Skill-9

February 4, 2022

1 RAHUL KUMAR

2 2000031900, AIDS-SEC05

3 Plotting with Pandas

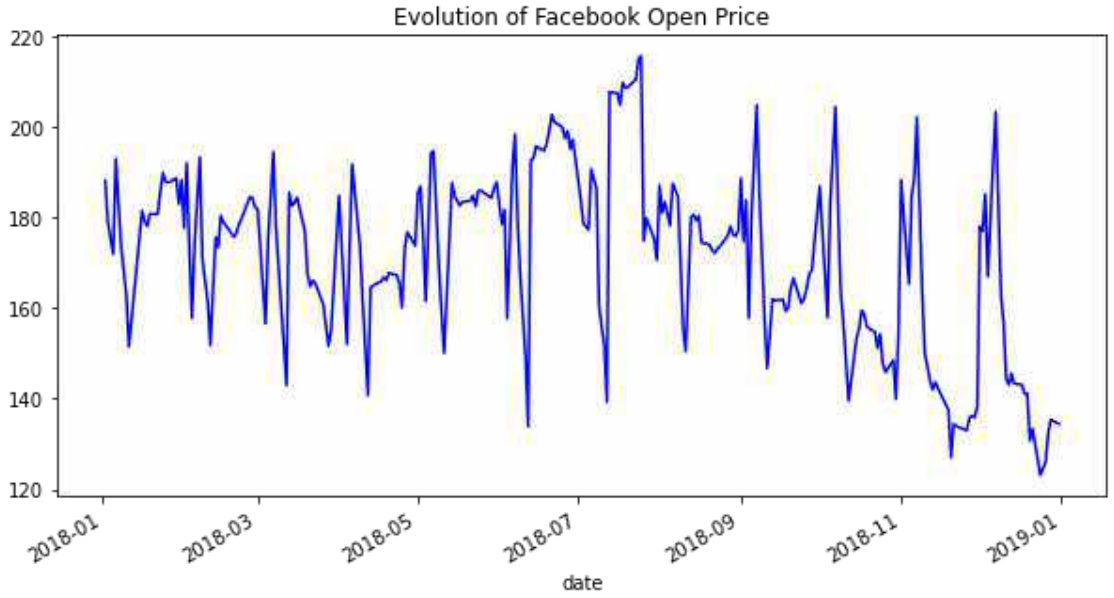
[1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

[3]:
```
fb = pd.read_csv(
    'fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
quakes = pd.read_csv('earthquakes.csv')
covid = pd.read_csv('covid19_cases.csv',encoding='unicode_escape').assign(
    date=lambda x: pd.to_datetime(x.dateRep, format='%d-%m-%Y')
).set_index('date').replace(
    'United_States_of_America', 'USA'
).sort_index()['2020-01-18':'2020-09-18']
```

4 Evolution Over Time

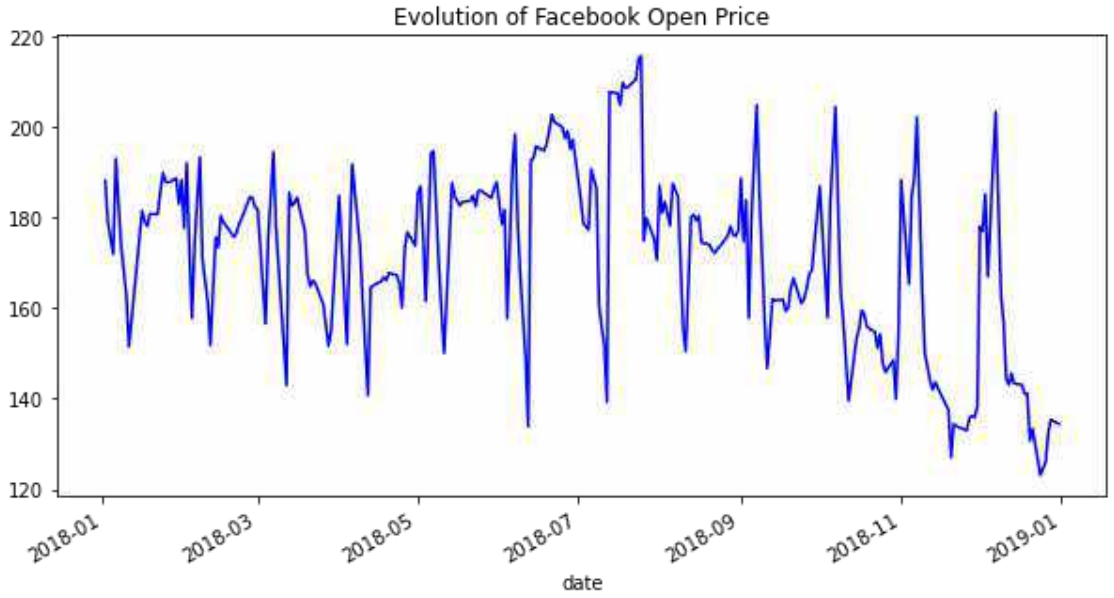
[4]:
```
fb.plot(
    kind='line',
    y='open',
    figsize=(10, 5),
    style='-b',
    legend=False,
    title='Evolution of Facebook Open Price '
)
```

[4]: <AxesSubplot:title={'center':'Evolution of Facebook Open Price'}, xlabel='date'>

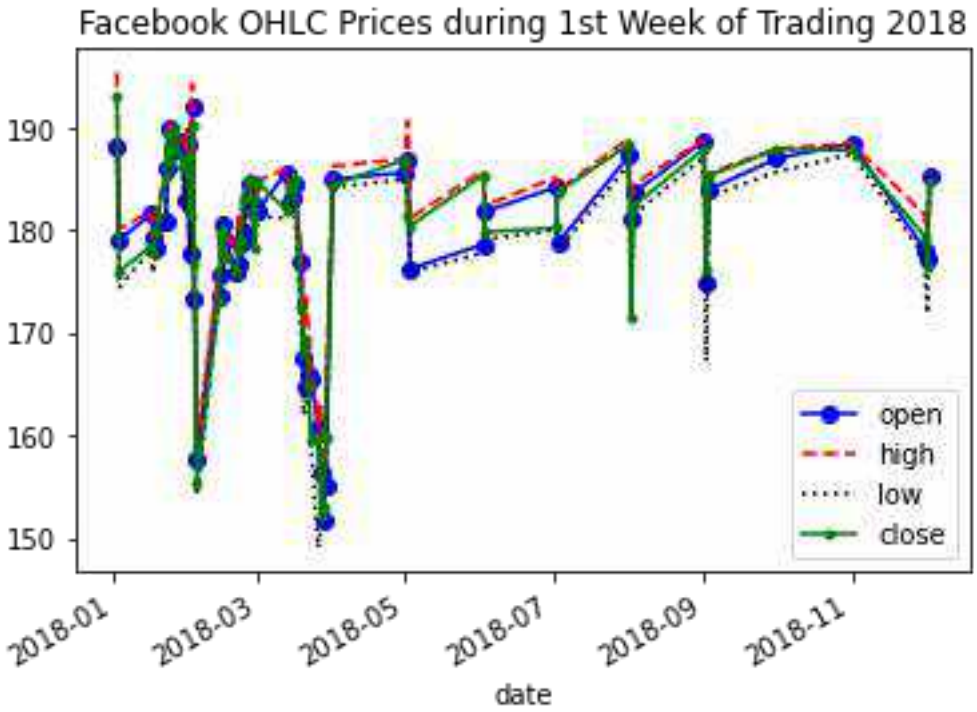


[5]:
```
fb.plot(
    kind='line',
    y='open',
    figsize=(10, 5),
    style='-b',
    legend=False,
    title='Evolution of Facebook Open Price '
)
```

[5]: <AxesSubplot:title={'center':'Evolution of Facebook Open Price'}, xlabel='date'>



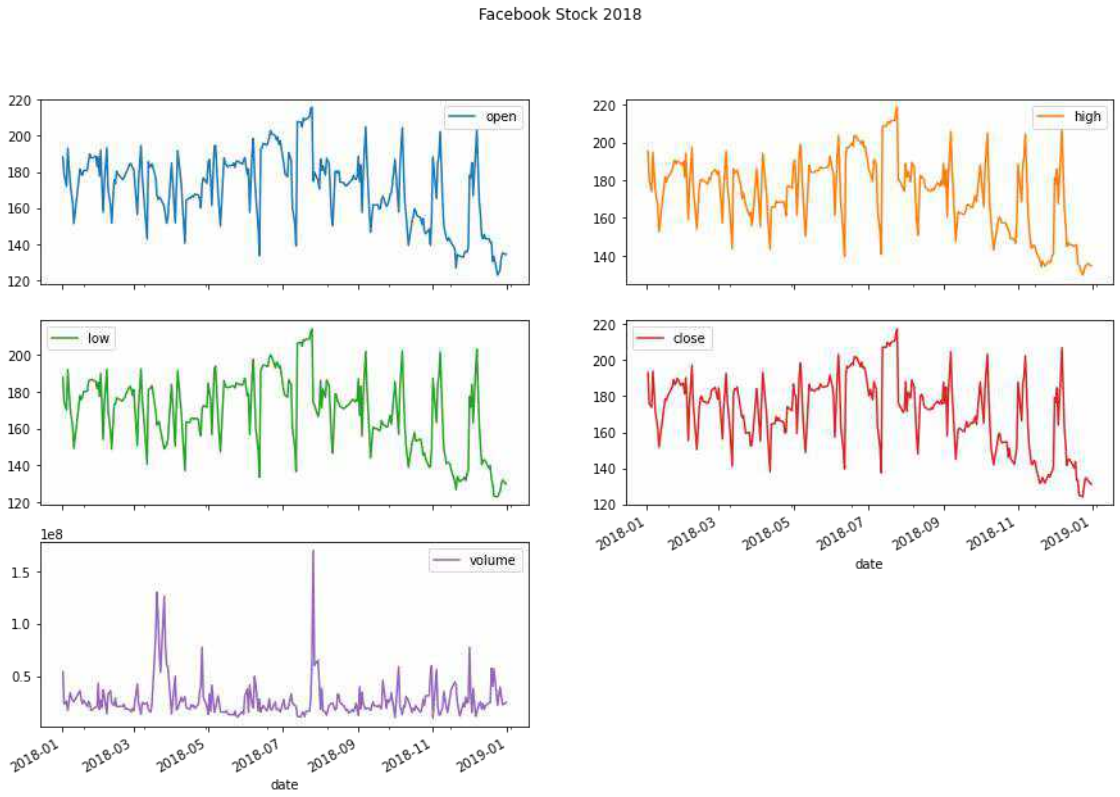
[6]:
```
fb.first('1W').plot(
    y=['open', 'high', 'low', 'close'],
    style=['o-b', '--r', ':k', '.-g'],
    title='Facebook OHLC Prices during 1st Week of Trading 2018 '
).autoscale()
```



5 Creating subplots

[7]:
```
fb.plot(
    kind='line',
    subplots=True,
    layout=(3, 2),
    figsize=(15, 10),
    title='Facebook Stock 2018'
)
```

[7]:
```
array([[<AxesSubplot:xlabel='date'>, <AxesSubplot:xlabel='date'>],
       [<AxesSubplot:xlabel='date'>, <AxesSubplot:xlabel='date'>],
       [<AxesSubplot:xlabel='date'>, <AxesSubplot:xlabel='date'>]],
      dtype=object)
```

