệm nhỏ. Ngoài ra, độ lệch chuẩn trong kiểm định T chưa được xác định, trong khi kiểm định Z giả định rằng độ lệch chuẩn đã biết. Nếu độ lệch chuẩn của tổng thể chưa được xác định thì phải giả định rằng phương sai của mẫu chính là phương sai của tổng thể.

Một số so sánh kiểm tra t và kiểm tra z:

1. Kiểm tra t là thử nghiệm thống kê được sử dụng để so sánh và phân tích xem giá trị kỳ vọng của hai tổng thể có khác nhau hay không khi độ lệch chuẩn không được biết. Ngược lại, z-test là một thử nghiệm tham số, được áp dụng khi độ lệch chuẩn được biết, để so sánh và phân tích giá trị kỳ vọng của hai bộ dữ liệu khác nhau.
2. Bài kiểm tra t dựa trên phân phối T - Student. Ngược lại, z-test dựa vào giả định rằng phân phối của mẫu là có phân phối chuẩn.
3. Một trong những điều kiện quan trọng để áp dụng thử nghiệm t là phương sai tổng thể chưa biết. Ngược lại, phương sai tổng thể được cho trước trong trường hợp kiểm tra z.
4. Thử nghiệm z được sử dụng khi kích thước mẫu lớn, tức là $n > 30$ và thử nghiệm t là phù hợp khi kích thước của mẫu nhỏ, theo nghĩa là $n < 30$.

Chúng ta có thể sử dụng hàm **ztest ()** từ gói [statsmodels](#) để thực hiện bài kiểm tra z một mẫu và hai mẫu trong Python.

Hàm này sử dụng cú pháp cơ bản sau:

```
statsmodels.stats.weightstats.ztest(x1, x2=None, value=0)
```

ở đây:

- **x1** : giá trị của mẫu đầu tiên
- **x2** : giá trị của mẫu thứ hai (nếu thực hiện kiểm tra z hai mẫu)

- **giá trị** : **giá trị** trung bình với giả thuyết rỗng (trong một trường hợp mẫu) hoặc chênh lệch trung bình (trong hai trường hợp mẫu)

Các ví dụ sau đây cho thấy cách sử dụng hàm này trong thực tế.

Ví dụ 1

Giả sử chỉ số IQ trong một quần thể có phân phối chuẩn với giá trị trung bình là $\mu = 100$ và độ lệch chuẩn là $\sigma = 15$.

Một nhà nghiên cứu muốn biết liệu một loại thuốc mới có ảnh hưởng đến chỉ số IQ hay không, vì vậy ông đã tuyển 20 bệnh nhân để dùng thử và ghi lại mức chỉ số IQ của họ.

Đoạn mã sau đây cho thấy cách thực hiện kiểm tra z-test một mẫu trong Python để xác định xem loại thuốc mới có gây ra sự khác biệt đáng kể về mức IQ hay không:

```
from statsmodels.stats.weightstats import ztest as ztest

#enter IQ levels for 20 patients
data = [88, 92, 94, 94, 96, 97, 97, 97, 99, 99,
        105, 109, 109, 109, 110, 112, 112, 113, 114, 115]

#perform one sample z-test
ztest(data, value=100)

(1.5976240527147705, 0.1101266701438426)
```

Thống kê thử nghiệm cho thử nghiệm z một mẫu là **1,5976** và giá trị p tương ứng là **0,1101**.

Vì giá trị p này không nhỏ hơn 0,05, chúng ta không có đủ bằng chứng để bác bỏ giả thuyết rỗng. Nói cách khác, loại thuốc mới không ảnh hưởng đáng kể đến chỉ số IQ.

Ví dụ 2

Giả sử mức IQ giữa các cá nhân ở hai thành phố khác nhau, với quần thể có phân phối chuẩn với độ lệch chuẩn đã biết.

Một nhà nghiên cứu muốn biết liệu mức IQ trung bình giữa các cá nhân ở thành phố A và thành phố B có khác nhau hay không, vì vậy cô ấy chọn một mẫu ngẫu nhiên đơn giản gồm 20 cá nhân từ mỗi thành phố và ghi lại mức IQ của họ.

Đoạn mã sau cho biết cách thực hiện hai bài kiểm tra z mẫu bằng Python để xác định xem mức IQ trung bình có khác nhau giữa hai thành phố hay không:

```
from statsmodels.stats.weightstats import ztest as ztest
```

```
#enter IQ levels for 20 individuals from each city
cityA = [82, 84, 85, 89, 91, 91, 92, 94, 99, 99,
         105, 109, 109, 109, 110, 112, 112, 113, 114, 114]

cityB = [90, 91, 91, 91, 95, 95, 99, 99, 108, 109,
         109, 114, 115, 116, 117, 117, 128, 129, 130, 133]

#perform two sample z-test
ztest(cityA, cityB, value=0)

(-1.9953236073282115, 0.046007596761332065)
```

Thống kê thử nghiệm cho hai thử nghiệm z mẫu là **-1,9953** và giá trị p tương ứng là **0,0460**.

Vì giá trị p này nhỏ hơn 0,05, chúng ta có đủ bằng chứng để bác bỏ giả thuyết rỗng. Nói cách khác, mức độ IQ trung bình có sự khác biệt đáng kể giữa hai thành phố.

7.11. Phép kiểm tra z-test một tỷ lệ và hai tỷ lệ

Phép kiểm tra z-test một tỷ lệ được sử dụng để so sánh tỷ lệ quan sát được với tỷ lệ lý thuyết.

Thử nghiệm này sử dụng các giả thuyết không sau:

- **H_0** : $p = p_0$ (tỷ lệ dân số bằng tỷ lệ giả thuyết p_0)

Giả thuyết thay thế có thể là hai bên, bên trái hoặc bên phải:

- **H_1 (hai mặt)**: $p \neq p_0$ (tỷ lệ dân số không bằng một giá trị giả thuyết p_0)
- **H_1 (bên trái)**: $p < p_0$ (tỷ lệ dân số nhỏ hơn một số giá trị giả thuyết p_0)
- **H_1 (bên phải)**: $p > p_0$ (tỷ lệ dân số lớn hơn một số giá trị giả thuyết p_0)

Thống kê thử nghiệm được tính như sau:

$$z = (p - p_0) / \sqrt{p_0 (1 - p_0) / n}$$

ở đây:

- **p**: tỷ lệ mẫu quan sát
- **p_0** : tỷ lệ dân số giả định
- **n**: kích thước mẫu

Nếu giá trị p tương ứng với thống kê thử nghiệm z nhỏ hơn mức ý nghĩa đã chọn (các lựa chọn phổ biến là 0,10, 0,05 và 0,01) thì có thể bác bỏ giả thuyết rỗng.

Đối với kiểm định Z test, thì mẫu phải phản ánh được sự phân bố của tổng thể. Đối với những thử nghiệm này, một nguyên tắc chung là: Nếu cỡ mẫu là **n** và tỷ lệ mẫu của **p**, khi đó cả **np** và **n (1-p)** ít nhất phải bằng **10**.

Ví dụ: nếu một mẫu cho thấy 80% vấn đề đã được giải quyết trong 5 ngày và 20% không được giải quyết, thì mẫu đó phải có ít nhất 10 vấn đề được giải quyết trong vòng 5 ngày và ít nhất 10 vấn đề được giải quyết trong hơn 5 ngày.

Để thực hiện kiểm tra z một tỷ lệ trong Python, chúng ta có thể sử dụng hàm `proportions_ztest()` từ thư viện **statsmodels**, sử dụng cú pháp sau:

https://www.statsmodels.org/stable/generated/statsmodels.stats.proportion.proportions_ztest.html

proportions_ztest(count, nobs, value=None, alternative='two-sided')

ở đây:

- **count:** Số lần thành công
- **nobs:** Số lần thử nghiệm
- **value:** Tỷ lệ dân số giả định
- **alternative:** Giả thuyết thay thế

Hàm này trả về thống kê kiểm tra z và giá trị p tương ứng.

Ví dụ

Giả sử chúng ta muốn biết liệu tỷ lệ cư dân ở một quận nào đó ủng hộ một luật nào đó có bằng 60% hay không. Để kiểm tra điều này, chúng ta thu thập dữ liệu sau trên một mẫu ngẫu nhiên:

- **p₀** : tỷ lệ dân số giả định = 0,60
- **x**: cư dân ủng hộ luật: 64
- **n**: cỡ mẫu = 100

```
#import proportions_ztest function
from statsmodels.stats.proportion import proportions_ztest

#perform one proportion z-test
Proportions_ztest(count=60, nobs=100, value=0.64)
(-0.8164965809277268, 0.41421617824252466)
```

Từ kết quả đầu ra, chúng ta có thống kê kiểm định z là **-0,8165** và xác suất ý nghĩa p - giá trị tương ứng là **0,4142**. Vì giá trị này không nhỏ hơn $\alpha = 0,05$ nên chúng

ta không thể bác bỏ giả thuyết rỗng. Tức không có đủ bằng chứng để nói rằng tỷ lệ cư dân ủng hộ luật khác với 0,60.

<https://www.statology.org/one-proportion-z-test-python/>

Z – test 2 tỷ lệ

Giả sử có hai mẫu, được xác định theo tỷ lệ và chúng ta cần đưa ra khẳng định, liệu tỷ lệ tổng thể của một trong hai tổng thể đã được lấy mẫu có lớn hơn hay nhỏ hơn hay khác với tỷ lệ của tổng thể kia hay không. Như vậy ở đây ta có:

- Giả thuyết không là *tỷ lệ từ hai tổng thể là như nhau*
- Giả thuyết thay thế là *tỷ lệ từ hai tổng thể là khác nhau*

Ví dụ:

Từ một tổng thể, chúng ta lấy mẫu 500 bài kiểm tra và thấy 410 bài là có điểm đậu. Từ một tổng thể khác, chúng ta lấy mẫu 400 bài kiểm tra và thấy rằng 379 bài có điểm đậu.

Chúng ta sẽ sử dụng kiểm tra z-test 2 mẫu để kiểm tra xem mẫu đó có cho phép chúng ta chấp nhận hay bác bỏ giả thuyết không: Tỷ lệ bài đạt điểm đậu của hai tổng thể là như nhau?

Ta có đoạn mã sau trong Python, để tiến hành kiểm tra z –test 2 mẫu cho ví dụ trên như sau: (xem [6], [7])

```
from statsmodels.stats.proportion import proportions_ztest
import numpy as np
# can we assume anything from our sample
significance = 0.025
# our samples - 82% are good in one, and ~79% are good in the
other
# note - the samples do not need to be the same size
sample_success_a, sample_size_a = (410, 500)
sample_success_b, sample_size_b = (379, 400)
# check our sample against
successes = np.array([sample_success_a, sample_success_b])
samples = np.array([sample_size_a, sample_size_b])
# note, no need for a Ho value here - it's derived from the
other parameters
stat, p_value = proportions_ztest(count=successes,
nobs=samples, alternative='two-sided')
# report
print('z_stat: %0.3f, p_value: %0.3f' % (stat, p_value))
if p_value > significance:
print ("Fail to reject the null hypothesis - we have nothing
else to say")
else:
```



```
print ("Reject the null hypothesis - suggest the alternative hypothesis is true")
```

7.12. Giá trị tới hạn Z

Để xác định xem kết quả của cuộc thử nghiệm giả thuyết có ý nghĩa về mặt thống kê, ta có thể so sánh các số liệu thống kê kiểm tra với **giá trị tới hạn Z**. Nếu giá trị tuyệt đối của thống kê thử nghiệm lớn hơn giá trị tới hạn Z, thì kết quả của thử nghiệm có ý nghĩa thống kê.

Để tìm giá trị tới hạn Z trong Python, có thể sử dụng hàm [scipy.stats.norm.ppf\(\)](#), sử dụng cú pháp sau:

scipy.stats.norm.ppf (q)

ở đây:

- **q:** Mức ý nghĩa sử dụng

Các ví dụ sau minh họa cách tìm giá trị tới hạn Z cho kiểm tra bên trái, kiểm tra bên phải và kiểm tra hai bên.

Kiểm tra bên trái

Giả sử chúng ta muốn tìm giá trị tới hạn Z cho một phép thử bên trái với mức ý nghĩa là 0,05:

```
import scipy.stats

#find Z critical value
scipy.stats.norm.ppf (.05)

-1.64485
```

Giá trị tới hạn của Z là **-1,64485**. Do đó, nếu thống kê thử nghiệm nhỏ hơn giá trị này, kết quả của thử nghiệm có ý nghĩa thống kê.

Kiểm tra bên phải

Giả sử chúng ta muốn tìm giá trị tới hạn Z cho phép thử bên phải với mức ý nghĩa là 0,05:

```
import scipy.stats

#find Z critical value
scipy.stats.norm.ppf (1- .05)

1.64485
```

Giá trị tới hạn của Z là **1,64485** . Như vậy, nếu thống kê kiểm định lớn hơn giá trị này thì kết quả kiểm định có ý nghĩa thống kê.

Kiểm tra hai mặt

Giả sử chúng ta muốn tìm giá trị tới hạn Z cho phép thử hai phía với mức ý nghĩa là 0,05:

```
import scipy.stats

#find Z critical value
scipy.stats.norm.ppf(1- .05/2)

1.95996
```

Khi thực hiện kiểm tra hai phía, ta sẽ có hai giá trị tới hạn. Trong trường hợp này, các giá trị tới hạn Z là **1,95996** và **-1,95996** . Do đó, nếu thống kê thử nghiệm nhỏ hơn -1,95996 hoặc lớn hơn 1,95996, kết quả của thử nghiệm có ý nghĩa thống kê.

7.13. Mức độ ý nghĩa

“Ý nghĩa thống kê không chỉ thú vị về kết quả. Chúng ta nên mô tả kết quả dưới dạng các thước đo về mức độ để không chỉ nói lên liệu một phương pháp điều trị có ảnh hưởng đến con người hay không, mà còn cần chỉ ra được nó ảnh hưởng đến họ như thế nào”. Gene V. Glass

Trong thống kê, chúng ta thường sử dụng giá trị p để xác định xem có sự khác biệt có ý nghĩa thống kê giữa hai nhóm hay không.

Ví dụ, giả sử chúng ta muốn biết liệu hai kỹ thuật học tập khác nhau có dẫn đến điểm thi khác nhau hay không. Vì vậy, chúng ta xét một nhóm 20 sinh viên sử dụng một kỹ thuật học tập để chuẩn bị cho một bài kiểm tra trong khi một nhóm 20 sinh viên khác sử dụng một kỹ thuật học tập khác. Sau đó, chúng ta cho mỗi học sinh làm bài kiểm tra giống nhau.

Sau khi chạy thử nghiệm t-test hai mẫu để tìm sự khác biệt về giá trị trung bình (mean), chúng ta thấy rằng giá trị p của thử nghiệm là 0,001. Nếu chúng ta sử dụng mức ý nghĩa 0,05 thì điều này có nghĩa là có sự khác biệt có ý nghĩa thống kê giữa điểm kiểm tra trung bình của hai nhóm. Vì vậy, kỹ thuật nghiên cứu có ảnh hưởng đến điểm số của bài kiểm tra.

Tuy nhiên, trong khi giá trị p cho chúng ta biết rằng kỹ thuật nghiên cứu có tác động đến điểm số của bài kiểm tra, nó không cho chúng ta biết *quy mô* của tác động. Để hiểu điều này, chúng ta cần biết **mức độ ý nghĩa** .

Mức độ ý nghĩa là gì?

Mức độ ý nghĩa là cách để định lượng sự khác biệt giữa hai nhóm. Trong khi giá trị p cho chúng ta biết liệu có sự khác biệt có ý nghĩa thống kê giữa hai nhóm hay không, thì mức độ ý nghĩa có thể cho chúng ta *biết* sự khác biệt này thực sự *lớn như thế nào*. Trong thực tế, mức độ ý nghĩa thú vị và hữu ích hơn nhiều so với giá trị p .

Với mức độ ý nghĩa, chúng ta không phán xét là “tốt” hoặc “xấu”, vì nó chỉ đơn giản đo lường quy mô của sự khác biệt giữa hai nhóm hoặc sức mạnh của sự liên kết giữa hai nhóm.

Có ba cách để đo lường mức độ ý nghĩa, tùy thuộc vào loại phân tích bạn đang thực hiện:

a. Sự khác biệt về giá trị trung bình

Khi bạn quan tâm đến việc nghiên cứu sự khác biệt trung bình giữa hai nhóm, cách thích hợp để tính toán mức độ ý nghĩa là thông qua **sự khác biệt trung bình được chuẩn hóa**. Công thức phổ biến nhất được sử dụng được gọi là Cohen's d , được tính như sau:

$$\text{Cohen's } d = (x_1 - x_2) / s$$

trong đó x_1 và x_2 lần lượt là trung bình mẫu của nhóm 1 và nhóm 2, và s là độ lệch chuẩn của tổng thể mà từ đó hai nhóm được lấy.

Với công thức này, mức độ ý nghĩa được hiểu:

- Nếu $d = 1$, tức hai nhóm có giá trị trung bình là khác nhau một độ lệch chuẩn.
- $d = 2$ có nghĩa là ý nghĩa của nhóm khác nhau hai lần độ lệch chuẩn.
- $d = 2.5$, tức hai giá trị trung bình (mean) khác nhau 2,5 độ lệch chuẩn.

Ví dụ 2:

Mức độ ý nghĩa là 0,3 có nghĩa là điểm của người bình thường trong nhóm 2 *cao hơn* 0,3 độ lệch chuẩn so với người bình thường trong nhóm 1, và do đó vượt quá 62% của những người ở nhóm 1.

Bảng sau đây cho thấy các mức độ ý nghĩa khác nhau và tỷ lệ phần trăm tương ứng của chúng:

Độ hiệu quả	Tỷ lệ phần trăm của Nhóm 2 sẽ thấp hơn người bình thường trong Nhóm 1
0,0	50%
0,2	58%
0,4	66%
0,6	73%
0,8	79%
1,0	84%
1,2	88%
1,4	92%
1,6	95%
1,8	96%
2,0	98%
2,5	99%
3,0	99,9%

Mức độ ý nghĩa càng lớn thì sự khác biệt giữa các cá nhân trung bình trong mỗi nhóm càng lớn.

Nói chung, d từ 0,2 trở xuống được coi là mức độ ý nghĩa nhỏ, d khoảng 0,5 được coi là mức độ ý nghĩa trung bình và d từ 0,8 trở lên được coi là mức độ ý nghĩa lớn.

