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 - Object 13, 2018 (obtained from the US Geological Survey (USGS) using the USGS API)

Plotting lines

```
fb = pd.read_csv(
    'fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)

plt.plot(fb.index, fb.open) # visual representation; x is index which is date and y is the c
plt.show() # showing of the plot
```


```
fb
```

	open	high	low	close	volume
date					
2018-01-02	177.68	181.58	177.5500	181.42	18151903
2018-01-03	181.88	184.78	181.3300	184.67	16886563
2018-01-04	184.90	186.21	184.0996	184.33	13880896
2018-01-05	185.59	186.90	184.9300	186.85	13574535
2018-01-08	187.20	188.90	186.3300	188.28	17994726
...	...	...	...	...	...
2018-12-24	123.10	129.74	123.0200	124.06	22066002
2018-12-26	126.00	134.24	125.8900	134.18	39723370
2018-12-27	132.44	134.99	129.6700	134.52	31202509
2018-12-28	135.34	135.92	132.2000	133.20	22627569
2018-12-31	134.45	134.64	129.9500	131.09	24625308

251 rows × 5 columns

Next steps: View recommended plots

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.

```
%matplotlib inline
# an auto shower of the plot
import matplotlib.pyplot as plt
import pandas as pd

fb = pd.read_csv('fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
plt.plot(fb.index, fb.open) # same as before, plots the data index and 'open' column but now
```

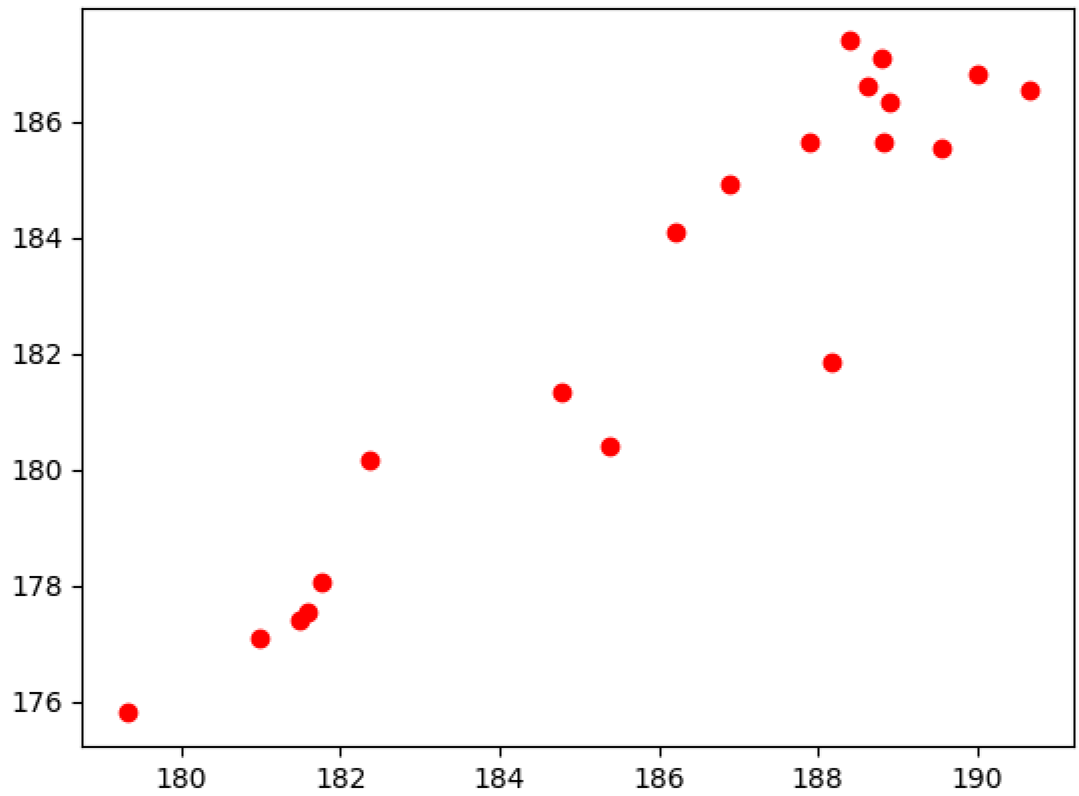
```
[<matplotlib.lines.Line2D at 0x788fc45a3700>]
```