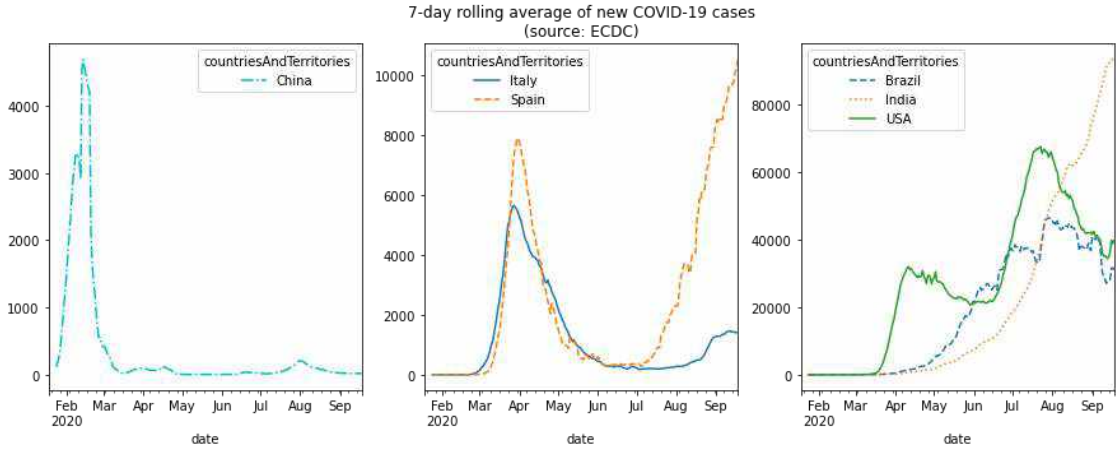


[8]:
```
new_cases_rolling_average = covid.pivot_table(
    index=covid.index,
    columns='countriesAndTerritories',
    values='cases'
).rolling(7).mean()
```

[9]:
```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
new_cases_rolling_average[['China']].plot(ax=axes[0],
```

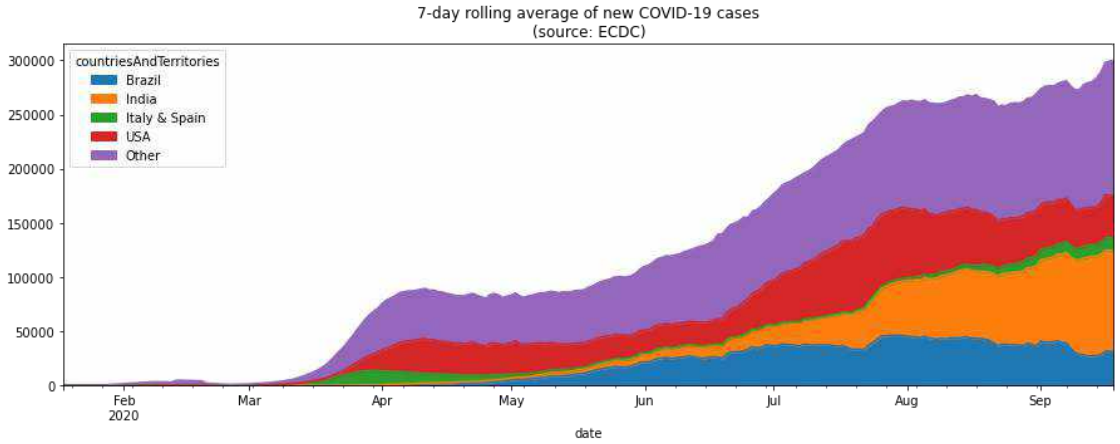
```
style='-.c') new_cases_rolling_average[['Italy',
'Spain']].plot( ax=axes[1], style=['-', '--'],
  title='7-day rolling average of new COVID-19 cases\n(source:
                                                        ECDC)'
)
new_cases_rolling_average[['Brazil', 'India', 'USA']]\
    .plot(ax=axes[2], style=['--', ':', '-'])
```

[9]: <AxesSubplot:xlabel='date'>



[10]:
```
cols = [
    col for col in new_cases_rolling_average.columns
    if col not in ['USA', 'Brazil', 'India', 'Italy & Spain']
]
new_cases_rolling_average.assign(
    **{'Italy & Spain': lambda x: x.Italy + x.Spain}
).sort_index(axis=1).assign(
    Other=lambda x: x[cols].sum(axis=1)
).drop(columns=cols).plot(
    kind='area', figsize=(15, 5),
    title='7-day rolling average of new COVID-19 cases \n(source: ECDC)'
)
```

[10]: <AxesSubplot:title={'center':'7-day rolling average of new COVID-19 cases\n(source: ECDC)'}, xlabel='date'>

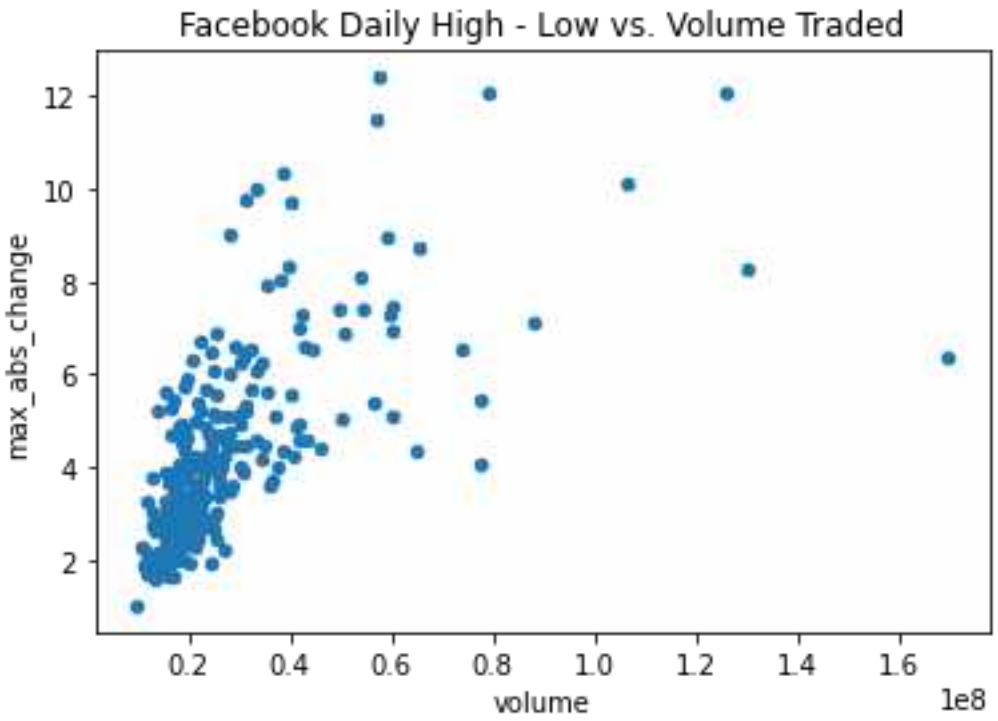


6 Visualizing relationships between variables

7 Visualizing relationships between variables

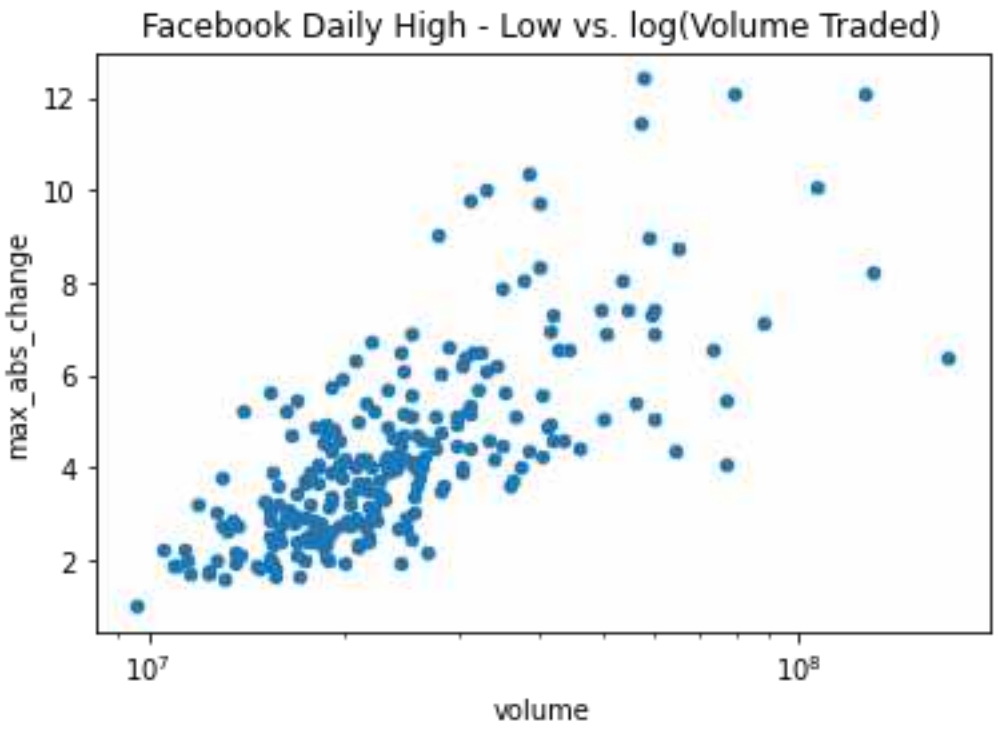
[11]:
```
fb.assign(
    max_abs_change=fb.high - fb.low
).plot(
    kind='scatter',  ='volume',  ='max_abs_change',
    title='Facebook Daily High - Low vs. Volume Traded '
)
```

[11]: <AxesSubplot:title={'center':'Facebook Daily High - Low vs. Volume Traded'}, xlabel='volume', ylabel='max_abs_change'>



[12]:
```
fb.assign(
    max_abs_change=fb.high - fb.low
).plot(
    kind='scatter',  ='volume',  ='max_abs_change',
    title='Facebook Daily High - Low vs. log(Volume Traded) ',
    logx=True
)
```

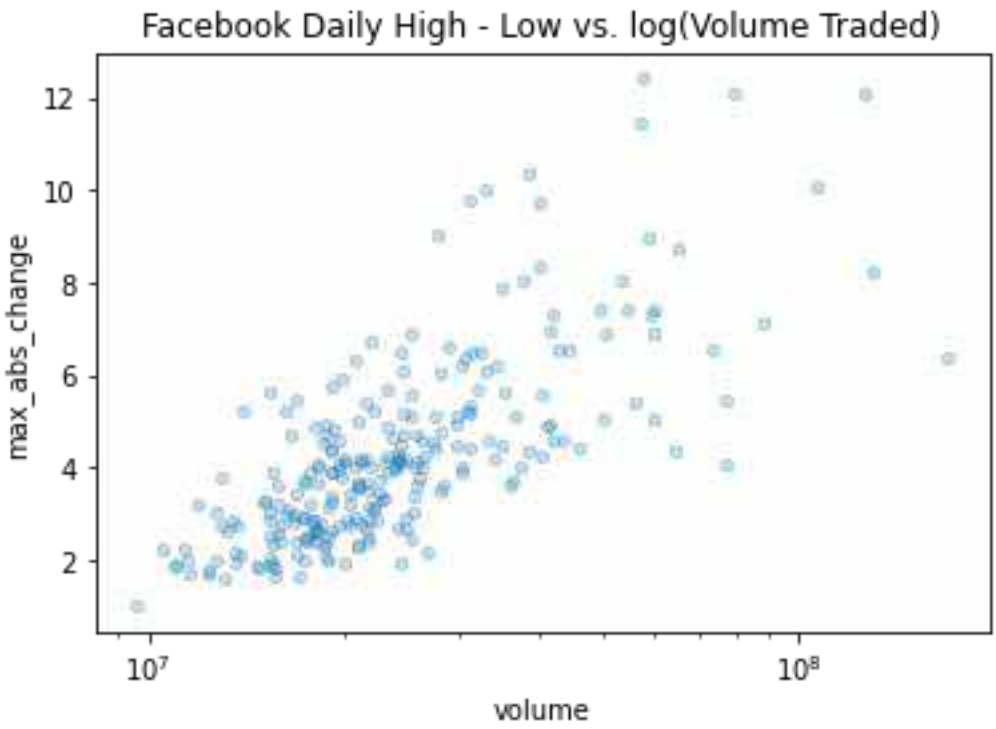
[12]: <AxesSubplot:title={'center':'Facebook Daily High - Low vs. log(Volume Traded)'}, xlabel='volume', ylabel='max_abs_change'>



