Introduction to Data Visualization with Matplotlib

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Ariel Rokem
Data Scientist

datacamp

Data visualization

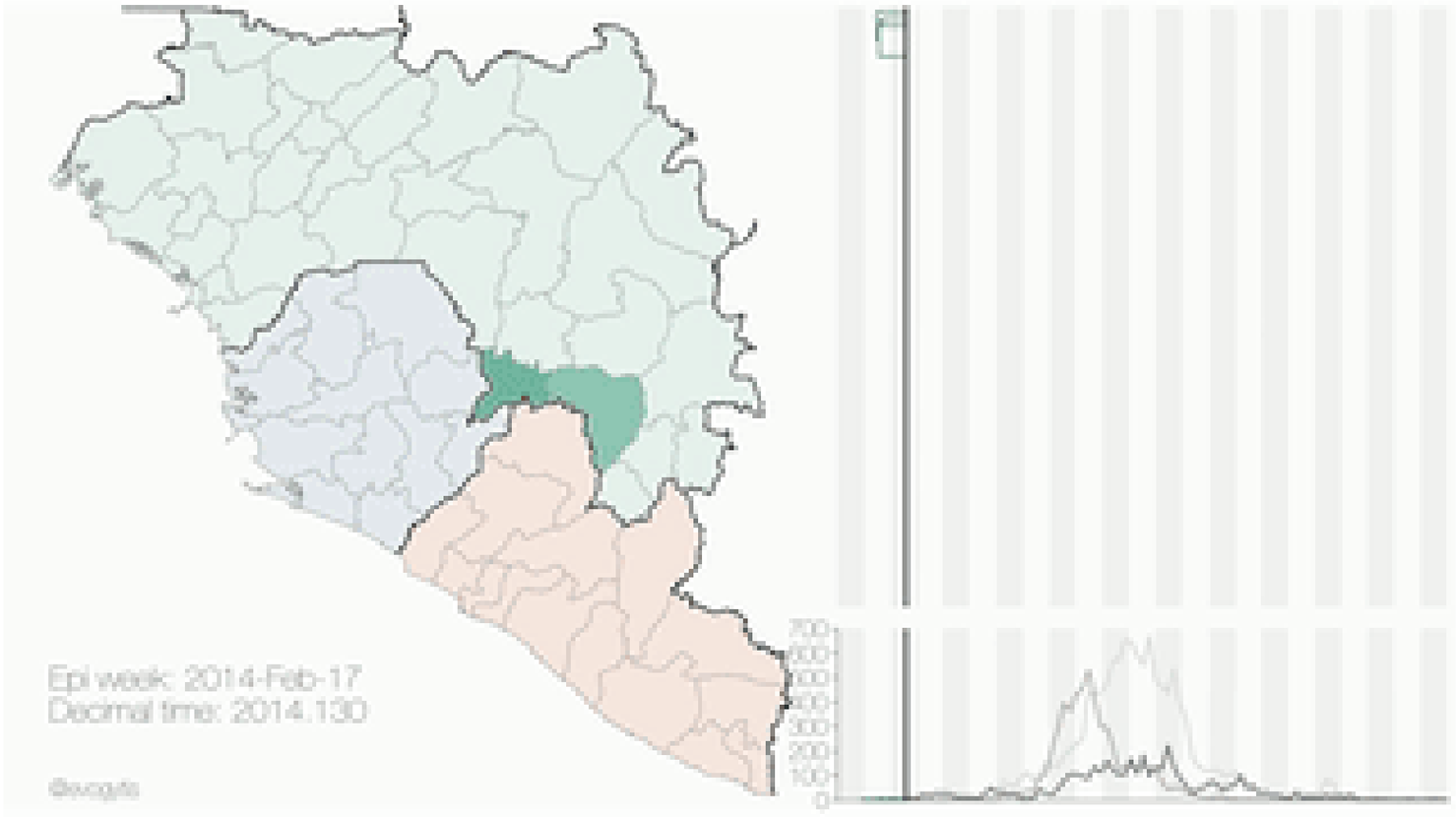
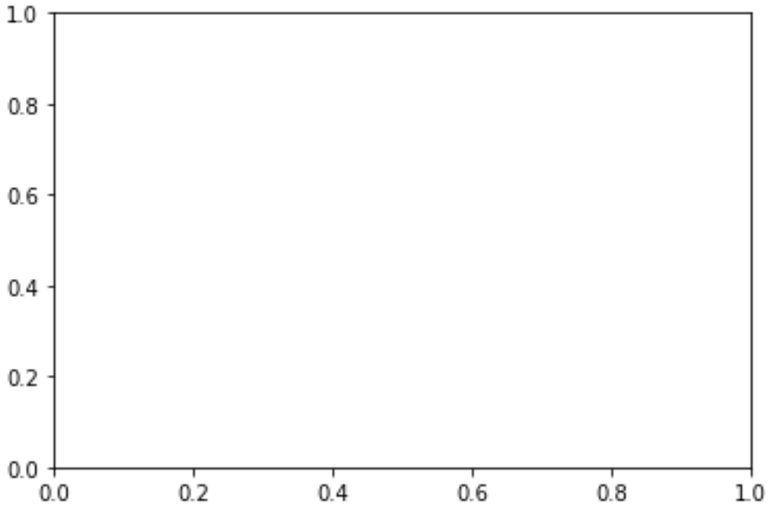


Image credit: **Gytis Dudas** and **Andrew Rambaut**

Introducing the pyplot interface

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
plt.show()
```



3. Introducing the pyplot interface
There are many different ways to use Matplotlib. In this course, we will use the main object-oriented interface. This interface is provided through the pyplot submodule. Here, we import this submodule and name it plt. While using the name plt is not necessary for the program to work, this is a very strongly-followed convention, and we will follow it here as well. The plt-dot-subplots command, when called without any inputs, creates two different objects: a Figure object and an Axes object. The Figure object is a container that holds everything that you see on the page. Meanwhile, the Axes is the part of the page that holds the data. It is the canvas on which we will draw with our data, to visualize it. Here, you can see a Figure with empty Axes. No data has been added yet.

Adding data to axes

