CPE 311 - Computational Thinking with Python

Gwyneth D. Esperat

CPE22S3

April 02, 2024

Engr. Roman M. Richard

Instructions:

Create a Python notebook to answer all shown procedures, exercises and analysis in this section.

Resources:

Download the following datasets:

- earthquakes-1.csv
- fb_stock_prices_2018.csv

Procedures:

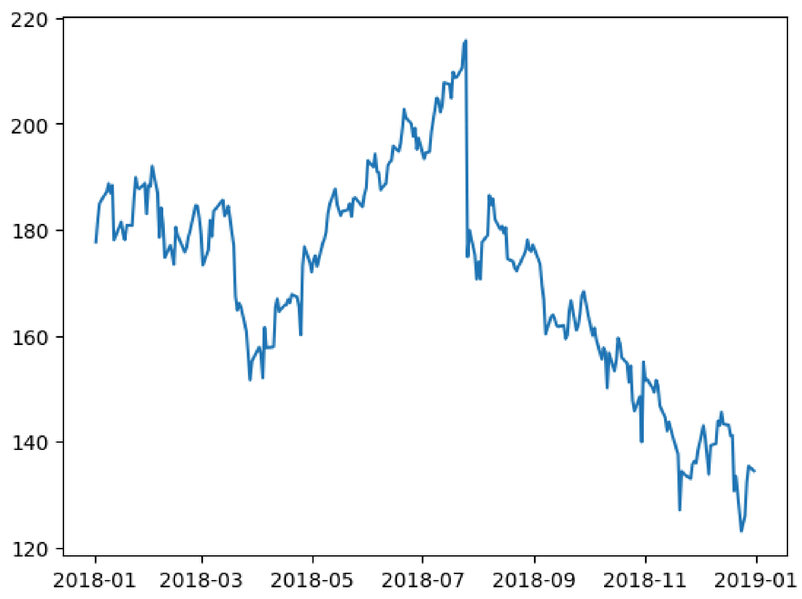
- 9.1 Introduction to Matplotlib
- 9.2 Plotting with Pandas
- 9.3 Pandas Plotting Subpackage

9.1 Introduction to Matplotlib

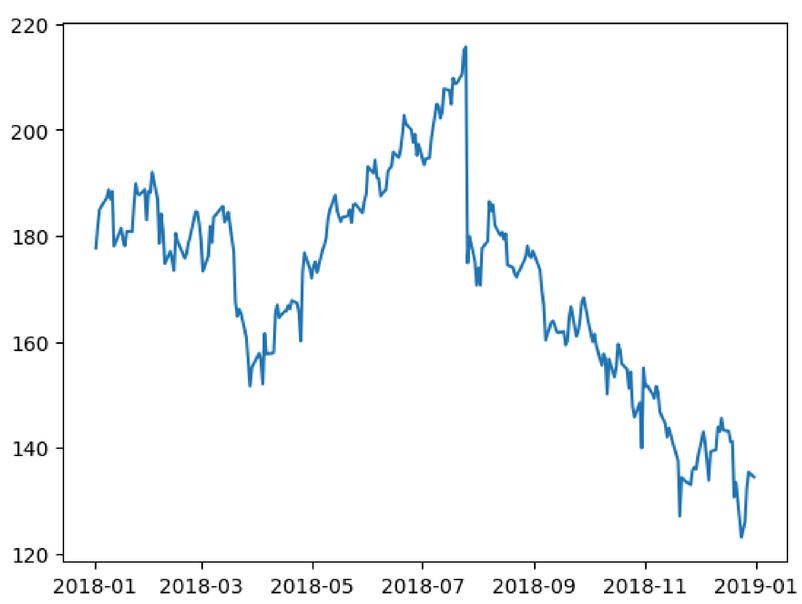
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```
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)
plt.show()
```




```
%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)
plt.show()
```




```
plt.plot('high', 'low', 'ro', data=fb.head(20))
```

```
quakes = pd.read_csv('/content/earthquakes.csv')
plt.hist(quakes.query('magType == "ml"').mag)
```

```
x = quakes.query('magType == "ml"').mag
fig, axes = plt.subplots(1, 2, figsize=(10, 3))
for ax, bins in zip(axes, [7, 35]):
    ax.hist(x, bins=bins)
    ax.set_title(f'bins param: {bins}')
plt.show()
```

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

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

```
fig = plt.figure(figsize=(3, 3))
outside = fig.add_axes([0.1, 0.1, 0.9, 0.9])
inside = fig.add_axes([0.7, 0.7, 0.25, 0.25])
```

```
fig = plt.figure(figsize=(8, 8))
gs = fig.add_gridspec(3, 3)
top_left = fig.add_subplot(gs[0, 0])
mid_left = fig.add_subplot(gs[1, 0])
top_right = fig.add_subplot(gs[:2, 1:])
bottom = fig.add_subplot(gs[2,:])
```

```
fig.savefig('empty.png')
```

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

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

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

```
import random
import matplotlib as mpl

rcparams_list = list(mpl.rcParams.keys())
random.seed(20) # make this repeatable
random.shuffle(rcparams_list)
sorted(rcparams_list[:20])
```

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

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

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

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

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

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

```
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'
)
plt.show()
```

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

```
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'
)
plt.show()
```

```
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
)
plt.show()
```

```
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
)
plt.show()
```

```
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
)
plt.show()
```

```
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', clim=(-1, 1))
fig.colorbar(im)
labels = [col.lower() for col in fb_corr.columns]
ax.set_xticklabels([''] + labels, rotation=45)
ax.set_yticklabels([''] + labels)
plt.show()
```

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

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

```
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
plt.show()
```

```
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
plt.show()
```

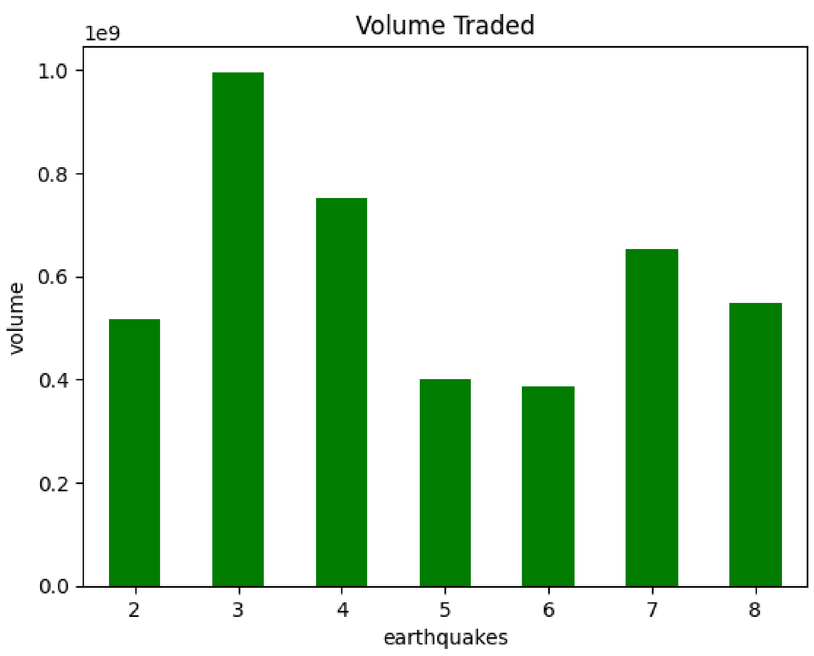
```
from statsmodels.distributions.empirical_distribution import ECDF
ecdf = ECDF(quakes.query('magType == "ml"').mag)
plt.plot(ecdf.x, ecdf.y)

