Getting Started with Matplotlib

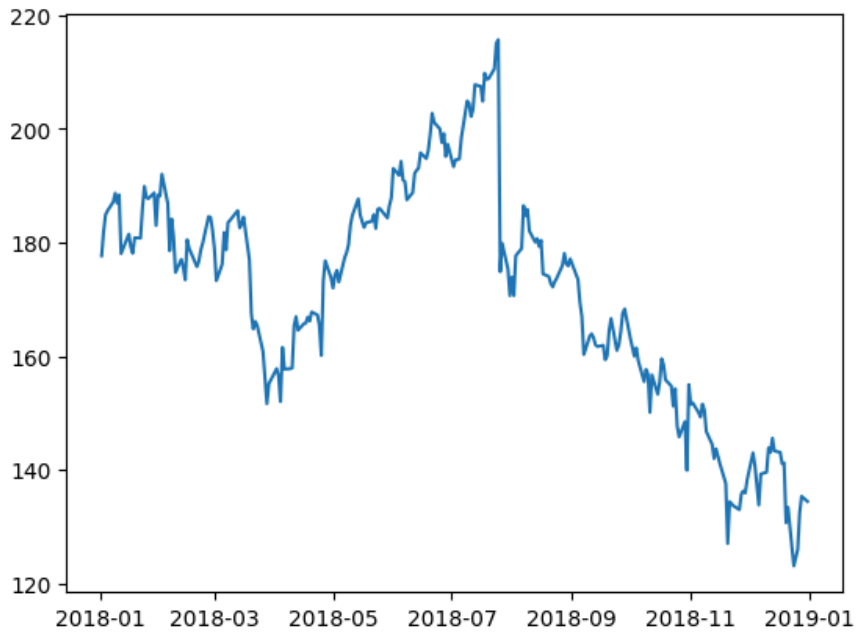
We need matplotlib.pyplot for plotting.

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

About the Data In this notebook, we will be working with 2 datasets: Facebook's stock price throughout 2018 (obtained using the stock_analysis package) Earthquake data from September 18, 2018 - October 13, 2018 (obtained from the US Geological Survey (USGS) using the USGS API)

Plotting lines

```
fb = pd.read_csv( # Read CSV file containing Facebook stock prices for 2018, setting date column as index, and parsing
    '/content/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
plt.plot(fb.index, fb.open) # Plotting the opening prices against dates
plt.show()
```



Since we are working in a Jupyter notebook, we can use the magic command %matplotlib inline once and not have to call plt.show() for each plot.

```
# Importing matplotlib for plotting and pandas
import matplotlib.pyplot as plt
import pandas as pd
fb = pd.read_csv(
    '/content/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
plt.plot(fb.index, fb.open)
```

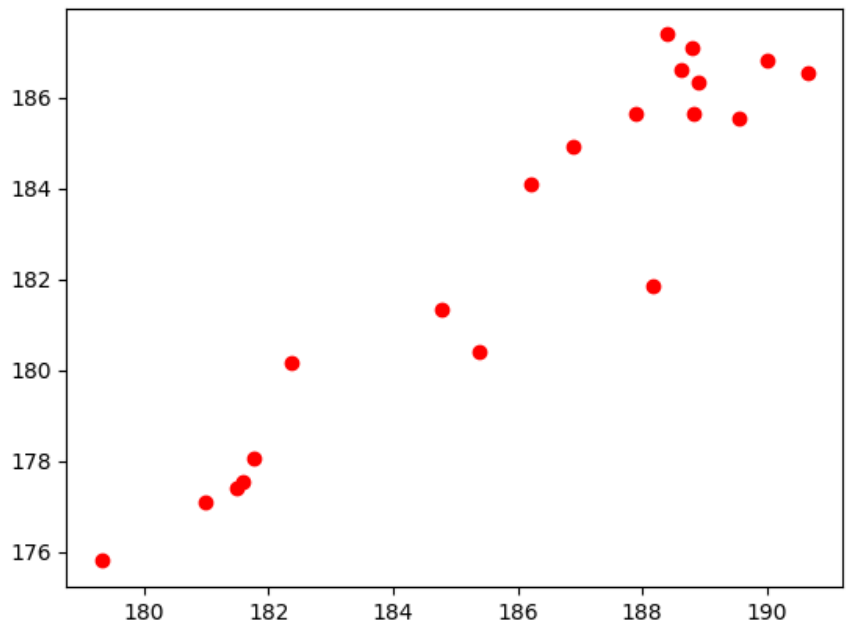
```
[<matplotlib.lines.Line2D at 0x7ed0c5702fb0>]
```

