In [2]:
```
import matplotlib.pyplot as plt
import pandas as pd
```

9.1 Introduction to Matplotlib

Plotting lines

In [21]:
```
fb = pd.read_csv(
'fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
plt.plot(fb.index, fb.open)
plt.show()
```

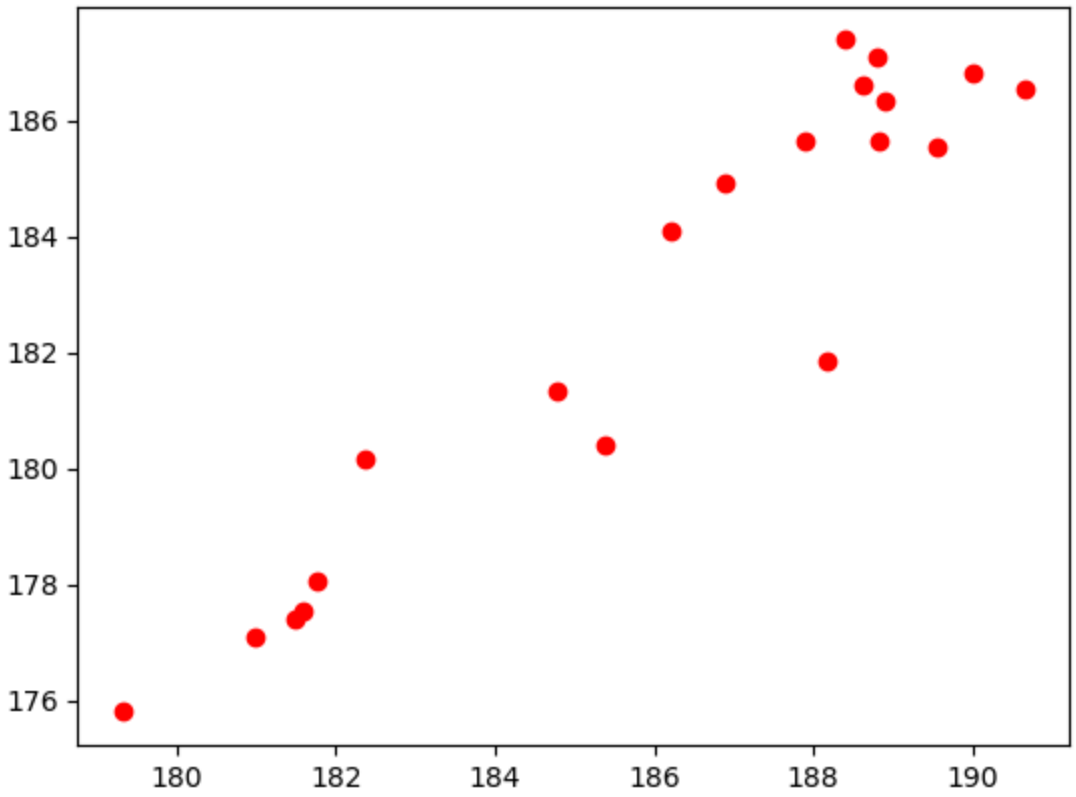
In [22]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
#matplotlib inline so we dont have to type plt.show everytime
#although it doesnt work so i need to put plt.show
```

In [26]:
```
fb = pd.read_csv(
'fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
plt.plot(fb.index, fb.open)