8 Adding Transparency to Plots with alpha

[13]:
```
fb.assign(
    max_abs_change=fb.high - fb.low
).plot(
    kind='scatter',  ='volume',  ='max_abs_change',
    title='Facebook Daily High - Low vs. log(Volume Traded) ',
    logx=True, alpha=0.25
)
```

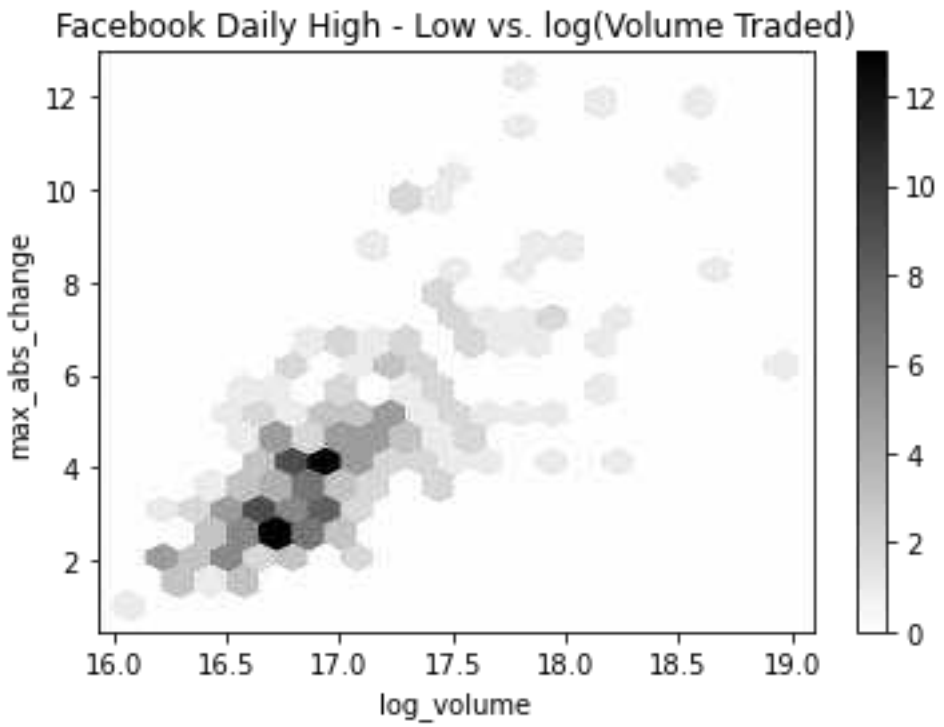
[13]: <AxesSubplot:title={'center':'Facebook Daily High - Low vs. log(Volume
Traded)'}, xlabel='volume', ylabel='max_abs_change'>



9 Hexbins

[14]:
```
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
)
```

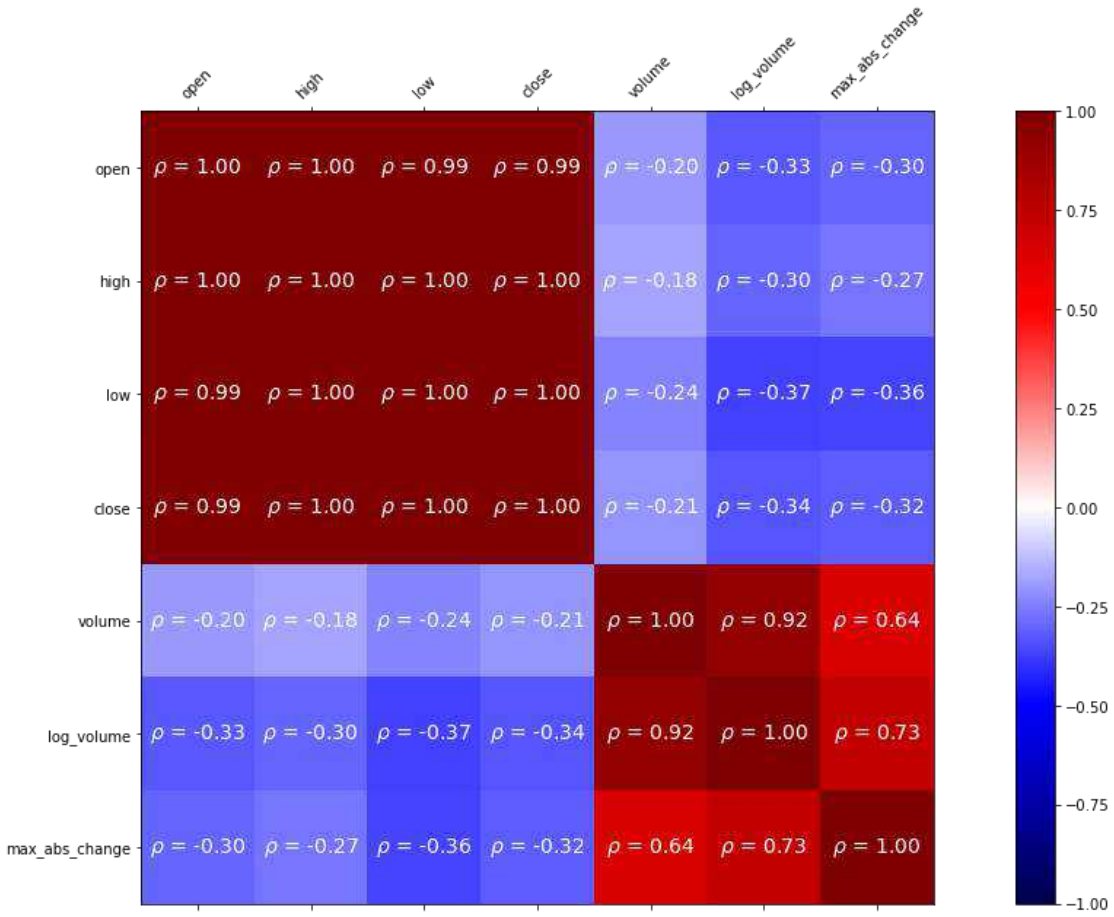
[14]:
```
<AxesSubplot:title={'center':'Facebook Daily High - Low vs.
log(Volume
    Traded)'}, xlabel='log_volume', ylabel='max_abs_change'>
```



10 Visualizing Correlations with Heatmaps

[15]:
```
fig, ax = plt.subplots(figsize=(20, 10))

# calculate the correlation matrix
fb_corr = fb.assign(
    log_volume=np.log(fb.volume),
    max_abs_change=fb.high - fb.low
).corr()

# create the heatmap and colorbar
im = ax.matshow(fb_corr, cmap='seismic')
im.set_clim(-1, 1)
fig.colorbar(im)

# label the ticks with the column names
labels = [col.lower() for col in fb_corr.columns]
ax.set_xticks(ax.get_xticks()[1:-1]) # to handle bug in matplotlib
ax.set_xticklabels(labels, rotation=45)
ax.set_yticks(ax.get_yticks()[1:-1]) # to handle bug in matplotlib
ax.set_yticklabels(labels)
```

```
# include the value of the correlation coefficient in the boxes
for (i, j), coef in np.ndenumerate(fb_corr):
    ax.text(
        i, j, fr'$\rho$ = {coef:.2f}', # raw (r), format (f) string
        ha='center', va='center',
        color='white', fontsize=14
    )
```

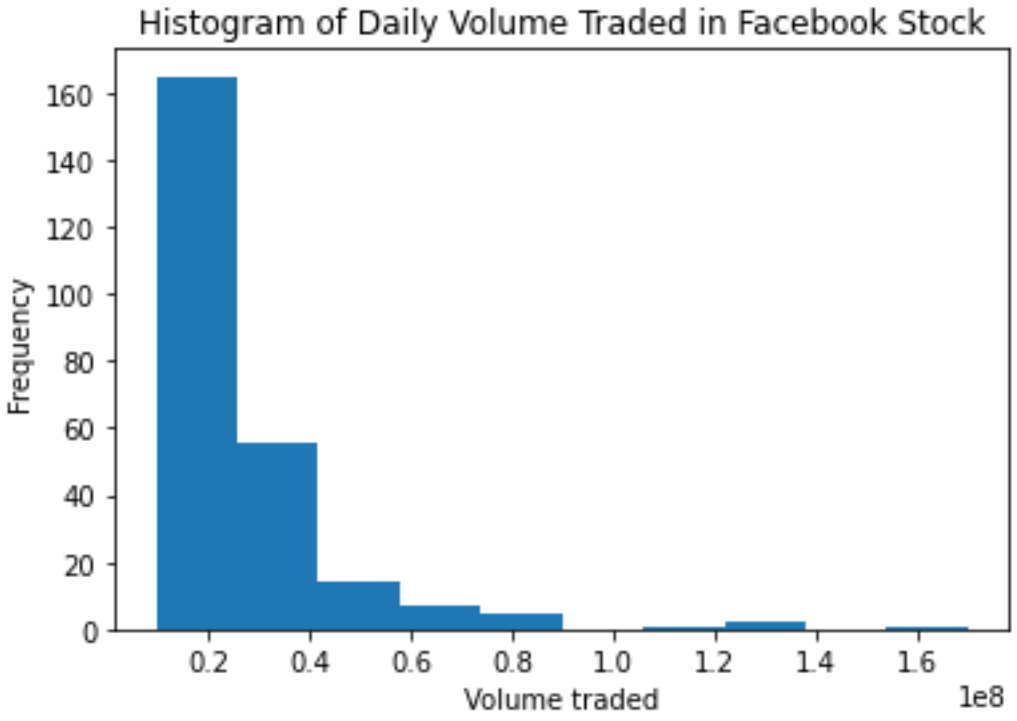


11 Visualizing distributions

12 Histograms

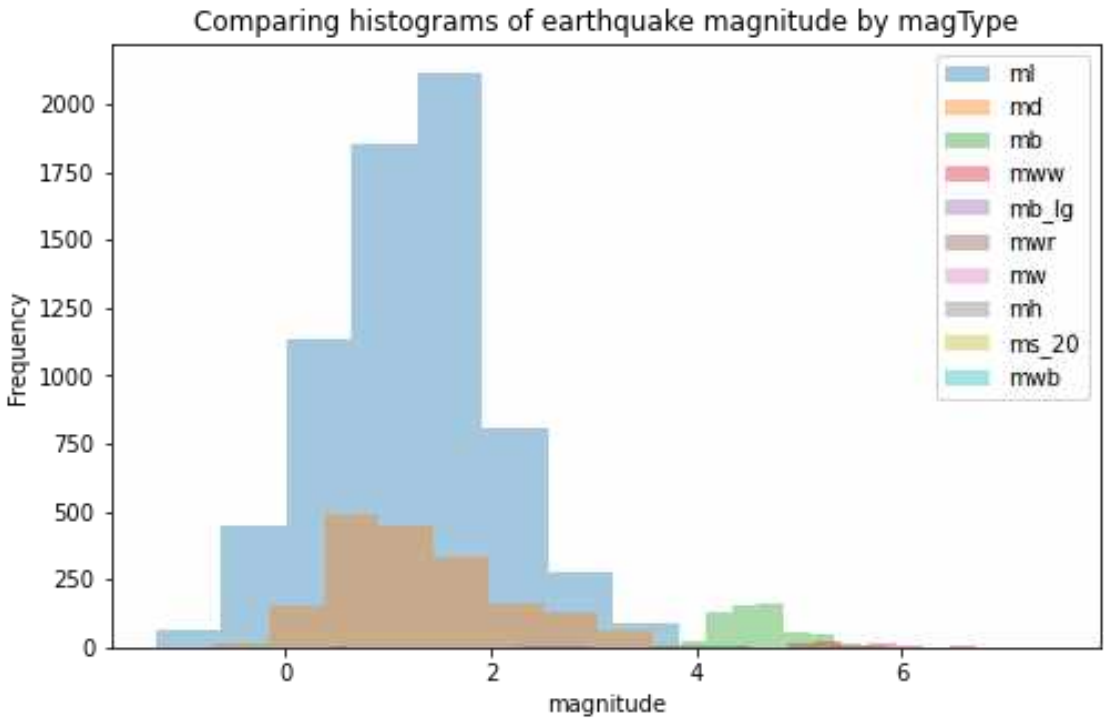
[16]:
```
fb.volume.plot(
    kind='hist',
    title='Histogram of Daily Volume Traded in Facebook Stock '
)
plt.xlabel('Volume traded')
```