Scatter plots

We can pass in a string specifying the style of the plot. This is of the form '[color][marker][linestyle]'. For example, we can make a black dashed line with 'k--' or a red scatter plot with 'ro':

```
plt.plot('high', 'low', 'ro', data=fb.head(20))
# Plotting the high vs. low prices for the first 20 rows of the dataset with red circles
```

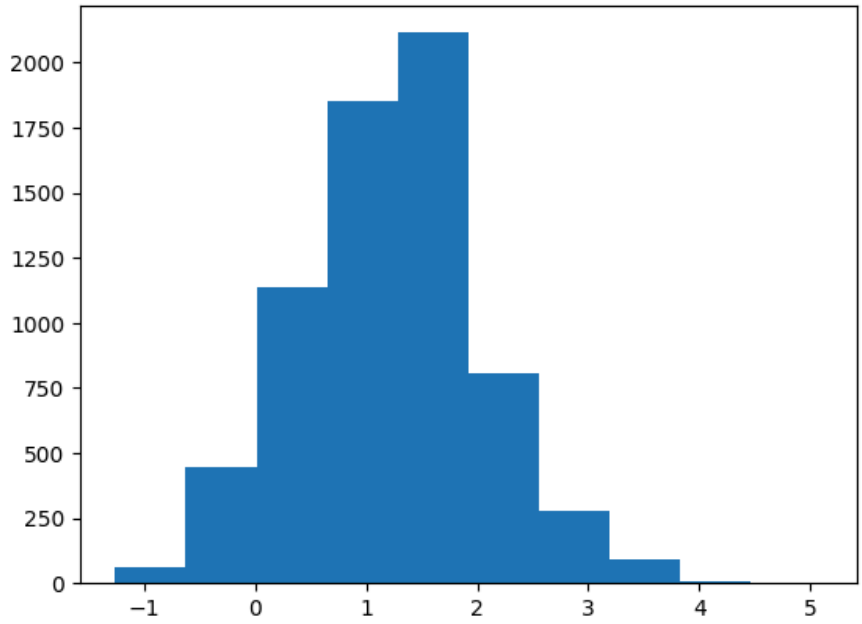
```
[<matplotlib.lines.Line2D at 0x7ed0c35e8400>]
```



Histograms

```
quakes = pd.read_csv('/content/earthquakes.csv')
plt.hist(quakes.query('magType == "ml"').mag)
# Creating a histogram of earthquake magnitudes with magnitudes measured in "ml" scale
```

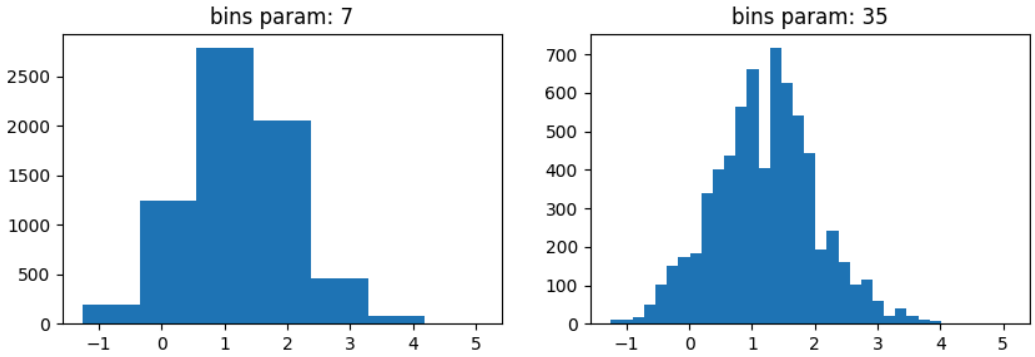
```
(array([6.400e+01, 4.450e+02, 1.137e+03, 1.853e+03, 2.114e+03, 8.070e+02,
        2.800e+02, 9.200e+01, 9.000e+00, 2.000e+00]),
 array([-1.26 , -0.624,  0.012,  0.648,  1.284,  1.92 ,  2.556,  3.192,
         3.828,  4.464,  5.1  ]),
 <BarContainer object of 10 artists>)
```



