Hands-on Activity 9.2 Customized Visualizations using Seaborn

Submitted by: Dela Cruz, Eugene D.G.

Submitted to: Engr. Roman Richard

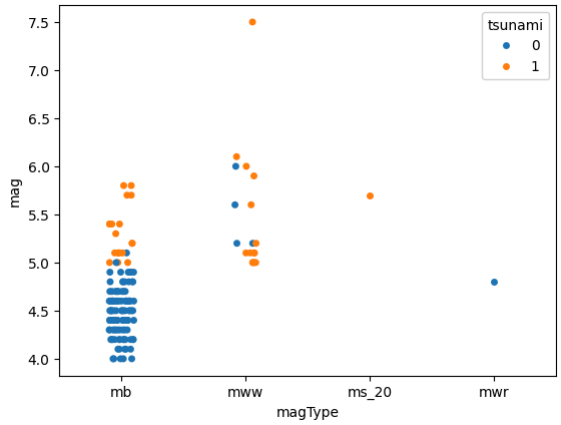
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
fb = pd.read_csv(
 '/content/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
quakes = pd.read_csv('/content/earthquakes.csv')
```

```
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

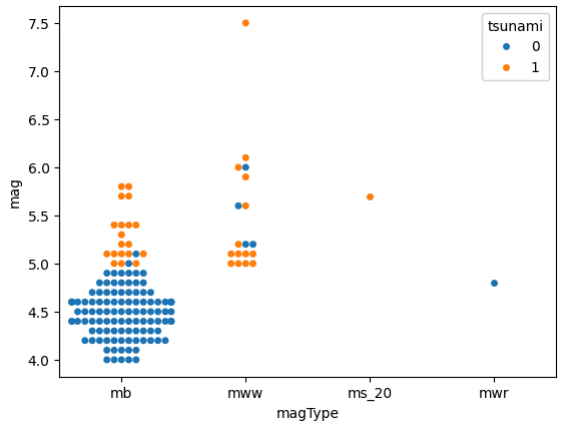
```
sns.stripplot(
 x='magType',
 y='mag',
 hue='tsunami',
 data=quakes.query('parsed_place == "Indonesia"')
)
```

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



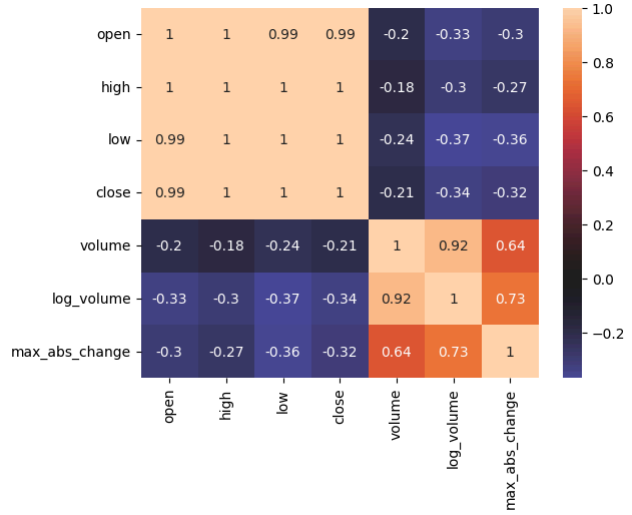

```
sns.swarmplot(
 x='magType',
 y='mag',
 hue='tsunami',
 data=quakes.query('parsed_place == "Indonesia"')
)
```

```
<Axes: xlabel='magType', ylabel='mag'>
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3398: UserWarning: 10.2% of the points cannot be plac
  warnings.warn(msg, UserWarning)
```



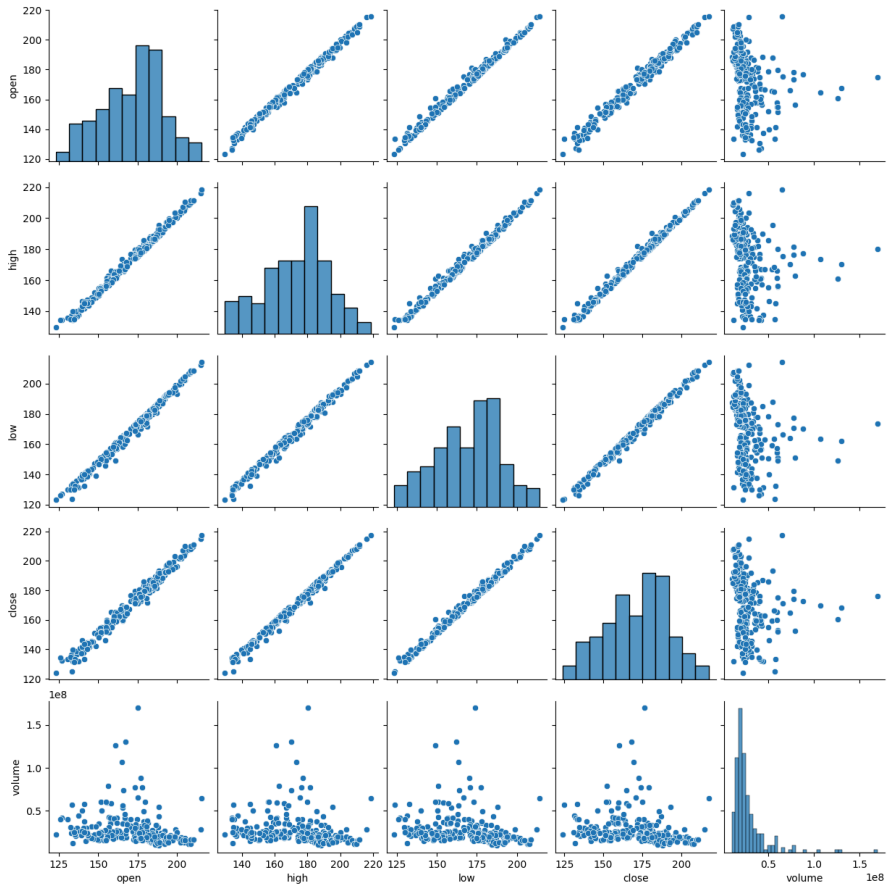

```
sns.heatmap(
 fb.sort_index().assign(
 log_volume=np.log(fb.volume),
 max_abs_change=fb.high - fb.low
).corr(),
 annot=True, center=0
)
```

```
<Axes: >
```



