Hands-on Activity 9.2 Customized Visualizations using Seaborn

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:

- 9.4 Introduction to Seaborn
- 9.5 Formatting Plots
- 9.6 Customizing Visualizations

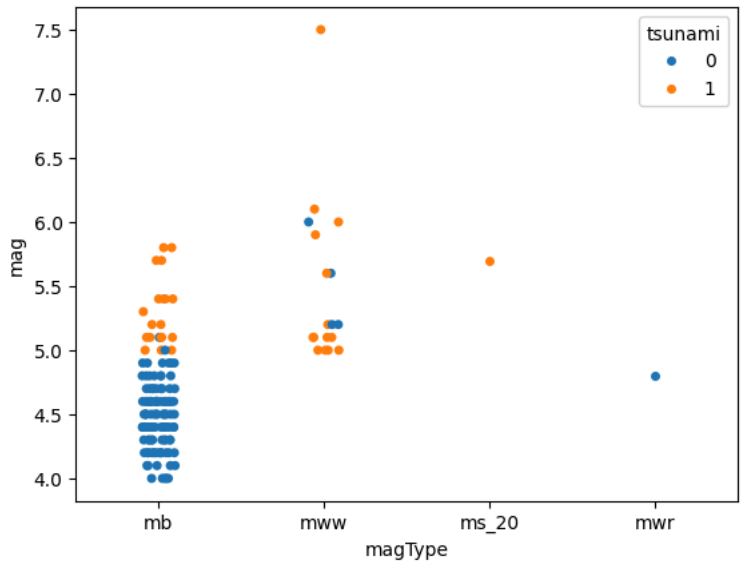
9.4 Introduction to Seaborn

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
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')


# categorical data

quakes.assign(
    time=lambda x: pd.to_datetime(x.time, unit='ms')
).set_index('time').loc['2018-09-28'].query(
    "parsed_place == 'Indonesia' and tsunami == 1 and mag == 7.5"
)
```

time	mag	magType	place	tsunami	parsed_place
2018-09-28 10:02:43.480	7.5	mww	78km N of Palu, Indonesia	1	Indonesia

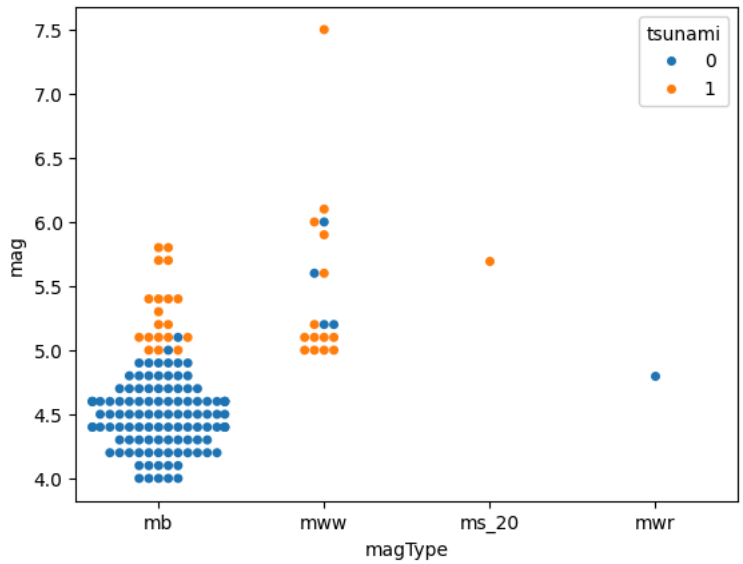
```
# stripplot

sns.stripplot(
    x='magType',
    y='mag',
    hue='tsunami',
    data=quakes.query('parsed_place == "Indonesia"')
)
```

```
<Axes: xlabel='magType', ylabel='mag'>
```



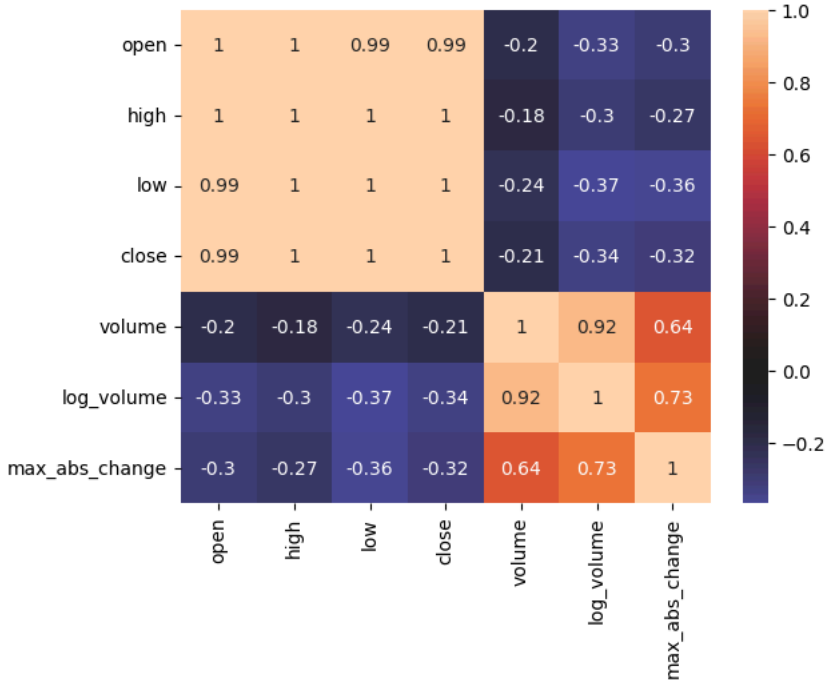

```
# swarmplot

sns.swarmplot(
    x='magType',
    y='mag',
    hue='tsunami',
    data=quakes.query('parsed_place == "Indonesia"')
)
```

```
<Axes: xlabel='magType', ylabel='mag'>
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 10.2%
  warnings.warn(msg, UserWarning)
```



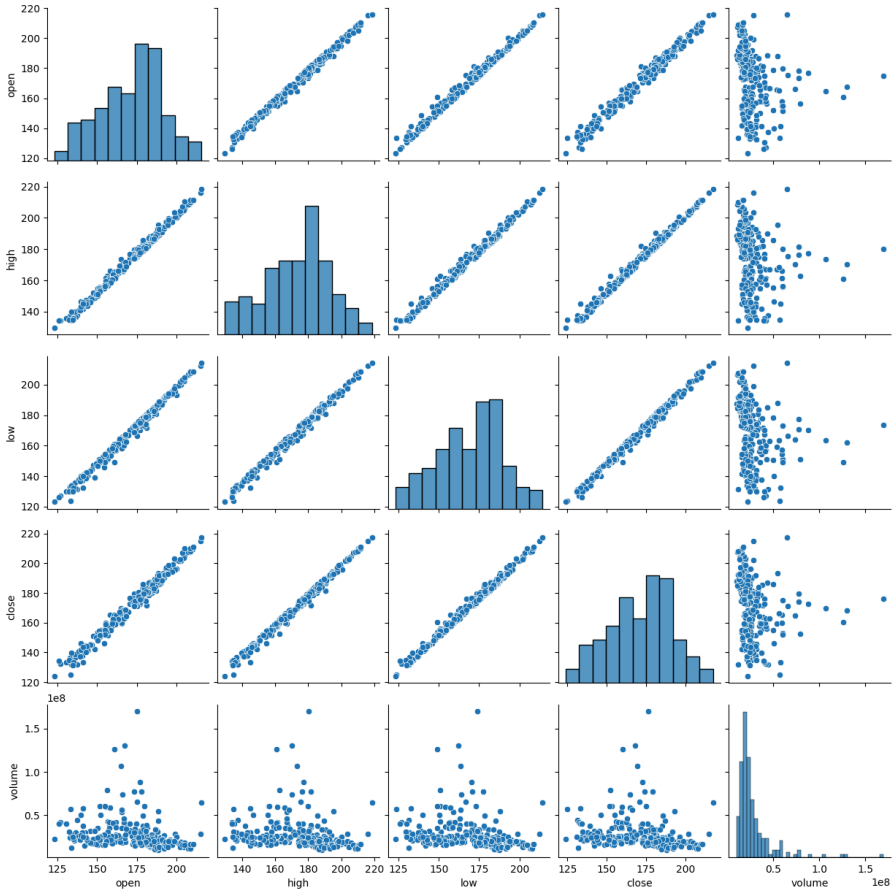

