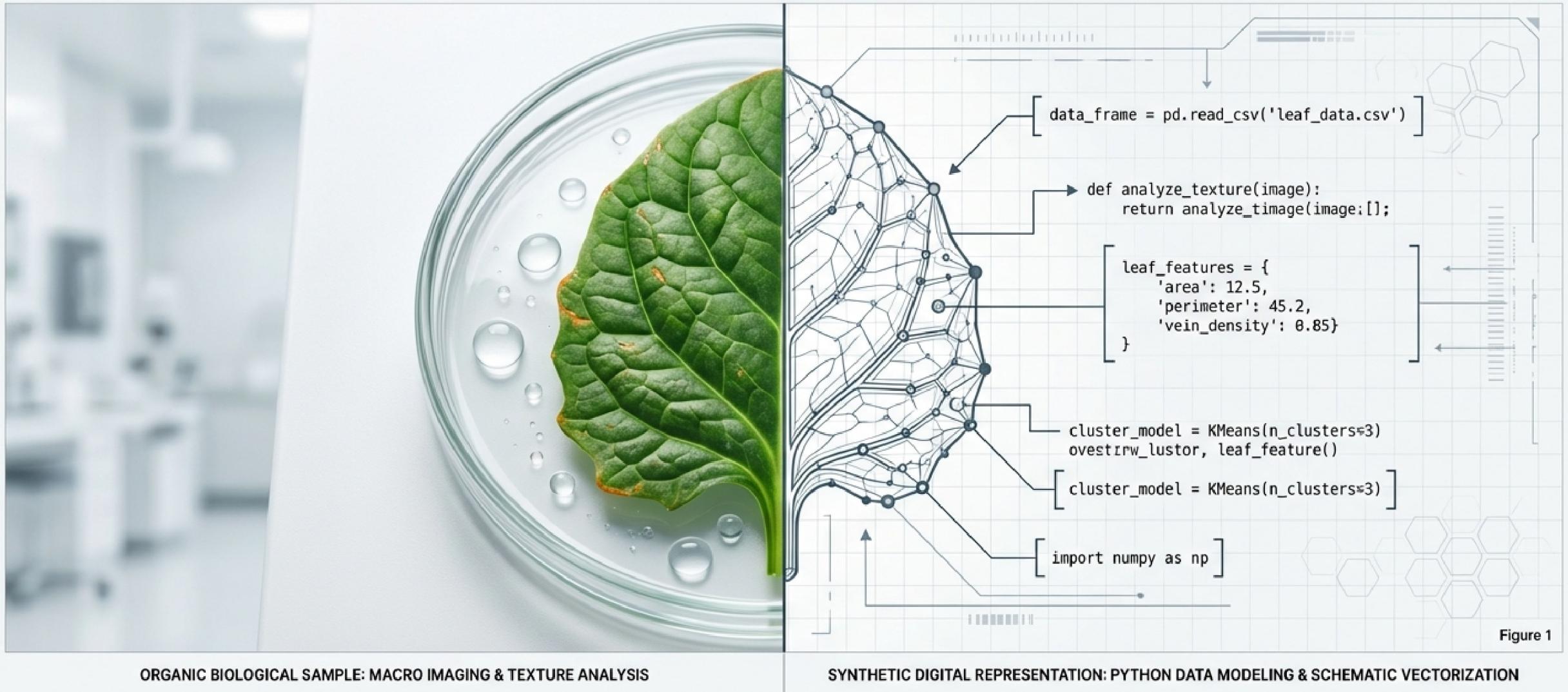


# AI-Assisted Food Science Data Analysis with Python

## A Pedagogical Framework for the Next Generation of Researchers



# Research begins with questions, not software.

VOL. 42 | ISSUE 7 | SCIENTIFIC METHODS JOURNAL | OCTOBER 26, 2023



## CORE STATEMENT

Python and AI do not replace your thinking; they amplify it.

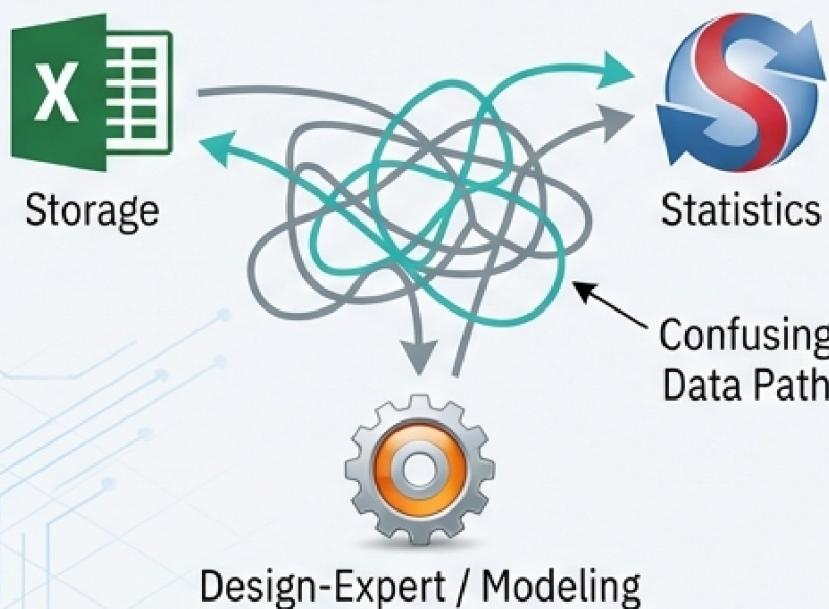
## KEY POINTS

- **The Critical Component:** You set the question, design the experiment, and interpret the limits.
- **The Tool's Role:** Speed, systemization, and deeper analysis.
- **Your Role:** The decision maker who commands the tool.

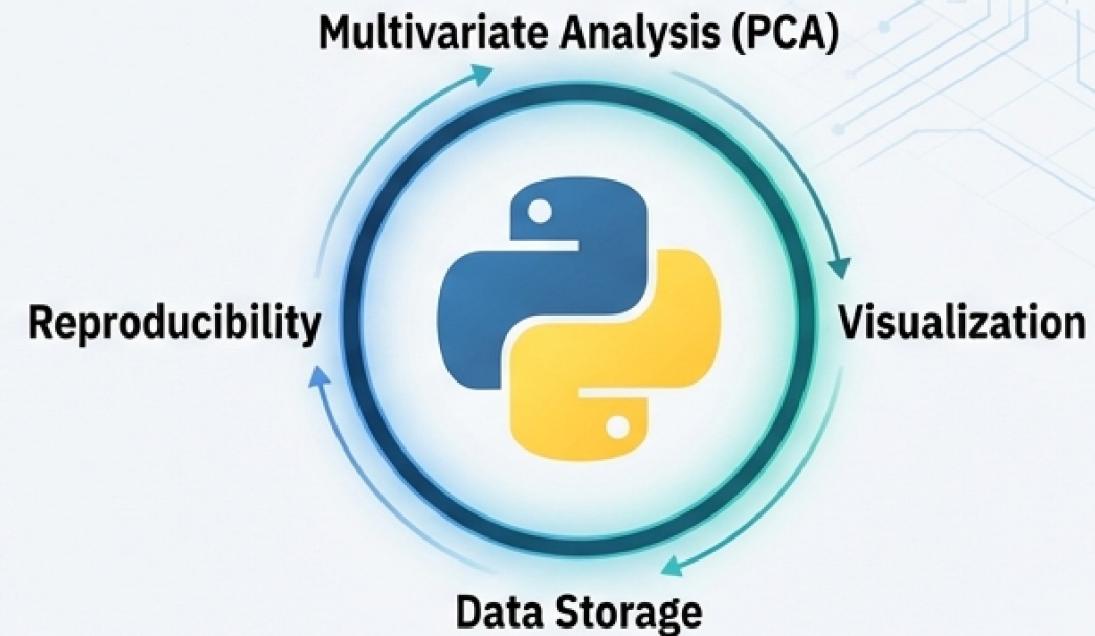
# The Unified Workbench

VOL. 42 | ISSUE 7 | SCIENTIFIC METHODS JOURNAL | OCTOBER 26, 2023

## CURRENT STATE: FRAGMENTED WORKFLOW



## FUTURE STATE: CONSOLIDATED ENVIRONMENT



## BENEFITS:

- 1. Centralized:** Perform analysis, modeling, and plotting in one environment.
- 2. Reproducible:** Code provides a traceable history of every step, unlike clicking menus.
- 3. Academic Standard:** Ideal for graduate-level research where auditability is key.

[Home](#)[Environments](#)[Learning](#)[Community](#)**Anaconda Quick Start Environments**

Jump into pre-configured environments for ML, Finance & more. No setup headaches, just productivity.

[Launch Your Environment](#)[Documentation](#)[Anaconda Blog](#)

All applications on base (root) Channels



PyCharm

The only Python IDE you need – built for data and AI/ML professionals. Supercharged with an AI-enhanced IDE experience. Free forever, plus one month of Pro included.

[Install](#)

Anaconda AI Navigator

Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.

[Install](#)

Anaconda Toolbox

0.4.0

Anaconda Assistant

JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.

[Launch](#)

Anaconda Cloud Notebooks

Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

[Launch](#)

JupyterLab

3.6.7

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

[Launch](#)

Notebook

6.5.4

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

[Launch](#)

Qt Console

5.5.1

PyQt GUI that supports inline figures, proper multi-line editing with syntax highlighting, graphical calltips, and more.

[Launch](#)

Spyder

5.5.1

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

[Launch](#)

EduBlocks

Web-based coding platform from Anaconda designed for students. Learn Python coding through an interactive, block-based visual environment.

[Launch](#)

watsonx™

IBM watsonx

IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring.

[Launch](#)

Cloud Infrastructure

Oracle Data Science Service

OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools

[Launch](#)

PyScript

Code and share Python in the Browser. A vibrant community of makers, builders, and hackers building the next frontier of Python-powered web applications.

[Launch](#)

PythonAnywhere

Host, run, and code Python in the cloud! Get started for free.



Glueviz

Multidimensional data visualization across files. Explore relationships within and among related datasets.

1.2.4



Orange 3

3.38.1

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows



RStudio

1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Spyder (Python 3.11)

/Users/saowalukrungchang/Desktop/CMU/การงาน/Publication2025/Oil/untitled3.py

abc.py GCMS PDF.py untitled0.py\* 1\_DPPH.py\* untitled3.py < > =

1 #!/usr/bin/env python3  
2 # -\*- coding: utf-8 -\*-  
3  
4 Created on Tue Feb 3 10:39:47 2026  
5  
6 @author: saowalukrungchang  
7  
8

Command

Graph

Plots Variable Explorer

Console 1/A

In [37]: n

Result

rcparams

completions: conda(base) ✓ LSP: Python Line 9, Col 1 UTF-8 LF RW Mem 94%

Spyder (Python 3.11)

/Users/saowalukrungchang/Desktop/AJ JUM/OatUltrasound/untitled54.py

untitled52.py\*    untitled53.py\*    untitled54.py\* < > =

1 #!/usr/bin/env python3  
2 # -\*- coding: utf-8 -\*-  
3 """  
4 Created on Wed Jul 9 17:11:01 2025  
5 @author: saowalukrungchang  
6 """  
7  
8

Variable

Graph

Command

Console 1/A    Console 2/A

Python 3.11.7 (main, Dec 15 2023, 12:09:04) [Clang 14.0.6]  
Type "copyright", "credits" or "license" for more information.  
IPython 8.20.0 -- An enhanced Interactive Python.  
In [1]:

Result

conda: base (Python 3.11.7)    Completions: conda(base)    LSP: Python Line 8, Col 1    UTF-8    LF    RW    Mem 93%

Run

Shift+Enter

A screenshot of the Spyder Python IDE interface. A large red arrow points from the 'Run' button at the top left towards the code editor area. The code editor shows a Python script named 'untitled54.py' with the following content:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Wed Jul 9 17:11:01 2025
5 @author: saowalukrungchang
6 """
7
8
```

The word '1+1' is overlaid in red text on the left side of the code editor. In the bottom right corner of the code editor, the text 'Result????' is overlaid in red. The file browser on the right shows a single folder entry. The bottom right corner of the interface displays the system status bar with the text: 'conda: base (Python 3.11.7) Completions: conda(base) LSP: Python Line 8, Col 1 UTF-8 LF RW Mem 93%'. The bottom center of the screen shows the Python console output:

```
Python 3.11.7 (main, Dec 15 2023, 12:09:04) [Clang 14.0.6 ]
Type "copyright", "credits" or "license" for more information.

IPython 8.20.0 -- An enhanced Interactive Python.

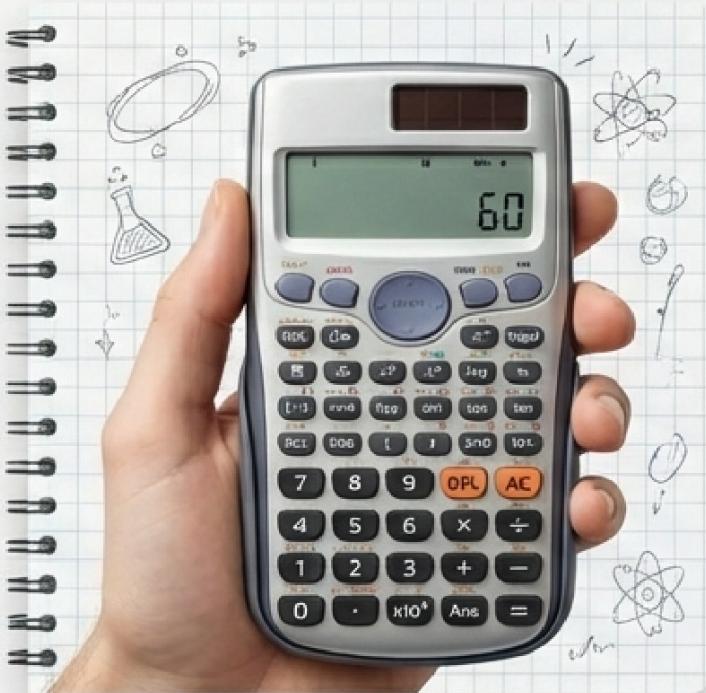
In [1]:
```

# starts a comment in Python.

Python does not execute commented text.

# The Super-Calculator

At its core, Python performs the same arithmetic you do in a notebook, just faster.

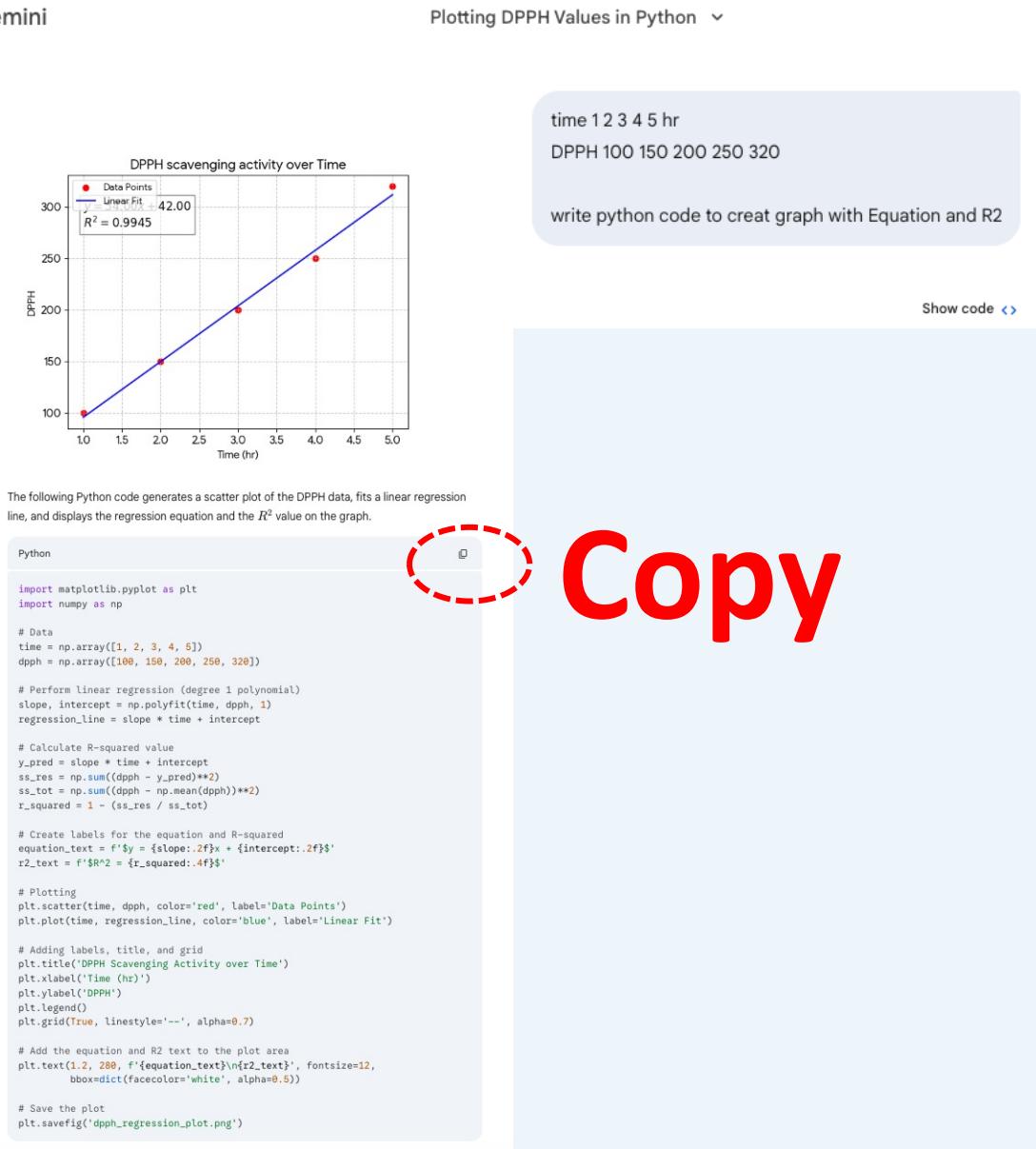