# 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')
```

```
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%')
```

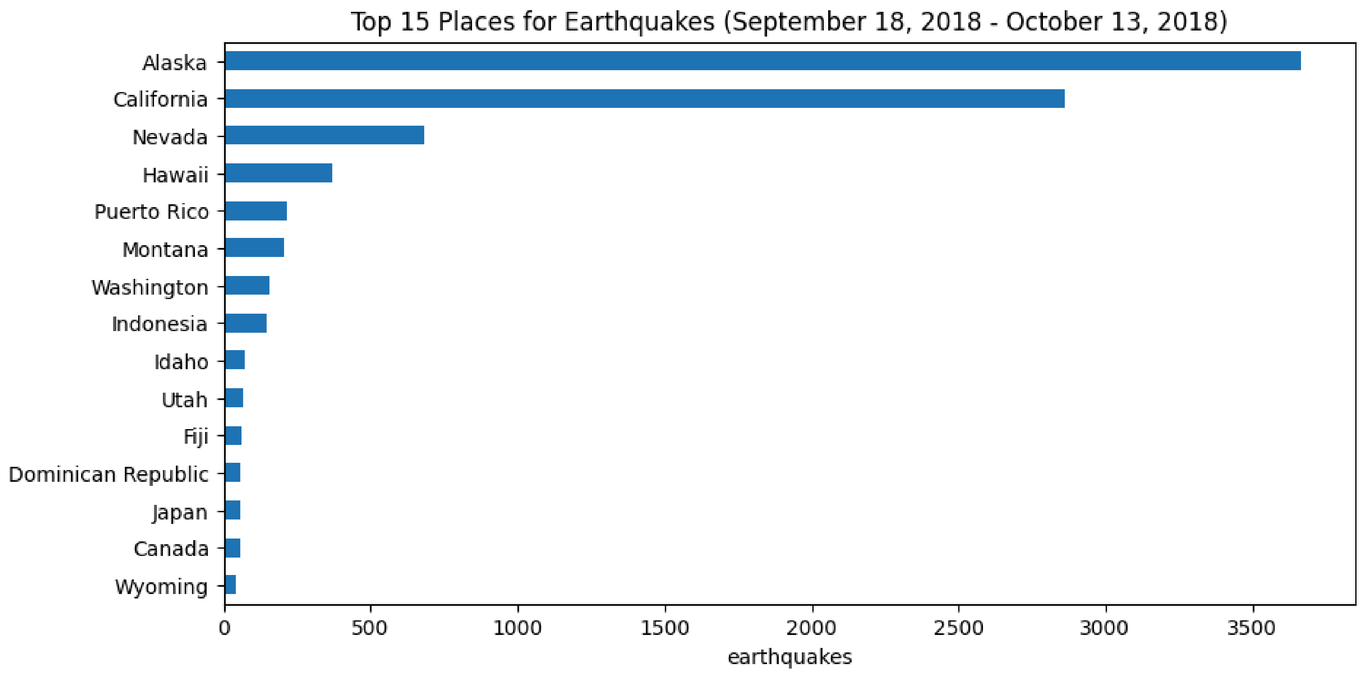
```
fb.iloc[:,:4].plot(kind='box', title='Facebook OHLC Prices Boxplot')
plt.ylabel('price ($)') # label the x-axis (discussed in chapter 6)
```

```
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
plt.show()
```

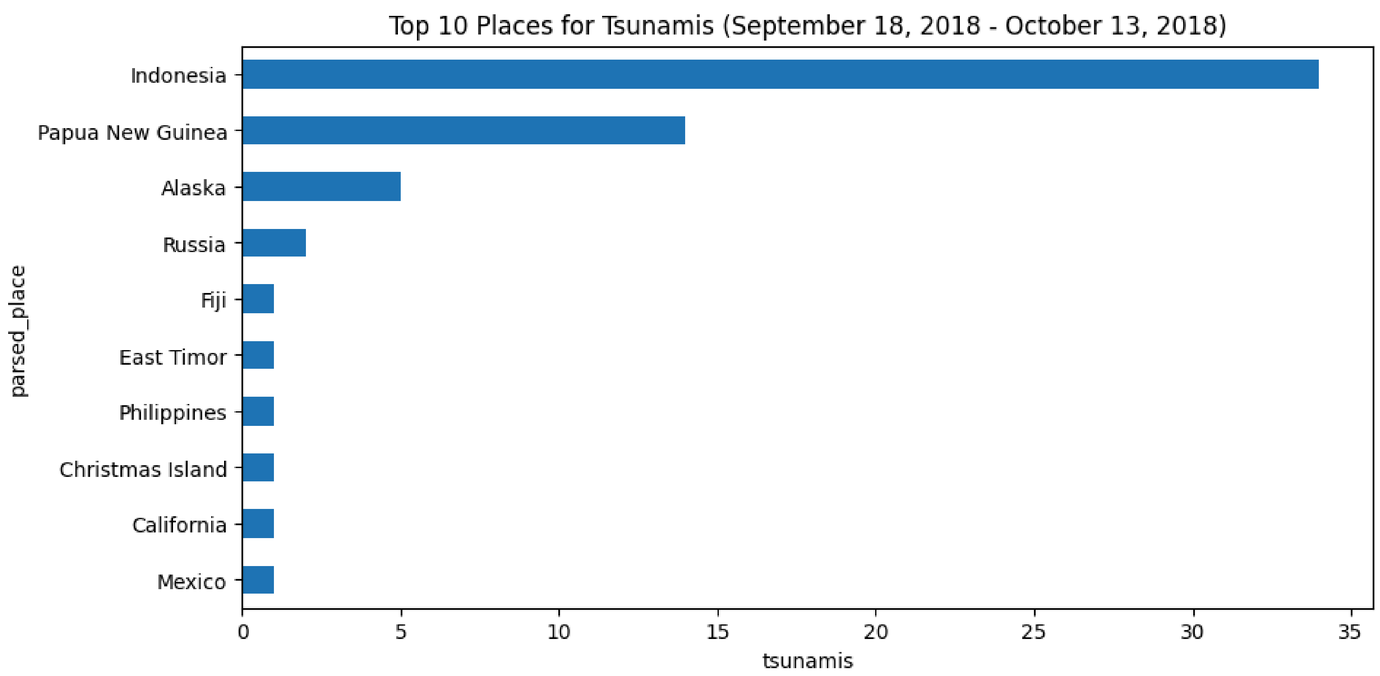
```
fb['2018-02':'2018-08'].assign(
    month=lambda x: x.index.month
).groupby('month').sum().volume.plot.bar(
    color='green', rot=0, title='Volume Traded'
)
plt.xlabel('earthquakes')  # label the x-axis
plt.ylabel('volume')  # label the y-axis
plt.show()
```