Bin size matters

Notice how our assumptions of the distribution of the data can change based on the number of bins (look at the drop between the two highest peaks on the righthand plot):

```
x = quakes.query('magType == "ml"').mag # Selecting earthquake magnitudes measured in "ml" scale
fig, axes = plt.subplots(1, 2, figsize=(10, 3)) # Creating subplots with 1 row and 2 columns
for ax, bins in zip(axes, [7, 35]): # Looping through axes and corresponding bins
  ax.hist(x, bins=bins) # Plotting histogram
  ax.set_title(f'bins param: {bins}') # Setting title for each subplot
```



Plot components

Figure

Top-level object that holds the other plot components.

```
fig = plt.figure() # Creating a new figure
```

```
<Figure size 640x480 with 0 Axes>
```

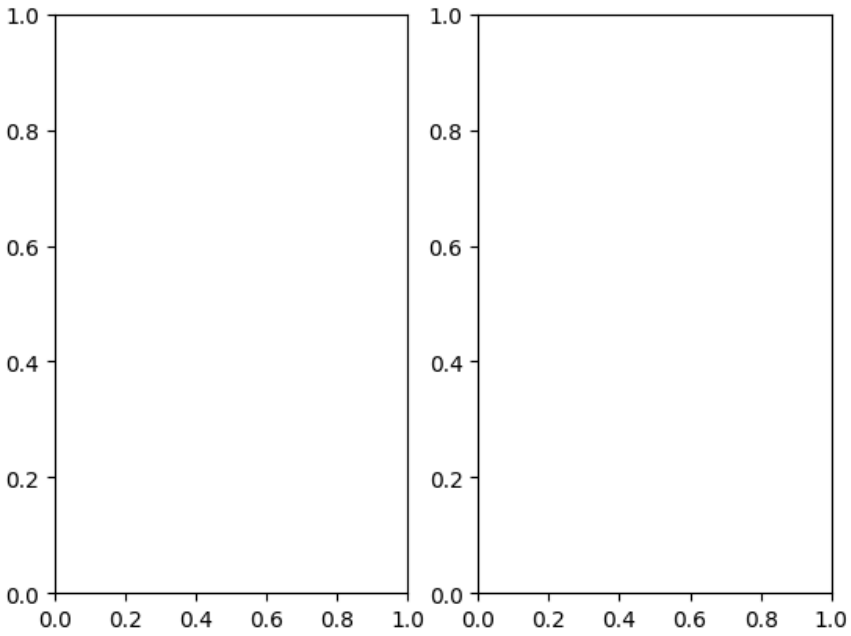
Axes

Individual plots contained within the Figure

Creating subplots

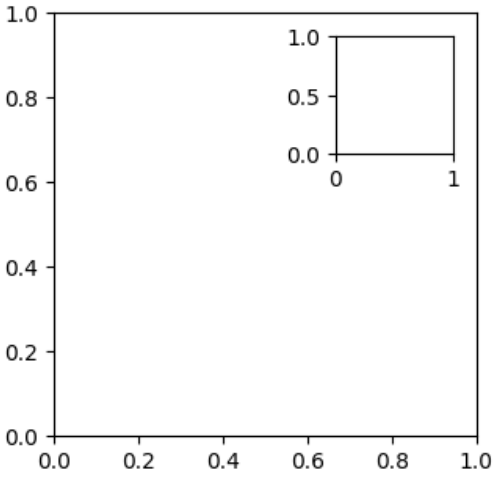
Simply specify the number of rows and columns to create:

```
fig, axes = plt.subplots(1, 2) # Creating a figure with 1 row and 2 columns of subplots
```