```
# heatmap

sns.heatmap(
    fb.sort_index().assign(
    log_volume=np.log(fb.volume),
    max_abs_change=fb.high - fb.low
).corr(),
    annot=True, center=0
)
```

<Axes: >



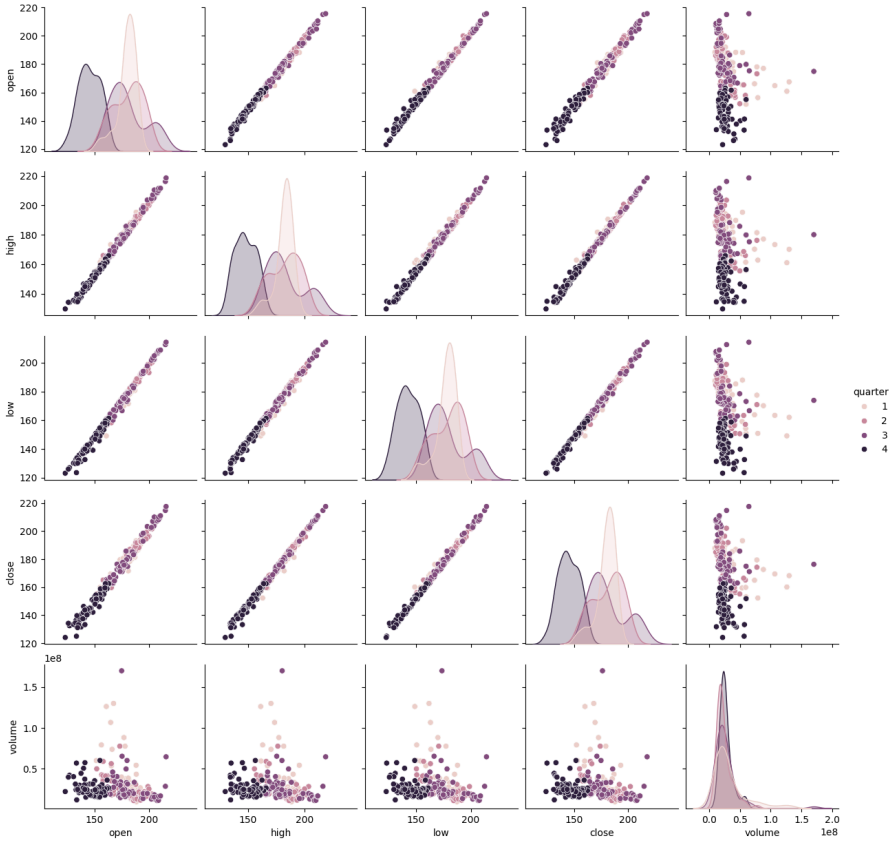

```
# pairplot

sns.pairplot(fb)
```

<seaborn.axisgrid.PairGrid at 0x7da4048ec520>



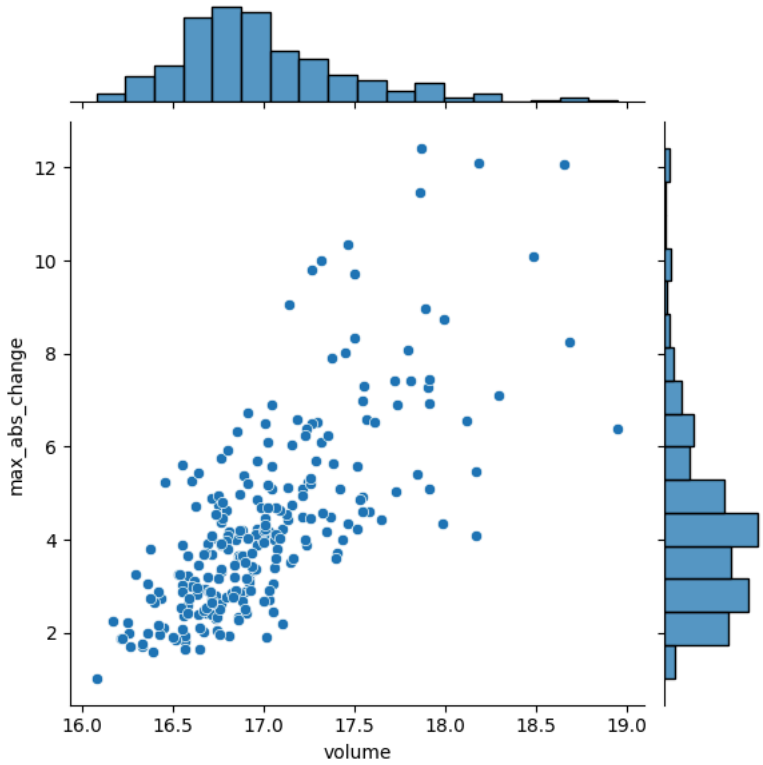

```
sns.pairplot(
    fb.assign(quarter=lambda x: x.index.quarter),
    diag_kind='kde',
    hue='quarter'
)
```

<seaborn.axisgrid.PairGrid at 0x7da4048efb20>



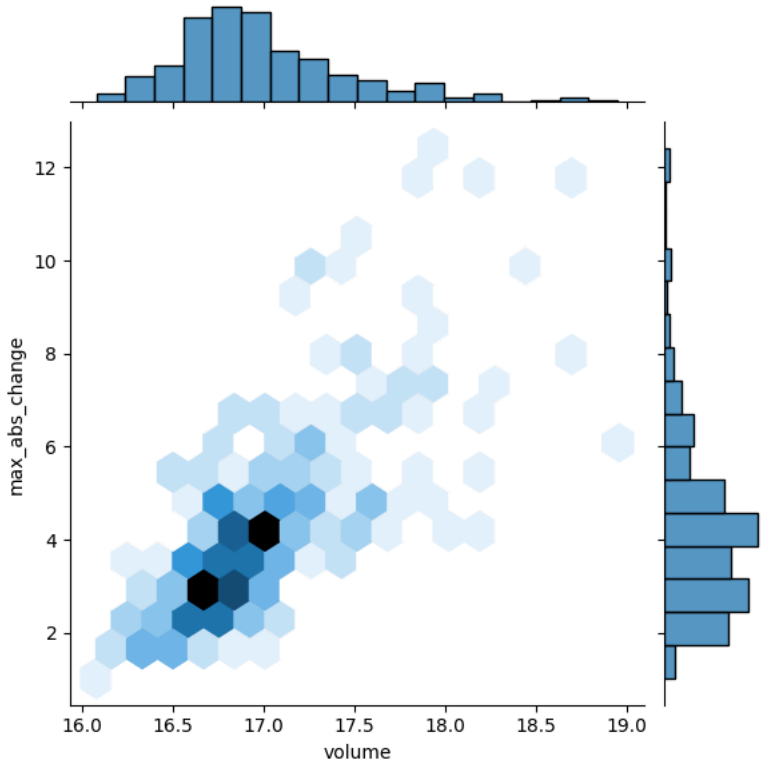
```
# jointplot

sns.jointplot(
    x='volume',
    y='max_abs_change',
    data=fb.assign(
        volume=np.log(fb.volume),
        max_abs_change=fb.high - fb.low
        )
)
```

```
<seaborn.axisgrid.JointGrid at 0x7da4036553f0>
```



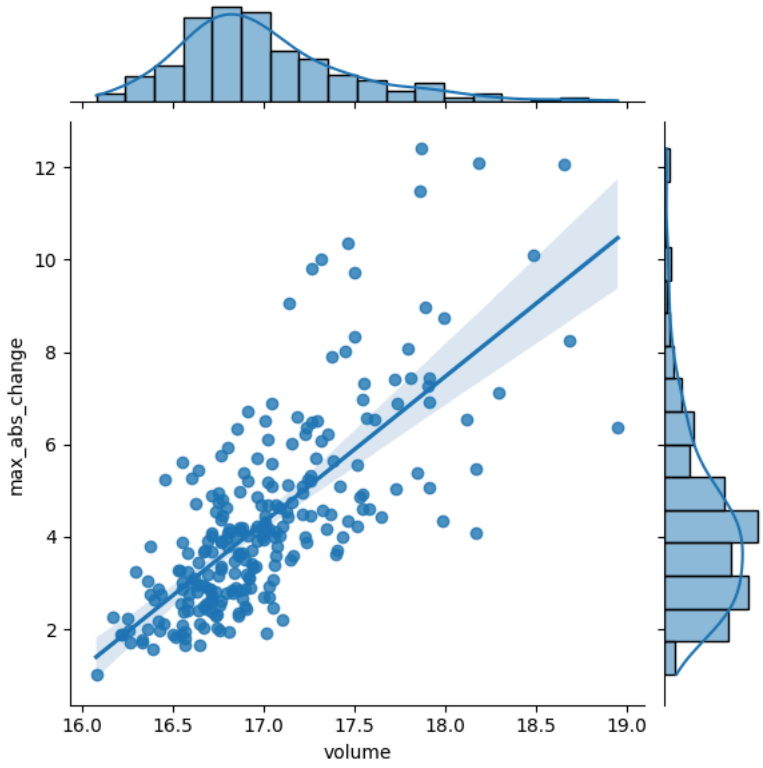

```
sns.jointplot(
    x='volume',
    y='max_abs_change',
    kind='hex',
    data=fb.assign(
        volume=np.log(fb.volume),
        max_abs_change=fb.high - fb.low
        )
)
```

<seaborn.axisgrid.JointGrid at 0x7da403b7a020>



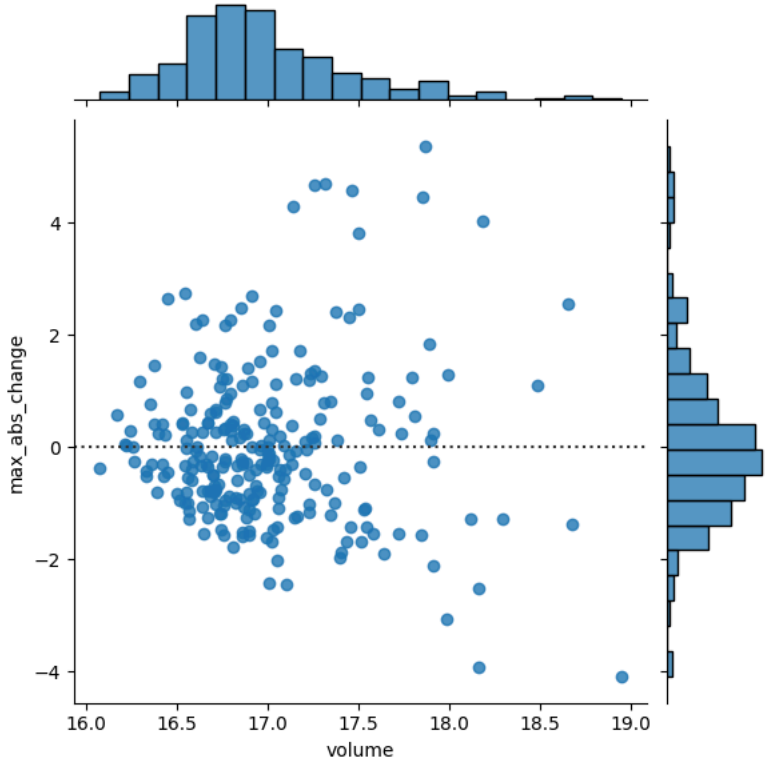