```
#Basic Addition  
2 + 3  
  
# Calculating an Average  
(50 + 60 + 70) / 3
```

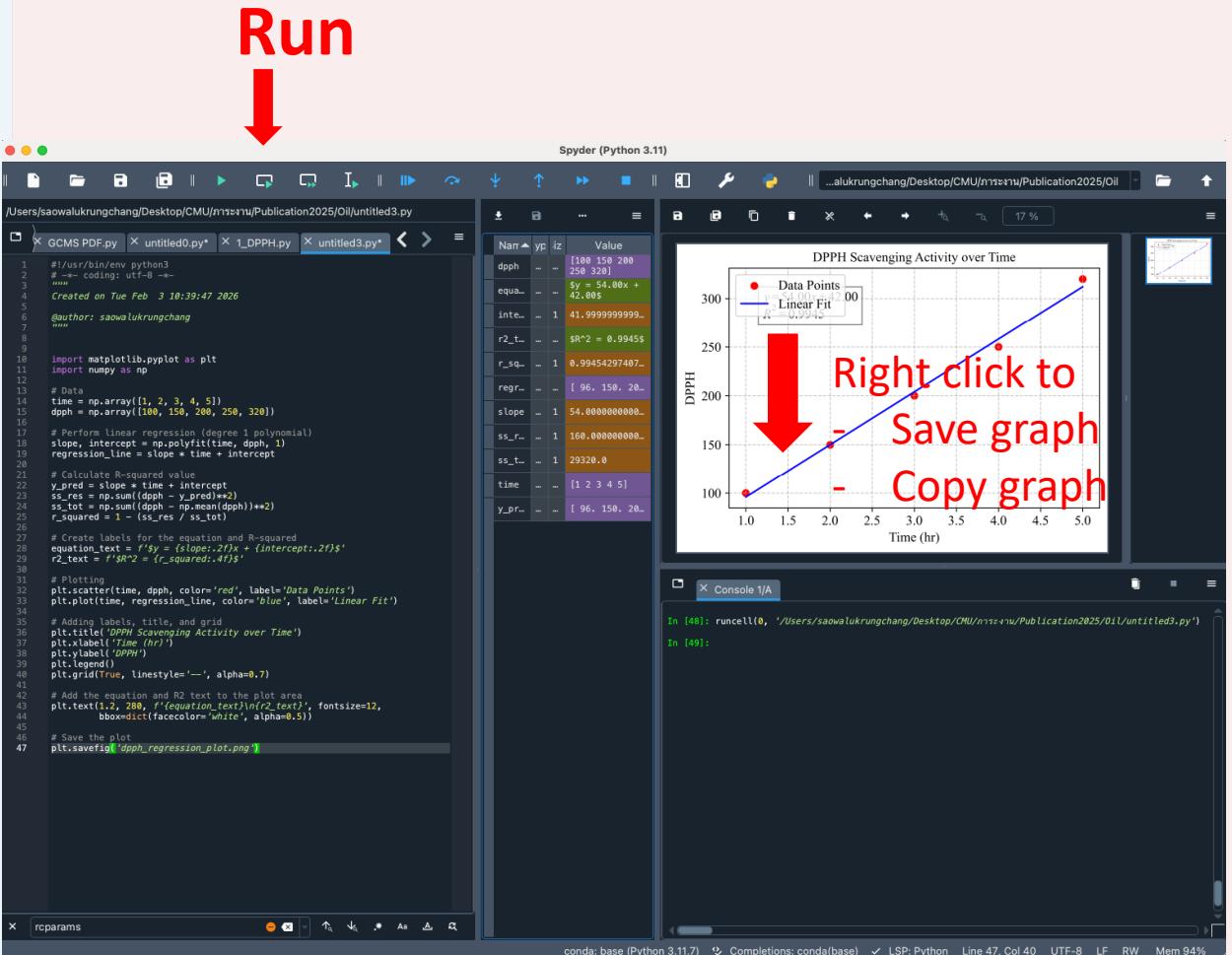
Translates to:  
Calculating the mean of  
three sample runs.

# AI (Chat, Gemini, Copilot,.....)

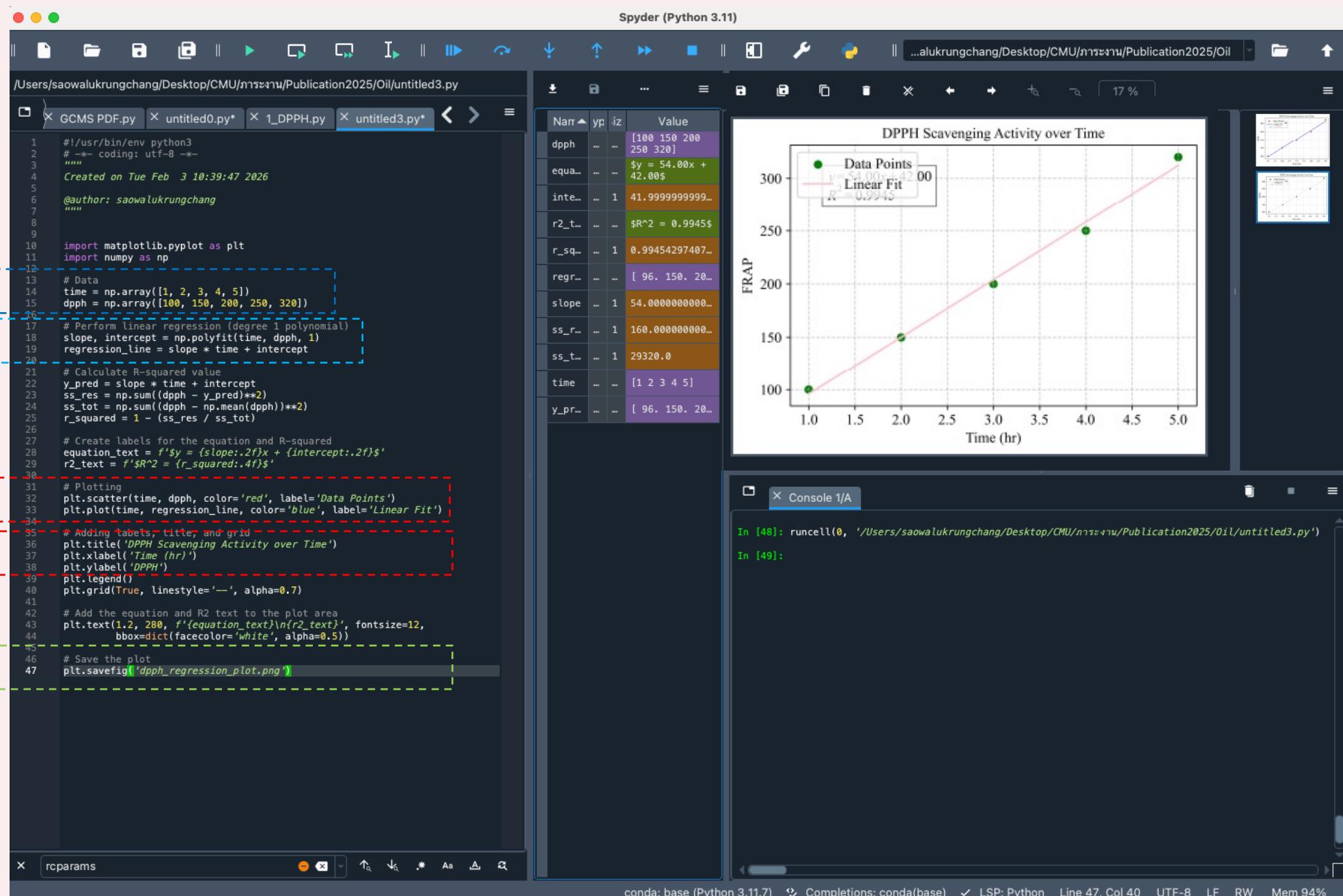
Gemini

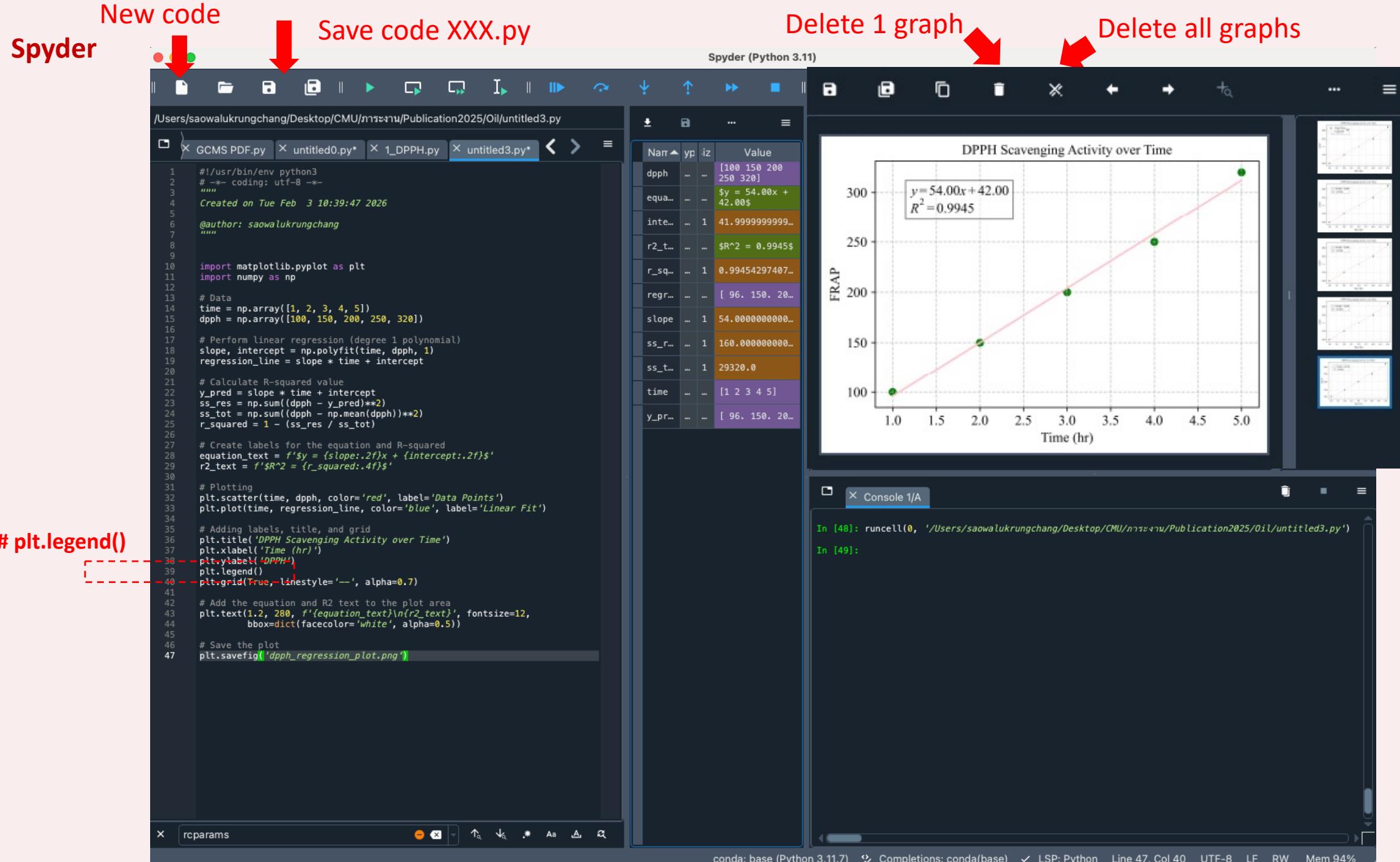


# Spyder

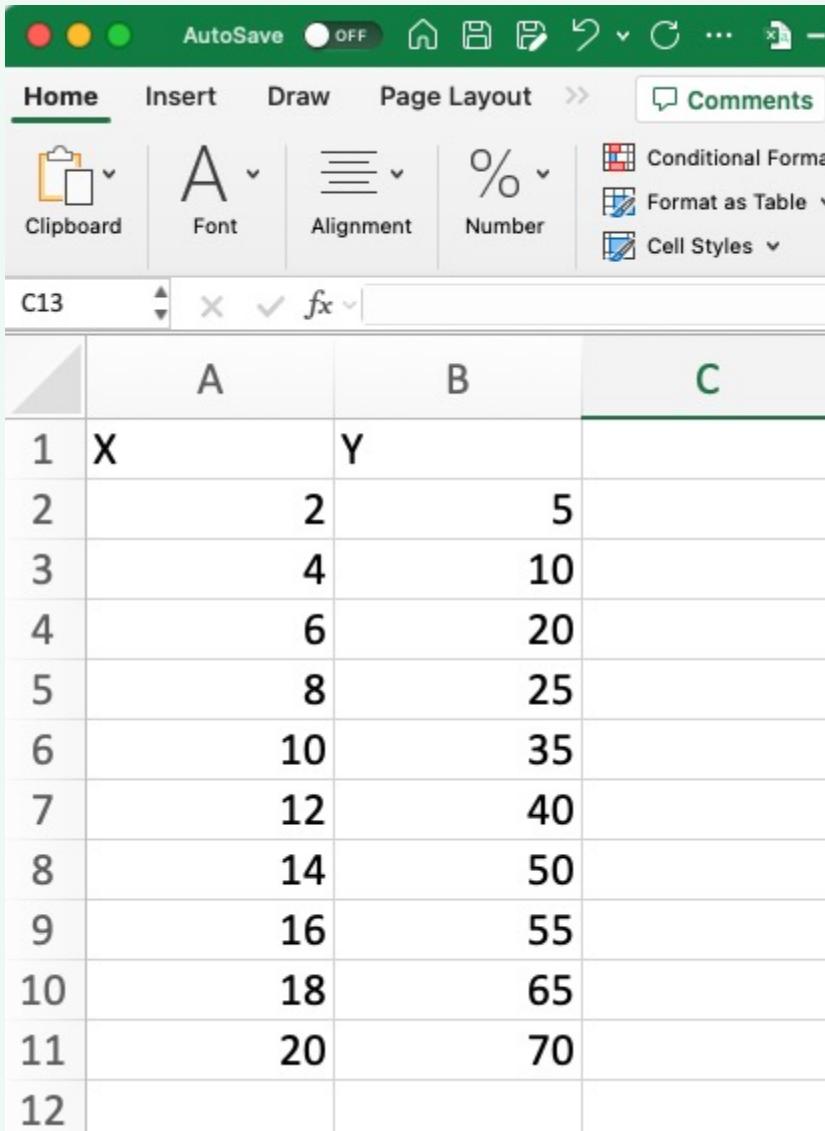


# Spyder





# Excel

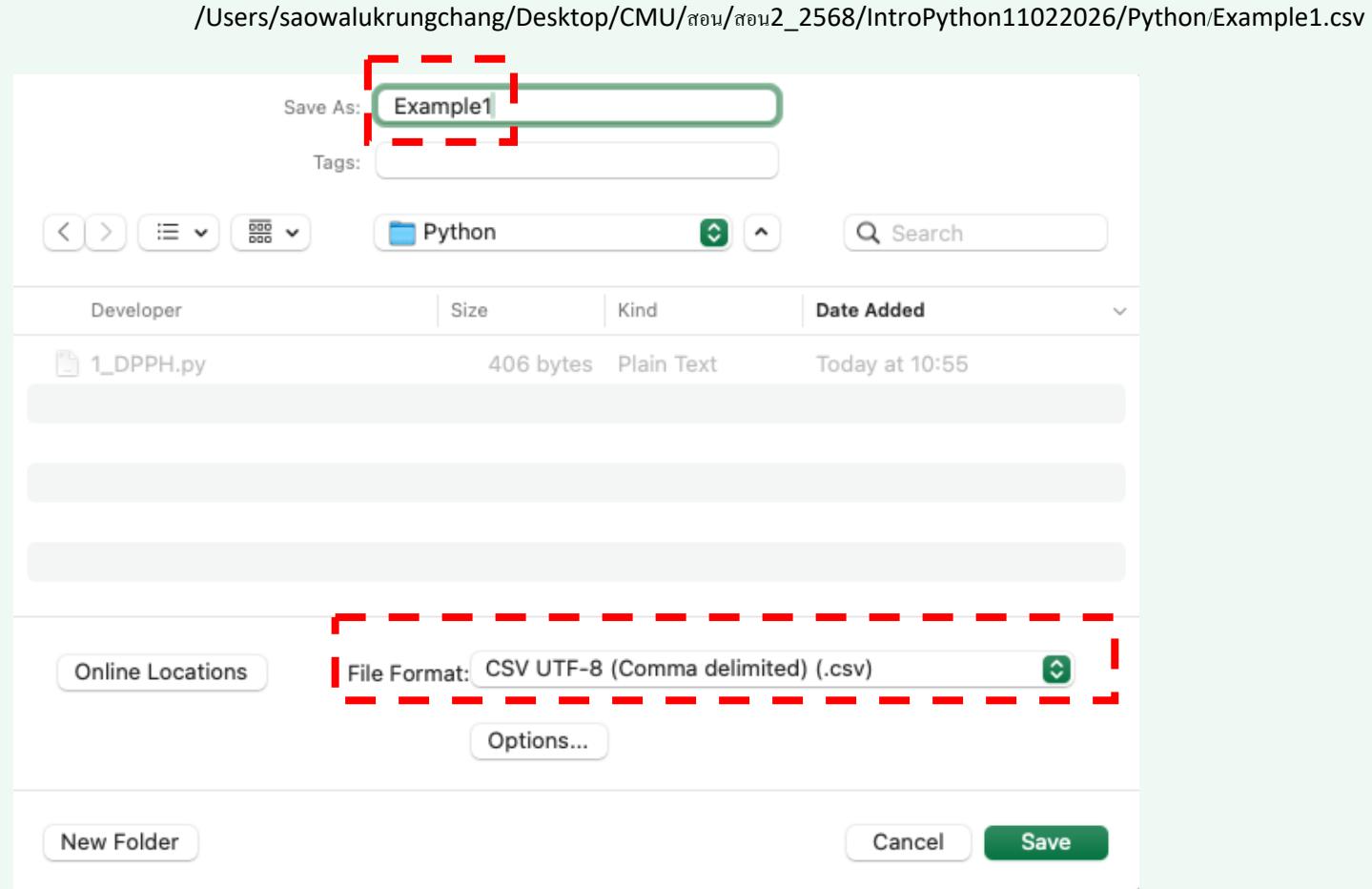


	A	B	C
1	X	Y	
2	2	5	
3	4	10	
4	6	20	
5	8	25	
6	10	35	
7	12	40	
8	14	50	
9	16	55	
10	18	65	
11	20	70	
12			

- Input X, Y 10 Data

- Save file : Desktop or C:, D:.....

- File name : Example1.csv



# AI (Chat, Gemini, Copilot,.....)

1. Where is your data?
2. What is your data? Please describe it.
3. What do you want to analyze?

The data are stored at the following file path:

/Users/saowalukrungchang/Desktop/CMU/สอน/สหศว2\_2568/IntroPython11022026/Python/Example1.csv

- X is located in column 1.
- Y is located in column 2.

Please write Python code to plot the graph and display the regression equation and the R<sup>2</sup> value.