```
seattle_weather["MONTH"]
```

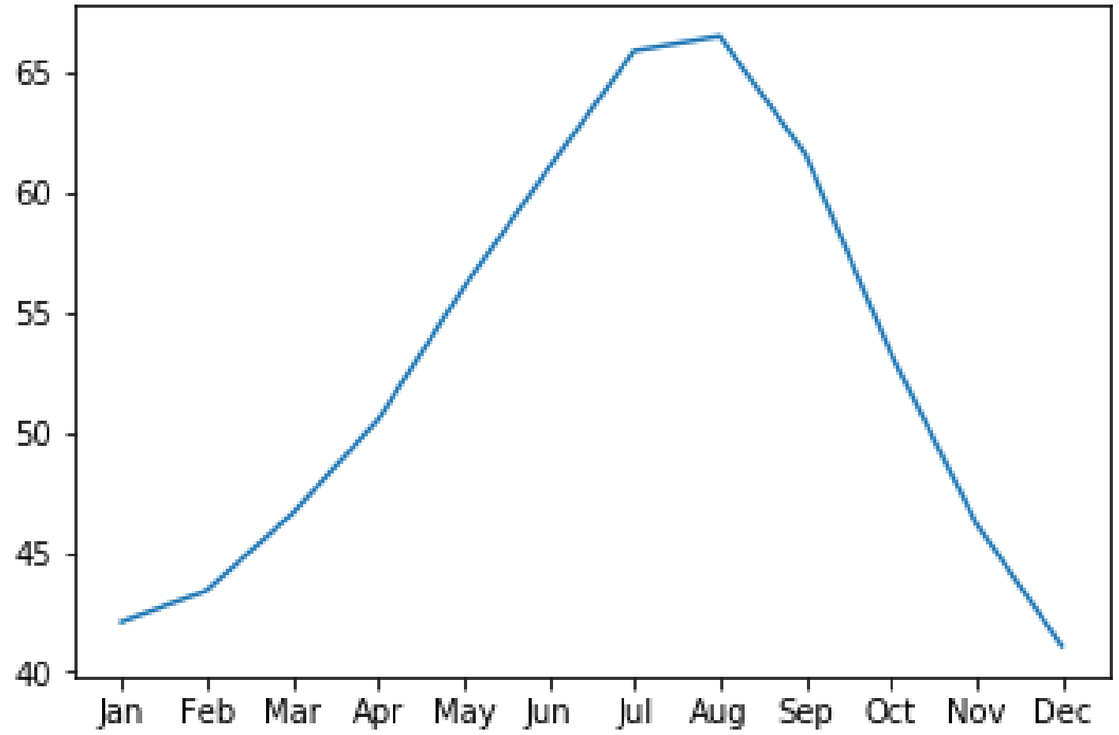
```
DATE
1     Jan
2     Feb
3     Mar
4     Apr
5     May
6     Jun
7     Jul
8     Aug
9     Sep
10    Oct
11    Nov
12    Dec
Name: MONTH, dtype: object
```

```
seattle_weather["MLY-TAVG-NORMAL"]
```

```
1     42.1
2     43.4
3     46.6
4     50.5
5     56.0
6     61.0
7     65.9
8     66.5
9     61.6
10    53.3
11    46.2
12    41.1
Name: MLY-TAVG-NORMAL, dtype: float64
```

Adding data to axes

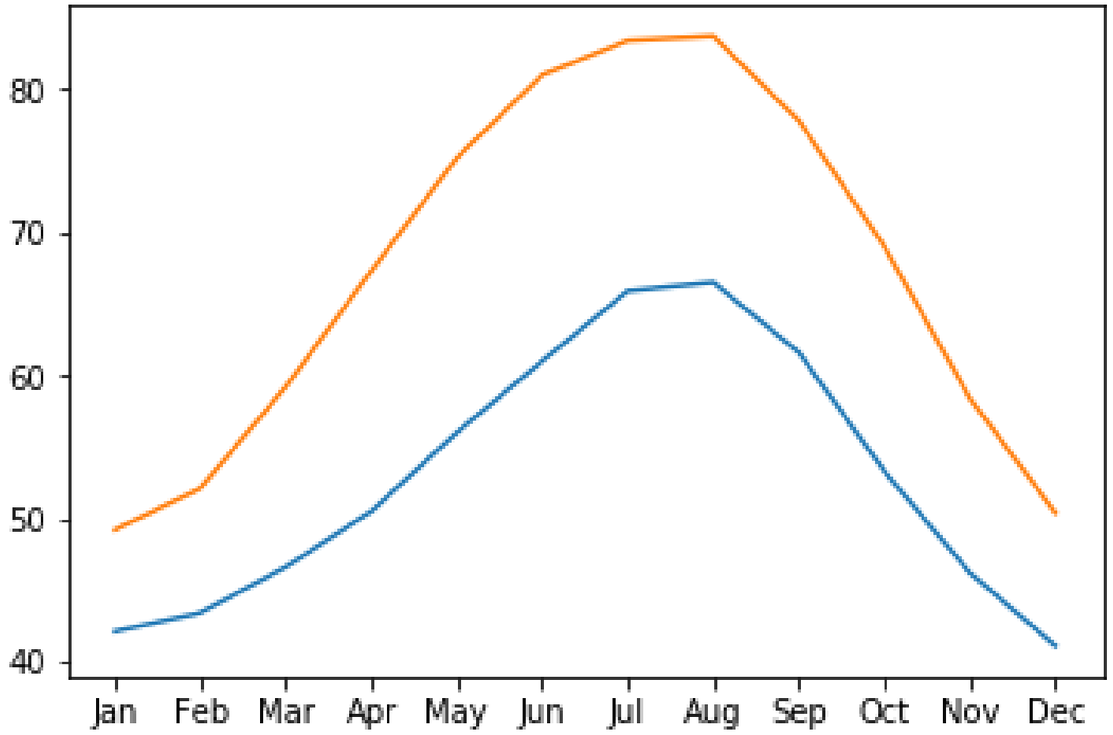
```
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])
plt.show()
```



4. Adding data to axes
Let's add some data to our figure. Here is some data. This is a DataFrame that contains information about the weather in the city of Seattle in the different months of the year. The "MONTH" column contains the three-letter names of the months of the year. The "monthly average normal temperature" column contains the temperatures in these months, in Fahrenheit degrees, averaged over a ten-year period.

Adding more data

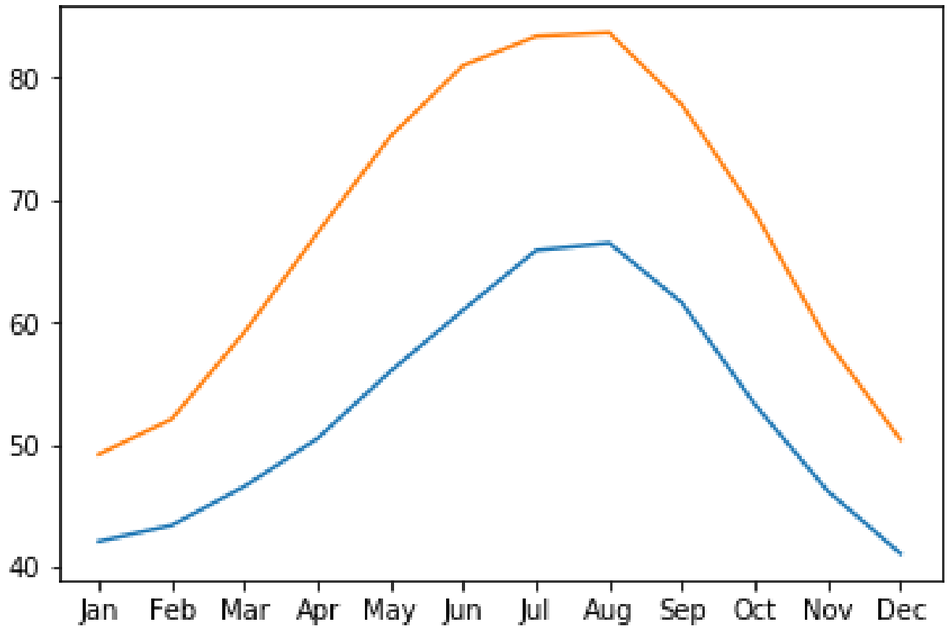
```
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])
plt.show()
```



5. Adding data to axes
To add the data to the Axes, we call a plotting command. The plotting commands are methods of the Axes object. For example, here we call the method called plot with the month column as the first argument and the temperature column as the second argument. Finally, we call the plt-dot-show function to show the effect of the plotting command. This adds a line to the plot. The horizontal dimension of the plot represents the months according to their order and the height of the line at each month represents the average temperature. The trends in the data are now much clearer than they were just by reading off the temperatures from the table.

Putting it all together