```
sns.jointplot(
    x='volume',
    y='max_abs_change',
    kind='reg',
    data=fb.assign(
        volume=np.log(fb.volume),
        max_abs_change=fb.high - fb.low
        )
)
```

<seaborn.axisgrid.JointGrid at 0x7da4030e6800>



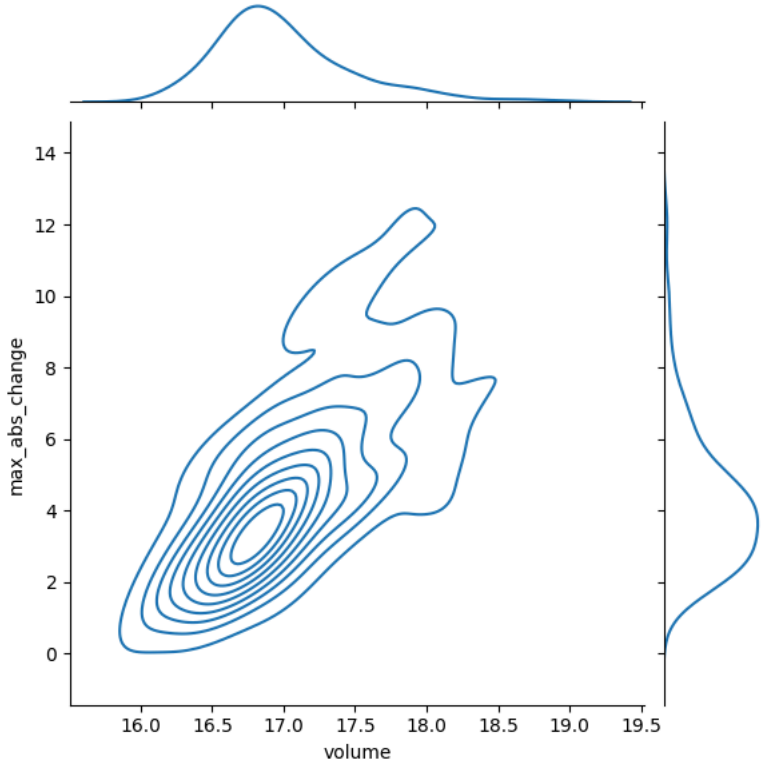
```
sns.jointplot(
    x='volume',
    y='max_abs_change',
    kind='resid',
    data=fb.assign(
        volume=np.log(fb.volume),
        max_abs_change=fb.high - fb.low
        )
)
```

```
<seaborn.axisgrid.JointGrid at 0x7da40305b280>
```




```
sns.jointplot(
    x='volume',
    y='max_abs_change',
    kind='kde',
    data=fb.assign(
        volume=np.log(fb.volume),
        max_abs_change=fb.high - fb.low
        )
)
```

```
<seaborn.axisgrid.JointGrid at 0x7da402f17d60>
```




```
# regression plots

fb_reg_data = fb.assign(
    volume=np.log(fb.volume),
    max_abs_change=fb.high - fb.low
).iloc[:,-2:]


import itertools


iterator = itertools.repeat("I'm an iterator", 1)

for i in iterator:
  print(f'-->{i}')
print('This printed once because the iterator has been exhausted')
for i in iterator:
  print(f'-->{i}')
```

```
-->I'm an iterator
This printed once because the iterator has been exhausted
```

```
iterable = list(itertools.repeat("I'm an iterable", 1))

for i in iterable:
  print(f'-->{i}')
print('This prints again because it\'s an iterable:')
for i in iterable:
  print(f'-->{i}')
```

```
-->I'm an iterable
This prints again because it's an iterable:
-->I'm an iterable
```

```
from reg_resid_plot import reg_resid_plots
reg_resid_plots(fb_reg_data)
```

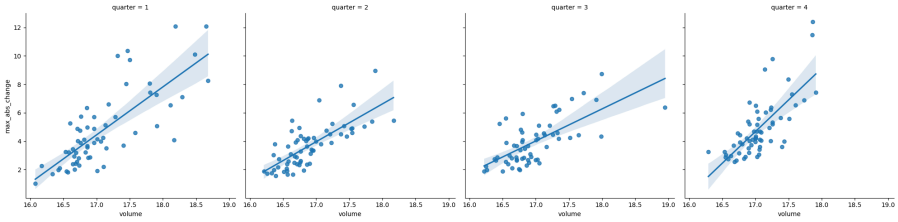
```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-81-ae2d095ec697> in <cell line: 1>()
----> 1 from reg_resid_plot import reg_resid_plots
      2 reg_resid_plots(fb_reg_data)

ModuleNotFoundError: No module named 'reg_resid_plot'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

OPEN EXAMPLES

Next steps: Explain error

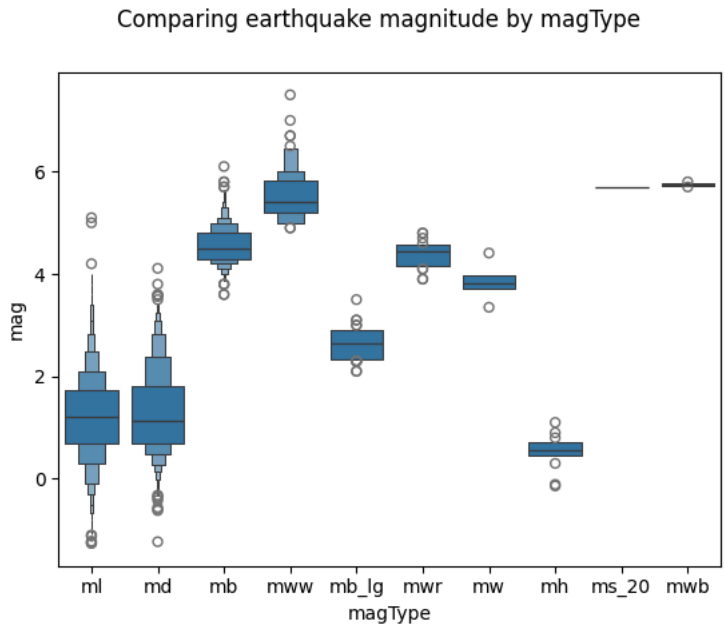
```
sns.lmplot(
    x='volume',
    y='max_abs_change',
    data=fb.assign(
        volume=np.log(fb.volume),
        max_abs_change=fb.high - fb.low,
        quarter=lambda x: x.index.quarter
        ),
    col='quarter'
)
```

```
<seaborn.axisgrid.FacetGrid at 0x7da40ef08dc0>
```



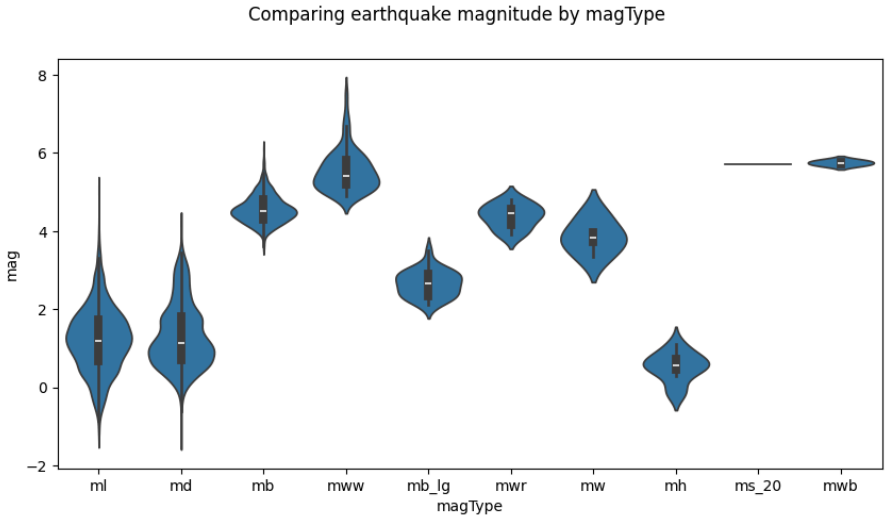

```
# boxenplot

sns.boxenplot(
    x='magType', y='mag', data=quakes[['magType', 'mag']]
)
plt.suptitle('Comparing earthquake magnitude by magType')
```

```
Text(0.5, 0.98, 'Comparing earthquake magnitude by magType')
```



