Formatting Plots

Name: Cuadra, Audrick Zander G.

Section: CPE22S3

Date: March 30, 2024

Submitted to: Engr. Roman Richard

About the Data

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

Setup

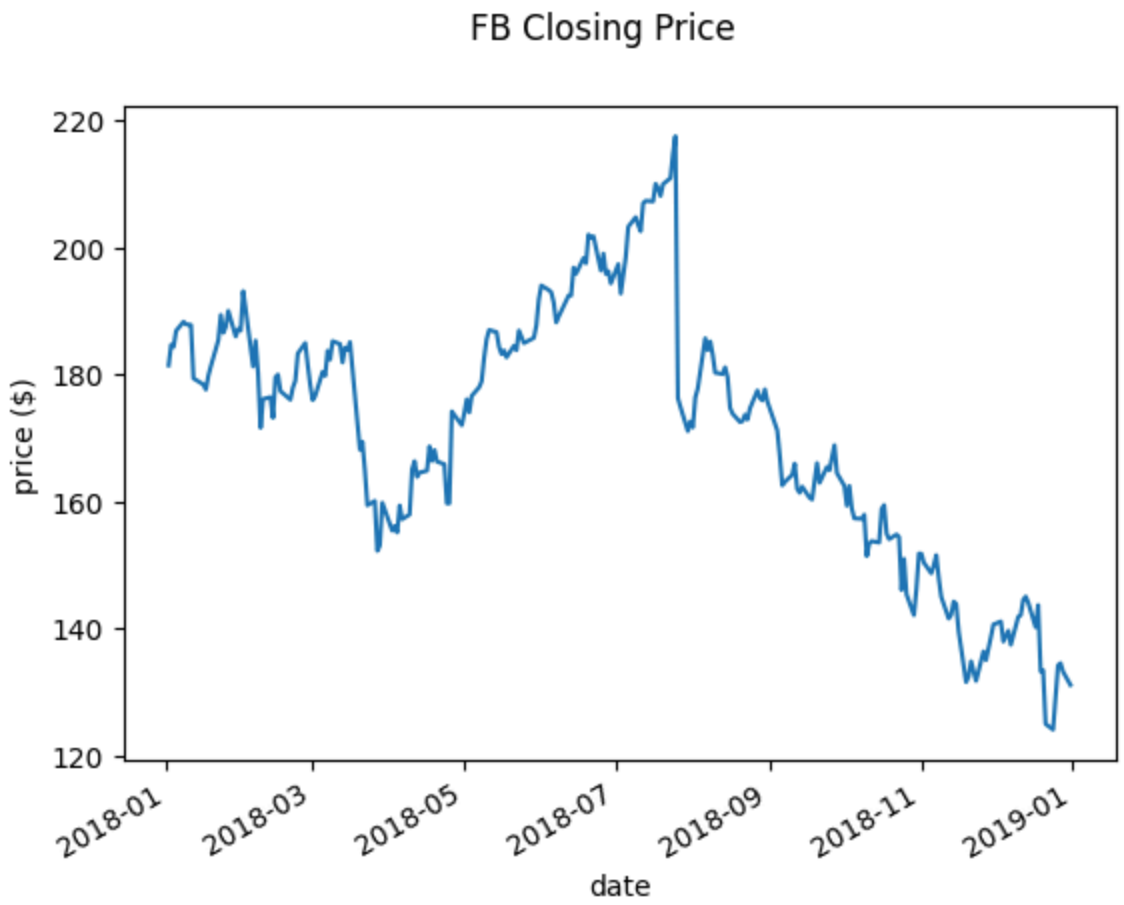
```
%matplotlib inline
import matplotlib.pyplot as mpl
import numpy as ny
import pandas as p
import seaborn as sbn

fb = p.read_csv(
    '/content/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
```