```
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
plt.show()
```

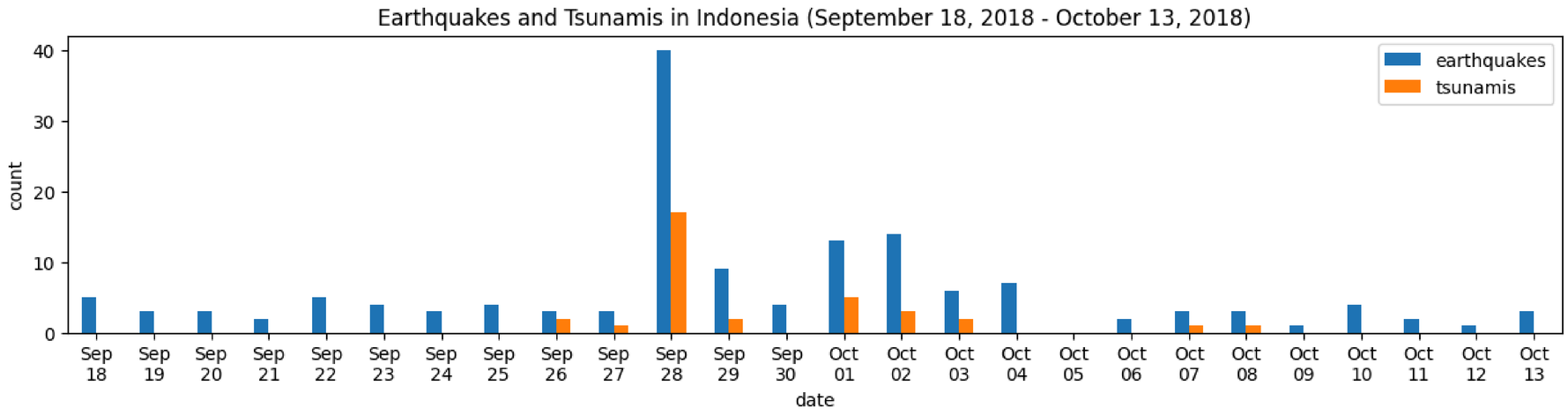



```
quakes.groupby('parsed_place').tsunami.sum().sort_values().iloc[-10:].plot(
    kind='barh',
    figsize=(10, 5),
    title='Top 10 Places for Tsunamis (September 18, 2018 - October 13, 2018)'
)
plt.xlabel('tsunamis')  # label the x-axis
plt.show()
```

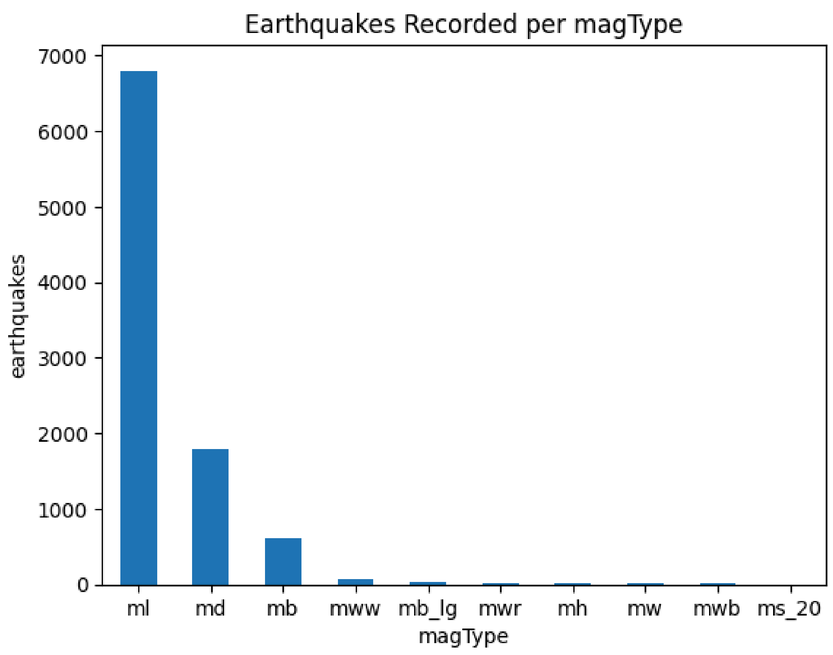



```
indonesia_quakes = quakes.query('parsed_place == "Indonesia"').assign(
    time=lambda x: pd.to_datetime(x.time, unit='ms'),
    earthquake=1
).set_index('time').resample('1D').sum()
indonesia_quakes.index = indonesia_quakes.index.strftime('%b\n%d')
indonesia_quakes.plot(
    y=['earthquake', 'tsunami'],
    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
plt.xlabel('date')
plt.ylabel('count')
plt.show()
```

```
<ipython-input-205-48b997b4dff8>: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 sele
  ).set_index('time').resample('1D').sum()
```

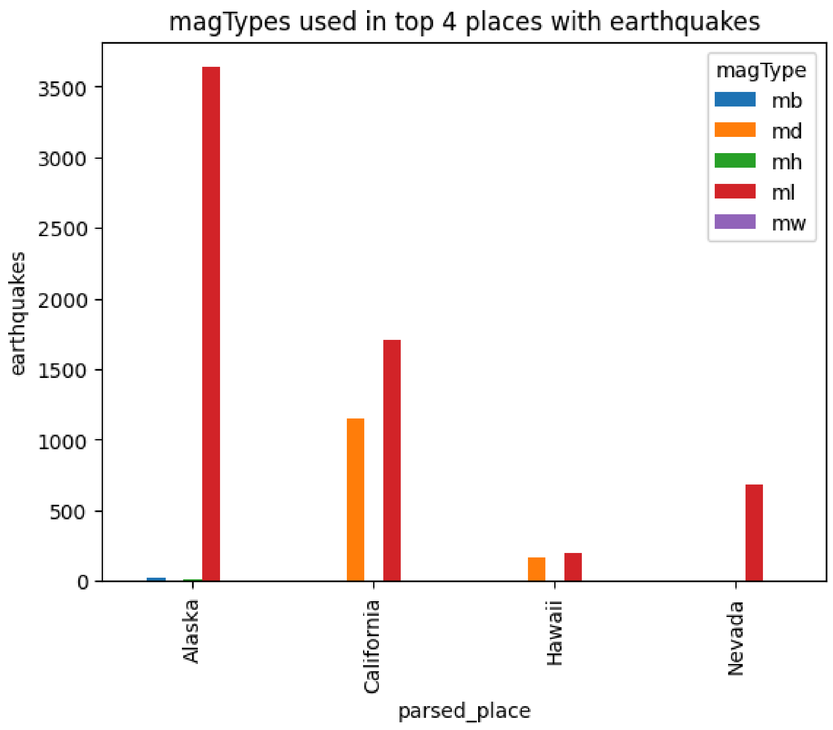