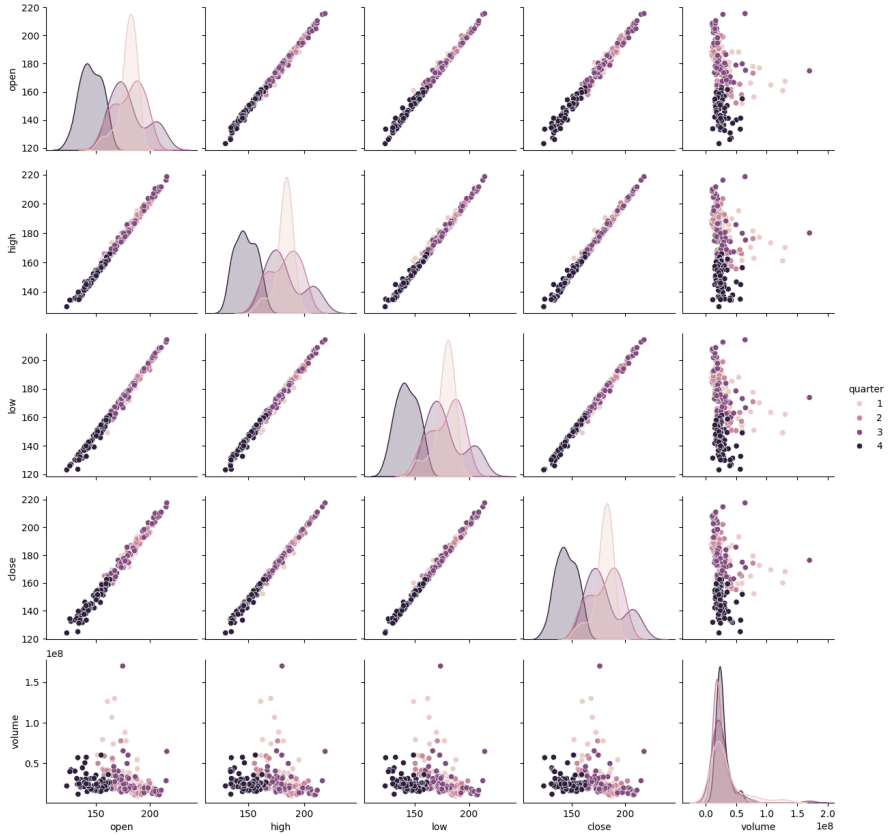

```
sns.pairplot(fb)
```

```
<seaborn.axisgrid.PairGrid at 0x78a741afbe80>
```



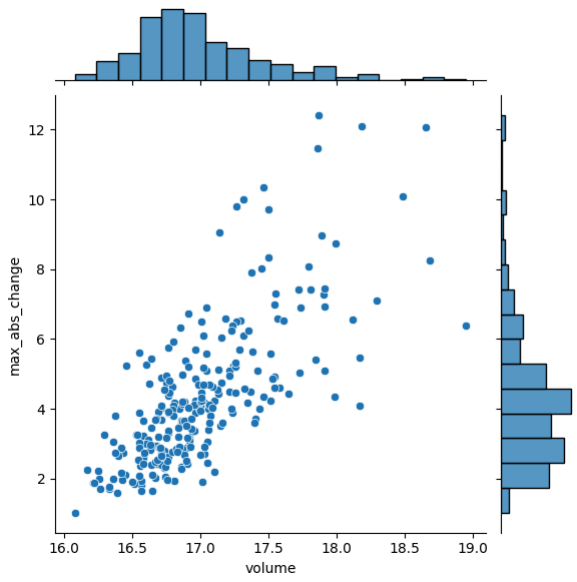

```
sns.pairplot(
 fb.assign(quarter=lambda x: x.index.quarter),
 diag_kind='kde',
 hue='quarter'
)
```

```
<seaborn.axisgrid.PairGrid at 0x78a74186fc70>
```



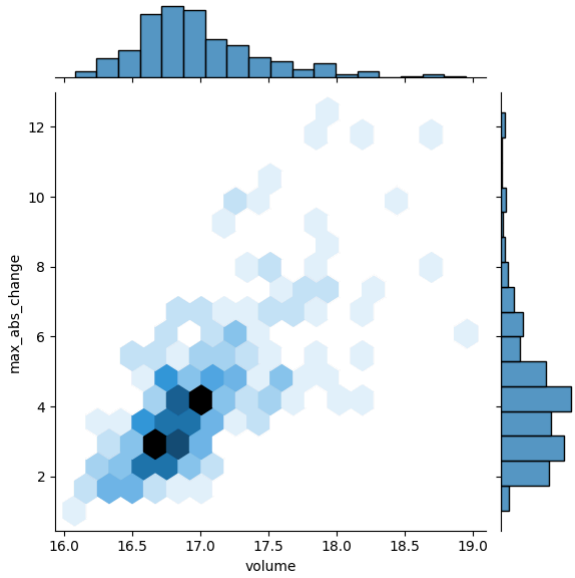

```
sns.jointplot(
 x='volume',
 y='max_abs_change',
 data=fb.assign(
 volume=np.log(fb.volume),
 max_abs_change=fb.high - fb.low
 )
)
```

```
<seaborn.axisgrid.JointGrid at 0x78a740d752a0>
```



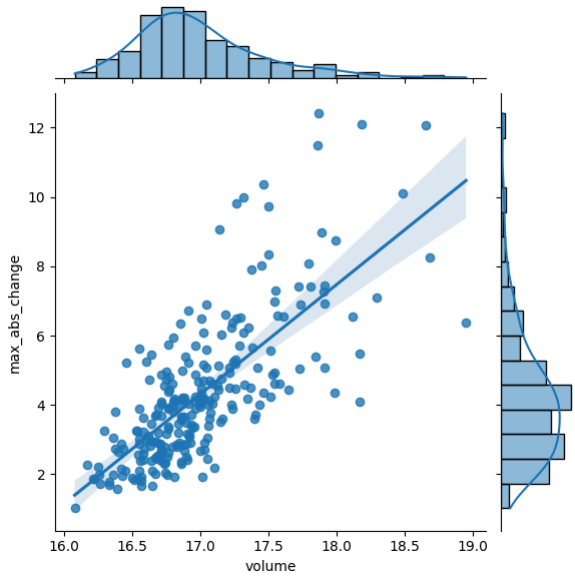
```
sns.jointplot(
 x='volume',
 y='max_abs_change',
 kind='hex',
 data=fb.assign(
 volume=np.log(fb.volume),
 max_abs_change=fb.high - fb.low
 )
)
```

```
<seaborn.axisgrid.JointGrid at 0x78a74932f6d0>
```