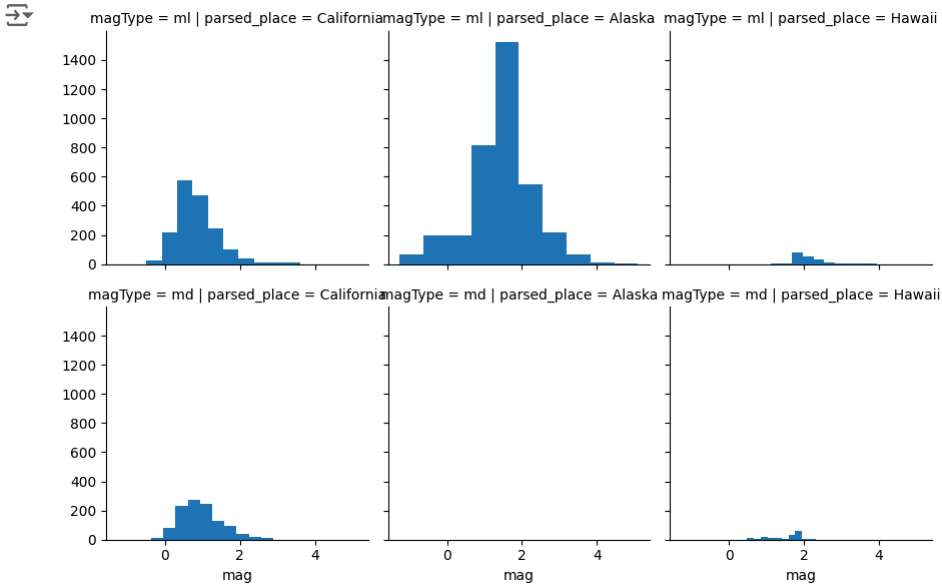

```
# violinplot

fig, axes = plt.subplots(figsize=(10, 5))
sns.violinplot(
    x='magType', y='mag', data=quakes[['magType', 'mag']],
    ax=axes, scale='width' # all violins have same width
)
plt.suptitle('Comparing earthquake magnitude by magType')
```

```
<ipython-input-84-479ebbbdd49f>:4: FutureWarning:

The `scale` parameter has been renamed and will be removed in v0.15.0. Pass `density_nor
  sns.violinplot(
Text(0.5, 0.98, 'Comparing earthquake magnitude by magType')
```




```
# faceting

g = sns.FacetGrid(
    quakes[
        (quakes.parsed_place.isin([
            'California', 'Alaska', 'Hawaii'
        ]))\
        & (quakes.magType.isin(['ml', 'md']))
```

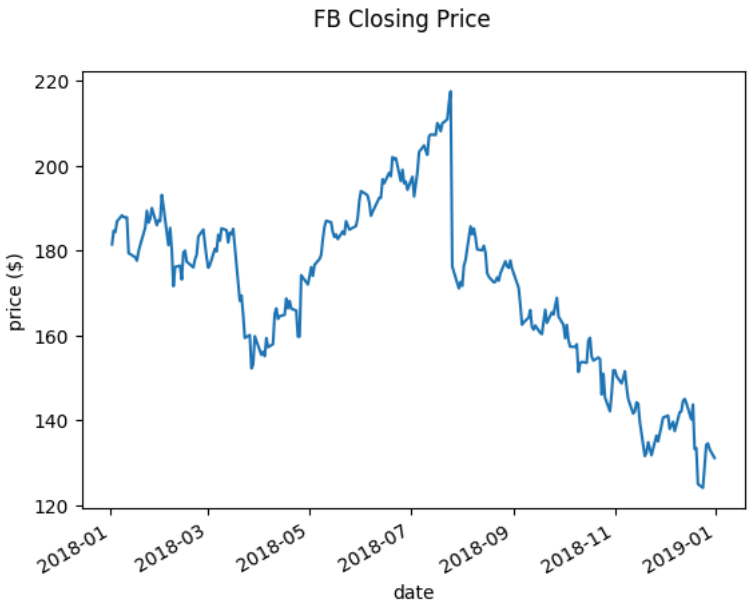
```
    ],
    row='magType',
    col='parsed_place'
)
g = g.map(plt.hist, 'mag')
```



9.5 Formatting Plots

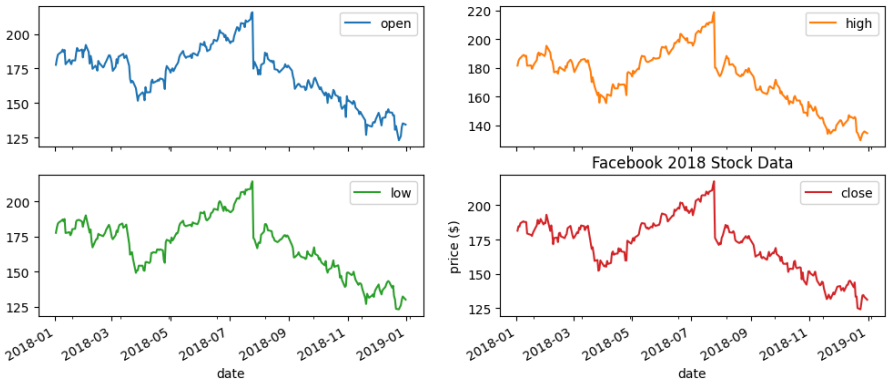
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
fb = pd.read_csv(
    'data/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)


# titles and axis labels

fb.close.plot()
plt.suptitle('FB Closing Price')
plt.xlabel('date')
plt.ylabel('price ($)')
```

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



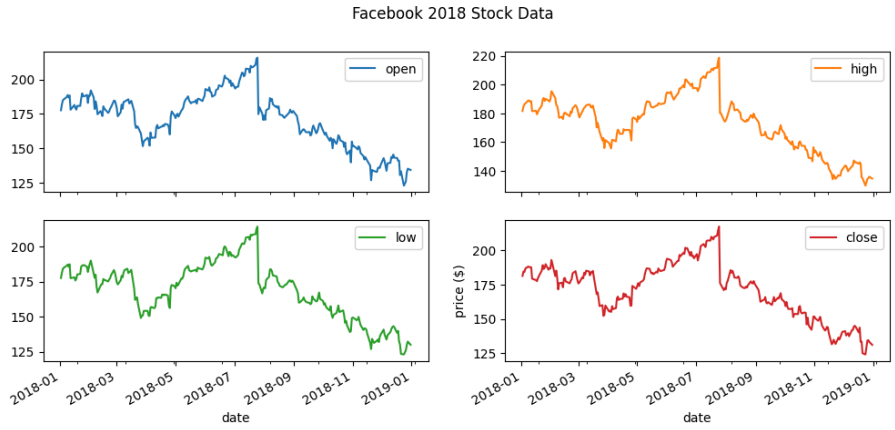

```
fb.iloc[:,:4].plot(subplots=True, layout=(2, 2), figsize=(12, 5))
plt.title('Facebook 2018 Stock Data')
plt.xlabel('date')
plt.ylabel('price ($)')
```

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



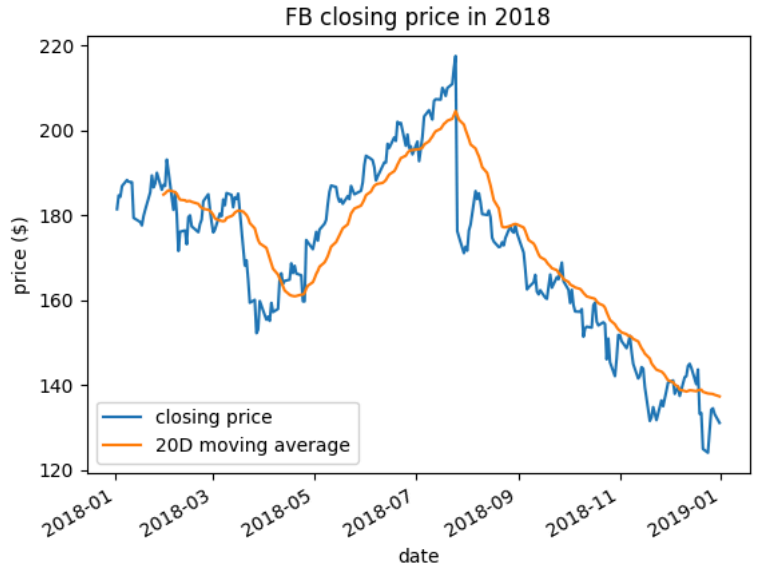

```
fb.iloc[:,:4].plot(subplots=True, layout=(2, 2), figsize=(12, 5))
plt.suptitle('Facebook 2018 Stock Data')
plt.xlabel('date')
plt.ylabel('price ($)')
```

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



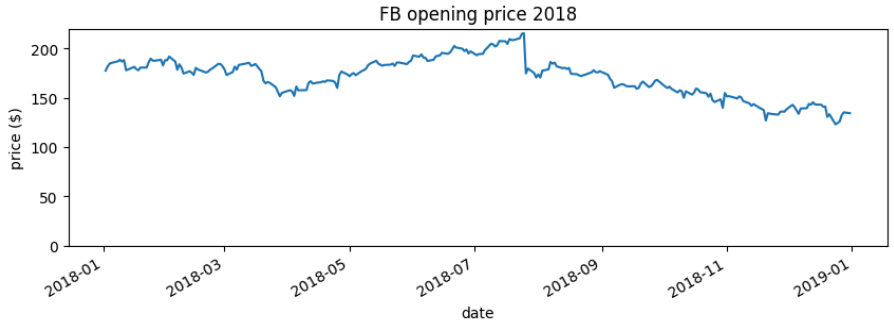