```
quakes.magType.value_counts().plot(
    kind='bar',
    title='Earthquakes Recorded per magType',
    rot=0
)
# label the axes
plt.xlabel('magType')
plt.ylabel('earthquakes')
plt.show()
```



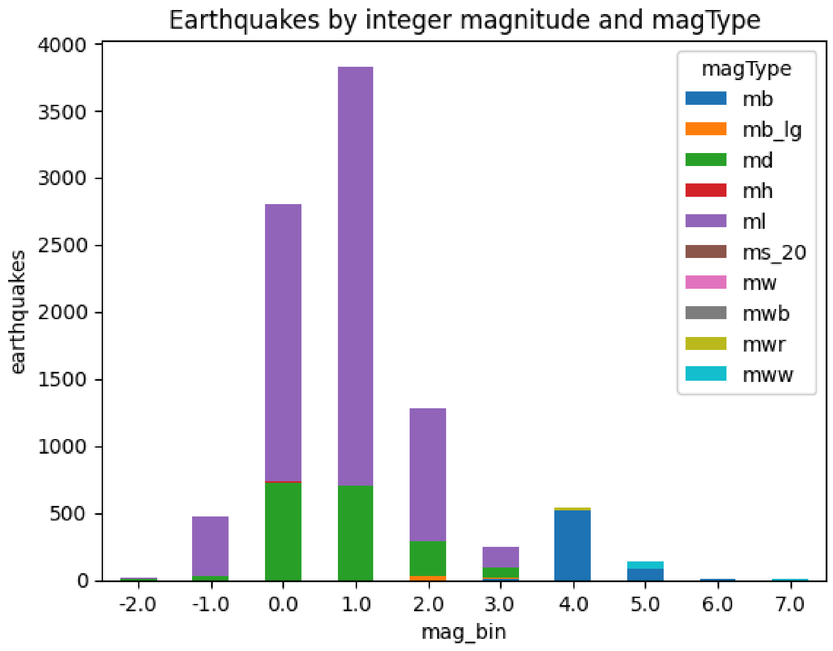

```
quakes[
quakes.parsed_place.isin(['California', 'Alaska', 'Nevada', 'Hawaii'])
].groupby(['parsed_place', 'magType']).mag.count().unstack().plot.bar(
title='magTypes used in top 4 places with earthquakes'
)
plt.ylabel('earthquakes') # label the axes (discussed in chapter 6)
```

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



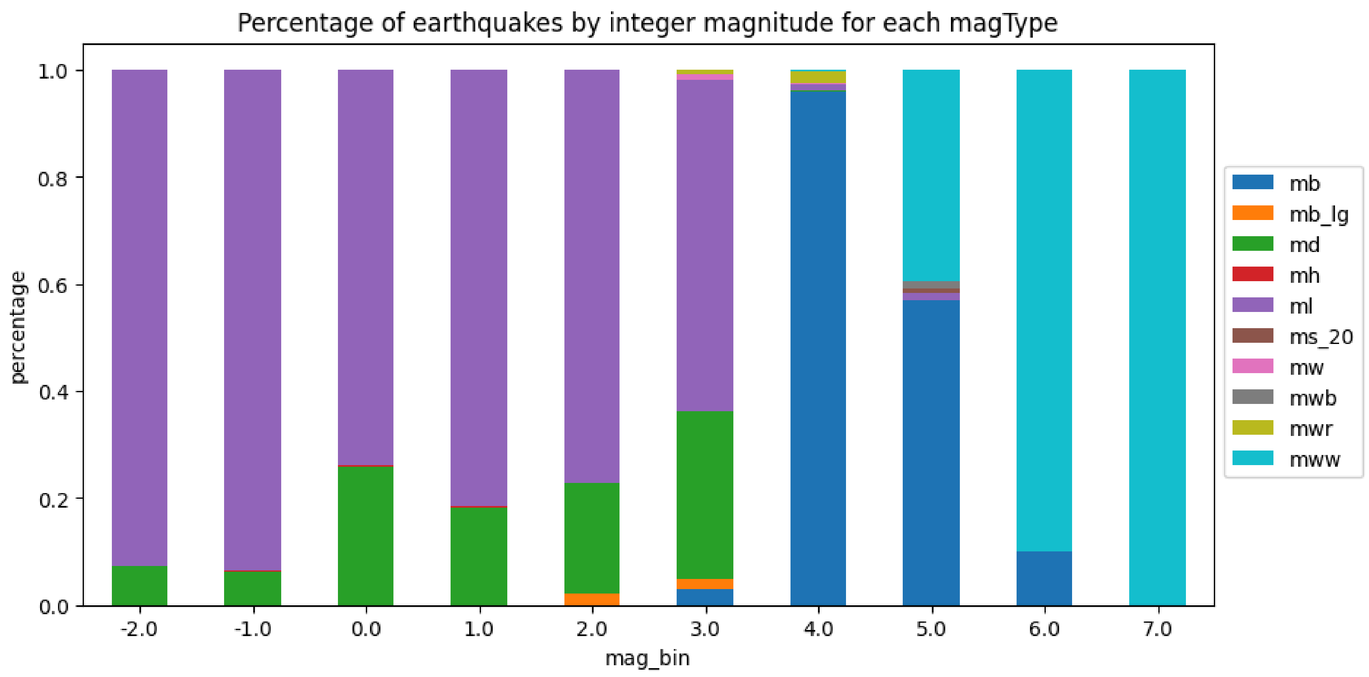

```
pivot = quakes.assign(
mag_bin=lambda x: np.floor(x.mag)
).pivot_table(
index='mag_bin', columns='magType', values='mag', aggfunc='count'
)
pivot.plot.bar(
stacked=True, rot=0,
title='Earthquakes by integer magnitude and magType'
)
plt.ylabel('earthquakes') # label the axes (discussed in chapter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