```
Python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# 1. Load the data
# Replace 'Example1.csv' with the full path if the script is not in the same folder
file_path = 'Example1.csv'
df = pd.read_csv(file_path)

# 2. Extract X and Y (1st and 2nd columns)
# .iloc[:, 0] selects the first column, .iloc[:, 1] selects the second column
x = df.iloc[:, 0].values
y = df.iloc[:, 1].values

# 3. Perform linear regression (Degree 1 polynomial)
slope, intercept = np.polyfit(x, y, 1)
y_pred = slope * x + intercept

# 4. Calculate R-squared
ss_res = np.sum((y - y_pred)**2)
ss_tot = np.sum((y - np.mean(y))**2)
r_squared = 1 - (ss_res / ss_tot)

# 5. Create strings for the equation and R2
equation_text = f'y = {slope:.4f}x + {intercept:.4f}'
r2_text = f'R^2 = {r_squared:.4f}'

# 6. Plotting
plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='blue', label='Data Points')
plt.plot(x, y_pred, color='red', label='Linear Fit')

# Adding labels (change these to match your actual data names)
plt.title('Linear Regression Analysis')
plt.xlabel(df.columns[0]) # Uses the name of the 1st column
plt.ylabel(df.columns[1]) # Uses the name of the 2nd column

# Add the equation and R2 to the plot
# Adjust the coordinates (x_pos, y_pos) to move the text box
plt.text(0.05, 0.95, f'{equation_text}\n{r2_text}', transform=plt.gca().transAxes, fontsize=12,
        verticalalignment='top', bbox=dict(facecolor='white', alpha=0.5))

plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)

# Save and Show
plt.savefig('regression_plot.png', dpi=300)
plt.show()

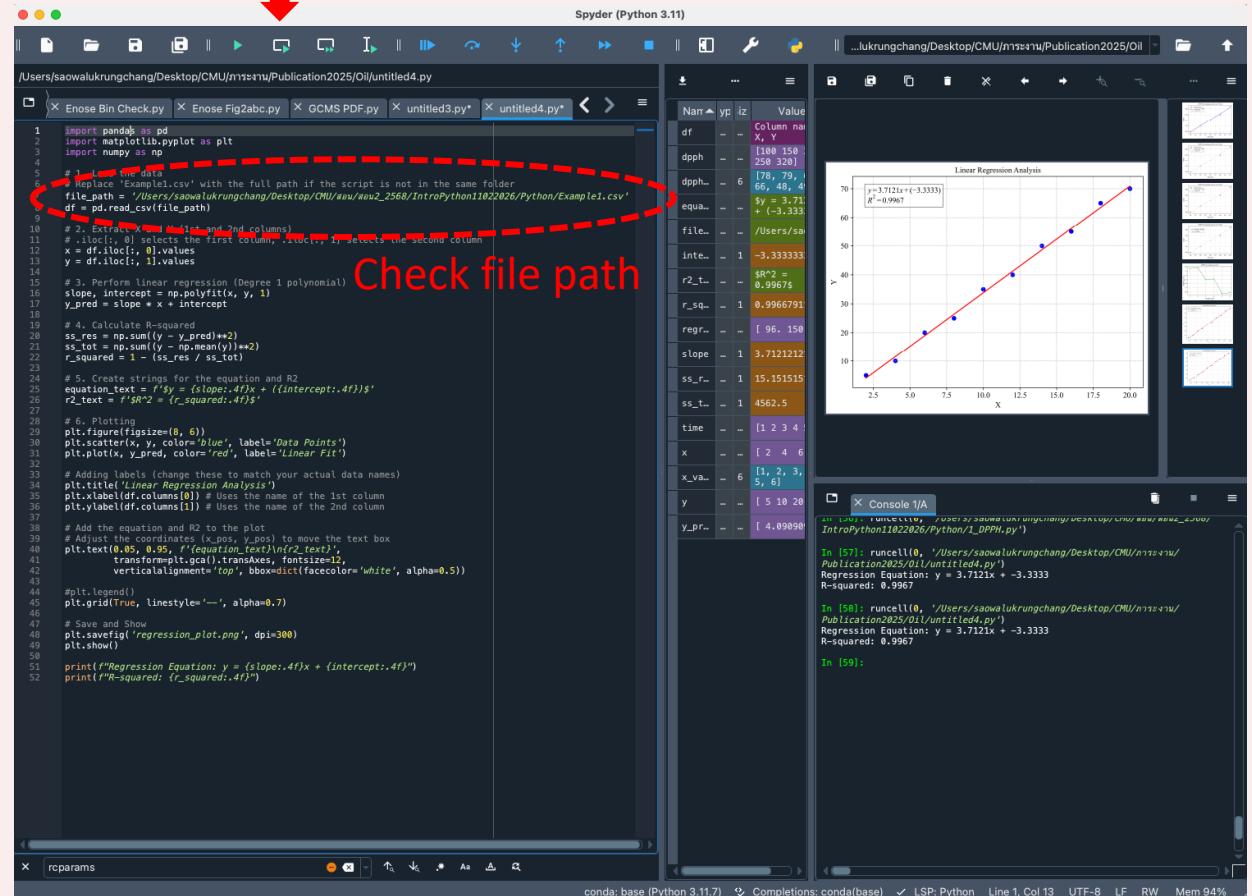
print(f'Regression Equation: y = {slope:.4f}x + {intercept:.4f}')
print(f'R-squared: {r_squared:.4f}')
```

file path

Copy

# Spyder

Run



**Right click**

file path

How to find a File Path on Windows

1. Right-click Method (Similar to Mac's "Get Info")
  - Right-click on the file or folder.
  - Select **Properties** (at the bottom of the menu).
  - In the **General** tab, look for **Location**: This is your file path.
  - You can highlight and **Copy** it from there.
2. "Copy as Path" Shortcut (Fastest for Coding)
 

This is the most convenient way to get a path for your Python script:

  - Hold the **Shift** key on your keyboard and **Right-click** the file.
  - Select "**Copy as path**".
  - This copies the full path to your clipboard, looking like this:  
"C:\Users\Name\Desktop\Example1.csv".
3. Using the Address Bar
  - Open the folder where your file is located.
  - Click on the **Address Bar** at the top of the File Explorer window.
  - The breadcrumbs will turn into a full text path (e.g., C:\Users\Documents\Python) which you can copy immediately.

Excel

- File name : Example1.csv

A	B
X	Y
2	5
4	10
6	20
8	25
10	35
12	40
14	50
16	55
18	65
20	70

- Input X, Y more 10 Data

A	B
X	Y
2	5
4	10
6	20
8	25
10	35
12	40
14	50
16	55
18	65
20	70
22	78.33333333
24	85.7575758
26	93.1818182
28	100.606061
30	108.030303
32	115.454545
34	122.878788
36	130.30303
38	137.727273
40	145.151515

- Save file : Example1.csv

# Spyder Run

Spyder (Python 3.11)

File Edit View Insert Cell Kernel Help

...lukrunchang/Desktop/CMU/ກາຊະໜາ/Publication2025/Oil/untitled4.py

Enose Bin Check.py Enose Fig2abc.py GCMS PDF.py untitled3.py\* untitled4.py\* < > ⌂

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 # 1. Load the data
6 # Replace 'Example1.csv' with the full path if the script is not in the same folder
7 file_path = '/Users/saowalukrungchang/Desktop/CMU/ສວຸນ້ອງໜັງປົງ/IntroPython11022026/Python/Example1.csv'
8 df = pd.read_csv(file_path)
9
10 # 2. Extract X and Y (1st and 2nd columns)
11 # .iloc[:, 0] selects the first column, .iloc[:, 1] selects the second column
12 x = df.iloc[:, 0].values
13 y = df.iloc[:, 1].values
14
15 # 3. Perform linear regression (Degree 1 polynomial)
16 slope, intercept = np.polyfit(x, y, 1)
17 y_pred = slope * x + intercept
18
19 # 4. Calculate R-squared
20 ss_res = np.sum((y - y_pred)**2)
21 ss_tot = np.sum((y - np.mean(y))**2)
22 r_squared = 1 - (ss_res / ss_tot)
23
24 # 5. Create strings for the equation and R2
25 equation_text = f'y = {slope:.4f}x + ({intercept:.4f})$'
26 r2_text = f'R^2 = {r_squared:.4f}$'
27
28 # 6. Plotting
29 plt.figure(figsize=(8, 6))
30 plt.scatter(x, y, color='blue', label='Data Points')
31 plt.plot(x, y_pred, color='red', label='Linear Fit')
32
33 # Adding labels (change these to match your actual data names)
34 plt.title('Linear Regression Analysis')
35 plt.xlabel(df.columns[0]) # Uses the name of the 1st column
36 plt.ylabel(df.columns[1]) # Uses the name of the 2nd column
37
38 # Add the equation and R2 to the plot
39 # Adjust the coordinates (x_pos, y_pos) to move the text box
40 plt.text(0.05, 0.95, f'{equation_text}\n{r2_text}', transform=plt.gca().transAxes, fontsize=12,
41         verticalalignment='top', bbox=dict(facecolor='white', alpha=0.5))
42
43 #plt.legend()
44 plt.grid(True, linestyle='--', alpha=0.7)
45
46 # Save and Show
47 plt.savefig('regression_plot.png', dpi=300)
48 plt.show()
49
50 print(f'Regression Equation: y = {slope:.4f}x + {intercept:.4f}')
51 print(f'R-squared: {r_squared:.4f}')
52
```

variable explorer

Name	Type	Value
df	...	Column names: X, Y [100 150 250 320]
dpph	...	6 [78, 79, 66, 48, 4]
equa...	...	\$y = 3.7121x + (-3.3333)
file...	...	/Users/sa...
inte...	...	1 -3.333333
r2_t...	...	R^2 = 0.99675
r_sq...	...	0.9966791
regr...	...	[ 96. 150]
slope	...	1 3.7121212
ss_r...	...	1 15.151515
ss_t...	...	1 4562.5
time	...	[1 2 3 4]
x	...	[ 2 4 6]
x_va...	...	6 [1, 2, 3, 5, 6]
y	...	[ 5 10 20]
y_pr...	...	[ 4.09090]

plot explorer

Linear Regression Analysis

Console 1/A

```
In [57]: runcell(0, '/Users/saowalukrungchang/Desktop/CMU/ກາຊະໜາ/Publication2025/Oil/untitled4.py')
Regression Equation: y = 3.7121x + -3.3333
R-squared: 0.9967

In [58]: runcell(0, '/Users/saowalukrungchang/Desktop/CMU/ກາຊະໜາ/Publication2025/Oil/untitled4.py')
Regression Equation: y = 3.7121x + -3.3333
R-squared: 0.9967

In [59]:
```

# Excel

- Input X, Y 5 Data
- Save file : Same folder as Example1.csv
- File name : Example2.csv

A screenshot of Microsoft Excel. The ribbon at the top shows tabs for Home, Insert, Draw, and Page Layout. The Home tab is selected. On the left, there are icons for Paste, Bold (B), Italic (I), Underline (U), and Alignment. The main area displays a table with columns A, B, C, and D. Row 1 contains headers X and Y. Rows 2 through 6 contain data points: (100, 500), (150, 400), (200, 300), (250, 200), and (300, 100). A message bar at the bottom says "Possible Data Loss Some features might not work".

	A	B	C	D
1	X	Y		
2	100	500		
3	150	400		
4	200	300		
5	250	200		
6	300	100		
7				
8				
9				

# Spyder

A screenshot of the Spyder Python IDE. The top menu bar shows "Spyder (Python 3.11)". The left sidebar lists open files: Enose Bin Check.py, Enose Fig2abc.py, GCMS PDF.py, untitled3.py\*, and untitled4.py\*. The main area shows a code editor with a script named "untitled4.py". The script performs linear regression on data from "Example1.csv" and saves the results to "Example2.csv". A red dashed circle highlights the "Example2.csv" line in the code. To the right of the code editor is a data frame viewer showing the contents of "df". Below the code editor is a plot titled "Linear Regression Analysis" showing a scatter plot of X vs Y with a linear regression line. The bottom section is the "Console 1/A" which shows the execution of the script and its output.