```
# legends

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

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



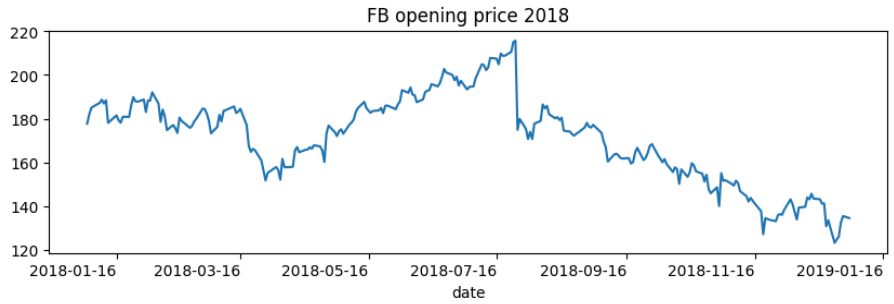

```
# specify axis limits

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

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




```
# format axis ticks

import calendar
fb.open.plot(figsize=(10, 3), rot=0, title='FB opening price 2018')
locs, labels = plt.xticks()
plt.xticks(locs + 15 , calendar.month_name[1::2])
plt.ylabel('price ($)')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-92-db216ab18dec> in <cell line: 6>()
      4 fb.open.plot(figsize=(10, 3), rot=0, title='FB opening price 2018')
      5 locs, labels = plt.xticks()
----> 6 plt.xticks(locs + 15 , calendar.month_name[1::2])
      7 plt.ylabel('price ($)')

                                  3 frames
/usr/local/lib/python3.10/dist-packages/matplotlib/axis.py in set_ticklabels(self,
labels, minor, fontdict, **kwargs)
   1967             # remove all tick labels, so only error for > 0 labels
   1968             if len(locator.locs) != len(labels) and len(labels) != 0:
-> 1969                 raise ValueError(
   1970                     "The number of FixedLocator locations"
   1971                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (7), usually from a call to set_ticks,
does not match the number of labels (6).
```

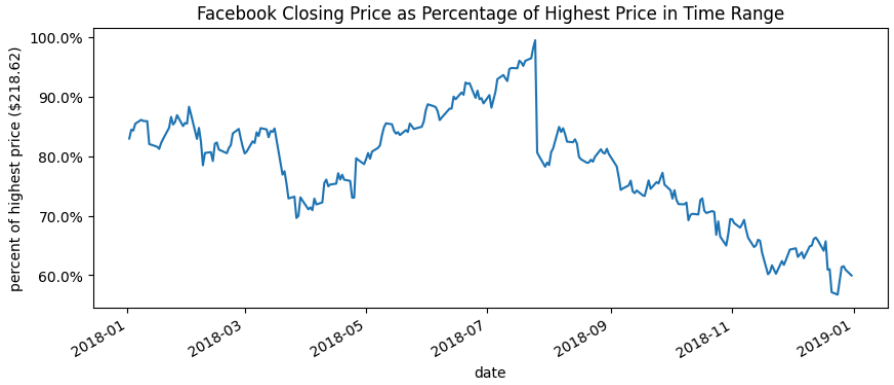


Next steps: Explain error

```
# using percent formatter

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 np.linspace(0.6, 1, num=5)
]) # show round percentages only (60%, 80%, etc.)
ax.set_ylabel(f'percent of highest price (${fb.high.max()})')
```

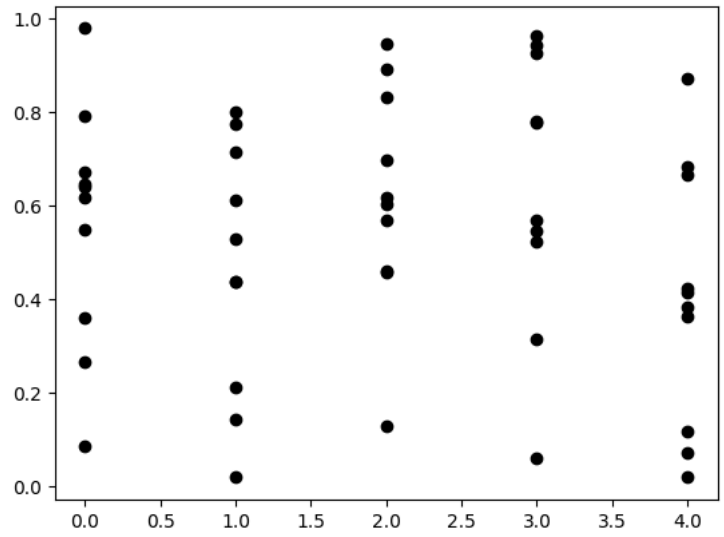
```
ax.set_ylabel('percent of highest price ($218.62)')
```

```
Text(0, 0.5, 'percent of highest price ($218.62)')
```

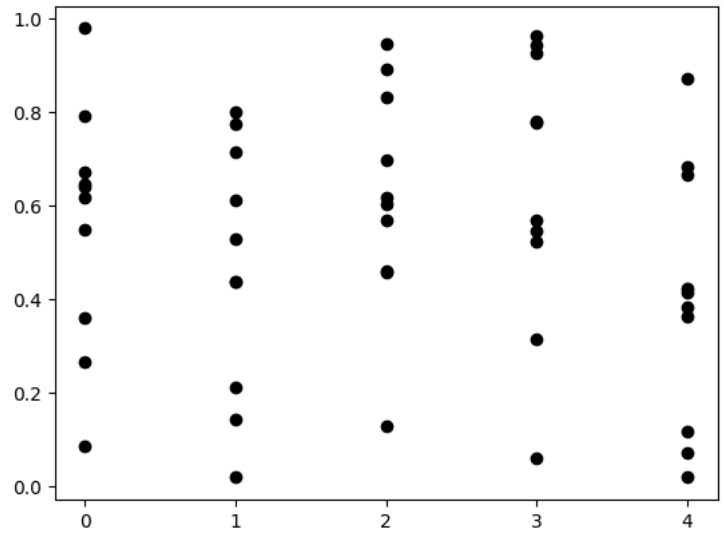



```
# using multiple locator

fig, ax = plt.subplots(1, 1)
np.random.seed(0)
ax.plot(np.tile(np.arange(0, 5), 10), np.random.rand(50), 'ko')
```

```
[<matplotlib.lines.Line2D at 0x7da40208f460>]
```



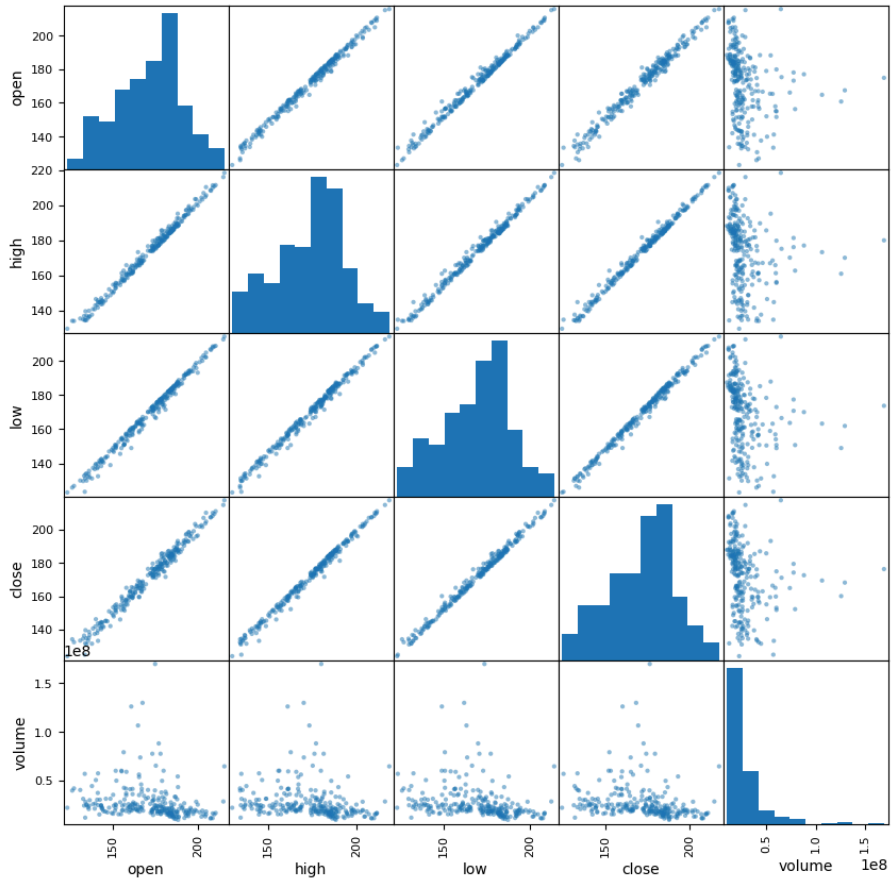

