

# Part 2 — Workshop 3: Plotting

TECH2: Introduction to Programming, Data, and Information Technology

Richard Foltyn  
NHH Norwegian School of Economics

October 10, 2025

## Exercise 1: Plotting functions

Consider the following function in one variable  $x$ ,

$$f(x) = \sin(20x) + 5x^2$$

which is the sum of the sine function and a quadratic function.

Perform the following tasks:

1. Define a Python function that returns the function value  $f(x)$  for some argument  $x$ . The function should work with both scalar numbers and NumPy arrays.

*Hint:* The sine function is implemented in `np.sin()`.

2. Evaluate the function on a uniformly spaced grid of  $x$ -values on the interval  $[-1, 1]$  using 51 grid points.

Create a plot depicting the function on this interval. Set the  $x$ -ticks to the values  $[-1, 0, 1]$  and add a legend.

3. Locate the point  $x_{min}$  where the function attains its minimum on the grid you created and print the values  $(x_{min}, f(x_{min}))$ .

*Hint:* You can use the NumPy function `np.argmin()` for this.

4. Augment the plot you created earlier with a horizontal and vertical line indicating where the minimum is located. Add a black dot at the point  $(x_{min}, f(x_{min}))$ .

*Hint:* Horizontal and vertical lines can be added with `axhline()` and `axvline()`. A single dot can be added in various ways, e.g., using `scatter()`.

5. Repeat the exercise but now use  $N = 501$  grid points to evaluate the function.

## Exercise 2: Visualizing US business cycles and recessions

In this exercise you are asked to plot two main business cycle indicators, GDP growth and the unemployment rate, for the United States.

1. Load the data from the CSV file `FRED/FRED_quarterly.csv` located in the repository's data folder.

Familiarize yourself with the data by printing the columns present in the data set and the first 5 rows of data.

*Hint:* Use `pd.read_csv(..., parse_dates=['DATE'], index_col='DATE')` to automatically parse strings stored in the DATE column as dates and set DATE as the index.

2. Compute the quarterly GDP growth (i.e., the relative change of the column GDP vs. the previous quarter) in percent.

*Hint:* This can be achieved using the `pct_change()` method.

3. Create a figure with two subplots (arranged in 2 rows) which show the GDP growth and the unemployment rate.

*Hint:* You can construct the graph either using Matplotlib's plotting function or using pandas's `DataFrame.plot`.

4. You want to understand how the business cycle dynamics correspond to the official US recession dates as determined by National Bureau of Economic Research (NBER), the agency tasked with dating business cycles in the US.

Recessions are defined as periods after GDP peaked (the corresponding quarter is stored in the peak column) until GDP started growing again (stored in the trough column).

1. Load the business cycle dates from `NBER_cycle_dates.csv` located in the `data/` folder.

Drop all rows where the peak date starts before 1948.

*Hint:* Use `pd.read_csv(..., parse_dates=['peak', 'trough'])` to automatically parse strings in the peak and trough columns.

2. Loop through all remaining business cycle peak dates and add a vertical line to each of the two subplots for GDP growth and unemployment at each date. These lines indicate quarters which were followed by a recession.
3. Can you detect any patterns for GDP growth and the unemployment rate after each peak date?

## Exercise 3: Daily returns of US stock market indices

In this exercise, we examine how the three major US stock market indices performed this year. Use the data in `indices.csv` from the folder `../data/stockmarket` to solve the following tasks:

1. Load the CSV data and restrict the sample to the calendar year 2024.

*Hint:* The data is tab-separated, so you need to specify `sep='\\t'` when reading in the data.

*Hint:* You should use `pd.read_csv(..., parse_dates=['Date'])` to automatically parse strings stored in the Date column as dates.

2. The data comes in a “long” format where each date-ticker combination is stored in a separate row. For the following analysis, the data needs to be reshaped such that each ticker symbol is in its own column (“wide” format).

Use the `pivot()` method to reshape the DataFrame into the desired form. Consult the [user guide](#) to see a graphical illustration.

Your pivoted DataFrame should look as follows:

Ticker	DJIA	NASDAQ	SP500
Date			
2024-01-02	37715.0	14765.9	4742.8
2024-01-03	37430.2	14592.2	4704.8

3. Create a line plot which shows the time series for each of the three ticker symbols in a single graph, for example using `DataFrame.plot()`.
4. The graph you just created is not very satisfactory as the three indices are recorded on vastly different scales. Express each index relative to its initial value in 2024 (so that all three start with the value 1.0) and recreate the previous graph with this normalized data.

5. Another way to check whether the three indices are co-moving is to compute and plot their daily returns. For each index, compute the daily returns, i.e., the relative change vs. the previous closing price in percent (e.g., using `pct_change()`), and plot the three time series of daily returns in a single graph.
6. The previous plots suggests that the three indices co-move a lot.
  1. In order to quantify the strength of this co-movement, compute the pairwise correlations of daily returns using `corr()`.
  2. Create a figure with 3-by-3 subplots where each off-diagonal plot shows the scatter plot for two of the indices.

*Hint:* You can use the function `scatter_matrix()` to accomplish this task. Alternatively, you can create a figure with 3-by-3 subplots using Matplotlib's `subplots()`, iterate over all rows and columns and add a `scatter()` plot to each axes object.
  3. Add the correlation coefficient for each stockmarket index pair as text to the corresponding subplot in the figure you created (this can be done using `text()`).