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



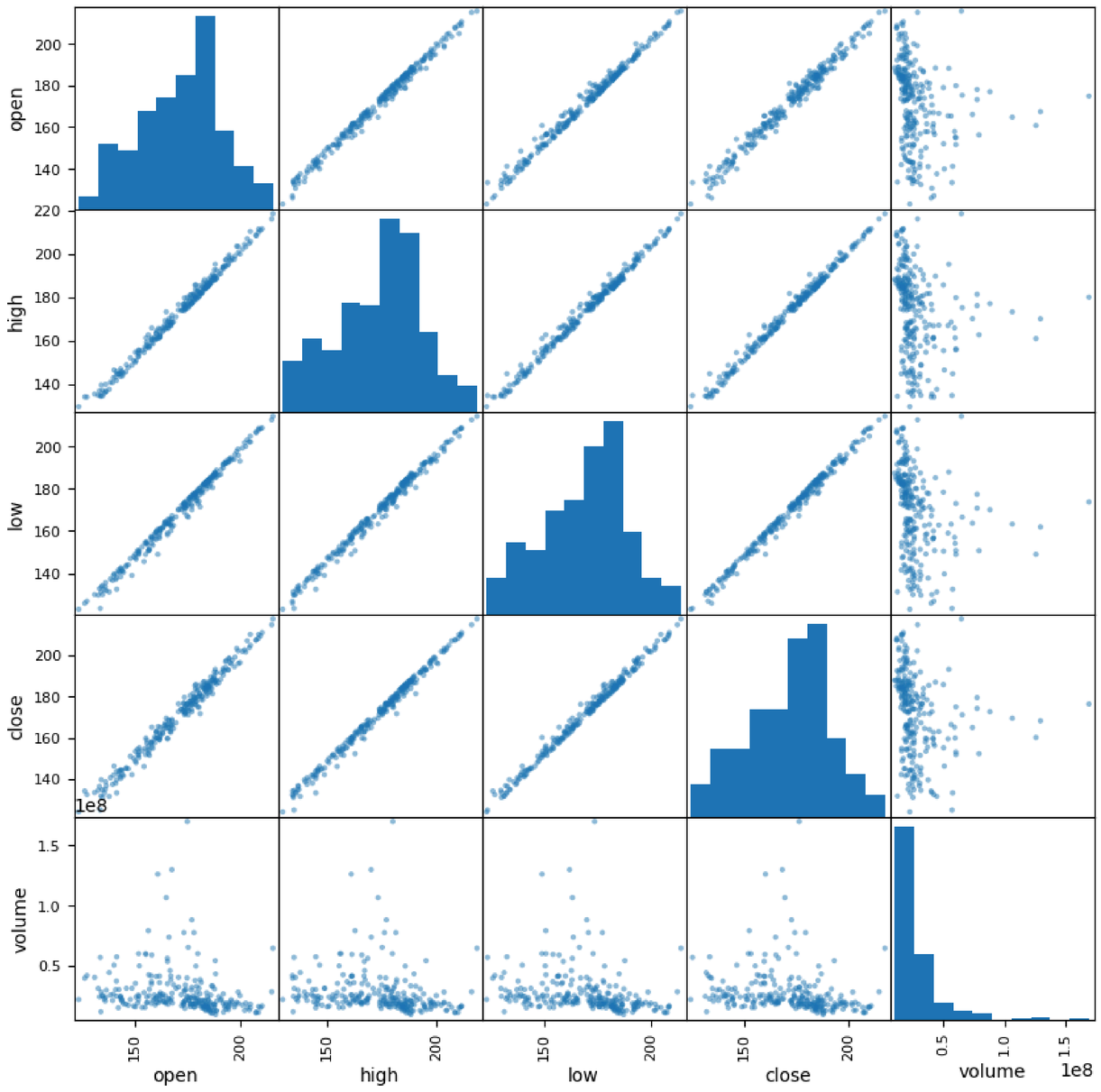

```
normalized_pivot = pivot.fillna(0).apply(lambda x: x/x.sum(), axis=1)
ax = normalized_pivot.plot.bar(
stacked=True, rot=0, figsize=(10, 5),
title='Percentage of earthquakes by integer magnitude for each magType'
)
ax.legend(bbox_to_anchor=(1, 0.8)) # move legend to the right of the plot
plt.ylabel('percentage')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

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




```
%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))
```

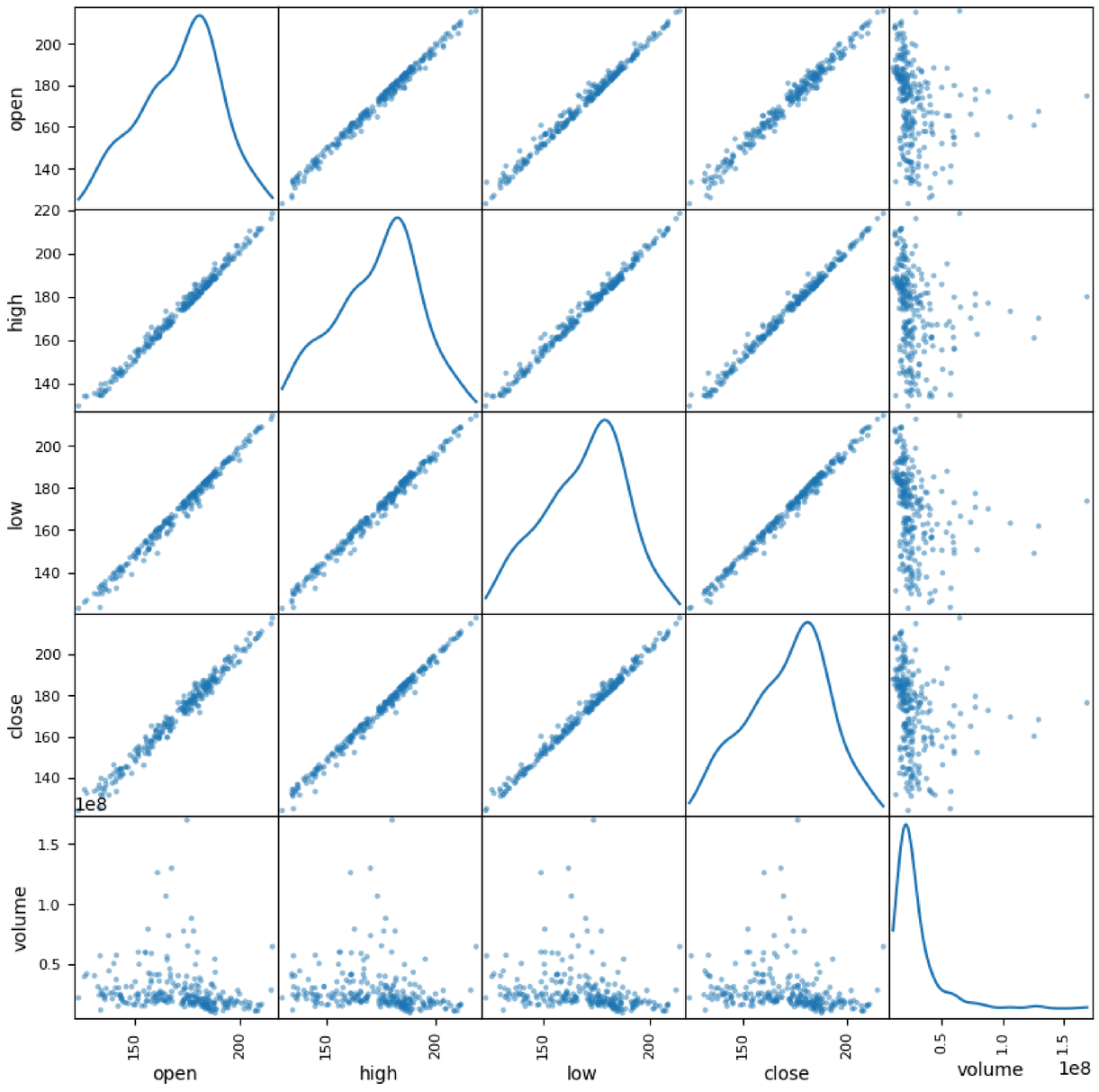
```
array([[<Axes: xlabel='open', ylabel='open'>,
        <Axes: xlabel='high', ylabel='open'>,
        <Axes: xlabel='low', ylabel='open'>,
        <Axes: xlabel='close', ylabel='open'>,
        <Axes: xlabel='volume', ylabel='open'>],
       [<Axes: xlabel='open', ylabel='high'>,
        <Axes: xlabel='high',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