[16]: Text(0.5, 0, 'Volume traded')



[17]:
```
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')
```

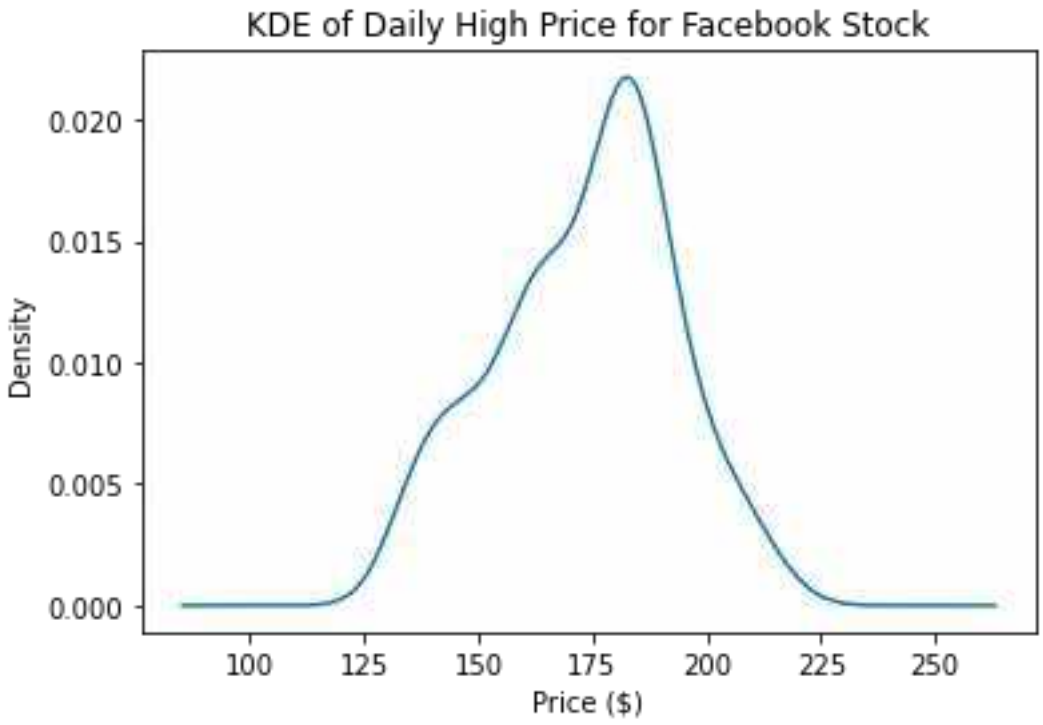
[17]: Text(0.5, 0, 'magnitude')



13 Kernel Density Estimation (KDE)

[18]:
```
fb.high.plot(
    kind='kde',
    title='KDE of Daily High Price for Facebook Stock '
)
plt.xlabel('Price ($)')
```

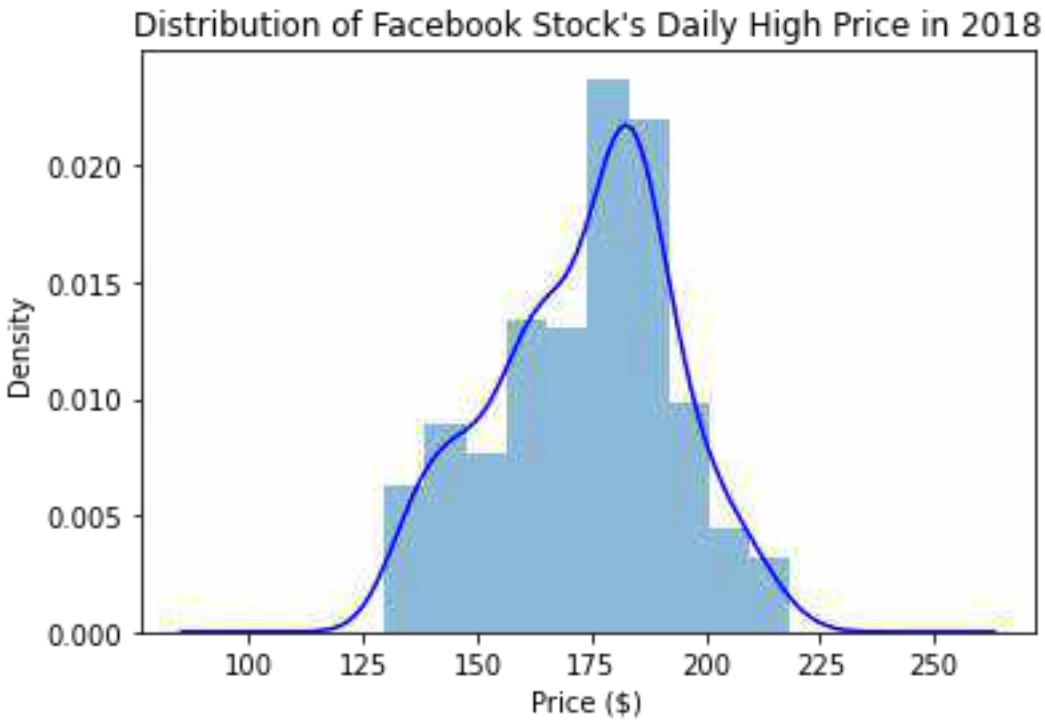
[18]:
```
Text(0.5, 0, 'Price ($)')
```



14 Adding to the result of plot()

[19]:
```
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 ($)')
```

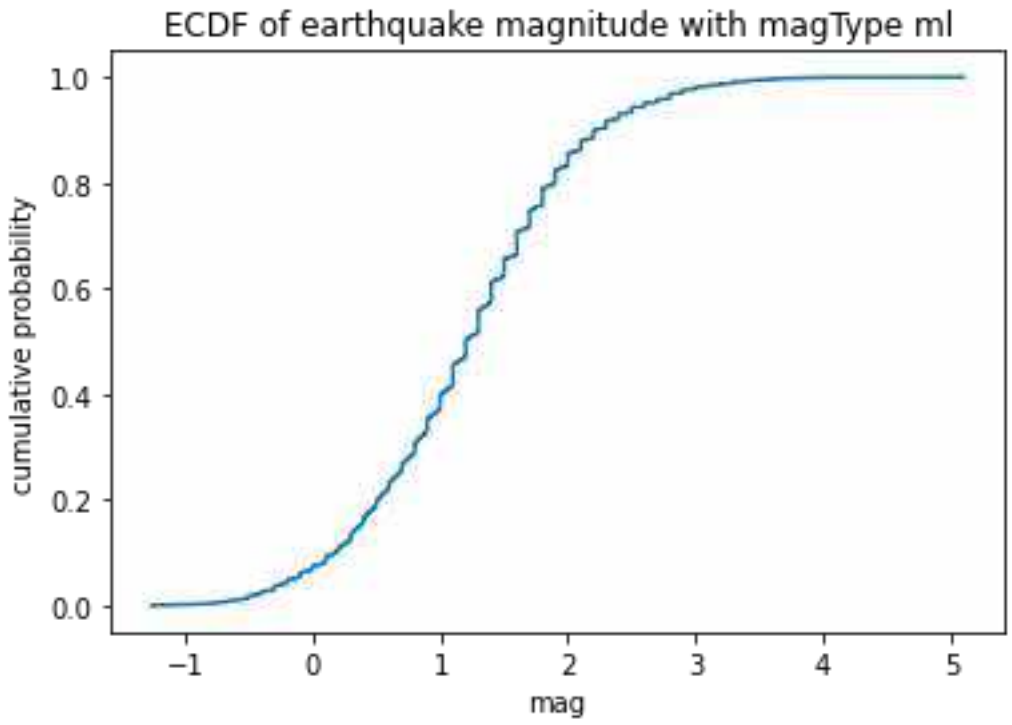
[19]:
```
Text(0.5, 0, 'Price ($)')
```



15 Plotting the ECDF

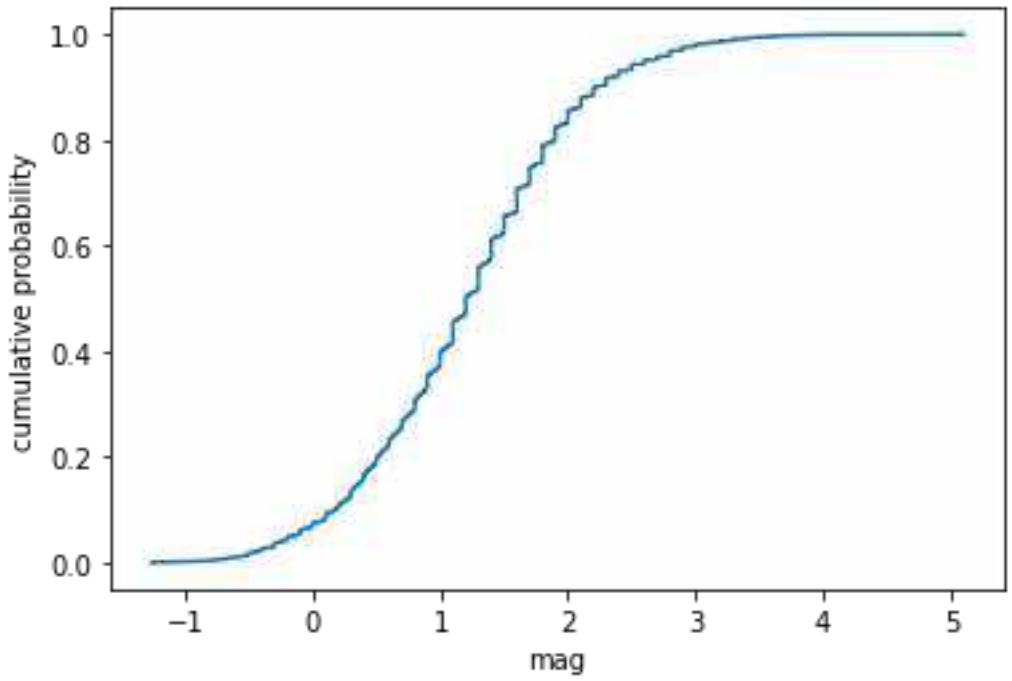
[20]:
```
from statsmodels.distributions.empirical_distribution
import ECDF ecdf = ECDF(quakes.query('magType ==
"ml"').mag) plt.plot(ecdf.x, ecdf.y)
# axis labels (we will cover this in chapter 6)
plt.xlabel('mag') # add x-axis label
plt.ylabel('cumulative probability') # add y-axis
label # add title (we will cover this in chapter
6) plt.title('ECDF of earthquake magnitude with
magType ml')
```

