9.2 Plotting with Pandas

The plot() method is available on Series and DataFrame objects. Many of the parameters get passed down to matplotlib. The kind argument let's us vary the plot type.

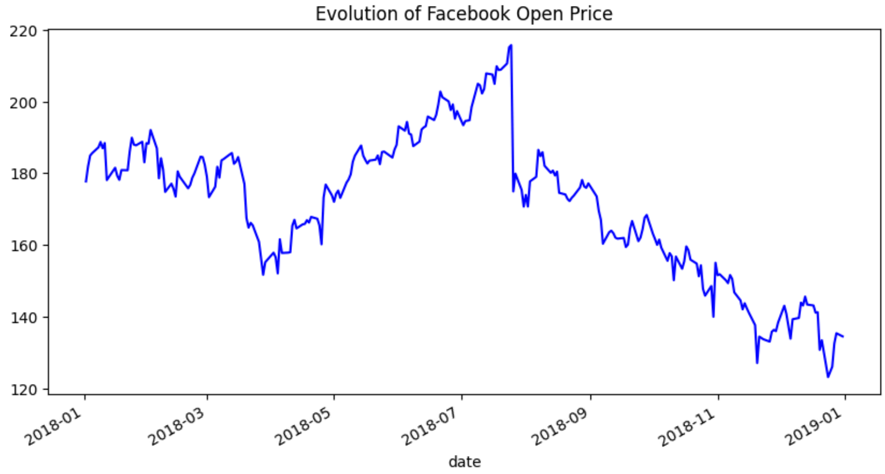
About the Data In this notebook, we will be working with 2 datasets: *#Facebook's stock price throughout 2018 (obtained using the stock_analysis package) *Earthquake data from September 18, 2018 - October 13, 2018 (obtained from the US Geological Survey (USGS) using the USGS API)

Setup

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

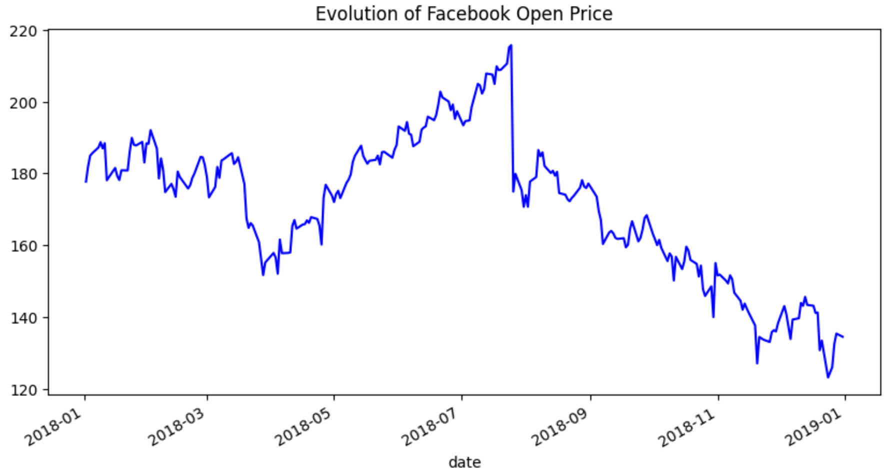
```
fb.plot(
 kind='line',
 y='open',
 figsize=(10, 5),
 style='b-',
 legend=False,
 title='Evolution of Facebook Open Price'
)
```

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



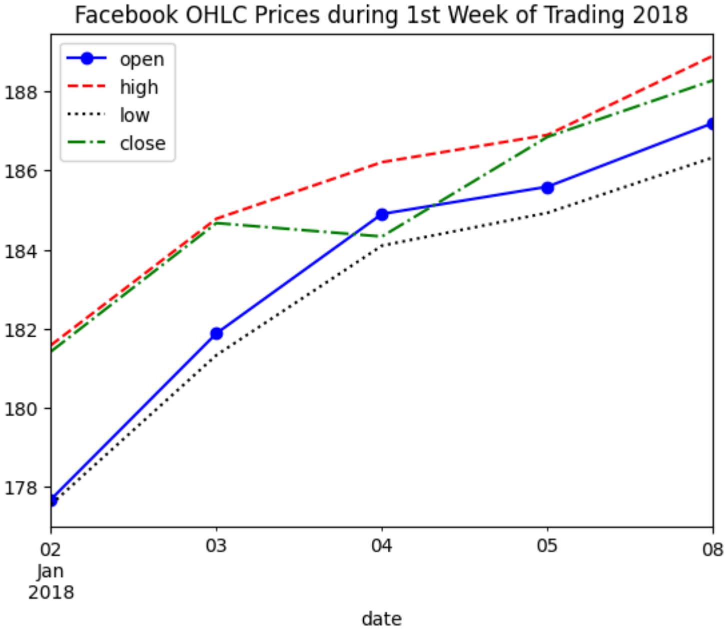
```
fb.plot(
 kind='line',
 y='open',
 figsize=(10, 5),
 color='blue',
 linestyle='solid',
 legend=False,
 title='Evolution of Facebook Open Price'
)
```

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



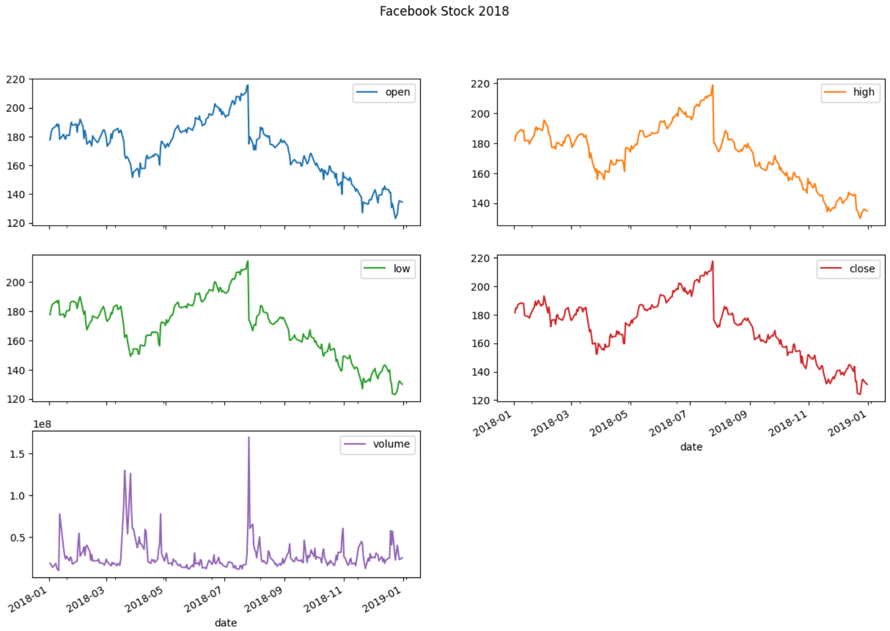
```
fb.iloc[:5,].plot(
 y=['open', 'high', 'low', 'close'],
 style=['b-o', 'r--', 'k:', 'g-.'],
 title='Facebook OHLC Prices during 1st Week of Trading 2018'
)
```

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



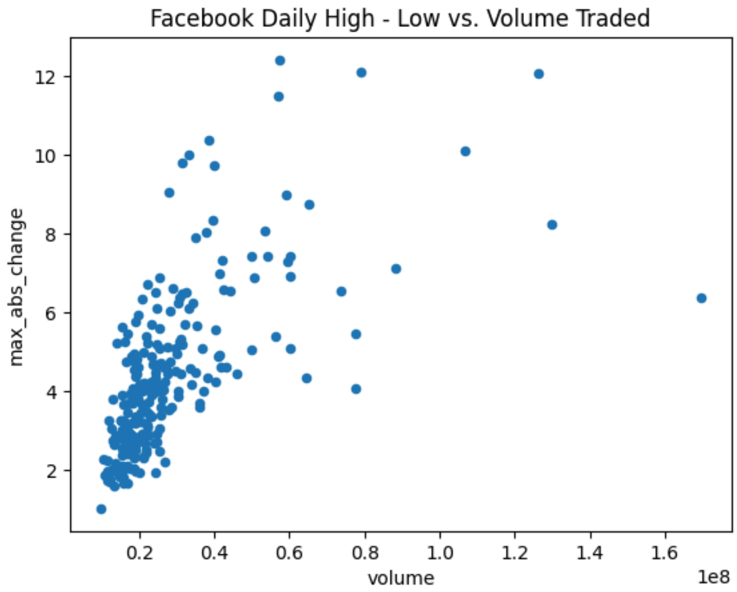
```
fb.plot(
 kind='line',
 subplots=True,
 layout=(3,2),
 figsize=(15,10),
 title='Facebook Stock 2018'
)
```

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



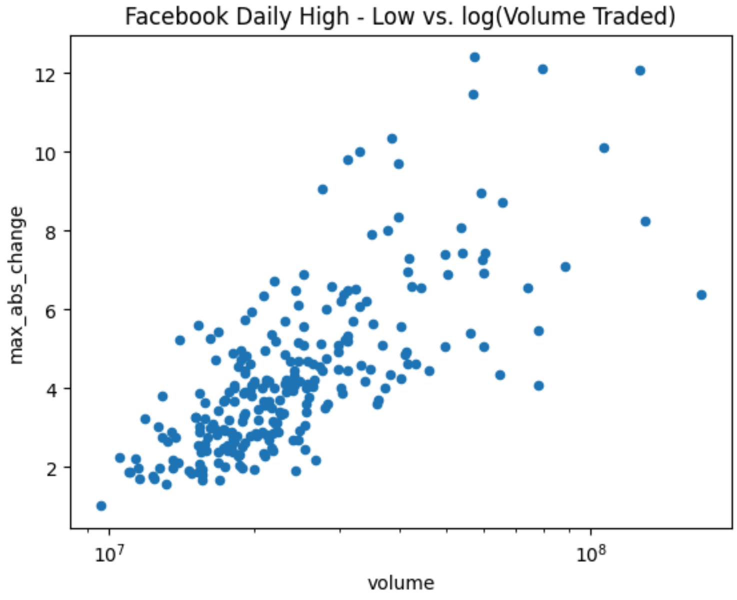
```
fb.assign(
 max_abs_change=fb.high - fb.low
).plot(
 kind='scatter', x='volume', y='max_abs_change',
 title='Facebook Daily High - Low vs. Volume Traded'
)
```

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



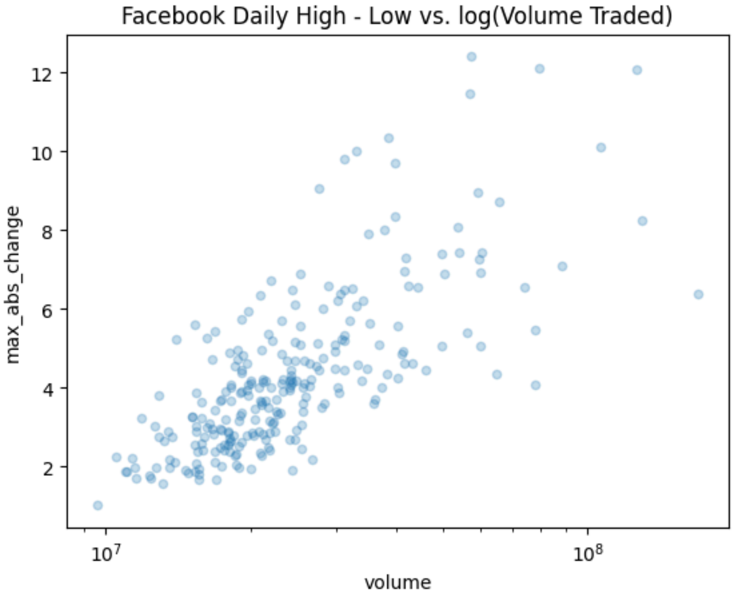
```
fb.assign(
 max_abs_change=fb.high - fb.low
).plot(
 kind='scatter', x='volume', y='max_abs_change',
 title='Facebook Daily High - Low vs. log(Volume Traded)',
 logx=True
)
```

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



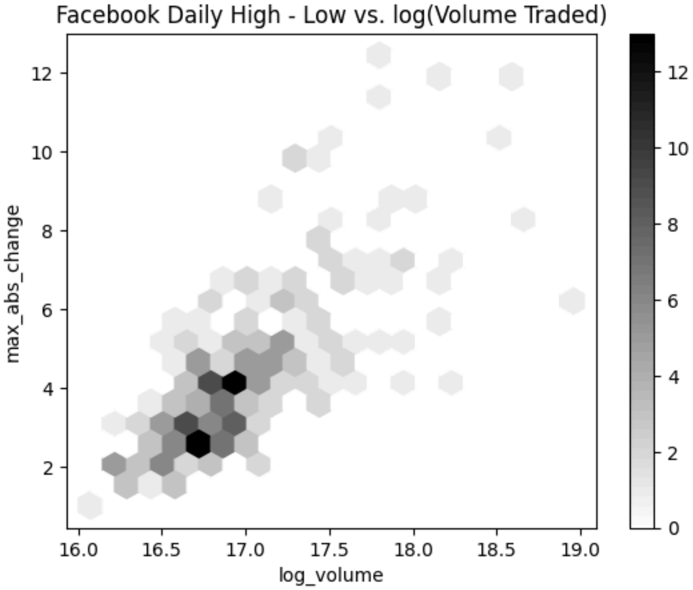
```
fb.assign(
 max_abs_change=fb.high - fb.low
).plot(
 kind='scatter', x='volume', y='max_abs_change',
 title='Facebook Daily High - Low vs. log(Volume Traded)',
 logx=True, alpha=0.25
)
```

```
<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
)
```

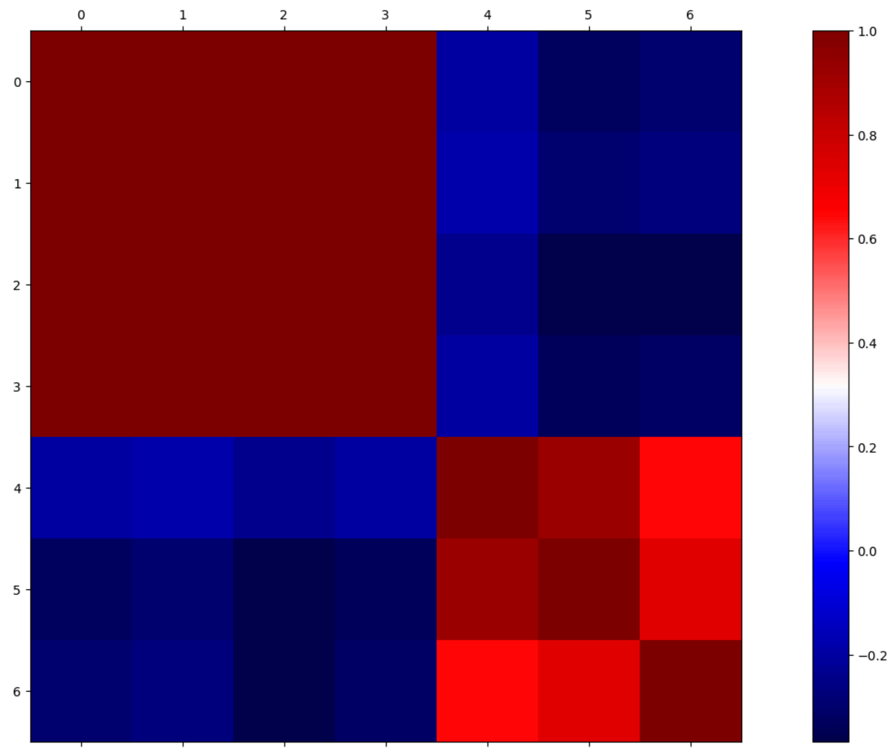
```
<Axes: title={'center': 'Facebook Daily High - Low vs. log(Volume Traded)'},
xlabel='log_volume', ylabel='max_abs_change'>
```



```
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)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-14-c8dce279c6e7> in <cell line: 9>()
      7 
      8 im = ax.matshow(fb_corr, cmap='seismic')
----> 9 fig.colorbar(im).set_clim(-1, 1)
     10 
     11 labels = [col.lower() for col in fb_corr.columns]

AttributeError: 'Colorbar' object has no attribute 'set_clim'
```

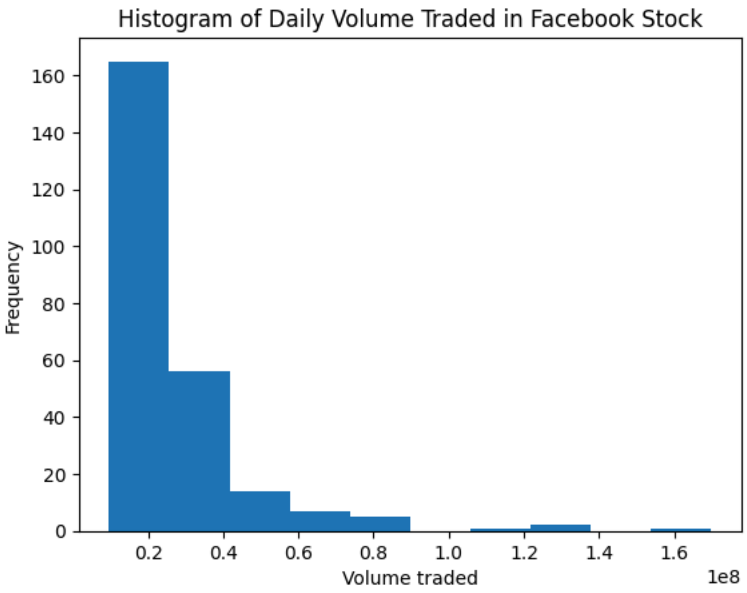


```
fb_corr.loc['max_abs_change', ['volume', 'log_volume']]
```

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

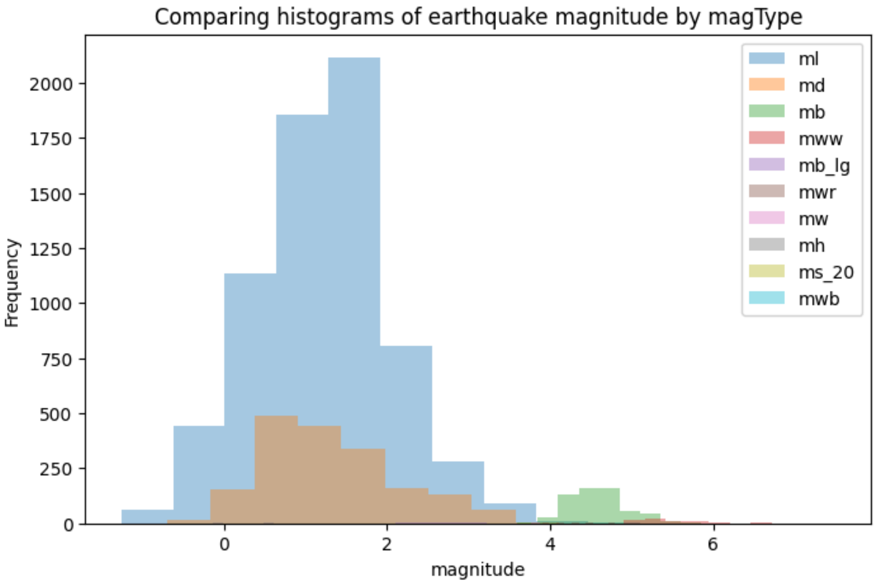
```
fb.volume.plot(
 kind='hist',
 title='Histogram of Daily Volume Traded in Facebook Stock'
)
plt.xlabel('Volume traded') # label the x-axis (discussed in chapter 6)
```

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



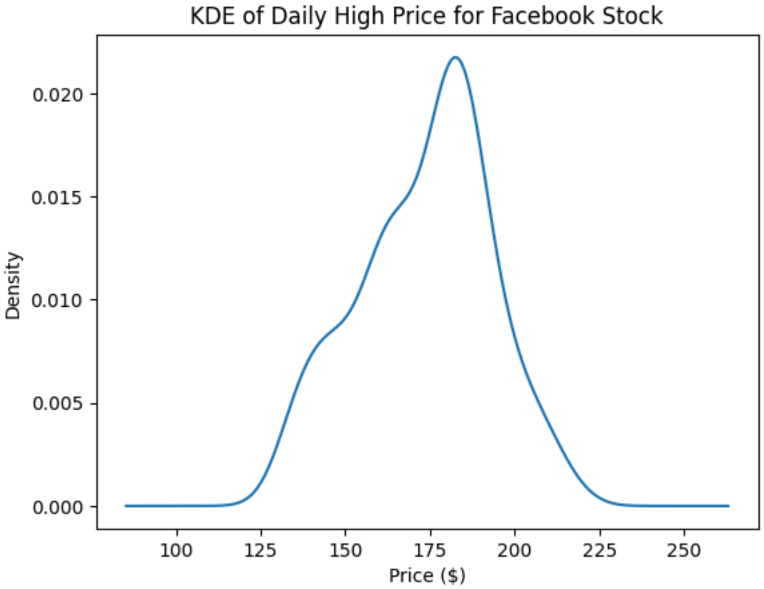
```
fig, axes = plt.subplots(figsize=(8, 5))
for magtype in quakes.magType.unique():
  data = quakes.query(f'magType == "{magtype}"').mag
  if not data.empty:
    data.plot(
      kind='hist', ax=axes, alpha=0.4,
      label=magtype, legend=True,
      title='Comparing histograms of earthquake magnitude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

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



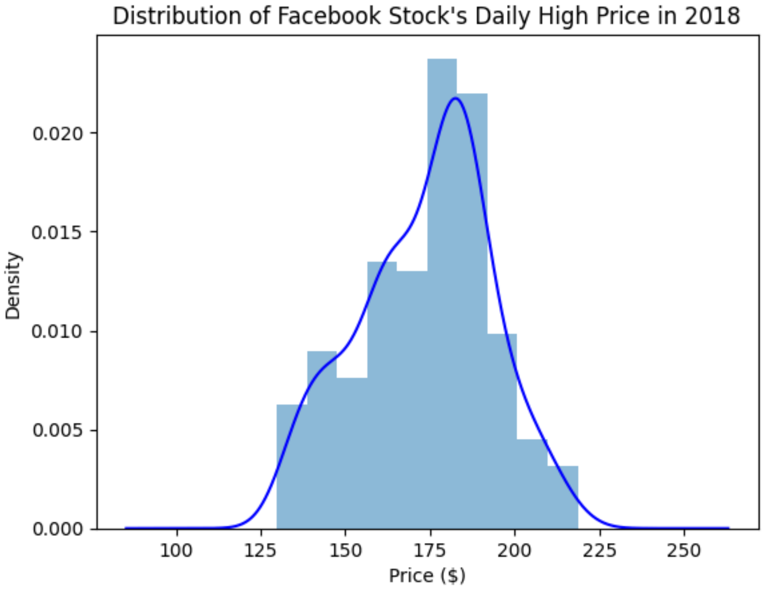
```
fb.high.plot(
 kind='kde',
 title='KDE of Daily High Price for Facebook Stock'
)
plt.xlabel('Price ($)') # label the x-axis (discussed in chapter 6)
```

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



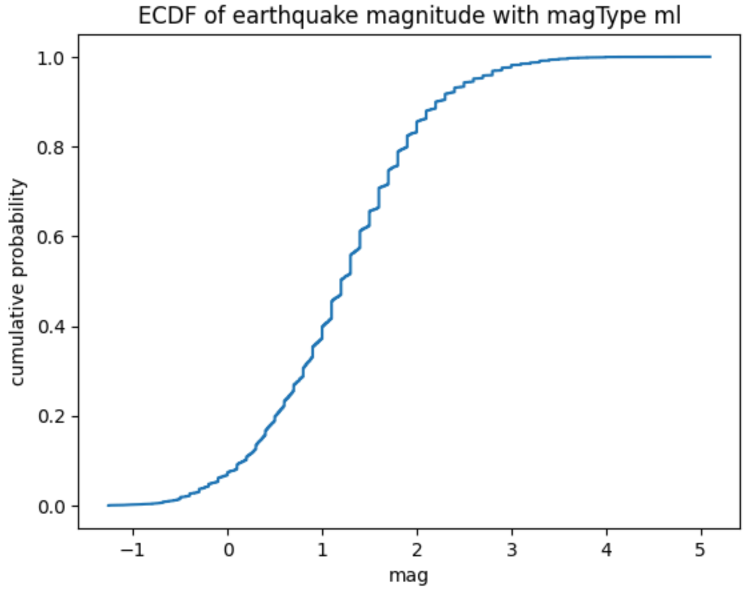
```
ax = fb.high.plot(kind='hist', density=True, alpha=0.5)
fb.high.plot(
 ax=ax, kind='kde', color='blue',
 title='Distribution of Facebook Stock\'s Daily High Price in 2018'
)
plt.xlabel('Price ($)') # label the x-axis (discussed in chapter 6)
```

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



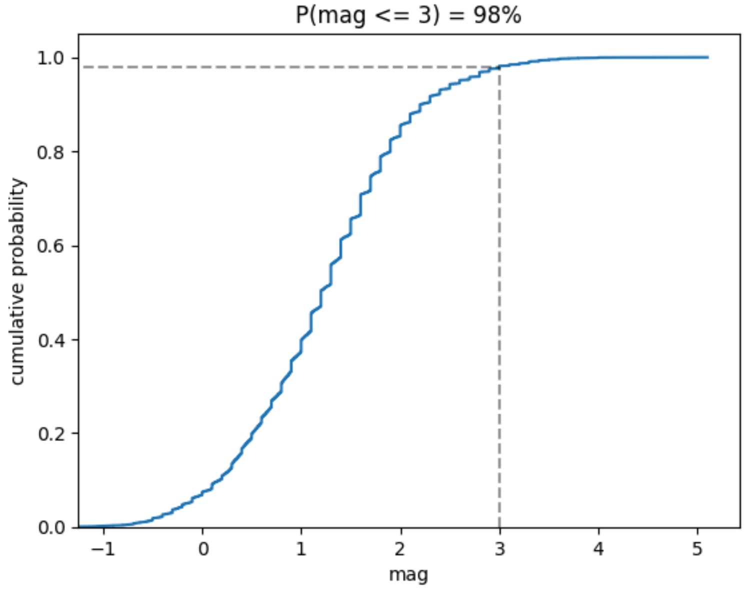
```
from statsmodels.distributions.empirical_distribution import ECDF
ecdf = ECDF(quakes.query('magType == "ml"').mag)
plt.plot(ecdf.x, ecdf.y)
# axis labels (we will cover this in chapter 6)
plt.xlabel('mag') # add x-axis label
plt.ylabel('cumulative probability') # add y-axis label
# add title (we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



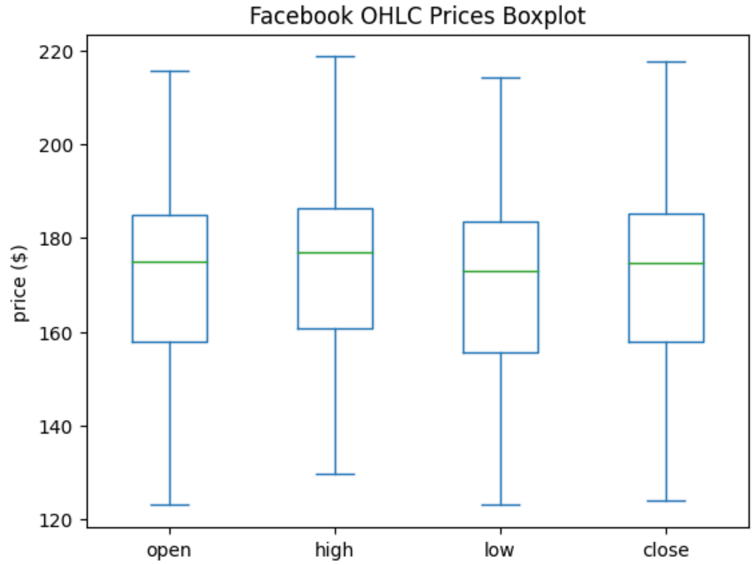
```
from statsmodels.distributions.empirical_distribution import ECDF
ecdf = ECDF(quakes.query('magType == "ml"').mag)
plt.plot(ecdf.x, ecdf.y)
# formatting below will all be covered in chapter 6
# axis labels
plt.xlabel('mag') # add x-axis label
plt.ylabel('cumulative probability') # add y-axis label
# 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%')
```



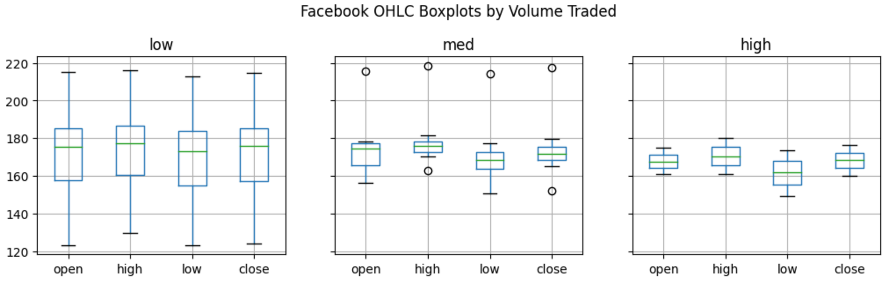
```
fb.iloc[:,:4].plot(kind='box', title='Facebook OHLC Prices Boxplot')
plt.ylabel('price ($)') # label the x-axis (discussed in chapter 6)
```

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



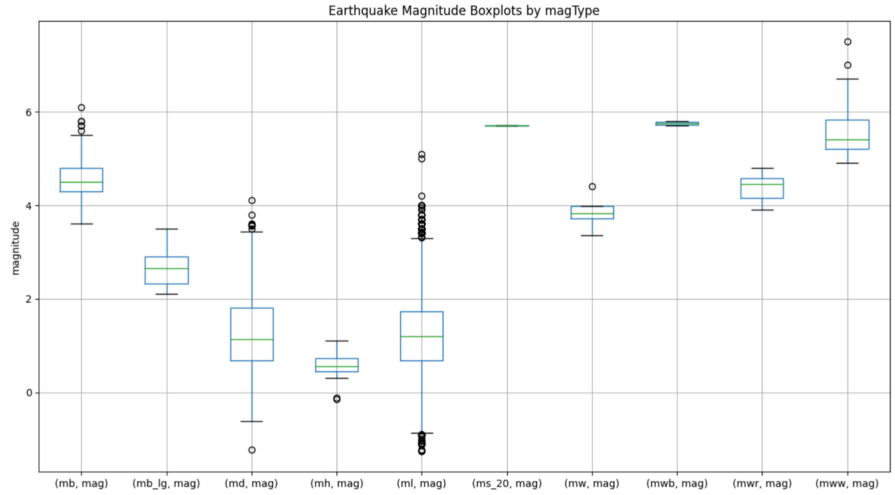
```
fb.assign(
 volume_bin=pd.cut(fb.volume, 3, labels=['low', 'med', 'high'])
).groupby('volume_bin').boxplot(
 column=['open', 'high', 'low', 'close'],
 layout=(1, 3), figsize=(12, 3)
)
plt.suptitle('Facebook OHLC Boxplots by Volume Traded', y=1.1)
```

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



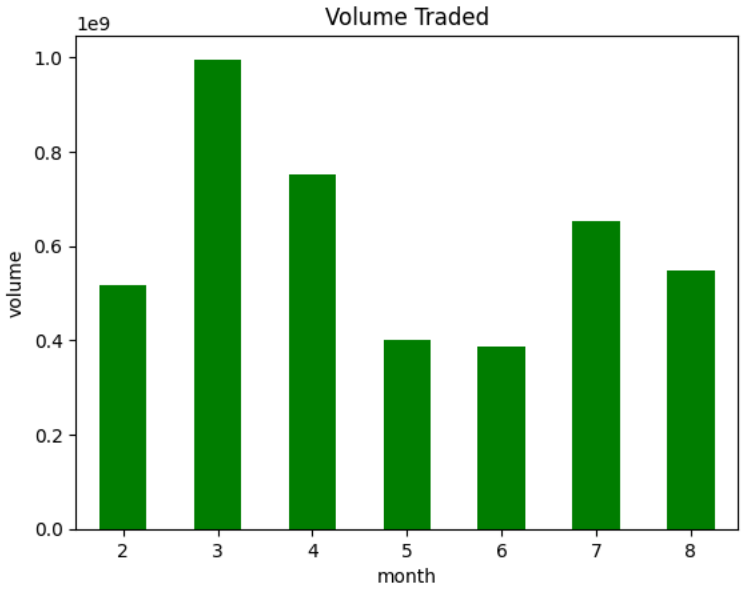
```
quakes[['mag', 'magType']].groupby('magType').boxplot(
 figsize=(15, 8), subplots=False
)
plt.title('Earthquake Magnitude Boxplots by magType')
plt.ylabel('magnitude') # label the y-axis (discussed in chapter 6)
```

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



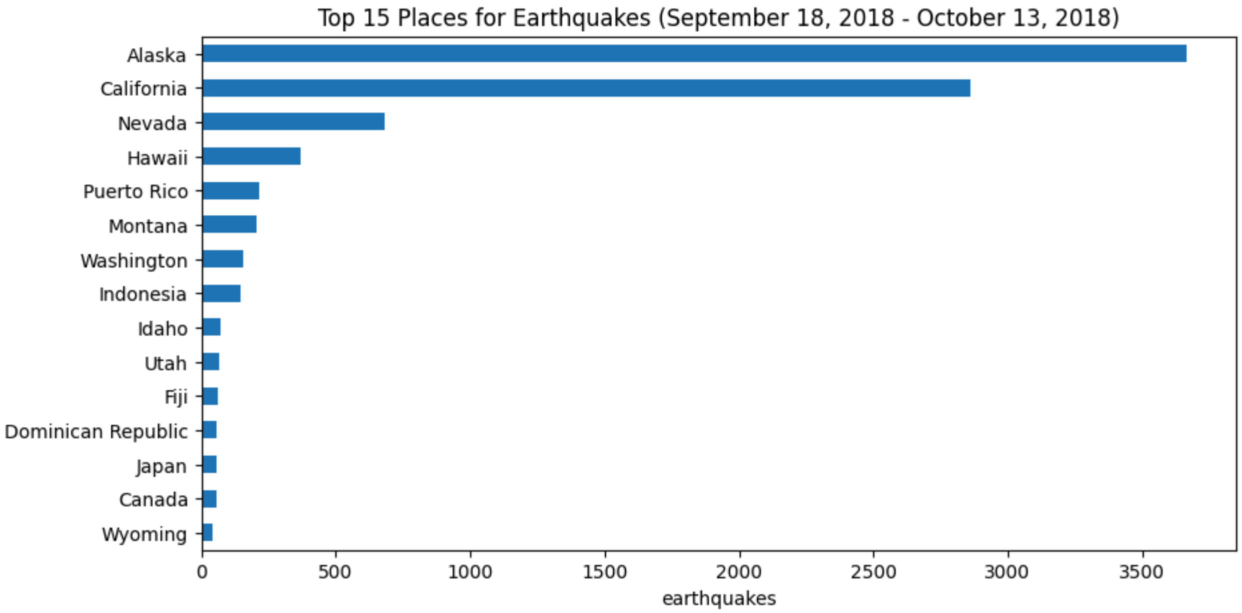
```
fb['2018-02':'2018-08'].assign(
 month=lambda x: x.index.month
).groupby('month').sum().volume.plot.bar(
 color='green', rot=0, title='Volume Traded'
 )
plt.ylabel('volume') # label the y-axis (discussed in chapter 6)
```

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



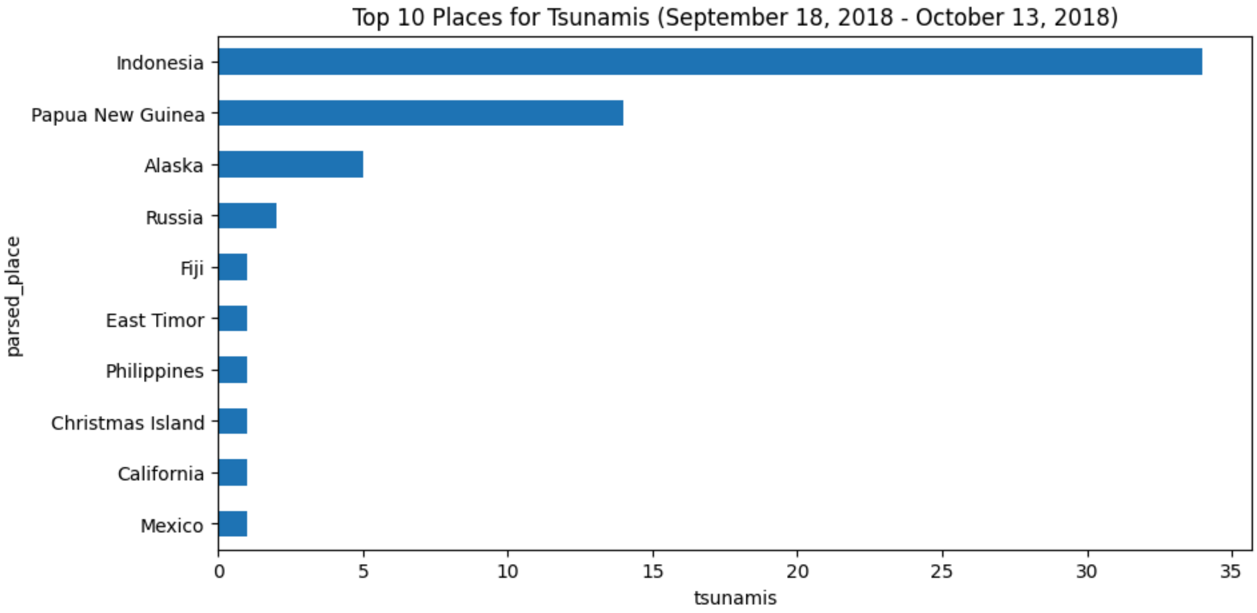
```
quakes.parsed_place.value_counts().iloc[14::-1,].plot(
 kind='barh', figsize=(10, 5),
 title='Top 15 Places for Earthquakes '\
 '(September 18, 2018 - October 13, 2018)'
)
plt.xlabel('earthquakes') # label the x-axis (discussed in chapter 6)
```

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



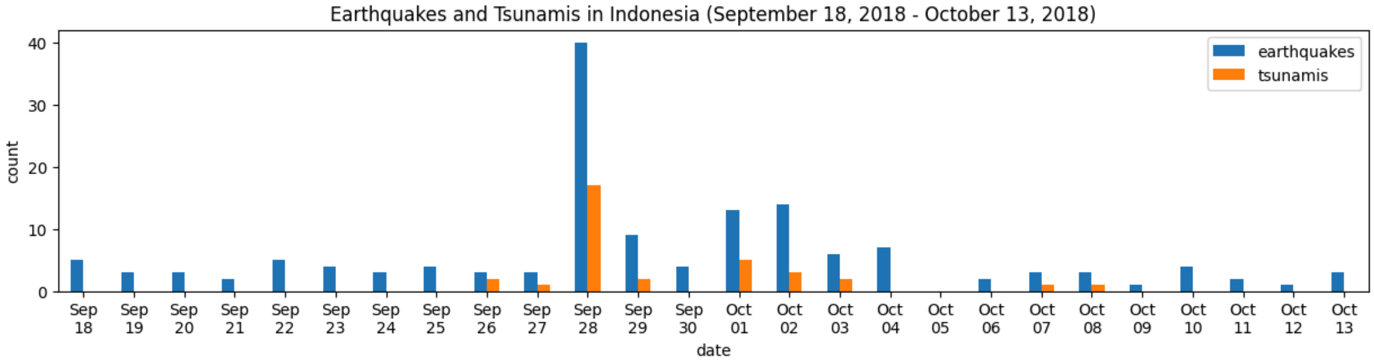
```
quakes.groupby('parsed_place').tsunami.sum().sort_values().iloc[-10::,].plot(
 kind='barh', figsize=(10, 5),
 title='Top 10 Places for Tsunamis '\
 '(September 18, 2018 - October 13, 2018)'
)
plt.xlabel('tsunamis') # label the x-axis (discussed in chapter 6)
```

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



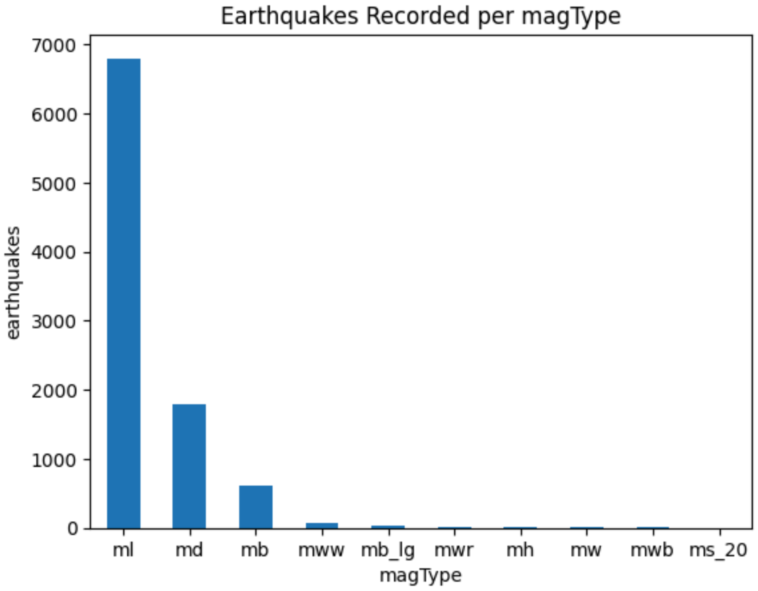
```
indonesia_quakes = quakes.query('parsed_place == "Indonesia"').assign(
 time=lambda x: pd.to_datetime(x.time, unit='ms'),
 earthquake=1
).set_index('time').resample('1D').sum()
indonesia_quakes.index = indonesia_quakes.index.strftime('%b\n%d')
indonesia_quakes.plot(
 y=['earthquake', 'tsunami'], 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-30-3671e7677b7a>:4: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
  ).set_index('time').resample('1D').sum()
Text(0, 0.5, 'count')
```



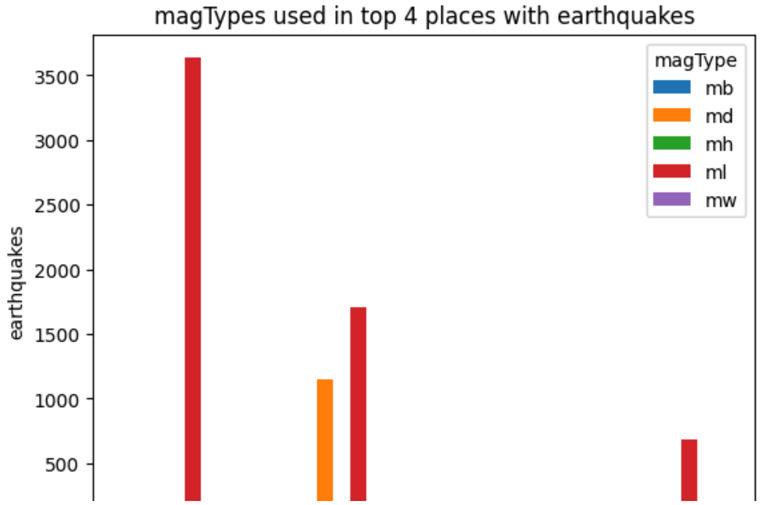
```
quakes.magType.value_counts().plot(
 kind='bar', title='Earthquakes Recorded per magType', rot=0
)
# label the axes (discussed in chapter 6)
plt.xlabel('magType')
plt.ylabel('earthquakes')
```

```
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')
```


magTypes used in top 4 places with earthquakes
earthquakes
magType
mb
md
mh
ml
mw