```
fig, ax = plt.subplots()
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])
plt.show()
```



7. Putting it all together
Here is what all of the code to create this figure would then look like. First, we create the Figure and the Axes objects. We call the Axes method plot to add first the Seattle temperatures, and then the Austin temperatures to the Axes. Finally, we ask Matplotlib to show us the figure.

Practice making a figure!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

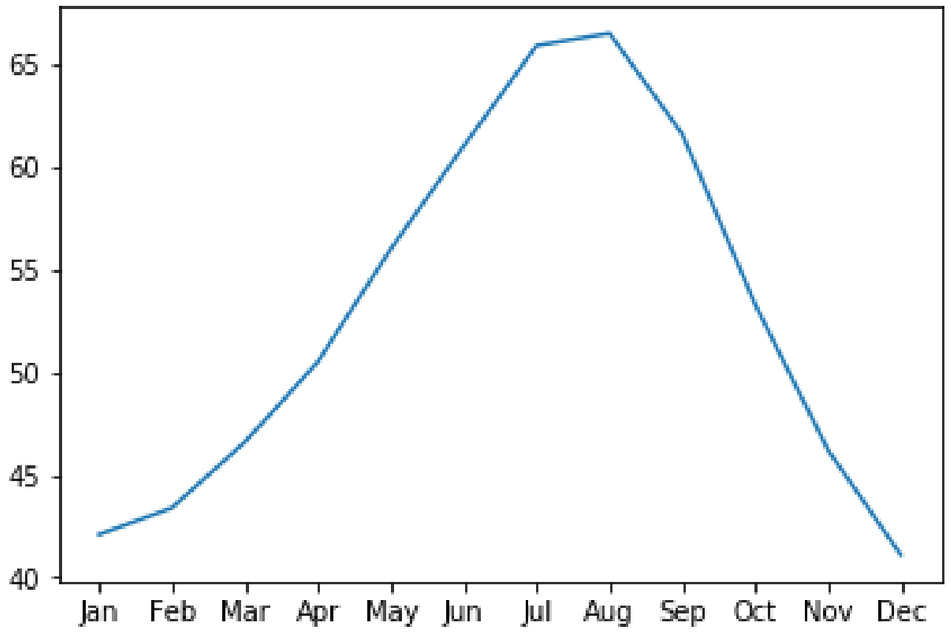
Customizing your plots

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Ariel Rokem
Data Scientist

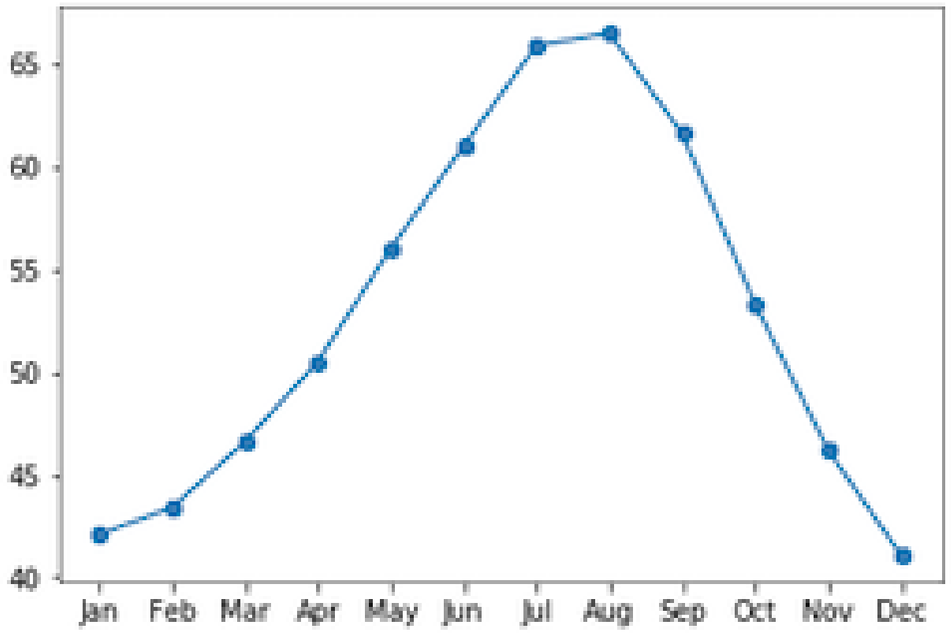
Customizing data appearance