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

# 1. Load the data
# Replace 'Example1.csv' with the full path if the script is not in the same folder
file_path = '/Users/saowalukrungchang/Desktop/CMU/นิสิตานุ/Publication2025/Oil/untitled4.py'
df = pd.read_csv(file_path)

# 2. Extract X and Y (1st and 2nd columns)
# .iloc[:, 0] selects the first column, .iloc[:, 1] selects the second column
x = df.iloc[:, 0].values
y = df.iloc[:, 1].values

# 3. Perform linear regression (Degree 1 polynomial)
slope, intercept = np.polyfit(x, y, 1)
y_pred = slope * x + intercept

# 4. Calculate R-squared
ss_res = np.sum((y - y_pred)**2)
ss_tot = np.sum((y - np.mean(y))**2)
r_squared = 1 - (ss_res / ss_tot)

# 5. Create strings for the equation and R2
equation_text = f'y = {slope:.4f}x + ({intercept:.4f})'
r2_text = f'R^2 = {r_squared:.4f}'

# 6. Plotting
plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='blue', label='Data Points')
plt.plot(x, y_pred, color='red', label='Linear Fit')
plt.title('Linear Regression Analysis')
plt.xlabel(df.columns[0]) # Use the name of the 1st column
plt.ylabel(df.columns[1]) # Use the name of the 2nd column

# Adding labels (change these to match your actual data names)
plt.text(0.05, f'(equation_text)\n{r2_text}', transform=plt.gca().transAxes, fontsize=12,
        verticalalignment='top', bbox=dict(facecolor='white', alpha=0.5))

# Save and Show
plt.savefig('regression_plot.png', dpi=300)
plt.show()

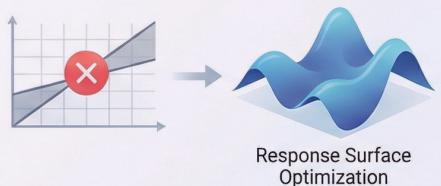
print(f'Regression Equation: y = {slope:.4f}x + {intercept:.4f}')
print(f'R-squared: {r_squared:.4f}')
```

In [57]: runcell(0, '/Users/saowalukrungchang/Desktop/CMU/นิสิตานุ/Publication2025/Oil/untitled4.py')  
Regression Equation: y = 3.7121x + -3.3333  
R-squared: 0.9967  
In [58]: runcell(0, '/Users/saowalukrungchang/Desktop/CMU/นิสิตานุ/Publication2025/Oil/untitled4.py')  
Regression Equation: y = 3.7121x + -3.3333  
R-squared: 0.9967  
In [59]:

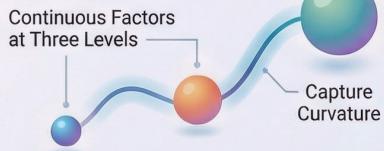
# RSM Example

## Finding the “Sweet Spot”: Mastering Response Surface Methodology (RSM)

### The Transition to Optimization



### The Three-Level Rule



### Balancing Competing Goals



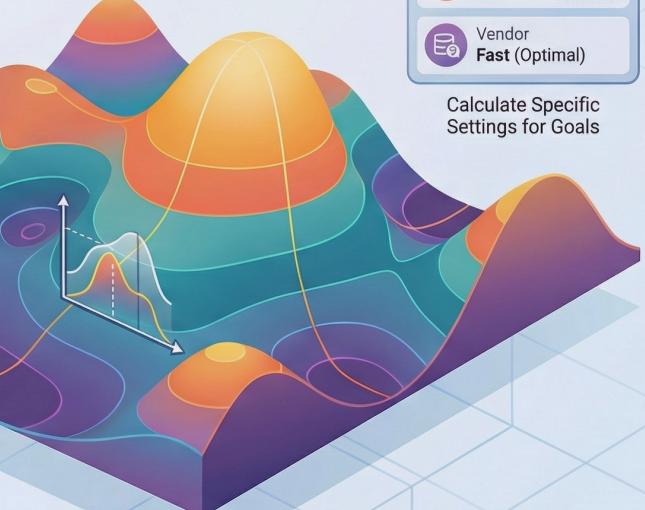
### The Workflow in Action

#### Model the Response Surface



#### Visualize the “Landscape”

3D Surface Plots & Cross-Section Profiles

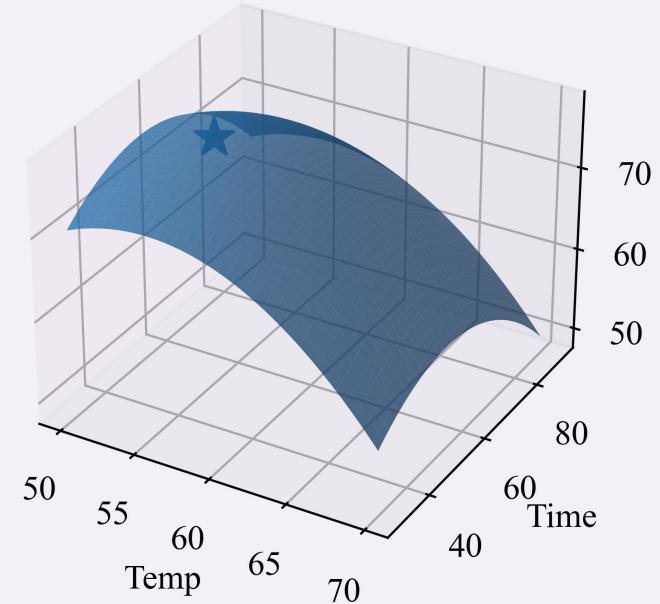
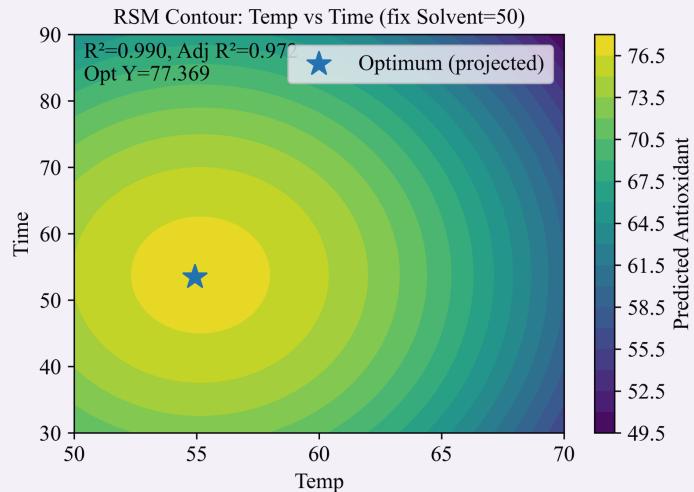


#### Predict the Optimum

- pH Level 6.85 (Optimal)
- Temperature 34.25° C (Optimal)
- Vendor Fast (Optimal)

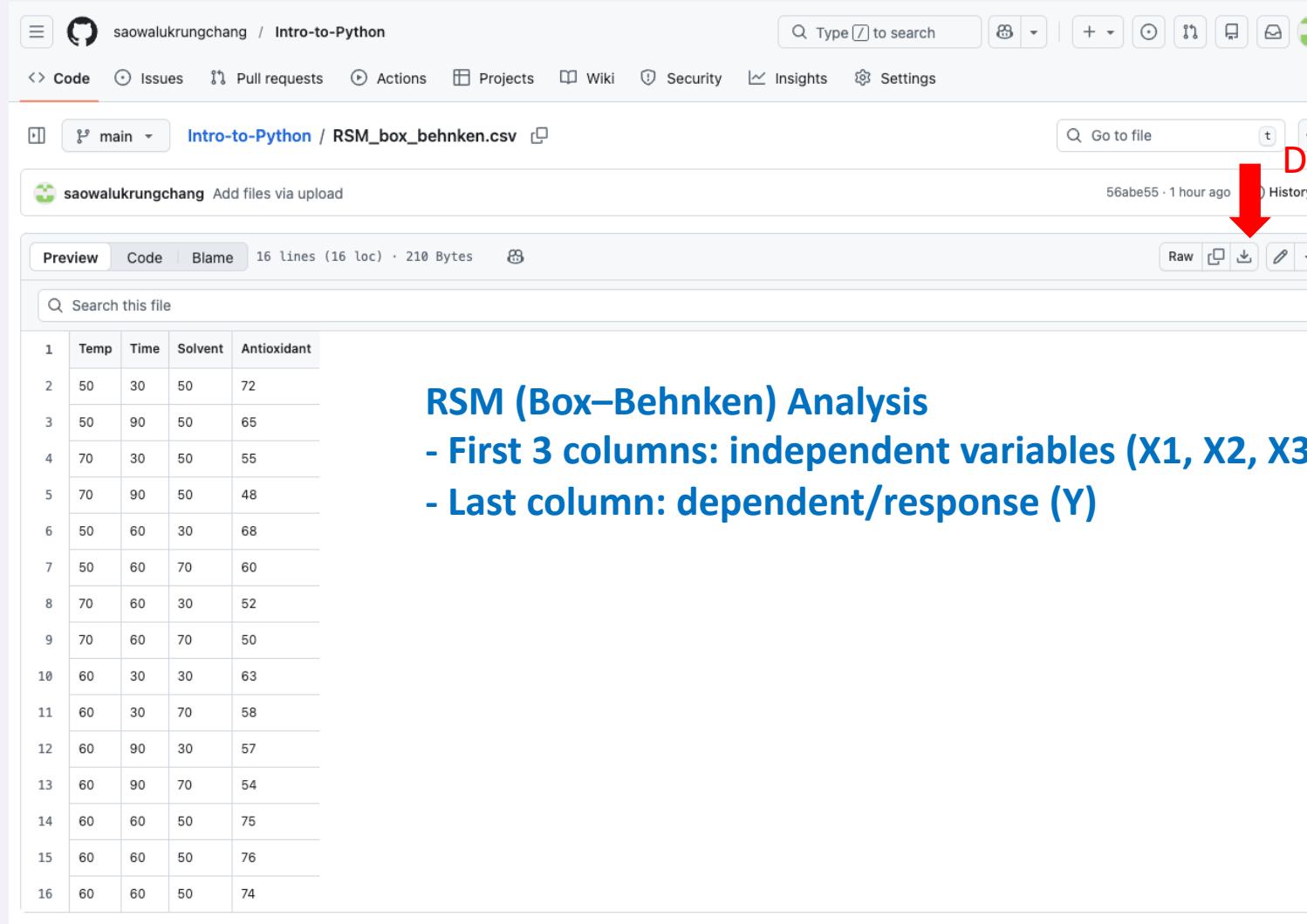
Calculate Specific Settings for Goals

© NotebookLM



# RSM Example

**Data** [https://github.com/saowalukrungchang/Intro-to-Python/blob/main/RSM\\_box\\_behnken.csv](https://github.com/saowalukrungchang/Intro-to-Python/blob/main/RSM_box_behnken.csv)



saowalukrungchang / Intro-to-Python

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main / Intro-to-Python / RSM\_box\_behnken.csv

saowalukrungchang Add files via upload

56abe55 · 1 hour ago History

Preview Code Blame 16 lines (16 loc) · 210 Bytes

Raw Download Edit

Search this file

	Temp	Time	Solvent	Antioxidant
1	50	30	50	72
2	50	90	50	65
3	70	30	50	55
4	70	90	50	48
5	50	60	30	68
6	50	60	70	60
7	70	60	30	52
8	70	60	70	50
9	60	30	30	63
10	60	90	30	58
11	60	90	70	57
12	60	60	50	54
13	60	60	50	75
14	60	60	50	76
15	60	60	50	74
16	60	60	50	76

Download



**RSM (Box–Behnken) Analysis**  
- First 3 columns: independent variables (X<sub>1</sub>, X<sub>2</sub>, X<sub>3</sub>)  
- Last column: dependent/response (Y)

# AI (Chat, Gemini, Copilot,.....)

## 1. Where is your data?

File path.....

File name.....

# Copy



## 3. What do you want to analyze?

RSM (Box–Behnken) Analysis  
create graph 2D and 3D to show the optimum condition with star on response graph, show statistic analysis equation r2

```
python

# =====#
# RSM (Box–Behnken) Analysis from GitHub CSV
# - First 3 columns: independent variables (X1, X2, X3)
# - Last column: dependent/response (Y)
#
# Outputs:
# 1) Quadratic RSM model (statsmodels) + equation + R² / Adj R²
# 2) ANOVA table (and Lack-of-Fit if replicates exist)
# 3) Optimum condition (maximize Y) via grid search
# 4) 2D contour plots (pairwise) with optimum marked by STAR
# 5) 3D surface plots (pairwise) with optimum marked by STAR
#
# Notes:
# - Uses RAW GitHub link (recommended for pd.read_csv)
# - Plots show surfaces for each pair while fixing the 3rd factor at its center value
# =====#


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import statsmodels.formula.api as smf
from statsmodels.stats.anova import anova_lm

# -----
# 1) Load data from GitHub (RAW URL)
# -----
url = "https://raw.githubusercontent.com/saowalukrungchang/Intro-to-Python/main/RSM_box_benken.csv"
df = pd.read_csv(url)

# Identify columns
x_cols = df.columns[:3].tolist() # first 3 columns
y_col = df.columns[-1] # last column

# Ensure numeric & drop missing
for c in x_cols + [y_col]:
    df[c] = pd.to_numeric(df[c], errors='coerce')
df = df.dropna(subset=x_cols + [y_col]).reset_index(drop=True)

print("== Columns ==")
print("X:", x_cols)
print("Y:", y_col)
print("N:", len(df))
print(df.head())

# -----
# 2) Fit Quadratic RSM model (full quadratic for 3 factors)
#   Y ~ X1 + X2 + X3 + X1X2 + X1X3 + X2X3 + X1^2 + X2^2 + X3^2
# -----