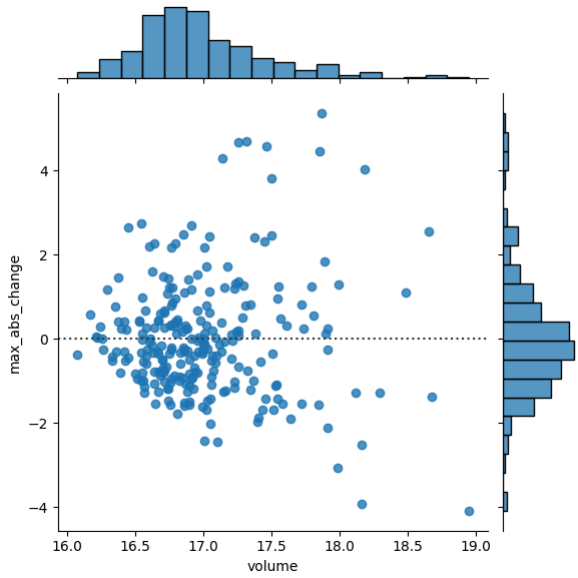

```
sns.jointplot(
 x='volume',
 y='max_abs_change',
 kind='reg',
 data=fb.assign(
 volume=np.log(fb.volume),
 max_abs_change=fb.high - fb.low
 )
)
```

```
<seaborn.axisgrid.JointGrid at 0x78a740d77040>
```



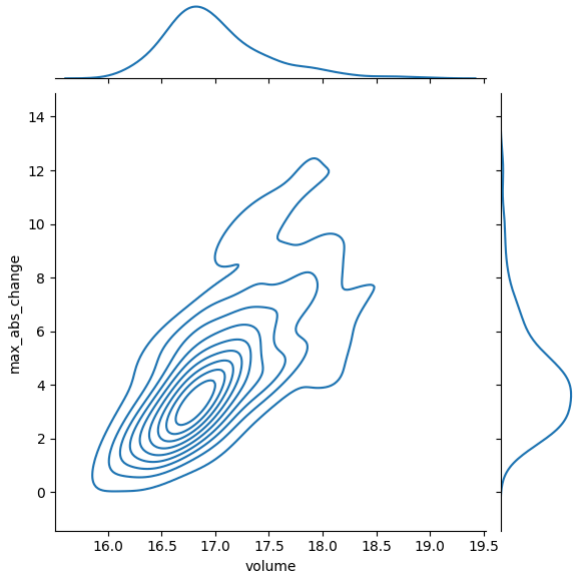

```
sns.jointplot(
 x='volume',
 y='max_abs_change',
 kind='resid',
 data=fb.assign(
 volume=np.log(fb.volume),
 max_abs_change=fb.high - fb.low
 )
)
```

```
<seaborn.axisgrid.JointGrid at 0x78a73fcc4e20>
```




```
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 0x78a73fb96cb0>
```




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

```
import itertools
```

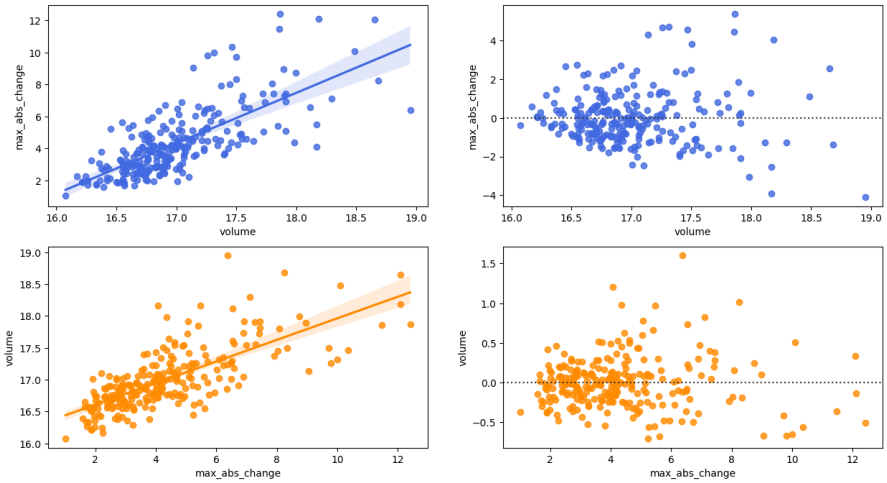
```
iterator = itertools.repeat("I'm an iterator", 1)
for i in iterator:
 print(f'-->{i}')
print('This printed once because the iterator has been exhausted')
for i in iterator:
 print(f'-->{i}')
```

```
-->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}')
```

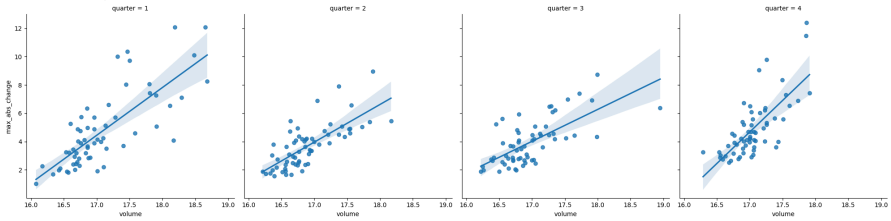
```
-->I'm an iterable
This prints again because it's an iterable:
-->I'm an iterable
```

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



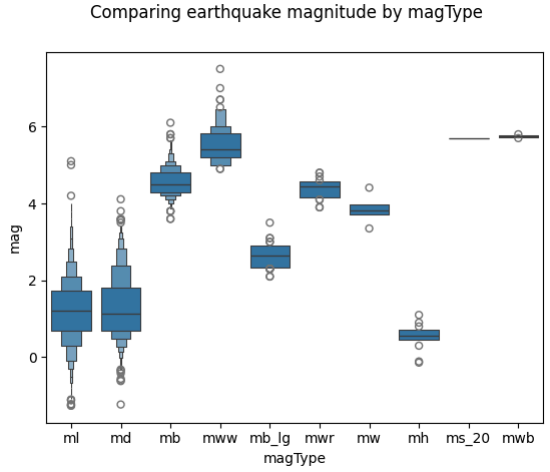

```
sns.lmplot(
 x='volume',
 y='max_abs_change',
 data=fb.assign(
 volume=np.log(fb.volume),
 max_abs_change=fb.high - fb.low,
 quarter=lambda x: x.index.quarter
 ),
 col='quarter'
)
```

```
<seaborn.axisgrid.FacetGrid at 0x78a73f714f10>
```