```
ax.plot(seattle_weather["MONTH"],
        seattle_weather["MLY-PRCP-NORMAL"])
plt.show()
```



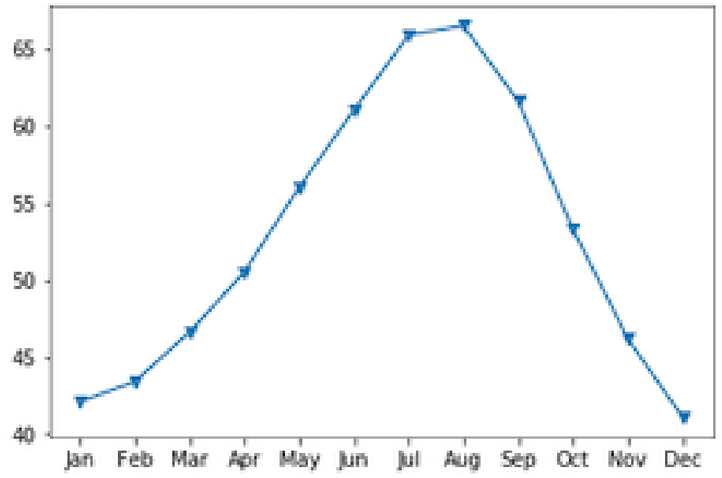
Adding markers

```
ax.plot(seattle_weather["MONTH"],
        seattle_weather["MLY-PRCP-NORMAL"],
        marker="o")
plt.show()
```



Choosing markers

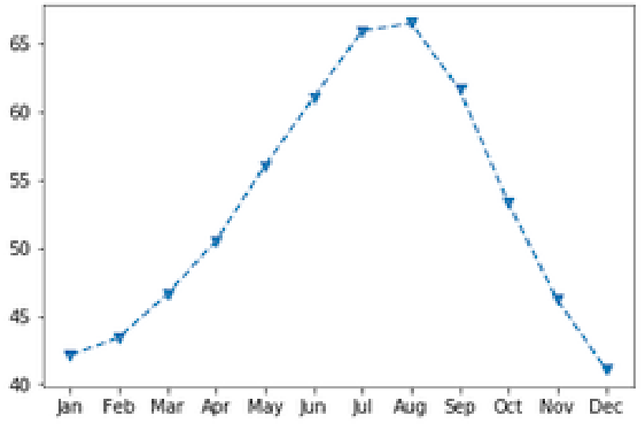
```
ax.plot(seattle_weather["MONTH"],
        seattle_weather["MLY-PRCP-NORMAL"],
        marker="v")
plt.show()
```



https://matplotlib.org/api/markers_api.html

Setting the linestyle

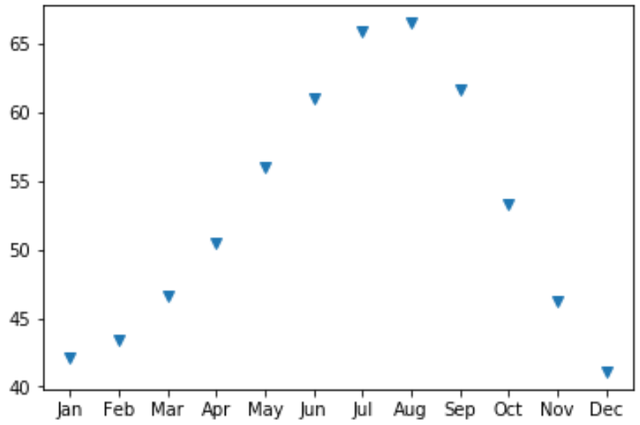
```
fig, ax = plt.subplots()
ax.plot(seattle_weather["MONTH"],
        seattle_weather["MLY-TAVG-NORMAL"],
        marker="v", linestyle="--")
plt.show()
```



https://matplotlib.org/gallery/lines_bars_and_markers/line_styles_reference.html

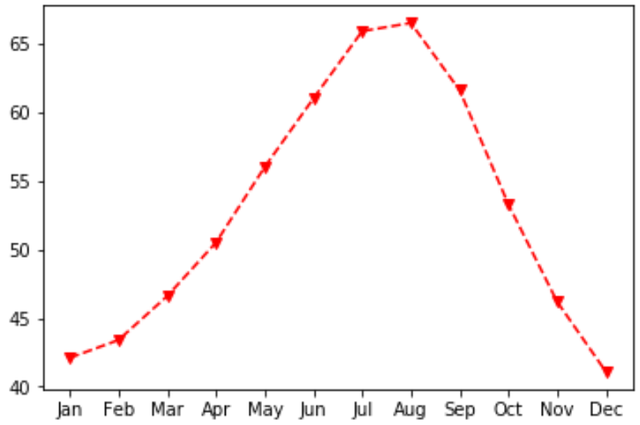
Eliminating lines with linestyle