Double-click (or enter) to edit

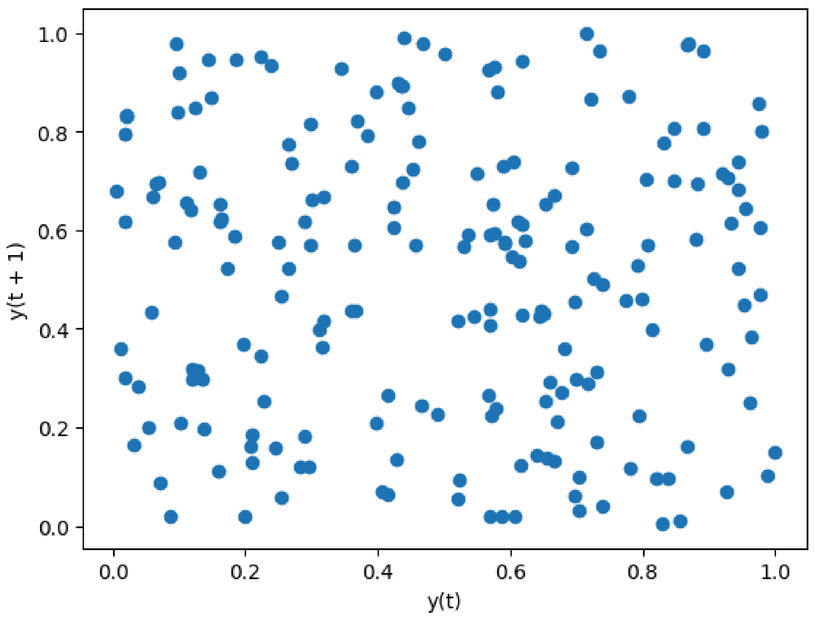
```
scatter_matrix(fb, figsize=(10, 10), diagonal='kde')
```

```
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)
```



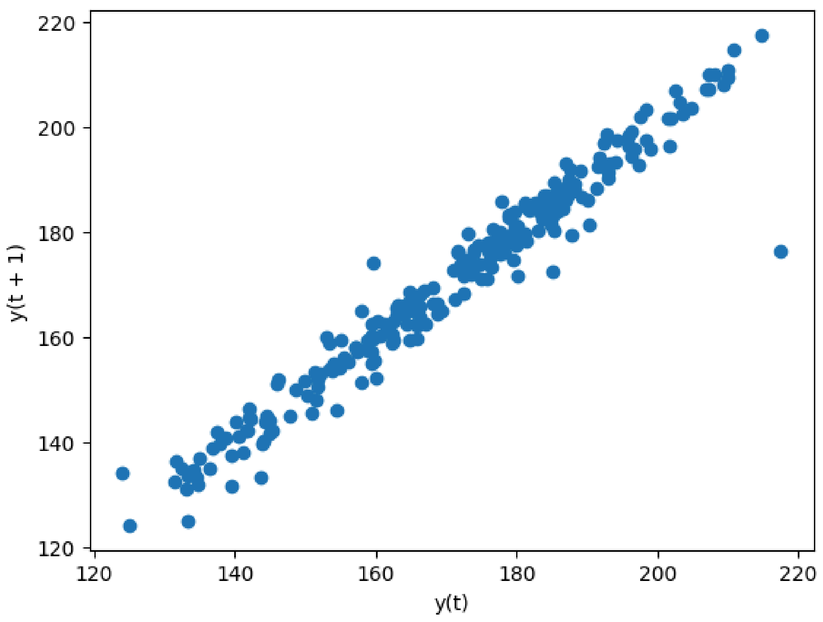

```
from pandas.plotting import lag_plot
np.random.seed(0) # make this repeatable
lag_plot(pd.Series(np.random.random(size=200)))
```