As an alternative to using plt.subplots() we can add the Axes to the Figure on our own. This allows for some more complex layouts, such as picture in picture:

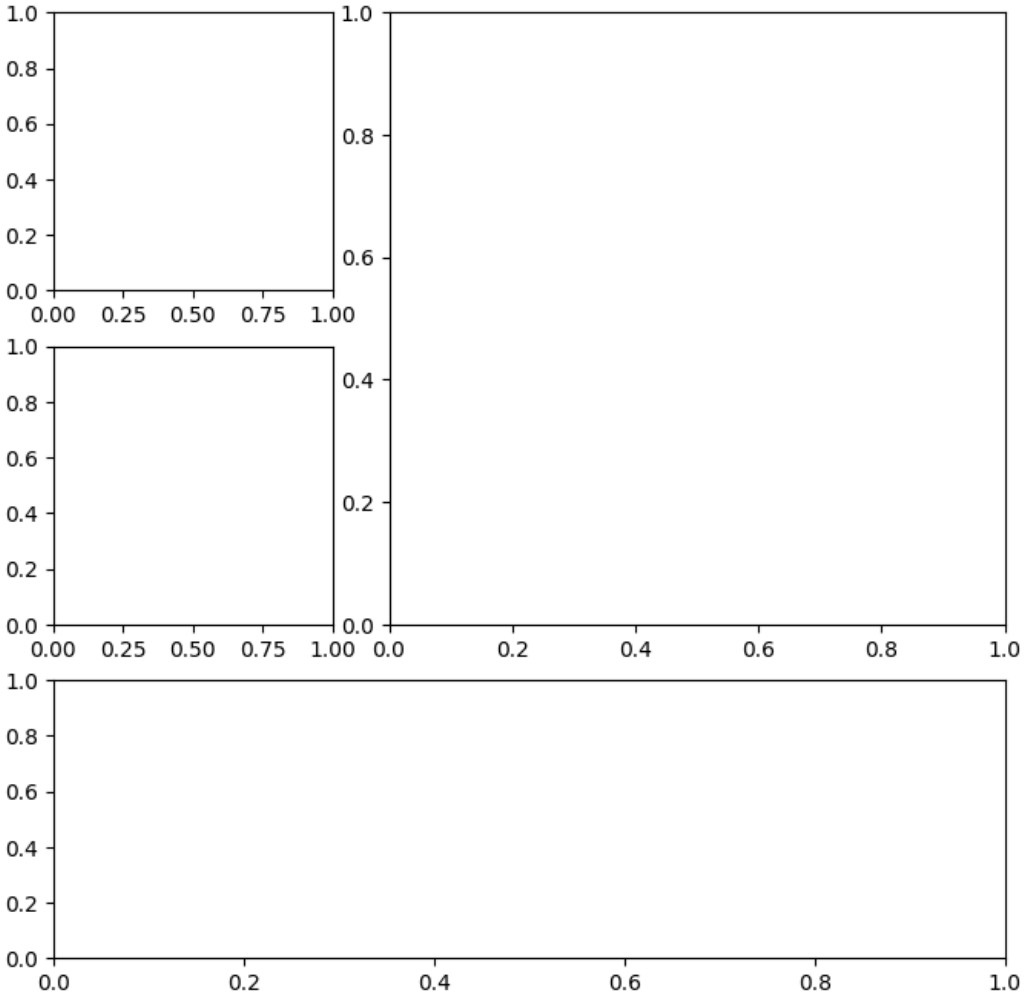
```
fig = plt.figure(figsize=(3, 3)) # Creating a new figure with specified size
outside = fig.add_axes([0.1, 0.1, 0.9, 0.9]) # Adding an outer axes to the figure
inside = fig.add_axes([0.7, 0.7, 0.25, 0.25]) # Adding an inner axes to the figure
```