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)) # plots the high and low column and uses rec
```

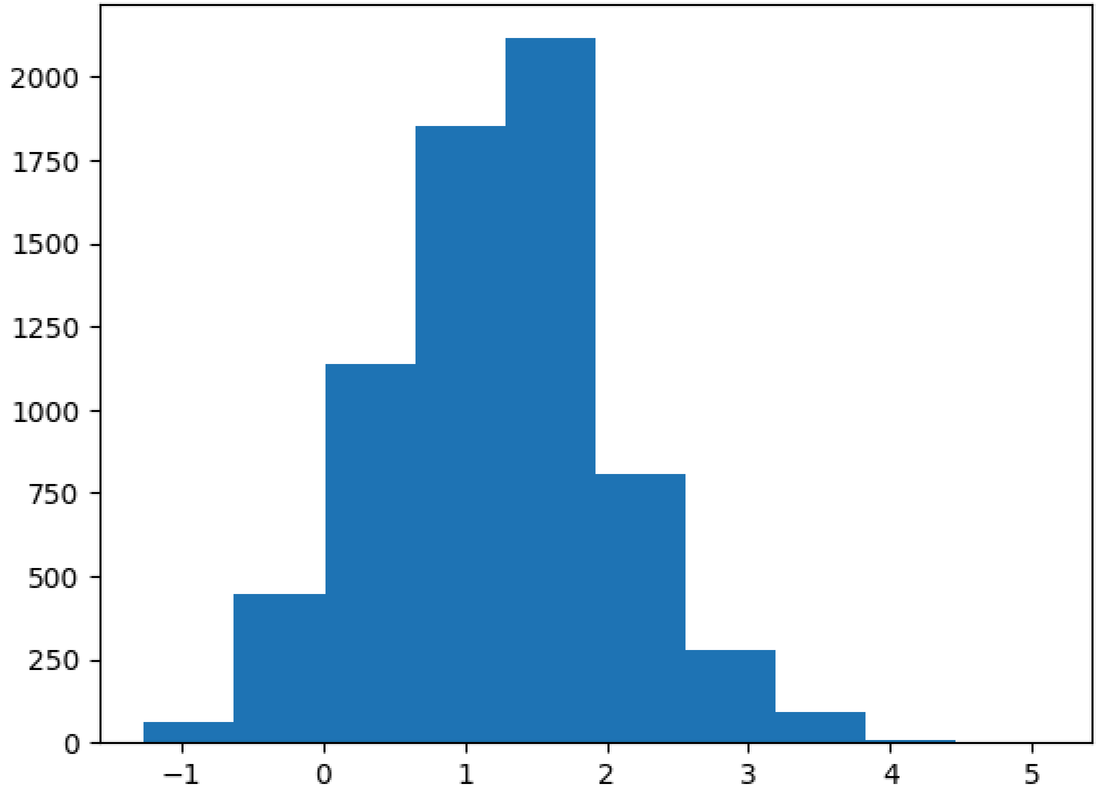
```
[<matplotlib.lines.Line2D at 0x788fc44a92a0>]
```



Histograms

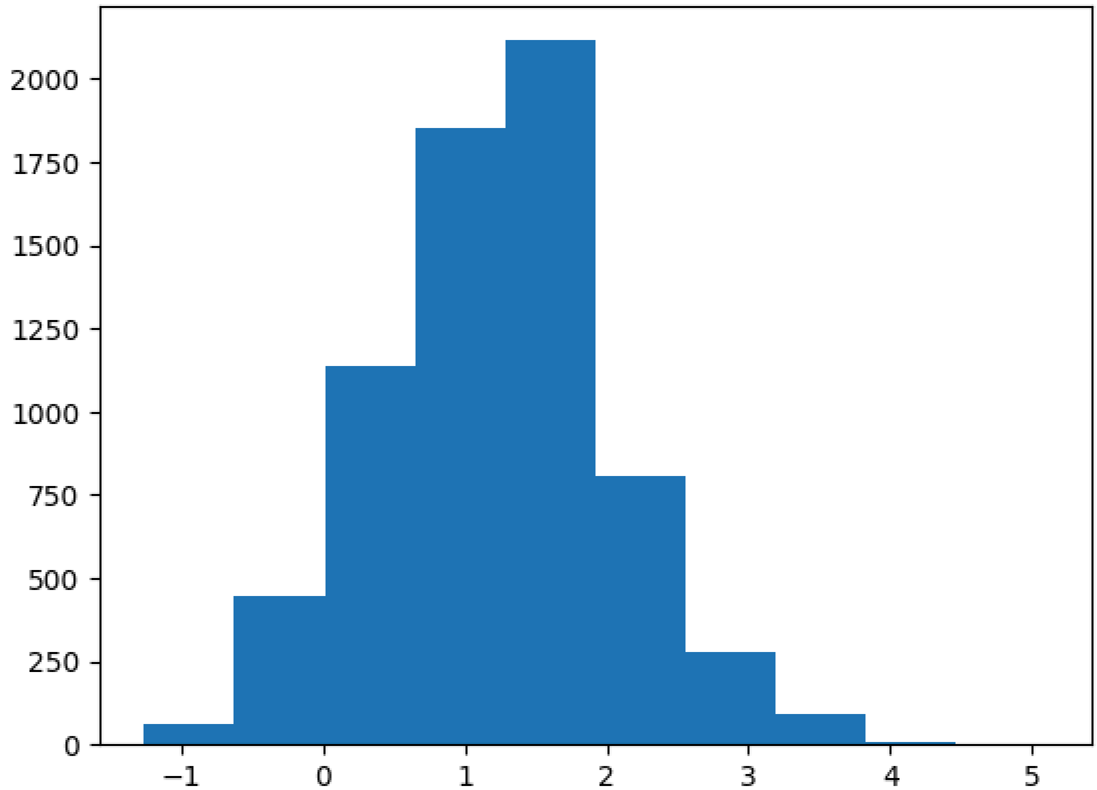
```
quakes = pd.read_csv('earthquakes-1.csv') # accesses eathquakes csv file
plt.hist(quakes.query('magType == "ml"').mag) # 'hist' is the call name for histograms
# Ploting the filtered out the magType data that is equal to 'ml' and the .mag is the one th
# so basically plotting the .mag with rows magType == 'ml'
```