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

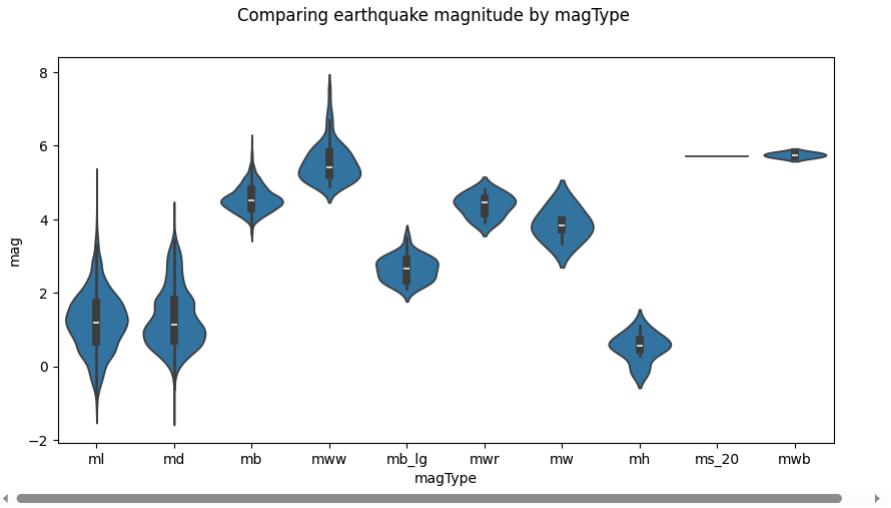
```
Text(0.5, 0.98, 'Comparing earthquake magnitude by magType')
```



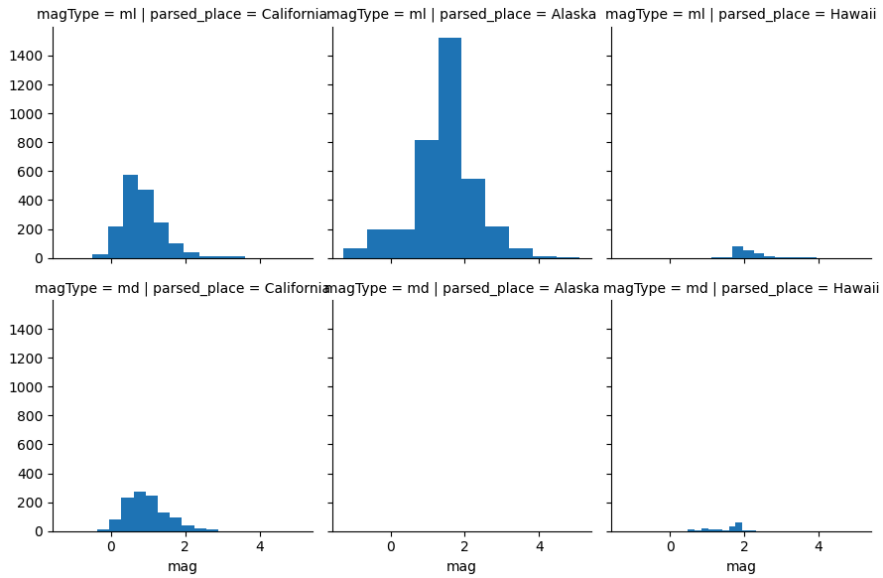
```
fig, axes = plt.subplots(figsize=(10, 5))
sns.violinplot(
 x='magType', y='mag', data=quakes[['magType', 'mag']],
 ax=axes, scale='width' # all violins have same width
)
plt.suptitle('Comparing earthquake magnitude by magType')
```

```
<ipython-input-49-bd4e17637bf6>:2: FutureWarning:

The `scale` parameter has been renamed and will be removed in v0.15.0. Pass `density_norm='width'` for the same effe
  sns.violinplot(
Text(0.5, 0.98, 'Comparing earthquake magnitude by magType')
```




```
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')
```

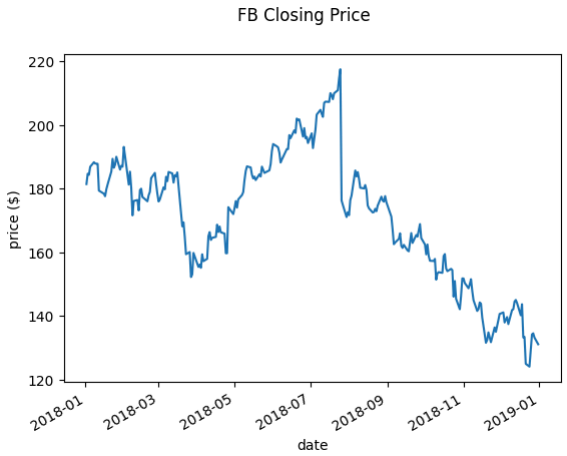


9.5 Formatting Plots

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

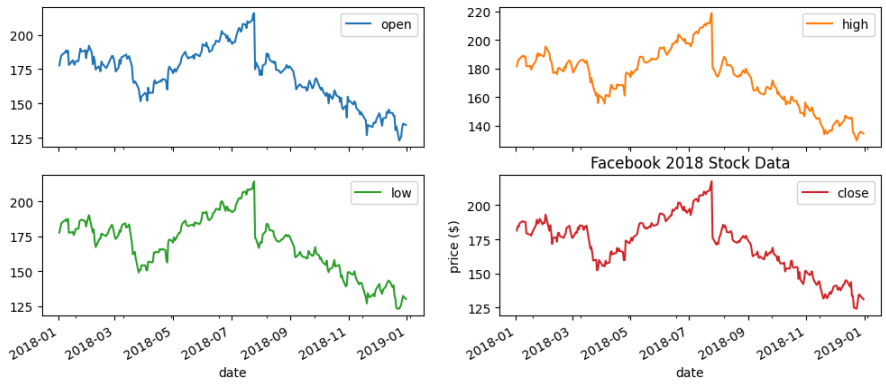
```
fb.close.plot()
plt.suptitle('FB Closing Price')
plt.xlabel('date')
plt.ylabel('price ($)')
```

