Formatting Plots

About the Data

In this notebook, we will be working with Facebook's stock price throughout 2018 (obtained using the stock_analysis package).

Setup

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import seaborn as sns

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

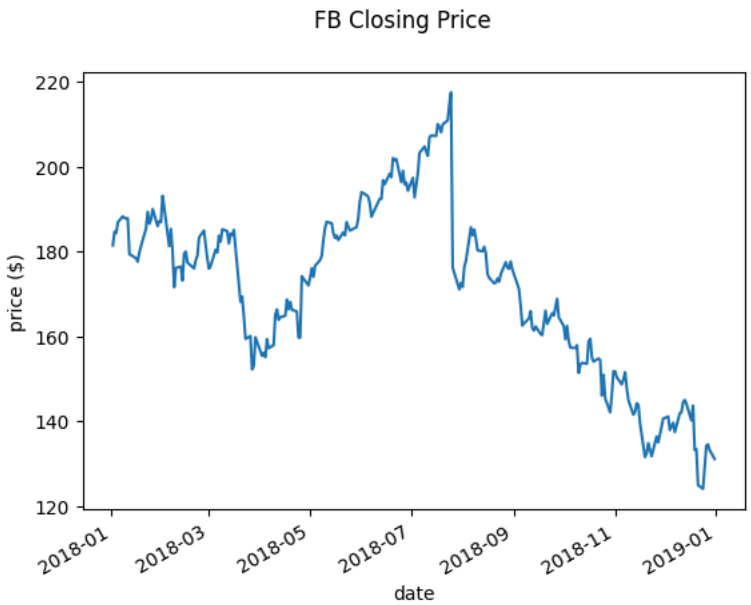
251 rows × 5 columns

Tiles and Axis Labels

- plt.suptitle() adds a title to plots and subplots
- plt.title() adds a title to a single plot. Note if you use subplots, it will only put the title on the last subplot, so you will need to use plt.suptitle()
- plt.xlabel() labels the x-axis
- plt.ylabel() labels the y-axis

```
fb.close.plot()
plt.suptitle('FB Closing Price')
plt.xlabel('date')
plt.ylabel('price ($)')
```

```
Text(0, 0.5, 'price ($)')
```

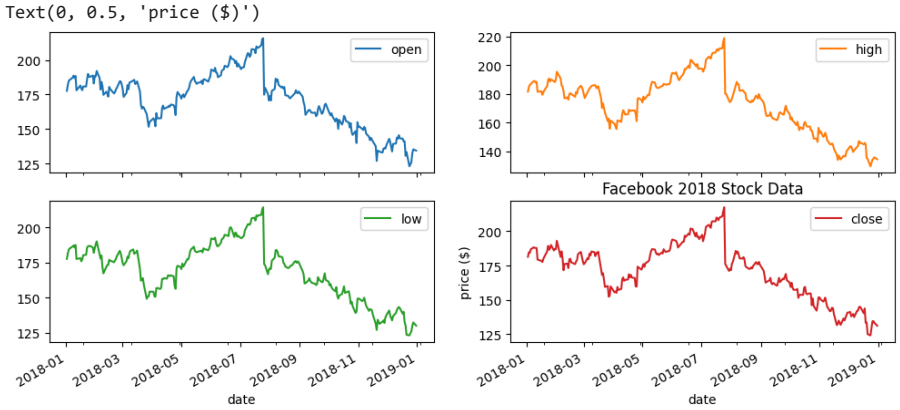


plt.suptitle() vs plt.title()

Check out what happens when we call plt.title() with subplots:

```
fb.iloc[:,:4].plot(subplots=True, layout=(2,2), figsize=(12,5))
plt.title('Facebook 2018 Stock Data')
plt.xlabel('date')
plt.ylabel('price ($)')
```

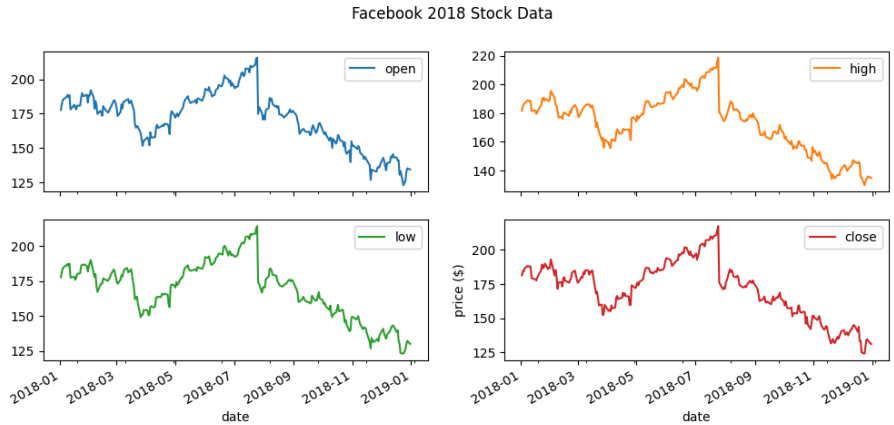
```
Text(0, 0.5, 'price ($)')
```



Simply getting into the habit of using plt.suptitle() instead of plt.title() will save you this confusion:

```
fb.iloc[:, :4].plot(subplots=True, layout=(2,2), figsize=(12,5))
plt.suptitle('Facebook 2018 Stock Data')
plt.xlabel('date')
plt.ylabel('price ($)')
```

```
Text(0, 0.5, 'price ($)')
```