plt.show()
```

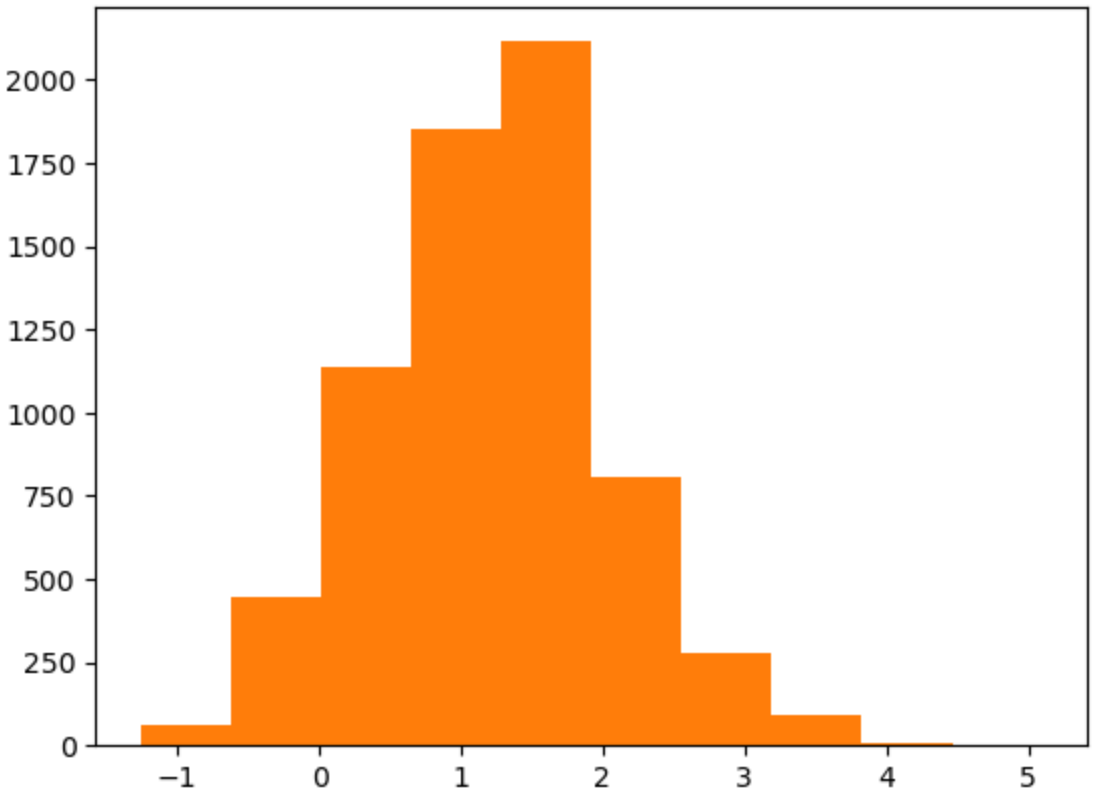
Scatter plots

In [27]:
```
plt.plot('high', 'low', 'ro', data=fb.head(20))
plt.show()
```



Histograms

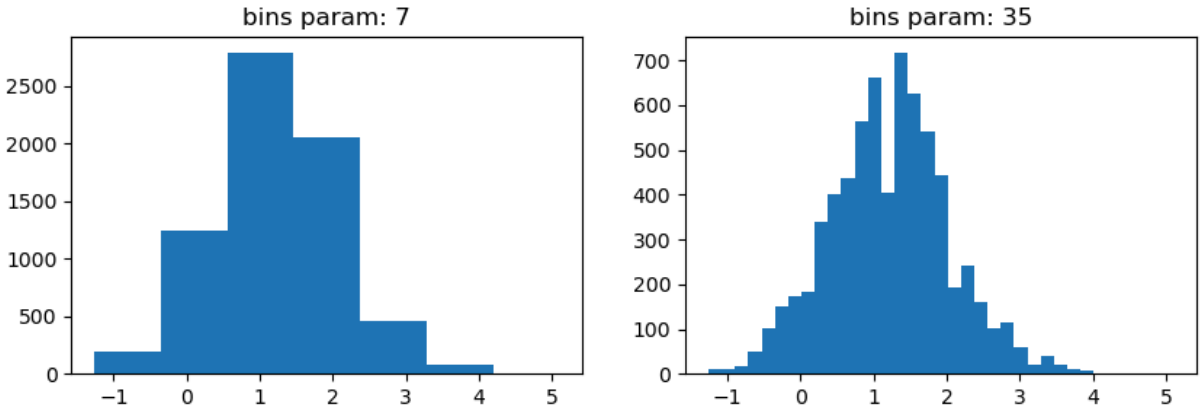
In [30]:
```
quakes = pd.read_csv('earthquakes.csv')