```
fig, ax = plt.subplots()
ax.plot(seattle_weather["MONTH"],
        seattle_weather["MLY-TAVG-NORMAL"],
        marker="v", linestyle="None")
plt.show()
```



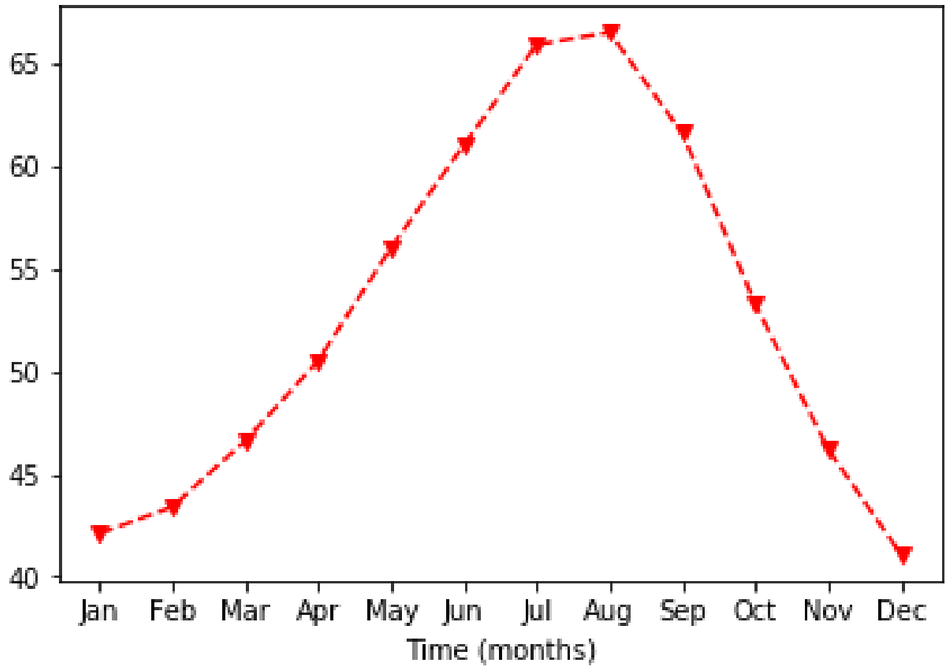
Choosing color

```
fig, ax = plt.subplots()
ax.plot(seattle_weather["MONTH"],
        seattle_weather["MLY-TAVG-NORMAL"],
        marker="v", linestyle="--", color="r")
plt.show()
```



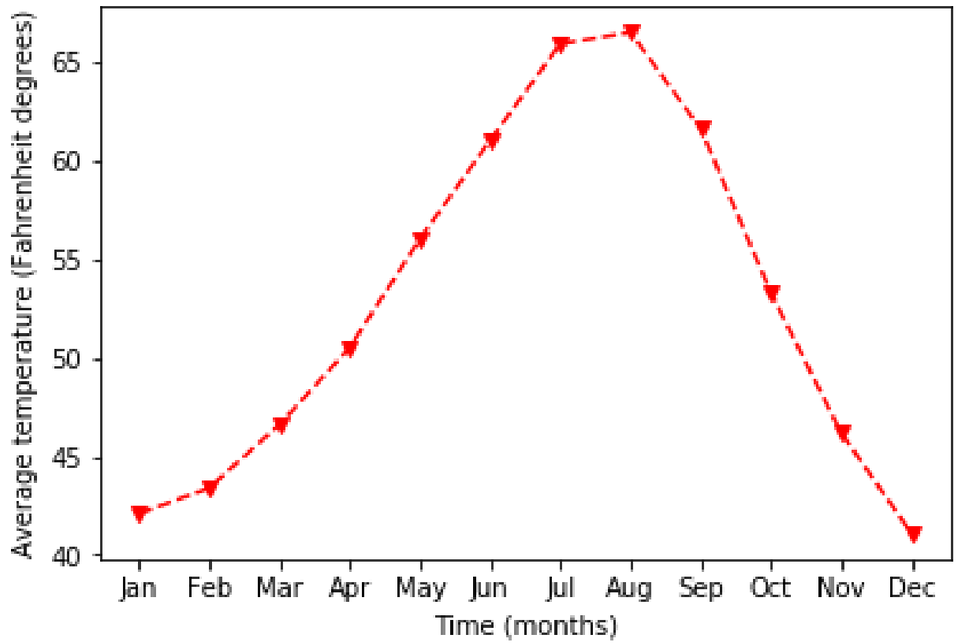
Customizing the axes labels

```
ax.set_xlabel("Time (months)")
plt.show()
```



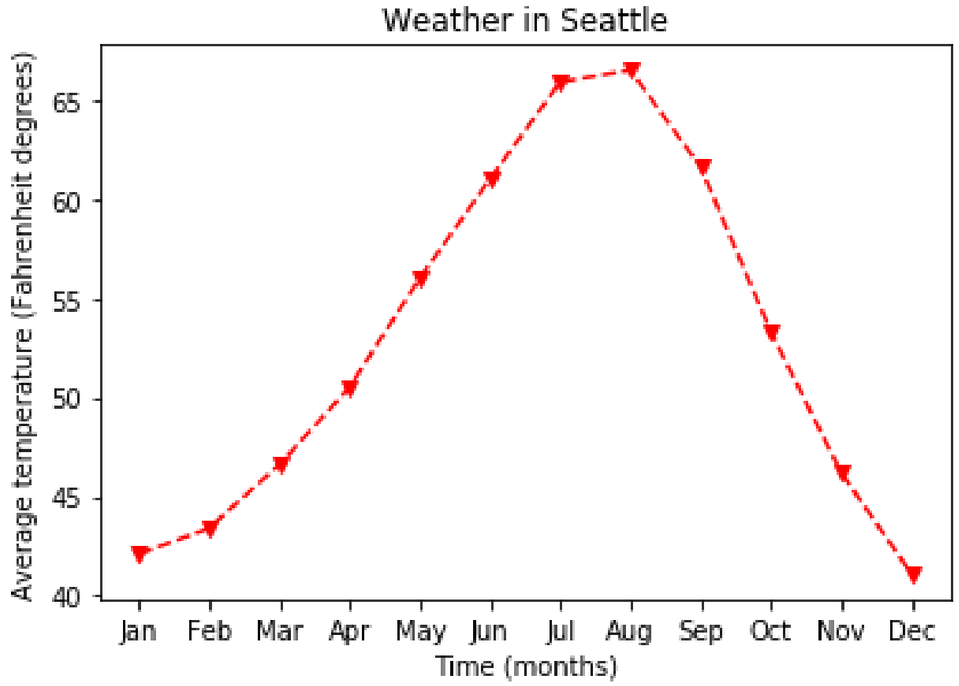
Setting the y axis label