```
fig, ax = plt.subplots(1, 1)
np.random.seed(0)
ax.plot(np.tile(np.arange(0, 5), 10), np.random.rand(50), 'ko')
ax.get_xaxis().set_major_locator(
    ticker.MultipleLocator(base=1)
)
```



9.6 Customizing Visualizations

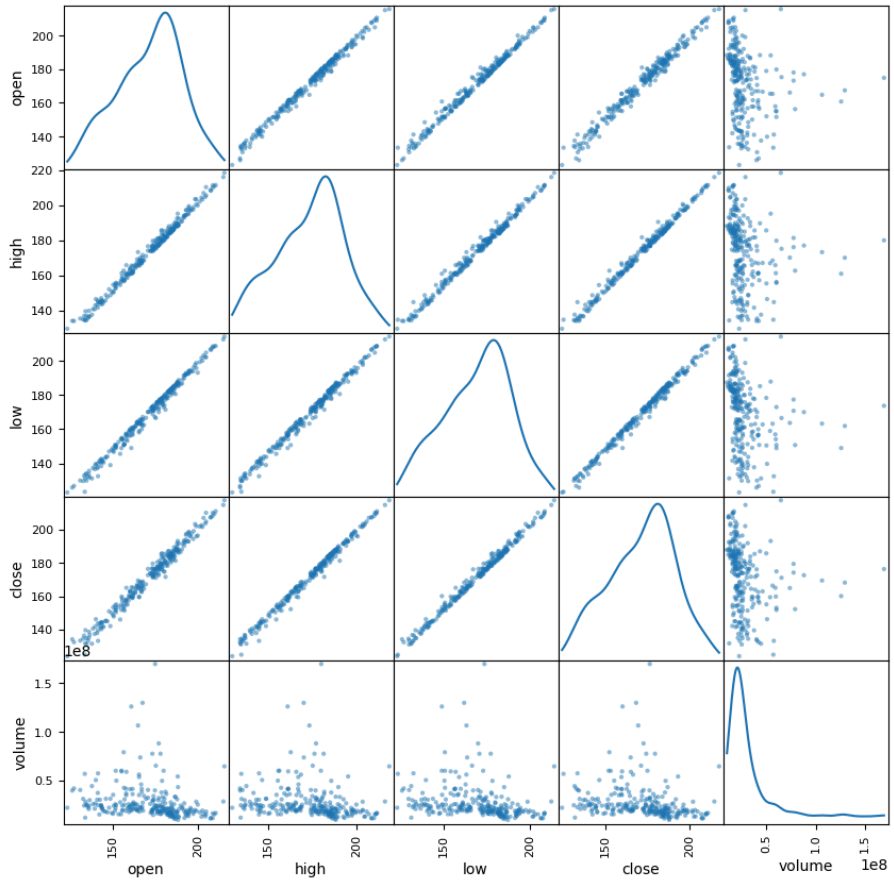
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
fb = pd.read_csv(
    'data/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)


# scatter matrix

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)
```



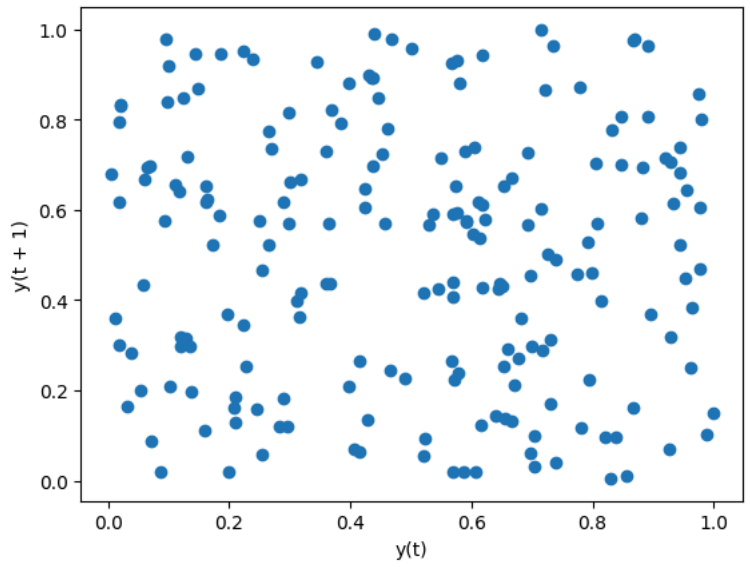

```
scatter_matrix(fb, figsize=(10, 10), diagonal='kde')
```

```
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)
```



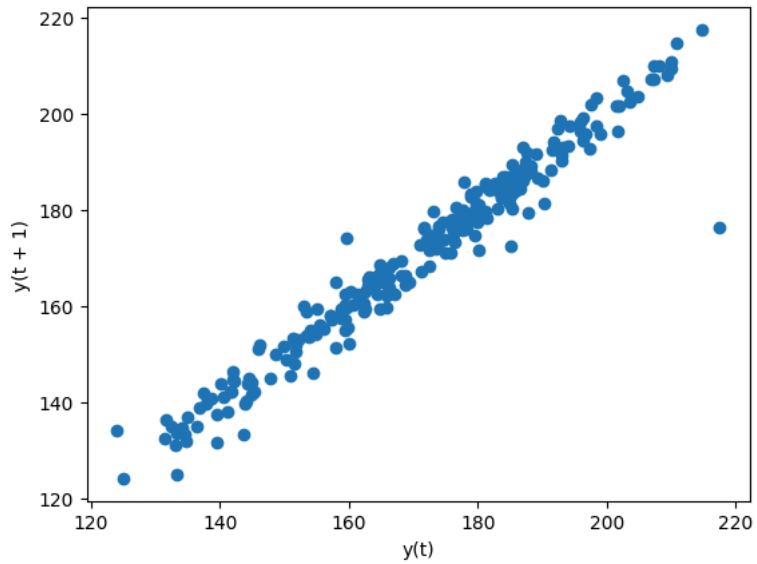

```
# lag plot

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)'>
```



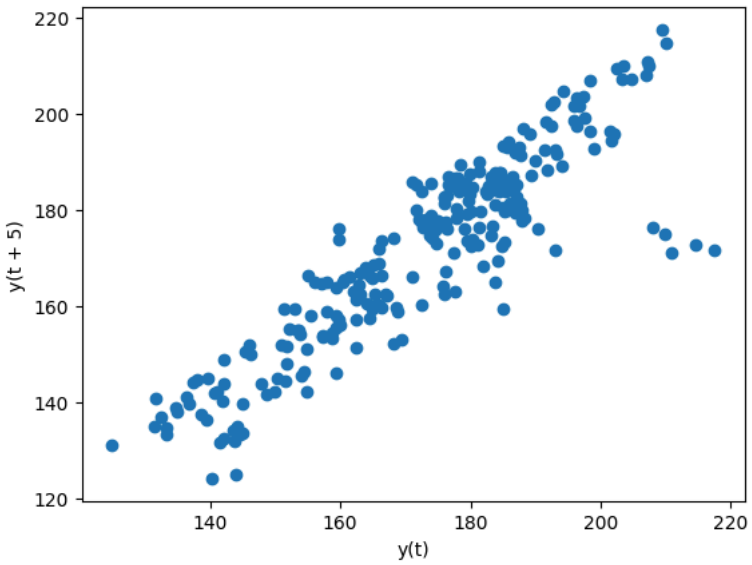

```
lag_plot(fb.close)
```

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



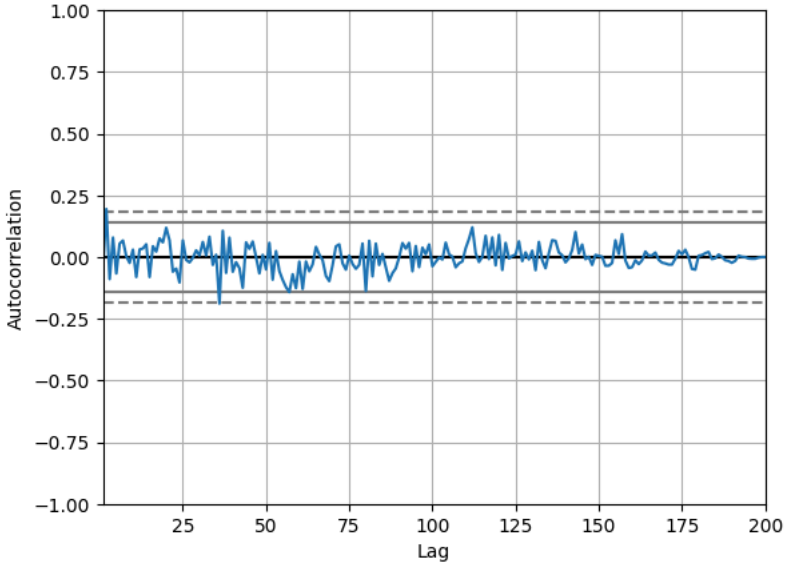

```
lag_plot(fb.close, lag=5)
```

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



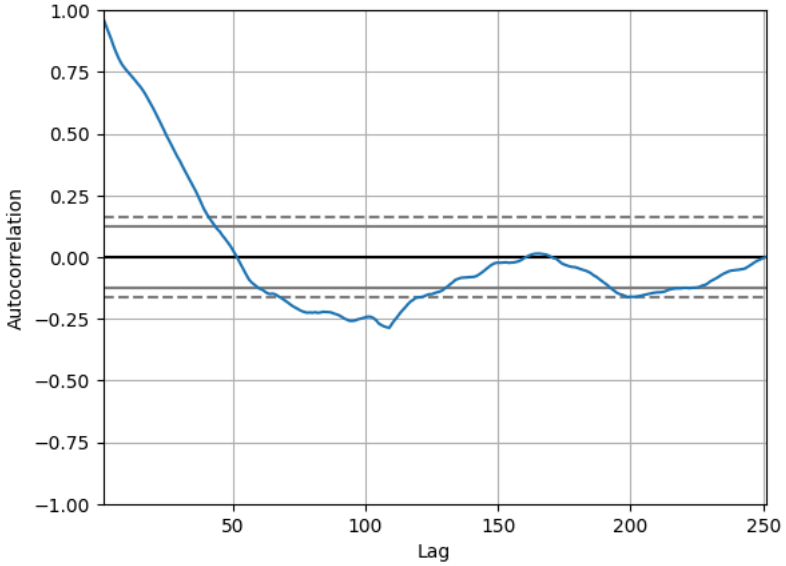

```
# autocorrelation plots

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'>
```