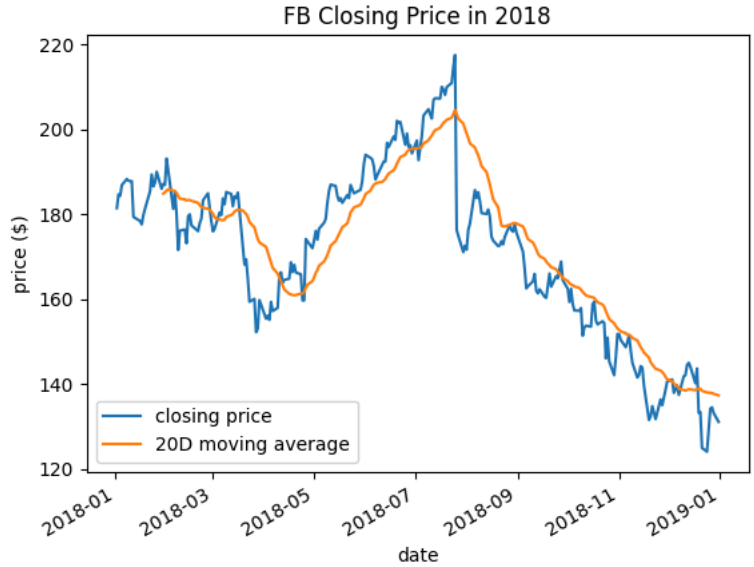


Legend

plt.legend() adds a legend to the plot. We can specify where to place it with the loc parameter:

```
fb.assign(
    ma = lambda x:x.close.rolling(20).mean()
).plot(
    y = ['close', 'ma'],
    title = 'FB Closing Price in 2018',
    label = ['closing price', '20D moving average']
)
plt.legend(loc = 'lower left')
plt.ylabel('price ($)')
```

```
Text(0, 0.5, 'price ($)')
```



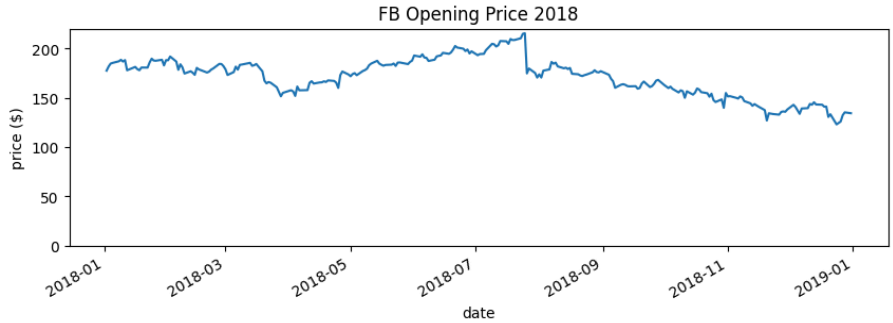
Formatting Axes

Specifying axis limits

plt.xlim() and plt.ylim() can be used to specify the minimum and maximum values for the axis. Passing None will have matplotlib determine the limit.

```
fb.open.plot(figsize=(10,3), title= 'FB Opening Price 2018')
plt.ylim(0, None)
plt.ylabel('price ($)')
```

```
Text(0, 0.5, 'price ($)')
```



Formatting The Axis Ticks

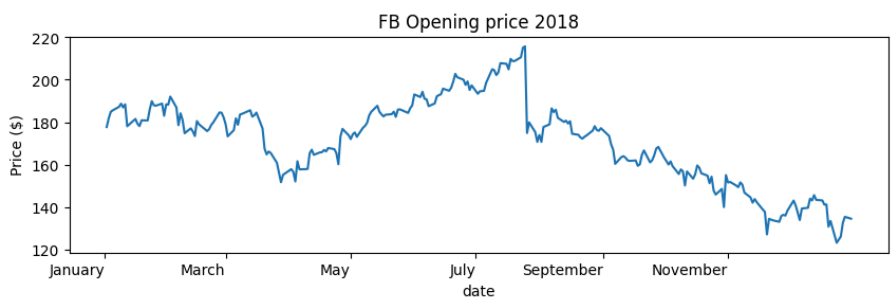
We can use plt.xticks() and plt.yticks() to provide tick labels and specify, which ticks to show. Here, we show every other month:

```
import calendar

fb.open.plot(figsize = (10, 3), rot= 0, title='FB Opening price 2018')
locs, labels = plt.xticks()
plt.xticks(locs + 15, calendar.month_name[1::2])
plt.ylabel('price ($)')
```

```
import calendar

fb.open.plot(figsize=(10, 3), rot=0, title='FB Opening price 2018')
plt.ylabel('Price ($)')
locs, labels = plt.xticks()
plt.xticks(locs[:-1], calendar.month_name[1::2])
plt.show()
```