[20]:
```
Text(0.5, 1.0, 'ECDF of earthquake magnitude with magType ml')
```



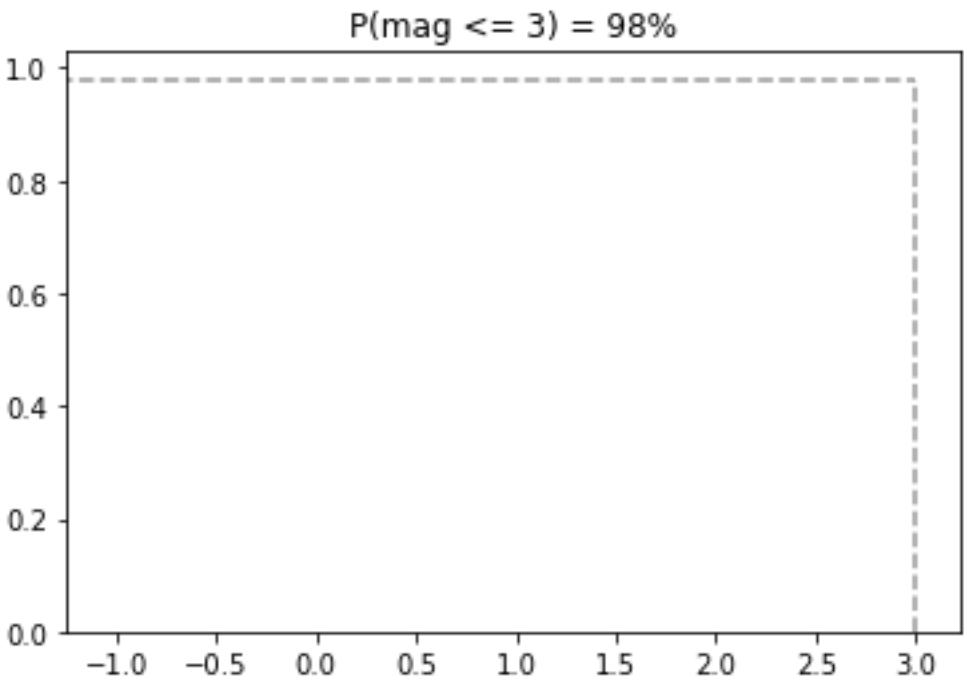
[21]:
```
from statsmodels.distributions.empirical_distribution
import ECDF ecdf = ECDF(quakes.query('magType ==
"ml"').mag) plt.plot(ecdf.x, ecdf.y)
# formatting below will all be covered in chapter
6
# axis labels
plt.xlabel('mag') # add x-axis label
plt.ylabel('cumulative probability') # add y-axis label
```

[21]:
```
Text(0, 0.5, 'cumulative probability')
```



[23]:
```
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 %')
```

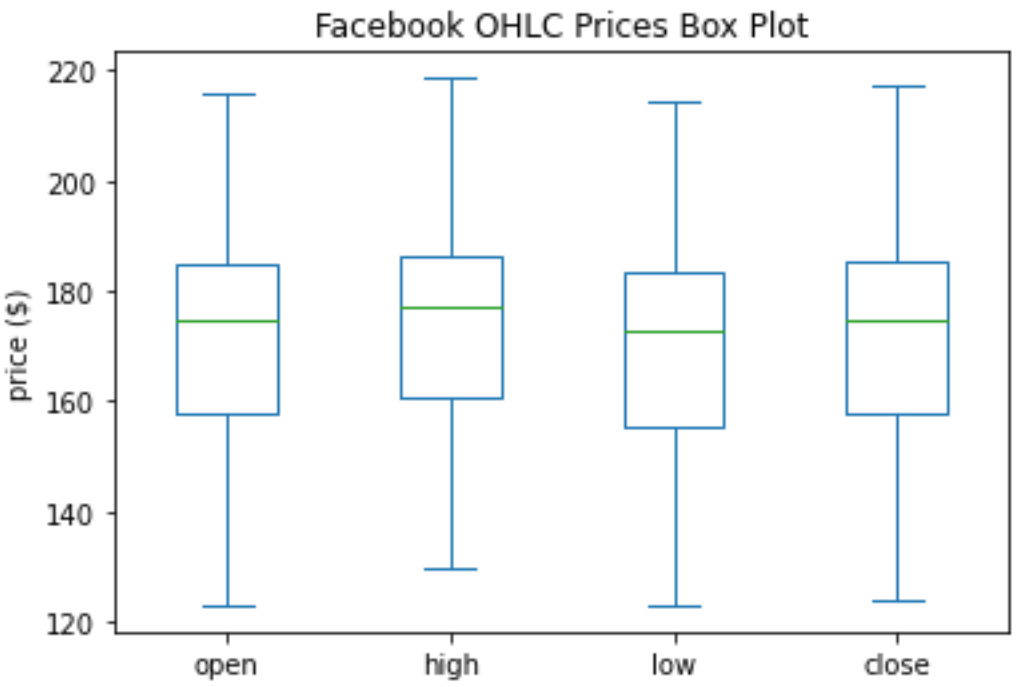
[23]:
```
Text(0.5, 1.0, 'P(mag <= 3) = 98%')
```



16 Box plots

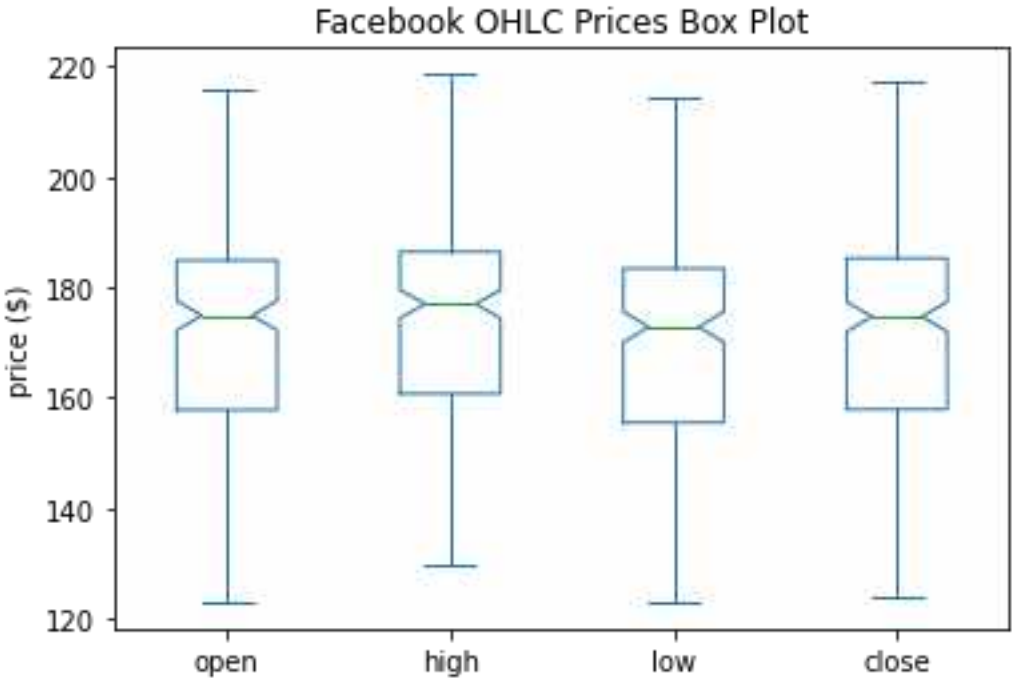
[24]:
```
fb.iloc[:,:4].plot(kind='box', title='Facebook OHLC Prices Box Plot') plt.ylabel('price ($)') # label the x-axis (discussed in chapter 6)
```

[24]:
```
Text(0, 0.5, 'price ($)')
```



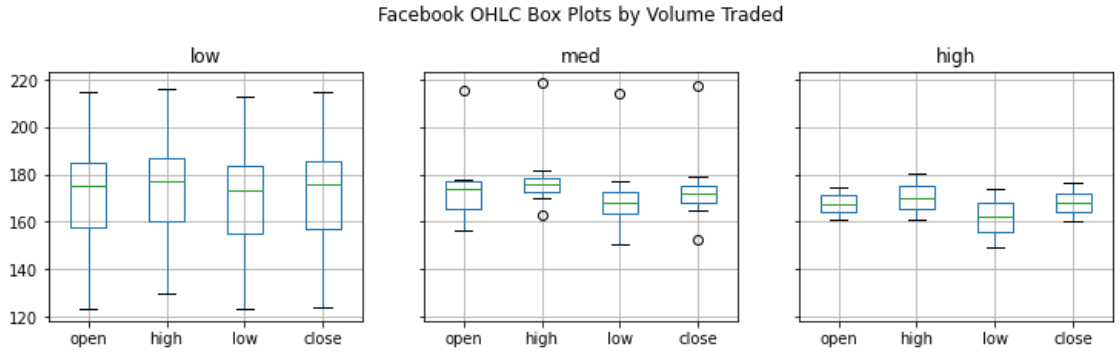
[25]:
```
fb.iloc[:,:4].plot(kind='box', title='Facebook OHLC Prices Box Plot ',
↪notch=True)
plt.ylabel('price ($)')
```

[25]: Text(0, 0.5, 'price ($)')



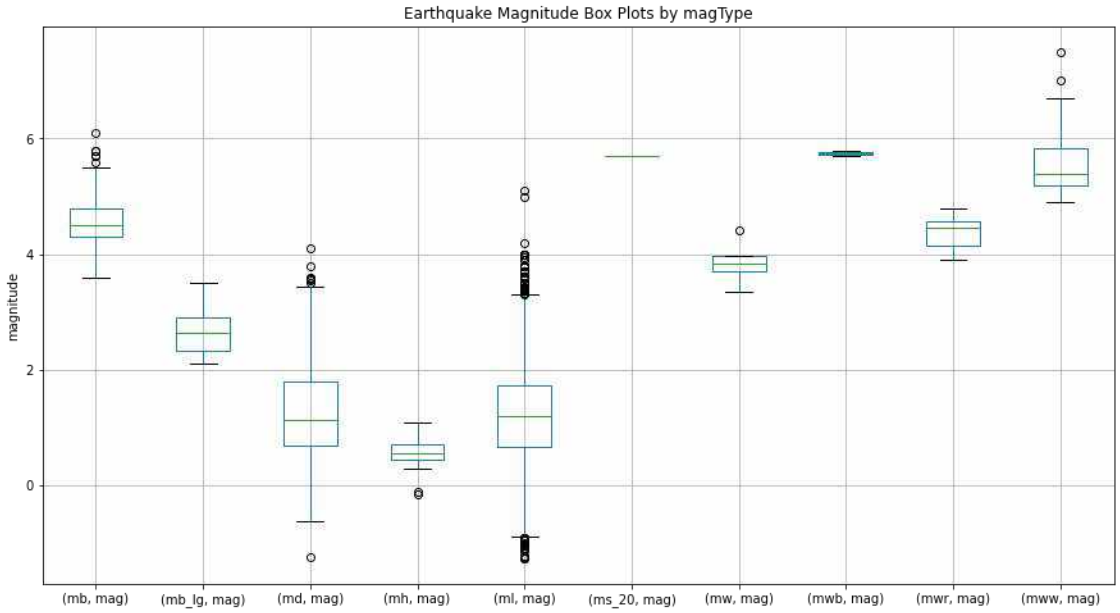
[26]:
```
fb.assign( volume_bin=pd.cut(fb.volume, 3,
    labels=['low', 'med', 'high'])
).groupby('volume_bin').boxplot(
    column=['open', 'high', 'low',
    'close'], layout=(1, 3), figsize=(12,
    3)
)
plt.suptitle('Facebook OHLC Box Plots by Volume Traded', y=1.1)
```

[26]: Text(0.5, 1.1, 'Facebook OHLC Box Plots by Volume Traded')



[27]:
```
quakes[['mag', 'magType']].groupby('magType').boxplot(
    figsize=(15, 8), subplots=False
)
plt.title('Earthquake Magnitude Box Plots by magType ')
plt.ylabel('magnitude')
```

[27]: Text(0, 0.5, 'magnitude')