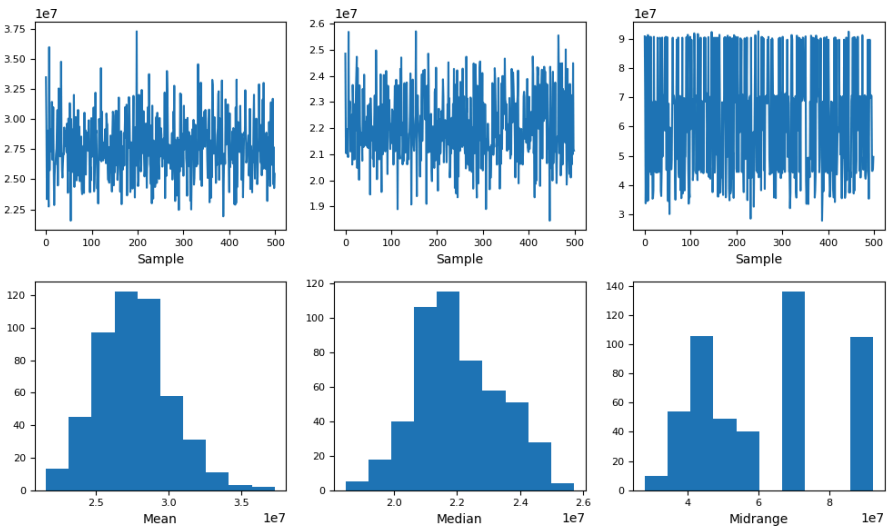



```
autocorrelation_plot(fb.close)
```

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




```
# bootstrap plot

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



Supplementary Activity:

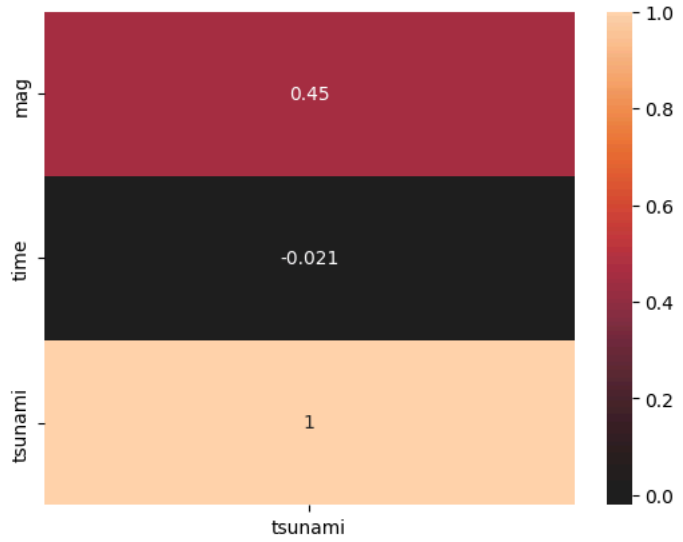
Using the CSV files provided and what we have learned so far in this module complete the following exercises:

1. Using seaborn, create a heatmap to visualize the correlation coefficients between earthquake magnitude and whether there was a tsunami with the magType of mb.
2. Create a box plot of Facebook volume traded and closing prices, and draw reference lines for the bounds of a Tukey fence with a multiplier of 1.5. The bounds will be at Q1 - 1.5 * IQR and Q3 + 1.5 * IQR. Be sure to use the quantile() method on the data to make this easier. (Pick whichever orientation you prefer for the plot, but make sure to use subplots.)
3. Fill in the area between the bounds in the plot from exercise #2.

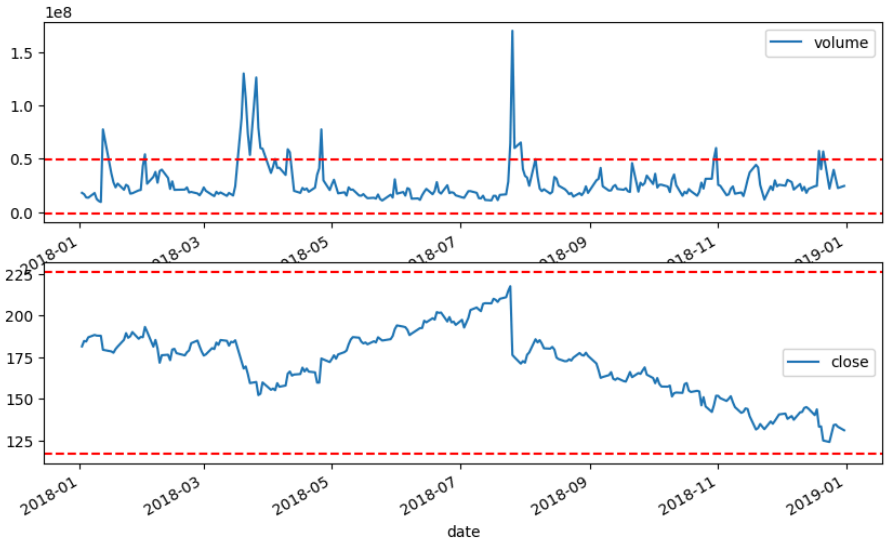
4. Use axvspan() to shade a rectangle from '2018-07-25' to '2018-07-31', which marks the large decline in Facebook price on a line plot of the closing price.
5. Using the Facebook stock price data, annotate the following three events on a line plot of the closing price:
    - Disappointing user growth announced after close on July 25, 2018
    - Cambridge Analytica story breaks on March 19, 2018 (when it affected the market)
    - FTC launches investigation on March 20, 2018
6. Modify the reg_resid_plots() function to use a matplotlib colormap instead of cycling between two colors. Remember, for this use case, we should pick a qualitative colormap or make our own.

```
# using seaborn, create a heatmap to visualize the correlation coefficients between earthquake magnitude and whether there was a tsunami wit

numerical_quakes = quakes.query('magType == "mb"').select_dtypes(include='number')

sns.heatmap(
    numerical_quakes.corr()['tsunami'].to_frame(),
    annot=True, center=0
)
```

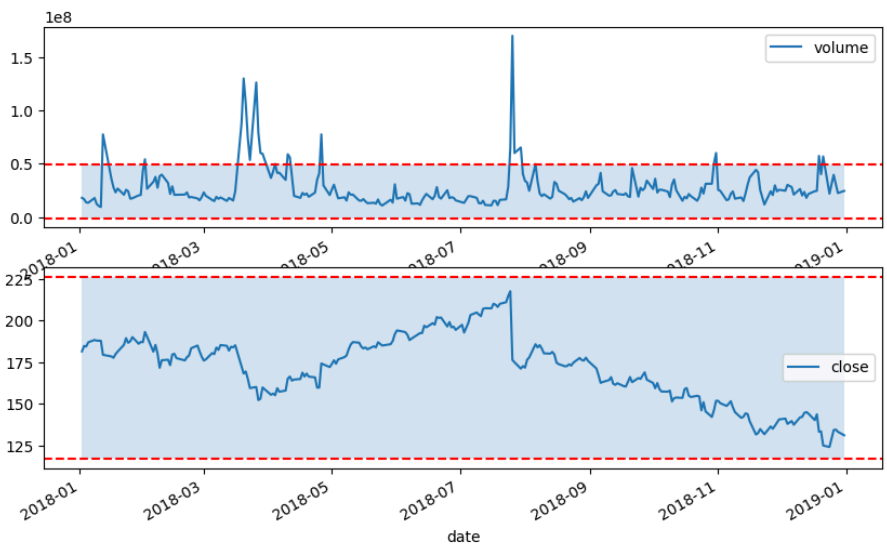
```
<Axes: >
```



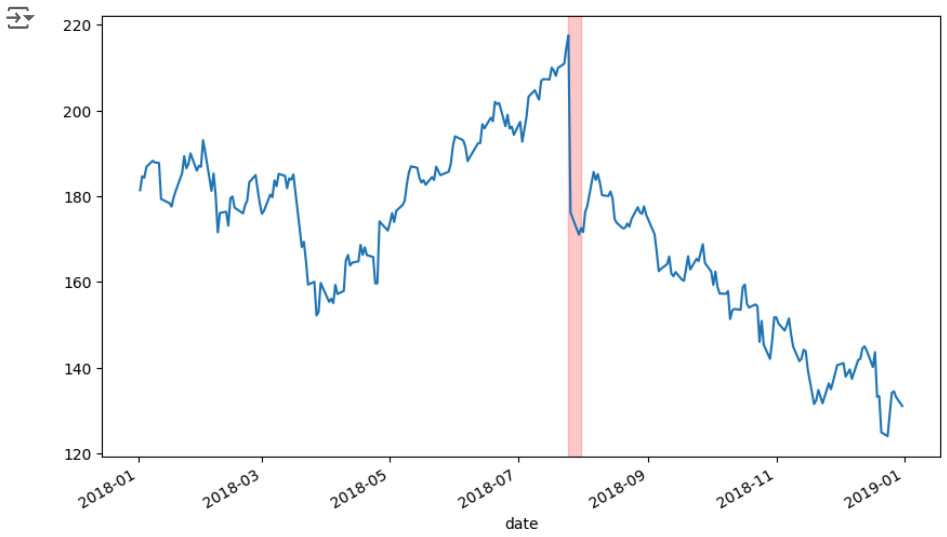