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



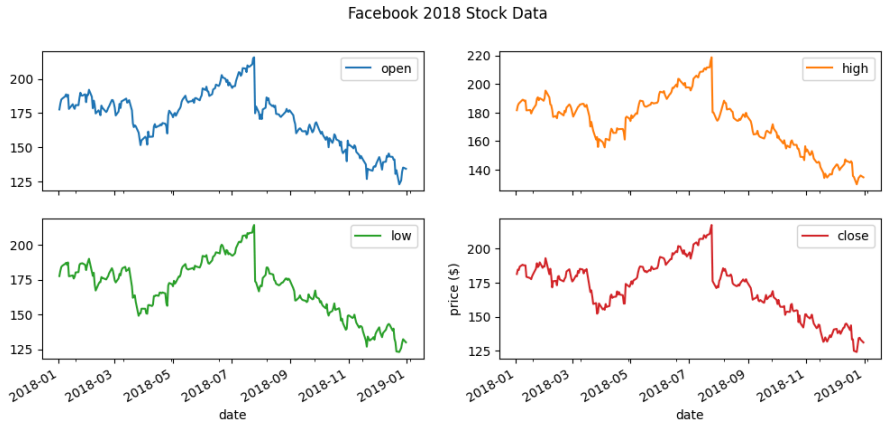

```
fb.iloc[:,:4].plot(subplots=True, layout=(2, 2), figsize=(12, 5))
plt.title('Facebook 2018 Stock Data')
plt.xlabel('date')
plt.ylabel('price ($)')
```

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



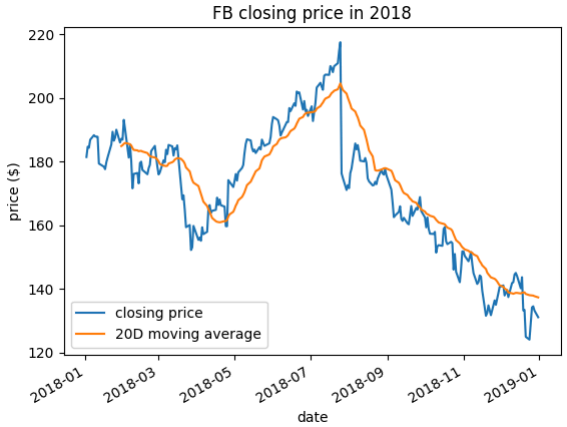

```
fb.iloc[:,:4].plot(subplots=True, layout=(2, 2), figsize=(12, 5))
plt.suptitle('Facebook 2018 Stock Data')
plt.xlabel('date')
plt.ylabel('price ($)')
```

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



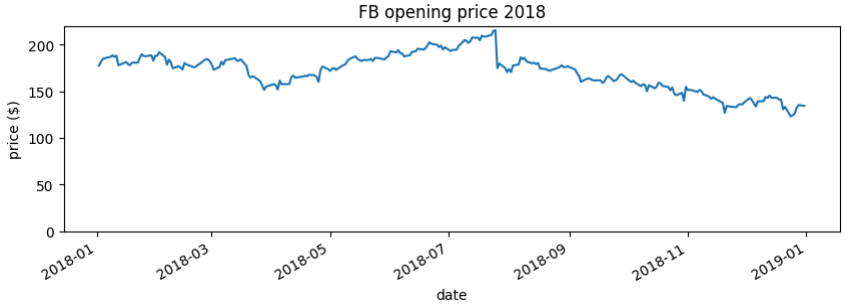

```
fb.assign(
 ma=lambda x: x.close.rolling(20).mean()
).plot(
 y=['close', 'ma'],
 title='FB closing price in 2018',
 label=['closing price', '20D moving average']
)
plt.legend(loc='lower left')
plt.ylabel('price ($)')
```

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




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

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




```
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 ($)')
```

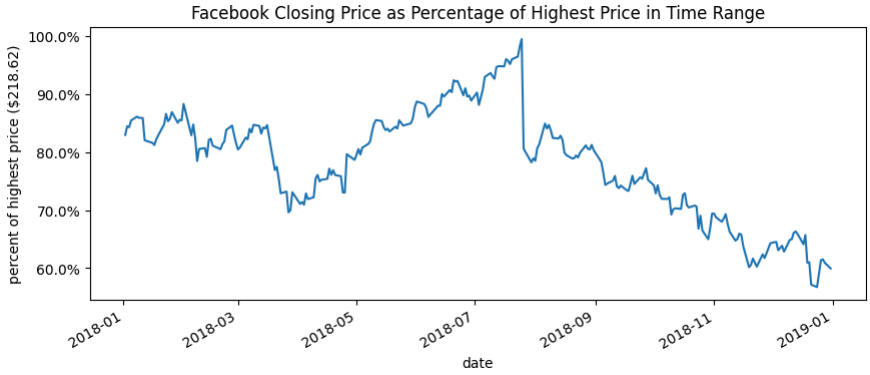
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-58-49f9a03c7ca6> in <cell line: 4>()
      2 fb.open.plot(figsize=(10, 3), rot=0, title='FB opening price 2018')
      3 locs, labels = plt.xticks()
----> 4 plt.xticks(locs + 15 , calendar.month_name[1::2])
      5 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).
```

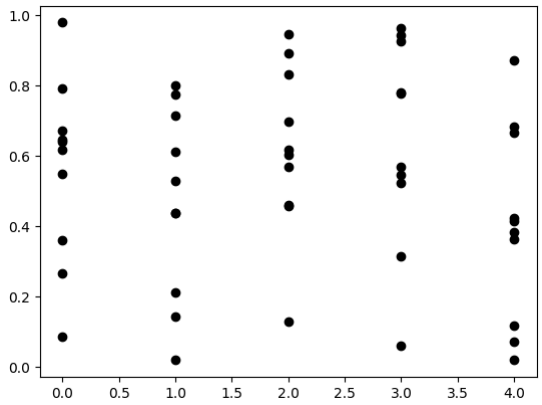