x1, x2, x3 = x_cols

formula = (
    f'{y_col} ~ {x1} + {x2} + {x3}' +
    f' + {x1}*{x2} + {x1}*{x3}' +
    f' + {x2}*{x3}' +
    f' + {x1}**2 + {x2}**2 + {x3}**2'
)
```

# Spyder

Output : 6 graph

The Spyder interface shows the following components:

- Code Editor:** Displays the Python script `RSM.py` containing the RSM analysis code.
- Output Console:** Shows the command `conda: base (Python 3.11.7)` and the results of the ANOVA table.
- Data Viewer:** Shows a table of data with columns `ax`, `yp`, `iz`, and `Va`.
- Figure Area:** Shows a 2D contour plot with a star mark indicating the optimum condition.

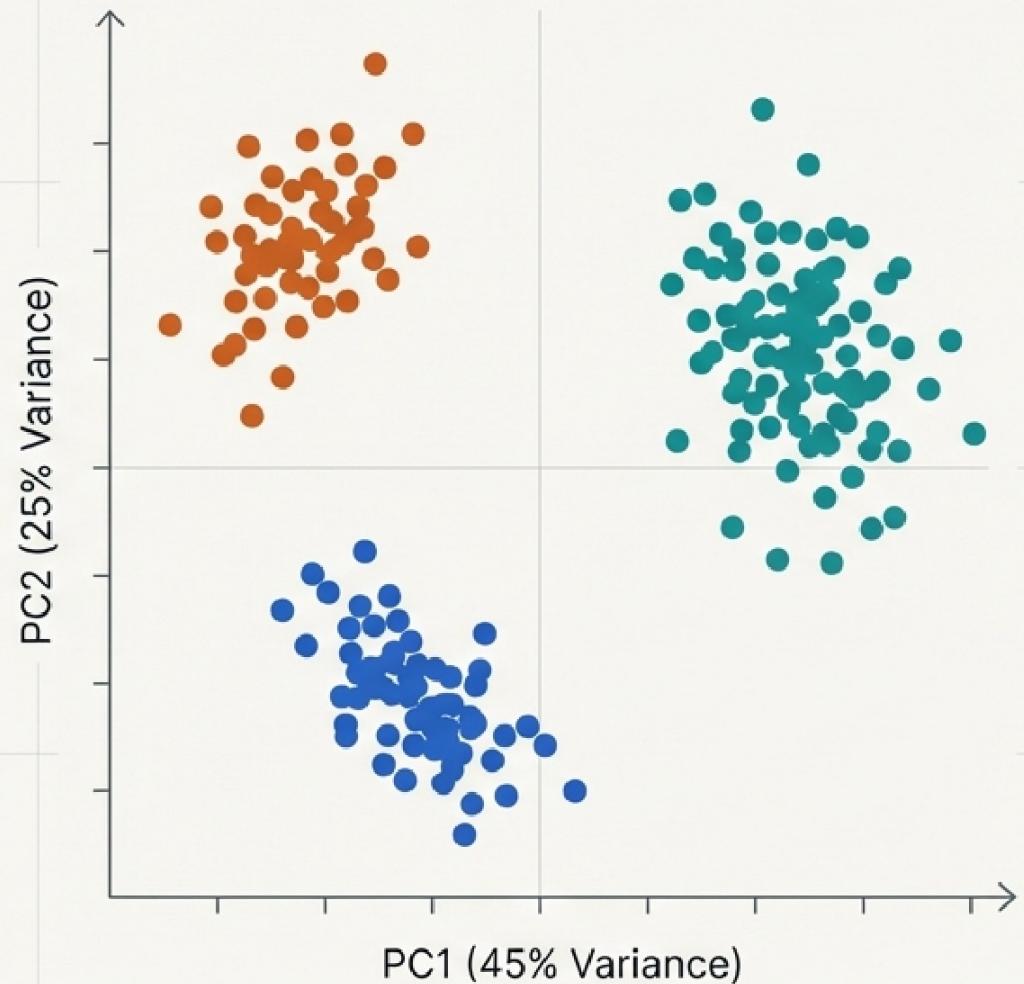
**Running**

# Python PCA Protocol

From Raw CSV to Visual Insight

A step-by-step workflow for analyzing  
antioxidant activity.

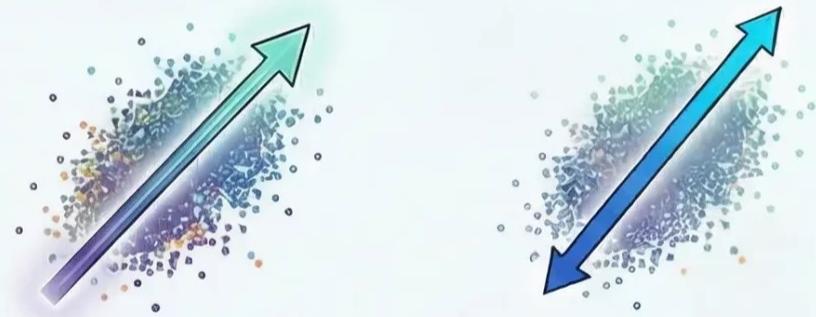
12.4,	135.2,	0.004,	89.1,	1.2,	24.5,	156.8,	0.022,	92.3
18.7,	148.1,	0.022,	92.3,	0.8,	148.1,	0.008,	85.6,	1.0
2.4,	135.6,	0.004,	89.1,	1.2,	148.1,	0.008,	85.6,	10.5
12.4,	135.0,	0.004,	89.1,	1.2,	24.5,	156.8,	0.022,	92.3
18.7,	148.1,	0.022,	92.3,	0.8,	148.1,	0.008,	85.6,	1.0
8.4,	135.6,	0.004,	92.3,	0.8,	148.1,	0.008,	85.6,	1.0
12.4,	135.2,	0.004,	89.1,	1.2,	24.5,	156.8,	0.022,	92.3
18.7,	148.8,	0.022,	92.3,	0.8,	148.1,	0.008,	85.6,	1.0
2.4,								12.4
12.4,								



# PCA Explained:

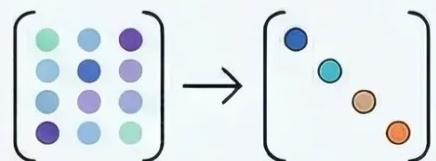
## From Complexity to Clarity

PCA simplifies complex, high-dimensional datasets by identifying principal components that capture the majority of the original data's variance, making it easier to visualize and efficient to process.



Eigenvectors point in a direction; Eigenvalues measure their importance.

Eigenvalues quantify the amount of variance (information) along each eigenvector's direction.

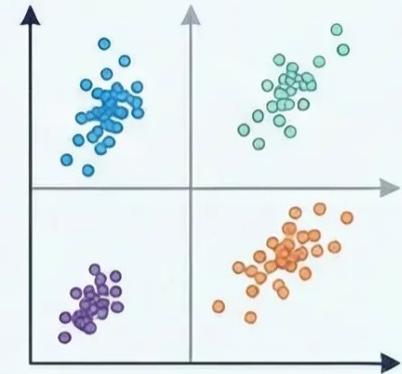
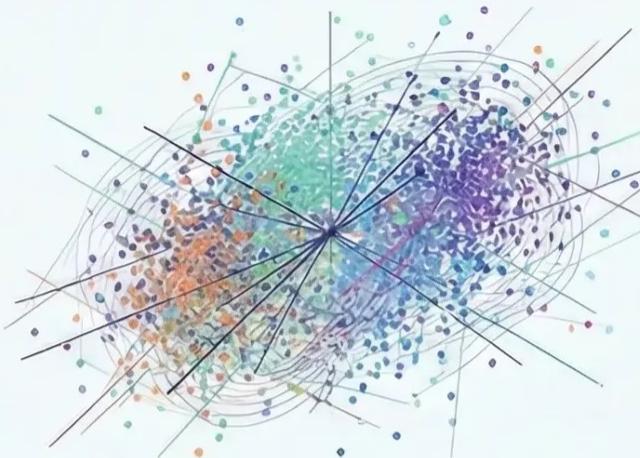


PCA's goal is to find the directions of maximum variance in data.

These directions of highest variance are called the "Principal Components".

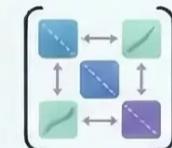
Principal Components are the eigenvectors of the data's covariance matrix.

Sorting them by their eigenvalues reveals which directions contain the most information.



### 1. STANDARDIZE DATA & CALCULATE COVARIANCE MATRIX

	PC 1	PC 2	PC 3	PC 4
1	0.62	0.43	0.39	0.25
2	0.23	0.82	2.12	0.39
3	0.22	0.56	2.16	0.68
4	0.27	1.27	0.86	0.45
5	0.25	0.93	0.56	0.38
6	0.49	0.82	0.88	0.50

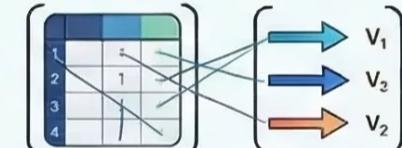


This prepares the data and shows how features vary with each other.

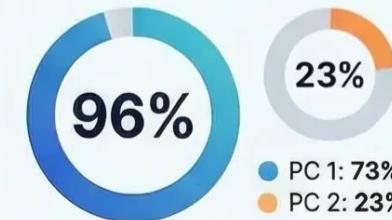
	PC 1	PC 2	PC 3	PC 4
1	0.62	0.43	0.39	0.25
2	0.23	0.82	2.12	0.39
3	0.22	0.56	2.16	0.68
4	0.27	1.27	0.86	0.45
5	0.25	0.93	0.56	0.38
6	0.49	0.82	0.88	0.50

This step extracts the Principal Components and their corresponding variance scores.

### 2. DECOMPOSE TO FIND EIGENVECTORS & EIGENVALUES

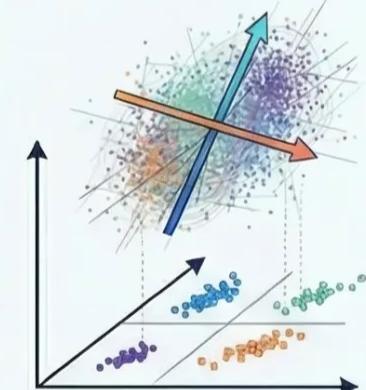


This step extracts the Principal Components and their corresponding variance scores.



First 2 PCs Combined: 96%  
(From a 4-dimensional dataset, e.g., Iris)

### 3. PROJECT DATA ONTO TOP 'K' COMPONENTS



The original high-dimensional data is transformed into a new, lower-dimensional space.

# PCA Example

# Do not download data

<https://github.com/saowalukrungchang/Intro-to-Python/blob/main/PCA.csv>



Sample name

The screenshot shows a GitHub repository page for 'saowalukrungchang / Intro-to-Python'. The 'main' branch is selected. A file named 'PCA.csv' is listed. The file has been updated 3 minutes ago. The CSV data starts with:

1	Sample	Area	DPPH	FRAP	ABTS	TPC	TFC	ORAC	Color_L	Moisture
2	1	a	81.4914246	704.9573347	67.04165152	150.7766204	99.74619168	910.019714	46.251347	6.171267153
3	2	a	79.5852071	706.3069546	67.62868205	157.749159	89.54367527	913.857924	43.2868489	6.340669692
4	3	a	81.94306561	719.6625694	65.78733005	144.3835782	103.4167659	872.7990111	42.858215	6.288200529
5	4	a	84.56908957	719.6625694	65.78733005	144.3835782	86.58889362	900.2901479	45.96494483	6.195417375
6	5	a	79.29753988	740.5636228	69.62505164	146.6631348	98.2214256	911.7228989	44.55307443	5.995419227
7	6	a	79.29753913	740.5636228	69.62505164	138.2918804	108.1427338	871.4259433	46.42800099	6.227690766
8	7	a	84.73763845	725.2215899	67.656597376	152.3689622	89.05678205	974.6309804	45.94647525	5.768152436
9	8	a	82.30230419	671.0082469	70.0070658	152.0884422	91.60221362	918.9533168	44.85434217	5.928954419
10	9	a	78.59157684	686.7953488	68.72327205	150.0409077	95.9791089	852.3478601	43.30641266	5.884390936
11	10	a	81.62768013	724.9215309	67.709760409	148.1233029	91.97914608	926.2621443	41.97030555	6.024562242
12	11	b	63.60974692	628.4616645	55.72279121	108.6770341	60.69601941	721.0127332	51.1069701	7.69438975
13	12	b	63.60281074	614.284207	58.07607313	116.6348874	70.41137785	791.4933841	53.71279759	6.439620442
14	13	b	65.72588681	607.1057929	54.92834792	117.2582893	63.2617772	806.3489232	52.42818749	7.205878057
15	14	b	59.26015927	602.4724076	58.12928731	113.5817818	72.84155458	727.1727073	49.50852244	6.516186239
16	15	b	59.8252465	573.0369502	49.760508979	118.7097143	64.48345459	798.6350452	52.34636185	6.85842044
17	16	b	63.31313741	592.0038948	66.64380501	123.2324069	78.29960643	776.5112371	52.77069476	7.326685179
18	17	b	61.96150664	598.4840307	65.17409414	135.0894872	65.30048025	792.8824064	50.23228513	7.019284006
19	18	b	65.942742	636.4280557	64.4019853	121.3966225	68.0876309	835.8717193	52.30745021	6.676676667
20	19	b	62.27592777	618.5904672	55.18352150	122.0604031	74.8811033	750.1844754	52.11641744	6.785408887
21	20	b	60.7630889	565.9239961	61.02486217	119.4044327	82.0148141	729.8605534	49.7140594	7.203879325
22	21	c	49.39694631	488.1020982	41.56065622	74.64993028	51.36475961	84.4194228	60.71557472	7.78089001
23	22	c	44.3226711	470.372943	42.71422514	89.787989	57.84285663	587.3675886	61.12156905	8.064937577
24	23	c	45.20258401	463.07695	44.95678809	90.48184168	40.35610059	610.9169310	62.16610249	8.019371552
25	24	c	40.72575544	495.2919072	40.96345956	109.7059369	51.10780315	633.646079	62.1076041	7.804519896
26	25	c	43.366865183	505.7499881	40.3801279	88.46111220	51.55929677	631.067632	57.24468126	8.843183227
27	26	c	45.33276777	503.282003	40.99648591	92.41237874	54.89093723	653.08733	58.12434992	8.190175707
28	27	c	41.64701927	459.0195619	43.83080424	89.72230584	42.57829573	620.5200757	61.03007053	7.392457224
29	28	c	46.12709406	472.2696906	42.65750222	80.6560757	42.07726032	678.1413631	61.0275719	8.055936294
30	29	c	43.19808393	498.2815658	40.94047959	99.14268252	53.31648939	609.4137267	61.03009537	7.901464061
31	30	c	44.12491875	504.3080282	43.02653407	90.01546429	51.78190004	728.6067887	677.0546298	8.25522

Column1 = Sample name  
Column2 = Sampling Area

Column 3 onward are dependent variable

# AI (Chat, Gemini, Copilot,.....)

1. Where is your data?
2. What is your data? Please describe it.
3. What do you want to analyze?

## GITHUB

The dataset is available at <https://github.com/saowalukrungchang/Intro-to-Python/blob/main/PCA.csv>

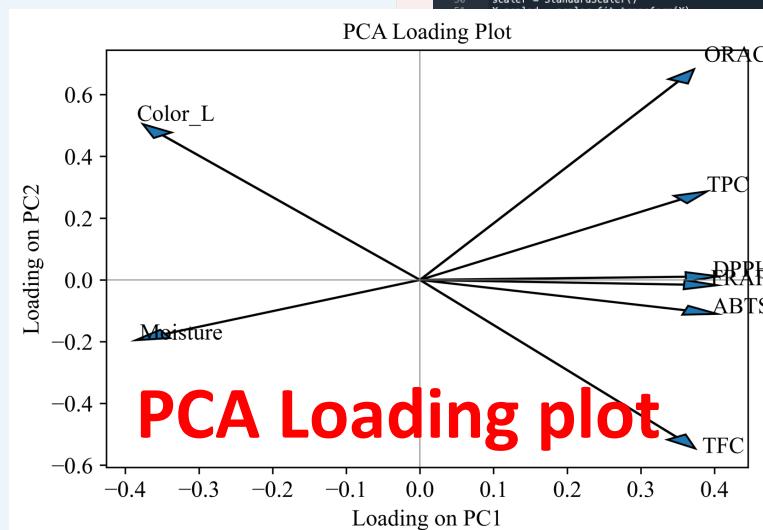
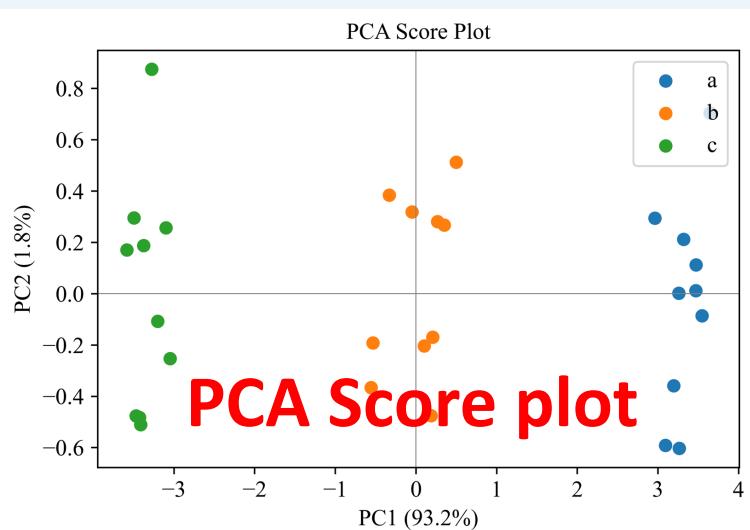
Column 1 contains the sample name.

Column 2 contains the sampling area.

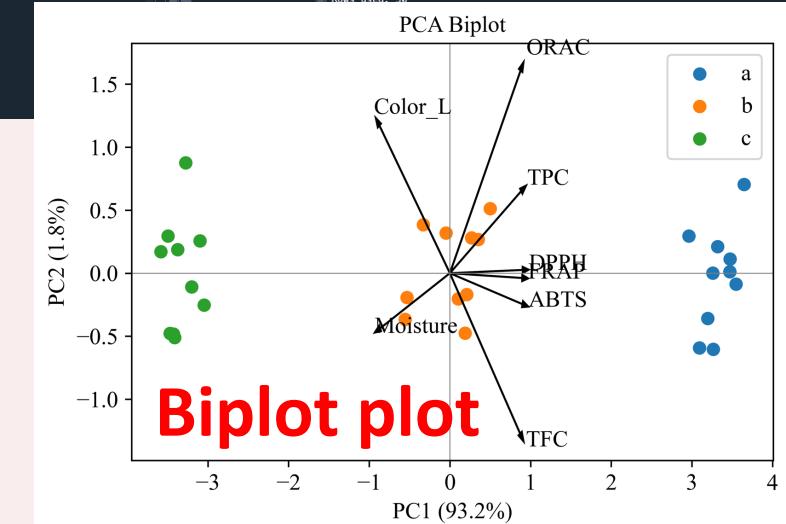
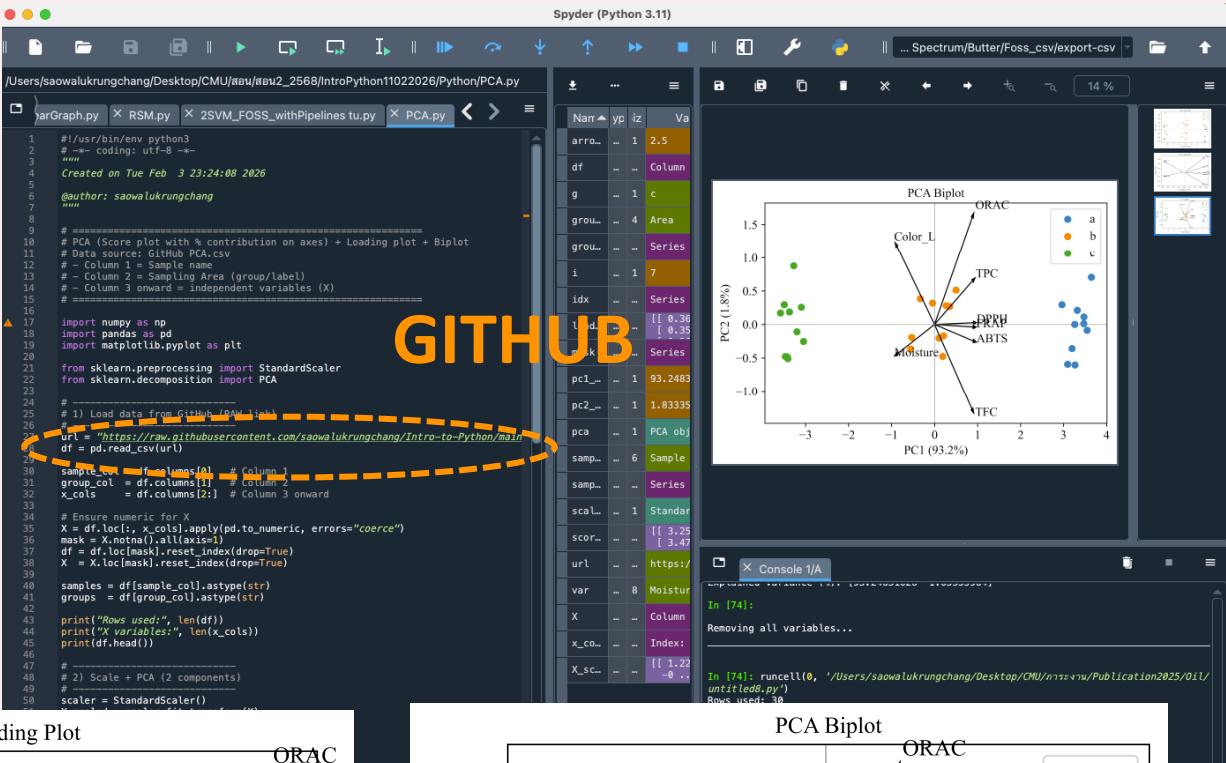
Columns from column 3 onward are dependent variables.

Please write Python code to plot:

- (1) a PCA score plot showing the contribution (%) on the X and Y axes,
- (2) a loading plot, and
- (3) a biplot.



## Spyder



# ANOVA Example

<https://github.com/saowalukrungchang/Intro-to-Python/blob/main/ANOVA.csv>



The screenshot shows a GitHub repository page for 'saowalukrungchang / Intro-to-Python'. The 'main' branch is selected, and the file 'ANOVA.csv' is open. The file contains 32 rows of data with four columns: 'Temp', 'Time', 'Pretreatment', and 'Antioxidant'. The first few rows of data are as follows:

	Temp	Time	Pretreatment	Antioxidant
1	Low	Short	No	73.66317300825904
2	Low	Short	No	72.43055794879682
3	Low	Short	No	67.90919941577746
4	Low	Short	Mild	72.98323230014296
5	Low	Short	Mild	74.24267194778096
6	Low	Short	Mild	71.5598288785622
7	Low	Short	Yes	78.53102317138423
8	Low	Short	Yes	78.21709705142995
9	Low	Short	Yes	78.00858286188006
10	Medium	No	No	66.65079957881412
11	Medium	No	No	67.8660523799072
12	Medium	No	No	69.40607474782443
13	Medium	Mild	No	67.06968605896847
14	Medium	Mild	No	71.0565481559054
15	Medium	Mild	No	69.45726026024931
16	Medium	Mild	Yes	74.890275226607
17	Medium	Mild	Yes	71.7267956763379
18	Medium	Mild	Yes	74.27027375689727
19	Long	Short	No	65.96907400367317
20	Long	Short	No	60.84039022842945
21	Long	Short	No	59.04454343868419
22	Long	Short	Mild	61.513256408202194
23	Long	Short	Mild	66.5321403280011
24	Long	Short	Mild	69.7699346614222
25	Long	Short	Mild	72.24738250681888
26	Long	Short	Yes	73.34524442661565
27	Long	Short	Yes	70.1982984316705
28	Medium	Short	No	69.79599275418715
29	Medium	Short	No	66.45750402406844
30	Medium	Short	No	68.22640836971928
31	Medium	Short	Mild	68.46536562265287
32	Medium	Short	Mild	68.46536562265287

Column 1-3  
independent  
variables

Column 4  
dependent  
variables

Please write Python code to perform **ANOVA analysis** using the data available at <https://github.com/saowalukrungchang/Intro-to-Python/blob/main/ANOVA.csv>.

The experiment follows a  $3 \times 3$  factorial design with three replicates.  
Column1-3 are independent variables  
Column 4 is the dependent variable

# ANOVA Example

The image shows a Jupyter Notebook environment with three main components:

- Code Cell (Left):** Contains Python code for performing a 3-factor factorial ANOVA. It imports pandas and statsmodels, loads data from a CSV file, defines columns, fits a model, and prints the ANOVA table.
- Jupyter Notebook (Middle):** Displays an ANOVA table (Type II) with columns: sum\_sq, df, F, and PR(>F). The table includes terms for Temperature, Time, Pretreatment, and their interactions.
- Terminal (Right):** Shows the output of the ANOVA model summary, including the OLS Regression Results table with coefficients, standard errors, t-values, P>|t|, and confidence intervals.

**Result**