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



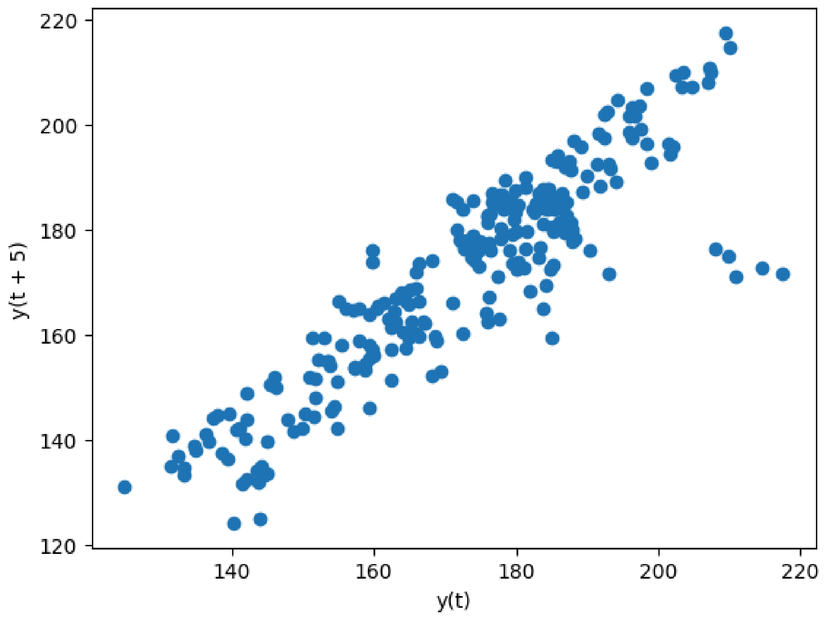

```
lag_plot(fb.close)
```

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



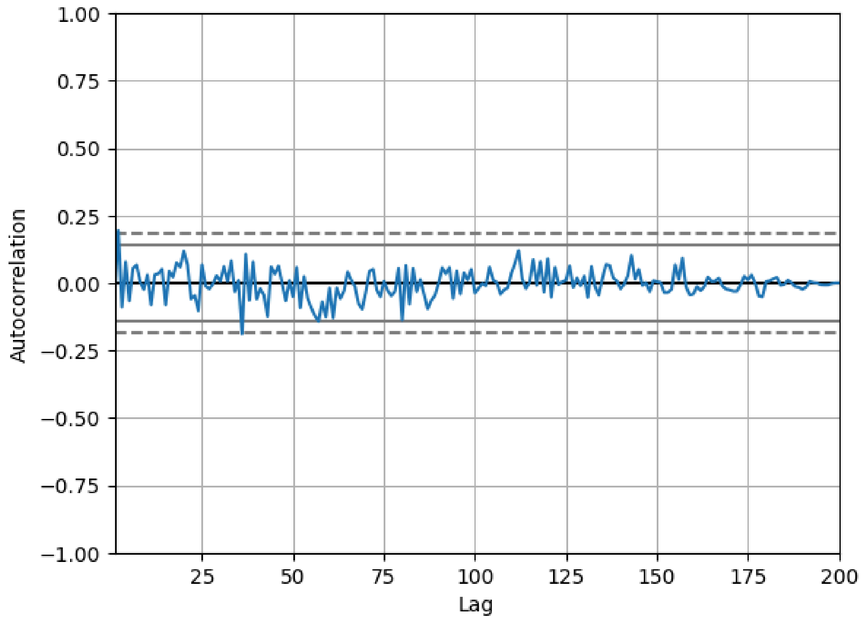

```
lag_plot(fb.close, lag=5)
```

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



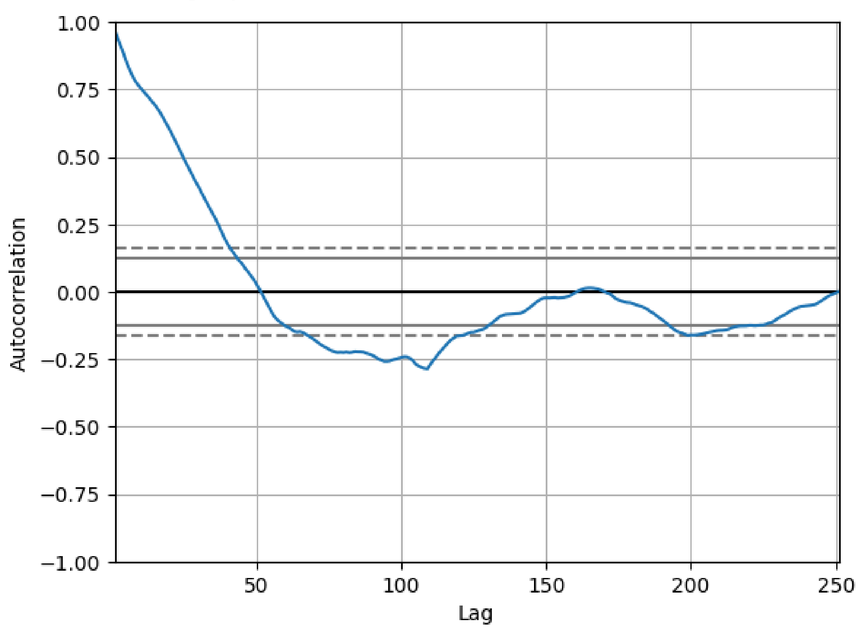

```
from pandas.plotting import autocorrelation_plot
np.random.seed(0) # make this repeatable
autocorrelation_plot(pd.Series(np.random.random(size=200)))
```

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

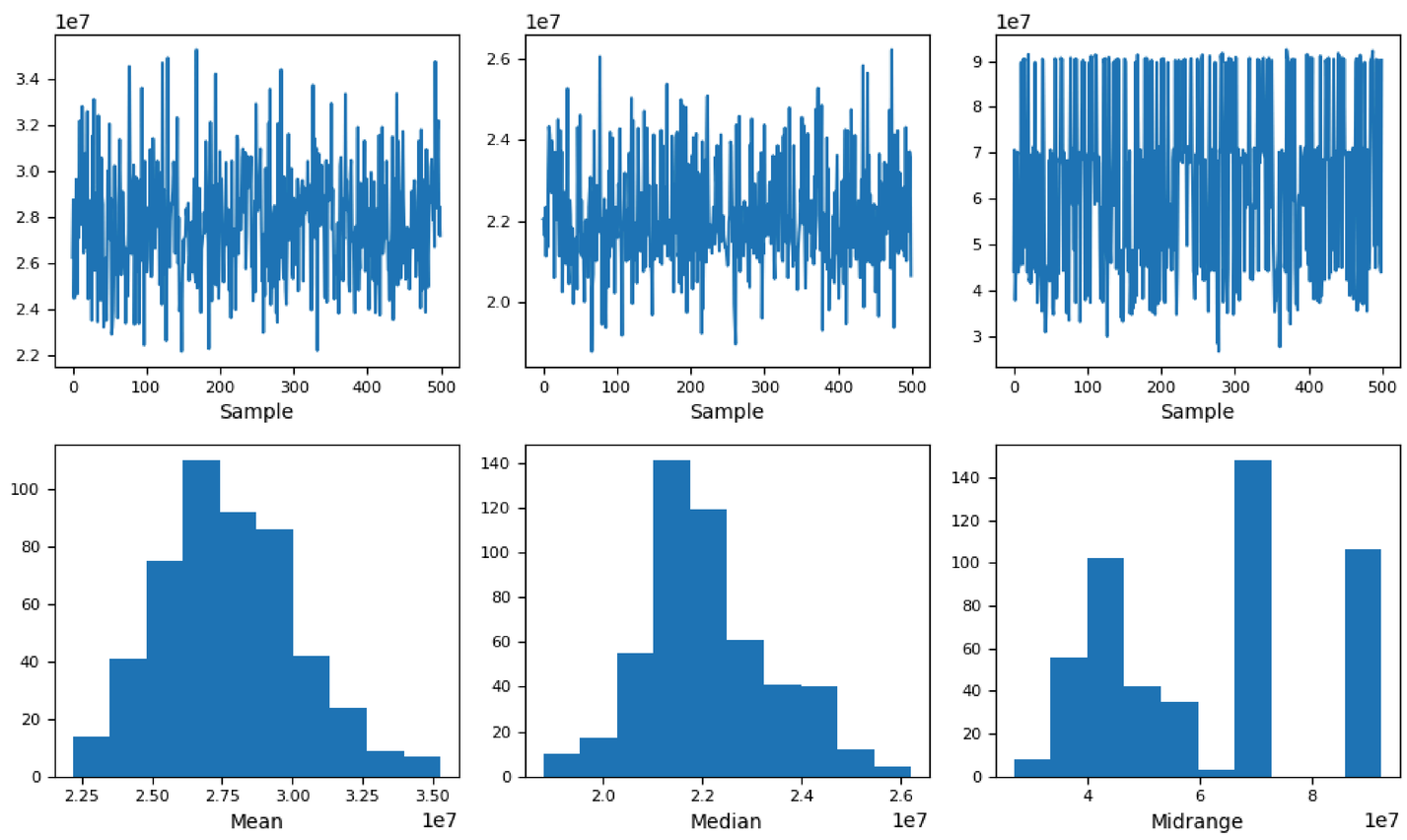



```
autocorrelation_plot(fb.close)
```

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




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



Data Analysis

Facebook's 2018 Stock Prices Analysis:

1. **Basic Line Plot of Open Prices**: Plotting Facebook's open prices over time allows us to visually assess trends, volatility, and any significant price movements within the year. Sharp increases or drops could indicate market reactions to news or events related to the company.