plt.hist(quakes.query('magType == "ml"').mag)
plt.show()
```

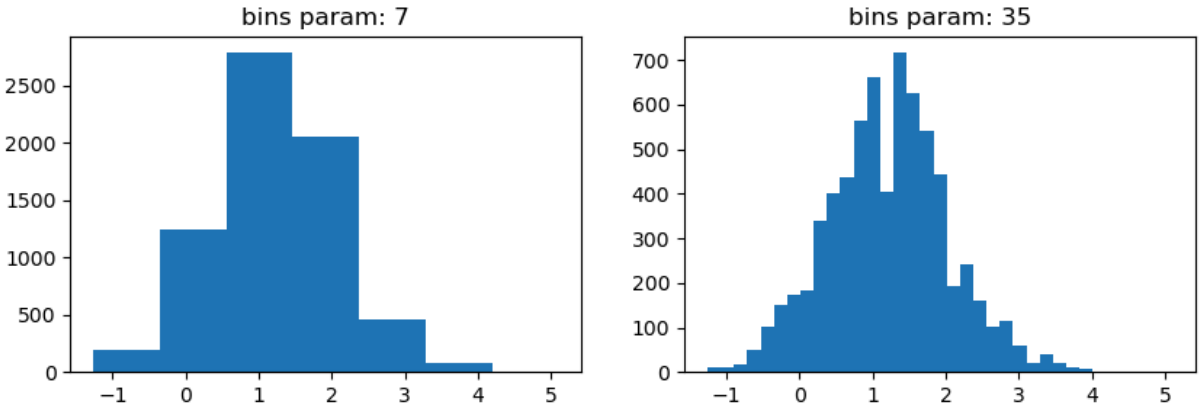


Bin size matters

In [34]:
```
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()
```





Plot components

Figure