Titles 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()
mpl.suptitle('FB Closing Price')
mpl.xlabel('date')
mpl.ylabel('price ($)')
```

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

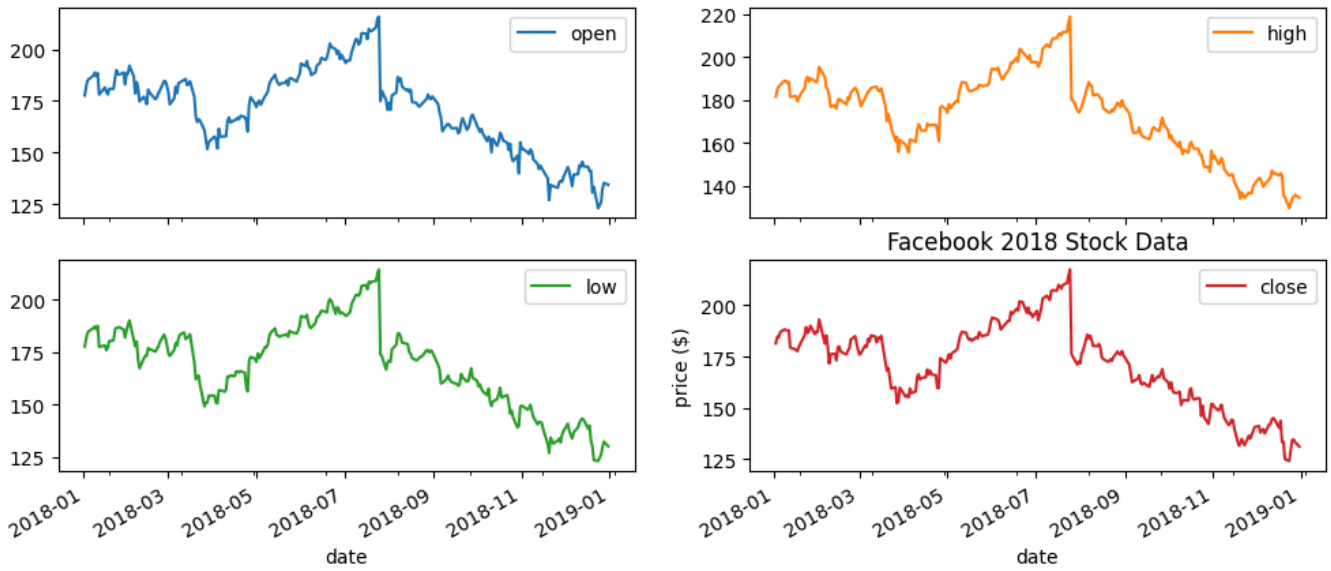


plt.suptitle() vs. plt.title()

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

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