```
(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>)
```




```
plt.hist(quakes.loc[quakes['magType'] == 'ml', 'mag'])
# same fucntionality but different method; I used .loc to access the filtered data
# the query method is much efficient
```

```
(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>)
```



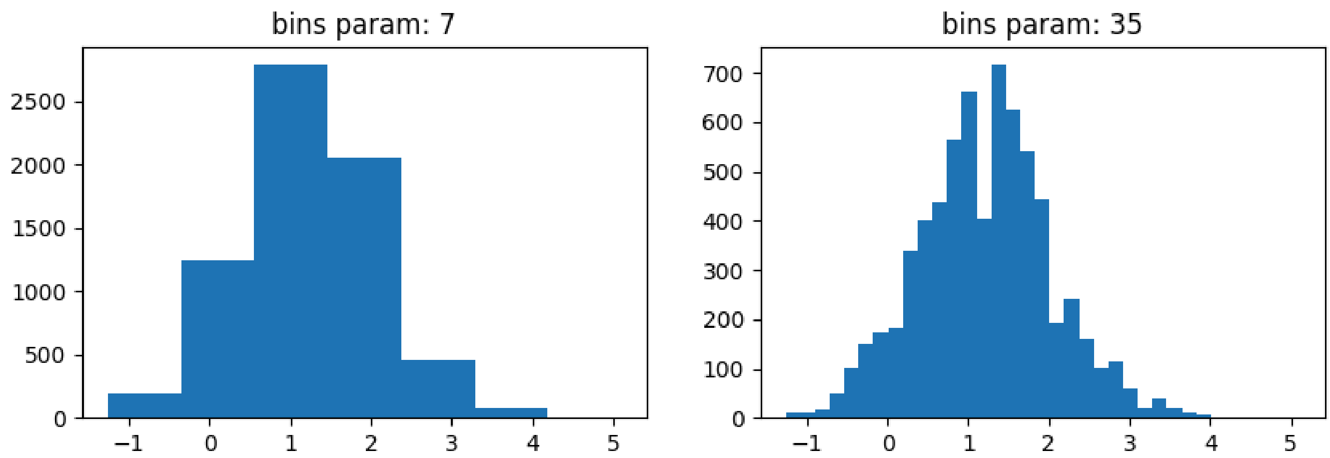
Bin size matters

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):