```
import matplotlib.ticker as ticker
ax = fb.close.plot(
 figsize=(10, 4),
 title='Facebook Closing Price as Percentage of Highest Price in Time Range'
)
ax.yaxis.set_major_formatter(
 ticker.PercentFormatter(xmax=fb.high.max())
)
ax.set_yticks([
 fb.high.max()*pct for pct in np.linspace(0.6, 1, num=5)
]) # show round percentages only (60%, 80%, etc.)
ax.set_ylabel(f'percent of highest price (${fb.high.max()})')
```

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

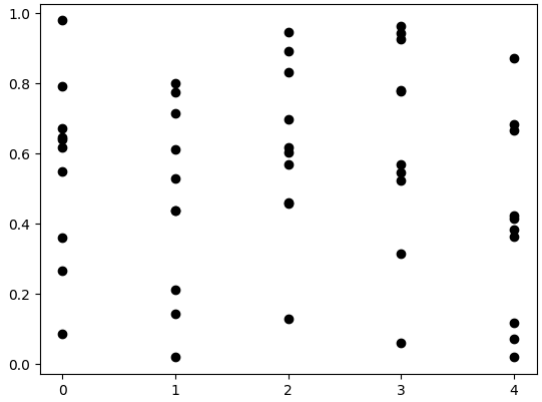



```
fig, ax = plt.subplots(1, 1)
np.random.seed(0)
ax.plot(np.tile(np.arange(0, 5), 10), np.random.rand(50), 'ko')
```

```
[<matplotlib.lines.Line2D at 0x78a73df759c0>]
```




```
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)
)
```

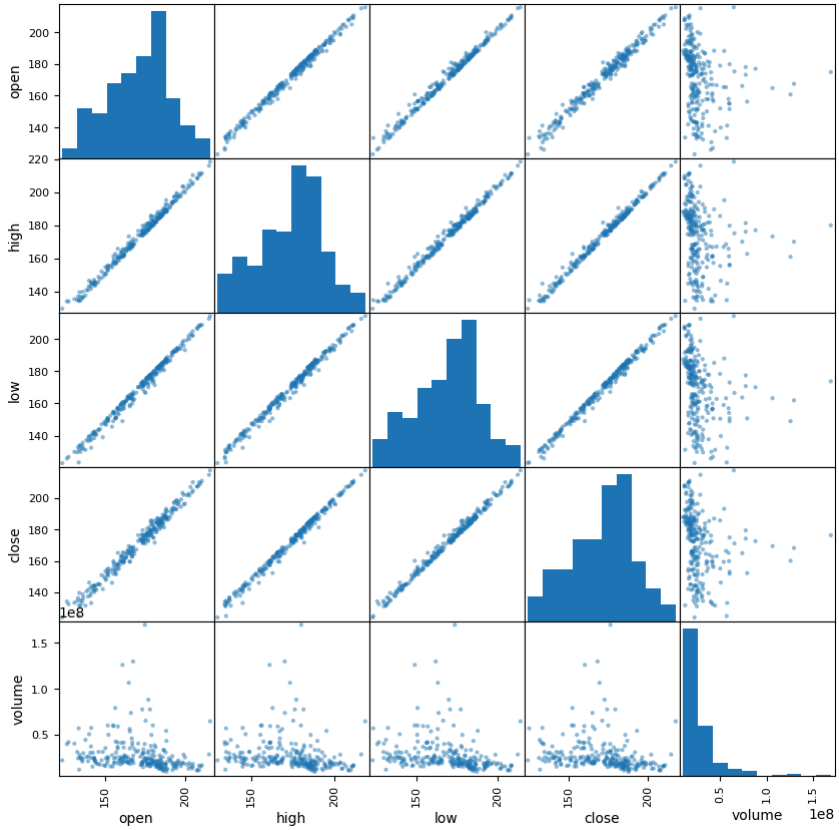


9.6 Customizing Visualizations

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

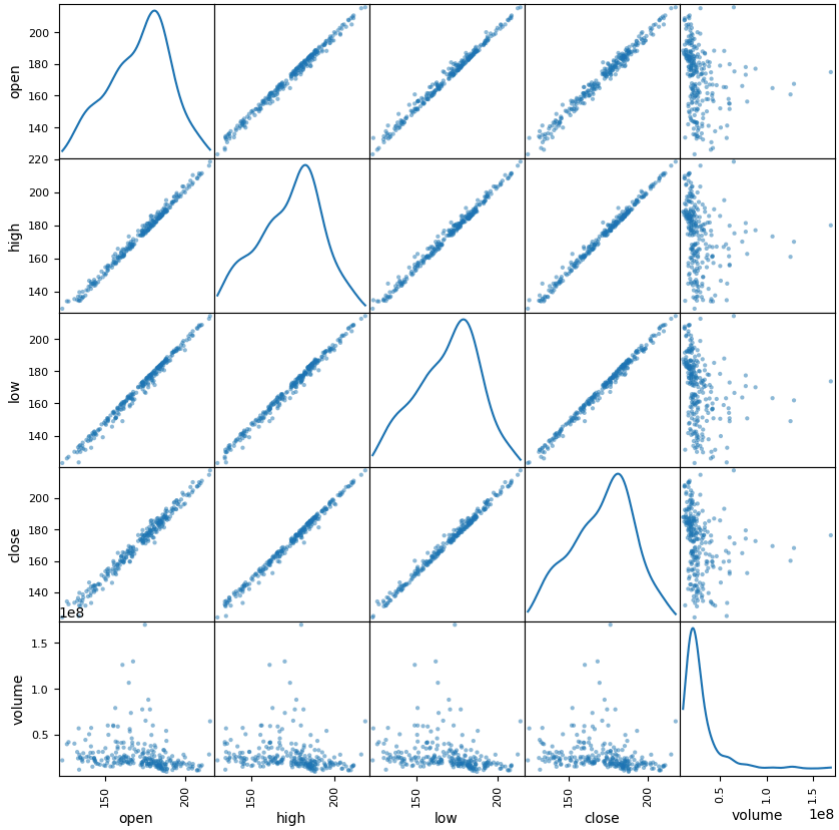
```
from pandas.plotting import scatter_matrix
scatter_matrix(fb, figsize=(10, 10))
```