Creating Plot Layouts with gridspec

We can create subplots with varying sizes as well:

```
fig = plt.figure(figsize=(8, 8))  # Creating a new figure with specified size
gs = fig.add_gridspec(3, 3)  # Adding a 3x3 grid layout to the figure
top_left = fig.add_subplot(gs[0, 0])  # Adding a subplot to the top-left position
mid_left = fig.add_subplot(gs[1, 0])  # Adding a subplot to the middle-left position
top_right = fig.add_subplot(gs[:2, 1:])  # Adding a subplot spanning the top-right positions
bottom = fig.add_subplot(gs[2,:])  # Adding a subplot spanning the entire bottom row
```



Saving plots

Use plt.savefig() to save the last created plot. To save a specific Figure object, use its savefig() method.

```
fig.savefig('empty.png') # Saving the figure
```

Cleaning up

It's important to close resources when we are done with them. We use plt.close() to do so. If we pass in nothing, it will close the last plot, but we can pass the specific Figure to close or say 'all' to close all Figure objects that are open. Let's close all the Figure objects that are open with plt.close():

```
plt.close('all') # Closing all figures
```

Additional plotting options

Specifying figure size

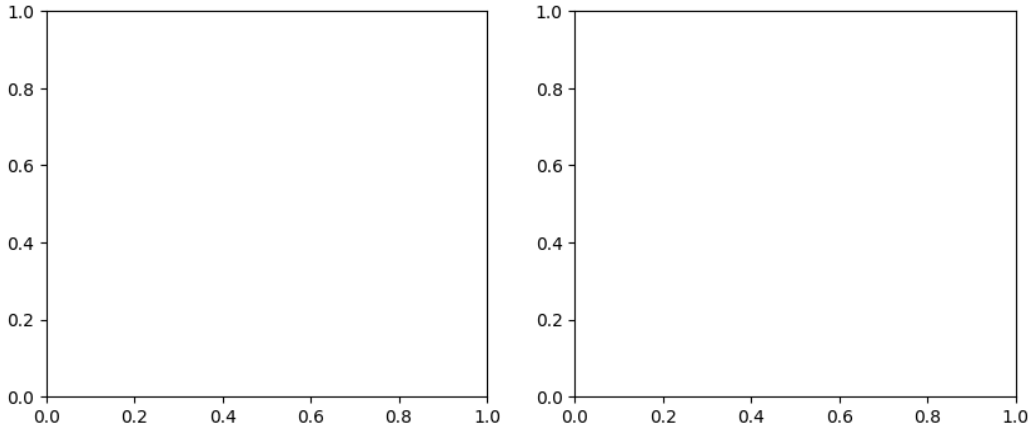
Just pass the figsize parameter to plt.figure() . It's a tuple of (width, height):

```
fig = plt.figure(figsize=(10, 4)) # Creating a new figure with 10 inches wide, 4 inches tall
```

```
<Figure size 1000x400 with 0 Axes>
```

This can be specified when creating subplots as well:

```
fig, axes = plt.subplots(1, 2, figsize=(10, 4))
# Creating a figure with 1 row and 2 columns of subplots with 10 inches wide, 4 inches tall
```



rcParams

A small subset of all the available plot settings (shuffling to get a good variation of options):

```
import random
import matplotlib as mpl
rcparams_list = list(mpl.rcParams.keys()) # Getting a list of all rcparams keys
random.seed(20) # make this repeatable
random.shuffle(rcparams_list) # Shuffling the list of rcparams keys
sorted(rcparams_list[:20]) # Sorting and displaying the first 20 elements of the shuffled list
```

```
['animation.convert_args',
 'axes.edgecolor',
 'axes.formatter.use_locale',
 'axes.spines.right',
 'boxplot.meanprops.markersize',
 'boxplot.showfliers',
 'keymap.home',
 'lines.markerfacecolor',
 'lines.scale_dashes',
 'mathtext.rm',
 'patch.force_edgecolor',
 'savefig.facecolor',
 'svg.fonttype',
 'text.hinting_factor',
 'xtick.alignment',
 'xtick.minor.top',
 'xtick.minor.width',
 'ytick.left',
 'ytick.major.left',
 'ytick.minor.width']
```

We can check the current default figsize using rcParams :