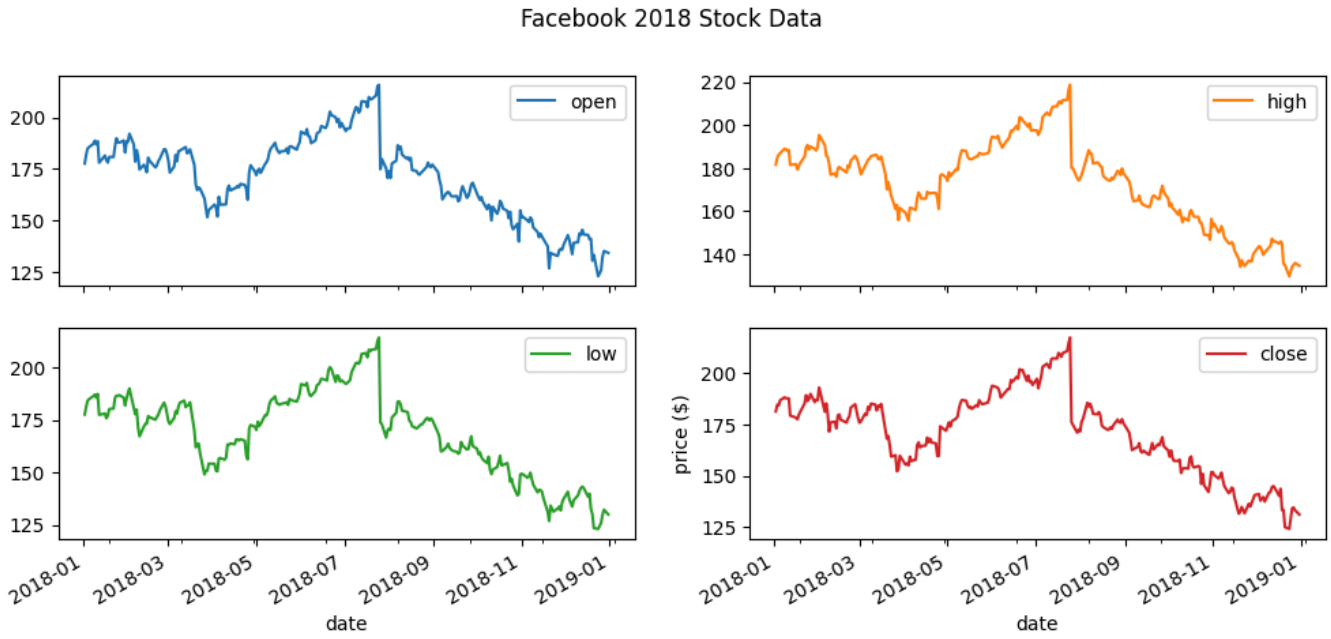
```
Text(0, 0.5, 'price ($)')
```



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

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

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

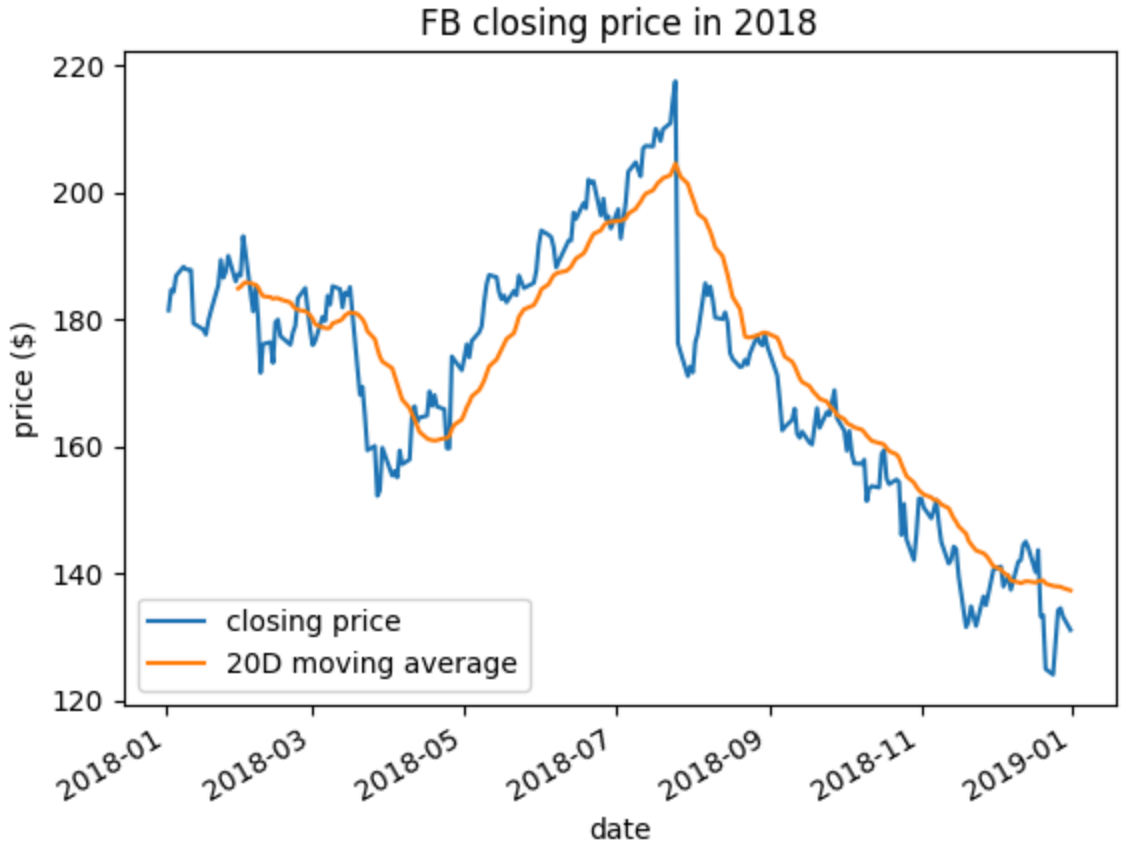


Legends

`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']
)
mpl.legend(loc='lower left')
mpl.ylabel('price ($)')
```