```
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)
```



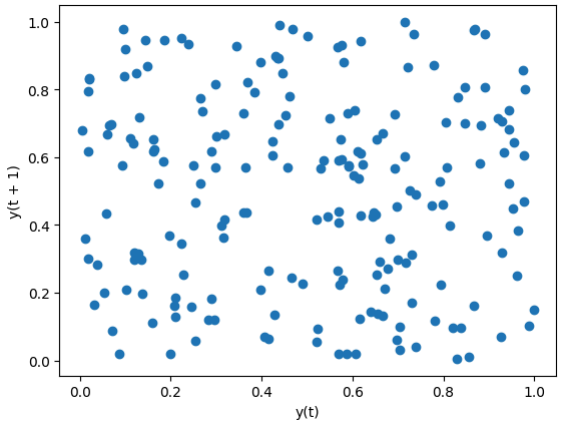

```
scatter_matrix(fb, figsize=(10, 10), diagonal='kde')
```

```
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)
```



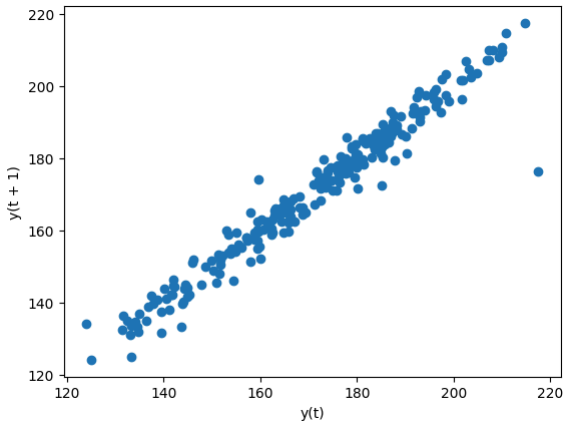

```
from pandas.plotting import lag_plot
np.random.seed(0) # make this repeatable
lag_plot(pd.Series(np.random.random(size=200)))
```

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



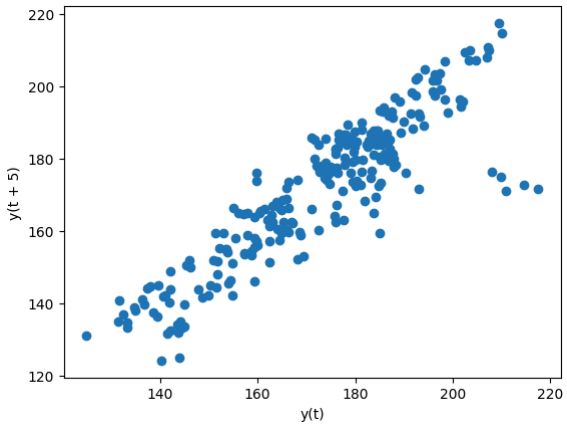

```
lag_plot(fb.close)
```

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



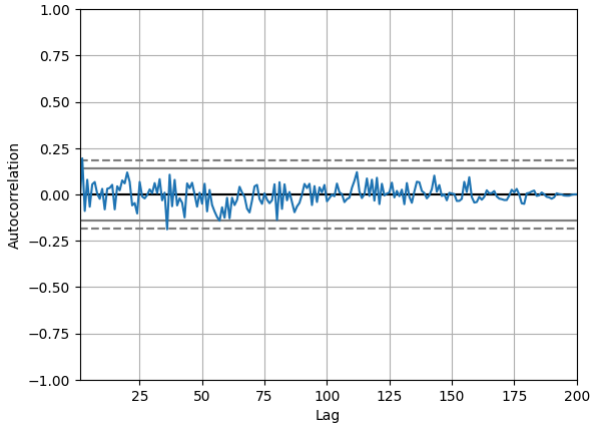

```
lag_plot(fb.close, lag=5)
```

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



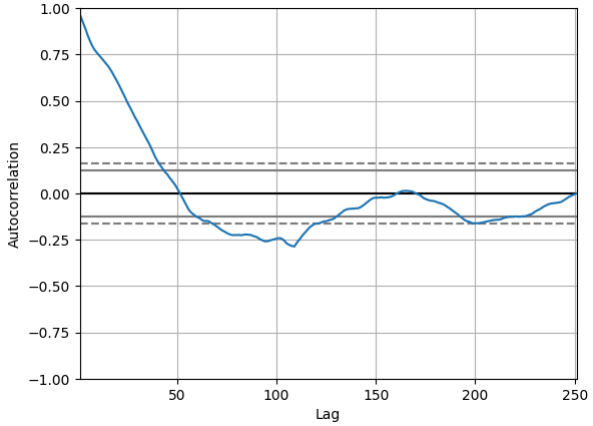

```
from pandas.plotting import autocorrelation_plot
np.random.seed(0) # make this repeatable
autocorrelation_plot(pd.Series(np.random.random(size=200)))
```

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

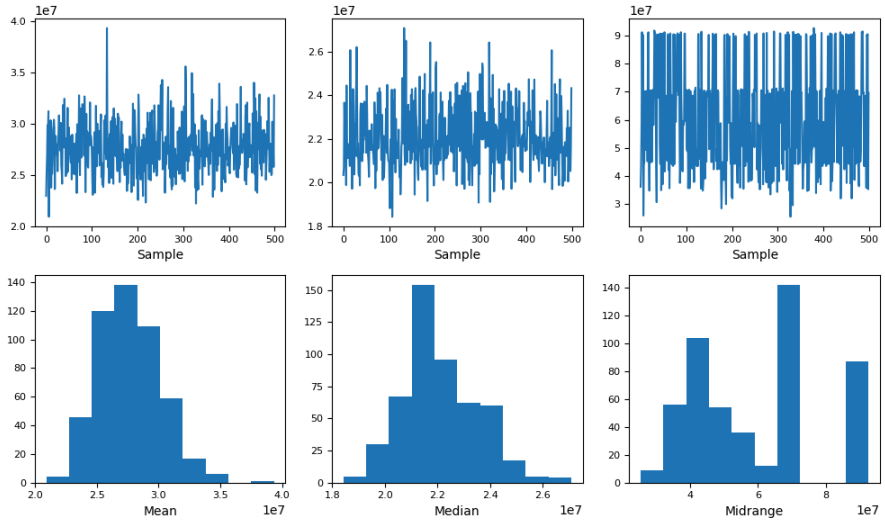



```
autocorrelation_plot(fb.close)
```

```
<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. 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.