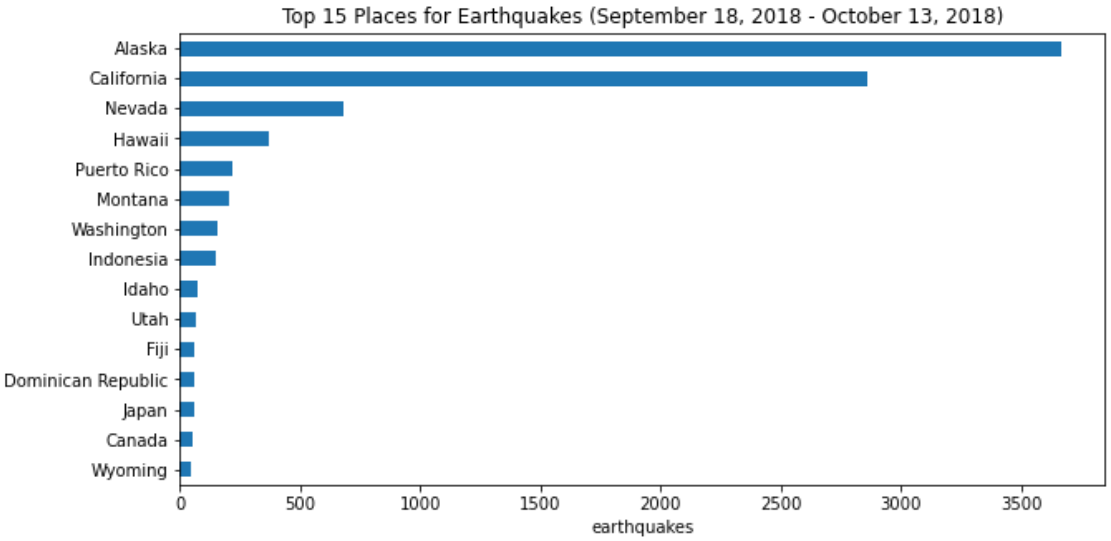


17 Counts and frequencies

18 Bar charts

[28]:
```
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')
```

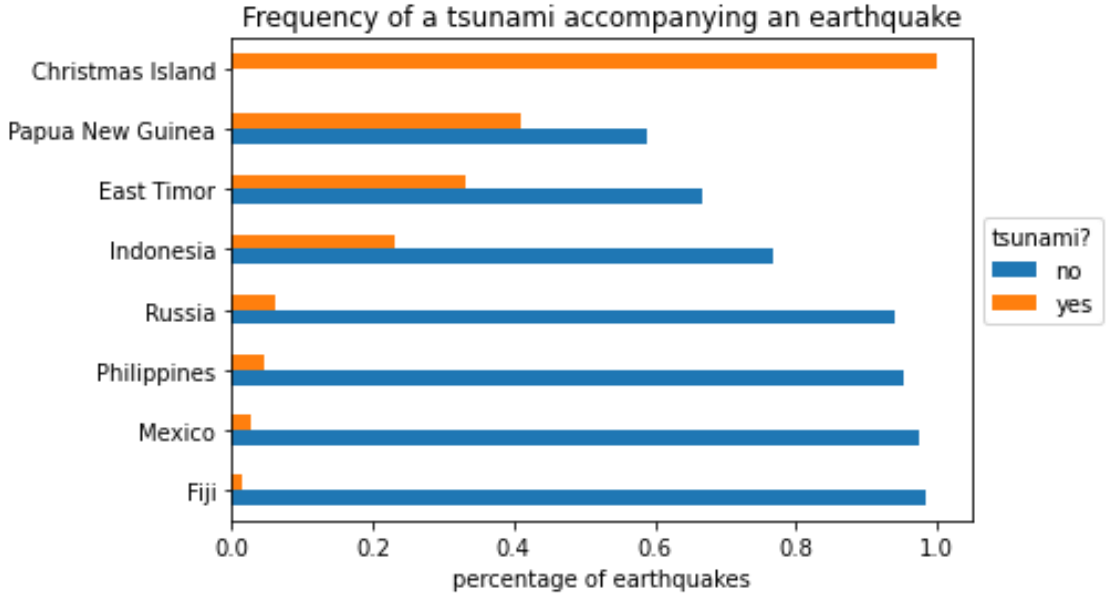
[28]:
```
Text(0.5, 0, 'earthquakes')
```



19 Grouped Bars

[29]:
```
quakes.groupby(['parsed_place', 'tsunami']).mag.count()\
    .unstack().apply(lambda x: x / x.sum(), axis=1)\
    .rename(columns={0: 'no', 1: 'yes'})\
    .sort_values('yes', ascending=False)[7::-1]\
    .plot.barh(
        title='Frequency of a tsunami accompanying an earthquake '
    )

# move legend to the right of the plot
plt.legend(title='tsunami?', bbox_to_anchor=(1, 0.65))

# label the axes (discussed in chapter 6)
plt.xlabel('percentage of earthquakes')
plt.ylabel('')
```

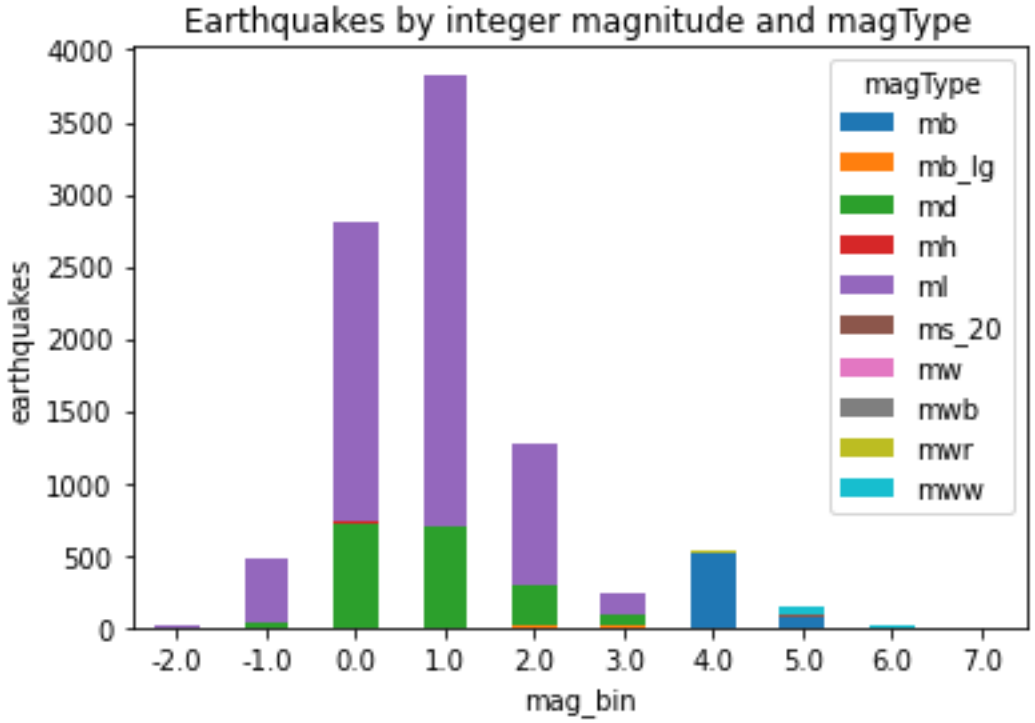
[29]:
```
Text(0, 0.5, '')
```



20 Stacked bars

[30]:
```
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, ylabel='earthquakes',
    title='Earthquakes by integer magnitude and magType '
)
```

[30]: <AxesSubplot:title={'center':'Earthquakes by integer magnitude and magType'}, xlabel='mag_bin', ylabel='earthquakes'>

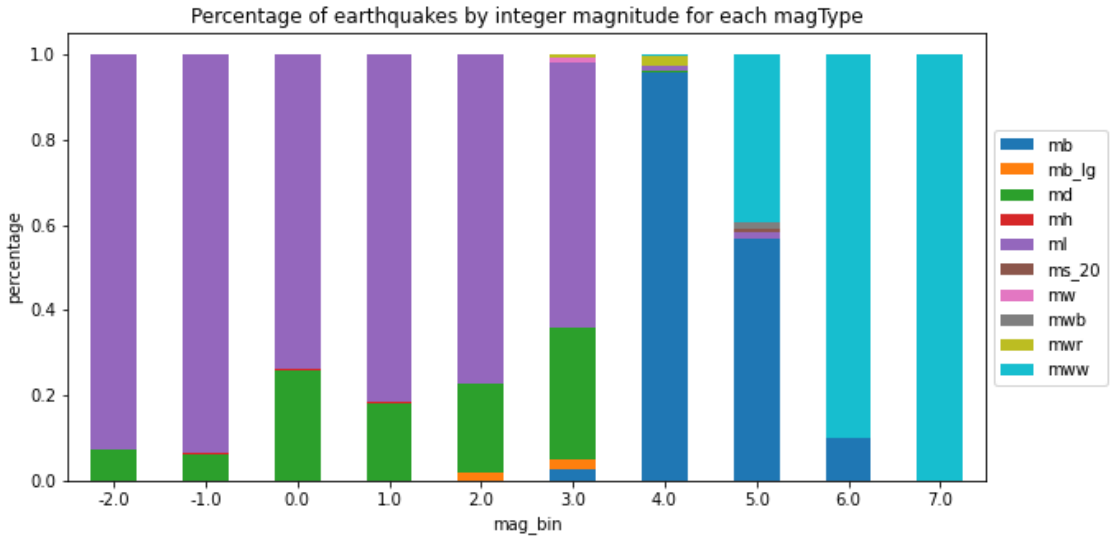


21 Normalized stacked bars

[31]:
```
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')
```

[31]:
```
Text(0, 0.5, 'percentage')
```



22 Part-2

23 The pandas.plotting module

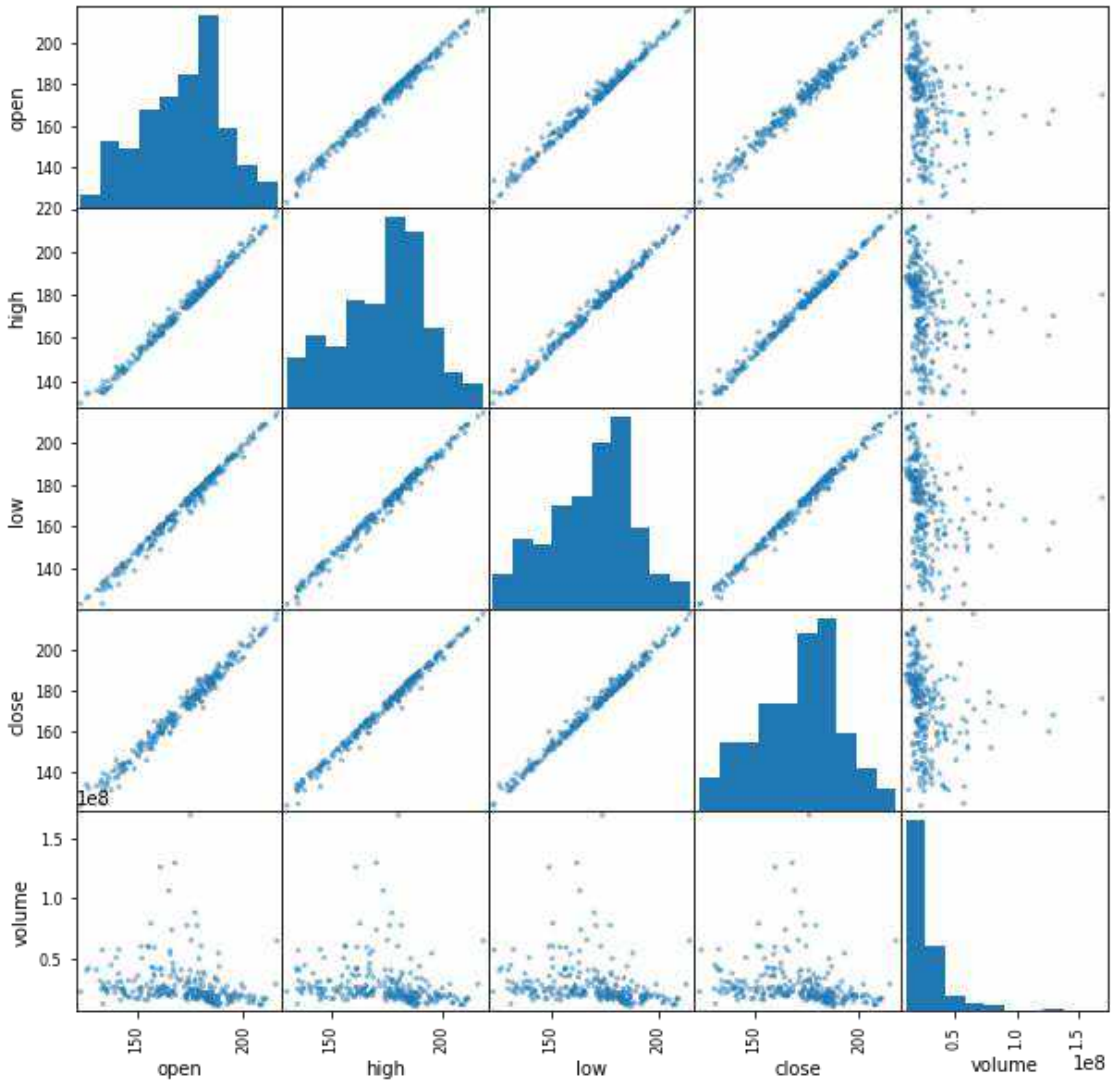
[33]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

