Matplotlib's *pyplot* module for data visualization

Matplotlib's *pyplot* module helps visualize data from list, `ndarray`, and `DataFrame` data structures. Get access via `import matplotlib.pyplot as plt`.

Basic plots

- `plt.plot(x, y)` makes a line plot or scatter plot for the (x, y) points in arrays `x` and `y`; e.g.

  `x = np.linspace(start=0, stop=2*np.pi, num=20); y = np.sin(x); plt.plot(x, y)`

  Add an optional format string of the form `'[marker][line][color]'` (each optional; search for "`fmt =`" in pyplot.plot), e.g. `plt.plot(x, y, 'o--r')`, `plt.plot(x, y, '.')`
- `plt.boxplot(x, vert=True)` makes a boxplot from array `x`
- `plt.hist(x, bins=None, density=None)` makes a histogram from array `x`; set `bins` to a number or sequence of edges; set `density=True` to get total bar area 1; e.g.

  `plt.hist(x=y, bins=8, edgecolor='white')`,

  `plt.hist(x=y, bins=np.linspace(start=-1, stop=1, num=5), edgecolor='black', density=True)`
- `plt.bar(x, height[, tick_label])` makes a bar plot with x-coordinates of bars in array `x`, heights of bars in array `height`, and optional bar labels in array `tick_label`, e.g.

```
plt.bar(x=range(3), height=(6, 9, 11), tick_label=('Anna', 'Teresa', 'Margaret'))
plt.xticks(rotation=45) # rotate labels to avoid overlap
```

Title, axis labels, text annotations, and axis limits

- Add labels via `plt.title()`, `plt.xlabel()`, and `plt.ylabel()` and `plt.text(x, y, s)`, which adds string `s` at (`x`, `y`)

  Use a *raw* string `r'...'` to include Latex between dollar signs in the "`...`" portion.
- Use `plt.xlim(left, right)` and `plt.ylim(bottom, top)` to set x- and y-axis limits, e.g.

```
plt.plot(x, y, '-') # the following labels must be in the current cell
plt.title(r'The function $y=\sin(x)$')
plt.xlabel(r'$x$, with domain $[0, 2 \pi]$')
plt.ylabel(r'$y$, with range is $[-1, 1]$')
plt.xlim(0, 7)
plt.ylim(-2, 2)
plt.text(x=np.pi/2, y=1, s=r'$\left(\frac{\pi}{2}, 1\right)$')
```

Write a figure to a file

`plt.savefig(fname)` writes figure to file `fname` in format deduced from file extension (`.pdf`, `.png`, `.jpeg`, etc.), e.g. `plt.savefig('sinGraph.pdf') # must be in cell that made figure`

Legends

Use a format string and a label in each partial plot; then call `plt.legend()`. e.g.

```
plt.plot(x, y, 'og', label=r'$y=\sin(x)$')
plt.plot(x, -y, '--b', label=r'$y=-\sin(x)$')
plt.legend() # extracts marker/line type/color and label from plt.plot() calls
```

Try the default legend via `plt.legend()`. For more, see Legend guide.

Multiple subplots within a plot

- Make an empty plot with `fig = plt.figure()`.
- Use `gs = fig.add_gridspec(nrows, ncols)` to specify a `nrows`-by-`ncols` subplot layout.
- Use `ax = fig.add_subplot(gs[...])`, where `...` selects part of the layout.
- Plot on each subplot axis `ax`.

e.g. Here are three subplots of sizes 1-by-1, 1-by-1, and 2-by-1 in a 4-by-1 layout:

```
fig = plt.figure() # create new blank figure
fig.suptitle('gridspec: three plots using 1/4, 1/4, and 1/2 of figure')
gs = fig.add_gridspec(nrows=4, ncols=1) # grid of plot axes
x = np.linspace(start=0, stop=4*np.pi, num=100) # fake data
xlim = (0, 13) # common x-axis limits

ax1 = fig.add_subplot(gs[0, 0]) # first plot in top 1/4
ax1.plot(x, -2 + x/3, label=r'$y = -2 + \frac{x}{3}$')
ax1.set(xlim=xlim, ylim=(-2, 2))
ax1.xaxis.set_ticks([]) # suppress axis ticks & numbers
ax1.legend()

ax2 = fig.add_subplot(gs[1, 0]) # second plot in second 1/4
ax2.plot(x, np.sin(x), label=r'$y = \sin(x)$')
ax2.set(xlim=xlim, ylim=(-2, 2))
ax2.xaxis.set_ticks([]) # suppress axis ticks & numbers
ax2.legend()

ax3 = fig.add_subplot(gs[2:4, 0]) # third plot in bottom 1/2
ax3.plot(x, (-2 + x/3) + np.sin(x),
label=r'$y=\left(-2 + \frac{x}{3}\right) + \sin(x)$')
ax3.set(xlim=xlim, ylim=(-2, 6))
ax3.legend()
```

To learn more, see Pyplot tutorial and Pyplot examples.

Plots of several variables from a pandas `DataFrame`

e.g. `df = pd.read_csv('http://www.stat.wisc.edu/~jgillett/451/data/DJIA.csv', index_col='Symbol')`

- Use `df.groupby()` for a plot grouped by a categorical variable:
  - line plots: `df.groupby('Exchange')['AvgVol'].plot.line(xticks=[]); plt.legend() # silly`
  - histograms: `df.groupby('Exchange')['AvgVol'].plot.hist(alpha=.5); plt.legend()`
  - density plots: `df.groupby('Exchange')['AvgVol'].plot.density(); plt.legend()`
- Plot a `column` variable grouped by a `by` categorical variable:
  - boxplots: `df.boxplot(column='AvgVol', by='Exchange')`
  - histograms:
    ```
    df.hist(column='AvgVol', by='Exchange', bins=10,
                        sharex=True, edgecolor='white', layout=(2, 1))
    ```
- barplot, one bar or bar cluster per group:

  `df['State'].value_counts().plot.bar(rot=20) # try .barh()`
- scatter plot, colored by `c` argument:

  `df.plot.scatter(x='MarketCap', y='AvgVol', c='Price', cmap='magma') # cmap = 'colormap'`
- scatter matrix plot: `pd.plotting.scatter_matrix(df[['Price', 'MarketCap', 'AvgVol']])`
- plot all columns of a `DataFrame` with `df.plot()`:

```
# make n_series fake time series to get data suitable for showing df.plot()
n_days = 100
df = pd.DataFrame(data=np.random.randn(n_days, 3), # random numbers from N(0, 1)
                  index=pd.date_range('1/1/2020', periods=n_days), # date sequence
                  columns=list('ABC'))
df = df.cumsum()
df.plot()
```

Font size

Use a readable font size (close to text size) in presentations and reports.

e.g. Set all fontsizes to 10 (the default) via `plt.rcParams.update({'font.size': 10})`. Other values are relative to 10.

To learn more, see pandas visualization.