```
ax.set_xlabel("Time (months)")
ax.set_ylabel("Average temperature (Fahrenheit degrees)")
plt.show()
```



Adding a title

```
ax.set_title("Weather in Seattle")
plt.show()
```



Practice customizing your plots!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

Small multiples

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

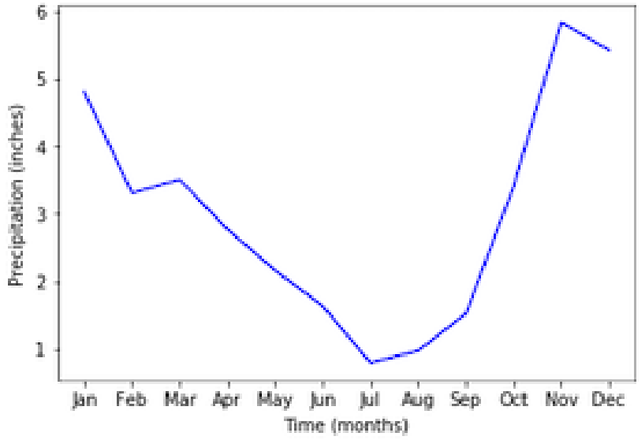
Ariel Rokem
Data Scientist

datacamp

Adding data

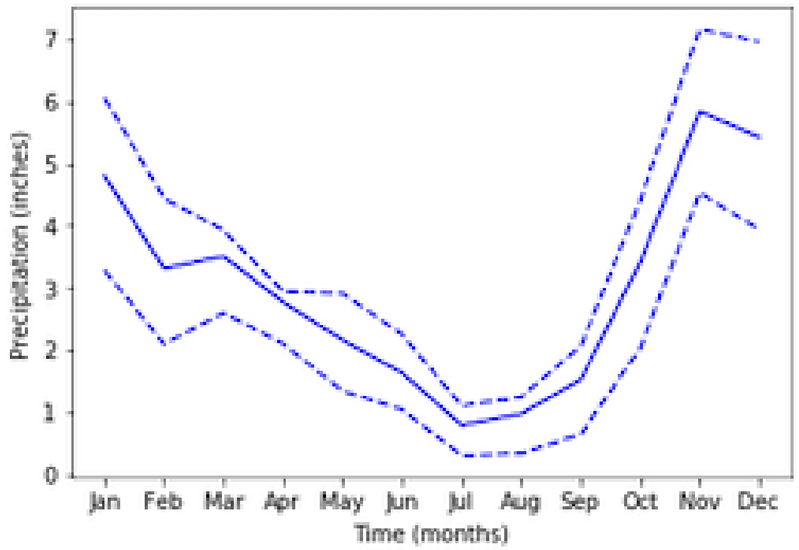
```
ax.plot(seattle_weather["MONTH"],
        seattle_weather["MLY-PRCP-NORMAL"],
        color='b')
ax.set_xlabel("Time (months)")
ax.set_ylabel("Precipitation (inches)")
plt.show()
```

2. Adding data
For example, let's explore the data we have about weather in Seattle. Here we plot average precipitation in Seattle during the course of the year. But let's say that we are also interested in the range of values.



Adding more data

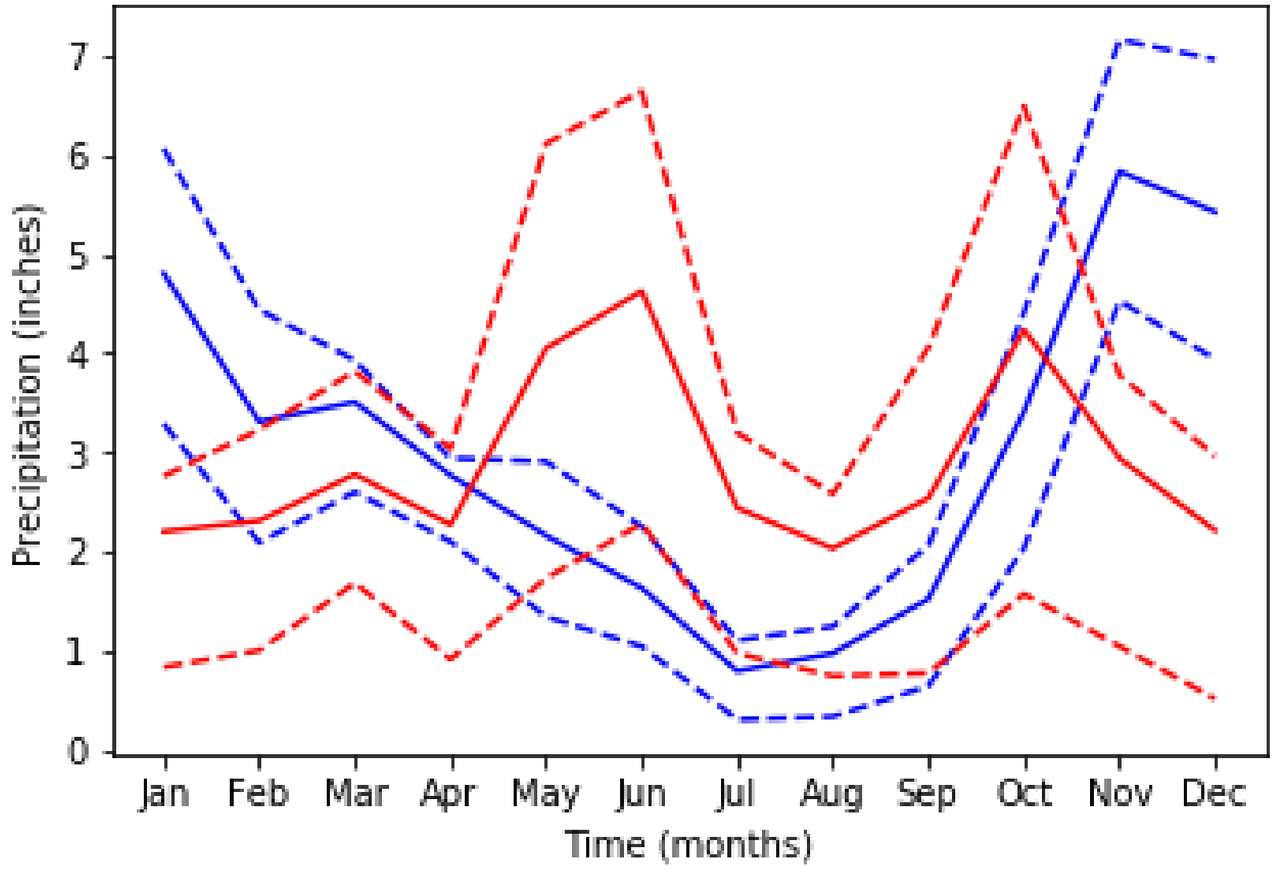
```
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-25PCTL"],
        linestyle='--', color='b')
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-75PCTL"],
        linestyle='--', color=color)
plt.show()
```



And more data