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



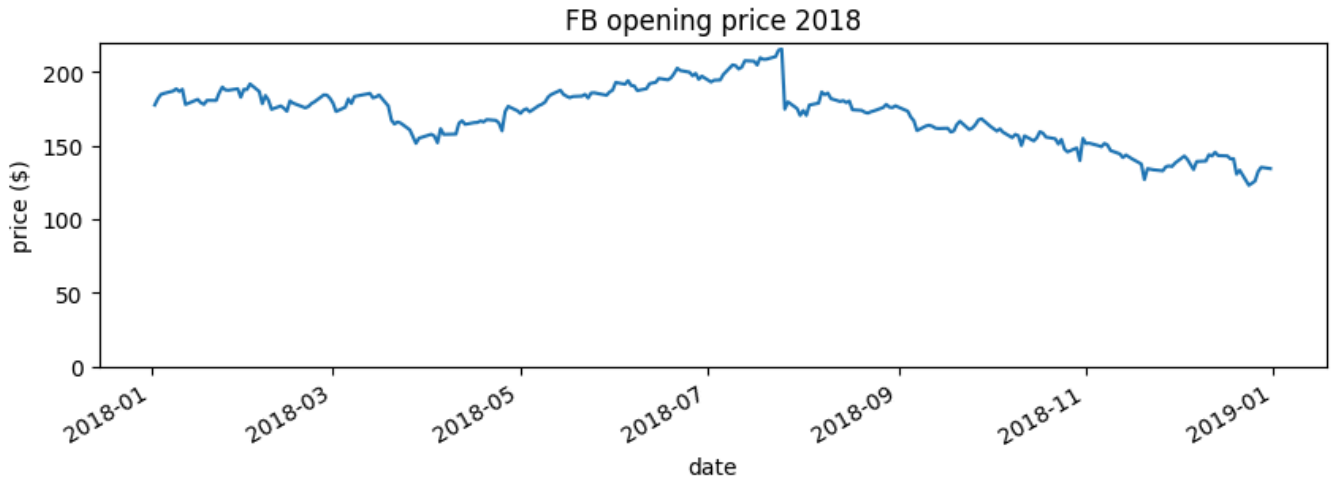
Formatting Axes

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.

```
1 fb.open.plot(figsize=(10, 3), title='FB opening price 2018')
2 mpl.ylim(0, None)
3 mpl.ylabel('price ($)')
```

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

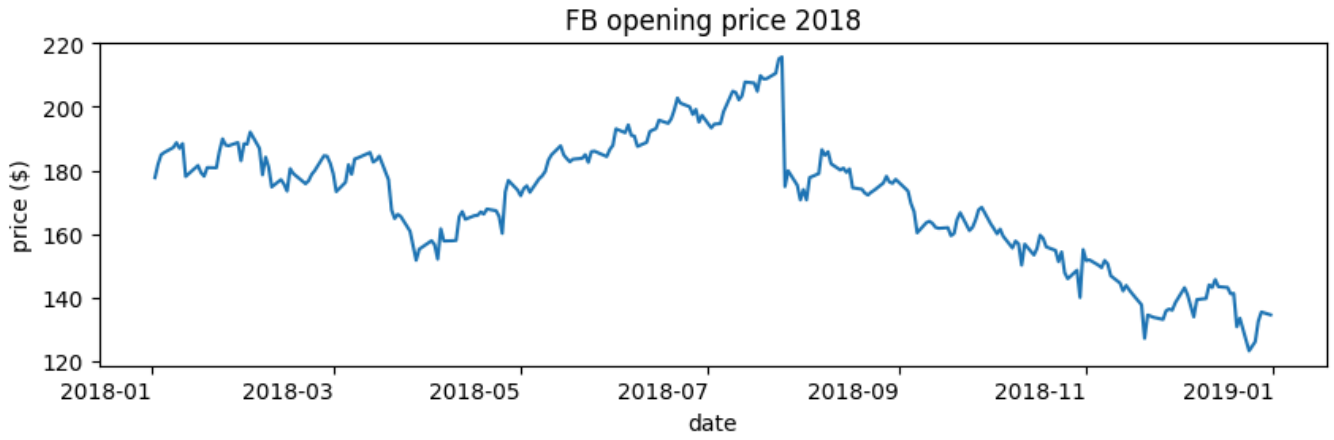


Formatting the Axis Ticks

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 = mpl.xticks()
new_locs = locs + 15
new_labels = calendar.month_name[1::2][:len(new_locs)]
#mpl.xticks(locs + 15 , calendar.month_name[1::2])
mpl.ylabel('price ($)')
```

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



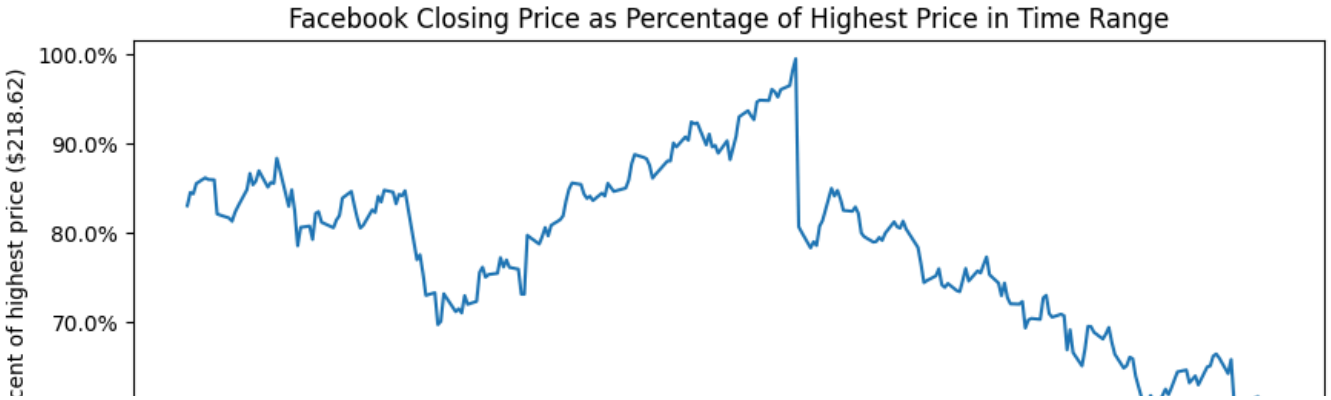
Using `ticker`

`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 th `e xaxis o r yaxis` on th `e 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 ny.linspace(0.6, 1, num=5)
]) # show round percentages only (60%, 80%, etc.)
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`.


```
1 fig, ax = mpl.subplots(1, 1)
2 ny.random.seed(0)
3 ax.plot(ny.tile(ny.arange(0, 5), 10), ny.random.rand(50), 'ko')
```

```
[<matplotlib.lines.Line2D at 0x78f06a8b1480>]
```