```
# create a box plot of Facebook volume traded and closing prices, and draw reference lines for the bounds of a Tukey fence with a multiplier

q1_volume = fb.volume.quantile(0.25)
q3_volume = fb.volume.quantile(0.75)
iqr_volume = q3_volume - q1_volume
lower_bound_volume = q1_volume - 1.5 * iqr_volume
upper_bound_volume = q3_volume + 1.5 * iqr_volume

q1_close = fb.close.quantile(0.25)
q3_close = fb.close.quantile(0.75)
iqr_close = q3_close - q1_close
lower_bound_close = q1_close - 1.5 * iqr_close
upper_bound_close = q3_close + 1.5 * iqr_close

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 6))

fb.plot(y='volume', ax=ax1)
ax1.axhline(y=lower_bound_volume, color='red', linestyle='--')
ax1.axhline(y=upper_bound_volume, color='red', linestyle='--')

fb.plot(y='close', ax=ax2)
ax2.axhline(y=lower_bound_close, color='red', linestyle='--')
ax2.axhline(y=upper_bound_close, color='red', linestyle='--')

plt.show()
```



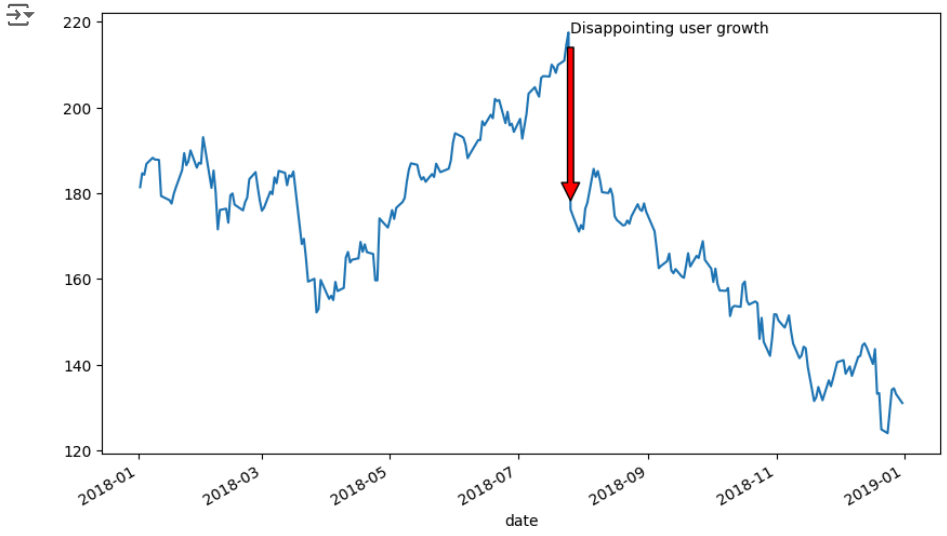

```
# fill in the area between the bounds in the plot from exercise #2.

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 6))

fb.plot(y='volume', ax=ax1)
ax1.axhline(y=lower_bound_volume, color='red', linestyle='--')
ax1.axhline(y=upper_bound_volume, color='red', linestyle='--')
ax1.fill_between(
    fb.index, lower_bound_volume, upper_bound_volume, alpha=0.2
)

fb.plot(y='close', ax=ax2)
ax2.axhline(y=lower_bound_close, color='red', linestyle='--')
ax2.axhline(y=upper_bound_close, color='red', linestyle='--')
ax2.fill_between(
    fb.index, lower_bound_close, upper_bound_close, alpha=0.2
)

plt.show()
```



```
# use axvspan() to shade a rectangle from '2018-07-25' to '2018-07-31', which marks the large decline in Facebook price on a line plot of th

fig, ax = plt.subplots(figsize=(10, 6))
fb.close.plot(ax=ax)
ax.axvspan(
    '2018-07-25', '2018-07-31', color='red', alpha=0.2
)
plt.show()
```

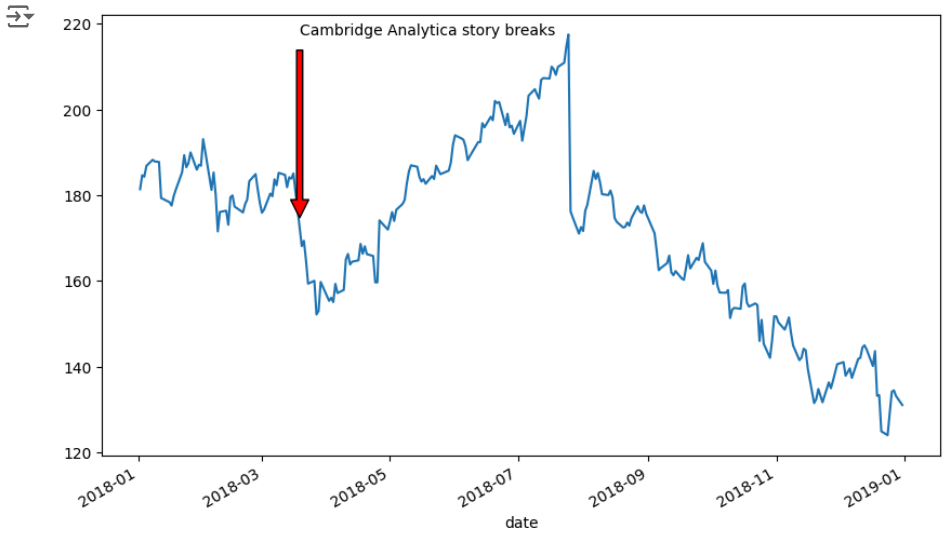



```
# disappointing user growth announced after close on July 25, 2018

fig, ax = plt.subplots(figsize=(10, 6))
fb.close.plot(ax=ax)
ax.annotate(
    'Disappointing user growth', xy=('2018-07-26', fb.close['2018-07-26']),
    xytext=('2018-07-26', fb.close.max()),
    arrowprops=dict(facecolor='red', shrink=0.05)
)
plt.show()
```

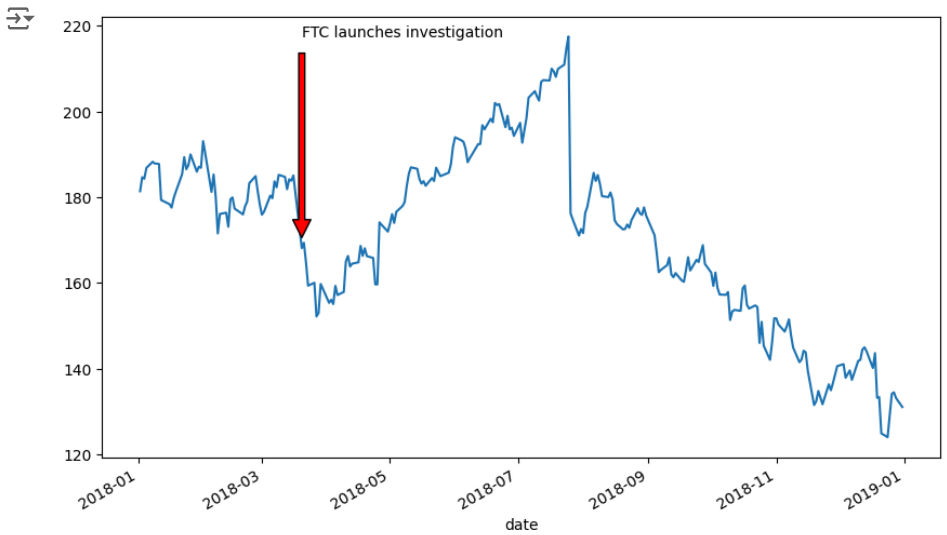



```
# Cambridge Analytica story breaks on March 19, 2018 (when it affected the market)
```

```
# Cambridge Analytica story breaks on March 19, 2018 (when it affected the market)

fig, ax = plt.subplots(figsize=(10, 6))
fb.close.plot(ax=ax)
ax.annotate(
    'Cambridge Analytica story breaks', xy=('2018-03-19', fb.close['2018-03-19']),
    xytext=('2018-03-19', fb.close.max()),
    arrowprops=dict(facecolor='red', shrink=0.05)
)
plt.show()
```




```
# FTC launches investigation on March 20, 2018

fig, ax = plt.subplots(figsize=(10, 6))
fb.close.plot(ax=ax)
ax.annotate(
    'FTC launches investigation', xy=('2018-03-20', fb.close['2018-03-20']),
    xytext=('2018-03-20', fb.close.max()),
    arrowprops=dict(facecolor='red', shrink=0.05)
)
plt.show()
```



```
# modify the reg_resid_plots() function to use a matplotlib colormap instead of cycling between two colors. Remember, for this use case, we

import matplotlib.pyplot as plt
import seaborn as sns

def reg_resid_plots(data, x, y, colormap='coolwarm'):
  """
  Plots the regression line and residuals for a given dataset.

  Args:
    data: A Pandas DataFrame containing the data.
    x: The name of the independent variable.
    y: The name of the dependent variable.
    colormap: The name of the colormap to use for the regression line.
  """
  sns.regplot(x=x, y=y, color=colormap, data=data)
  sns.residplot(x=x, y=y, data=data, color='gray')
  plt.show()
```

Conclusion

To conclude, similar to HOA 9.1, these activities explore more about plots and I have learned some new different plots to visualize data. Moreover, it uses seaborn to customize the visualization of the given datasets.