Do đó, nếu giá trị trung bình của hai nhóm không chênh lệch ít nhất 0,2 độ lệch chuẩn, thì sự khác biệt là nhỏ, ngay cả khi giá trị p là có ý nghĩa thống kê.

b. Hệ số tương quan

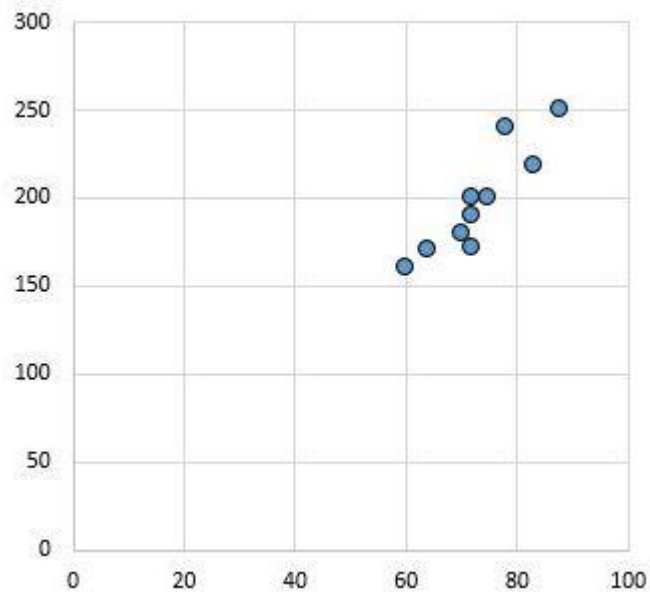
Khi quan tâm mối quan hệ định lượng giữa hai biến, cách phổ biến nhất để đánh giá quy mô ảnh hưởng là thông qua [Hệ số tương quan Pearson](#). Đây là thước đo sự kết hợp tuyến tính giữa hai biến X và Y . Nó có giá trị từ -1 đến 1 trong đó:

- -1 chỉ ra mối tương quan tuyến tính hoàn toàn ngược giữa hai biến
- 0 cho thấy không có mối tương quan tuyến tính giữa hai biến
- 1 chỉ ra mối tương quan tuyến tính hoàn toàn thuận giữa hai biến

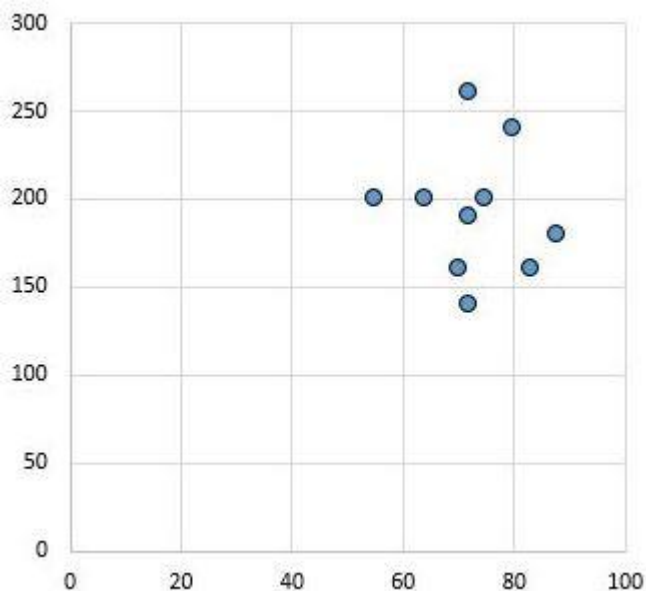
Hệ số tương quan càng khác 0 thì mối quan hệ tuyến tính giữa hai biến càng mạnh. Đây cũng có thể được nhìn thấy bằng cách tạo ra một phân tán đơn giản của các giá trị cho các biến X và Y .

Ví dụ, xét biểu đồ phân tán sau đây, của hai biến X , Y có hệ số tương quan là $r = 0,94$.

Giá trị r khác 0 nhiều, điều này cho thấy rằng có một mối quan hệ thuận chặt chẽ giữa hai biến.



Ngược lại, biểu đồ phân tán sau đây cho thấy giá trị của hai biến có hệ số tương quan là $r = 0,03$. Giá trị này gần bằng 0, điều này cho thấy rằng hầu như không có mối quan hệ nào giữa hai biến.



Nói chung, mức độ ý nghĩa được coi là thấp nếu giá trị của hệ số tương quan Pearson r là khoảng 0,1, trung bình nếu r khoảng 0,3 và lớn nếu r lớn hơn hoặc bằng 0,5.

c. Tỷ lệ chênh lệch

Khi bạn quan tâm đến việc nghiên cứu tỷ lệ thành công trong một nhóm điều trị so với tỷ lệ thành công trong nhóm đối chứng, cách phổ biến nhất để tính toán quy mô ảnh hưởng là thông qua **tỷ lệ chênh lệch**.

Ví dụ, giả sử chúng ta có bảng sau:

Độ hiệu quả	# Thành công	# Thất bại
Nhóm điều trị	A	B
Nhóm kiểm soát	C	D

Tỷ lệ chênh lệch sẽ được tính như sau:

$$\text{Tỷ lệ chênh lệch} = (A.D) / (B.C)$$

Tỷ lệ chênh lệch càng xa 1, khả năng điều trị có hiệu quả thực tế càng cao.

Ưu điểm của việc sử dụng mức độ ý nghĩa so với giá trị p

Mức độ ý nghĩa có một số lợi thế so với giá trị p:

1. Mức độ ý nghĩa giúp chúng ta hiểu rõ hơn về sự khác biệt *lớn như thế nào* giữa hai nhóm hoặc *mức độ* liên kết chặt chẽ giữa hai nhóm. Một p-giá trị chỉ có thể cho chúng ta biết có hay không có một số khác biệt đáng kể, hay một số khác biệt kết hợp.

2. Không giống như giá trị p, mức độ ý nghĩa có thể được sử dụng để so sánh định lượng các kết quả của các nghiên cứu khác nhau được thực hiện trong các bối cảnh khác nhau. Vì lý do này, mức độ ý nghĩa thường được sử dụng trong phân tích tổng hợp.

3. Giá trị P có thể bị ảnh hưởng bởi kích thước mẫu lớn. Cỡ mẫu càng lớn, sức mạnh thống kê của kiểm định giả thuyết càng lớn, cho phép nó phát hiện ra những tác động dù là nhỏ. Điều này có thể dẫn đến giá trị p thấp, mặc dù mức độ ý nghĩa nhỏ có thể không có ý nghĩa thực tế.

Một ví dụ đơn giản có thể làm rõ điều này: Giả sử chúng ta muốn biết liệu hai kỹ thuật nghiên cứu có dẫn đến điểm kiểm tra khác nhau hay không. Chúng ta có một nhóm 20 sinh viên sử dụng một kỹ thuật nghiên cứu trong khi một nhóm khác gồm 20 sinh viên sử dụng một kỹ thuật nghiên cứu khác. Sau đó, chúng ta cho mỗi học sinh làm bài kiểm tra giống nhau.

Điểm trung bình cho nhóm 1 là **90,65** và điểm trung bình cho nhóm 2 là **90,75**. Độ lệch chuẩn cho mẫu 1 là **2,77** và độ lệch chuẩn cho mẫu 2 là **2,78**.

Khi chúng ta thực hiện kiểm định t hai mẫu độc lập, kết quả là thống kê kiểm định là **-0.113** và giá trị p tương ứng là **0.91**. Sự khác biệt giữa các điểm kiểm tra trung bình không có ý nghĩa thống kê.

Tuy nhiên, nếu xem xét khi cỡ mẫu của hai mẫu đều là **200**, nhưng giá trị trung bình và độ lệch chuẩn vẫn hoàn toàn giống nhau. Trong trường hợp này, với phép thử t hai mẫu độc lập chỉ ra rằng thống kê thử nghiệm là **-1.97** và giá trị p tương ứng dưới **0.05**. Sự khác biệt giữa các trung bình có ý nghĩa thống kê.

Lý do cơ bản mà kích thước mẫu lớn có thể dẫn đến kết luận có ý nghĩa thống kê là do công thức được sử dụng để tính toán thống kê thử nghiệm t :

$$\text{thống kê kiểm định } t = [(x_1 - x_2) - d] / (\sqrt{s^2_1 / n_1 + s^2_2 / n_2})$$

Chú ý rằng khi n_1 và n_2 nhỏ thì toàn bộ mẫu số của thống kê kiểm định t cũng nhỏ. Và ngược lại khi chúng ta chia cho một số nhỏ, chúng ta sẽ có một số lớn. Điều này có nghĩa là thống kê kiểm định t sẽ lớn và giá trị p tương ứng sẽ nhỏ, do đó dẫn đến kết quả có ý nghĩa thống kê.

7.14. ANOVA một chiều

Một **one-way ANOVA** (“phân tích phương sai”) được sử dụng để xác định có hay không có sự khác biệt ý nghĩa thống kê giữa các giá trị trung bình của ba hoặc nhiều nhóm tổng thể nghiên cứu độc lập.

Ví dụ:

Một nhà nghiên cứu tuyển 30 sinh viên để tham gia vào một nghiên cứu. Các học sinh được **chỉ định ngẫu nhiên** để sử dụng một trong ba kỹ thuật học tập trong ba tuần tiếp theo để chuẩn bị cho một kỳ thi. Vào cuối ba tuần, tất cả học sinh đều làm bài kiểm tra giống nhau.

Sử dụng các bước sau để thực hiện ANOVA một chiều nhằm xác định xem điểm trung bình có giống nhau trên cả ba nhóm hay không.

Bước 1: Nhập dữ liệu.

Đầu tiên, chúng ta sẽ nhập điểm thi cho từng nhóm vào ba mảng riêng biệt:

```
#enter exam scores for each group
group1 = [85, 86, 88, 75, 78, 94, 98, 79, 71, 80]
group2 = [91, 92, 93, 85, 87, 84, 82, 88, 95, 96]
group3 = [79, 78, 88, 94, 92, 85, 83, 85, 82, 81]
```

Bước 2: Thực hiện ANOVA một chiều

Tiếp theo, chúng ta sẽ sử dụng hàm `f_oneway()` từ thư viện SciPy để thực hiện ANOVA một chiều:

```
from scipy.stats import f_oneway

#perform one-way ANOVA
f_oneway(group1, group2, group3)

(statistic=2.3575, pvalue=0.1138)
```

Bước 3: Diễn giải kết quả

ANOVA một chiều sử dụng **giả thuyết không** và **giả thuyết thay thế** sau:

- **H_0 (giả thuyết không):** $\mu_1 = \mu_2 = \mu_3 = \dots = \mu_k$ (tất cả các trung bình tổng thể đều bằng nhau)
- **H_1 (giả thuyết thay thế):** ít nhất một trung bình của tổng thể khác với phần còn lại

Với giá trị kiểm tra `f_oneway` ANOVA là **2,3575** và p-giá trị tương ứng là **0,1138**. Vì giá trị p không nhỏ hơn 0,05, chúng ta không thể bác bỏ giả thuyết rỗng. Tức không có đủ bằng chứng để nói rằng có sự khác biệt về điểm thi giữa ba phương pháp nghiên cứu.

7.15. ANOVA hai chiều

Một phép kiểm tra **ANOVA** hai chiều được sử dụng để xác định có hay không có sự khác biệt ý nghĩa thống kê giữa các giá trị trung bình của ba hoặc nhiều nhóm độc lập, được chia theo nhóm dựa trên hai yếu tố.

Mục đích của ANOVA hai chiều là xác định cách hai yếu tố tác động đến biến phản hồi và xác định xem có hay không có sự tương tác giữa hai yếu tố trên biến phản hồi.

Ví dụ:

Một nhà thực vật học muốn biết liệu sự phát triển của thực vật có bị ảnh hưởng bởi sự tiếp xúc với ánh sáng mặt trời và tần suất tưới nước hay không. Cô gieo 30 hạt giống và để chúng phát triển trong hai tháng trong các điều kiện tiếp xúc với ánh sáng mặt trời và tần suất tưới nước khác nhau. Sau hai tháng, cô ấy ghi lại chiều cao của từng cây, tính bằng inch.

Sử dụng các bước sau để thực hiện ANOVA hai chiều để xác định xem tần suất tưới nước và tiếp xúc với ánh sáng mặt trời có ảnh hưởng đáng kể đến sự phát triển của cây hay không và để xác định xem có bất kỳ tác động tương tác nào giữa tần suất tưới nước và tiếp xúc với ánh sáng mặt trời hay không.

Bước 1: Nhập dữ liệu.

Đầu tiên, chúng ta sẽ tạo một DataFrame có chứa ba biến sau:

- **water:** tần suất mỗi cây được tưới: hàng ngày hoặc hàng tuần
- **sun:** mức độ tiếp xúc với ánh sáng mặt trời mà mỗi cây nhận được: thấp, trung bình hoặc cao
- **height:** chiều cao của mỗi cây (tính bằng inch) sau hai tháng

```
import numpy as np
import pandas as pd

#create data
df = pd.DataFrame({'water': np.repeat(['daily', 'weekly'], 15),
                  'sun': np.tile(np.repeat(['low', 'med', 'high'],
5), 2),
                  'height': [6, 6, 6, 5, 6, 5, 5, 6, 4, 5,
                             6, 6, 7, 8, 7, 3, 4, 4, 4, 5,
                             4, 4, 4, 4, 4, 5, 6, 6, 7, 8]})

#view first ten rows of data
df[:10]
```

	water	sun	height
0	daily	low	6
1	daily	low	6
2	daily	low	6
3	daily	low	5
4	daily	low	6
5	daily	med	5
6	daily	med	5
7	daily	med	6
8	daily	med	4
9	daily	med	5

Bước 2: Thực hiện ANOVA hai chiều.

Tiếp theo, chúng ta sẽ thực hiện ANOVA hai chiều bằng cách sử dụng hàm [anova_lm\(\)](#) từ thư viện statsmodels:

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

```
#perform two-way ANOVA
model = ols('height ~ C(water) + C(sun) + C(water):C(sun)',
data=df).fit()
sm.stats.anova_lm(model, typ=2)
```

	sum_sq	df	F	PR(>F)
C(water)	8.533333		1.0	16.0000 0.000527
C(sun)	24.866667	2.0	23.3125	0.000002
C(water):C(sun)	2.466667		2.0	2.3125 0.120667
Residual	12.800000	24.0	NaN	NaN

Bước 3: Diễn giải kết quả

Chúng ta có thể thấy các giá trị p sau cho từng yếu tố trong bảng:

- **water:** p-value = .000527
- **sun:** p-value = .0000002
- **water * sun:** p-value = .120667

Vì giá trị p của nước và mặt trời đều nhỏ hơn 0,05, điều này có nghĩa là cả hai yếu tố đều có ảnh hưởng có ý nghĩa thống kê đến chiều cao của cây.

Và vì giá trị p cho hiệu ứng tương tác (.120667) không nhỏ hơn 0,05, điều này cho chúng ta biết rằng không có hiệu ứng tương tác đáng kể nào giữa việc tiếp xúc với ánh sáng mặt trời và tần suất tưới nước.

***Lưu ý:** Mặc dù kết quả ANOVA cho chúng ta biết rằng tần suất tưới nước và tiếp xúc với ánh sáng mặt trời có ảnh hưởng có ý nghĩa thống kê đến chiều cao của cây, nhưng chúng ta sẽ cần thực hiện các **bài kiểm tra hậu nghiệm (Post Hoc Tests)** để xác định chính xác mức độ khác nhau của nước và ánh sáng mặt trời ảnh hưởng đến chiều cao của cây.*

7.16. Khoảng tin cậy (confidence interval)

Khoảng **tin cậy** cho giá trị **trung bình** là một dải giá trị có khả năng chứa trung bình tổng thể với một mức độ tin cậy nhất định.

Công thức tính khoảng tin cậy (CI):

$$CI = \bar{x} \pm t * (s / \sqrt{n})$$

ở đây:

- **x** : trung bình mẫu

- **t**: giá trị t tương ứng với mức độ tin cậy
- **s**: độ lệch chuẩn mẫu
- **n**: kích thước mẫu

Khoảng tin cậy sử dụng phân phối t

Nếu đang làm việc với một mẫu nhỏ ($n < 30$), chúng ta có thể sử dụng hàm `t.interval()` từ thư viện `scipy.stats` để tính khoảng tin cậy cho giá trị trung bình của tổng thể.

Ví dụ sau đây cho thấy cách tính khoảng tin cậy cho chiều cao trung bình thực của quần thể (tính bằng inch) của một loài thực vật nhất định, với một mẫu gồm 15 phân tử:

```
import numpy as np
import scipy.stats as st

#define sample data
data = [12, 12, 13, 13, 15, 16, 17, 22, 23, 25, 26, 27, 28, 28,
29]

#create 95% confidence interval for population mean weight
st.t.interval(alpha=0.95, df=len(data)-1, loc=np.mean(data),
scale=st.sem(data))

(16.758, 24.042)
```

Khoảng tin cậy 95% cho chiều cao trung bình của dân số thực là **(16,758, 24,042)**.

Chúng ta thấy rằng mức độ tin cậy càng lớn thì khoảng tin cậy càng rộng. Ví dụ dưới đây là cách tính CI 99% cho cùng một dữ liệu trên:

```
#create 99% confidence interval for same sample
st.t.interval(alpha=0.99, df=len(data)-1, loc=np.mean(data),
scale=st.sem(data))

(15.348, 25.455)
```

Khoảng tin cậy 99% cho chiều cao trung bình của dân số thực là **(15,348, 25,455)**. Lưu ý rằng khoảng này rộng hơn khoảng tin cậy 95% trước đó.

Tính khoảng tin cậy sử dụng phân phối chuẩn

Nếu chúng ta đang làm việc với các mẫu lớn hơn ($n \geq 30$), chúng ta có thể giả định rằng phân phối lấy mẫu của trung bình mẫu là phân phối chuẩn (nhờ [Định lý giới hạn trung tâm](#)) và thay vào đó có thể sử dụng hàm [norm.interval\(\)](#) từ thư viện `scipy.stats`.

Ví dụ sau đây cho thấy cách tính khoảng tin cậy cho chiều cao trung bình thực của quần thể (tính bằng inch) của một loài thực vật, với mẫu gồm 50 phần tử:

```
import numpy as np
import scipy.stats as st

#define sample data
np.random.seed(0)
data = np.random.randint(10, 30, 50)

#create 95% confidence interval for population mean weight
st.norm.interval(alpha=0.95, loc=np.mean(data), scale=st.sem(data))

(17.40, 21.08)
```

Khoảng tin cậy 95% cho chiều cao trung bình của dân số thực là **(17,40, 21,08)** .

Tương tự như phân phối t, mức độ tin cậy lớn hơn dẫn đến khoảng tin cậy rộng hơn.

Ví dụ: Cách tính CI 99% cho cùng một dữ liệu ở ví dụ trên:

```
#create 99% confidence interval for same sample
st.norm.interval(alpha=0.99, loc=np.mean(data), scale=st.sem(data))

(16.82, 21.66)
```

Khoảng tin cậy 99% cho chiều cao trung bình của dân số thực là **(17,82, 21,66)** .

Hiểu thêm về khoảng tin cậy

Giả sử khoảng tin cậy 95% của chúng ta đối với chiều cao trung bình thực của quần thể thực vật là: **(16,758, 24,042)**.

Tức có 95% khả năng khoảng tin cậy của [16,758, 24,042] chứa chiều cao trung bình thực của cây trồng. Nói cách khác có 5% cơ hội để trung bình mẫu của chiều cao của các cây đã thu thập nằm ngoài khoảng tin cậy. Nói cách khác, có 5% khả năng là chiều cao trung bình của quần thể thực vật nhỏ hơn 16,758 inch hoặc lớn hơn 24,042 inch.

