Plotting data

In the next chapter, we will learn how to plot data. For this, as we mentioned we will do with library Matplotlib. In this course, we will focus on bar graphs, but with Matplotlib, you can draw lines, markers, histograms, pies, etc…

Bar graphs (or bar charts) are used to represent categorical data with rectangular bars. Each bar's height (or length) corresponds to the of the category it represents. They are useful for:

- Comparing discrete categories: Each bar shows how a particular category compares to others.
- Visualizing counts or proportions: Bars help visualize the frequency of occurrence for different categories.
- Grouping data: Bar graphs can show grouped or stacked comparisons, like sales of products across different years.

We need to import our library as plt. After that, we are going to use the same data that we created in the previous chapter. To show data we can use function bar() or we can use function plot() on our DataFrame and just pass the argument that we want bar. It is shown in Figure 1.

```
1  import matplotlib.pyplot as plt
2  import pandas as pd
3
4
5  data = pd.read_csv('data.csv')
6
7  ax = data.plot(kind='bar')
8
9  plt.show()
```

Figure 1: example 4

Now when we know how to draw bar graphs we should learn how to stylize our graph. There are two ways to do it: we can use plt(Pyplot interface) or ax(Object-oriented interface).

1. plt (Pyplot interface):
   - plt is part of Matplotlib's Pyplot module, which provides a state-based interface.
   - It automatically manages figure and axes creation behind the scenes. This means you don’t need to explicitly create a figure or axes unless necessary. It’s a simpler and more intuitive interface, especially for beginners or simple plots.
   - You can call plt functions (like plt.bar(), plt.show(), etc.) directly, and Matplotlib will implicitly apply them to the current figure and axes.

2. ax (Object-oriented interface):
    - ax refers to AxesSubplot objects in the object-oriented interface of Matplotlib.
    - It gives you more control by explicitly creating and managing figures and axes. This is useful for creating complex, multi-plot figures or when fine-tuning plot elements.
    - You first create a figure and axes, then call methods on them (ax.plot(), ax.set_title(), etc.) instead of relying on global states.

Differences in use:

- plt is more intuitive for simpler plots but can become tricky when dealing with multiple subplots.
- ax provides more control and is better suited for customizing each subplot or figure, especially in complex visualizations.

Matplotlib provides many options to customize the appearance of bar charts, including color, edge color, hatch patterns, transparency, and more. You can apply these customizations either to all bars or to individual bars in your graph.

- color: Sets the fill color of the bars.
- edgecolor: Defines the color of the bar edges.
- linewidth: Specifies the thickness of the bar edges.
- hatch: Adds patterns (e.g., //, .., etc.) inside the bars.
- alpha: Controls the transparency of the bars (0 is fully transparent, 1 is fully opaque).
- width: Adjusts the width of each bar.
- align: Aligns the bars on the x coordinates ('center' or 'edge').
- zorder: Specifies the drawing order of the bars relative to other plot elements.
- title: Set title of chart
- xlabel: Name x axis
- ylabel: Name y label

We can also set up the grid by calling the grid() function. We can draw a grid on both axes or separately. Also, we can stylize our grid by adding background color or changing line width, line style, etc.

Now we can try this in our example. We can name our x label “food”, and y label “value”, and we can set our grid on both axes with a dotted style, red color. Grid should be behind bars. We can set our title to “Calories and proteins”. Under each bar, it should be written the name of the food. How this should look is shown in Figure 2.

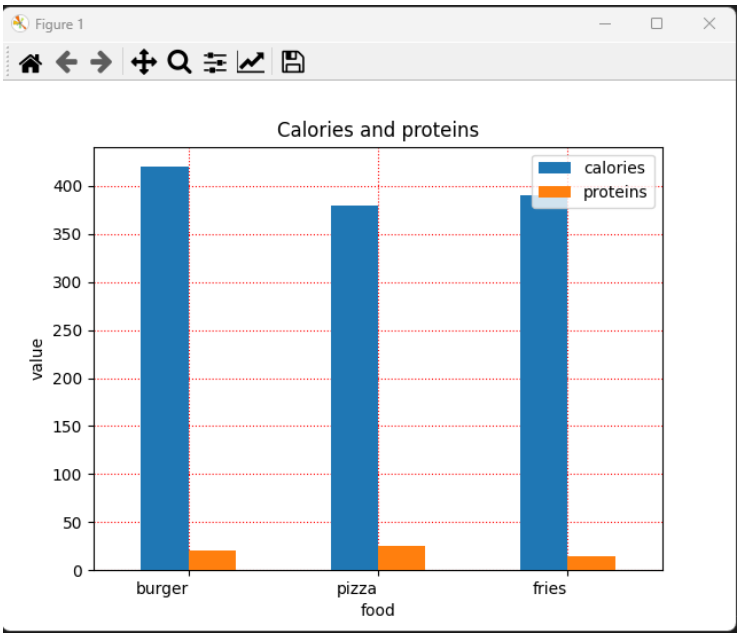

Figure 2: stylized example 4