fb = pd.read_csv(
    'fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
```

24 Scatter matrix

[34]:
```
from pandas.plotting import scatter_matrix
scatter_matrix(fb, figsize=(10, 10))
```

[34]:
```
array([[<AxesSubplot:xlabel='open', ylabel='open'>,
        <AxesSubplot:xlabel='high', ylabel='open'>,
        <AxesSubplot:xlabel='low', ylabel='open'>,
        <AxesSubplot:xlabel='close', ylabel='open'>,
        <AxesSubplot:xlabel='volume', ylabel='open'>],
```

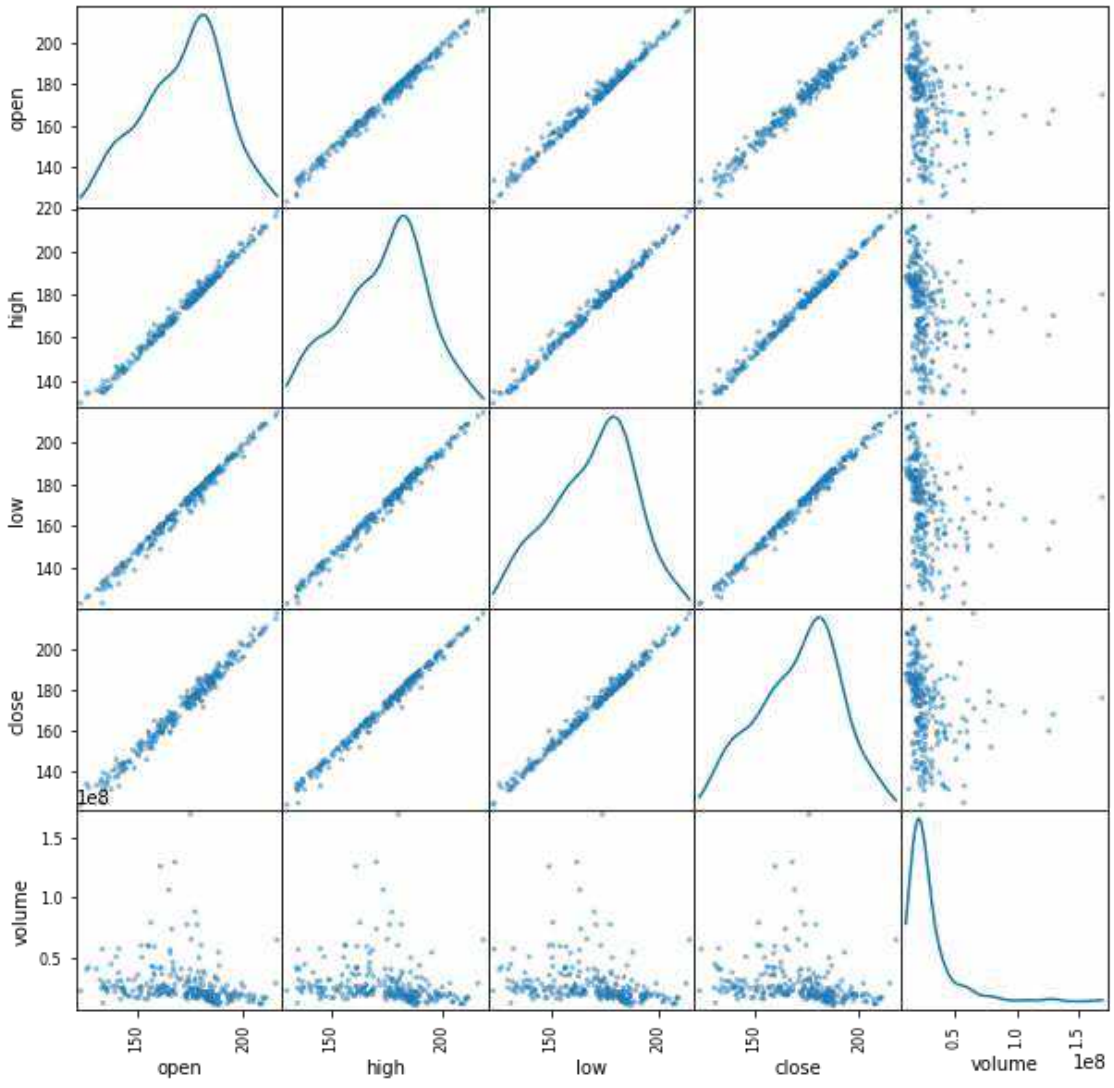
```
[<AxesSubplot:xlabel='open', ylabel='high'>,
 <AxesSubplot:xlabel='high', ylabel='high'>,
 <AxesSubplot:xlabel='low', ylabel='high'>,
 <AxesSubplot:xlabel='close', ylabel='high'>,
 <AxesSubplot:xlabel='volume', ylabel='high'>],
[<AxesSubplot:xlabel='open', ylabel='low'>,
 <AxesSubplot:xlabel='high', ylabel='low'>,
 <AxesSubplot:xlabel='low', ylabel='low'>,
 <AxesSubplot:xlabel='close', ylabel='low'>,
 <AxesSubplot:xlabel='volume', ylabel='low'>],
[<AxesSubplot:xlabel='open', ylabel='close'>,
 <AxesSubplot:xlabel='high', ylabel='close'>,
 <AxesSubplot:xlabel='low', ylabel='close'>,
 <AxesSubplot:xlabel='close', ylabel='close'>,
 <AxesSubplot:xlabel='volume', ylabel='close'>],
[<AxesSubplot:xlabel='open', ylabel='volume'>,
 <AxesSubplot:xlabel='high', ylabel='volume'>,
 <AxesSubplot:xlabel='low', ylabel='volume'>,
 <AxesSubplot:xlabel='close', ylabel='volume'>,
 <AxesSubplot:xlabel='volume', ylabel='volume'>]],
 dtype=object)
```



[35]:
```
scatter_matrix(fb, figsize=(10, 10), diagonal='kde')
```

[35]:
```
array([[<AxesSubplot:xlabel='open', ylabel='open'>,
        <AxesSubplot:xlabel='high', ylabel='open'>,
        <AxesSubplot:xlabel='low', ylabel='open'>,
        <AxesSubplot:xlabel='close', ylabel='open'>,
        <AxesSubplot:xlabel='volume', ylabel='open'>],
       [<AxesSubplot:xlabel='open', ylabel='high'>,
        <AxesSubplot:xlabel='high', ylabel='high'>,
        <AxesSubplot:xlabel='low', ylabel='high'>,
        <AxesSubplot:xlabel='close', ylabel='high'>,