```
== ANOVA table (Type II) ==
sum_sq    df      F    PR(>F)
C(Temp)    2148.564472  2.0  259.440111  2.027254e-28
C(Time)    1291.468286  2.0  155.945367  3.665586e-23
C(Pretreatment) 315.277241  2.0  38.069866  4.848015e-11
C(Temp):C(Time) 98.908342  4.0  5.971613  4.796827e-04
C(Temp):C(Pretreatment) 58.574280  4.0  3.536435  1.234917e-02
C(Time):C(Pretreatment) 14.133272  4.0  0.853299  4.979045e-01
C(Temp):C(Time):C(Pretreatment) 21.296799  8.0  0.642899  7.382131e-01
Residual   223.601665  54.0  NaN   NaN

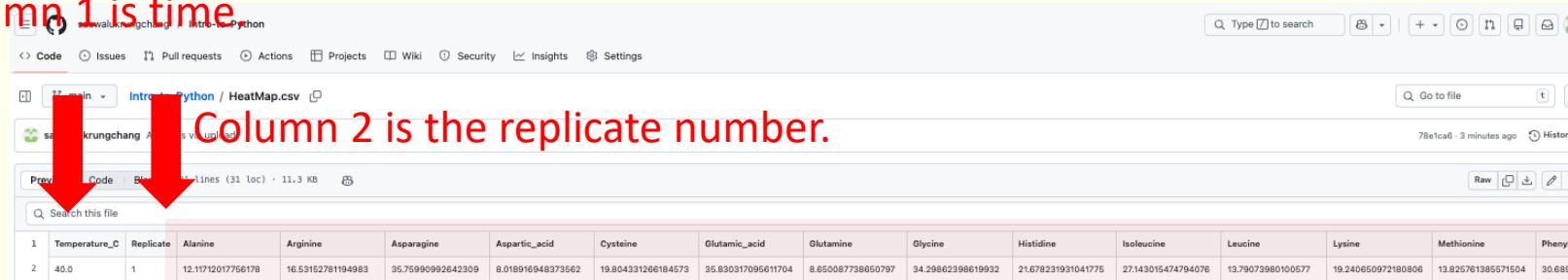
== Model summary ==
OLS Regression Results
=====
Dep. Variable: Antioxidant R-squared: 0.946
Model: OLS Adj. R-squared: 0.921
Method: Least Squares F-statistic: 36.67
Date: Tue, 03 Feb 2026 Prob (F-statistic): 1.01e-25
Time: 23:43:45 Log-Likelihood: -156.06
No. Observations: 81 AIC: 366.1
Df Residuals: 54 BIC: 430.8
Df Model: 26
Covariance Type: nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	51.0533	1.175	43.455	0.000	48.698	53.409
C(Temp)[T.Low]	14.5518	1.661	8.758	0.000	11.221	17.883
C(Temp)[T.Medium]	10.2079	1.661	6.144	0.000	6.877	13.539
C(Time)[T.Medium]	7.4568	1.661	4.488	0.000	4.126	10.788
C(Time)[T.Short]	12.2421	1.661	7.368	0.000	8.911	15.573
C(Pretreatment)[T.No]	-2.3279	1.661	-1.401	0.167	-5.659	1.003
C(Pretreatment)[T.Yes]	1.0413	1.661	0.627	0.533	-2.290	4.372
C(Temp)[T.Low]:C(Time)[T.Medium]	-3.8673	2.350	-1.646	0.106	-8.578	0.843
C(Temp)[T.Medium]:C(Time)[T.Medium]	-3.9998	2.350	-1.702	0.094	-8.711	0.711
C(Temp)[T.Low]:C(Time)[T.Short]	-4.9186	2.350	-2.093	0.041	-9.629	-0.208
C(Temp)[T.Medium]:C(Time)[T.Short]	-4.9592	2.350	-2.111	0.039	-9.670	-0.248
C(Temp)[T.Low]:C(Pretreatment)[T.No]	-1.3258	2.350	-0.564	0.575	-6.037	3.385
C(Temp)[T.Medium]:C(Pretreatment)[T.No]	-0.0931	2.350	-0.040	0.969	-4.804	4.618
C(Temp)[T.Low]:C(Pretreatment)[T.Yes]	5.2839	2.350	2.249	0.029	0.573	9.995
C(Temp)[T.Medium]:C(Pretreatment)[T.Yes]	1.0024	2.350	0.427	0.671	-3.708	5.713
C(Time)[T.Medium]:C(Pretreatment)[T.No]	-0.0543	2.350	-0.023	0.982	-4.765	4.657
C(Time)[T.Short]:C(Pretreatment)[T.No]	2.3263	2.350	0.990	0.327	-2.385	7.037
C(Time)[T.Medium]:C(Pretreatment)[T.Yes]	1.8875	2.350	0.803	0.425	-2.823	6.598
C(Time)[T.Short]:C(Pretreatment)[T.Yes]	0.9266	2.350	0.394	0.695	-3.784	5.637
C(Temp)[T.Low]:C(Time)[T.Medium]:C(Pretreatment)[T.No]	2.4878	3.323	0.749	0.457	-4.174	9.150
C(Temp)[T.Medium]:C(Time)[T.Medium]:C(Pretreatment)[T.No]	1.6364	3.323	0.492	0.624	-5.026	8.299
C(Temp)[T.Low]:C(Time)[T.Short]:C(Pretreatment)[T.No]	-0.2668	3.323	-0.080	0.936	-6.929	6.395
C(Temp)[T.Medium]:C(Time)[T.Short]:C(Pretreatment)[T.No]	-0.2894	3.323	-0.087	0.931	-6.951	6.373
C(Temp)[T.Low]:C(Time)[T.Medium]:C(Pretreatment)[T.Yes]	2.7791	3.323	0.127	0.261	-2.440	2.884

# Heat Map Example

<https://github.com/saowalukrungchang/Intro-to-Python/blob/main/HeatMap.csv>

Column 1 is time.



1	Temperature_C	Replicate	Alanine	Arginine	Asparagine	Aspartic_acid	Cysteine	Glutamic_acid	Glutamine	Glycine	Histidine	Isoleucine	Leucine	Lysine	Methionine	Phenyl		
2	40.0	1	12.11712017756178	10.53152781194983	35.75990992642309	8.018916948373562	19.804331266184573	35.830317095611704	8.650087738650797	34.29862398619932	21.678231931014775	27.14301547494076	13.7907398010577	19.240650972180806	13.825761385571504	30.689		
3	40.0	2	11.581197942739134	17.47293817035875	34.301889756461414	7.927127687781889	19.50841512107171	35.94259282802247	12.0113875141617	32.02513660767385	22.386274355628177	29.840646107893415	13.394307336					
4	40.0	3	12.31780972510202	11.88990992642309	37.950751073895	18.45814605242307	19.009816948373562	35.830317095611704	8.650087738650797	34.29862398619932	21.678231931014775	27.14301547494076	13.7907398010577	19.240650972180806	13.825761385571504	30.689		
5	45.0	1	11.473970748078704	18.017468390474697	33.89246914902674	9.781106380909111	19.7441590126141	36.98634203141515	11.58277418569654	33.0240370894621	22.61716754690167	29.4758048794202	13.482064068					
6	45.0	2	12.172786222295793	19.20866824048553	33.42608033762221	8.481420986402075	20.089706400520197	36.390391506298855	10.63339726630085	33.04314658648872	22.031906986818296	29.047536380119246	13.872766506					
7	45.0	3	11.283441485275402	18.10978145774498	34.98781912390079	9.735938470384077	20.864985217380636	37.520850416793034	10.380631170386824	34.43249516178465	22.247347151929808	29.164494800560643	14.244574065					
8	50.0	1	12.276679325994194	17.84465156328221	34.37461611285254	11.73889927520184	21.0218010837857	39.22541662410075	10.0122768083918337	35.6781903796246	21.39143710335246	28.462617232125137	13.87985620					
9	50.0	2	12.9410504856716	20.02750095436308	32.62502930942286	11.797934583892177	20.6236367376993633	37.94016019365817	9.299560423496757	34.6877019271048	21.128877560468367	28.307976750834395	14.417040316					
10	50.0	3	11.54548747025433	18.844700287149219	34.1898848225944	10.711501223933439	22.37912273553552	36.967317047341545	10.47332195086355	36.09682250117626	22.904349380561	28.706080591819572	13.00436800					
11	55.0	1	12.9327537102029	21.047954513767629	33.29805062901699	14.068718502429333	23.41549424937727	40.61419315292501	11.26768604668155	38.25876012089751	20.83712095451923	28.674327808007944	14.7936398					
12	55.0	2	10.97299693596284	19.923254831271	32.621928717872535	15.52129809148601	21.184023174514484	40.0488618054573374	10.163091650365103	21.125028197051197	20.010050202295474	14.733259498						
13	55.0	3	12.827180448580705	21.055871530192842	32.820123945639516	14.78546221717856	22.54087806116382	41.77035609533801	10.6234381992009613	37.7722973789675	20.231631776741224	29.3102816005261542	15.41618344					
14	60.0	1	14.019056952904666	22.32495005590528	34.295179680267804	15.316677238477191	25.130783242753136	41.97986623746514	10.138243904807198	39.64145161024795	21.33128211907783	30.4229519674439	14.462340795					
15	60.0	2	14.824756468807047	21.825644446567101	34.678521922794834	14.927052791196906	23.673471144769906	40.59250554655861	10.626539844286113	39.55760450567573	19.9526451202372	29.0779421024679	15.68935649					
16	60.0	3	12.1047004512199669	21.86379964351844	34.05222623483812	15.4088685226601962	22.717064199491717	42.06231295296423	11.137518185675764	39.23251863677556	21.141842520614553	30.405059316914162	16.0681772111					
17	65.0	1	16.28443451919059	22.75955053481408	33.32096591512638	14.883211727172784	25.45425876236534	42.01224857893852	10.83262217732379	41.099247181085204	22.68463766156701	34.69797201991829	18.605192162					
18	65.0	2	18.04210664193032	23.180716136890553	34.483999375818242	14.880357770146006	24.31091300840677	44.2655486326793	10.81832419913894	41.2365024518621	21.440825039074827	33.985878447257036	19.741391551					
19	65.0	3	15.38637369207306	21.2819980801812	33.64137419718972	14.889173245844889	23.5532442125653	41.8184909578875	10.06701664536029	40.58293993953953	21.76819004944606	33.616440276133	19.904096910					
20	70.0	1	19.646475632366565	22.55473485216915	35.566424727496521	15.068329489200964	24.916566349607198	41.40106581629975	11.43973449109328	41.128493271268816	22.581611147939355	37.448713758393998	21.105851437					
21	70.0	2	20.00238093206432	22.6096591999050763	34.33841362112429	16.195208739078087	23.9324795616004836	44.732296640210656	15.1386361520591	41.42873671078706	20.46446587741745	36.166646508655817	22.127466281					
22	70.0	3	19.848648680197956	22.47687423107609	33.53364210044094	15.67938079654219	24.242632693006087	41.99241059374615	10.1152049441809	39.0120106019439	22.106729382957027	36.86372723237589	22.98027869					
23	75.0	1	20.29903077531883	18.996586876866045	33.9168830029781994	15.131315792362991	21.72483448070613	43.19404928918121	10.3472794045156	39.105203325321386	20.28279732454931	37.733548715809849	23.57893479					
24	75.0	2	22.36196478118044	23.386000280905371	15.36913918486501	20.2420879273858	42.43462484451178	11.6368788209305	41.23485658743435	22.427962957730113	37.78616283569425	24.31595768						
25	75.0	3	22.192177583836344	19.354989500495925	34.97130834747488	21.282678384590079	22.93107233559295	39.3318410722886	22.81720607237762	39.72607619124226	24.46883839							
26	80.0	1	20.50919184515647	15.202833237510737	34.9546160061421	16.730365847990581	17.127518148688997	43.82331233237564	12.268639760753407	41.150255376756948	22.275842903049876	40.212404037086214	24.622266586					
27	80.0	2	20.83457139449846	16.09884378788884	34.44730916087762	15.886119036993949	17.92266421793786	43.56889138241224	10.63629649874799	40.74453114677714	23.65565490154816	39.47260804930364	24.65197599					
28	80.0	3	21.515269815939183	14.42792395225864	35.272399079645658	15.91994306891545	16.97654718522124	42.40824530361948	11.30848347227868	40.508049161483912	22.32727668478278	38.3031295564731	23.510880762					
29	85.0	1	21.089857878288675	15.25562355404374	35.7949955505071015	14.214388970274515	15.034905844830156	40.9449513730428	11.90042878528147	39.807543990527584	23.678849961644246	38.312952666471276	25.160614462					
30	85.0	2	22.768010442440663	16.03162990993688	36.101462319395567	16.88543905160958	18.828398570678645	42.9284129821515	11.2554727575331	40.55646105706713	23.620347480860704	38.11513590901218	23.46868881					
31	85.0	3	21.703570921677975	14.744279804297797	34.958208642183929	15.8087713777909	15.8216707415107019	42.70885120745555	10.481154534971	41.20587488244119	24.1298075198462	37.94727703484077	23.302731215207474	11.6192770314706515	42.298			



Please write Python code to plot a heatmap using the data at  
<https://github.com/saowalukrungchang/Intro-to-Python/blob/main/HeatMap.csv>.

Column 1 is time.  
Column 2 is the replicate number.  
Columns from column 3 onward are amino acids.