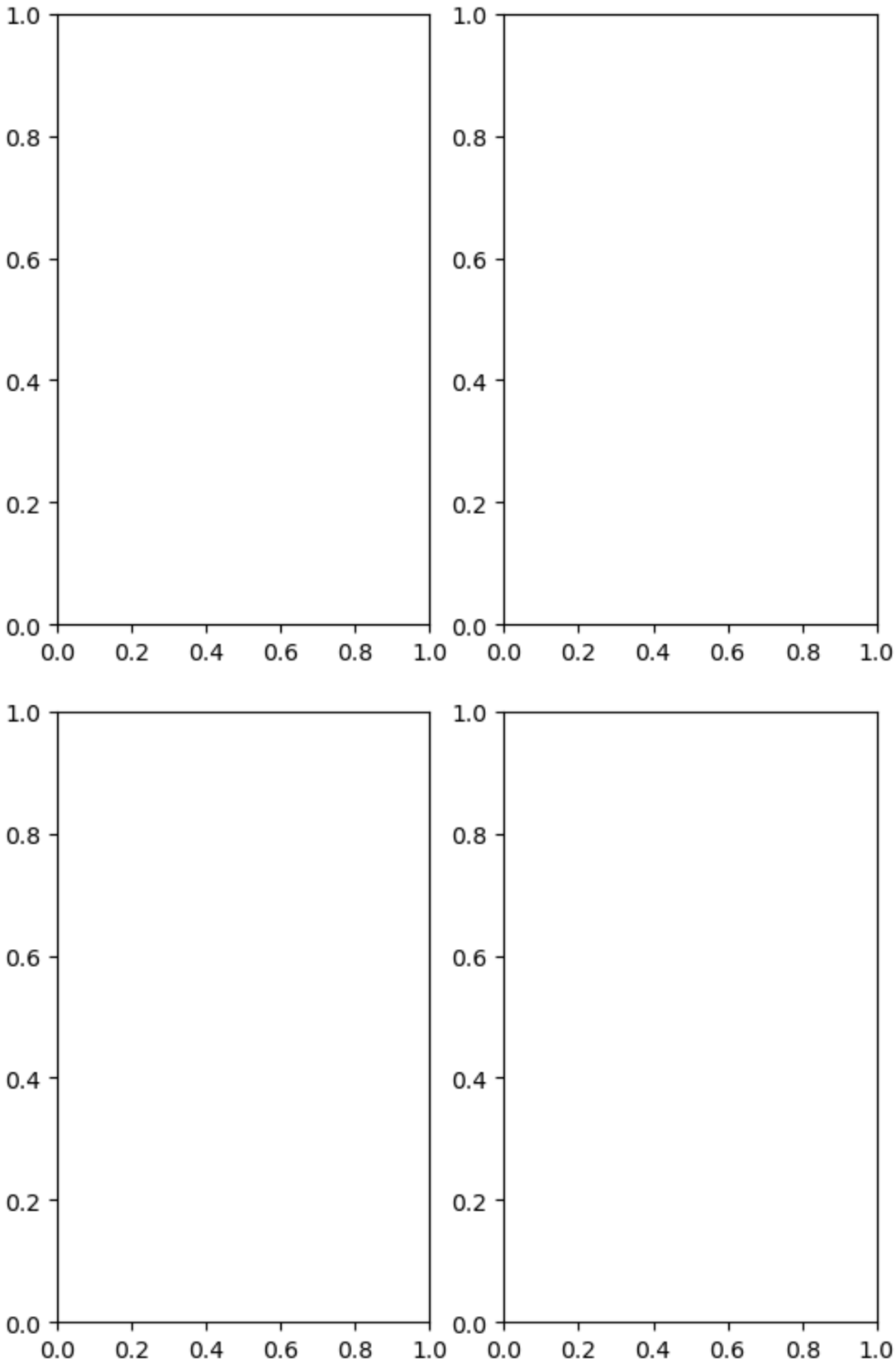
In [35]:
```
fig = plt.figure()
```

Axes

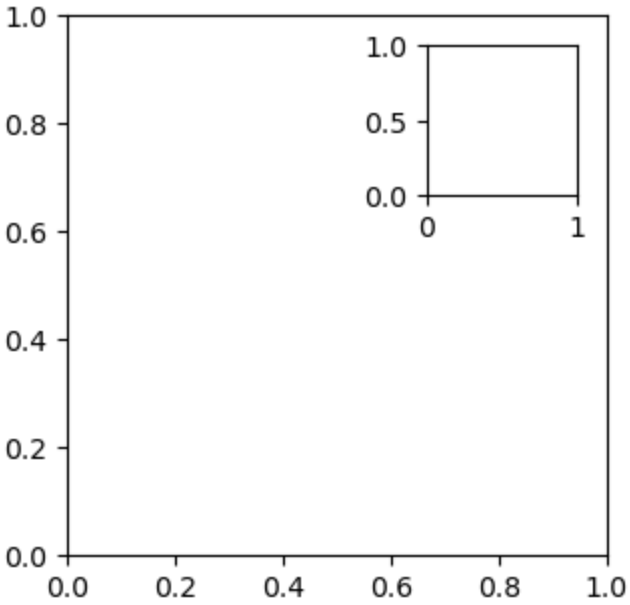
Creating subplots

In [37]:
```
fig, axes = plt.subplots(1, 2)
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