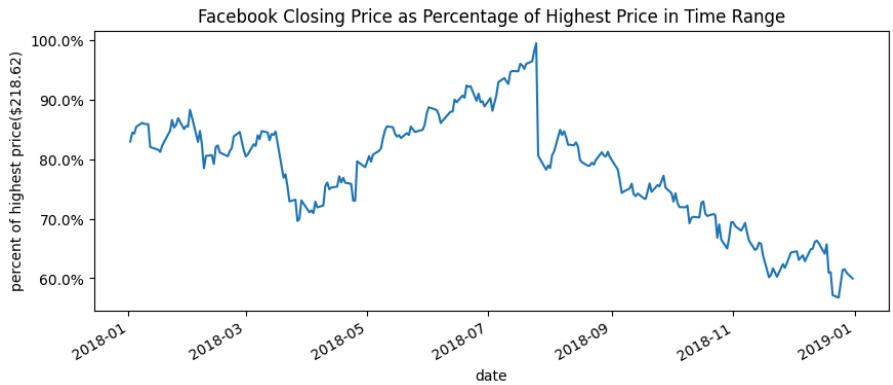
PercentFormatter

We can use ticker.PercentFormatter and specify the denominator (xmax) to use when calculating the percentages. This gets passed to the set_major_formatter() method of the xaxis or yaxis on the Axes

```
import matplotlib.ticker as ticker

ax = fb.close.plot(
    figsize=(10,4),
    title= 'Facebook Closing Price as Percentage of Highest Price in Time Range'
)
ax.yaxis.set_major_formatter(
    ticker.PercentFormatter(xmax = fb.high.max())
)
ax.set_yticks(
    [fb.high.max()* pct for pct in np.linspace(0.6, 1, num=5)]
)
ax.set_ylabel(f'percent of highest price(${fb.high.max()})')
```

```
Text(0, 0.5, 'percent of highest price($218.62)')
```

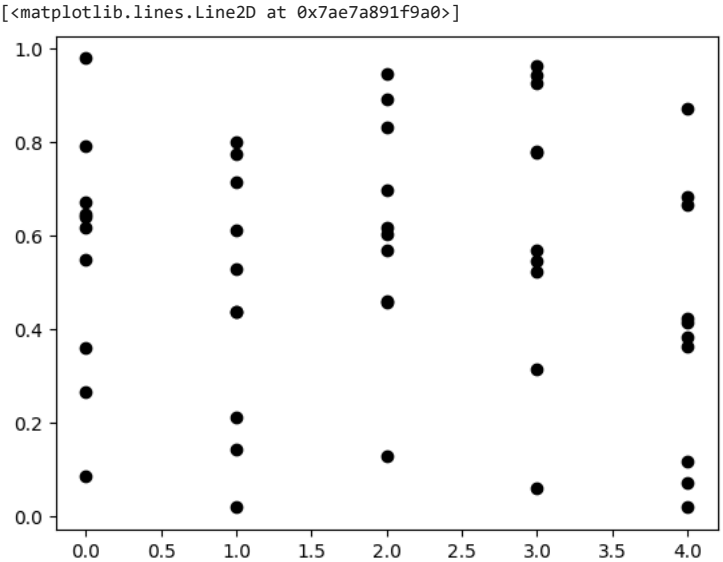


MultipleLocator

Say we have the following data. The points only take on integer values for x.

```
fig, ax = plt.subplots(1,1)
np.random.seed(0)
ax.plot(np.tile(np.arange(0,5), 10), np.random.rand(50), 'ko')
```

```
[<matplotlib.lines.Line2D at 0x7ae7a891f9a0>]
```



If we don't want to show decimal values on the x-axis, we can use the MultipleLocator . This will give ticks for all multiples of a number specified with the base parameter. To get integer values, we use base=1 :

```
fig, ax = plt.subplots(1, 1)
np.random.seed(0)
ax.plot(np.tile(np.arange(0, 5), 10), np.random.rand(50), 'ko')
ax.get_xaxis().set_major_locator(
ticker.MultipleLocator(base=1)
)
```