2. **Scatter Plot of High vs. Low Prices**: A scatter plot comparing daily high and low prices can reveal the daily price volatility. Days with wider spreads between high and low prices suggest greater volatility, which might be driven by significant news events or market conditions.
3. **Histogram of Earthquake Magnitudes**: While this plot pertains to earthquake data, if applied to Facebook's stock price changes, a histogram could show the distribution of daily price changes, helping to understand the most common price movement ranges.
4. **Subplots**: Creating subplots for different stock attributes (open, close, high, low, and volume) over the same time period can help in comparing these metrics side-by-side to spot correlations or divergences that might not be apparent when viewed in isolation.
5. **Scatter Matrix**: This visualization aids in examining the relationships between all numerical columns in the dataset, potentially revealing correlations between variables like volume and price changes, which could indicate patterns of buying or selling pressure following price movements.
6. **Lag Plot**: Analyzing the lag plot of Facebook's closing prices could help identify patterns or autocorrelation in the data, indicating whether past prices can be predictive of future prices, a crucial insight for time series forecasting.
7. **Autocorrelation Plot**: Similar to the lag plot, this visualization can further delve into the predictability of the stock's movements based on its past values, which is fundamental for models like ARIMA used in financial forecasting.
8. **Bootstrap Plot**: By examining the bootstrap plot of trading volume, we can understand the variability and confidence intervals for mean, median, and other statistics of the volume, providing insights into the stability of trading activity over time.

Earthquake Data Analysis:

1. **Hexbins and Scatter Plots**: When applied to Facebook's data, methods like scatter plots and hexbins could be used to explore the relationship between trading volume and price movements. For example, high volumes might coincide with significant price changes, indicating key events or market reactions.
2. **Heatmaps**: Using a heatmap to visualize the correlation matrix of Facebook's stock metrics (like open, close, high, low, and volume) could reveal how these variables move in relation to each other. Strong correlations might suggest that certain metrics consistently move together, indicating underlying patterns in the stock's behavior.

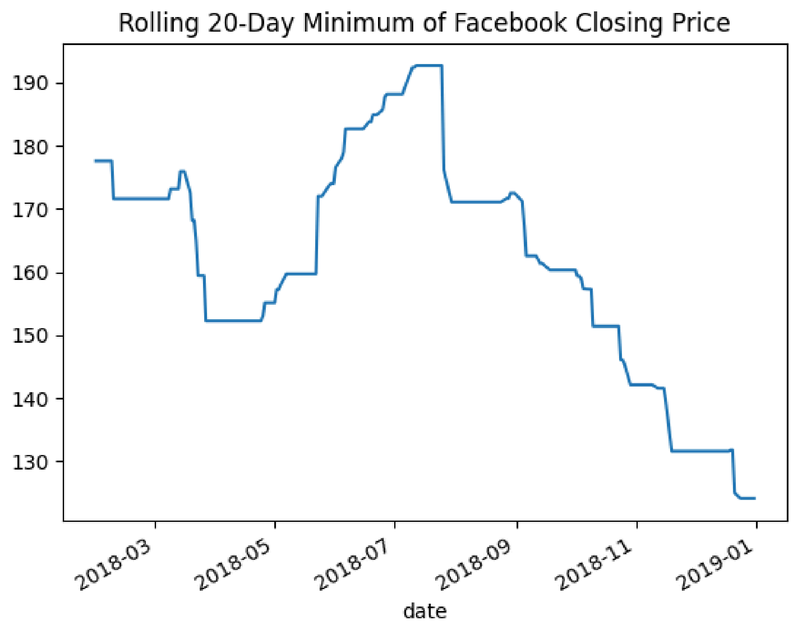
Supplementary Activity:

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.

```
fb['close'].rolling(window=20).min().plot(title='Rolling 20-Day Minimum of Facebook Closing Price')
```

```
<Axes: title={'center': 'Rolling 20-Day Minimum of Facebook Closing Price'}, xlabel='date'>
```

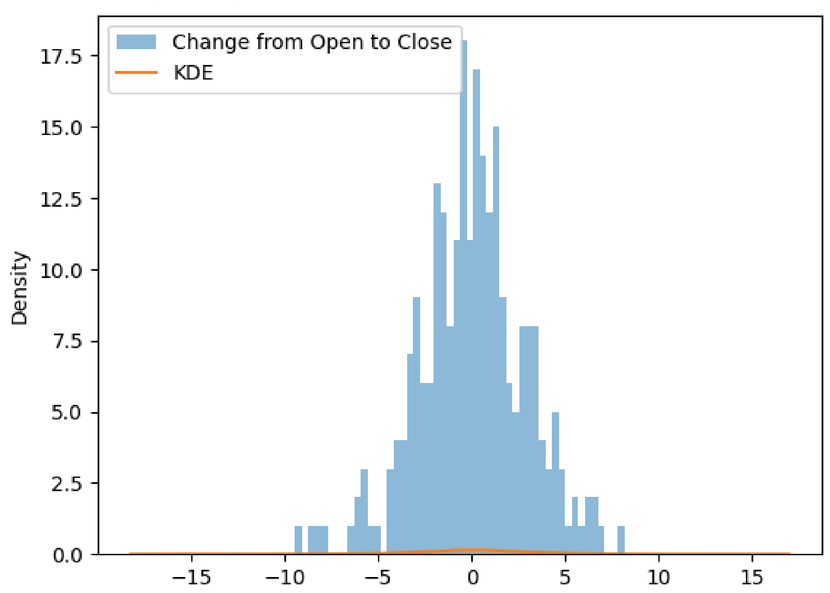


This plot will show the lowest closing price of Facebook stock over each rolling 20-day period, providing insight into short-term lows and potential support levels.

2. Create a histogram and KDE of the change from open to close in the price of Facebook stock.

```
(fb['close'] - fb['open']).plot(kind='hist', bins=50, alpha=0.5, label='Change from Open to Close')
(fb['close'] - fb['open']).plot(kind='kde', label='KDE')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x783bebe8a620>
```



This visualization compares the distribution of Facebook's daily price changes from open to close, highlighting the volatility and how often prices increase versus decrease during the trading day.

3. Using the earthquake data, create box plots for the magnitudes of each magType used in Indonesia.

```
quakes_indonesia = quakes[quakes['place'].str.contains('Indonesia')]
quakes_indonesia.boxplot(column='mag', by='magType', figsize=(10, 6))
```