        <AxesSubplot:xlabel='volume', ylabel='high'>],
       [<AxesSubplot:xlabel='open', ylabel='low'>,
        <AxesSubplot:xlabel='high', ylabel='low'>,
        <AxesSubplot:xlabel='low', ylabel='low'>,
        <AxesSubplot:xlabel='close', ylabel='low'>,
```

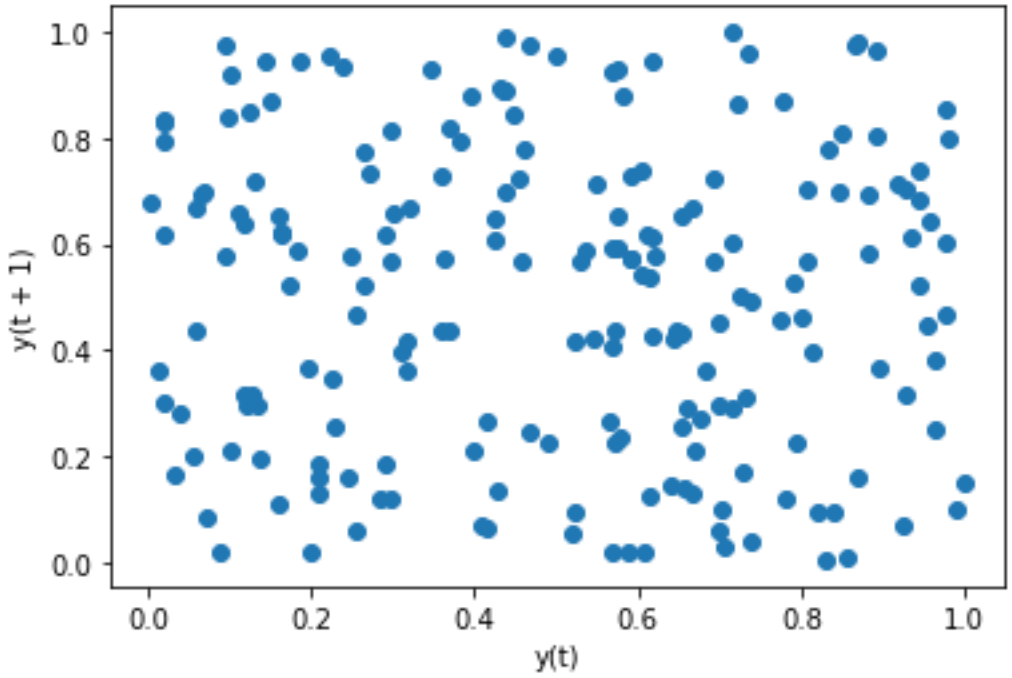
```
 <AxesSubplot:xlabel='volume', ylabel='low'>],
[<AxesSubplot:xlabel='open', ylabel='close'>,
 <AxesSubplot:xlabel='high', ylabel='close'>,
 <AxesSubplot:xlabel='low', ylabel='close'>,
 <AxesSubplot:xlabel='close', ylabel='close'>,
 <AxesSubplot:xlabel='volume', ylabel='close'>],
[<AxesSubplot:xlabel='open', ylabel='volume'>,
 <AxesSubplot:xlabel='high', ylabel='volume'>,
 <AxesSubplot:xlabel='low', ylabel='volume'>,
 <AxesSubplot:xlabel='close', ylabel='volume'>,
 <AxesSubplot:xlabel='volume', ylabel='volume'>]],
 dtype=object)
```



25 Lag plot

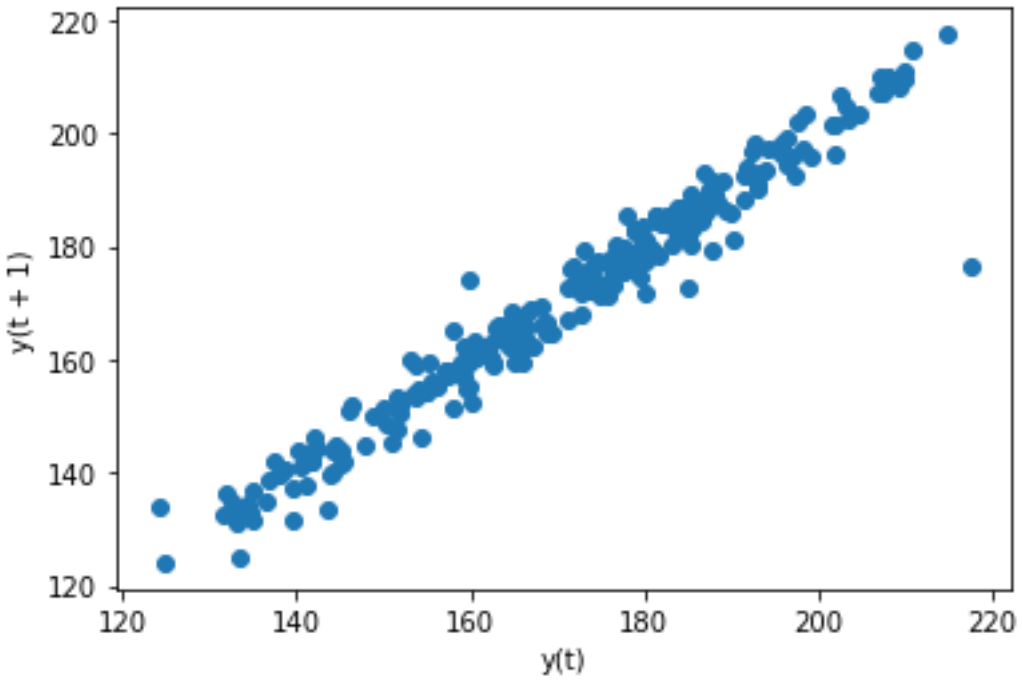
[36]:
```
from pandas.plotting import lag_plot
np.random.seed(0) # make this repeatable
lag_plot(pd.Series(np.random.random(size=200)
))
```

[36]: <AxesSubplot:xlabel='y(t)', ylabel='y(t + 1)'>



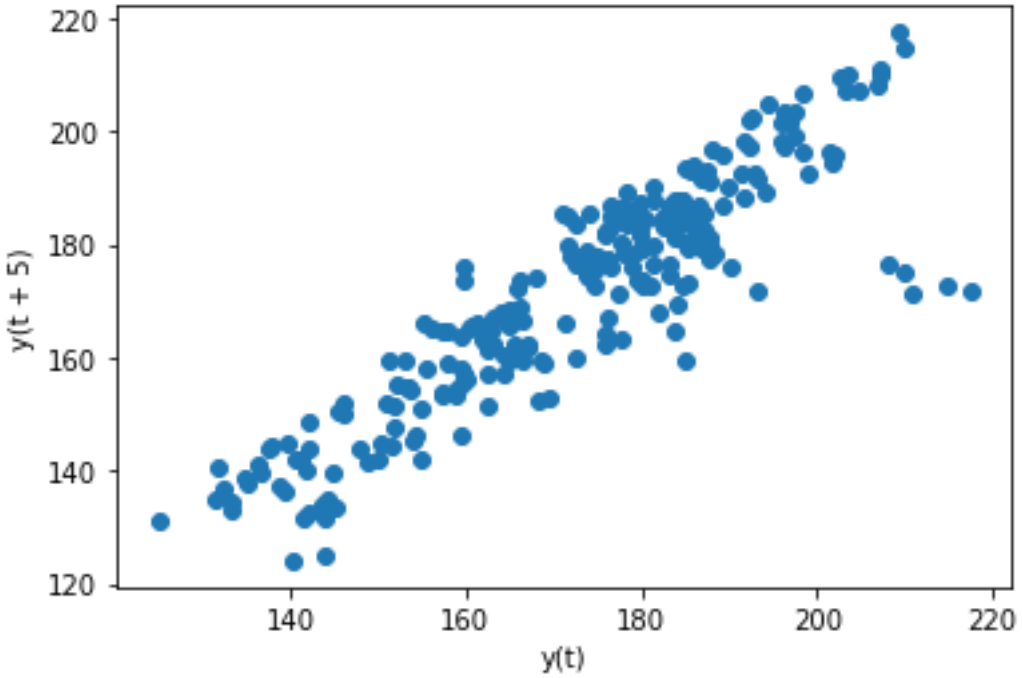
[38]:
```
lag_plot(fb.close)
```

[38]: <AxesSubplot:xlabel='y(t)', ylabel='y(t + 1)'>



[39]:
```
lag_plot(fb.close,lag=5)
```

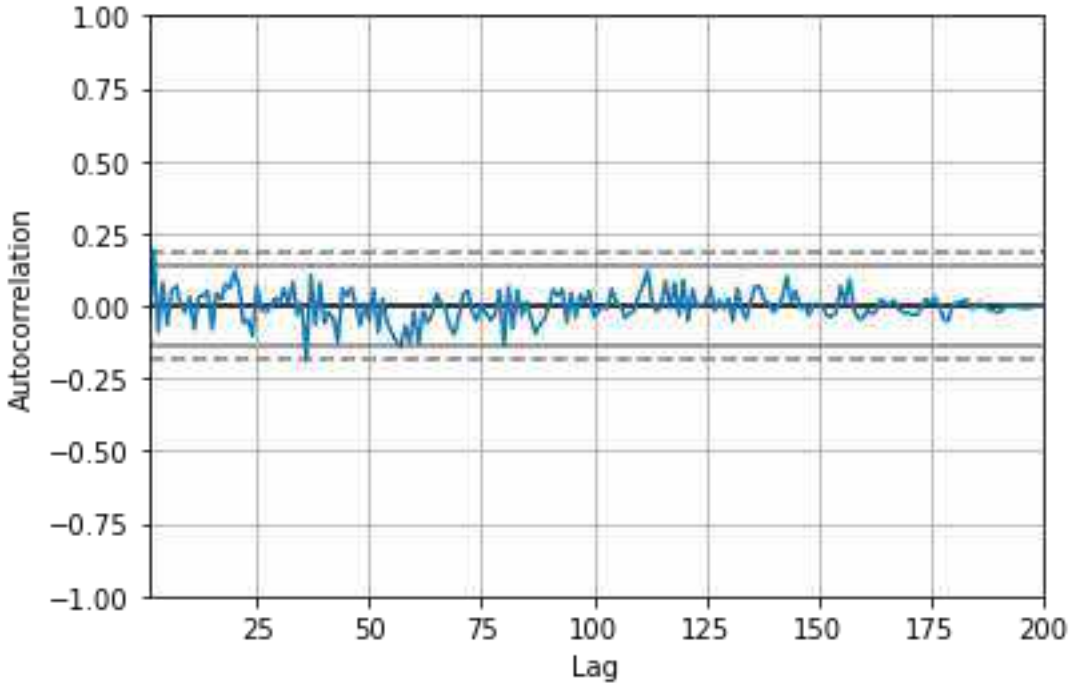
[39]: <AxesSubplot:xlabel='y(t)', ylabel='y(t + 5)'>



26 Autocorrelation plots

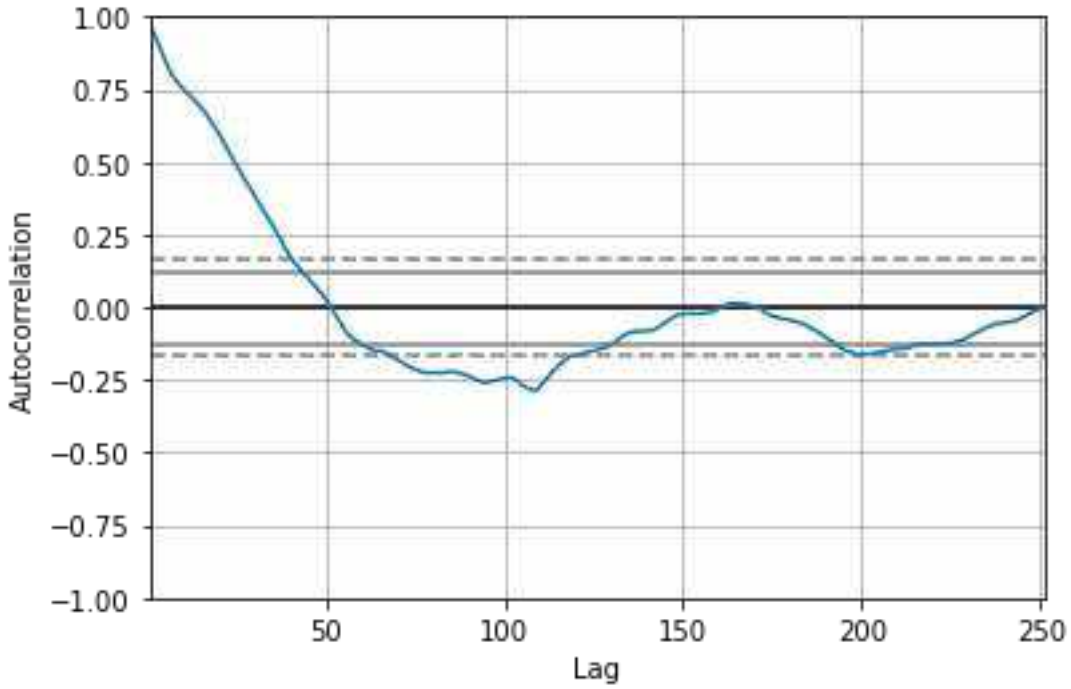
[40]:
```
from pandas.plotting import autocorrelation_plot
np.random.seed(0) # make this repeatable
autocorrelation_plot(pd.Series(np.random.random(size=200
                                                ))))
```

[40]: <AxesSubplot:xlabel='Lag', ylabel='Autocorrelation'>



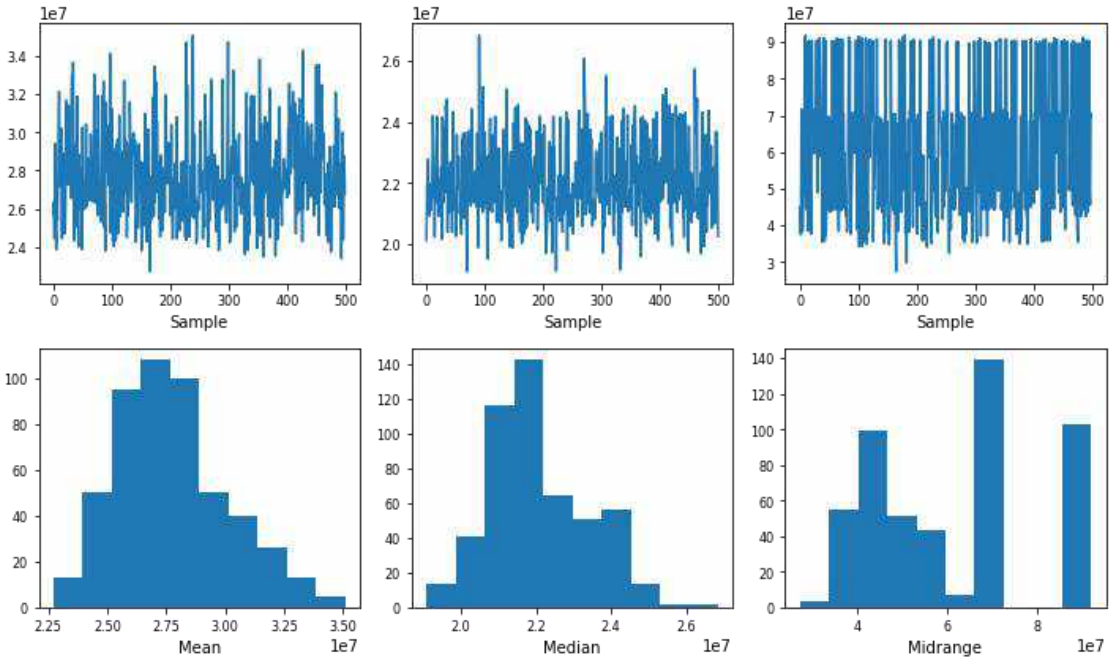
[41]:
```
autocorrelation_plot(fb.close)
```

[41]: <AxesSubplot:xlabel='Lag', ylabel='Autocorrelation'>



27 Bootstrap plot

[42]:
```
from pandas.plotting import bootstrap_plot
fig = bootstrap_plot(fb.volume, fig=plt.figure(figsize=(10, 6)))
```



[ ]:

```

```