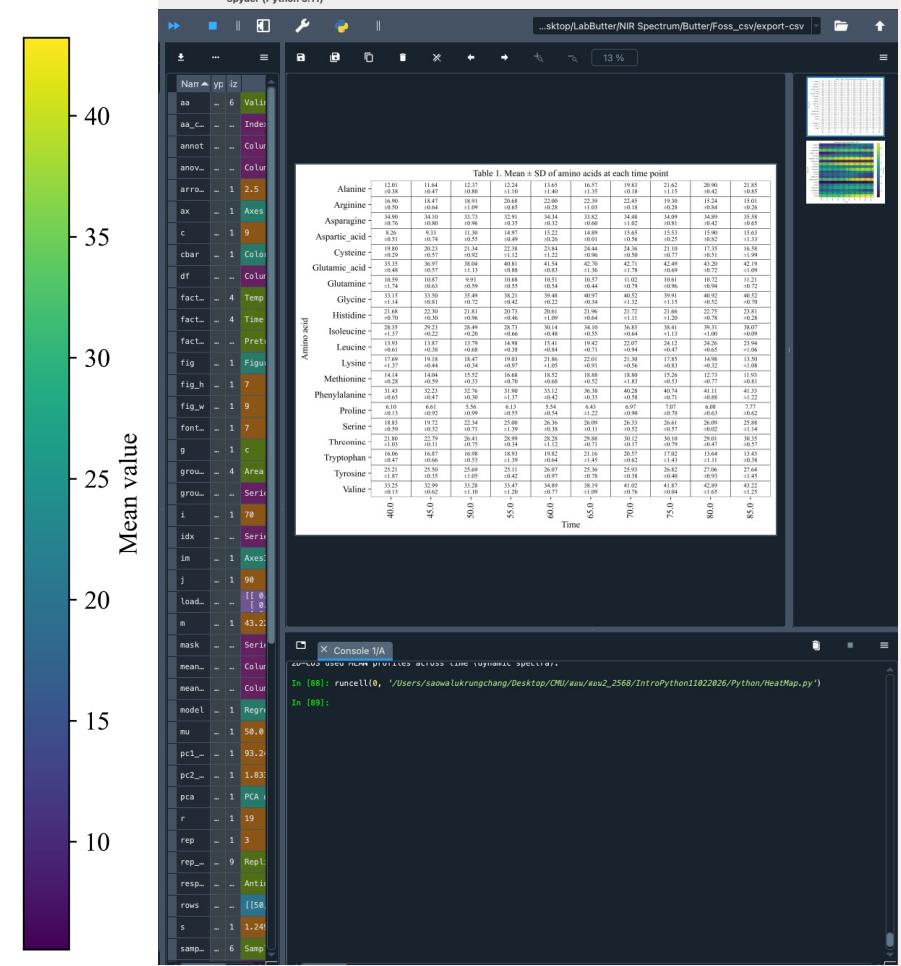
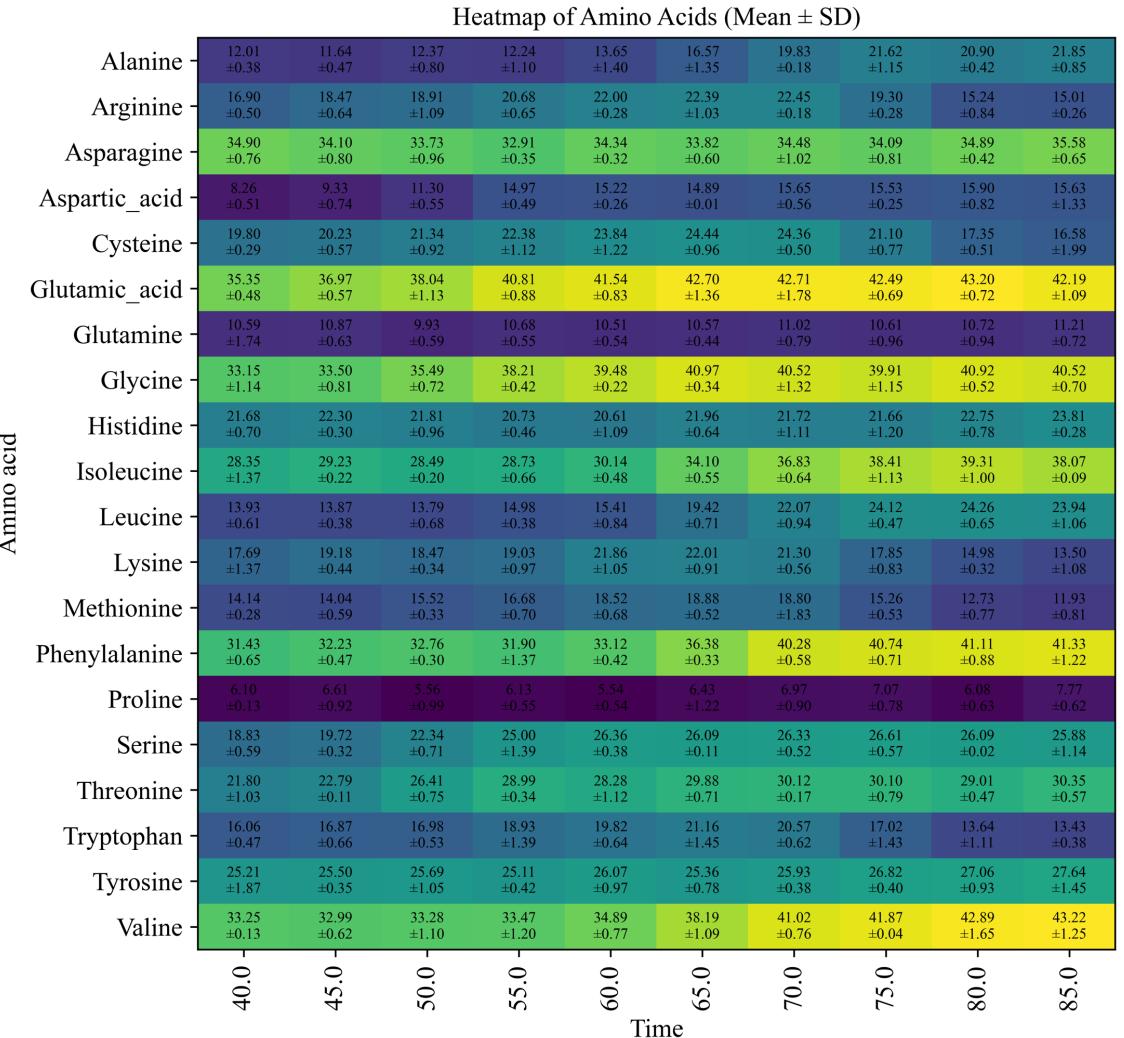
I would like a heatmap where the x-axis is time and the y-axis is amino acids, and each cell shows the mean and SD.

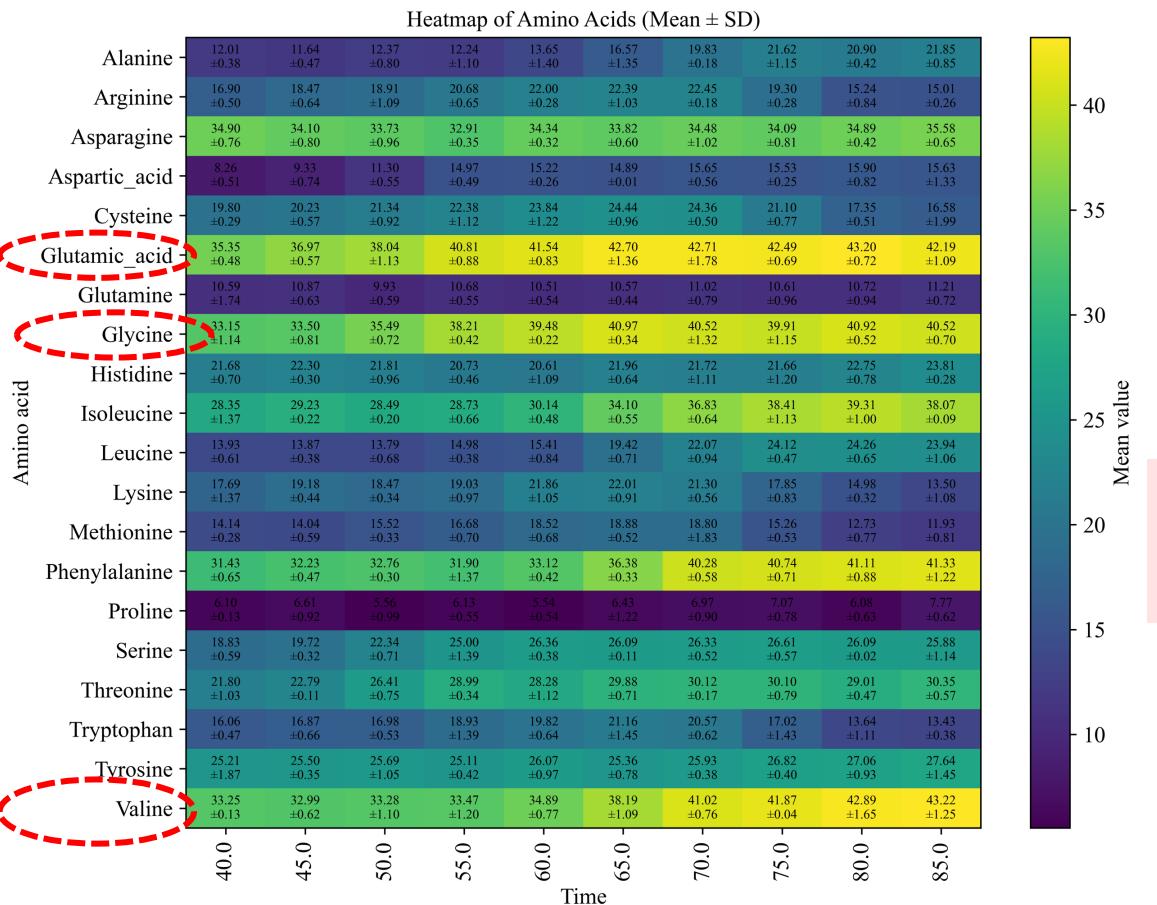
# Heat Map Examples

Please write Python the data at  
<https://github.com/Python/blob/main>

Column 1 is time.  
 Column 2 is the rep.  
 Columns from colu

I would like a heatr  
 and the y-axis is an  
 the mean and SD.





• This figure is a heatmap showing the mean ± SD of amino acids over time.

• The x-axis represents time (40–85).

• The y-axis represents 20 individual amino acids.

• Warmer colors (yellow) indicate higher values, while cooler colors (purple) indicate lower values.

• Some amino acids show a continuous increase with time (e.g., glutamic acid, glycine, valine).

• The generally low SD values indicate good measurement reproducibility.

• The heatmap effectively illustrates overall temporal trends but does not reveal inter-amino-acid relationships.

# Heatmaps describe how amino acid levels change.

# Heat Map Example

Please write Python code to plot a heatmap using the data at

<https://github.com/saowalukrungchang/Intro-to-Python/blob/main/HeatMap.csv>.

Column 1 is time.

Column 2 is the replicate number.

Columns from column 3 onward are amino acids.

I would like a heatmap where the x-axis is time and the y-axis is amino acids, and each cell shows the mean and SD. **Synchronous**

# Synchronous correlation map

# Spyder

Please write Python code to plot a heatmap using the data at

<https://github.com/saowalukrungchang/Intro-to-Python/blob/main/HeatMap.csv>.

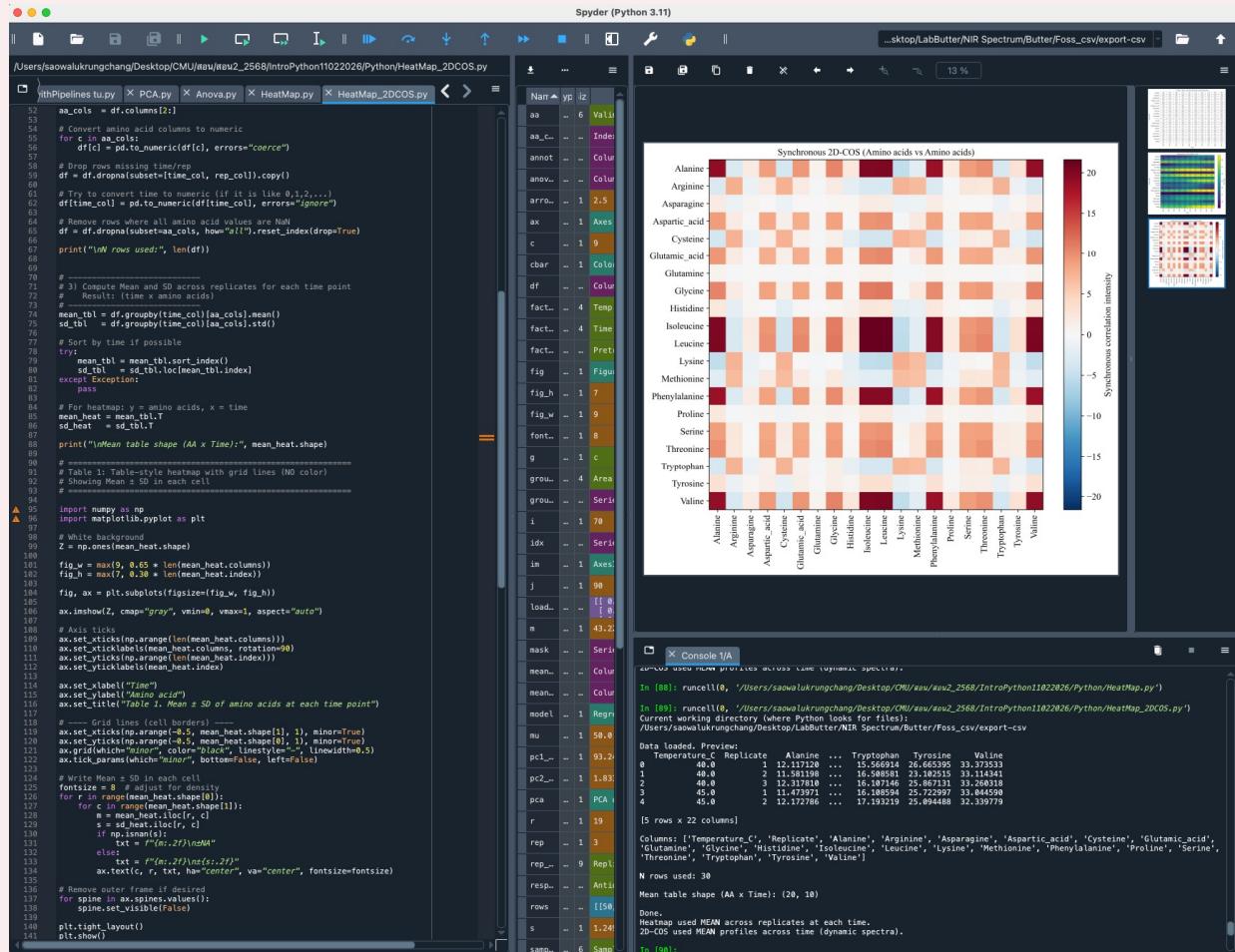
Column 1 is time.

Column 2 is the replicate number.

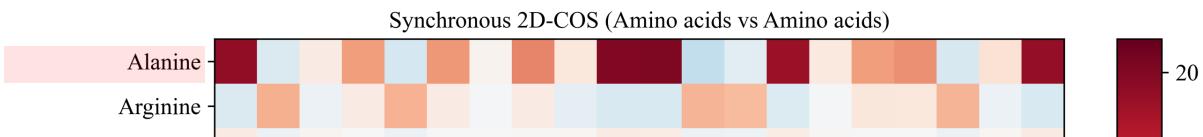
Columns from column 3 onward are amino acids.