```
ax.plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-NORMAL"],
        color='r')
ax.plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-25PCTL"],
        linestyle='--', color='r')
ax.plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-75PCTL"],
        linestyle='--', color='r')
plt.show()
```

Too much data!

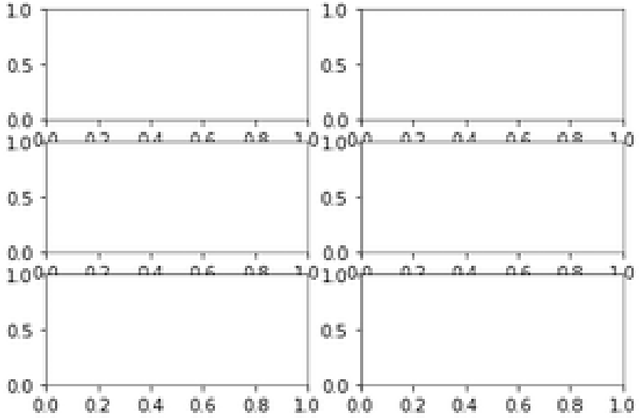


5. Too much data!
it's a bit of a mess. There's too much data in this plot. One way to overcome this kind of mess is to use what are called small multiples. These are multiple small plots that show similar data across different conditions. For example, precipitation data across different cities.

Small multiples with plt.subplots

```
fig, ax = plt.subplots()
```

```
fig, ax = plt.subplots(3, 2)
plt.show()
```



6. Small multiples with plt.subplots
In Matplotlib, small multiples are called sub-plots. That is also the reason that the function that creates these is called subplots. Previously, we called this function with no inputs. This creates one subplot. Now, we'll give it some inputs. Small multiples are typically arranged on the page as a grid with rows and columns. Here, we are creating a Figure object with three rows of subplots, and two columns. This is what this would look like before we add any data to it. In this case, the variable ax is no longer only one Axes object.

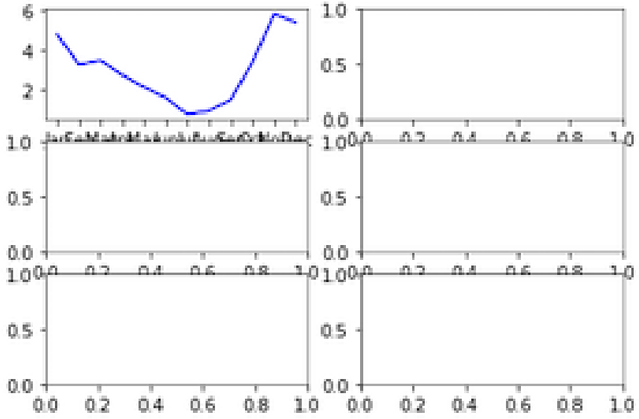
Adding data to subplots

```
ax.shape
```

```
(3, 2)
```

7. Adding data to subplots
Instead, it is an array of Axes objects with a shape of 3 by 2. To add data, we would now have to index into this object and call the plot method on an element of the array.

```
ax[0, 0].plot(seattle_weather["MONTH"],
              seattle_weather["MLY-PRCP-NORMAL"],
              color='b')
plt.show()
```



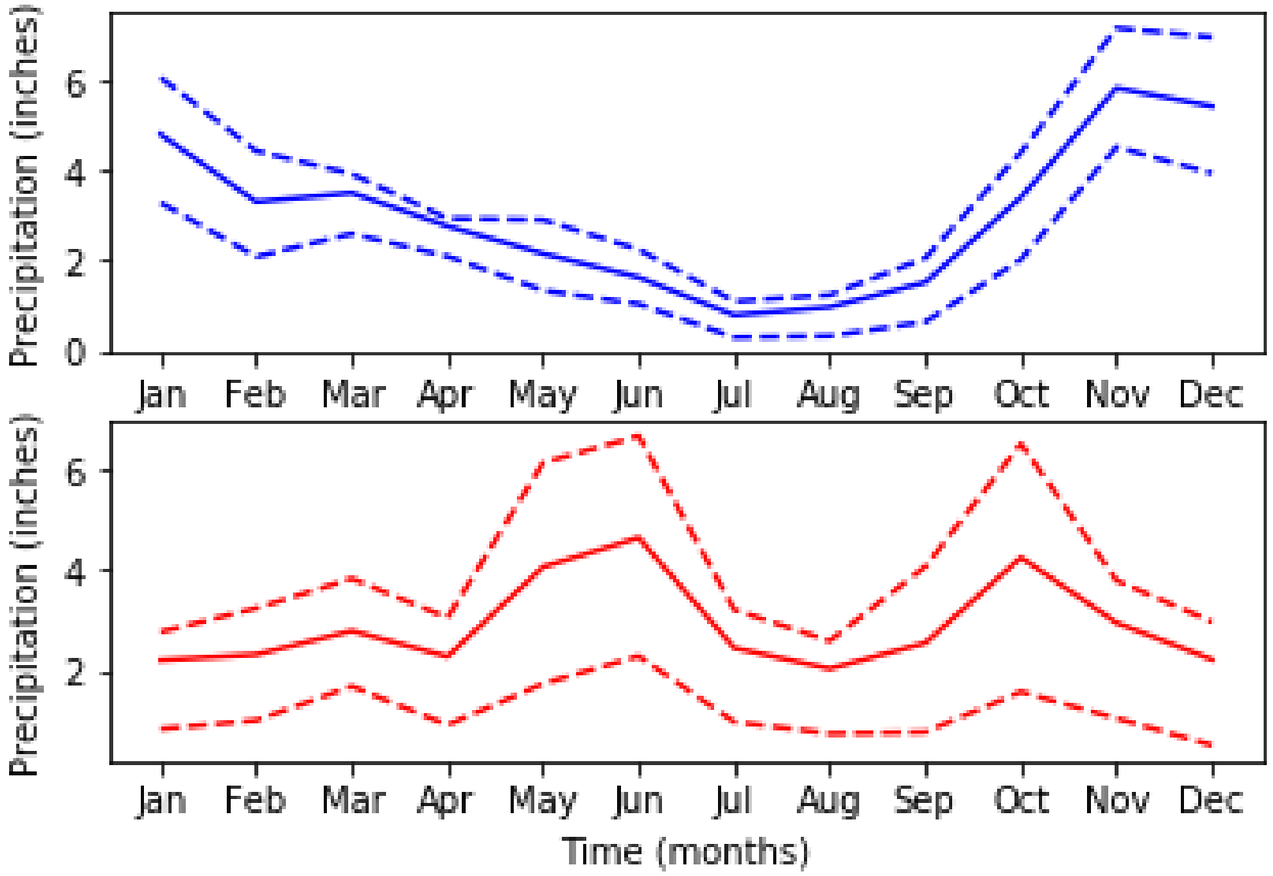
Subplots with data