Tóm tắt chương 7

Chương này đã trình bày các vấn đề của bài toán ước lượng và kiểm định thống kê cơ bản, thông qua các ví dụ minh họa.

- Kiểm định trung bình cho một tổng thể
- Kiểm định sai khác trung bình cho hai tổng thể độc lập
- Kiểm định sai khác trung bình cho mẫu cặp
- + Kiểm định giả thiết cho tỷ lệ
- Kiểm định tỷ lệ cho một tổng thể
- Kiểm định sai khác tỷ lệ cho hai tổng thể
- + Phân tích ANOVA

Để hiểu rõ bản chất của các bài toán, người đọc nên tìm hiểu các công thức tính toán của các bài toán kiểm định (xem thêm trong các tài liệu đã dẫn trong phần danh mục tham khảo), để thực hiện cài đặt lại các hàm thay vì sẵn có của Python, nhằm hiểu rõ bản chất của quá trình kiểm định thống kê.

Câu hỏi ôn tập và bài tập chương 7

- *Hãy nêu ý nghĩa của giả thuyết không và giả thuyết thay thế trong bài toán kiểm định*
- *Trình bày và cho được ví dụ thực tiễn về các bước cơ bản để thực hiện bài toán kiểm định.*
- *Cài đặt thực hiện các ví dụ minh họa trong chương.*
- *Giải thích các câu lệnh cơ bản trong các ví dụ*
- *Vận dụng các ví dụ minh họa cho dữ liệu thực tiễn được lưu trong file **nghegi.xlsx** tại địa chỉ:*

<https://docs.google.com/spreadsheets/d/1q7RVfoTKEuLP0Ud0a50mJu7e1-YJJbUe/edit?usp=sharing&ouid=106843557461060771783&rtpof=true&sd=true>

- *Tìm hiểu các công thức tính toán của các bài toán kiểm định, để thực hiện cài đặt lại các hàm thay vì sẵn có của Python, nhằm hiểu rõ bản chất của quá trình kiểm định thống kê.*

Chương 8. KIỂM ĐỊNH PHI THAM SỐ

Mục đích

Trong các phần trước ta đã nói đến các kiểm định giả thuyết về các đặc trưng như trung bình và tỷ lệ tổng thể và thường dựa trên giả định tổng thể có phân phối chuẩn. Trong phần này ta đề cập đến các kiểm định không yêu cầu về điều kiện phân phối chuẩn của tổng thể.

- Kiểm định Wilcoxon
- Kiểm định Mann-Whitney
- Kiểm định về tính độc lập của hai biến định tính
- Kiểm định phân phối
- Kiểm định phương sai

(Một số mã nguồn trong chương này được tham khảo từ [6], [7])

Yêu cầu

- Trình bày và cho được ví dụ thực tiễn về các bước cơ bản để thực hiện bài toán kiểm định phi tham số.
- Cài đặt thực hiện các ví dụ minh họa trong chương.
- Giải thích các câu lệnh cơ bản trong các ví dụ
- Tìm hiểu các công thức tính toán của các bài toán kiểm định phi tham số, để thực hiện cài đặt lại các hàm thay vì sẵn có của Python, nhằm hiểu rõ bản chất của quá trình kiểm định thống kê.

8.1. Kiểm tra Wilcoxon

Phép kiểm tra Wilcoxon là một thống kê kiểm định phi tham số, phép kiểm tra này được xem là mở rộng của phép kiểm tra t-test so sánh mẫu theo cặp. Phép kiểm tra này được sử dụng để kiểm tra xem có hay không sự khác biệt đáng kể giữa hai giá trị trung bình tổng thể, khi phân bố của sự khác biệt giữa hai mẫu được giả định là không có phân phối chuẩn.

Ví dụ 1

Các nhà nghiên cứu muốn biết liệu cách xử lý nhiên liệu mới có dẫn đến sự thay đổi giá trị trung bình mpg (miles per gallon - Thông số đo lường hiệu quả sử dụng

nhân liệu của các loại xe) của một chiếc xe nhất định hay không. Để kiểm tra điều này, họ đo mpg của 12 chiếc xe có và không có xử lý nhân liệu.

Bước 1: Tạo dữ liệu

Đầu tiên, chúng ta sẽ tạo hai mảng để giữ các giá trị mpg cho mỗi nhóm ô tô:

```
group1 = [20, 23, 21, 25, 18, 17, 18, 24, 20, 24, 23, 19]
group2 = [24, 25, 21, 22, 23, 18, 17, 28, 24, 27, 21, 23]
```

Bước 2: Tiến hành kiểm tra Wilcoxon.

Tiếp theo, chúng ta sẽ sử dụng hàm `wilcoxon()` từ thư viện `scipy.stats` để tiến hành kiểm tra Wilcoxon, sử dụng cú pháp sau:

```
wilcoxon(x, y, alternative='two-sided')
```

ở đây:

- **x**: mảng chứa các phần tử của mẫu 1
- **y**: mảng chứa các phần tử của mẫu 2

```
import scipy.stats as stats

#perform the Wilcoxon-Signed Rank Test
stats.wilcoxon(group1, group2)

(statistic=10.5, pvalue=0.044)
```

Thống kê kiểm định là **10,5** và p - giá trị hai phía tương ứng là **0,044**.

Bước 3: Diễn giải kết quả

Trong ví dụ này, kiểm tra Wilcoxon sử dụng các giả thuyết không và giả thuyết thay thế sau:

H₀ : mpg bằng nhau giữa hai nhóm

H_A : mpg *không* bằng nhau giữa hai nhóm

Vì p – giá trị là **0,044** nhỏ hơn 0,05, chúng ta bác bỏ giả thuyết không. Tức có đủ bằng chứng để nói rằng mpg trung bình thực sự không bằng nhau giữa hai nhóm.

8.2. Kiểm tra tính độc lập Chi-Square

Phép thử **Chi-Square về tính độc lập** được sử dụng để xác định xem có hay không mối liên hệ đáng kể giữa hai biến phân loại.

Ví dụ:

Giả sử chúng ta muốn biết liệu giới tính có liên quan đến ưu tiên đảng phái chính trị hay không. Lấy một mẫu ngẫu nhiên đơn giản gồm 500 cử tri và khảo sát họ về sở thích đảng chính trị của họ. Bảng sau đây cho thấy kết quả của cuộc khảo sát:

	Đảng cộng hòa	Đảng viên dân chủ	Độc lập	Toàn bộ
Nam giới	120	90	40	250
Giống cái	110	95	45	250
Toàn bộ	230	185	85	500

Sử dụng các bước sau để thực hiện kiểm tra tính độc lập Chi-Square bằng Python để xác định xem giới tính có liên quan đến lựa chọn đảng phái chính trị hay không.

Bước 1: Tạo dữ liệu.

Đầu tiên, chúng ta sẽ tạo một bảng để chứa dữ liệu

```
data = [[120, 90, 40], [110, 95, 45]]
```

Bước 2: Thực hiện bài kiểm tra tính độc lập của Chi-Square.

Tiếp theo, chúng ta có thể thực hiện Kiểm tra Độc lập Chi-Square bằng cách sử dụng **hàm chi2_contingency** từ thư viện SciPy, với cú pháp sau:

chi2_contingency (observed)

ở đây:

- **observed:** Một bảng lưu các giá trị quan sát.

Đoạn mã sau đây cho thấy cách sử dụng hàm này trong ví dụ cụ thể trên:

```
import scipy.stats as stats

#perform the Chi-Square Test of Independence
stats.chi2_contingency(data)
```

```
(0.864,
0.649,
2,
array([[115. , 92.5, 42.5],
       [115. , 92.5, 42.5]]))
```

Diễn giải đầu ra như sau:

- ✓ Thống kê kiểm tra Chi-Square: **0,864**
- ✓ giá trị p: **0,649**
- ✓ Bậc tự do: **2** (được tính là: (số lượng hàng -1) hoặc (số lượng cột-1))
- ✓ array: Mảng cuối cùng hiển thị các giá trị dự kiến cho mỗi ô trong bảng dự phòng.

Các giả thuyết không và giả thuyết thay thế như sau:

- **H_0 : (giả thuyết không)** Hai biến độc lập.
- **H_1 : (giả thuyết thay thế)** Hai biến *không* độc lập.

Vì giá trị p (.649) của thử nghiệm không nhỏ hơn 0,05, chúng ta không thể bác bỏ giả thuyết không. Tức không có đủ bằng chứng để nói rằng có mối liên hệ giữa giới tính và sở thích đảng phái chính trị.

Nói cách khác, giới tính và ưu tiên đảng phái chính trị là độc lập.

8.3. Giá trị tới hạn Chi-Square

Khi tiến hành kiểm tra Chi-Square, ta sẽ nhận được kết quả thống kê. Để xác định xem kết quả của kiểm tra Chi-Square có ý nghĩa thống kê hay không, ta có thể so sánh kết quả thống kê với **giá trị tới hạn Chi-Square** . Nếu thống kê thử nghiệm lớn hơn giá trị tới hạn Chi-Square, thì kết quả của thử nghiệm có ý nghĩa thống kê.

Để tìm giá trị tới hạn Chi-Square, ta cần xác định:

- Mức ý nghĩa (các lựa chọn phổ biến là 0,01, 0,05 và 0,10)
- Mức độ tự do

Sử dụng hai giá trị này, chúng ta có thể xác định giá trị Chi-Square để so sánh với thống kê thử nghiệm.

Cách tìm giá trị tới hạn Chi-Square

Để tìm giá trị tới hạn Chi-Square trong Python, chúng ta có thể sử dụng hàm [`scipy.stats.chi2.ppf\(\)`](#), với cú pháp sau:

`scipy.stats.chi2.ppf(q, df)`

ở đây:

- **q**: Mức ý nghĩa sử dụng
- **df**: Bậc tự do

Hàm này trả về giá trị tới hạn từ phân phối Chi-Square dựa trên mức ý nghĩa và bậc tự do được cung cấp.

Ví dụ, giả sử chúng ta muốn tìm giá trị tới hạn Chi-Square cho mức ý nghĩa 0,05 và bậc tự do = 11.

```
import scipy.stats

#find Chi-Square critical value
scipy.stats.chi2.ppf(1-.05, df=11)

19.67514
```

Giá trị tới hạn Chi-Square cho mức ý nghĩa 0,05 và bậc tự do = 11 là **19,67514**.

Do đó, nếu ta đang thực hiện một số loại kiểm tra Chi-Square, chúng ta có thể so sánh thống kê kiểm tra Chi-Square với **19,67514**. Nếu thống kê thử nghiệm lớn hơn 19.67514, kết quả của thử nghiệm có ý nghĩa thống kê.

Lưu ý rằng các giá trị nhỏ hơn của mức ý nghĩa sẽ dẫn đến các giá trị tới hạn Chi-Square lớn hơn. Ví dụ sau xem xét giá trị tới hạn Chi-Square cho mức ý nghĩa là **0,01** và bậc tự do = 11.

```
scipy.stats.chi2.ppf(1-.01, df=11)

24.72497
```

Ví dụ tiếp theo cho ra giá trị tới hạn Chi-Square với cùng bậc tự do chính xác, nhưng với mức ý nghĩa là **0,005**:

```
scipy.stats.chi2.ppf(1-.005 df=11)

26.75685
```

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2.html#scipy.stats.chi2>

8.4. Kiểm tra Mann-Whitney U

Phép kiểm tra **Mann-Whitney U** được sử dụng để so sánh sự khác biệt giữa hai mẫu khi phân bố mẫu không có phân phối chuẩn và kích thước mẫu nhỏ ($n < 30$).

Phép kiểm tra này được xem là phép kiểm tra phi tham số tương đương với **phép thử t - test hai mẫu**.

Ví dụ 1:

Các nhà nghiên cứu muốn biết liệu việc xử lý nhiên liệu có dẫn đến sự thay đổi của giá trị trung bình mpg (miles per gallon of gasoline-equivalent.” Thông số đo lường hiệu quả sử dụng nhiên liệu của các loại xe) của một chiếc ô tô hay không. Để kiểm tra điều này, họ đo nồng độ mpg của 12 chiếc xe có xử lý nhiên liệu và 12 chiếc không xử lý nhiên liệu.

Vì kích thước mẫu nhỏ và các nhà nghiên cứu nghi ngờ rằng phân bố mẫu không có phân phối chuẩn, họ quyết định thực hiện kiểm tra Mann-Whitney U để xác định xem có sự khác biệt có ý nghĩa thống kê về giá trị mpg giữa hai nhóm hay không.

Bước 1: Tạo dữ liệu

Đầu tiên, chúng ta sẽ tạo hai mảng để giữ các giá trị mpg cho mỗi nhóm ô tô:

```
group1 = [20, 23, 21, 25, 18, 17, 18, 24, 20, 24, 23, 19]
group2 = [24, 25, 21, 22, 23, 18, 17, 28, 24, 27, 21, 23]
```

Bước 2: Tiến hành kiểm tra U Mann-Whitney

Tiếp theo, chúng ta sẽ sử dụng hàm **mannwhitneyu()** từ thư viện **scipy.stats** để tiến hành kiểm tra Mann-Whitney U, sử dụng cú pháp sau:

```
mannwhitneyu(x, y, use_continuity=True, alternative=None)
```

ở đây:

- **x**: Mảng chứa mẫu thống kê của nhóm 1
- **y**: Mảng chứa mẫu thống kê của nhóm 2
- **use_continuity**: tham số tùy chọn.

- **alternative: xác định giả thuyết thay thế.** Giá trị mặc định là 'None', để tính p-giá trị 1 phía hay 2 phía. Giá trị là 'two-sided', 'less', 'greater'.

```
import scipy.stats as stats

#perform the Mann-Whitney U test
stats.mannwhitneyu(group1, group2, alternative='two-sided')

(statistic=50.0, pvalue=0.2114)
```

Thống kê thử nghiệm là **50,0** và p- giá trị hai phía tương ứng là **0,2114** .

Bước 3: Diễn giải kết quả

Trong ví dụ này, phép thử Mann-Whitney U sử dụng các giả thuyết không và giả thuyết thay thế sau:

H_0 : mpg bằng nhau giữa hai nhóm

H_A : mpg *không* bằng nhau giữa hai nhóm

Vì p-giá trị là **0,2114** không nhỏ hơn 0,05, chúng ta không thể bác bỏ giả thuyết không.

Tức không có đủ bằng chứng để nói rằng mpg trung bình thực sự là khác nhau giữa hai nhóm.

8.5. Kiểm tra mối liên quan giữa hai biến phân loại (kiểm tra Fisher)

Kiểm định Fisher được sử dụng để xác định xem có hay không mối liên hệ đáng kể giữa hai biến phân loại. Nó thường được sử dụng thay thế cho Kiểm tra Độc lập Chi-Square khi một hoặc nhiều ô trong bảng dữ liệu nhỏ hơn 5.

Ví dụ:

Giả sử chúng ta muốn biết liệu giới tính có liên quan đến ưu tiên đảng phái chính trị tại một trường đại học ở Mỹ hay không.

Để khám phá điều này, chúng ta thăm dò ngẫu nhiên 25 sinh viên trong khuôn viên trường. Số lượng sinh viên là Đảng viên Đảng Dân chủ hoặc Đảng Cộng hòa, dựa trên giới tính, được trình bày trong bảng dưới đây:

	Đảng viên dân chủ	Đảng cộng hòa
Nữ giới	số 8	4

Nam giới

4

9

Để xác định xem có mối liên hệ đáng kể về mặt thống kê giữa giới tính và sở thích đảng chính trị hay không, chúng ta có thể sử dụng các bước sau để thực hiện Kiểm tra chính xác của Fisher bằng Python:

Bước 1: Nhập dữ liệu.

Đầu tiên, ta tạo một bảng để chứa dữ liệu

```
data = [[8, 4], [4, 9]]
```

Bước 2: Thực hiện Kiểm tra Chính xác của Fisher.

Tiếp theo, chúng ta có thể thực hiện Kiểm tra Chính xác của Fisher bằng cách sử dụng [hàm Fisher_exact](#) từ thư viện SciPy, sử dụng cú pháp sau:

```
fisher_exact(table, alternative='two-sided')
```

ở đây:

- **table:** Một bảng contingency 2×2
- **alternative:** Xác định giả thuyết thay thế. Mặc định là 'hai mặt', nhưng ta cũng có thể chọn 'ít hơn' hoặc 'lớn hơn' cho các thử nghiệm một mặt.

Đoạn mã thực hiện như sau:

```
import scipy.stats as stats

print(stats.fisher_exact(data))

(4.5, 0.1152)
```

Giá trị p cho các bài kiểm tra là **0,1152**.

Kiểm tra chính xác của Fisher sử dụng các giả thuyết khác và giả thuyết thay thế sau:

- **H_0 :** (giả thuyết không) Hai biến độc lập.
- **H_1 :** (giả thuyết thay thế) Hai biến *không* độc lập.

Vì giá trị p này không nhỏ hơn 0,05 nên chúng ta không bác bỏ giả thuyết không. Tức không có đủ bằng chứng để nói rằng có mối liên hệ đáng kể giữa giới tính và sở thích đảng phái chính trị.

Nói cách khác, giới tính và ưu tiên đảng phái chính trị là độc lập.

8.6. Kiểm tra tính chuẩn của dữ liệu

Có nhiều cách để nhận biết một phân phối chuẩn

(1) Đơn giản nhất là xem biểu đồ với đường cong chuẩn (Histograms with normal curve) với dạng hình chuông đối xứng với tần số cao nhất nằm ngay giữa và các tần số thấp dần nằm ở 2 bên. Trị trung bình (mean) và trung vị (median) gần bằng nhau và độ xiên (skewness) gần bằng zero.

(2) Vẽ biểu đồ xác suất chuẩn (Probability plots).

(3) Dùng phép kiểm định Kolmogorov-Smirnov khi cỡ mẫu lớn hơn 50 hoặc phép kiểm Shapiro-Wilk khi cỡ mẫu nhỏ hơn 50. Được coi là có phân phối chuẩn khi mức ý nghĩa (Sig.) lớn hơn 0,05.

KIỂM ĐỊNH TÍNH PHÙ HỢP

Phép kiểm định tính phù hợp (Goodness of Fit) được sử dụng để kiểm tra độ phù hợp của dữ liệu mẫu với một phân phối của tổng thể. Tổng thể có thể có phân phối chuẩn hoặc phân phối Weibull. Nói cách khác, kiểm định phù hợp khẳng định rằng dữ liệu mẫu thể hiện độ chính xác của dữ liệu mà chúng ta đang kỳ vọng tìm ra từ tập dữ liệu tổng thể thực tế. Các kiểm thử sau thường được sử dụng:

- Chi-bình phương (Chi-square)
- Kolmogorov-Smirnov
- Anderson-Darling
- Shapiro-Wilk

Phép thử Chi - square là phép kiểm tra hay được sử dụng nhất trong phép kiểm định tính phù hợp và cũng được dùng cho các phân phối rời rạc như phân phối nhị thức và phân phối Poisson, trong khi đó phép thử Kolmogorov-Smirnov và Anderson-Darling dùng để kiểm định tính phù hợp trong trường hợp phân phối liên tục.

8.7. Kiểm tra Shapiro-Wilk

Kiểm tra **Shapiro-Wilk** là một phép kiểm tra chuẩn. Nó được sử dụng để xác định xem một mẫu dữ liệu thu thập có **phân phối chuẩn hay không** .

Để thực hiện kiểm tra Shapiro-Wilk, chúng ta có thể sử dụng hàm `scipy.stats.shapiro ()` , theo cú pháp sau:

scipy.stats.shapiro (x)

ở đây:

- **x**: Mảng dữ liệu mẫu.

Hàm này trả về một thống kê kiểm tra và p – giá trị, tương ứng. Nếu giá trị p dưới một mức ý nghĩa nhất định, chúng ta có đủ bằng chứng để nói rằng dữ liệu mẫu không có phân phối chuẩn.

Ví dụ 1:

Giả sử chúng ta có dữ liệu mẫu sau:

```
from numpy.random import seed
from numpy.random import randn

#set seed (e.g. make this example reproducible)
seed(0)

#generate dataset of 100 random values that follow a standard normal
distribution
data = randn(100)
```

Đoạn mã sau cho biết cách thực hiện kiểm tra Shapiro-Wilk trên mẫu 100 giá trị dữ liệu trên để xác định xem nó có phân phối chuẩn hay không:

```
from scipy.stats import shapiro

#perform Shapiro-Wilk test
shapiro(data)

ShapiroResult(statistic=0.9926937818527222, pvalue=0.8689165711402893)
```

Từ kết quả đầu ra, chúng ta có thể thấy rằng thống kê thử nghiệm là **0,9927** và p – giá trị tương ứng là **0,8689**. Vì p – giá trị không nhỏ hơn 0,05, chúng ta không thể bác bỏ giả thuyết rỗng. Tức không có đủ bằng chứng để nói rằng dữ liệu mẫu không có phân phối chuẩn.

Ví dụ 2:

Giả sử chúng ta có dữ liệu mẫu sau:

```
from numpy.random import seed
from numpy.random import poisson

#set seed (e.g. make this example reproducible)
seed(0)
```



```
#generate dataset of 100 values that follow a Poisson distribution with mean=5
data = poisson(5, 100)
```

Đoạn mã sau cho biết cách thực hiện kiểm tra Shapiro-Wilk trên mẫu 100 giá trị dữ liệu này để xác định xem nó có phân phối chuẩn hay không:

```
from scipy.stats import shapiro

#perform Shapiro-Wilk test
shapiro(data)

ShapiroResult(statistic=0.9581913948059082, pvalue=0.002994443289935589)
```

Từ kết quả đầu ra, chúng ta có thể thấy rằng thống kê thử nghiệm là **0,9582** và p - giá trị tương ứng là **0,00299**. Vì p - giá trị nhỏ hơn 0,05, nên chúng ta bác bỏ giả thuyết không. Tức có đủ bằng chứng để nói rằng dữ liệu mẫu không có phân phối chuẩn.

8.8. Kiểm tra Anderson-Darling

Kiểm tra **Anderson-Darling** là phép kiểm định tính phù hợp với một phân phối chỉ định. Kiểm tra này thường được sử dụng nhất để xác định xem dữ liệu có tuân theo phân phối chuẩn hay không.

Đây là một giả định phổ biến được sử dụng trong nhiều thử nghiệm thống kê như [hồi quy](#), [ANOVA](#), [t-tests](#) và nhiều loại khác.

Ví dụ:

Để thực hiện kiểm tra Anderson-Darling bằng Python, chúng ta có thể sử dụng hàm [anderson\(\)](#) từ thư viện `scipy.stats`, sử dụng cú pháp sau:

`anderson(x, dist='norm')`

ở đây:

- **x** : mảng dữ liệu mẫu
- **dist** : loại phân phối để kiểm tra. Mặc định là 'phân phối chuẩn' nhưng ta cũng có thể chỉ định tham số khác như: 'expon', 'logistic', 'gumbel', 'gumbel_l', 'gumbel_r', 'extreme1'.

Ví dụ: Cách thực hiện kiểm tra Anderson-Darling trên một mẫu gồm 50 biến ngẫu nhiên có phân phối chuẩn:

```
import numpy as np

#create data
np.random.seed(0)
data = np.random.normal(size=50)

#perform Anderson-Darling Test
from scipy.stats import anderson
anderson(data)

AndersonResult(statistic=0.15006999533388665,
                critical_values=array([0.538, 0.613, 0.736, 0.858,
1.021]),
                significance_level=array([15. , 10. , 5. , 2.5, 1. ]))
```

Thống kê thử nghiệm là **0,150** . Chúng ta có thể so sánh giá trị này với từng giá trị tới hạn tương ứng với từng mức ý nghĩa để xem kết quả kiểm tra có ý nghĩa hay không. Ví dụ:

- Giá trị tới hạn của $\alpha = 0,01$ là **1,021** . Do thống kê kiểm định (0,150) không lớn hơn giá trị tới hạn này nên các kết quả không có ý nghĩa với mức ý nghĩa 0,01.
- Giá trị tới hạn của $\alpha = 0,025$ là **0,858** . Do thống kê kiểm định (0,150) không lớn hơn giá trị tới hạn này nên các kết quả không có ý nghĩa với mức ý nghĩa 0,025.

Chúng ta có thể thấy rằng kết quả kiểm định không có ý nghĩa ở bất kỳ mức ý nghĩa nào, chúng ta sẽ không bác bỏ giả thuyết không của kiểm định. Do đó, chúng ta không có đủ bằng chứng để nói rằng dữ liệu mẫu không có phân phối chuẩn (Chúng ta đã sử dụng hàm **np.rand.normal()** để tạo một mẫu gồm 50 giá trị được phân phối chuẩn).

Nếu chúng ta thực hiện kiểm tra Anderson-Darling trên một mẫu gồm 50 số nguyên ngẫu nhiên từ 0 đến 10:

```
import numpy as np

#create data
np.random.seed(0)
data = np.random.randint(0, 10, size=50)

#perform Anderson-Darling Test
from scipy.stats import anderson
anderson(data)

AndersonResult(statistic=1.1926463985076836,
                critical_values=array([0.538, 0.613, 0.736, 0.858,
1.021]),
                significance_level=array([15. , 10. , 5. , 2.5, 1. ]))
```

Thống kê thử nghiệm là **1.1926** . Chúng ta có thể so sánh giá trị này với từng giá trị tới hạn tương ứng với từng mức ý nghĩa để xem kết quả kiểm tra có ý nghĩa hay không. Ví dụ:

- Giá trị tới hạn của $\alpha = 0,01$ là **1,021** . Vì thống kê thử nghiệm (1.1926) lớn hơn giá trị tới hạn này, nên kết quả có ý nghĩa với mức ý nghĩa 0,01.
- Giá trị tới hạn của $\alpha = 0,025$ là **0,858** . Vì thống kê kiểm định (1.1926) lớn hơn giá trị tới hạn này, nên kết quả có ý nghĩa với mức ý nghĩa 0,025.

Chúng ta có thể thấy rằng các kết quả kiểm định đều có ý nghĩa ở mọi mức ý nghĩa, có nghĩa là chúng ta có thể bác bỏ giả thuyết không của kiểm định, cho dù chúng ta chọn sử dụng mức ý nghĩa nào. Tức chúng ta có đủ bằng chứng để nói rằng dữ liệu mẫu không có phân phối chuẩn (Chúng ta sử dụng hàm **np.rand.randint()** để tạo một mẫu gồm 50 số nguyên ngẫu nhiên từ 0 đến 10, sẽ không chắc chắn sẽ tuân theo phân phối chuẩn).

8.9. Kiểm tra Kolmogorov-Smirnov

Kiểm tra **Kolmogorov-Smirnov** được sử dụng để kiểm tra xem mẫu có một phân bố nhất định hay không.

Để thực hiện kiểm tra Kolmogorov-Smirnov bằng Python, chúng ta có thể sử dụng [`scipy.stats.kstest\(\)`](#) cho kiểm tra một mẫu hoặc [`scipy.stats.kstest_2samp\(\)`](#) cho kiểm tra hai mẫu.

Ví dụ 1: Thử nghiệm Kolmogorov-Smirnov 1 mẫu.

Giả sử chúng ta có dữ liệu mẫu sau:

```
from numpy.random import seed
from numpy.random import poisson

#set seed (e.g. make this example reproducible)
seed(0)

#generate dataset of 100 values that follow a Poisson distribution with
mean=5
data = poisson(5, 100)
```

Đoạn mã sau trình bày cách thực hiện kiểm tra Kolmogorov-Smirnov trên mẫu 100 giá trị dữ liệu này để xác định xem nó có phân phối chuẩn hay không:

```
from scipy.stats import kstest
```

```
#perform Kolmogorov-Smirnov test
kstest(data, 'norm')

KstestResult(statistic=0.9072498680518208, pvalue=1.0908062873170218e-103)
```

Chúng ta có thể thấy rằng các số liệu thống kê thử nghiệm là **0,9072** và p-giá trị tương ứng là **1.0908e-103**. Vì giá trị p nhỏ hơn 0,05, nên chúng ta bác bỏ giả thuyết không. Chúng ta có đủ bằng chứng để nói rằng dữ liệu mẫu không có phân phối chuẩn (Chúng ta đã tạo dữ liệu mẫu bằng cách sử dụng hàm **poisson()**, hàm này tạo ra các giá trị ngẫu nhiên tuân theo [phân phối Poisson](#)).

Ví dụ 2: thử nghiệm Kolmogorov-Smirnov 2 mẫu

Giả sử chúng ta có hai tập dữ liệu mẫu sau:

```
from numpy.random import seed
from numpy.random import randn
from numpy.random import lognormal

#set seed (e.g. make this example reproducible)
seed(0)

#generate two datasets
data1 = randn(100)
data2 = lognormal(3, 1, 100)
```

Đoạn mã sau cho biết cách thực hiện kiểm tra Kolmogorov-Smirnov trên hai mẫu để xác định xem chúng có cùng một phân phối hay không:

```
from scipy.stats import ks_2samp

#perform Kolmogorov-Smirnov test
ks_2samp(data1, data2)

KstestResult(statistic=0.99, pvalue=4.417521386399011e-57)
```

Chúng ta có thể thấy rằng thống kê thử nghiệm là **0,99** và p - giá trị tương ứng là **4,4175e-57**. Vì p - giá trị nhỏ hơn 0,05, nên chúng ta bác bỏ giả thuyết không. Tức có đủ bằng chứng để nói rằng hai tập dữ liệu mẫu không cùng một phân phối (Chúng ta đã tạo các giá trị cho mẫu đầu tiên bằng cách sử dụng phân phối chuẩn chuẩn và các giá trị cho mẫu thứ hai bằng cách sử dụng phân phối lognormal).

8.10. Kiểm tra phân phối xác suất của biến phân loại bằng phép kiểm tra Chi-Square

Để xác định một biến phân loại có tuân theo phân phối xác suất giả định trước hay không, chúng ta có thể sử dụng phép kiểm tra Chi-square như sau.

Ví dụ:

Một chủ cửa hàng tuyên bố rằng một lượng khách hàng đến cửa hàng của anh ta mỗi ngày trong tuần là như nhau. Để kiểm tra giả thuyết này, một nhà nghiên cứu ghi lại số lượng khách hàng đến cửa hàng trong một tuần nhất định và thu được kết quả sau:

- **Thứ hai:** 50 khách hàng
- **Thứ ba:** 60 khách hàng
- **Thứ 4:** 40 khách hàng
- **Thứ 5:** 47 khách hàng
- **Thứ sáu:** 53 khách hàng

Sử dụng các bước sau để xác định xem dữ liệu có phù hợp với mong đợi của chủ cửa hàng hay không.

Bước 1: Tạo dữ liệu

Đầu tiên, chúng ta sẽ tạo hai mảng để lưu số lượng khách hàng được quan sát và mong đợi của chúng ta cho mỗi ngày:

```
expected = [50, 50, 50, 50, 50]
observed = [50, 60, 40, 47, 53]
```

Bước 2:

Tiếp theo, chúng ta có thể thực hiện kiểm tra Chi-Square bằng cách sử dụng **hàm chisquare** từ thư viện SciPy, sử dụng cú pháp sau:

chisquare (f_obs, f_exp)

ở đây:

- **f_obs:** Một mảng các số đếm được quan sát.
- **f_exp:** Một mảng số lượng mong đợi. Theo mặc định, mỗi danh mục được giả định là có khả năng như nhau.

Đoạn mã sau đây cho thấy cách sử dụng hàm này trong ví dụ cụ thể trên:

```
import scipy.stats as stats

#perform Chi-Square Goodness of Fit Test
stats.chisquare(f_obs=observed, f_exp=expected)

(statistic=4.36, pvalue=0.35947)
```

Thống kê kiểm định Chi-Square là **4,36** và giá trị p tương ứng là **0,35947**.

Lưu ý rằng giá trị p tương ứng với giá trị Chi-Square với $n-1$ bậc tự do (dof), trong đó n là số loại khác nhau. Trong trường hợp này, $dof = 5-1 = 4$.

Trong phép kiểm tra này, chúng ta đã sử dụng các giả thuyết không và giả thuyết thay thế như sau:

- **H_0 : (giả thuyết rỗng)** Một biến tuân theo phân phối giả thiết.
- **H_1 : (giả thuyết thay thế)** Một biến không tuân theo phân phối theo giả thuyết.

Vì giá trị p (.35947) không nhỏ hơn 0,05, chúng ta không thể bác bỏ giả thuyết không. Tức không có đủ bằng chứng để nói rằng phân phối thực sự của số lượng khách hàng đến cửa hàng là khác với phân phối mà chủ cửa hàng đã tuyên bố.

8.11. Kiểm tra tỷ lệ (kiểm tra McNemar)

McNemar's Test được sử dụng để xác định xem có sự khác biệt có ý nghĩa thống kê về tỷ lệ giữa 2 tập dữ liệu mẫu hay không.

Ví dụ:

Giả sử các nhà nghiên cứu muốn biết liệu một video tiếp thị nào đó có thể thay đổi quan điểm của mọi người về một luật cụ thể hay không. Họ khảo sát 100 người để tìm hiểu xem họ làm hay không ủng hộ luật. Sau đó, họ cho tất cả 100 người xem video tiếp thị và khảo sát họ một lần nữa khi video kết thúc.

Bảng sau đây cho thấy tổng số người đã ủng hộ luật cả trước và sau khi xem video:

	Trước khi tiếp thị video	
Sau khi tiếp thị video	Ủng hộ	Không hỗ trợ

Ủng hộ	30	40
Không hỗ trợ	12	18

Để xác định xem có sự khác biệt có ý nghĩa thống kê về tỷ lệ người ủng hộ luật trước và sau khi xem video hay không, chúng tôi có thể thực hiện Thử nghiệm của McNemar.

Bước 1: Tạo dữ liệu.

Chúng ta tạo một bảng để chứa dữ liệu:

```
data = [[30, 40], [12, 18]]
```

Bước 2: Thực hiện Kiểm tra McNemar

Tiếp theo, chúng ta có thể sử dụng hàm `mcnemar()` từ thư viện Python statsmodels, sử dụng cú pháp sau:

```
mcnemar(table, exact=True, correction=True)
```

Ở đây:

- **table:** Một bảng pivot có dạng 2 hàng 2 cột
- **exact:** Nếu tham số này là **true**, khi đó phân phối mũ sẽ được sử dụng để kiểm định, ngược lại chúng ta sử dụng phân phối Chi – Square.
- **correction:** Nếu tham số này là **true**, khi đó các hiệu chỉnh được thực hiện. Theo quy tắc chung, hiệu chỉnh này thường được áp dụng khi bất kỳ số ô nào trong bảng nhỏ hơn 5..

Xem xét đoạn mã sau để có thể rõ cách sử dụng hàm này:

```
from statsmodels.stats.contingency_tables import mcnemar

#McNemar's Test with no continuity correction
print(mcnemar(data, exact=False))

pvalue      0.000181
statistic    14.019

#McNemar's Test with continuity correction
print(mcnemar(data, exact=False, correction=False))

pvalue      0.000103
statistic    15.077
```

Trong cả hai trường hợp - cho dù chúng ta có áp dụng hiệu chỉnh liên tục hay không - giá trị p của thử nghiệm đều nhỏ hơn 0,05.

Tức trong cả hai trường hợp, chúng ta sẽ bác bỏ giả thuyết không và kết luận rằng tỷ lệ người ủng hộ luật trước và sau khi xem video tiếp thị là khác nhau có ý nghĩa thống kê.

8.12. Kiểm tra nhị thức

Phép **thử nhị thức** nhằm so sánh một tỷ lệ mẫu với một tỷ lệ giả định.

Ví dụ, giả sử chúng ta có một con súc sắc 6 mặt. Nếu chúng ta tung nó 12 lần, chúng ta mong đợi mặt có 3 chấm hiển thị với tỷ lệ $1/6$, tức có $12 * (1/6) = 2$ lần xuất hiện.

Nếu mặt có 3 chấm xuất hiện 4 lần, thì đó có phải là bằng chứng cho thấy xúc xắc bị nghiêng về mặt có 3 chấm không? Chúng ta có thể thực hiện một phép thử nhị thức để trả lời câu hỏi đó.

Trong Python, chúng ta có thể thực hiện kiểm tra nhị thức bằng cách sử dụng hàm `binom_test()` từ thư viện `scipy.stats`, với cú pháp sau:

`binom_test(x, n=None, p=0.5, alternative='two-sided')`

ở đây:

- **x**: số lần "thành công"
- **n**: tổng số thử nghiệm
- **p**: xác suất thành công của mỗi lần thử
- **alternative**: giả thuyết thay thế. Mặc định là 'hai mặt' nhưng chúng ta cũng có thể chỉ định 'lớn hơn' hoặc 'nhỏ hơn.'

Hàm này trả về giá trị p của phép thử.

```
from scipy.stats import binom_test
binom_test(x=6, n=24, p=1/6, alternative='greater')

0.1995295129479586
```

Bởi vì giá trị p này (0,1995) không nhỏ hơn 0,05, chúng ta không thể bác bỏ giả thuyết không. Tức không có đủ bằng chứng để nói rằng xúc xắc lệch về mặt 3 chấm.

Ví dụ 2: Tung một đồng xu 30 lần, có 19 lần mặt sấp xuất hiện. Thực hiện kiểm tra nhị thức để xác định xem đồng xu có thiên về mặt sấp hay không.

Giả thuyết không và giả thuyết thay thế cho thử nghiệm của chúng ta là:

$$H_0 : \pi \leq 1/2 \text{ (đồng xu không thiên về mặt sấp đầu)}$$

$$H_A : \pi > 1/2$$

```
binom_test(x=19, n=30, p=1/2, alternative='greater')
0.10024421103298661
```

Vì giá trị p này (0,10024) không nhỏ hơn 0,05, chúng ta không thể bác bỏ giả thuyết không. Tức không có đủ bằng chứng để nói rằng đồng tiền này thiên về mặt sấp.

Ví dụ 3: Một công ty với 80% sản phẩm đạt chất lượng. Họ thực hiện một hệ thống mới mà họ hy vọng sẽ cải thiện tỷ lệ chất lượng. Họ chọn ngẫu nhiên 50 vật dụng từ quá trình sản xuất gần đây và thấy rằng 47 vật dụng trong số đó có chất lượng. Thực hiện kiểm tra nhị thức để xác định xem hệ thống mới có dẫn đến chất lượng cao hơn hay không.

Giả thuyết không và giả thuyết thay thế cho thử nghiệm của chúng ta như sau:

$$H_0 : \pi \leq 0,80 \text{ (hệ thống mới không làm tăng hiệu quả)}$$

$$H_A : \pi > 0,80$$

```
binom_test(x=47, n=50, p=0.8, alternative='greater')
0.005656361012155314
```

Vì giá trị p này (0,00565) nhỏ hơn 0,05, chúng ta bác bỏ giả thuyết không. Tức có đầy đủ bằng chứng để nói rằng hệ thống mới giúp tăng chất lượng.

KIỂM ĐỊNH PHƯƠNG SAI

8.13. Kiểm tra Bartlett

Kiểm định Bartlett là một kiểm định thống kê được sử dụng để xác định xem liệu phương sai giữa một số nhóm có bằng nhau hay không.

Nhiều thử nghiệm thống kê (như ANOVA một chiều) giả định rằng các phương sai là bằng nhau giữa các mẫu. Kiểm tra của Bartlett được sử dụng để xác minh giả định đó.

Thử nghiệm này sử dụng giả thuyết không và giả thuyết thay thế sau:

H_0 : Phương sai giữa các nhóm bằng nhau.

H_A : Có ít nhất một nhóm có phương sai không bằng nhóm còn lại.

Thống kê kiểm định tuân theo phân phối Chi-Square với $k-1$ bậc tự do trong đó k là số nhóm.

Nếu giá trị p tương ứng của thống kê thử nghiệm nhỏ hơn một mức ý nghĩa nào đó (như $\alpha = 0,05$) thì chúng ta có thể bác bỏ giả thuyết không và kết luận rằng không phải tất cả các nhóm đều có cùng phương sai.

Ví dụ sau đây giải thích cách thực hiện kiểm tra của Bartlett.

Bước 1: Tạo dữ liệu

Để xác định xem ba kỹ thuật học khác nhau có dẫn đến điểm thi khác nhau hay không, một giáo sư chỉ định ngẫu nhiên 10 sinh viên sử dụng mỗi kỹ thuật (Kỹ thuật A, B hoặc C) trong một tuần và sau đó yêu cầu mỗi sinh viên tham gia một kỳ thi có độ khó như nhau.

Điểm thi của 30 sinh viên được hiển thị dưới đây:

```
#create data
A = [85, 86, 88, 75, 78, 94, 98, 79, 71, 80]
B = [91, 92, 93, 85, 87, 84, 82, 88, 95, 96]
C = [79, 78, 88, 94, 92, 85, 83, 85, 82, 81]
```

Bước 2: Thực hiện kiểm tra Bartlett's

Để thực hiện kiểm tra Bartlett, chúng ta có thể sử dụng hàm `scipy.stats.bartlett()`.

```
import scipy.stats as stats

#perform Bartlett's test
stats.bartlett(A, B, C)

BartlettResult(statistic=3.30243757, pvalue=0.191815983)
```

Thử nghiệm trả về các kết quả sau:

- Thống kê thử nghiệm $B : 3,3024$
- Giá trị $P : 0,1918$

Vì giá trị p không nhỏ hơn 0,05, giáo sư sẽ không bác bỏ giả thuyết không. Nói cách khác, cô ấy không có đủ bằng chứng để nói rằng ba nhóm có sự khác biệt khác nhau. Do đó, giáo sư có thể tiến hành thực hiện ANOVA một chiều.

8.14. Kiểm định F-test

Kiểm định **F-test** được sử dụng để kiểm tra phương sai của hai mẫu có tương đương nhau.

Giả thuyết không H_0 và giả thuyết thay thế (H_1) cho thử nghiệm như sau:

$H_0 : \sigma_1^2 = \sigma_2^2$ (các phương sai tổng thể bằng nhau)

$H_1 : \sigma_1^2 \neq \sigma_2^2$ (các phương sai tổng thể không bằng nhau)

Giả sử chúng ta có hai mẫu sau:

```
x = [18, 19, 22, 25, 27, 28, 41, 45, 51, 55]
y = [14, 15, 15, 17, 18, 22, 25, 25, 27, 34]
```

Chúng ta có thể sử dụng hàm sau để thực hiện kiểm định F nhằm xác định xem hai quần thể với các mẫu ở trên có phương sai tương đương nhau hay không:

```
import numpy as np

#define F-test function
def f_test(x, y):
    x = np.array(x)
    y = np.array(y)
    f = np.var(x, ddof=1)/np.var(y, ddof=1) #calculate F test statistic
    dfn = x.size-1                         #define degrees of freedom numerator
    dfd = y.size-1                         #define degrees of freedom denominator
    p = 1-scipy.stats.f.cdf(f, dfn, dfd) #find p-value of F test statistic
    return f, p

#perform F-test
f_test(x, y)

(4.38712, 0.019127)
```

Thống kê kiểm định F là **4,38712** và giá trị p tương ứng là **0,019127**. Vì giá trị p này nhỏ hơn 0,05, chúng ta sẽ bác bỏ giả thuyết không. Tức có đủ bằng chứng để nói rằng hai phương sai dân số *không* bằng nhau.

Trong đoạn mã trên chúng ta đã thực hiện

- Thống kê kiểm định F được tính là s_1^2 / s_2^2 . Theo mặc định, hàm **numpy.var** tính toán phương sai tổng thể. Để tính phương sai mẫu, chúng ta cần chỉ định **ddof = 1**.
- P – giá trị, tương ứng với 1 - cdf của phân phối F, có bậc tự do ở tử số là $n_1 - 1$ và bậc tự do ở mẫu số là $n_2 - 1$.
- Hàm này chỉ hoạt động khi phương sai mẫu đầu tiên lớn hơn phương sai mẫu thứ hai.

Khi nào sử dụng F-Test?

F-test thường được sử dụng để trả lời một trong những câu hỏi sau:

1. Có phải hai mẫu đến từ các quần thể có phương sai bằng nhau không?
2. Liệu một phương pháp điều trị hoặc quy trình mới có làm giảm sự thay đổi của một số phương pháp điều trị hoặc quy trình hiện tại không?

8.15. Giá trị tới hạn F

Khi thực hiện một kiểm định F, chúng ta sẽ nhận được một thống kê F kết quả. Để xác định xem kết quả của cuộc thử nghiệm F có ý nghĩa về mặt thống kê, ta có thể so sánh các số liệu thống kê F với **giá trị tới hạn F**. Nếu thống kê F lớn hơn giá trị tới hạn F, thì kết quả của phép thử có ý nghĩa thống kê.

Giá trị tới hạn F có thể được tìm thấy bằng cách sử dụng [bảng phân phối F](#) hoặc bằng cách sử dụng phần mềm thống kê.

Để tìm giá trị tới hạn F, chúng ta cần:

- Mức ý nghĩa (các lựa chọn phổ biến là 0,01, 0,05 và 0,10)
- Bậc tự do của tử số
- Bậc tự do của mẫu số

Sử dụng ba giá trị này, có thể xác định giá trị quan trọng F được so sánh với thống kê F.

Giá trị tới hạn F

Để tìm giá trị tới hạn F trong Python, có thể sử dụng hàm [scipy.stats.f.ppf\(\)](#), sử dụng cú pháp sau:

`scipy.stats.f.ppf (q, dfn, dfd)`

Ở đây:

- **q**: Mức ý nghĩa
- **dfn**: Bậc tự do của tử số
- **dfd**: Bậc tự do của mẫu số

Hàm này trả về giá trị tới hạn từ phân phối F dựa trên mức ý nghĩa, bậc tự do tử số và bậc tự do mẫu số được cung cấp.

Ví dụ, giả sử chúng ta muốn tìm giá trị tới hạn F cho mức ý nghĩa 0,05, bậc tự do tử số = 6 và bậc tự do mẫu số = 8.

```
import scipy.stats

#find F critical value
scipy.stats.f.ppf(q=1-.05, dfn=6, dfd=8)
```

Giá trị tới hạn F cho mức ý nghĩa 0,05, bậc tự do tử số = 6 và bậc tự do mẫu số = 8 là **3,5806**.

Do đó, nếu chúng ta đang thực hiện một kiểm định F thì chúng ta có thể so sánh thống kê kiểm tra F với **3.5806**. Nếu thống kê F lớn hơn 3.580 thì kết quả của phép thử có ý nghĩa thống kê.

Lưu ý rằng các giá trị nhỏ hơn của mức ý nghĩa sẽ dẫn đến các giá trị tới hạn F lớn hơn. Ví dụ, hãy xem xét giá trị tới hạn F với mức ý nghĩa là **0,01**, bậc tự do của tử số = 6 và bậc tự do ở mẫu số = 8.

```
scipy.stats.f.ppf(q=1-.01, dfn=6, dfd=8)
```

Giá trị tới hạn F với cùng bậc tự do chính xác cho tử số và mẫu số, nhưng với mức ý nghĩa là **0,005**:

```
scipy.stats.f.ppf(q=1-.005, dfn=6, dfd=8)
```

7.9512

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f.html>

8.16. Kiểm tra so sánh phương sai của 2 nhóm (Kiểm tra Levene)

Levene's Test được sử dụng để xác định xem hai hay nhiều nhóm có phương sai bằng nhau hay không. Nó thường được sử dụng vì nhiều thử nghiệm thống kê đưa ra giả định rằng các nhóm có phương sai bằng nhau và kiểm tra Levene cho phép chúng ta xác định xem giả định này có được thỏa mãn hay không.

Ví dụ:

Các nhà nghiên cứu muốn biết liệu ba loại phân bón khác nhau có dẫn đến mức độ phát triển khác nhau của thực vật hay không. Họ chọn ngẫu nhiên 30 cây khác nhau và chia chúng thành ba nhóm 10 cây, bón một loại phân bón khác nhau cho mỗi nhóm. Vào cuối một tháng, họ đo chiều cao của từng cây.

Sử dụng các bước sau để thực hiện Kiểm tra Levene bằng Python để xác định xem ba nhóm có phương sai bằng nhau hay không.

Bước 1: Nhập dữ liệu.

Đầu tiên, chúng ta sẽ tạo ba mảng để chứa các giá trị dữ liệu:

```
group1 = [7, 14, 14, 13, 12, 9, 6, 14, 12, 8]
group2 = [15, 17, 13, 15, 15, 13, 9, 12, 10, 8]
group3 = [6, 8, 8, 9, 5, 14, 13, 8, 10, 9]
```

Bước 2: Thực hiện kiểm tra Levene.

Tiếp theo, chúng ta sẽ thực hiện kiểm tra Levene bằng cách sử dụng hàm `levене()` từ thư viện SciPy, sử dụng cú pháp sau:

```
levене(sample1, sample2,..., center = 'median')
```

ở đây:

- **sample1, sample2, v.v.:** Tên của các mẫu.
- **center:** Phương pháp sử dụng cho bài kiểm tra của Levene. Giá trị mặc định là 'median', có các lựa chọn khác bao gồm 'Mean' và 'trimmed.'

Có ba biến thể khác nhau của bài kiểm tra Levene có thể sử dụng. Các cách sử dụng được đề xuất như sau:

- **'median':** được khuyến nghị cho các phân phối lệch.
- **'mean':** được khuyến nghị cho các phân phối đối xứng, có đuôi vừa phải.

- **'trimmed':** được khuyến nghị cho các bản phân phối có nhiều đuôi.

Đoạn mã sau minh họa cách thực hiện bài kiểm tra của Levene bằng cách sử dụng cả giá **trị trung bình** và giá **trị trung vị** làm trung tâm:

```
import scipy.stats as stats

#Levene's test centered at the median
stats.levene(group1, group2, group3, center='median')

(statistic=0.1798, pvalue=0.8364)

#Levene's test centered at the mean
stats.levene(group1, group2, group3, center='mean')

(statistic=0.5357, pvalue=0.5914)
```

Trong cả hai phương pháp, giá trị p không nhỏ hơn 0,05. Điều này có nghĩa là trong cả hai trường hợp, chúng ta sẽ không bác bỏ giả thuyết không. Tức không có đủ bằng chứng để nói rằng sự khác biệt về sự phát triển của cây giữa ba loại phân bón là khác nhau đáng kể.

Nói cách khác, ba nhóm có phương sai bằng nhau. Nếu chúng ta thực hiện một thử nghiệm thống kê nào đó (như [ANOVA một chiều](#)) giả sử mỗi nhóm có phương sai bằng nhau, thì giả định này sẽ được đáp ứng.

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.levene.html#scipy.stats.levene>

Tóm tắt chương 8

Chương này đã trình bày các vấn đề của bài toán kiểm định thống kê cơ bản, thông qua các ví dụ minh họa.

Để hiểu rõ bản chất của các bài toán, người đọc nên tìm hiểu các công thức tính toán của các bài toán kiểm định phi tham số (xem thêm trong các tài liệu đã dẫn trong phần danh mục tham khảo), để thực hiện cài đặt lại các hàm thay vì sẵn có của Python, nhằm hiểu rõ bản chất của quá trình kiểm định thống kê.

Câu hỏi ôn tập và bài tập chương 8

- Trình bày và cho được ví dụ thực tiễn về các bước cơ bản để thực hiện bài toán kiểm định phi tham số.

- Cài đặt thực hiện các ví dụ minh họa trong chương.
- Giải thích các câu lệnh cơ bản trong các ví dụ
- Vận dụng các ví dụ minh họa cho dữ liệu thực tiễn được lưu trong file *ngheSi.xlsx* tại địa chỉ :
<https://docs.google.com/spreadsheets/d/1q7RVfoTKEuLP0Ud0a50mJu7e1-YJJbUe/edit?usp=sharing&oid=106843557461060771783&rtpof=true&sd=true>
- Tìm hiểu các công thức tính toán của các bài toán kiểm định phi tham số, để thực hiện cài đặt lại các hàm thay vì sẵn có của Python, nhằm hiểu rõ bản chất của quá trình kiểm định thống kê.

CHƯƠNG 9. TƯƠNG QUAN VÀ HỒI QUY

Mục đích

Trình bày các vấn đề của bài toán tương quan và hồi quy

- *Tương quan, hệ số tương quan*
- *Tương quan thứ hạng*
- *Tương quan Kendall*
- *Tương quan bộ phận*
- *Ma trận tương quan và ma trận hiệp phương sai*
- *Hồi quy tuyến tính, hồi quy bậc hai và hồi quy đa thức*

(Một số mã nguồn trong chương được tham khảo từ [6], [7])

Yêu cầu

- *Nêu được ý nghĩa của bài toán tương quan và bài toán hồi quy*
- *Trình bày và cho được ví dụ thực tiễn về các bước cơ bản để thực hiện bài toán tương quan.*
- *Trình bày được phương pháp bình phương tối thiểu và cho được ví dụ thực tiễn về các bước cơ bản để thực hiện bài toán hồi quy.*
- *Cài đặt thực hiện các ví dụ minh họa trong chương.*
- *Giải thích các câu lệnh cơ bản trong các ví dụ*
- *Vận dụng các ví dụ minh họa cho dữ liệu thực tiễn*

9.1. Hệ số tương quan

Tương quan đề cập đến sự kết hợp giữa các giá trị quan sát của hai biến. Các biến có thể có một liên kết dương, có nghĩa là khi giá trị của một biến tăng thì giá trị của biến kia cũng vậy. Sự kết hợp cũng có thể âm, có nghĩa là khi giá trị của một biến tăng lên, thì giá trị của những biến khác sẽ giảm. Hơn nữa, sự liên kết có thể là trung tính, có nghĩa là các biến không được liên kết với nhau.

Hệ số tương quan là chỉ số thống kê đo lường mức độ tương quan của mỗi quan hệ giữa hai biến số.

Trong đó: Hệ số tương quan có giá trị từ -1.0 đến 1.0. Kết quả được tính ra lớn hơn 1.0 hoặc nhỏ hơn -1, có nghĩa là đã có lỗi trong phép đo tương quan.

- Hệ số tương quan có giá trị âm cho thấy hai biến có mối quan hệ nghịch biến hoặc tương quan âm (nghịch biến tuyệt đối khi giá trị bằng -1)

- Hệ số tương quan có giá trị dương cho thấy mối quan hệ đồng biến hoặc tương quan dương (đồng biến tuyệt đối khi giá trị bằng 1)
- Tương quan bằng 0 cho hai biến độc lập với nhau.

Hệ số tương quan Pearson: Cho hai biến số x và y từ mẫu n phần tử, hệ số tương quan Pearson được tính bằng công thức sau đây:

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}$$

Hệ số tương quan Pearson (r) có giá trị dao động trong khoảng liên tục từ -1 đến +1:

- $r = 0$: Hai biến không có tương quan tuyến tính
- $r = 1$; $r = -1$: Hai biến có mối tương quan tuyến tính tuyệt đối.
- $r < 0$: Hệ số tương quan âm. Nghĩa là giá trị biến x tăng thì giá trị biến y giảm và ngược lại, giá trị biến y tăng thì giá trị biến x giảm.
- $r > 0$: Hệ số tương quan dương. Nghĩa là giá trị biến x tăng thì giá trị biến y tăng và ngược lại, giá trị biến y tăng thì giá trị biến x cũng tăng.

Lưu ý: Hệ số tương quan pearson (r) chỉ có ý nghĩa khi và chỉ khi mức ý nghĩa quan sát (sig.) nhỏ hơn mức ý nghĩa $\alpha = 5\%$

- Nếu r nằm trong khoảng từ 0,50 đến ± 1 , thì nó được cho là tương quan mạnh.
- Nếu r nằm trong khoảng từ 0,30 đến $\pm 0,49$, thì nó được gọi là tương quan trung bình.
- Nếu r nằm dưới $\pm .29$, thì nó được gọi là một mối tương quan yếu.

(https://rpubs.com/tranquangquy_ictu/769561)

Cách tính toán tương quan trong Python

Để tính toán mối tương quan giữa hai biến trong Python, chúng ta có thể sử dụng hàm **corrcoef()** của Numpy.

```
import numpy as np

np.random.seed(100)

#create array of 50 random integers between 0 and 10
var1 = np.random.randint(0, 10, 50)
```

```
#create a positively correlated array with some random noise
var2 = var1 + np.random.normal(0, 10, 50)

#calculate the correlation between the two arrays
np.corrcoef(var1, var 2)

[[ 1.  0.335]
 [ 0.335 1.  ]]
```

Ta thấy hệ số tương quan giữa hai biến này là **0,335** , là tương quan thuận.

Theo mặc định, hàm này tạo ra một ma trận các hệ số tương quan. Nếu chúng ta chỉ muốn trả về hệ số tương quan giữa hai biến, chúng ta có thể sử dụng cú pháp sau:

np.corrcoef (var1, var 2) [0,1]

Để kiểm tra xem mối tương quan này có ý nghĩa thống kê hay không, chúng ta có thể tính giá trị p (p-value) liên quan đến hệ số tương quan Pearson bằng cách sử dụng hàm Scipy **pearsonr()**, hàm này trả về hệ số tương quan Pearson cùng với p - giá trị hai phía.

```
from scipy.stats.stats import pearsonr

pearsonr(var1, var2)

(0.335, 0.017398)
```

Hệ số tương quan là **0,335** và giá trị p hai phía là **0,17** . Vì giá trị p này nhỏ hơn 0,05, chúng ta có thể kết luận rằng có mối tương quan có ý nghĩa thống kê giữa hai biến.

Để tính toán mối tương quan giữa các biến trong DataFrame, ta có thể sử dụng hàm **.corr()** của pandas.

```
import pandas as pd

data = pd.DataFrame(np.random.randint(0, 10, size=(5, 3)),
columns=['A', 'B', 'C'])
data
```

	A	B	C
0	8	0	9
1	4	0	7
2	9	6	8
3	1	8	1
4	8	0	8

```
#calculate correlation coefficients for all pairwise combinations
data.corr()
```

	A	B	C
A	1.000000	-0.775567	-0.493769
B	-0.775567	1.000000	0.000000
C	-0.493769	0.000000	1.000000

Nếu chỉ cần quan tâm đến việc tính toán mối tương quan giữa hai biến cụ thể trong DataFrame, ta có thể chỉ định các biến:

```
data['A'].corr(data['B'])
```

9.2. Tương quan thứ hạng

Tương quan thứ hạng đề cập đến các phương pháp định lượng mối liên kết giữa các biến bằng cách sử dụng mối quan hệ thứ tự giữa các giá trị thay vì các giá trị cụ thể. Dữ liệu thứ tự là dữ liệu có các giá trị nhãn và có mối quan hệ thứ tự hoặc thứ hạng; ví dụ: "low", "medium" và "high".

Hệ số tương quan Spearman

Hệ số tương quan Spearman ρ được sử dụng khi hai biến x và y không tuân theo luật phân phối chuẩn, trái ngược với hệ số tương quan Pearson. Đôi khi đây còn được gọi là hệ số của phương pháp phân tích phi tham số. Hệ số được ước tính bằng cách biến đổi biến x , y thành biến có thứ bậc (rank), sau đó xem xét độ tương quan giữa hai dãy số có bậc này.

Hệ số còn được gọi là: Hệ số tương quan hạng Spearman (Spearman's Rank Correlation). Tương quan hạng Spearman được sử dụng thay thế tương quan Pearson để kiểm tra mối quan hệ giữa hai biến được xếp hạng hoặc một biến được xếp hạng và một biến đo lường không yêu cầu có phân phối chuẩn. Nói một cách dễ hiểu, hệ số tương quan Pearson là hệ số tương quan tuyến tính. Nếu kiểm định Pearson và kết luận x có tương quan với y , thì có thể cho rằng x và y có quan hệ tuyến tính với nhau. Ngược lại cũng chỉ có thể tạm kết luận rằng x và y không quan hệ tuyến tính. Vì x và y có thể có quan hệ với nhau theo một quan hệ nào đó ta chưa biết.

Tương quan hạng Spearman giữa x và y chính là việc xem xét tính đơn điệu của 2 biến này với nhau. Nếu hệ số tương quan dương thì kết luận là x tăng thì y cũng tăng. Nếu hệ số tương quan âm thì kết luận là x tăng thì y giảm.

```
# calculate the spearman's correlation between two variables
from numpy.random import rand
from numpy.random import seed
```

```
from scipy.stats import spearmanr
# seed random number generator
seed(1)
# prepare data
data1 = rand(1000) * 20
data2 = data1 + (rand(1000) * 10)
# calculate spearman's correlation
coef, p = spearmanr(data1, data2)
print('Spearman's correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) p=%.3f' % p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p)
```

Ở trên, hàm `coef, p = spearmanr(data1, data2)`

Trả về hệ số tương quan spearman giữa `data1` và `data2` (`coef`) và `p` – giá trị (`p`).

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html>

Ví dụ:

Giả sử chúng ta có dataframe sau đây chứa điểm thi môn toán và điểm thi môn khoa học của 10 học sinh trong một lớp. Trong ví dụ này thứ hạng là thứ hạng của điểm thi môn toán của học sinh so với thứ hạng của điểm thi môn khoa học của họ.

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'student': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'],
                   'math': [70, 78, 90, 87, 84, 86, 91, 74, 83, 85],
                   'science': [90, 94, 79, 86, 84, 83, 88, 92, 76, 75]})
```

Để tính toán mối tương quan xếp hạng Spearman giữa điểm toán và khoa học, chúng ta có thể sử dụng hàm `spearmanr()` từ `scipy.stats` :

```
from scipy.stats import spearmanr

#calculate Spearman Rank correlation and corresponding p-value
rho, p = spearmanr(df['math'], df['science'])

#print Spearman rank correlation and p-value
print(rho)
```

```
-0.41818181818181815
```

```
print(p)
```

```
0.22911284098281892
```

Từ kết quả đầu ra, chúng ta có thể thấy rằng tương quan thứ hạng Spearman là **-0,41818** và giá trị p tương ứng là **0,22911**.

Điều này cho thấy có mối tương quan nghịch giữa điểm thi khoa học và toán.

Tuy nhiên, do giá trị p của mối tương quan không nhỏ hơn 0,05 nên mối tương quan không có ý nghĩa thống kê.

Lưu ý rằng chúng ta cũng có thể sử dụng cú pháp sau để chỉ trích xuất hệ số tương quan hoặc giá trị p:

```
#extract Spearman Rank correlation coefficient  
spearmanr(df['math'], df['science'])[0]
```

```
-0.41818181818181815
```

```
#extract p-value of Spearman Rank correlation coefficient  
spearmanr(df['math'], df['science'])[1]
```

```
0.22911284098281892
```

9.3. Hệ số tương quan Kendall

Hệ số tương quan Kendall τ là một phương pháp phân tích phi tham số được ước tính bằng cách tìm các cặp số (x,y) “song hành” (concordant) với nhau. Một cặp (x,y) song hành ở đây được định nghĩa là có hiệu số (độ khác biệt) trên trục hoành có cùng dấu hiệu (dương hay âm) với hiệu trên trục tung. Nếu hai biến số x, y không có liên hệ với nhau số cặp song hành được xem là tương đương với số cặp không song hành.

Thông thường, với phương pháp này đòi hỏi máy tính phải tính toán với thời gian khá cao do có nhiều cặp số cần phải tính toán và kiểm định. Nếu tập dữ liệu khoảng dưới 5000 đối tượng thì việc tính toán này có thể thực hiện được.

Trong Python ta sử dụng hàm sau của thư viện scipy:

```
scipy.stats.kendalltau(x, y, initial_lexsort=True)
```

<https://docs.scipy.org/doc/scipy0.15.1/reference/generated/scipy.stats.kendalltau.html>

```
import scipy.stats as stats

x = [12, 2, 1, 12, 2]
y = [1, 4, 7, 1, 0]

tau, p_value = stats.kendalltau(y, y)

tau
-0.47140452079103173

p_value
0.2482130915752147
```

Trong ví dụ trên hàm:

`scipy.stats.kendalltau(x, y, initial_lexsort=True)`, trả về hai giá trị là tau và p- giá trị.

Tham số *initial_lexsort* chỉ ra sử dụng phương pháp sắp xếp nào cho mảng dữ liệu x, y.

Ở đây, giá trị tau được tính theo công thức:

$$\text{tau} = (P - Q) / \sqrt{((P + Q + T) * (P + Q + U))}$$

trong đó P là số cặp dữ liệu (x, y) tương hợp, Q là số cặp (x, y) không tương hợp, T là số liên kết có x và U là số liên kết có trong y. Nếu các tương hợp xảy ra cho cùng một cặp ở cả x và y, thì cặp này không được thêm vào T hoặc U.

9.4. Tương quan bộ phận

Trong thống kê, chúng ta thường sử dụng [hệ số tương quan Pearson](#) để đo lường mối quan hệ tuyến tính giữa hai biến. Tuy nhiên, đôi khi chúng ta quan tâm đến việc hiểu mối quan hệ giữa hai biến **với sự kiểm soát biến thứ ba**.

Hệ số tương quan bộ phận (partial correlation coefficient) là đại lượng phản ánh cường độ hay mức độ chặt chẽ của mối quan hệ giữa hai biến số trong khi ảnh hưởng của các biến số khác không thay đổi. Khi có ba hay nhiều biến số có quan hệ tương quan với nhau, mối quan hệ giữa hai biến số có thể bị ảnh hưởng bởi mối quan hệ của chúng với biến số thứ ba, cho nên hệ số tương quan thông thường giữa hai biến số có thể đưa ra những chỉ dẫn sai lầm về mối quan hệ chính xác của chúng. Hệ số tương quan bộ phận biểu thị mối liên hệ độc lập giữa hai biến số bằng cách loại trừ sự can thiệp của biến số thứ ba.

Ví dụ: Giả sử chúng ta muốn đo lường mối liên hệ giữa số giờ học của học sinh và điểm thi cuối kỳ mà họ nhận được, đồng thời kiểm soát điểm hiện tại của học sinh

trong lớp. Trong trường hợp này, chúng ta có thể sử dụng mối tương quan bộ phận để đo lường mối quan hệ giữa số giờ đã học và điểm thi cuối kỳ.

Giả sử chúng ta có DataFrame sau đây hiển thị điểm hiện tại, tổng số giờ đã học và điểm thi cuối kỳ cho 10 sinh viên:

```
import numpy as np
import pandas as pd

data = {'currentGrade': [82, 88, 75, 74, 93, 97, 83, 90, 90, 80],
        'hours': [4, 3, 6, 5, 4, 5, 8, 7, 4, 6],
        'examScore': [88, 85, 76, 70, 92, 94, 89, 85, 90, 93],
        }

df = pd.DataFrame(data, columns = ['currentGrade', 'hours', 'examScore'])
df
```

	currentGrade	hours	examScore
0	82	4	88
1	88	3	85
2	75	6	76
3	74	5	70
4	93	4	92
5	97	5	94
6	83	8	89
7	90	7	85
8	90	4	90
9	80	6	93

Để tính toán mối tương quan bộ phận giữa **giờ** và **ExamScore** với biến kiểm soát **currentGrade**, chúng ta có thể sử dụng hàm **part_corr()** từ [gói pingouin](#), sử dụng cú pháp sau:

partial_corr(data, x, y, covar)

ở đây:

- **data:** tên của dataframe
- **x, y:** tên của các cột trong khung dữ liệu
- **covar:** tên của cột hiệp biến trong khung dữ liệu (ví dụ: biến bạn đang kiểm soát)

```
#install and import pingouin package
pip install pingouin
import pingouin as pg

#find partial correlation between hours and exam score while
controlling for grade
```



```
pg.partial_corr(data=df, x='hours', y='examScore',
covar='currentGrade')
```

	n	r	CI95%	r2	adj_r2	p-val
BF10	power					
pearson	10	0.191	[-0.5, 0.73]	0.036	-	
	0.238	0.598	0.438	0.082		

Chúng ta thấy rằng mối tương quan bộ phận giữa số giờ đã học và điểm thi cuối kỳ là **0,191**, đây là một mối tương quan thuận và nhỏ. Khi số giờ học tăng lên, điểm thi cũng có xu hướng tăng, giả sử điểm hiện tại không đổi.

Để tính toán mối tương quan từng phần giữa nhiều biến cùng một lúc, chúng ta có thể sử dụng hàm **.pcorr()** :

```
#calculate all pairwise partial correlations, rounded to three
decimal places
df.pcorr().round(3)
```

	currentGrade	hours	examScore
currentGrade	1.000	-0.311	0.736
hours	-0.311	1.000	0.191
examScore	0.736	0.191	1.000

Trong dữ liệu đầu ra ở trên, ta thấy:

- Mối tương quan bộ phận giữa điểm hiện tại và số giờ được nghiên cứu là **-0,311**.
- Tương quan bộ phận giữa điểm hiện tại và điểm thi **0,736**.
- Mối tương quan bộ phận giữa số giờ học và điểm thi **0,191**.

9.5. Ma trận tương quan

Một cách để định lượng mối quan hệ giữa hai biến là sử dụng [hệ số tương quan Pearson](#), là một thước đo [mối quan hệ](#) tuyến tính giữa hai biến. Nó nhận giá trị từ -1 đến 1 trong đó:

- -1 chỉ ra mối tương quan tuyến tính hoàn toàn âm.
- 0 cho thấy không có tương quan tuyến tính.
- 1 chỉ ra mối tương quan tuyến tính hoàn toàn thuận.

Hệ số tương quan càng xa 0 thì mối quan hệ giữa hai biến càng mạnh.

Nhưng trong một số trường hợp, chúng ta muốn hiểu mối tương quan giữa nhiều hơn một cặp biến. Trong những trường hợp này, chúng ta có thể tạo một ma trận tương quan, là một ma trận vuông cho thấy các hệ số tương quan giữa một số kết hợp từng cặp của các biến.

Sử dụng các bước sau để tạo ma trận tương quan trong Python.

Bước 1: Tạo tập dữ liệu

```
import pandas as pd

data = {'assists': [4, 5, 5, 6, 7, 8, 8, 10],
        'rebounds': [12, 14, 13, 7, 8, 8, 9, 13],
        'points': [22, 24, 26, 26, 29, 32, 20, 14]}

df = pd.DataFrame(data, columns=['assists', 'rebounds', 'points'])
df
```

	assist	rebounds	points
0	4	12	22
1	5	14	24
2	5	13	26
3	6	7	26
4	7	8	29
5	8	8	32
6	8	9	20
7	10	13	14

Bước 2: Tạo ma trận tương quan

```
#create correlation matrix
df.corr()
```

	assists	rebounds	points
assists	1.000000	-0.244861	-0.329573
rebounds	-0.244861	1.000000	-0.522092
points	-0.329573	-0.522092	1.000000

```
#create same correlation matrix with coefficients rounded to 3 decimals
df.corr().round(3)
```

	assists	rebounds	points
assists	1.000	-0.245	-0.330
rebounds	-0.245	1.000	-0.522
points	-0.330	-0.522	1.000

Bước 3: Diễn giải ma trận tương quan

Các hệ số tương quan dọc theo đường chéo của bảng đều bằng 1 vì mỗi biến có tương quan hoàn toàn với chính nó.

Tất cả các hệ số tương quan khác chỉ ra mối tương quan giữa các tổ hợp biến số theo cặp khác nhau. Ví dụ:

- Hệ số tương quan giữa **assists** và **rebounds** là **-0,245** .
- Hệ số tương quan giữa **assists** và **points** là **-0.330** .
- Hệ số tương quan giữa **rebounds** và **points** là **-0,522** .

Bước 4: Minh hoạ ma trận tương quan qua hình ảnh

Có thể hình dung ma trận tương quan bằng cách sử dụng các **tùy chọn tạo kiểu** (styling options) của thư viện pandas:

```
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	assists	rebounds	points
assists	1	-0.244861	-0.329573
rebounds	-0.244861	1	-0.522092
points	-0.329573	-0.522092	1

Hình 9. 1. Biểu đồ heatmap, minh hoạ mối tương quan giữa hai biến

Chúng ta có thể thay đổi đối số của **cmap** để tạo ra một ma trận tương quan với các màu khác nhau.

```
corr = df.corr()
corr.style.background_gradient(cmap='RdYlGn')
```

	assists	rebounds	points
assists	1	-0.244861	-0.329573
rebounds	-0.244861	1	-0.522092
points	-0.329573	-0.522092	1

```
corr = df.corr()
corr.style.background_gradient(cmap='bwr')
```

	assists	rebounds	points
assists	1	-0.244861	-0.329573
rebounds	-0.244861	1	-0.522092
points	-0.329573	-0.522092	1

```
corr = df.corr()
corr.style.background_gradient(cmap='PuOr')
```

	assists	rebounds	points
assists	1	-0.244861	-0.329573
rebounds	-0.244861	1	-0.522092
points	-0.329573	-0.522092	1

<https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html>

9.6. Ma trận hiệp phương sai (Covariance Matrix)

Hiệp phương sai là thước đo về những thay đổi của một biến kết hợp với những thay đổi cùng biến thứ hai. Cụ thể, nó là thước đo mức độ mà hai biến được liên kết tuyến tính.

Một **ma trận hiệp phương sai** là một ma trận vuông mô tả hiệp phương sai giữa nhiều biến số khác nhau. Đây là một cách hữu ích để hiểu các biến khác nhau có liên quan như thế nào trong một tập dữ liệu.

Ví dụ sau đây cho thấy cách tạo ma trận hiệp phương sai trong Python.

Bước 1: Tạo tập dữ liệu

Giả sử, chúng ta có tập dữ liệu chứa điểm kiểm tra của 10 học sinh khác nhau cho ba môn học: toán, khoa học và lịch sử, như sau:

```
import numpy as np

math = [84, 82, 81, 89, 73, 94, 92, 70, 88, 95]
science = [85, 82, 72, 77, 75, 89, 95, 84, 77, 94]
history = [97, 94, 93, 95, 88, 82, 78, 84, 69, 78]

data = np.array([math, science, history])
```

Bước 2: Tạo ma trận hiệp phương sai

Tiếp theo, chúng ta sẽ tạo ma trận hiệp phương sai cho tập dữ liệu này bằng cách sử dụng hàm `cov()` của thư viện `numpy`, với chỉ định tham số `bias = True` để chúng ta có thể tính toán ma trận hiệp phương sai tổng thể.

```
np.cov(data, bias=True)

array([[ 64.96,   33.2 , -24.44],
       [ 33.2 ,   56.4 , -24.1 ],
       [-24.44, -24.1 ,   75.56]])
```

Bước 3: Giải thích kết quả về ma trận hiệp phương sai tính được

Các giá trị nằm trên đường chéo của ma trận là phương sai của mỗi môn học. Ví dụ:

- Phương sai của điểm toán là 64,96
- Phương sai của điểm khoa học là 56,4
- Phương sai của điểm lịch sử là 75,56

Các giá trị khác trong ma trận đại diện cho hiệp phương sai giữa các đối tượng khác nhau. Ví dụ:

- Hiệp phương sai giữa điểm toán và khoa học là 33,2
- Hiệp phương sai giữa điểm toán và lịch sử là -24,44
- Hiệp phương sai giữa điểm khoa học và lịch sử là -24,1

Một **phần tử dương** trong ma trận hiệp phương sai chỉ ra rằng hai biến số có xu hướng tăng hoặc giảm cùng nhau. Ví dụ, toán học và khoa học có hiệp phương sai dương (33,2), điều này cho thấy rằng học sinh đạt điểm cao trong môn toán cũng có xu hướng đạt điểm cao trong môn khoa học. Ngược lại, học sinh đạt điểm môn toán thấp cũng có xu hướng đạt điểm môn khoa học thấp.

Một **số âm** trong ma trận hiệp phương sai chỉ ra rằng khi một biến tăng, biến thứ hai có xu hướng giảm. Ví dụ, toán học và lịch sử có hiệp phương sai âm (-24,44), điều này cho thấy rằng học sinh đạt điểm cao về môn toán có xu hướng đạt điểm thấp về môn lịch sử. Ngược lại, học sinh đạt điểm môn toán thấp có xu hướng đạt điểm môn lịch sử cao.

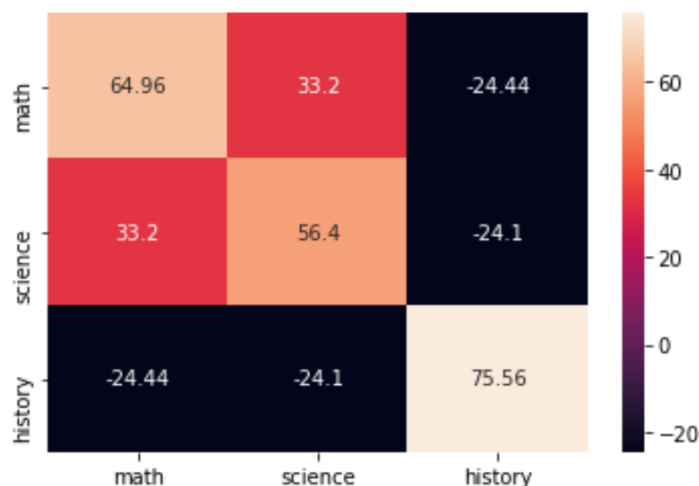
Bước 4: Minh họa ma trận hiệp phương sai qua hình ảnh

Có thể hình dung ma trận hiệp phương sai qua hình ảnh bằng cách sử dụng hàm `Heatmap()` từ thư viện `seaborn`:

```
import seaborn as sns
import matplotlib.pyplot as plt

cov = np.cov(data, bias=True)
labs = ['math', 'science', 'history']

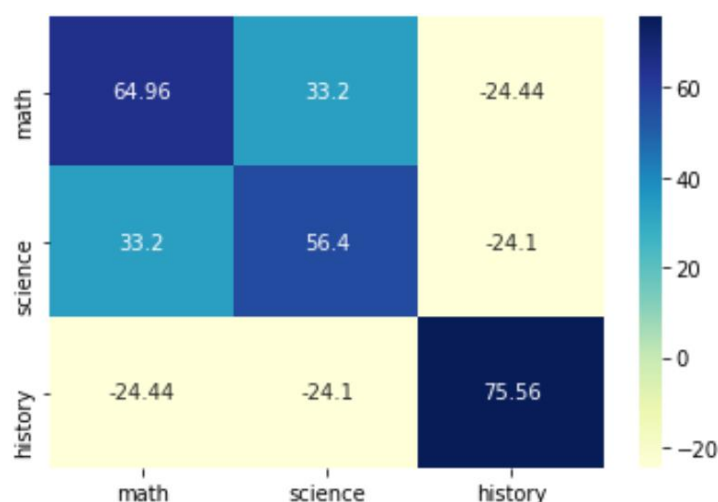
sns.heatmap(cov, annot=True, fmt='g', xticklabels=labs, yticklabels=labs)
plt.show()
```



Hình 9. 2. Biểu đồ headmap, minh họa ma trận hiệp phương sai

Bạn cũng có thể thay đổi bản đồ màu bằng cách thay đổi đối số **cmap** :

```
sns.heatmap(cov, annot=True, fmt='g', xticklabels=labs, yticklabels=labs,
cmap='YlGnBu')
plt.show()
```



<https://seaborn.pydata.org/generated/seaborn.heatmap.html>

Minh họa độ tương quan

Python code to compute the correlation coefficient

N.T.D

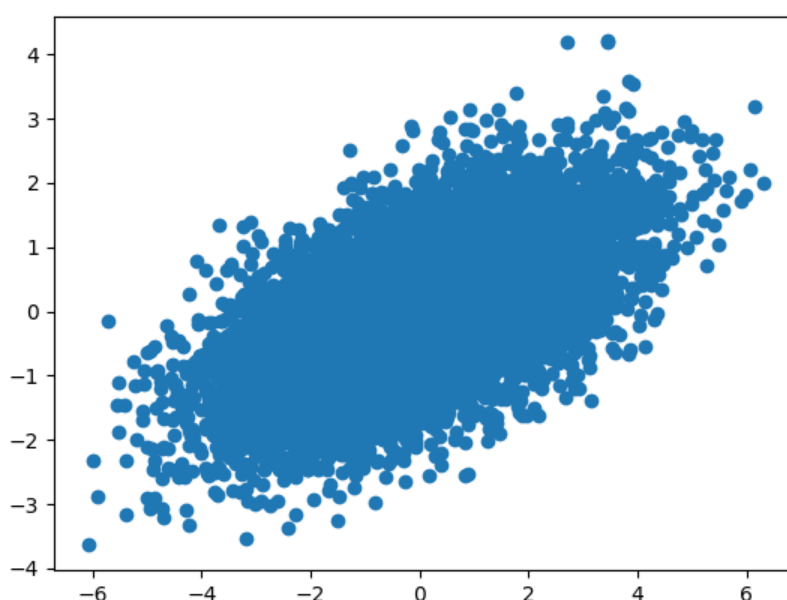
```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

x = stats.multivariate_normal.rvs([0,0], [[3,1],[1,1]], 10000)
plt.figure(); plt.scatter(x[:,0],x[:,1])

rho,_ = stats.pearsonr(x[:,0],x[:,1])
print(rho)

plt.show()

#rho = 0.5799884209952031
```



Hình 9. 3. Biểu đồ scatter minh hoạ mối tương quan giữa hai biến

9.7. Hồi quy tuyến tính

Hồi quy tuyến tính là một phương pháp được sử dụng để thấy được mối quan hệ giữa một hoặc nhiều biến dự báo và một biến phản hồi.

Ví dụ:

Giả sử chúng ta muốn biết liệu số giờ học và số giờ chuẩn bị cho kỳ thi có ảnh hưởng đến điểm số mà một học sinh nhận được trong một kỳ thi nhất định hay không.

Các bước thực hiện với Python để tiến hành thực hiện phép hồi quy tuyến tính, nhằm đánh giá dự báo trên.

Bước 1: Nhập dữ liệu

Đầu tiên, chúng ta sẽ tạo một DataFrame để lưu dữ liệu:

```
import pandas as pd

#create data
df = pd.DataFrame({'hours': [1, 2, 2, 4, 2, 1, 5, 4, 2, 4, 4, 3, 6, 5, 3, 4,
6, 2, 1, 2],
                    'exams': [1, 3, 3, 5, 2, 2, 1, 1, 0, 3, 4, 3, 2, 4, 4, 4,
5, 1, 0, 1],
                    'score': [76, 78, 85, 88, 72, 69, 94, 94, 88, 92, 90, 75,
96, 90, 82, 85, 99, 83, 62, 76]})

#view data
df
```

	hours	exams	score
0	1	1	76
1	2	3	78
2	2	3	85
3	4	5	88
4	2	2	72
5	1	2	69
6	5	1	94
7	4	1	94
8	2	0	88
9	4	3	92
10	4	4	90
11	3	3	75
12	6	2	96
13	5	4	90
14	3	4	82
15	4	4	85
16	6	5	99
17	2	1	83
18	1	0	62
19	2	1	76

Bước 2: Thực hiện hồi quy tuyến tính

Tiếp theo, chúng ta sẽ sử dụng hàm [OLS\(\)](#) từ thư viện statsmodels để thực hiện phép hồi quy dựa trên phương pháp bình phương tối thiểu, sử dụng biến “giờ” và “kỳ thi” làm biến dự đoán và “điểm” làm biến phản hồi:

```
import statsmodels.api as sm

#define response variable
y = df['score']

#define predictor variables
x = df[['hours', 'exams']]

#add constant to predictor variables
x = sm.add_constant(x)
```



```
#fit linear regression model
model = sm.OLS(y, x).fit()

#view model summary
print(model.summary())
```

```

=====
OLS Regression Results
=====
=
Dep. Variable:          score    R-squared:
0.734
Model:                  OLS      Adj. R-squared:
0.703
Method:                 Least Squares    F-statistic:
23.46
Date:                   Fri, 24 Jul 2020    Prob (F-statistic):      1.29e-05
Time:                   13:20:31          Log-Likelihood:         -
60.354
No. Observations:       20    AIC:
126.7
Df Residuals:           17    BIC:
129.7
Df Model:                2
Covariance Type:        nonrobust
=====
=

```

	coef	std err	t	P> t	[0.025	0.975]
const	67.6735	2.816	24.033	0.000	61.733	73.614
hours	5.5557	0.899	6.179	0.000	3.659	7.453
exams	-0.6017	0.914	-0.658	0.519	-2.531	1.327

```

=====
=
Omnibus:                0.341    Durbin-Watson:
1.506
Prob(Omnibus):           0.843    Jarque-Bera (JB):
0.196
Skew:                    -0.216    Prob(JB):
0.907
Kurtosis:                2.782    Cond. No.
10.8
=====
=

```

Bước 3: Diễn giải kết quả

Dưới đây là cách diễn giải các con số cơ bản trong đầu ra:

R - Bình phương: 0,734 . Hệ số này được gọi là hệ số xác định. Đây là tỷ lệ của phương sai của biến phản hồi có thể được giải thích bằng các biến dự báo. Với ví dụ cụ thể này, thì 73,4% sự thay đổi trong điểm thi có thể được giải thích bằng số giờ học và số giờ chuẩn bị thi đã được thực hiện.

Thông kê F: 23,46 . Đây là thông kê F tổng thể cho mô hình hồi quy.

Xác suất ý nghĩa (thông kê F): 1.29e-05. Đây là xác suất ý nghĩa (giá trị p) liên quan đến thông kê F tổng thể. Giá trị này cho chúng ta biết liệu tổng thể mô hình hồi quy có ý nghĩa thống kê hay không. Nói cách khác, nó cho chúng ta biết liệu hai biến dự báo kết hợp có mối liên hệ với biến phản hồi có ý nghĩa thống kê hay không. Trong trường hợp này, giá trị p nhỏ hơn 0,05, điều này cho thấy rằng các biến dự báo "số giờ đã học" và "các bài kiểm tra chuẩn bị đã thực hiện" có mối liên hệ có ý nghĩa thống kê với điểm thi.

coef: Các hệ số cho mỗi biến dự báo, các hệ số này cho chúng ta biết sự thay đổi giá trị kỳ vọng của biến phản hồi, với giả sử biến dự báo khác không đổi. Ví dụ: đối với mỗi giờ học thêm, điểm thi trung bình dự kiến sẽ tăng **5,56** , giả sử rằng các bài kiểm tra để chuẩn bị đã được thực hiện là không đổi.

Cũng có thể hiểu là: Nếu học sinh A và học sinh B đều chuẩn bị thi như nhau, nhưng học sinh A học thêm một giờ, thì học sinh A dự kiến sẽ kiếm được điểm cao hơn học sinh B. **5,56** điểm.

Ở đây chúng ta có điểm dự thi là **67,67** với một sinh viên không học giờ nào và không có kỳ thi chuẩn bị nào.

P> | t |. Các giá trị p riêng lẻ cho chúng ta biết liệu mỗi biến dự báo có ý nghĩa thống kê hay không. Chúng ta có thể thấy rằng "hour" có xác suất ý nghĩa ($p = 0,00$) trong khi "exam" ($p = 0,52$) không có ý nghĩa thống kê với $\alpha = 0,05$. Vì "exam" không có ý nghĩa thống kê, chúng ta có thể quyết định xóa nó khỏi mô hình.

Phương trình hồi quy ước lượng: Chúng ta có thể sử dụng các hệ số từ đầu ra của mô hình để tạo ra phương trình hồi quy ước tính sau:

$$\text{exam score} = 67.67 + 5.56 * (\text{hours}) - 0.60 * (\text{prep exams})$$

Chúng ta có thể sử dụng phương trình hồi quy ước tính này để tính điểm kỳ thi dự kiến cho một học sinh, dựa trên số giờ học và số kỳ thi chuẩn bị mà học sinh đã thực hiện. Ví dụ, một sinh viên học trong ba giờ và thực hiện một kỳ thi chuẩn bị sẽ nhận được số điểm là **83,75** :

Với lưu ý vì các bài kiểm tra chuẩn bị (exam) đã được thực hiện không có ý nghĩa thống kê ($p = 0,52$), chúng ta có thể quyết định loại bỏ vì không thêm bất kỳ cải tiến nào cho mô hình tổng thể. Trong trường hợp này, chúng ta có thể thực hiện hồi quy tuyến tính đơn giản chỉ sử dụng số giờ mà học sinh đã học để làm biến dự báo.

Bước 4: Kiểm tra các giả định của mô hình

Khi thực hiện hồi quy tuyến tính, có một số giả định ta cần kiểm tra để đảm bảo rằng kết quả của mô hình hồi quy là đáng tin cậy. Các giả định này bao gồm:

Giả định 1: Tồn tại mối quan hệ tuyến tính giữa các biến dự báo và biến phản hồi.

- Kiểm tra giả định này bằng cách tạo một đồ thị dư thừa (residual plot) để hiển thị các giá trị phù hợp với các giá trị dư thừa cho mô hình hồi quy.

Giả định 2: Tính độc lập của phần dư.

- Kiểm tra giả định này bằng cách thực hiện phép kiểm tra Durbin-Watson .

Giả định 3: Tính co giãn đồng nhất của các phần dư.

- Kiểm tra giả định này bằng cách thực hiện phép kiểm tra Breusch-Pagan .

Giả định 4: Phần dư có phân phối chuẩn.

- Kiểm tra giả định này một cách trực quan bằng cách sử dụng đồ thị Q-Q.
- Cũng có thể kiểm tra giả định này bằng các bài kiểm tra như kiểm tra Jarque-Bera hoặc kiểm tra Anderson-Darling .

Giả định 5: Xác minh rằng các biến dự báo không tồn tại đa cộng tuyến.

- Kiểm tra giả định này bằng cách tính toán giá trị hệ số lạm phát phương sai (variance inflation factor - VIF) của mỗi biến dự báo.

Nếu các giả định này được đáp ứng, chúng ta có thể tin tưởng rằng kết quả của mô hình hồi quy tuyến tính của chúng ta là đáng tin cậy.

9.8. Phần dư trong hồi quy tuyến tính

Hồi quy tuyến tính đơn giản là một phương pháp thống kê giúp chúng ta hiểu mối quan hệ giữa hai biến x và y .

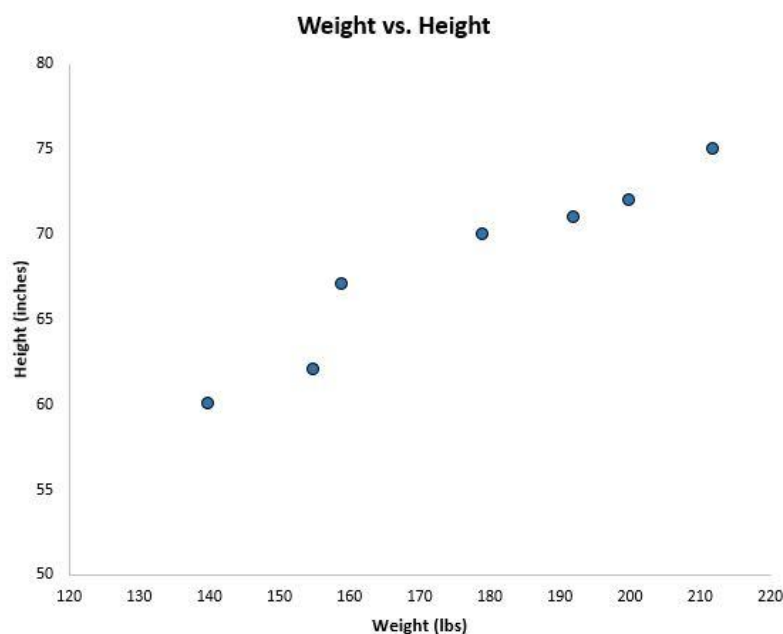
Một biến x được gọi là biến dự đoán. Biến còn lại, y , được gọi là **biến phản hồi**.

Ví dụ: giả sử chúng ta có tập dữ liệu sau về cân nặng và chiều cao của bảy cá nhân:

Weight (lbs)	Height (inches)
140	60
155	62
159	67
179	70
192	71
200	72
212	75

Đặt *cân nặng* là biến dự đoán và đặt *chiều cao* là biến phản hồi.

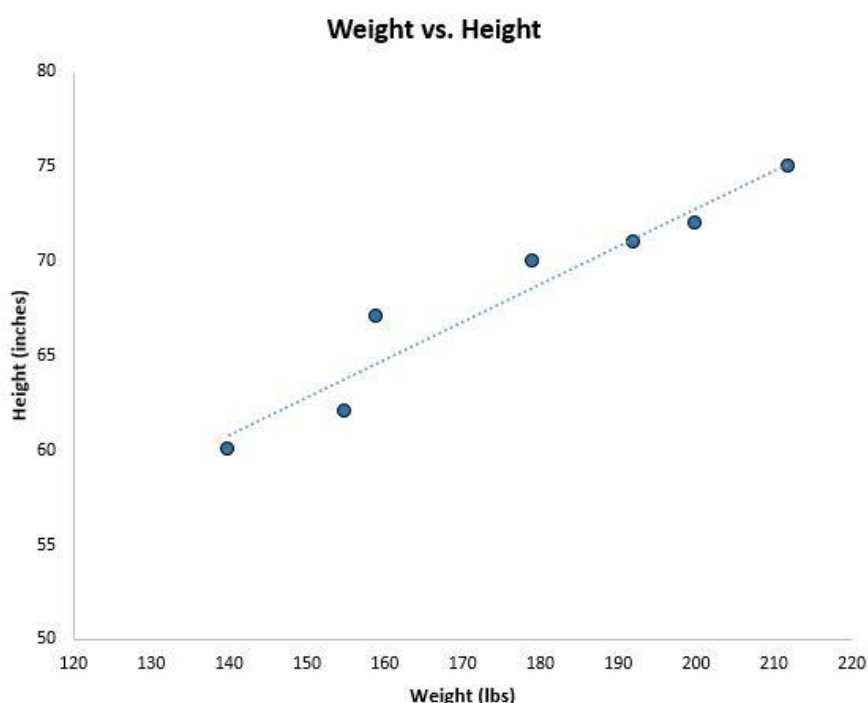
Vẽ biểu đồ hai biến này bằng biểu đồ phân tán, với cân nặng trên trục x và chiều cao trên trục y :



Hình 9. 4. Biểu đồ scatter minh hoạ mối tương quan giữa hai biến có tương quan tựa tuyến tính

Từ biểu đồ phân tán, chúng ta có thể thấy rằng khi cân nặng tăng, chiều cao cũng có xu hướng tăng, nhưng để thực sự *định lượng* mối quan hệ giữa cân nặng và chiều cao này, chúng ta cần sử dụng hồi quy tuyến tính.

Với hồi quy tuyến tính, chúng ta vẽ được đường thẳng “phù hợp” nhất với tập dữ liệu trên.



Hình 9. 5. Minh hoạ đường thẳng hồi quy tuyến tính (“đường thẳng phù hợp nhất”)

Công thức cho “đường thẳng phù hợp nhất” này được viết ở dạng:

$$\hat{y} = b_0 + b_1 x$$

trong đó \hat{y} là giá trị dự đoán của biến phản hồi, b_0 là ngưỡng y , b_1 là hệ số hồi quy và x là giá trị của biến dự đoán.

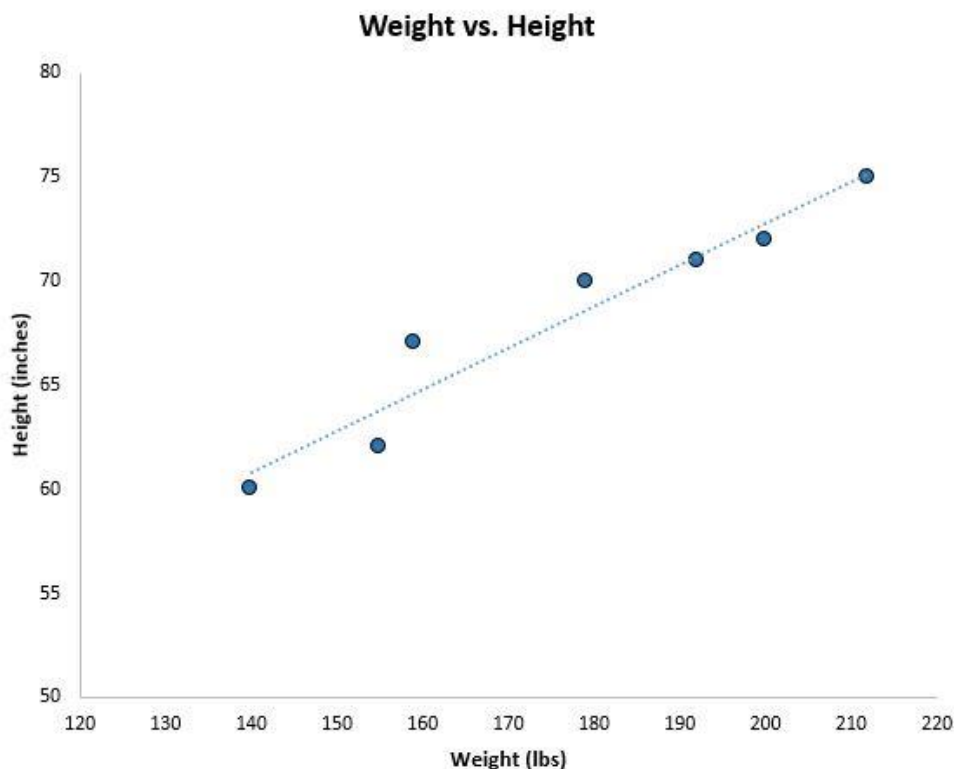
Trong ví dụ này, “đường thẳng phù hợp nhất” là:

$$\text{chiều cao} = 32,783 + 0,2001 * (\text{cân nặng})$$

(tham khảo phương pháp bình phương bé nhất)

Phần dư

Đề ý rằng các điểm dữ liệu trong biểu đồ phân tán của chúng ta không phải lúc nào cũng nằm chính xác trên “đường thẳng phù hợp nhất”:



Hình 9. 6. Minh họa phần dư

Sự khác biệt giữa điểm dữ liệu và đường được vẽ gọi là phần **dư**. Đối với mỗi điểm dữ liệu, chúng ta có thể tính toán phần dư của điểm đó bằng cách lấy chênh lệch giữa giá trị thực tế và giá trị dự đoán từ “đường thẳng phù hợp nhất”.

Ví dụ:

Xem xét ví dụ về cân nặng và chiều cao của bảy cá nhân trong tập dữ liệu của chúng ta:

Weight (lbs)	Height (inches)
140	60
155	62
159	67
179	70
192	71
200	72
212	75

Phần tử đầu tiên có trọng lượng **140 lbs.** và chiều cao **60** inch.

Để tìm ra chiều cao dự đoán cho phần tử này, chúng ta có thể đưa trọng lượng của phần tử này vào phương trình đường hồi quy:

$$\text{chiều cao} = 32,783 + 0,2001 * (\text{cân nặng})$$

Như vậy, chiều cao dự đoán của cá nhân này là:

$$\text{chiều cao} = 32,783 + 0,2001 * (140)$$

$$\text{chiều cao} = 60,797 \text{ inch}$$

Do đó, phần dư của điểm dữ liệu này là $60 - 60,797 = \mathbf{-0,797}$.

Tiếp theo hãy tính phần dư cho cá thể thứ hai trong tập dữ liệu:

Phần tử thứ hai có khối lượng **155 lbs.** và chiều cao **62** inch.

Để tìm ra chiều cao dự đoán cho cá nhân này, chúng ta có thể đưa trọng lượng của họ vào phương trình đường hồi quy:

$$\text{chiều cao} = 32,783 + 0,2001 * (\text{cân nặng})$$

Như vậy, chiều cao dự đoán của cá nhân này là:

$$\text{chiều cao} = 32,783 + 0,2001 * (155)$$

$$\text{chiều cao} = 63,7985 \text{ inch}$$

Do đó, phần dư cho điểm dữ liệu này là $62 - 63,7985 = \mathbf{-1,7985}$.

Sử dụng phương pháp tương tự đối với hai phần tử trên , chúng ta có thể tính toán phần dư cho mọi điểm dữ liệu:

Weight (lbs)	Height (inches)	Predicted Height	Residual
140	60	60.797	-0.797
155	62	63.7985	-1.7985
159	67	64.5989	2.4011
179	70	68.6009	1.3991
192	71	71.2022	-0.2022
200	72	72.803	-0.803
212	75	75.2042	-0.2042

Chú ý rằng, nếu **chúng ta cộng tất cả các phần dư, kết quả sẽ bằng không**. Điều này có được do chúng ta đã viết phương trình hồi quy tuyến tính bằng bình phương tối thiểu.

Minh hoạ cho hồi quy tuyến tính [5]

Python code to fit data points using a straight line

```
import numpy as np
import matplotlib.pyplot as plt

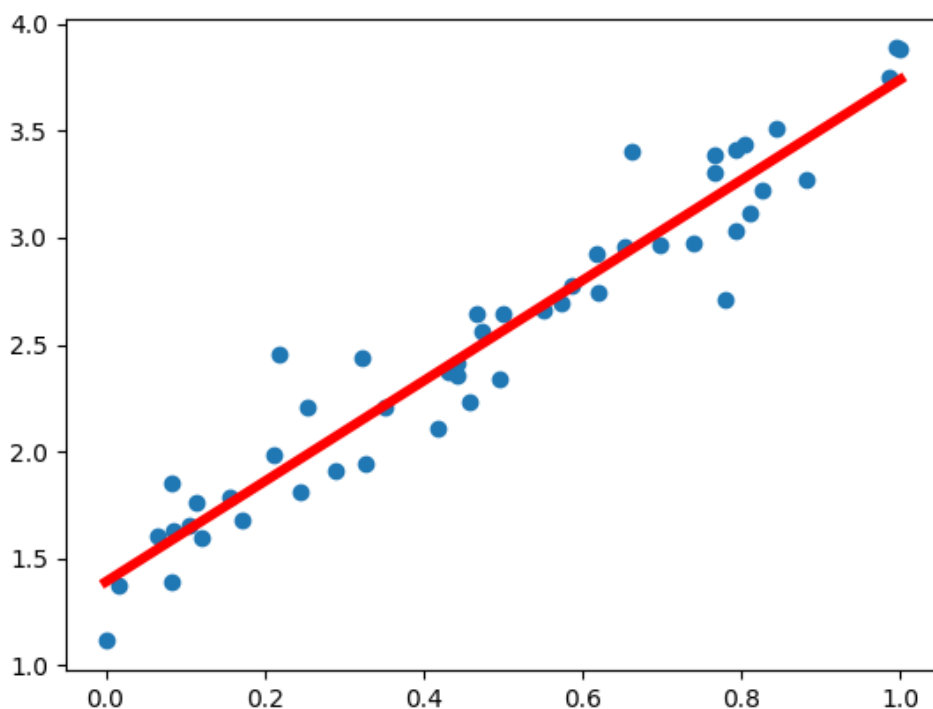
N = 50

x = np.random.rand(N)

a = 2.5 # true parameter
b = 1.3 # true parameter

y = a*x + b + 0.2*np.random.randn(N) # Synthesize training data
X = np.column_stack((x, np.ones(N))) # construct the X matrix
theta = np.linalg.lstsq(X, y, rcond=None)[0] # solve y = X theta
t = np.linspace(0,1,200) # interpolate and plot
yhat = theta[0]*t + theta[1]

plt.plot(x,y,'o')
plt.plot(t,yhat,'r',linewidth=4)
plt.show()
```

9.9. Hồi quy bậc hai

Hồi quy bậc hai là một loại hồi quy mà chúng ta có thể sử dụng để định lượng mối quan hệ giữa biến dự đoán và biến phản hồi khi các mối quan hệ thực sự là bậc hai. Nếu vẽ biểu đồ phân tán (scatter) sẽ có dạng đồ thị hàm bậc 2. Khi biến dự báo tăng thì biến phản hồi cũng có xu hướng tăng, nhưng sau một thời điểm nhất định, biến phản hồi bắt đầu giảm khi biến dự báo tiếp tục tăng.

Dưới đây là minh họa phương pháp hồi quy bậc 2 với Python

Python code to fit data using a quadratic equation [5]

```
import numpy as np

import matplotlib.pyplot as plt

N = 50

x = np.random.rand(N)

a = -2.5

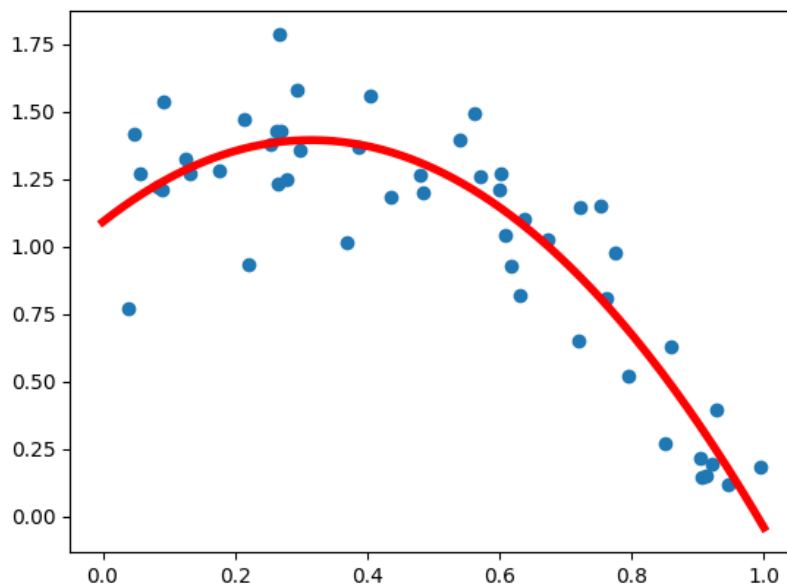
b = 1.3

c = 1.2

y = a*x**2 + b*x + c + 0.2*np.random.randn(N)

X = np.column_stack((np.ones(N), x, x**2))
```

```
theta = np.linalg.lstsq(X, y, rcond=None)[0]
t = np.linspace(0,1,200)
yhat = theta[0] + theta[1]*t + theta[2]*t**2
plt.plot(x,y,'o')
plt.plot(t,yhat,'r',linewidth=4)
plt.show()
```



Hình 9. 7. Minh hoạ hồi quy bậc hai

9.10. Hồi quy đa thức

Phân tích hồi quy được sử dụng để định lượng mối quan hệ giữa một hoặc nhiều biến dự báo và một biến phản hồi. Phân tích hồi quy phổ biến nhất là [hồi quy tuyến tính đơn giản](#), được sử dụng khi biến dự báo và biến phản hồi có mối quan hệ tuyến tính.

Tuy nhiên, đôi khi mối quan hệ giữa biến dự báo và biến phản hồi là phi tuyến tính. Trong những trường hợp này, nên sử dụng **hồi quy đa thức**.

Dưới đây là minh hoạ phương pháp đa thức, với đa thức Lagrăng với Python

#Python code to fit data using Legendre polynomials

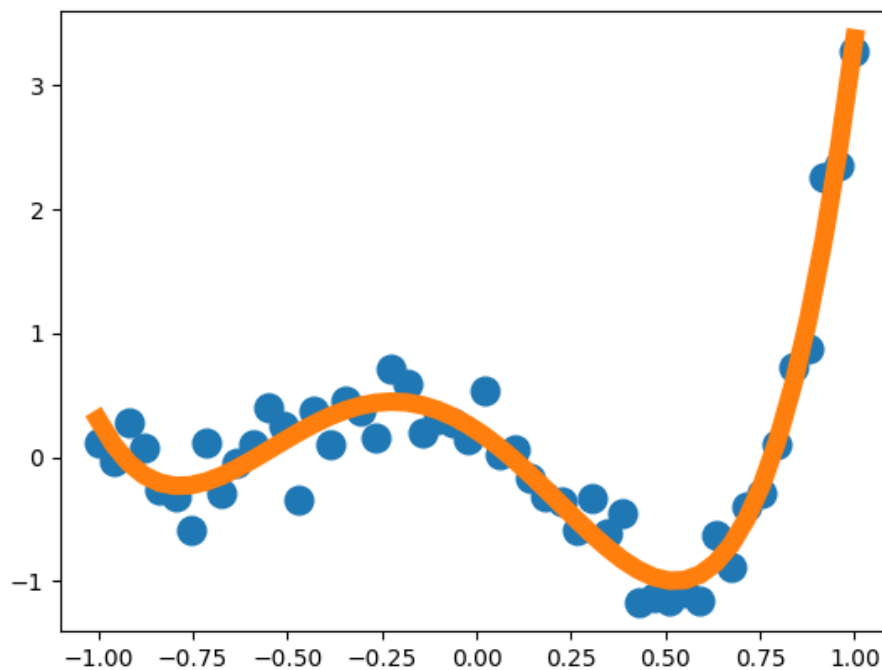
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import eval_legendre
```

```

N = 50
x = np.linspace(-1,1,N)
a = np.array([-0.001, 0.01, 0.55, 1.5, 1.2])
y = a[0]*eval_legendre(0,x) + a[1]*eval_legendre(1,x) + \
a[2]*eval_legendre(2,x) + a[3]*eval_legendre(3,x) + \
a[4]*eval_legendre(4,x) + 0.2*np.random.randn(N)
X = np.column_stack((eval_legendre(0,x), eval_legendre(1,x), \
eval_legendre(2,x), eval_legendre(3,x), \
eval_legendre(4,x)))
theta = np.linalg.lstsq(X, y, rcond=None)[0]
t = np.linspace(-1, 1, 50);

yhat = theta[0]*eval_legendre(0,t) + \
theta[1]*eval_legendre(1,t) + \
theta[2]*eval_legendre(2,t) + theta[3]*eval_legendre(3,t) + \
theta[4]*eval_legendre(4,t)
plt.plot(x,y,'o',markersize=12)
plt.plot(t,yhat, linewidth=8)
plt.show()

```



Hình 9. 8. Minh họa hồi quy với đường cong Lagrange

Tóm tắt chương 9

Chương này đã trình bày các vấn đề của bài toán tương quan và hồi quy thông qua các ví dụ minh họa.

- *Tương quan, hệ số tương quan*
- *Tương quan thứ hạng*
- *Tương quan Kendall*
- *Tương quan bộ phận*
- *Ma trận tương quan và ma trận hiệp phương sai*
- *Hồi quy tuyến tính, hồi quy bậc hai và hồi quy đa thức*

Để hiểu rõ bản chất của các bài toán, người đọc nên tìm hiểu các công thức tính toán của các bài toán tương quan và hồi quy (xem thêm trong các tài liệu đã dẫn trong phần danh mục tham khảo), để thực hiện cài đặt lại các hàm thay vì sẵn có của Python, nhằm hiểu rõ bản chất của quá trình thống kê.

Câu hỏi ôn tập và bài tập chương 9

- *Trình bày ý nghĩa của bài toán tương quan và bài toán hồi quy. Cho ví dụ thực tiễn minh họa*
- *Trình bày và cho được ví dụ thực tiễn về các bước cơ bản để thực hiện bài toán tương quan.*
- *Trình bày được phương pháp bình phương tối thiểu và cho được ví dụ thực tiễn về các bước cơ bản để thực hiện bài toán hồi quy.*
- *Cài đặt thực hiện các ví dụ minh họa trong chương.*
- *Giải thích các câu lệnh cơ bản trong các ví dụ*
- *Vận dụng các ví dụ minh họa cho dữ liệu thực tiễn*

Lời kết: Để hiểu rõ bản chất của các bài toán, người đọc nên tìm hiểu các công thức tính toán của các bài toán (xem thêm trong các tài liệu đã dẫn trong phần danh mục tham khảo), để thực hiện cài đặt lại các hàm thay vì sẵn có của Python, nhằm hiểu rõ bản chất của quá trình thống kê. Tiếp đó cần thực hiện qua các dự án có tính chất thực tiễn với dữ liệu có thể tham khảo trên một số trang như [kaggle.com](https://www.kaggle.com)... để nâng cao năng lực giải quyết vấn đề thực tiễn, tiếp tục phát triển cho các bài toán xử lý, phân tích, khai phá dữ liệu... đây cũng chính là hướng phát triển của sách này.

TÀI LIỆU THAM KHẢO

- [1]. Phạm Văn Chững (2016). Thống kê ứng dụng. NXB ĐHQG Thành phố Hồ Chí Minh
- [2]. Nguyễn Thế Dũng (2021). Đánh giá khả năng của ngôn ngữ lập trình Python trong thống kê ứng dụng. Đề tài cấp Cơ sở. T21 – TN – 05. ĐH Huế.
- [3]. Edouard Duchesnay, Tommy Löfstedt, Feki Younes (2021). Statistics and Machine Learning in Python, Release 0.5. <https://duchesnay.github.io/pystatsml/>
- [4]. Stanley H. Chan (2021). Introduction to Probability for Data Science. Published in the United States of America by Michigan Publishing.
- [5]. Đặng Hùng Thắng (1999). Thống kê và ứng dụng. NXB Giáo dục
- [6]. Thomas Haslwanter (2016). An Introduction to Statistics with Python With Applications in the Life Sciences. Springer International Publishing Switzerland 2016.
- [7]. Zach (2021). <https://www.statology.org/python-guides/>
- [8]. Zach (2021). <https://www.statology.org/tutorials/>

Website

<https://docs.python.org/>

<https://numpy.org/>

<https://pandas.pydata.org/>

<https://scipy.org/>

<https://matplotlib.org/>