Hands-on Activity 9.1 Data Visualization using Pandas and Matplotlib

Submitted by: Eugene D.G. Dela Cruz

Submitted to: Engr. Roman Richard

9.1 Introduction to Matplotlib

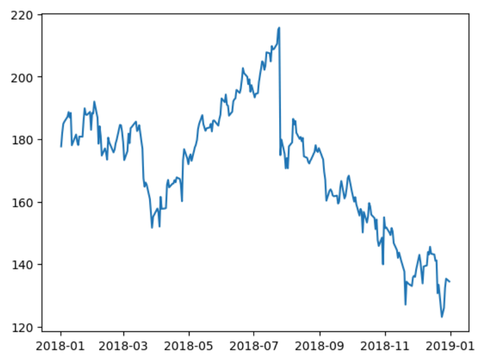
Getting Started with Matplotlib

We need matplotlib.pyplot for plotting.

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

Plotting lines

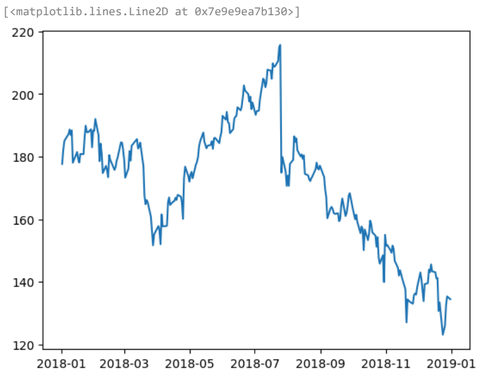
```
fb = pd.read_csv(
    '/content/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
                 )
plt.plot(fb.index, fb.open)
plt.show()
```



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

```
%matplotlib inline
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)
# the %matplotlib inline uses the command ".show()" command without calling the command itself when running the cell.
```

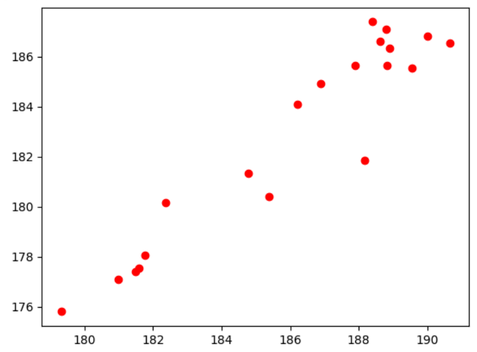
```
[<matplotlib.lines.Line2D at 0x7e9e9ea7b130>]
```



Scatter plots

```
plt.plot('high', 'low', 'ro', data=fb.head(20))
# calling the scatter plot with its data population
```

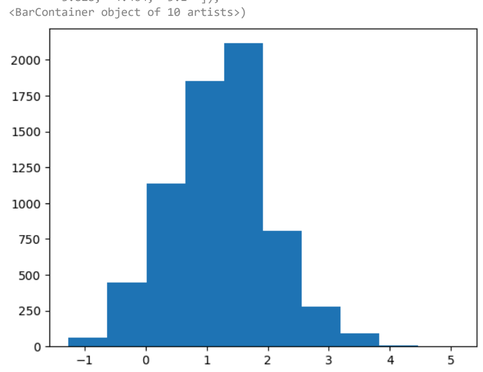
```
[<matplotlib.lines.Line2D at 0x7e9e9eb5c4c0>]
```



Histograms

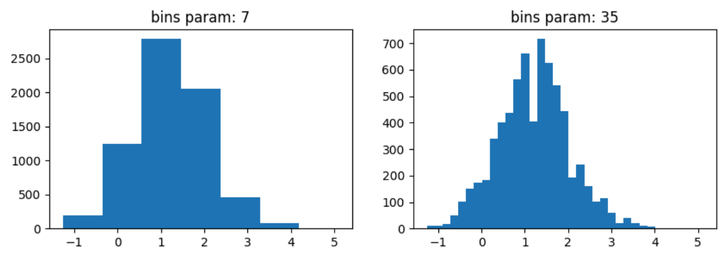
```
quakes = pd.read_csv('/content/earthquakes-1.csv')
plt.hist(quakes.query('magType == "ml"').mag)
# create histogram if magtype filtered by ml
```

```
(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>)
```



Bin size matters

```
x = quakes.query('magType == "ml"').mag
fig, axes = plt.subplots(1, 2, figsize=(10, 3)) #create fig with subplots and bin size
for ax, bins in zip(axes, [7, 35]):
 ax.hist(x, bins=bins) #creates histogram with bin size
 ax.set_title(f'bins param: {bins}') #set title
```



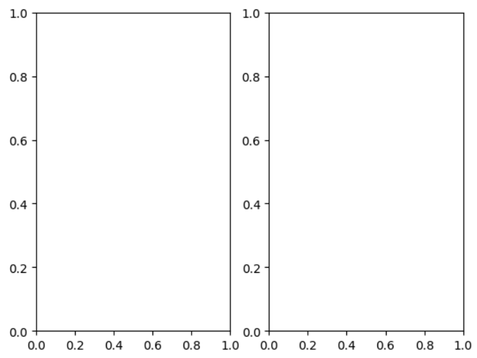
Plot components

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

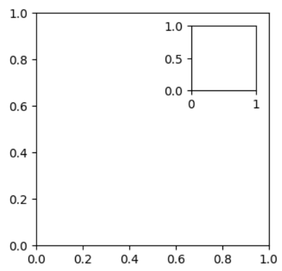
```
<Figure size 640x480 with 0 Axes>
```

Creating subplots

```
fig, axes = plt.subplots(1, 2) #create figure with its rows and column size
```

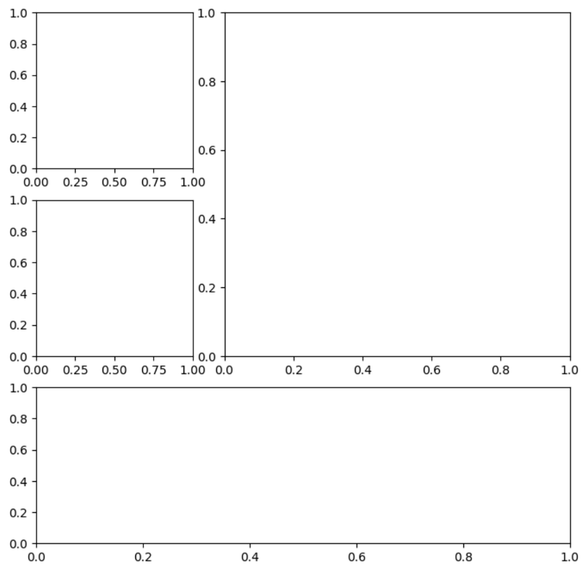


```
fig = plt.figure(figsize=(3, 3))
outside = fig.add_axes([0.1, 0.1, 0.9, 0.9]) #adding subplot to figure
inside = fig.add_axes([0.7, 0.7, 0.25, 0.25]) #create subplot inside the outer subplot
```



Creating Plot Layouts with gridspec

```
fig = plt.figure(figsize=(8, 8)) #create figure
gs = fig.add_gridspec(3, 3) #add grid that has a 3x3 layout
top_left = fig.add_subplot(gs[0, 0]) #adding subplots to the figure with its specific position
mid_left = fig.add_subplot(gs[1, 0]) #adding subplots to the figure with its specific position
top_right = fig.add_subplot(gs[:2, 1:]) #adding subplots to the figure with its specific position
bottom = fig.add_subplot(gs[2,:]) #adding subplots to the figure with its specific position
```



Saving plots

```
fig.savefig('empty.png') #this saves specified figure object
```

Cleaning up

```
plt.close('all')
```

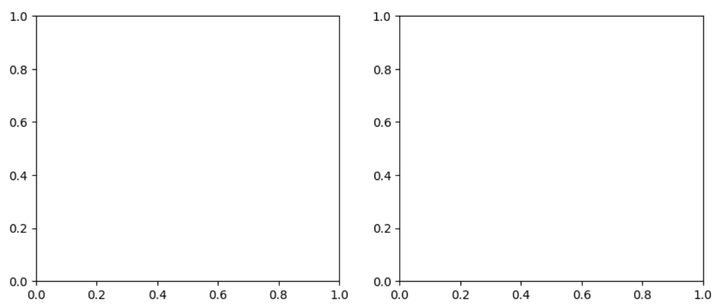
Additional plotting options

Specifying figure size

```
fig = plt.figure(figsize=(10, 4))
```

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

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



rcParams

```
import random
import matplotlib as mpl
rcparams_list = list(mpl.rcParams.keys()) #create list of all available keys
random.seed(20) # make this repeatable
random.shuffle(rcparams_list) #shuffling the list of keys
sorted(rcparams_list[:20]) #sort and display the first 20 selected keys
```

```
['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']
```

```
mpl.rcParams['figure.figsize']
```

```
[6.4, 4.8]
```

```
mpl.rcParams['figure.figsize'] = (300, 10)
mpl.rcParams['figure.figsize']
```

```
[300.0, 10.0]
```

```
mpl.rcdefaults()
mpl.rcParams['figure.figsize']
```

```
[6.4, 4.8]
```

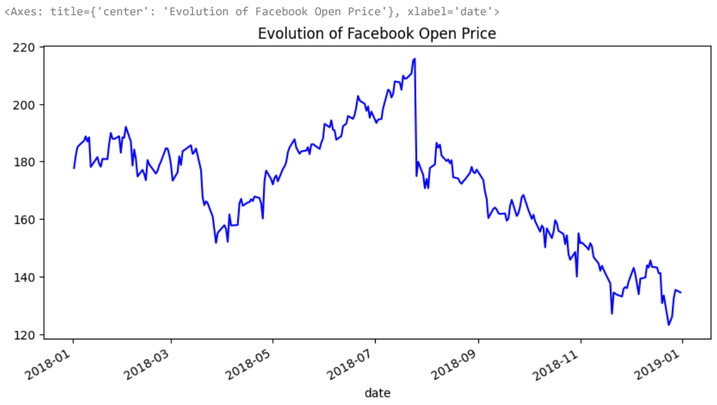
```
plt.rc('figure', figsize=(20, 20)) # change the figure size to (20, 20)
plt.rcdefaults() # reset default
```

9.2 Plotting with Pandas

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
fb = pd.read_csv(
 '/content/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
quakes = pd.read_csv('/content/earthquakes-1.csv')
```

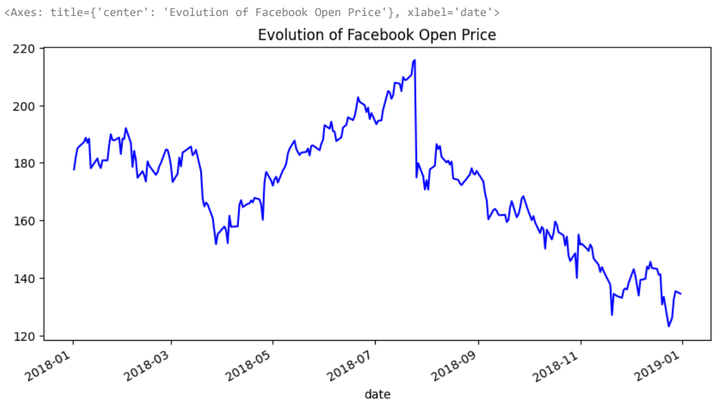
```
fb.plot(
 kind='line',
 y='open',
 figsize=(10, 5),
 style='b-',
 legend=False,
 title='Evolution of Facebook Open Price'
)
```

```
<Axes: title={'center': 'Evolution of Facebook Open Price'}, xlabel='date'>
```



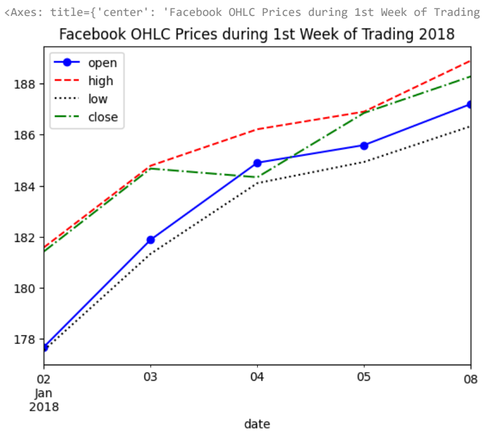

```
fb.plot(
 kind='line',
 y='open',
 figsize=(10, 5),
 color='blue',
 linestyle='solid',
 legend=False,
 title='Evolution of Facebook Open Price'
)
#plotting a line chart for open price
```

```
<Axes: title={'center': 'Evolution of Facebook Open Price'}, xlabel='date'>
```




```
fb.iloc[:5,].plot(
 y=['open', 'high', 'low', 'close'],
 style=['b-o', 'r--', 'k:', 'g-.'],
 title='Facebook OHLC Prices during 1st Week of Trading 2018'
)
#plotting the open high low close price for the first 5 row
```

```
<Axes: title={'center': 'Facebook OHLC Prices during 1st Week of Trading 2018'}, xlabel='date'>
```

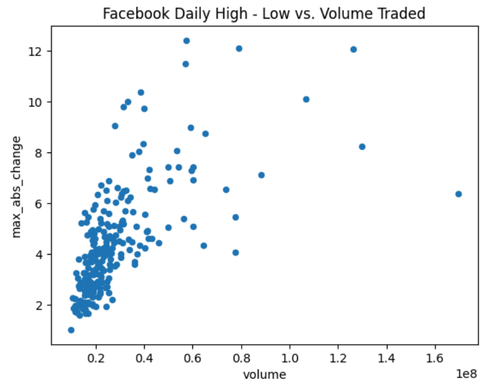



```
fb.plot(
 kind='line',
 subplots=True,
 layout=(3,2),
 figsize=(15,10),
 title='Facebook Stock 2018'
) #plotting multiple subplots for different columns of the data
```

```
array([[<Axes: xlabel='date'>, <Axes: xlabel='date'>],
       [<Axes: xlabel='date'>, <Axes: xlabel='date'>],
       [<Axes: xlabel='date'>, <Axes: xlabel='date'>]], dtype=object)
```

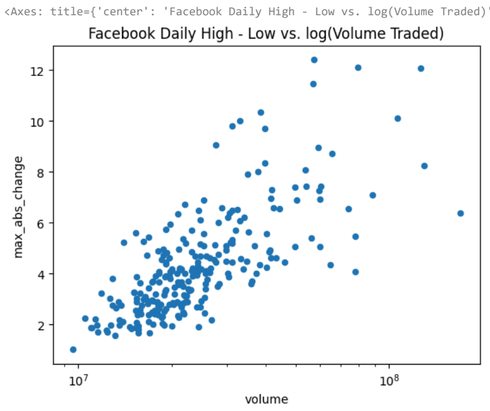

```
fb.assign(
 max_abs_change=fb.high - fb.low
).plot(
 kind='scatter', x='volume', y='max_abs_change',
 title='Facebook Daily High - Low vs. Volume Traded'
)
#adding new column to the dataframe
```

```
<Axes: title={'center': 'Facebook Daily High - Low vs. Volume Traded'}, xlabel='volume', ylabel='max_abs_change'>
```



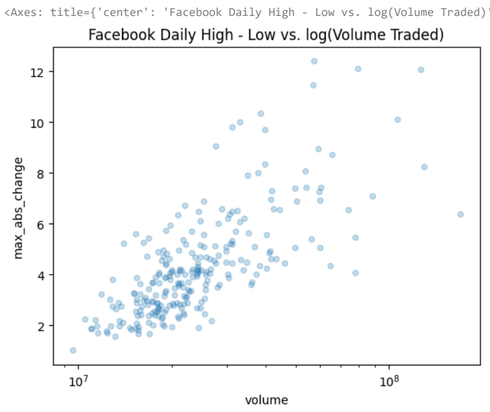

```
fb.assign(
 max_abs_change=fb.high - fb.low
).plot(
 kind='scatter', x='volume', y='max_abs_change',
 title='Facebook Daily High - Low vs. log(Volume Traded)',
 logx=True
) #adding new column to dataframe
```

```
<Axes: title={'center': 'Facebook Daily High - Low vs. log(Volume Traded)'}, xlabel='volume', ylabel='max_abs_change'>
```



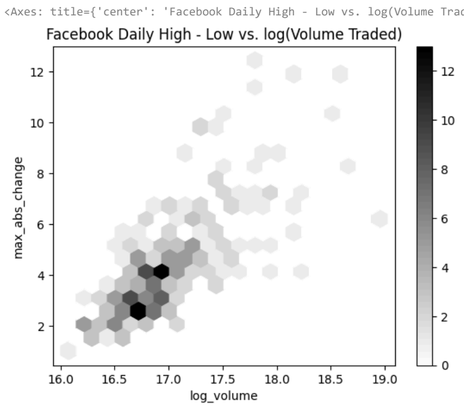

```
fb.assign(
 max_abs_change=fb.high - fb.low
).plot(
 kind='scatter', x='volume', y='max_abs_change',
 title='Facebook Daily High - Low vs. log(Volume Traded)',
 logx=True, alpha=0.25
) # adding new column to dataframe
```

```
<Axes: title={'center': 'Facebook Daily High - Low vs. log(Volume Traded)'}, xlabel='volume', ylabel='max_abs_change'>
```




```
fb.assign(
 log_volume=np.log(fb.volume),
 max_abs_change=fb.high - fb.low
).plot(
 kind='hexbin',
 x='log_volume',
 y='max_abs_change',
 title='Facebook Daily High - Low vs. log(Volume Traded)',
 colormap='gray_r',
 gridsize=20,
 sharex=False # we have to pass this to see the x-axis due to a bug in this version of pandas
)
#adding new columns and calculates the log of volume & absolute change between high and low price
# plotting the data to x and y axis
```

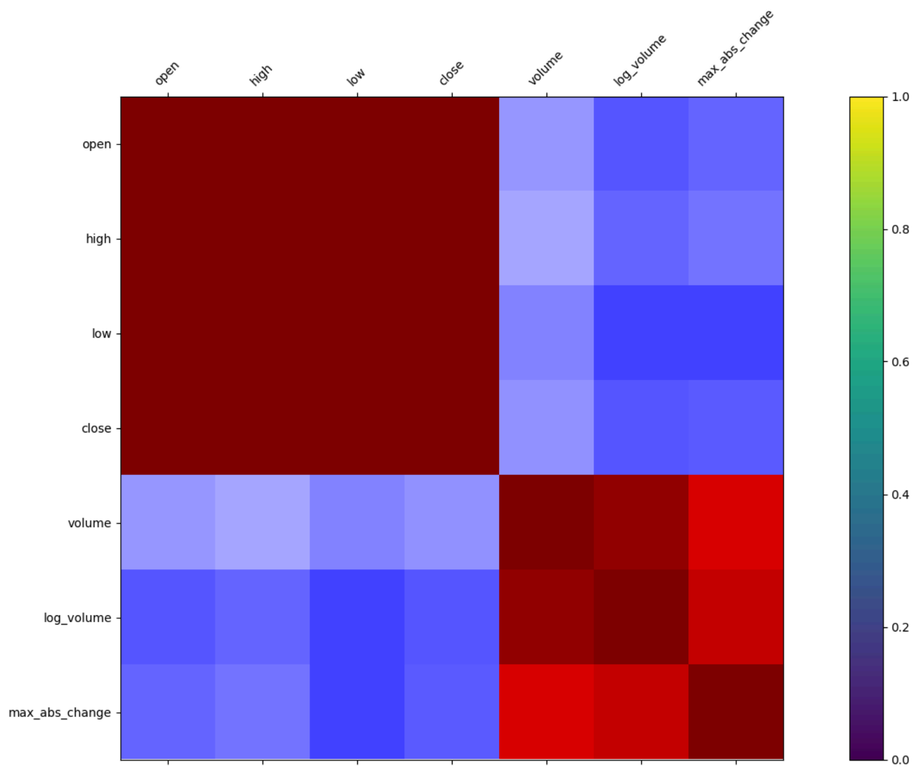
```
<Axes: title={'center': 'Facebook Daily High - Low vs. log(Volume Traded)'}, xlabel='log_volume', ylabel='max_abs_change'>
```




```
fig, ax = plt.subplots(figsize=(20, 10))

fb_corr = fb.assign(
  log_volume=np.log(fb.volume),
  max_abs_change=fb.high - fb.low
).corr()
```

```
im = ax.matshow(fb_corr, cmap='seismic')
fig.colorbar(im.set_clim(-1, 1))

labels = [col.lower() for col in fb_corr.columns]
ax.set_xticklabels([''] + labels, rotation=45)
ax.set_yticklabels([''] + labels)
```

```
<ipython-input-29-a062f5cc3939>:9: MatplotlibDeprecationWarning: Unable to determine Axes to steal space for Colorbar. Using gca(), but will raise in the future. Either provide the *cax* argument to use as the Axes for the Colorbar, provide the *ax* argument to steal space from it, or add *mappable* to an Axes.
  fig.colorbar(im.set_clim(-1, 1))
<ipython-input-29-a062f5cc3939>:12: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels([''] + labels, rotation=45)
<ipython-input-29-a062f5cc3939>:13: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_yticklabels([''] + labels)
[Text(0, -1.0, ''),
 Text(0, 0.0, 'open'),
 Text(0, 1.0, 'high'),
 Text(0, 2.0, 'low'),
 Text(0, 3.0, 'close'),
 Text(0, 4.0, 'volume'),
 Text(0, 5.0, 'log_volume'),
 Text(0, 6.0, 'max_abs_change'),
 Text(0, 7.0, '')]
```

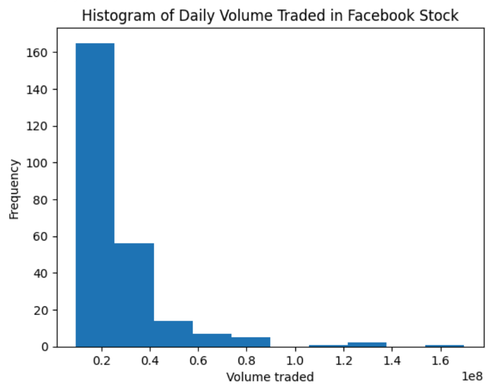



```
fb_corr.loc['max_abs_change', ['volume', 'log_volume']]
```

```
volume        0.642027
log_volume    0.731542
Name: max_abs_change, dtype: float64
```

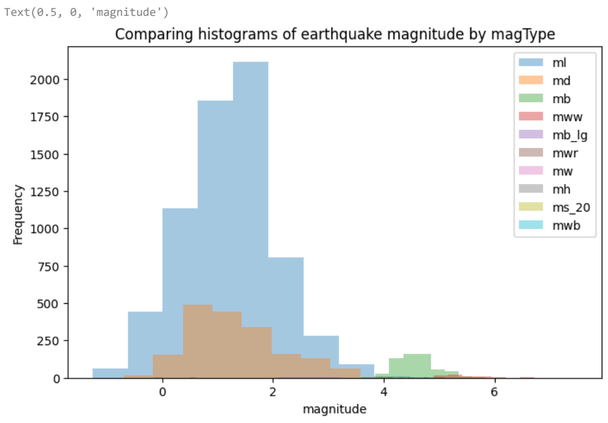
```
fb.volume.plot(
 kind='hist',
 title='Histogram of Daily Volume Traded in Facebook Stock'
)
plt.xlabel('Volume traded') # label the x-axis (discussed in chapter 6)
```

```
Text(0.5, 0, 'Volume traded')
```



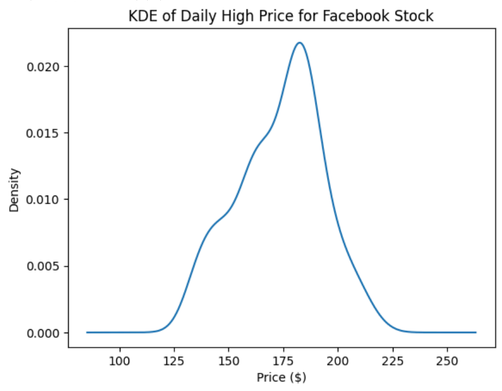

```
fig, axes = plt.subplots(figsize=(8, 5))
for magtype in quakes.magType.unique():
  data = quakes.query(f'magType == "{magtype}"').mag
  if not data.empty:
    data.plot(
      kind='hist', ax=axes, alpha=0.4,
      label=magtype, legend=True,
      title='Comparing histograms of earthquake magnitude by magType'
  )
plt.xlabel('magnitude') # label the x-axis (discussed in chapter 6)
```

```
Text(0.5, 0, 'magnitude')
```



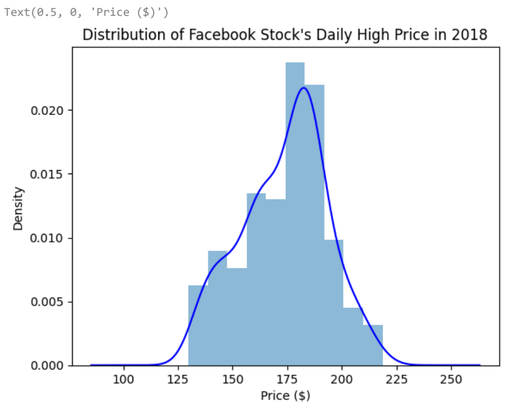

```
fb.high.plot(
 kind='kde',
 title='KDE of Daily High Price for Facebook Stock'
)
plt.xlabel('Price ($)') # label the x-axis (discussed in chapter 6)
```

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



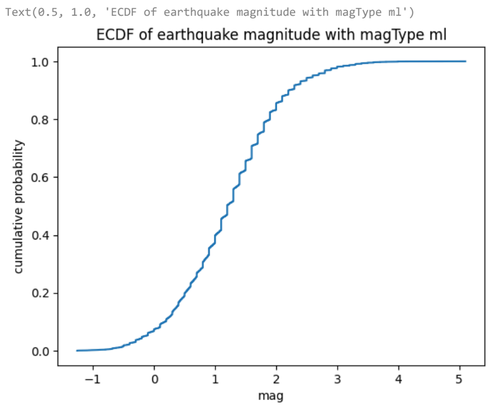

```
ax = fb.high.plot(kind='hist', density=True, alpha=0.5)
fb.high.plot(
 ax=ax, kind='kde', color='blue',
 title='Distribution of Facebook Stock\'s Daily High Price in 2018'
)
plt.xlabel('Price ($)') # label the x-axis (discussed in chapter 6)
```

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



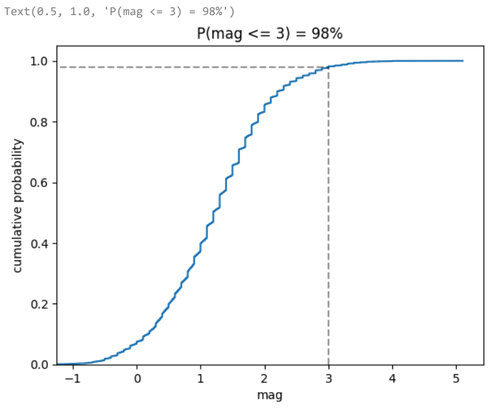

```
from statsmodels.distributions.empirical_distribution import ECDF
ecdf = ECDF(quakes.query('magType == "ml"').mag)
plt.plot(ecdf.x, ecdf.y)
# axis labels (we will cover this in chapter 6)
plt.xlabel('mag') # add x-axis label
plt.ylabel('cumulative probability') # add y-axis label
# add title (we will cover this in chapter 6)
plt.title('ECDF of earthquake magnitude with magType ml')
```

```
Text(0.5, 1.0, 'ECDF of earthquake magnitude with magType ml')
```



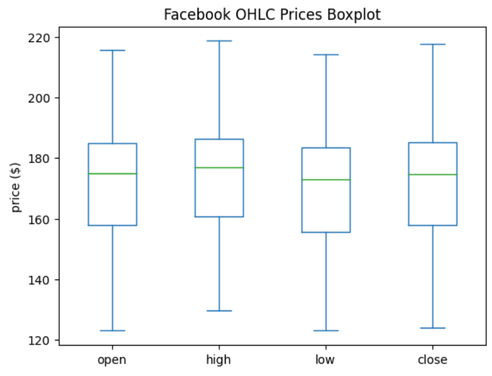

```
from statsmodels.distributions.empirical_distribution import ECDF
ecdf = ECDF(quakes.query('magType == "ml"').mag)
plt.plot(ecdf.x, ecdf.y)
# formatting below will all be covered in chapter 6
# axis labels
plt.xlabel('mag') # add x-axis label
plt.ylabel('cumulative probability') # add y-axis label
# add reference lines for interpreting the ECDF for mag <= 3
plt.plot(
 [3, 3], [0, .98], 'k--',
 [-1.5, 3], [0.98, 0.98], 'k--', alpha=0.4
)
# set axis ranges
plt.ylim(0, None)
plt.xlim(-1.25, None)
# add a title
plt.title('P(mag <= 3) = 98%')
```

```
Text(0.5, 1.0, 'P(mag <= 3) = 98%')
```



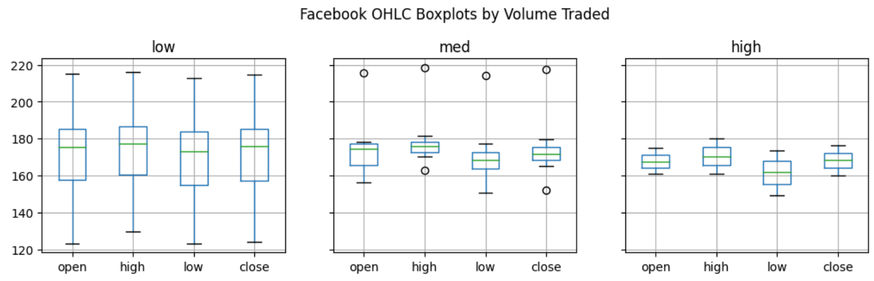

```
fb.iloc[:,:4].plot(kind='box', title='Facebook OHLC Prices Boxplot')
plt.ylabel('price ($)') # label the x-axis (discussed in chapter 6)
```

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



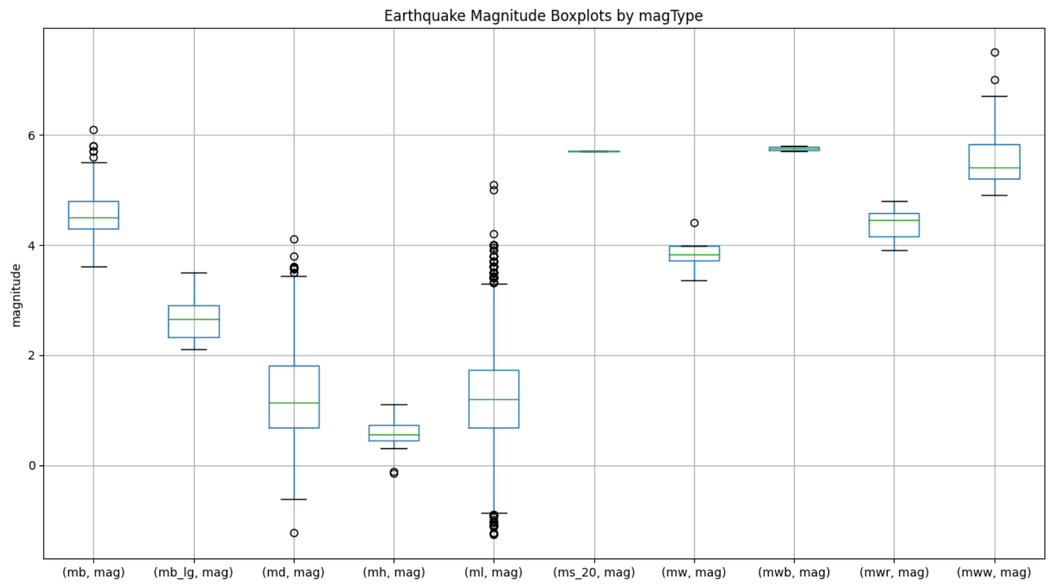

```
fb.assign(
 volume_bin=pd.cut(fb.volume, 3, labels=['low', 'med', 'high'])
).groupby('volume_bin').boxplot(
 column=['open', 'high', 'low', 'close'],
 layout=(1, 3), figsize=(12, 3)
)
plt.suptitle('Facebook OHLC Boxplots by Volume Traded', y=1.1)
```

```
Text(0.5, 1.1, 'Facebook OHLC Boxplots by Volume Traded')
```



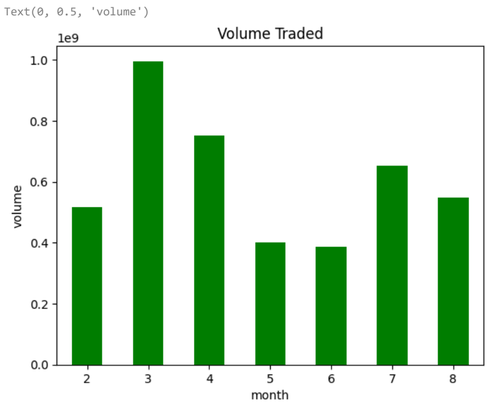

```
quakes[['mag', 'magType']].groupby('magType').boxplot(
 figsize=(15, 8), subplots=False
)
plt.title('Earthquake Magnitude Boxplots by magType')
plt.ylabel('magnitude') # label the y-axis (discussed in chapter 6)
```

```
Text(0, 0.5, 'magnitude')
```



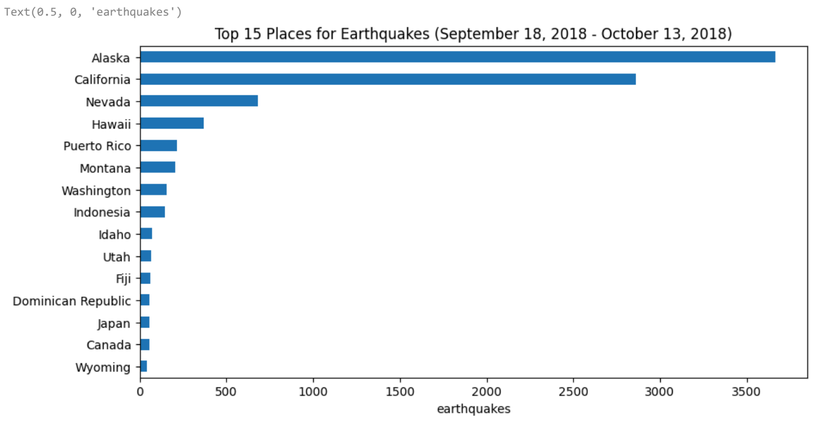

```
fb['2018-02':'2018-08'].assign(
 month=lambda x: x.index.month
).groupby('month').sum().volume.plot.bar(
 color='green', rot=0, title='Volume Traded'
 )
plt.ylabel('volume') # label the y-axis (discussed in chapter 6)
```

```
Text(0, 0.5, 'volume')
```



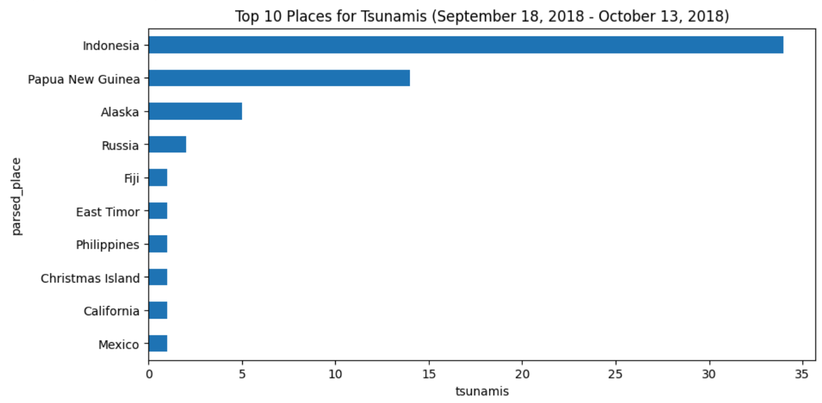

```
quakes.parsed_place.value_counts().iloc[14::-1,].plot(
 kind='barh', figsize=(10, 5),
 title='Top 15 Places for Earthquakes '\
 '(September 18, 2018 - October 13, 2018)'
)
plt.xlabel('earthquakes') # label the x-axis (discussed in chapter 6)
```

```
Text(0.5, 0, 'earthquakes')
```



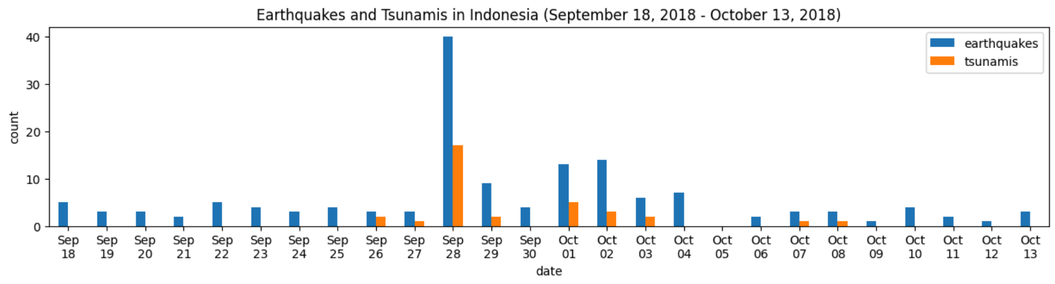

```
quakes.groupby('parsed_place').tsunami.sum().sort_values().iloc[-10::,].plot(
 kind='barh', figsize=(10, 5),
 title='Top 10 Places for Tsunamis '\
 '(September 18, 2018 - October 13, 2018)'
)
plt.xlabel('tsunamis') # label the x-axis (discussed in chapter 6)
```

```
Text(0.5, 0, 'tsunamis')
```



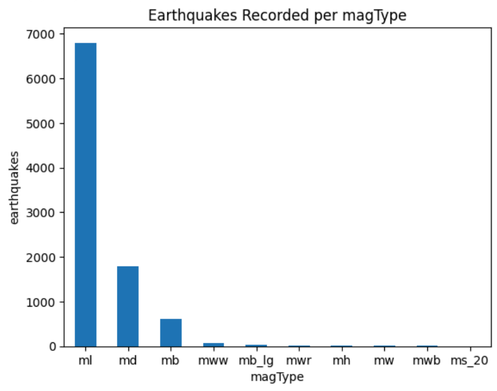

```
indonesia_quakes = quakes.query('parsed_place == "Indonesia"').assign(
 time=lambda x: pd.to_datetime(x.time, unit='ms'),
 earthquake=1
).set_index('time').resample('1D').sum()
indonesia_quakes.index = indonesia_quakes.index.strftime('%b\n%d')
indonesia_quakes.plot(
 y=['earthquake', 'tsunami'], kind='bar', figsize=(15, 3), rot=0,
 label=['earthquakes', 'tsunamis'],
 title='Earthquakes and Tsunamis in Indonesia '\
 '(September 18, 2018 - October 13, 2018)'
)
# label the axes (discussed in chapter 6)
plt.xlabel('date')
plt.ylabel('count')
```

```
<ipython-input-43-3671e7677b7a>:4: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  ).set_index('time').resample('1D').sum()
Text(0, 0.5, 'count')
```



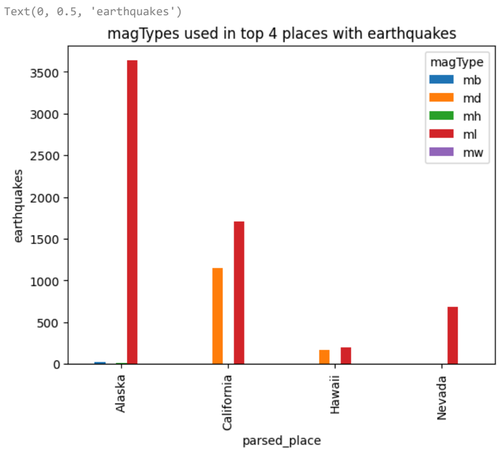

```
quakes.magType.value_counts().plot(
 kind='bar', title='Earthquakes Recorded per magType', rot=0
)
# label the axes (discussed in chapter 6)
plt.xlabel('magType')
plt.ylabel('earthquakes')
```

```
Text(0, 0.5, 'earthquakes')
```



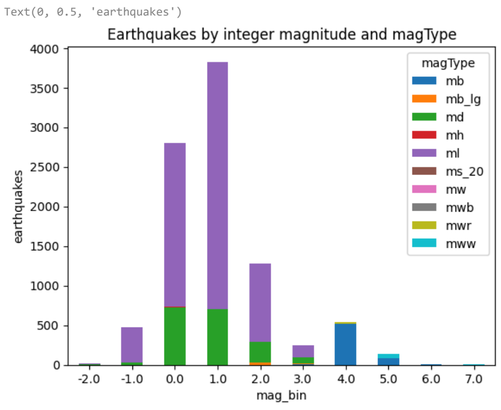

```
quakes[
 quakes.parsed_place.isin(['California', 'Alaska', 'Nevada', 'Hawaii'])
].groupby(['parsed_place', 'magType']).mag.count().unstack().plot.bar(
 title='magTypes used in top 4 places with earthquakes'
)
plt.ylabel('earthquakes') # label the axes (discussed in chapter 6)
```

```
Text(0, 0.5, 'earthquakes')
```



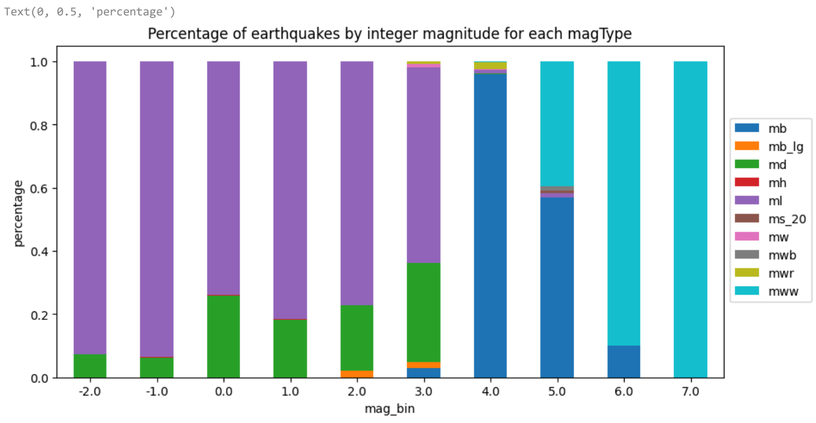

```
pivot = quakes.assign(
 mag_bin=lambda x: np.floor(x.mag)
).pivot_table(
 index='mag_bin', columns='magType', values='mag', aggfunc='count'
)
pivot.plot.bar(
 stacked=True, rot=0,
 title='Earthquakes by integer magnitude and magType'
)
plt.ylabel('earthquakes') # label the axes (discussed in chapter 6)
```

```
Text(0, 0.5, 'earthquakes')
```




```
normalized_pivot = pivot.fillna(0).apply(lambda x: x/x.sum(), axis=1)
ax = normalized_pivot.plot.bar(
 stacked=True, rot=0, figsize=(10, 5),
 title='Percentage of earthquakes by integer magnitude for each magType'
)
ax.legend(bbox_to_anchor=(1, 0.8)) # move legend to the right of the plot
plt.ylabel('percentage') # label the axes (discussed in chapter 6)
```

```
Text(0, 0.5, 'percentage')
```

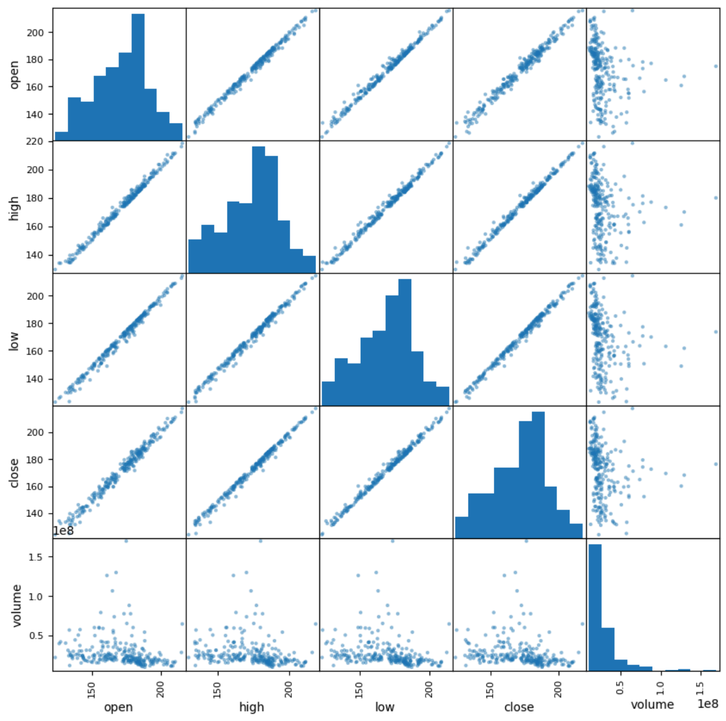


9.3 Pandas Plotting Subpackage

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
fb = pd.read_csv(
  '/content/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
```

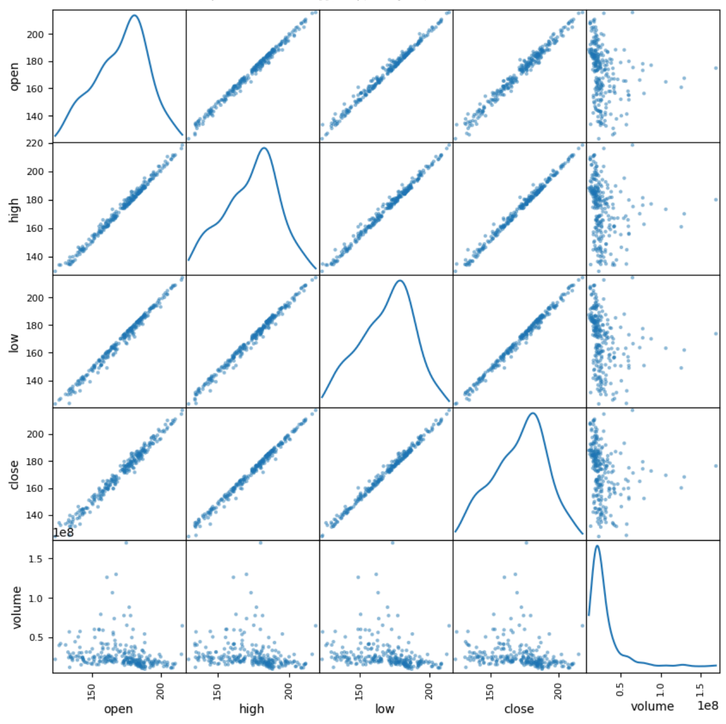
```
from pandas.plotting import scatter_matrix
scatter_matrix(fb, figsize=(10, 10))
```

```
array([[<Axes: xlabel='open', ylabel='open'>,
        <Axes: xlabel='high', ylabel='open'>,
        <Axes: xlabel='low', ylabel='open'>,
        <Axes: xlabel='close', ylabel='open'>,
        <Axes: xlabel='volume', ylabel='open'>],
       [<Axes: xlabel='open', ylabel='high'>,
        <Axes: xlabel='high', ylabel='high'>,
        <Axes: xlabel='low', ylabel='high'>,
        <Axes: xlabel='close', ylabel='high'>,
        <Axes: xlabel='volume', ylabel='high'>],
       [<Axes: xlabel='open', ylabel='low'>,
        <Axes: xlabel='high', ylabel='low'>,
        <Axes: xlabel='low', ylabel='low'>,
        <Axes: xlabel='close', ylabel='low'>,
        <Axes: xlabel='volume', ylabel='low'>],
       [<Axes: xlabel='open', ylabel='close'>,
        <Axes: xlabel='high', ylabel='close'>,
        <Axes: xlabel='low', ylabel='close'>,
        <Axes: xlabel='close', ylabel='close'>,
        <Axes: xlabel='volume', ylabel='close'>],
       [<Axes: xlabel='open', ylabel='volume'>,
        <Axes: xlabel='high', ylabel='volume'>,
        <Axes: xlabel='low', ylabel='volume'>,
        <Axes: xlabel='close', ylabel='volume'>,
        <Axes: xlabel='volume', ylabel='volume'>]], dtype=object)
```



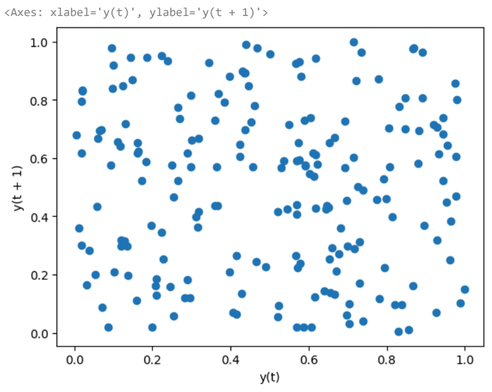

```
scatter_matrix(fb, figsize=(10, 10), diagonal='kde')
```

```
array([[<Axes: xlabel='open', ylabel='open'>,
        <Axes: xlabel='high', ylabel='open'>,
        <Axes: xlabel='low', ylabel='open'>,
        <Axes: xlabel='close', ylabel='open'>,
        <Axes: xlabel='volume', ylabel='open'>],
       [<Axes: xlabel='open', ylabel='high'>,
        <Axes: xlabel='high', ylabel='high'>,
        <Axes: xlabel='low', ylabel='high'>,
        <Axes: xlabel='close', ylabel='high'>,
        <Axes: xlabel='volume', ylabel='high'>],
       [<Axes: xlabel='open', ylabel='low'>,
        <Axes: xlabel='high', ylabel='low'>,
        <Axes: xlabel='low', ylabel='low'>,
        <Axes: xlabel='close', ylabel='low'>,
        <Axes: xlabel='volume', ylabel='low'>],
       [<Axes: xlabel='open', ylabel='close'>,
        <Axes: xlabel='high', ylabel='close'>,
        <Axes: xlabel='low', ylabel='close'>,
        <Axes: xlabel='close', ylabel='close'>,
        <Axes: xlabel='volume', ylabel='close'>],
       [<Axes: xlabel='open', ylabel='volume'>,
        <Axes: xlabel='high', ylabel='volume'>,
        <Axes: xlabel='low', ylabel='volume'>,
        <Axes: xlabel='close', ylabel='volume'>,
        <Axes: xlabel='volume', ylabel='volume'>]], dtype=object)
```



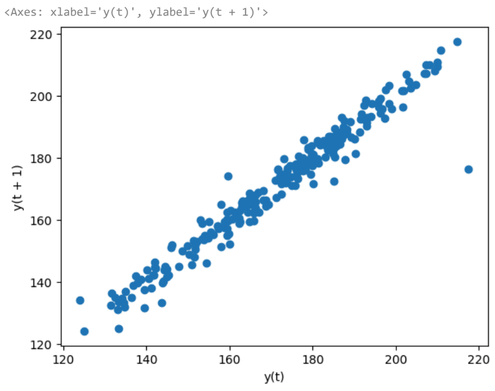

```
from pandas.plotting import lag_plot
np.random.seed(0) # make this repeatable
lag_plot(pd.Series(np.random.random(size=200)))
```

```
<Axes: xlabel='y(t)', ylabel='y(t + 1)'>
```



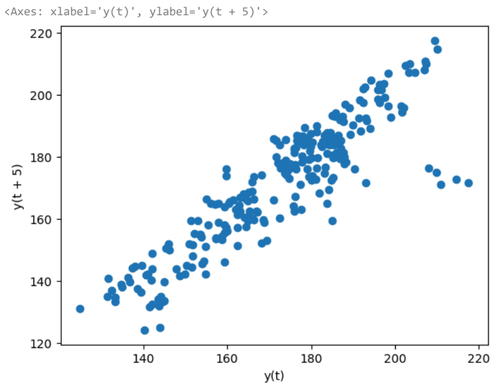

```
lag_plot(fb.close)
```

```
<Axes: xlabel='y(t)', ylabel='y(t + 1)'>
```



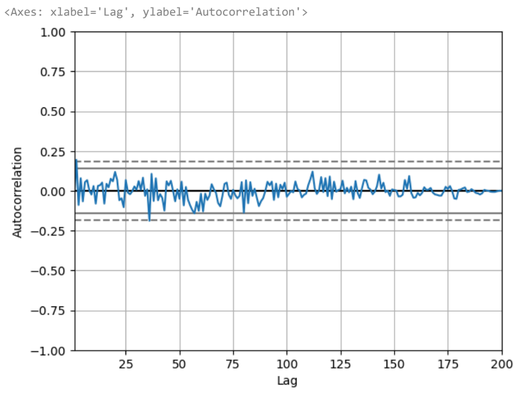

```
lag_plot(fb.close, lag=5)
```

```
<Axes: xlabel='y(t)', ylabel='y(t + 5)'>
```



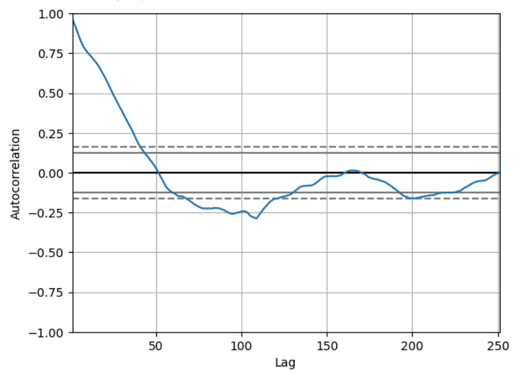

```
from pandas.plotting import autocorrelation_plot
np.random.seed(0) # make this repeatable
autocorrelation_plot(pd.Series(np.random.random(size=200)))
```

```
<Axes: xlabel='Lag', ylabel='Autocorrelation'>
```

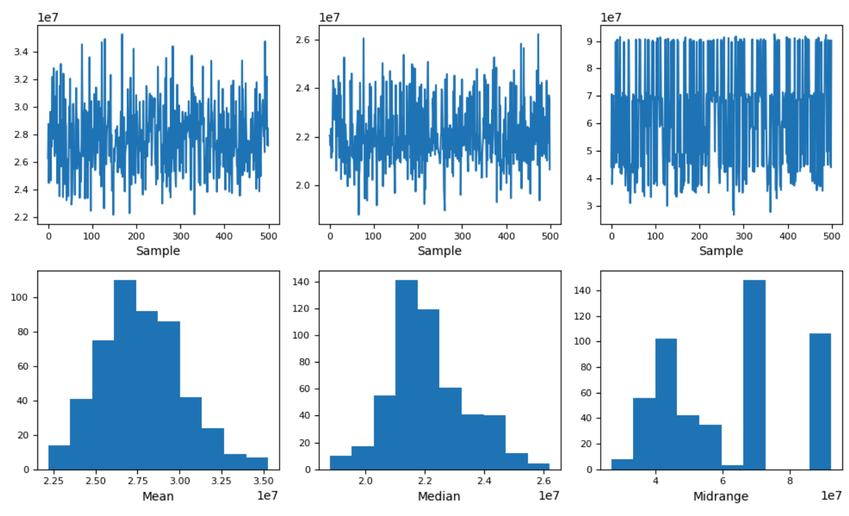



```
autocorrelation_plot(fb.close)
```

```
<Axes: xlabel='Lag', ylabel='Autocorrelation'>
```




```
from pandas.plotting import bootstrap_plot
fig = bootstrap_plot(fb.volume, fig=plt.figure(figsize=(10, 6)))
```



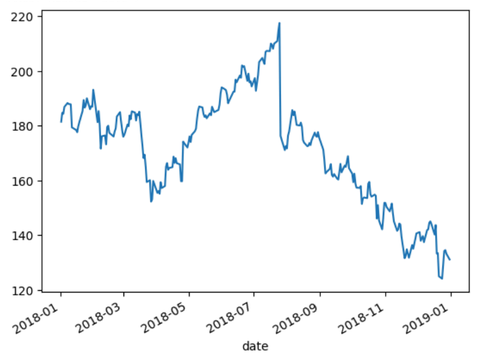
supplementary activity

Using the CSV files provided and what we have learned so far in this module complete the following exercises:

1. Plot the rolling 20-day minimum of the Facebook closing price with the pandas plot() method.
2. Create a histogram and KDE of the change from open to close in the price of Facebook stock.
3. Using the earthquake data, create box plots for the magnitudes of each magType used in Indonesia.
4. Make a line plot of the difference between the weekly maximum high price and the weekly minimum low price for Facebook. This should be a single line.
5. Using matplotlib and pandas, create two subplots side-by-side showing the effect that after-hours trading has had on Facebook's stock price:
    - The first subplot will contain a line plot of the daily difference between that day's opening price and the prior day's closing price (be sure to review the Time series section of Aggregating Pandas DataFrames for an easy way to do this).
    - The second subplot will be a bar plot showing the net effect this had monthly, using resample().
    - Bonus #1: Color the bars according to whether they are gains in the stock price (green) or drops in the stock price (red).
    - Bonus #2: Modify the x-axis of the bar plot to show the threeletter abbreviation for the month.

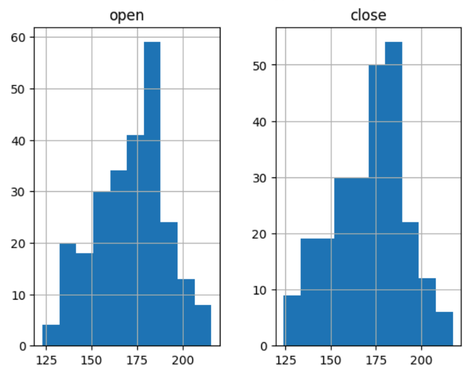
```
import matplotlib.pyplot as plt
import pandas as pd

fb = pd.read_csv('/content/fb_stock_prices_2018.csv', index_col='date', parse_dates=True)
quakes = pd.read_csv('/content/earthquakes-1.csv')
fb['close'].plot(label='FB closing price')
```

```
<Axes: xlabel='date'>
```



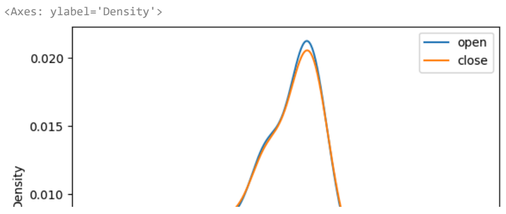

```
fb.hist(['open','close'])
```

```
array([[<Axes: title={'center': 'open'}>,
        <Axes: title={'center': 'close'}>]], dtype=object)
```




```
fb.plot(kind='kde', y = ['open','close'])
```

```
<Axes: ylabel='Density'>
```



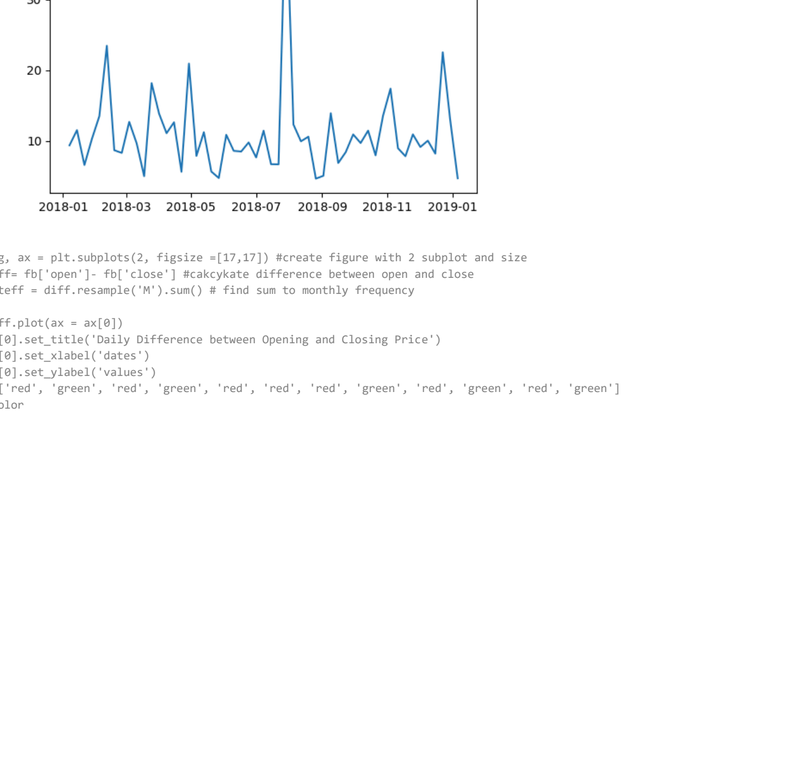

```
quakes.loc[quakes['parsed_place']=='Indonesia']
```

	mag	magType	time	place	tsunami	parsed_place
9	4.7	mb	1539472814760	219km SSE of Saparua, Indonesia	0	Indonesia
13	4.5	mb	1539470898340	120km SSW of Banda Aceh, Indonesia	0	Indonesia
180	5.2	mww	1539405255580	25km E of Bitung, Indonesia	0	Indonesia
421	4.7	mb	1539331098920	38km SSW of Nggongi Satu, Indonesia	0	Indonesia
660	4.4	mb	1539258833830	51km WSW of Kasiguncu, Indonesia	0	Indonesia
...	...	...	...	...	...	...
9041	4.3	mb	1537296305750	7km WSW of Karangsubagan, Indonesia	0	Indonesia
9075	4.4	mb	1537288723310	103km W of Kuripan, Indonesia	0	Indonesia
9108	4.0	mb	1537280181100	123km NE of Bitung, Indonesia	0	Indonesia
9209	4.7	mb	1537256021950	18km NE of Reuleuet, Indonesia	0	Indonesia
9212	4.7	mb	1537255636260	2km ESE of Lokokrangan, Indonesia	0	Indonesia

147 rows × 6 columns

```
indoq = quakes.query("parsed_place == 'Indonesia'")
indoq[['mag', 'magType']].groupby('magType').boxplot(
 figsize=(11, 8), subplots=False
)
plt.title('Earthquake in Indonesia Magnitude Boxplots by magType')
plt.ylabel('magnitude')
```

```
Text(0, 0.5, 'magnitude')
```




```
dwh = fb.resample('W').agg({'high': 'max', 'low': 'min'})
dwh['high_low_difference'] = dwh['high'] - dwh['low']
plt.plot(dwh.high_low_difference)
```

```
[<matplotlib.lines.Line2D at 0x7e9e880fb9a0>]
```



```
fig, ax = plt.subplots(2, figsize =[17,17]) #create figure with 2 subplot and size
diff= fb['open']- fb['close'] #cakcykate difference between open and close
neteff = diff.resample('M').sum() # find sum to monthly frequency

diff.plot(ax = ax[0])
ax[0].set_title('Daily Difference between Opening and Closing Price')
ax[0].set_xlabel('dates')
ax[0].set_ylabel('values')
c=['red', 'green', 'red', 'green', 'red', 'red', 'red', 'green', 'red', 'green', 'red', 'green']
#color
```