I would like a heatmap where the x-axis is time and the y-axis is amino acids, and each cell shows the mean and SD. **Synchronous**

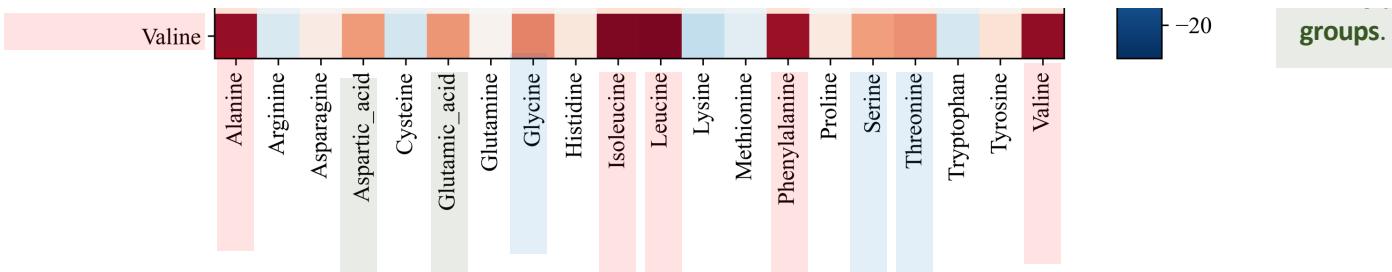
# Synchronous correlation map



- Heatmaps describe **how** amino acid levels change.
- **Synchronous correlation map reveals which amino acids change together.**



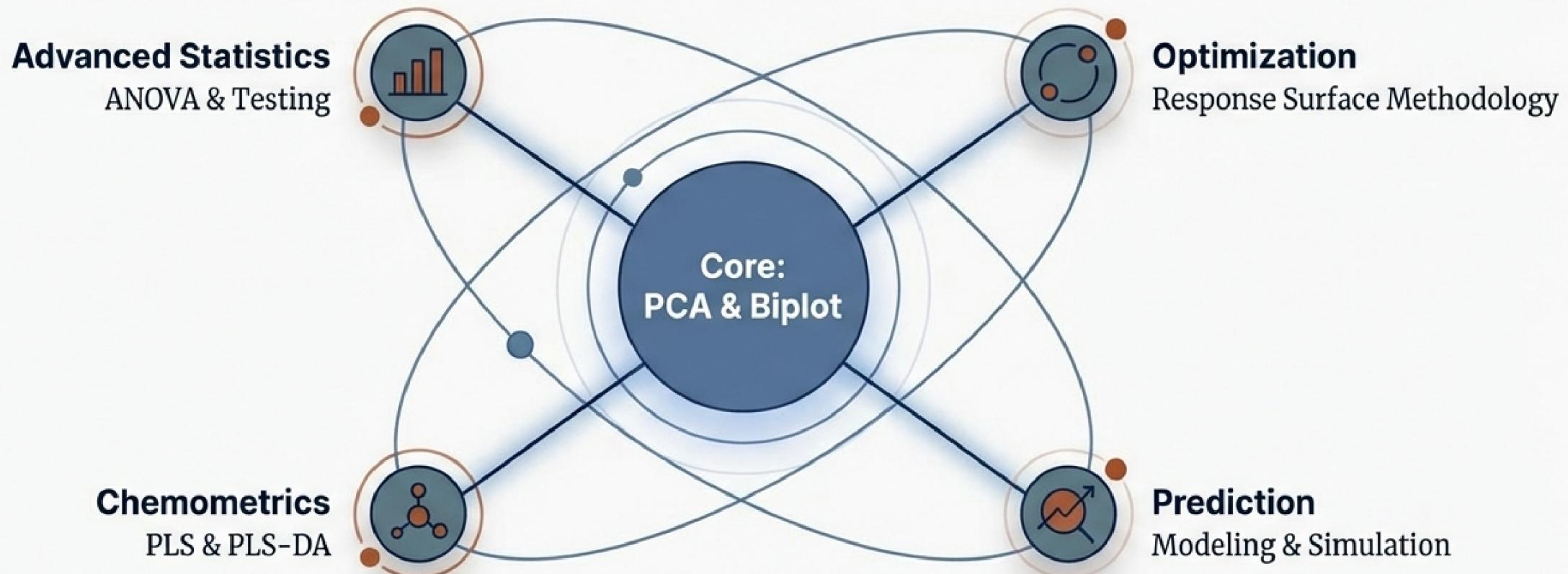
This example demonstrates that AI and Python can analyze data and generate graphs even for users with no prior programming experience. However, the resulting visualizations are only meaningful when the underlying logic and analytical principles are truly understood.



groups.

# Python Can Take You Much Further

Multidimensional analysis is powerful, but it is only one component of the research stack.  
The ecosystem supports the entire lifecycle of scientific inquiry.



# The integration of AI and Python has fundamentally altered the pace of discovery.

We are witnessing a shift in the fundamental mechanics of data science. The combination of Python's versatility and AI's processing capabilities has revolutionized how we handle data.



Faster

Accelerated insight.



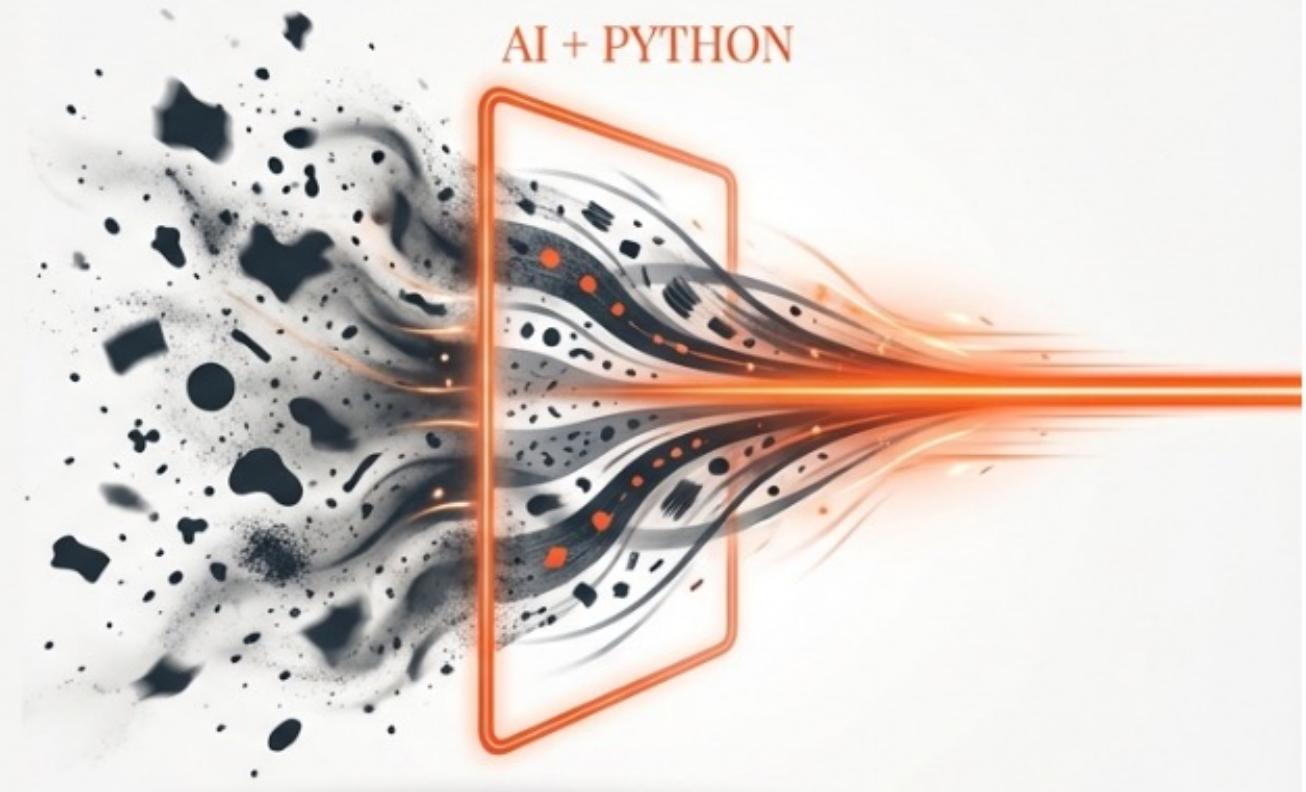
More Accessible

lowered barriers.



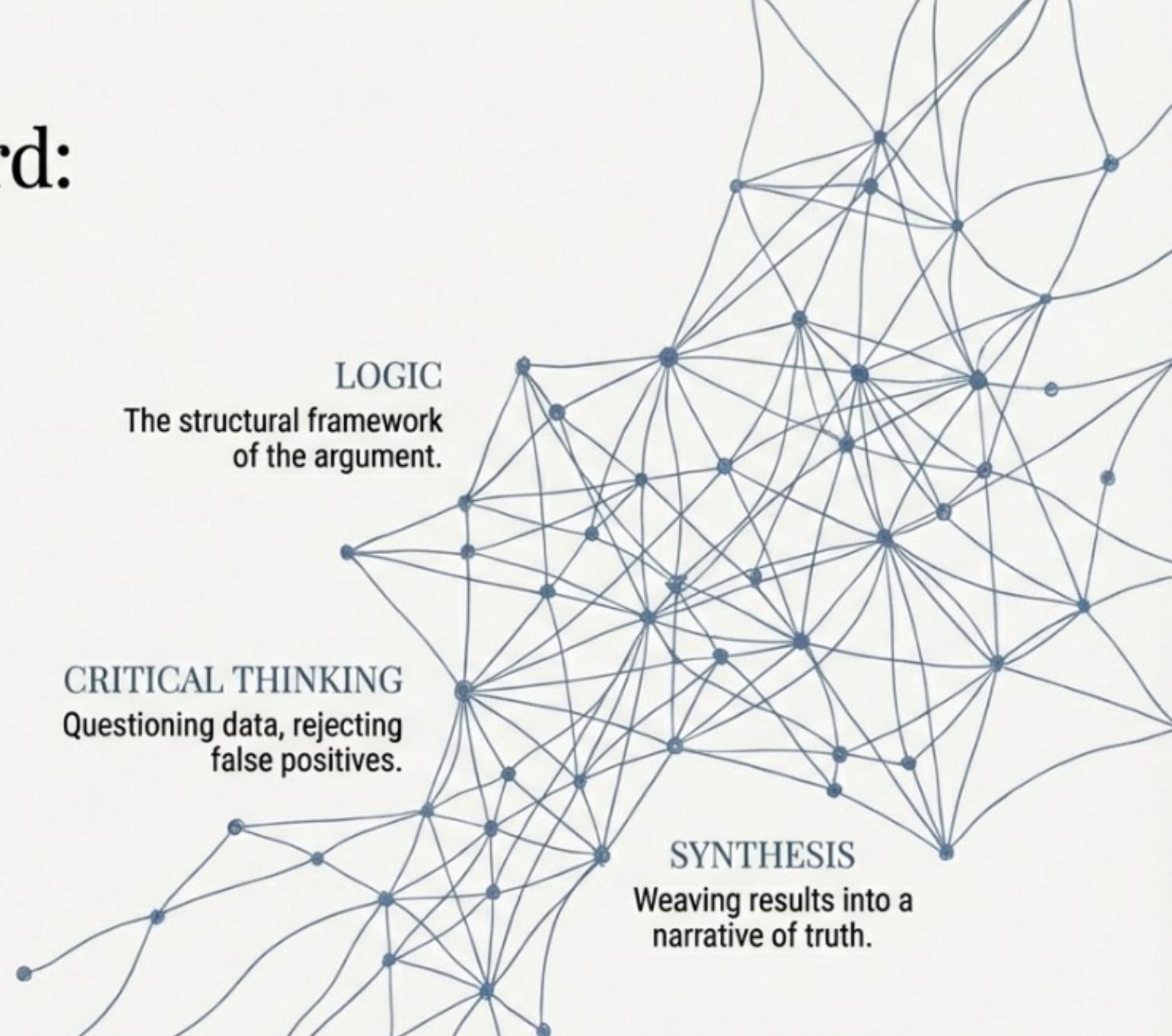
More Powerful

Massive scale.



# The Human Standard: The unchanging requirements of scientific inquiry.

Automation can process data, but it cannot frame a hypothesis or judge the weight of a contradiction. The core of research is not calculation, but cognition.



# A Crucial Reminder

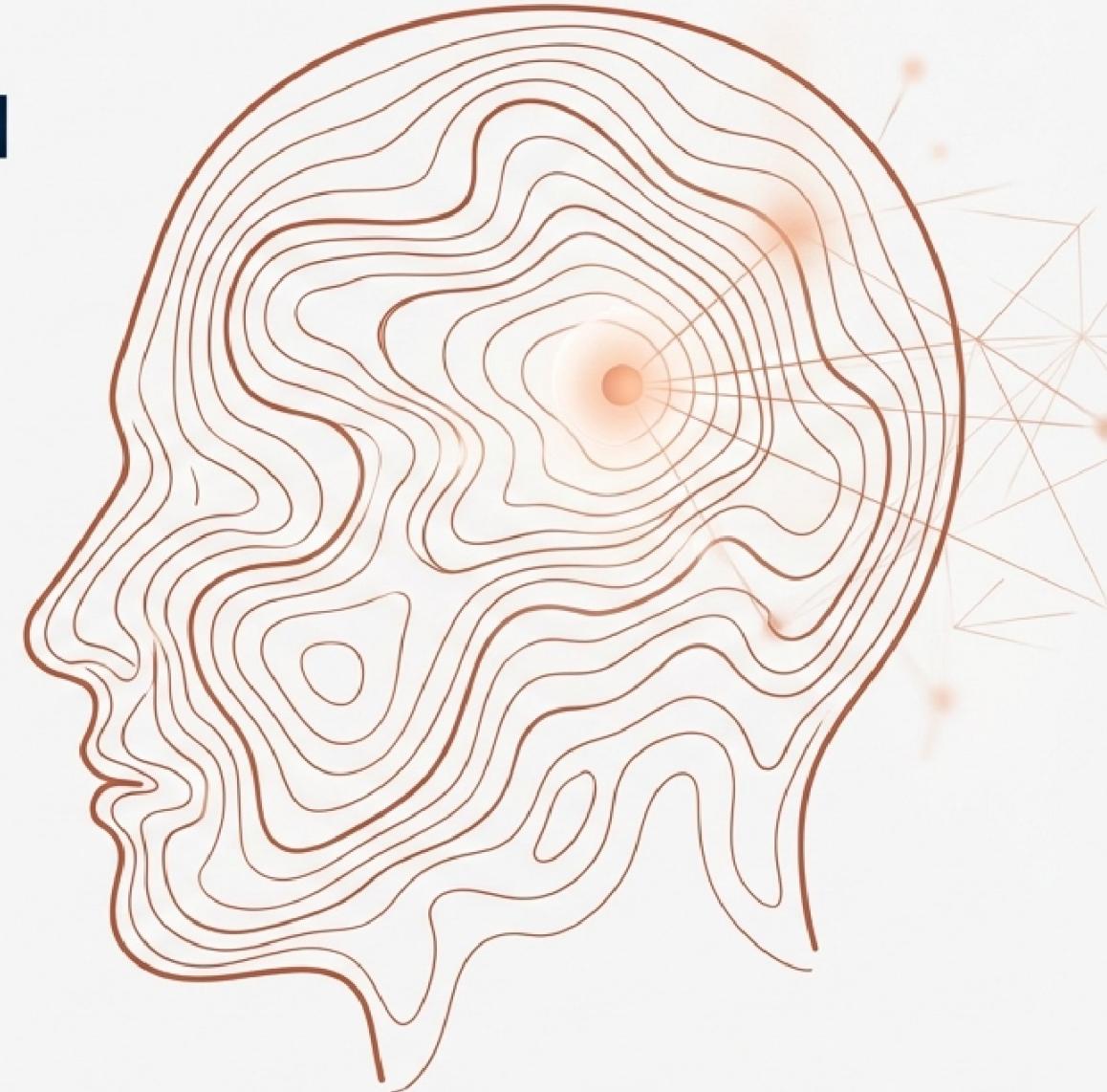
---

*The most critical variable in your research is  
not the Python script.  
It is not the Artificial Intelligence.*

# The Foundation is You

- ✓ How deeply do you understand your own study?
- ✓ What specific questions are you trying to answer?
- ✓ What exact effects are you looking to discover?
- ✓ Does your data actually possess the capacity to answer those questions?

**A tool is only as effective as the mind wielding it.**



# The Synergy of Human Intellect and Automated Analytics

**Human Cognition**  
Thinking, Hypothesis,  
Strategy

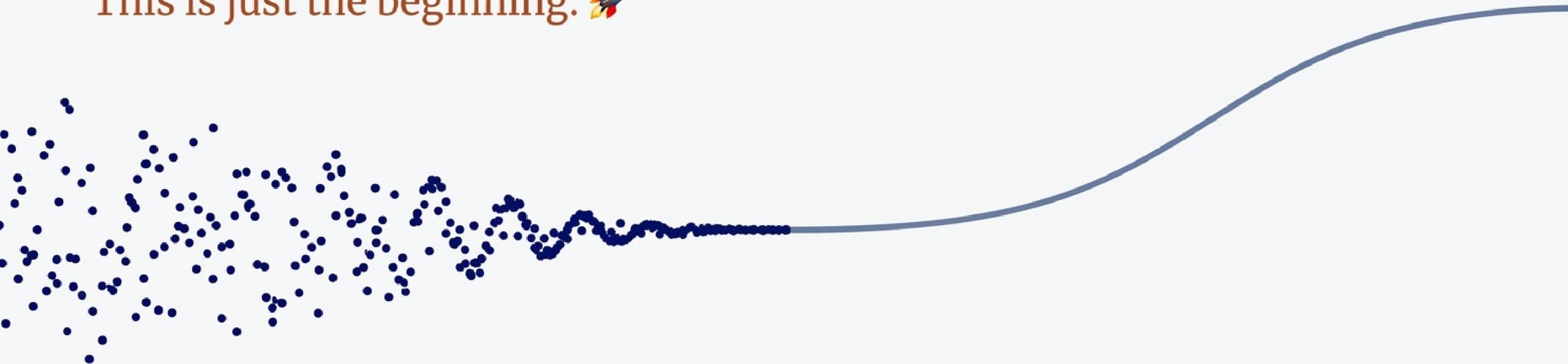


**Computational Power**  
Calculation, Pattern  
Finding, Speed

DRIVEN BY REASON. POWERED BY CODE.

# Python for Research: Beyond Multidimensional Analysis

This is just the beginning. 



# Let's Continue the Journey

Are you ready to apply these methods to your real-world data?

Are you ready to push your research boundaries beyond today's session?

## Contact Card

Discuss your specific data challenges.

Instructor Contact: [Saowaluk.run@cmu.ac.th](mailto:Saowaluk.run@cmu.ac.th)