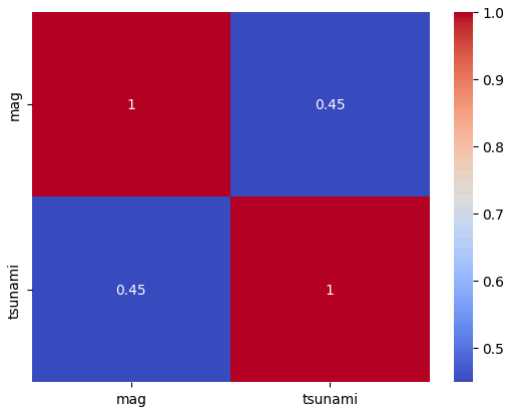
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import iqr

# Load data
fb_data = pd.read_csv("fb_stock_prices_2018.csv")
earthquake_data = pd.read_csv("earthquakes-1.csv")
```

1.

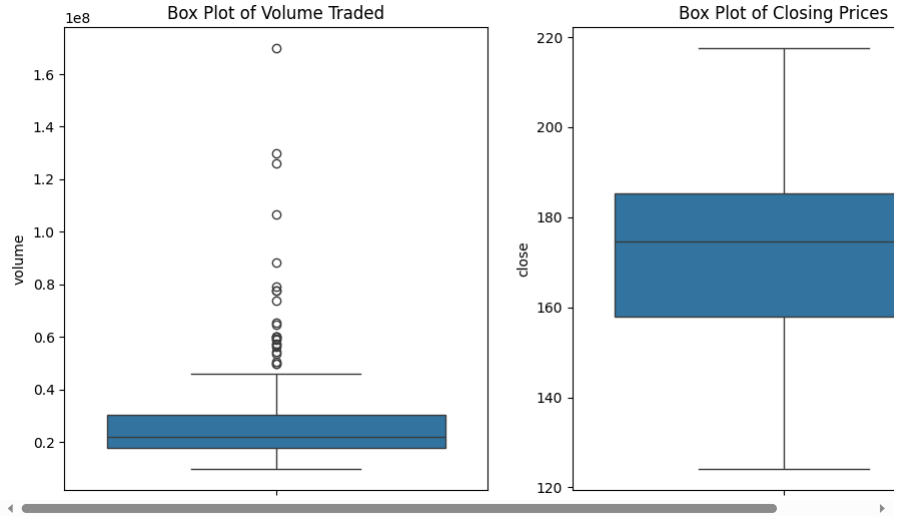
```
earthquake_data2= earthquake_data.query('magType == "mb"')[['mag','tsunami']]
sns.heatmap(earthquake_data2.corr(), cmap='coolwarm', # identify their correlation
            annot = True)
```

```
<Axes: >
```



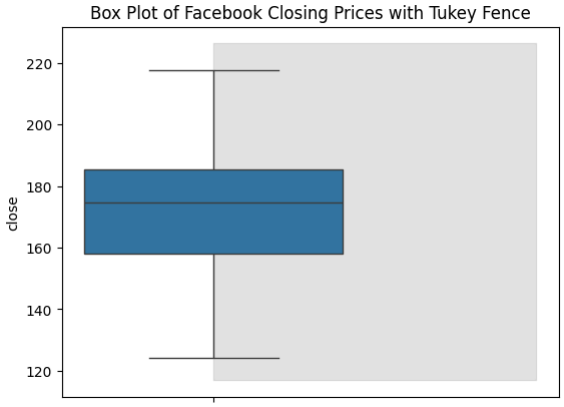
2.

```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
sns.boxplot(y=fb_data['volume'], ax=axes[0]) # Box plot for Volume Traded
axes[0].set_title('Box Plot of Volume Traded')
sns.boxplot(y=fb_data['close'], ax=axes[1]) # Box plot for Closing Prices
axes[1].set_title('Box Plot of Closing Prices')
plt.show()
```



3.

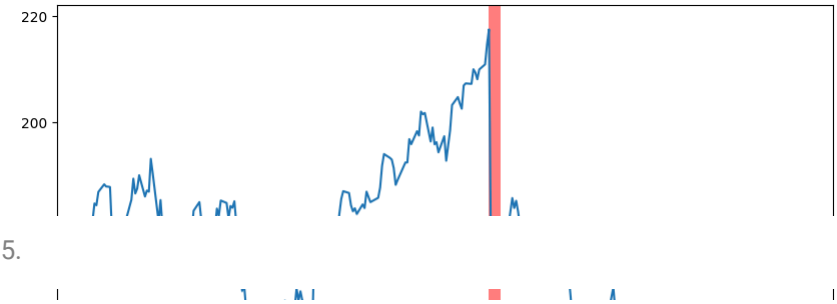
```
fig, ax = plt.subplots()
sns.boxplot(y=fb_data['close'], ax=ax)
q1 = fb_data['close'].quantile(0.25)
q3 = fb_data['close'].quantile(0.75)
iqr_val = iqr(fb_data['close'])
lower_bound = q1 - 1.5 * iqr_val # Fill area between bounds
upper_bound = q3 + 1.5 * iqr_val # Fill area between bounds
ax.fill_betweenx([lower_bound, upper_bound], 0, 1, alpha=0.2, color='grey')
plt.title('Box Plot of Facebook Closing Prices with Tukey Fence')
plt.show()
```



4.

```
fb_1 = pd.read_csv('/content/fb_stock_prices_2018.csv')
fb_1['date'] = pd.to_datetime(fb_1['date'])
fb_1.set_index('date', inplace = True) # set the index
start = '2018-07-25'
end = '2018-07-31'
fb_1.close.plot(figsize = (10,8)) # plotting close column
plt.axvspan(start, end, facecolor = 'red', alpha = 0.5)
```

```
<matplotlib.patches.Polygon at 0x78a73c670c70>
```



5.

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

fb_data = pd.read_csv("fb_stock_prices_2018.csv")
fb_data['date'] = pd.to_datetime(fb_data['date']) # Convert dates to datetime objects
fig, ax = plt.subplots() # Create the plot
ax.plot(fb_data['date'], fb_data['close'], label='Closing Price')

events = [
    ('Disappointing user growth', '2018-07-25', 170),
    ('Cambridge Analytica', '2018-03-19', 175),
    ('FTC investigation', '2018-03-20', 160)
]
```