```
quakes.query('magType == "ml"').mag.value_counts()
```

```
 1.60    299
 1.30    288
 1.10    265
 1.40    262
 1.20    234
        ...
-0.41      1
 2.79      1
 2.62      1
 3.22      1
 2.81      1
Name: mag, Length: 406, dtype: int64
```

```
x = quakes.query('magType == "ml"').mag
fig, axes = plt.subplots(1, 2, figsize=(10, 3))
for ax, bins in zip(axes, [7, 35]):
  ax.hist(x, bins=bins)
  ax.set_title(f'bins param: {bins}')
```



Plot components

Figure

Top-level object that holds the other plot components

```
fig = plt.figure()
```

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

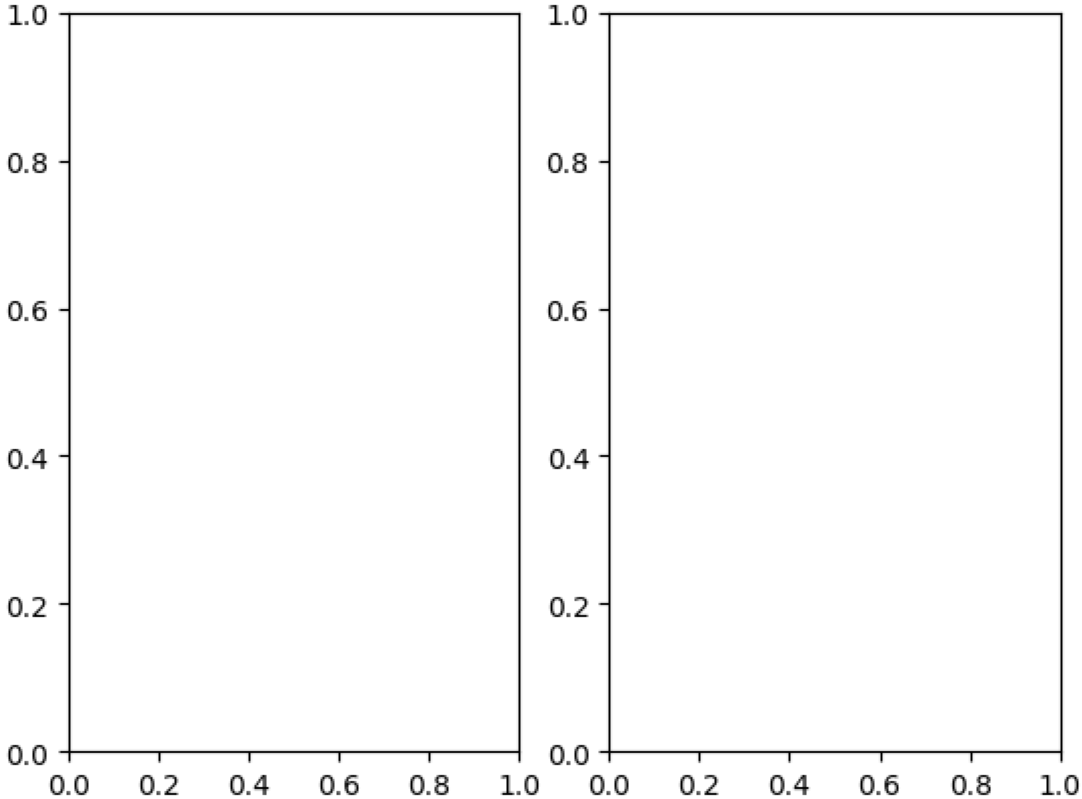
Axes

Individual plots contained the Figure.

Creating subplots

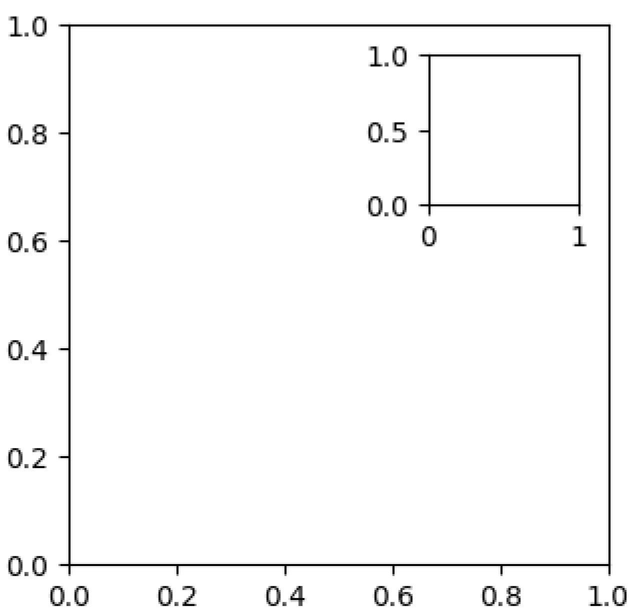
Simply specify the number of rows and columns to create:

```
fig, axes = plt.subplots(1, 2)
```



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:

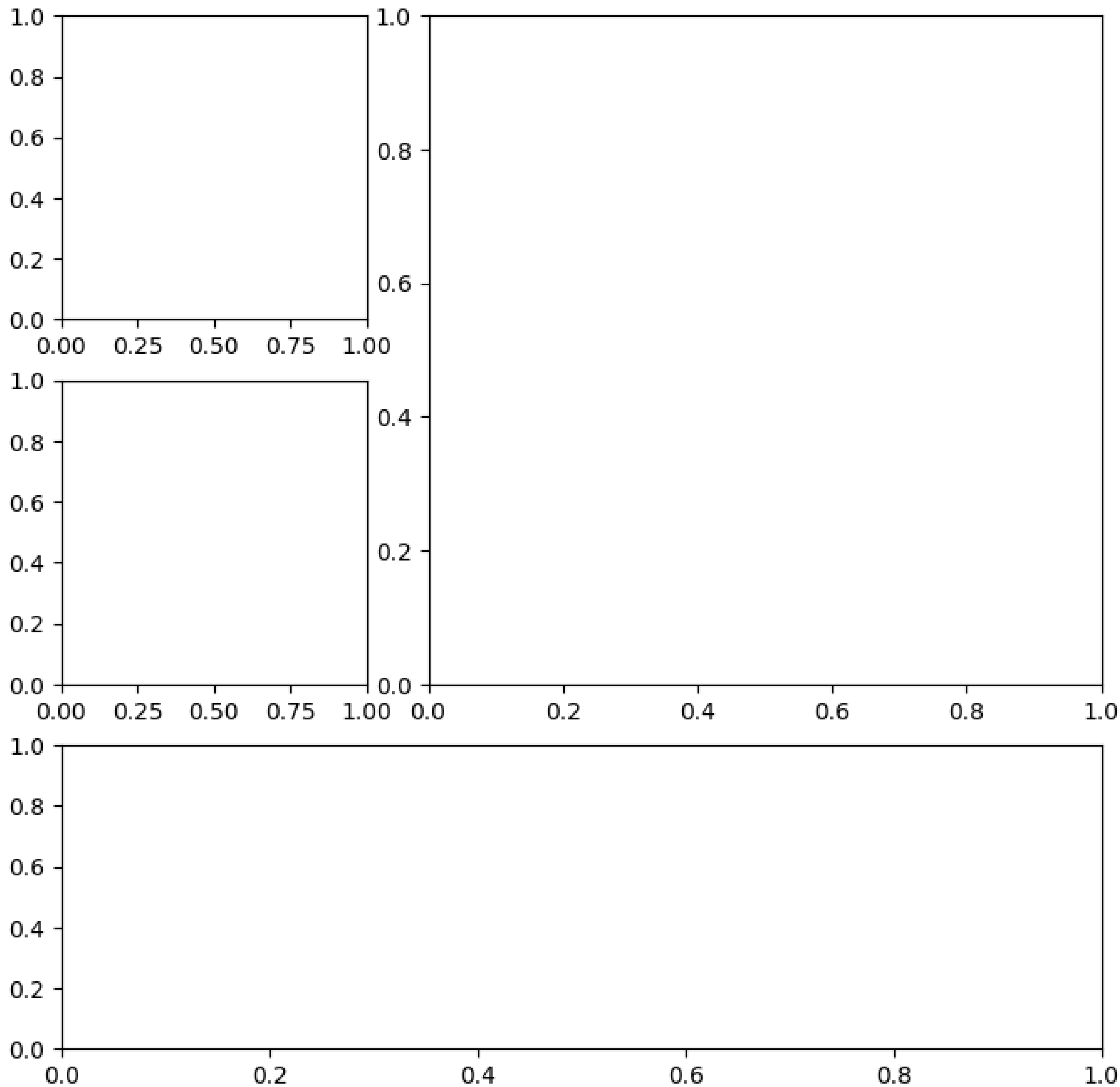
```
fig = plt.figure(figsize=(3,3))
outside = fig.add_axes([0.1, 0.1, 0.9, 0.9])
inside = fig.add_axes([0.7, 0.7, 0.25, 0.25])
```



Creating Plot Layouts with gridspec

We can create subplots with varying sizes as well:

```
fig = plt.figure(figsize=(8, 8))
gs = fig.add_gridspec(3, 3)
top_left = fig.add_subplot(gs[0, 0])
mid_left = fig.add_subplot(gs[1, 0])
top_right = fig.add_subplot(gs[:2, 1:])
bottom = fig.add_subplot(gs[2,:])
```



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')
```

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')
```

Additional plotting options

Specifying figure size

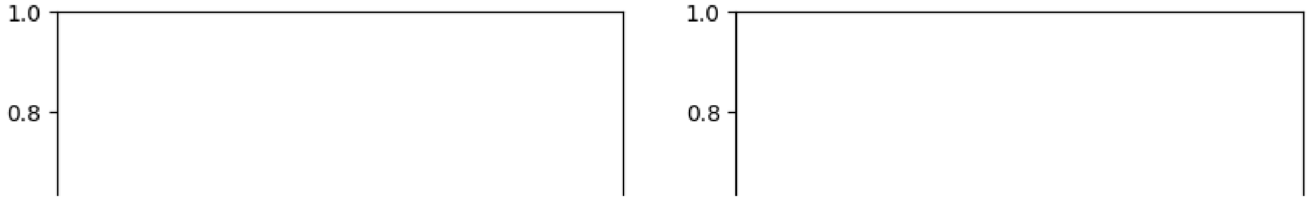
Just pass figsize parameter to plt.figure(). It's a tuple of (width, height):

```
fig = plt.figure(figsize=(7.2,2.88))
```

```
<Figure size 720x288 with 0 Axes>
```

This can be specified when creating subplots as well:

```
fig, axes = plt.subplots(1, 2, figsize=(10, 4))
```



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())
random.seed(20)
random.shuffle(rcparams_list)
sorted(rcparams_list[:20])
```

```
['animation.convert_args',
 'axes.edgecolor',
 'axes.formatter.use_locale',
 'axes.spines.right',
 'boxplot.meanprops.markersize',
 'boxplot.showfliers',
 'keymap.home',
```