```
fig, ax = plt.subplots(2, 1)
ax[0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-NORMAL"],
           color='b')
ax[0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-25PCTL"],
           linestyle='--', color='b')
ax[0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-75PCTL"],
           linestyle='--', color='b')
ax[1].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-NORMAL"],
           color='r')
ax[1].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-25PCTL"],
           linestyle='--', color='r')
ax[1].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-75PCTL"],
           linestyle='--', color='r')
ax[0].set_ylabel("Precipitation (inches)")
ax[1].set_ylabel("Precipitation (inches)")
ax[1].set_xlabel("Time (months)")
plt.show()
```

8. Subplots with data

There is a special case for situations where you have only one row or only one column of plots. In this case, the resulting array will be one-dimensional and you will only have to provide one index to access the elements of this array. For example, consider what we might do with the rainfall data that we were plotting before. We create a figure and an array of Axes objects with two rows and one column. We address the first element in this array, which is the top sub-plot, and add the data for Seattle to this plot. Then, we address the second element in the array, which is the bottom plot, and add the data from Austin to it. We can add a y-axis label to each one of these. Because they are one on top of the other, we only add an x-axis label to the bottom plot, by addressing only the second element in the array of Axes objects. When we show this,

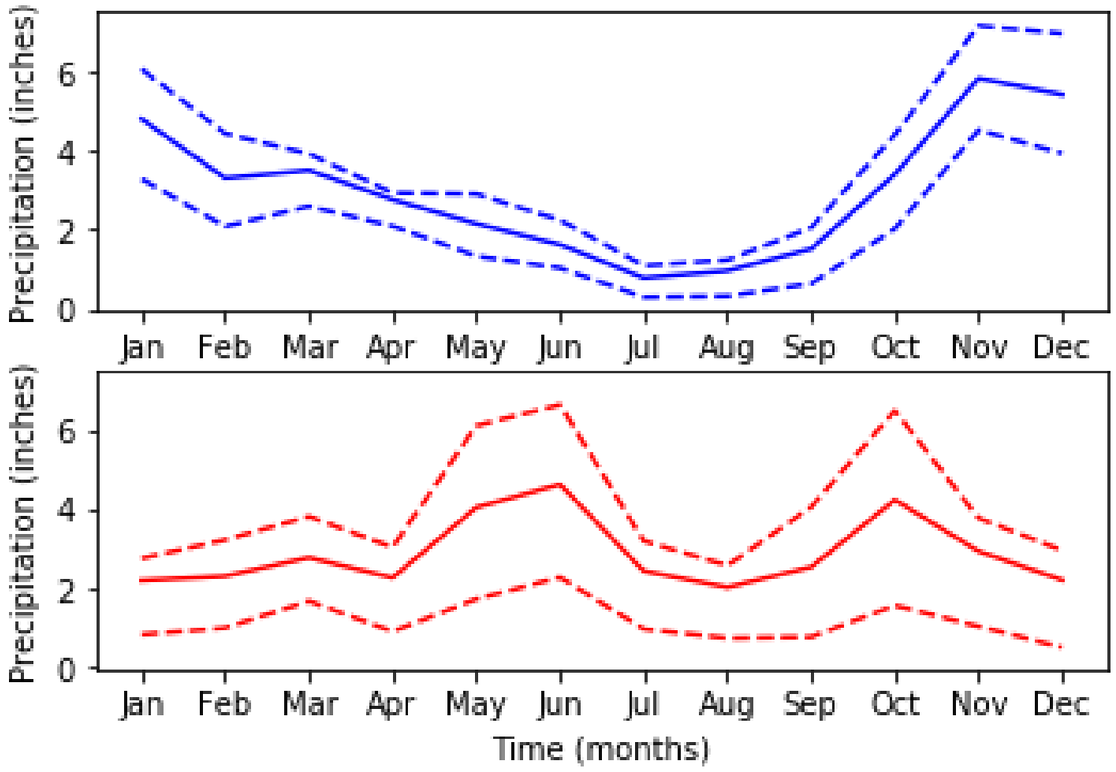
Subplots with data



9. Subplots with data
we see that the data are now cleanly presented in a way that facilitates the direct comparison between the two cities. One thing we still need to take care of is the range of the y-axis in the two plots, which is not exactly the same. This is because the highest and lowest values in the two datasets are not identical.

Sharing the y-axis range

```
fig, ax = plt.subplots(2, 1, sharey=True)
```



10. Sharing the y-axis range
To make sure that all the subplots have the same range of y-axis values, we initialize the figure and its subplots with the key-word argument sharey set to True. This means that both subplots will have the same range of y-axis values, based on the data from both datasets. Now the comparison across datasets is more straightforward.

Practice making subplots!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB