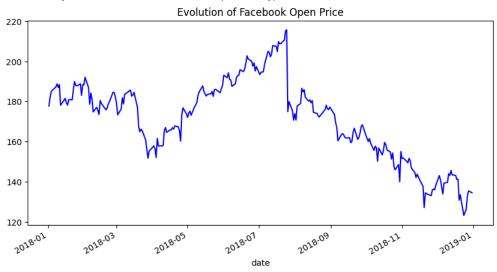
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

fb = pd.read_csv(
   'data/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
quakes = pd.read_csv('data/earthquakes.csv')

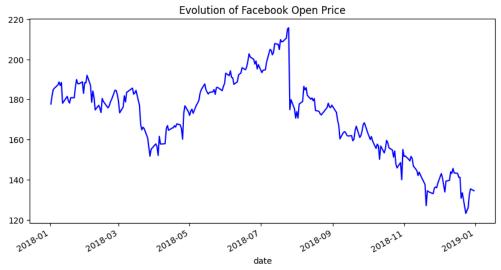
fb.plot( # using pandas to plot
   kind='line', # line graph
   y='open', # sets open at y axis
   # by default, x axis is the index which is date
   figsize=(10, 5), # 10x5 inches
   style='b-', # styles available are at matplotlib.style.available
   legend=False, # no legend, default is True
   title='Evolution 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', # uses color param instead of style
  linestyle='solid', # linestyle instead of style
  legend=False,
  title='Evolution of Facebook Open Price'
) # results in a similar graph as above
```

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



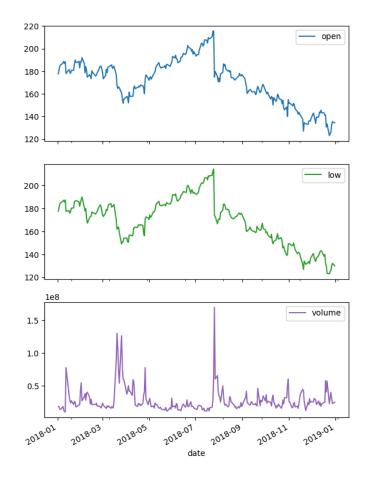
```
fb.iloc[:5,].plot(
  y=['open', 'high', 'low', 'close'], # many lines are possible
  style=['b-o', 'r--', 'k:', 'g-.'], # different styles per line
  title='Facebook OHLC Prices during 1st Week of Trading 2018'
)
```

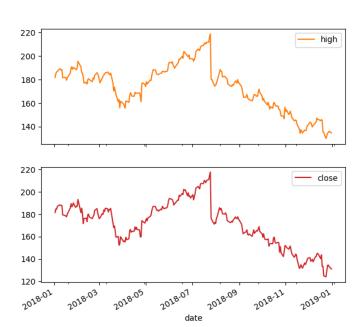
Facebook OHLC Prices during 1st Week of Trading 2018 -- open -- high 188 ····· low — · · · close 186 184 182 180 178 03 04 05 02 Jan 2018 08 date

```
fb.plot(
   kind='line',
   subplots=True, # allow subplots
   layout=(3,2), # specifies the layout of the subplots
   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)
```

Facebook Stock 2018



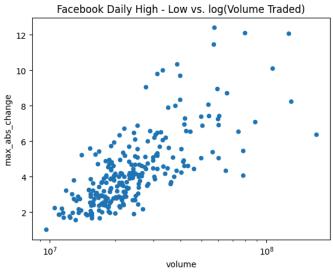


```
fb.assign(
  max_abs_change=fb.high - fb.low # adds absolute change column
).plot(
  kind='scatter', x='volume', y='max_abs_change', # scatter plot
  title='Facebook Daily High - Low vs. Volume Traded'
```



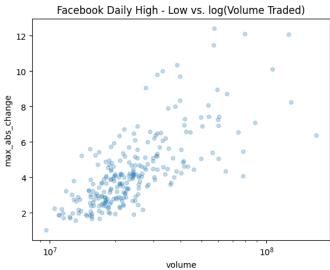
```
fb.assign(
  max_abs_change=fb.high - fb.low
).plot(
  kind='scatter', x='volume', y='max_abs_change',
  title='Facebook Daily High - Low vs. log(Volume Traded)',
  logx=True # sets the x axis to logarithmic
)
```

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

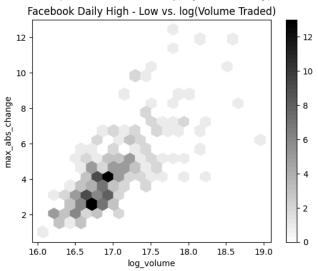


```
fb.assign(
  max_abs_change=fb.high - fb.low
).plot(
  kind='scatter', x='volume', y='max_abs_change',
  title='Facebook Daily High - Low vs. log(Volume Traded)',
  logx=True, alpha=0.25 # alpha is the transparency of the dots
  # a number from 0 - 1 where 1 is solid
)
```

<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), # adds logarithmic volume
  max_abs_change=fb.high - fb.low
).plot(
  kind='hexbin', # hexbin graph
  x='log_volume',
  y='max_abs_change',
  title='Facebook Daily High - Low vs. log(Volume Traded)',
  colormap='gray_r', # color of the hexes are gray
  gridsize=20,
  sharex=False # needed to be set to false
)
```



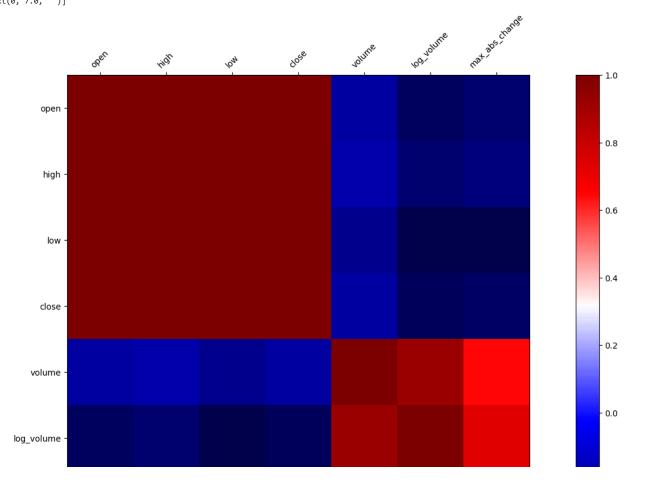
```
fig, ax = plt.subplots(figsize=(20, 10))
fb_corr = fb.assign(
log_volume=np.log(fb.volume),
    max_abs_change=fb.high - fb.low
).corr() # gets the correlation between every column

im = ax.matshow(fb_corr, cmap='seismic')
fig.colorbar(im) # creates the heatmap
labels = [col.lower() for col in fb_corr.columns] # gets the column names
ax.set_xticklabels([''] + labels, rotation=45) # sets the column names for x, rotated 45 degrees
ax.set_yticklabels([''] + labels) # sets the column names for y

# removes the .set_clim() method as it results in an error

<ipython-input-11-83890aefe672>:10: UserWarning: FixedFormatter should only be used together with FixedLocator
    ax.set_xticklabels([''] + labels, rotation=45) # sets the column names for x, rotated 45 degrees
    <ipython-input-11-83890aefe672>:11: UserWarning: FixedFormatter should only be used together with FixedLocator
    ax.set_yticklabels([''] + labels) # sets the column names for y

[Text(0, -1.0, ''),
    Text(0, -0.0, 'open'),
    Text(0, 1.0, 'high'),
    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']]
gets the r value between max_abs_change and volume and log_volume

volume 0.642027 log_volume 0.731542 Name: max_abs_change, dtype: float64

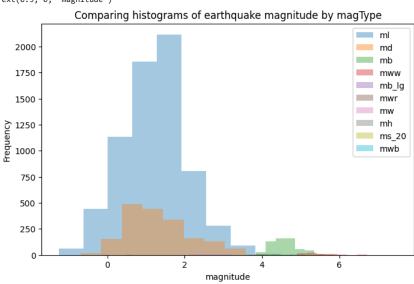
```
plt.xlabel('Volume traded')
     Text(0.5, 0, 'Volume traded')
                 Histogram of Daily Volume Traded in Facebook Stock
         160
         140
         120
      Frequency
80
          60
          40
          20
           0
                   0.2
                           0.4
                                  0.6
                                          0.8
                                                 1.0
                                                         1.2
                                                                1.4
                                                                        1.6
                                                                             1e8
                                        Volume traded
```

title='Histogram of Daily Volume Traded in Facebook Stock'

```
fig, axes = plt.subplots(figsize=(8, 5))
for magtype in quakes.magType.unique(): # iterates thru every magType
  data = quakes.query(f'magType == "{magtype}"').mag # gets the magnitudes for the certain magType
  if not data.empty:
    data.plot(
        kind='hist', ax=axes, alpha=0.4, # creates the histogram, recall alpha
        label=magtype, legend=True,
        title='Comparing histograms of earthquake magnitude by magType'
    )
plt.xlabel('magnitude')
```

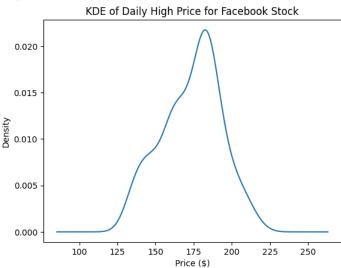
Text(0.5, 0, 'magnitude')

fb.volume.plot(
 kind='hist', # histogram



```
fb.high.plot( # uses the high column of fb datagram
    kind='kde', # kernel density estimation graph
    title='KDE of Daily High Price for Facebook Stock'
)
plt.xlabel('Price ($)')
```

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



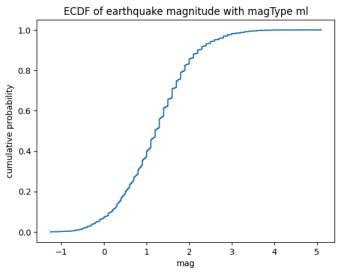
```
ax = fb.high.plot(kind='hist', density=True, alpha=0.5) # creates a histogram
fb.high.plot(
   ax=ax, kind='kde', color='blue', # creates a kde
   title='Distribution of Facebook Stock\'s Daily High Price in 2018'
)
plt.xlabel('Price ($)') # these two plots would overlap
```

0.020 - 0.015 - 0.005 - 0.000 - 100 125 150 175 200 225 250 Price (\$)

from statsmodels.distributions.empirical_distribution import ECDF # creates the empirical cumulative distribution function

```
ecdf = ECDF(quakes.query('magType == "m1"').mag) # gets magnitudes with magType ml
plt.plot(ecdf.x, ecdf.y) # plots the ecdf result
# labels and title
plt.xlabel('mag')
plt.ylabel('cumulative probability')
plt.title('ECDF of earthquake magnitude with magType m1')
```

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

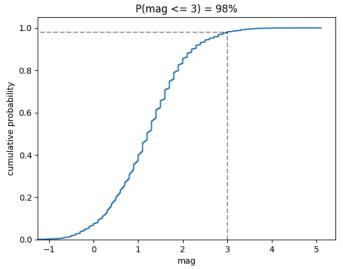


 $from \ stats models. distributions. empirical_distribution \ import \ ECDF$

```
ecdf = ECDF(quakes.query('magType == "ml"').mag)
plt.plot(ecdf.x, ecdf.y)
plt.xlabel('mag')
plt.ylabel('cumulative probability')

plt.plot( # creates a line to visualize the magnitudes less than 3
[3, 3], [0, .98], 'k--',
[-1.5, 3], [0.98, 0.98], 'k--', alpha=0.4
)
plt.ylim(0, None)
plt.xlim(-1.25, None)
plt.title('P(mag <= 3) = 98%')</pre>
```

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



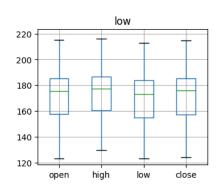
 $fb.iloc[:,:4].plot(kind='box', \ title='Facebook \ OHLC \ Prices \ Boxplot') \ \# \ creates \ a \ boxplot \ plt.ylabel('price ($)')$

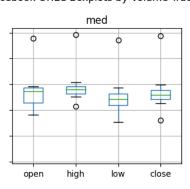
Facebook OHLC Prices Boxplot 200 (\$) 180 140 120 open high low close

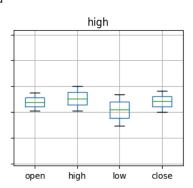
```
fb.assign(
  volume_bin=pd.cut(fb.volume, 3, labels=['low', 'med', 'high']) # creates 3 bins
).groupby('volume_bin').boxplot( # creates multiple boxplots
  column=['open', 'high', 'low', 'close'],
  layout=(1, 3), figsize=(12, 3)
)
plt.suptitle('Facebook OHLC Boxplots by Volume Traded', y=1.1)
```

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

Facebook OHLC Boxplots by Volume Traded

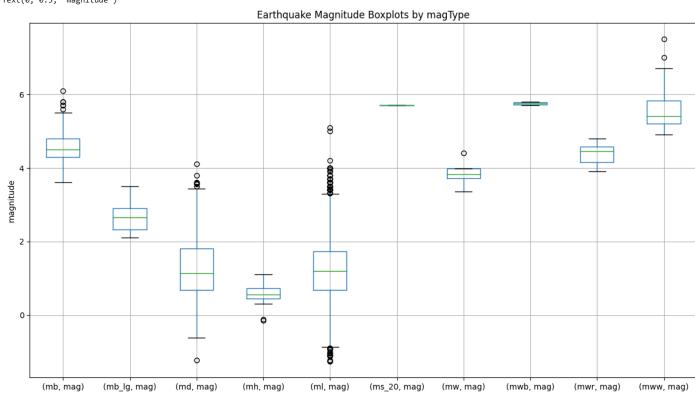






```
quakes[['mag', 'magType']].groupby('magType').boxplot( # creates a boxplot for magnitude
   figsize=(15, 8), subplots=False
)
plt.title('Earthquake Magnitude Boxplots by magType')
plt.ylabel('magnitude')
```

Text(0, 0.5, 'magnitude')



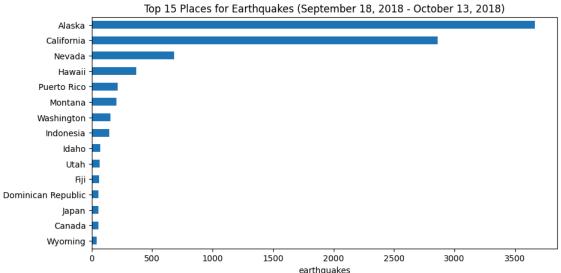
```
# rather than passing the plot type as a param,
# we could directly use it as a method
fb['2018-02':'2018-08'].assign( # gets the data from feb - aug
month=lambda x: x.index.month # creates a new column month
).groupby('month').sum().volume.plot.bar( # creates a bar graph
color='green', rot=0, title='Volume Traded'
)
plt.ylabel('volume')
```

```
Text(0, 0.5, 'volume')
                                Volume Traded
    0.8
    0.4
    0.0
```

month

```
quakes.parsed_place.value_counts().iloc[14::-1,].plot(
  kind='barh', figsize=(10, 5), # creates a horizontal bar graph title='Top 15 Places for Earthquakes '\
'(September 18, 2018 - October 13, 2018)'
plt.xlabel('earthquakes')
```

Text(0.5, 0, 'earthquakes')

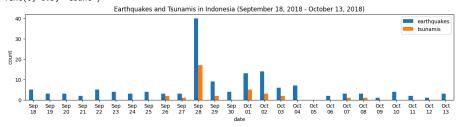


```
quakes.groupby('parsed_place').tsunami.sum().sort_values().iloc[-10::,].plot( # gets the number of tsunamis instead
kind='barh', figsize=(10, 5),
title='Top 10 Places for Tsunamis '\
'(September 18, 2018 - October 13, 2018)'
plt.xlabel('tsunamis')
```

Text(0.5, 0, 'tsunamis')

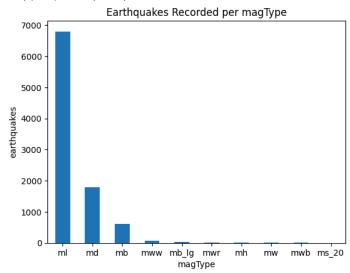
Top 10 Places for Tsunamis (September 18, 2018 - October 13, 2018) Indonesia Papua New Guinea Alaska Russia parsed_place Fiji East Timor Philippines Christmas Island California Mexico 5 10 15 20 25 30 35 tsunamis

```
indonesia_quakes = quakes.query('parsed_place == "Indonesia"').assign( # gets data from indonesia
   time=lambda x: pd.to_datetime(x.time, unit='ms'), # creates a new column time earthquake=1 # and a column earthquake, automatically set to 1
).set_index('time').resample('1D').sum()
indonesia_quakes.index = indonesia_quakes.index.strftime('%b\n%d')
indonesia_quakes.nlock = Indonesia_quakes.index.stritime( %b\n%d )
indonesia_quakes.plot(
y=['earthquake', 'tsunami'], kind='bar', figsize=(15, 3), rot=0, # creates a bar graph
label=['earthquakes', 'tsunamis'],
title='Earthquakes and Tsunamis in Indonesia '\
    '(September 18, 2018 - October 13, 2018)'
plt.xlabel('date')
plt.ylabel('count')
```



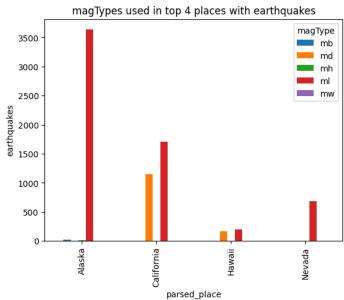
```
quakes.magType.value_counts().plot(
   kind='bar', title='Earthquakes Recorded per magType', rot=0 # creates a bar graph
)
plt.xlabel('magType')
plt.ylabel('earthquakes')
```

Text(0, 0.5, 'earthquakes')

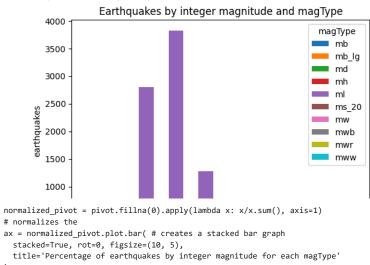


```
quakes[ # gets data from the list below
  quakes.parsed_place.isin(['California', 'Alaska', 'Nevada', 'Hawaii'])
].groupby(['parsed_place', 'magType']).mag.count().unstack().plot.bar( # creates an unstacked bar graph
  title='magTypes used in top 4 places with earthquakes'
)
plt.ylabel('earthquakes')
```

Text(0, 0.5, 'earthquakes')

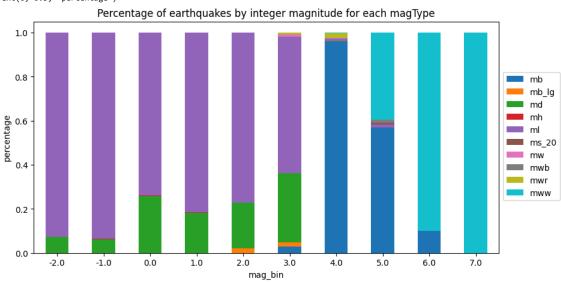


```
pivot = quakes.assign(
   mag_bin=lambda x: np.floor(x.mag)
).pivot_table( # creates a pivot table
   index='mag_bin', columns='magType', values='mag', aggfunc='count' # applying count on mag
)
pivot.plot.bar( # creates a stacked bar graph
   stacked=True, rot=0,
   title='Earthquakes by integer magnitude and magType'
)
plt.ylabel('earthquakes')
```



Text(0, 0.5, 'percentage')

ax.legend(bbox_to_anchor=(1, 0.8))
plt.ylabel('percentage')



 $_{\mathsf{T}}\mathsf{T}$ B I \Longleftrightarrow \square $^{\mathsf{99}}$ \trianglerighteq \boxminus - Ψ $\textcircled{\tiny{}}$ \blacksquare