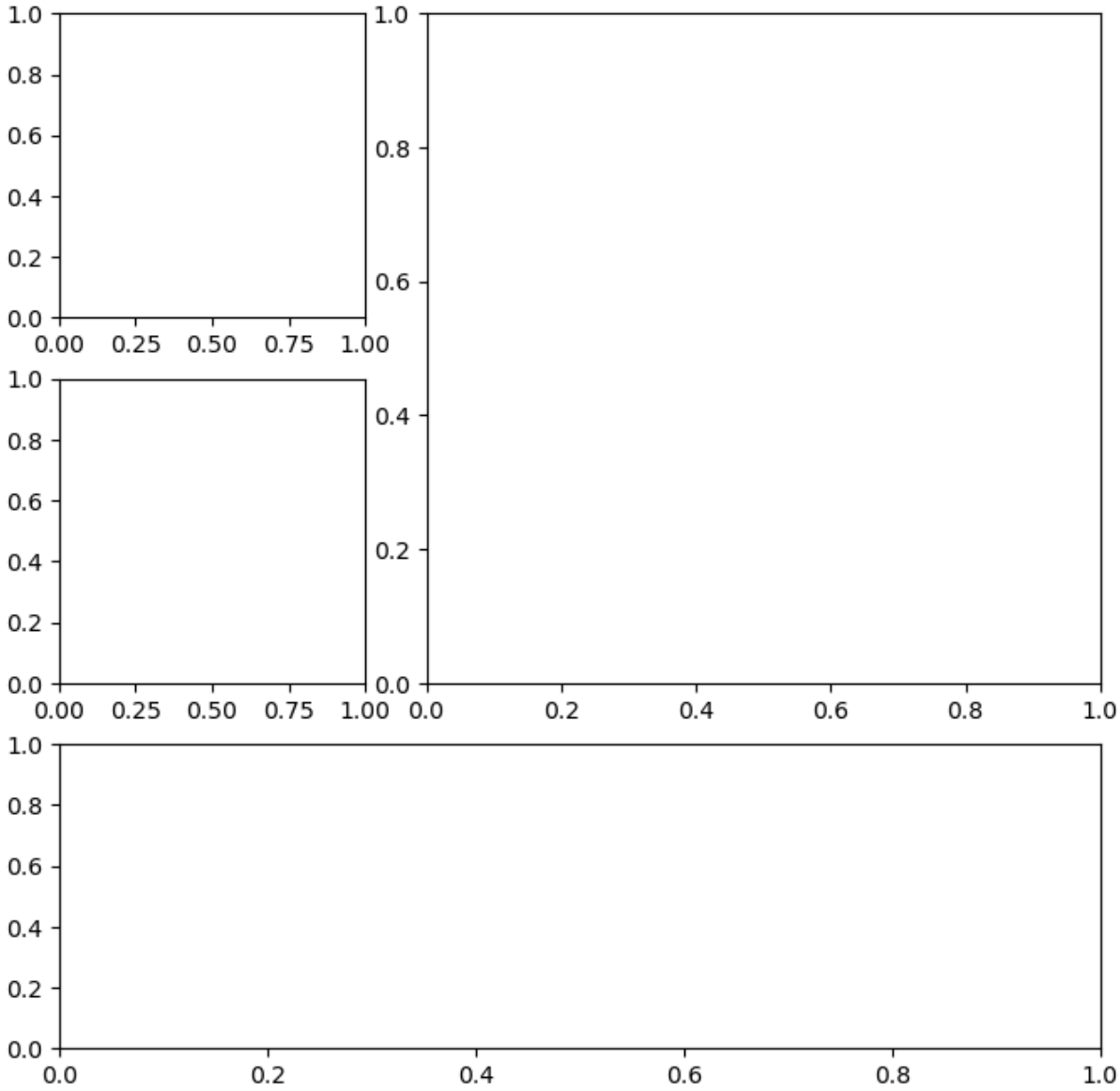
In [38]:
```
fig = plt.figure(figsize=(3, 3))
outside = fig.add_axes([0.1, 0.1, 0.9, 0.9])
inside = fig.add_axes([0.7, 0.7, 0.25, 0.25])
plt.show()
```



Creating Plot Layouts with gridspec

In [40]:
```
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,:])

plt.show()
```



Saving plots

In [42]:
```
# use to save the last plot
fig.savefig('empty.png')
```

Cleaning up

In [43]:
```
plt.close('all')
```

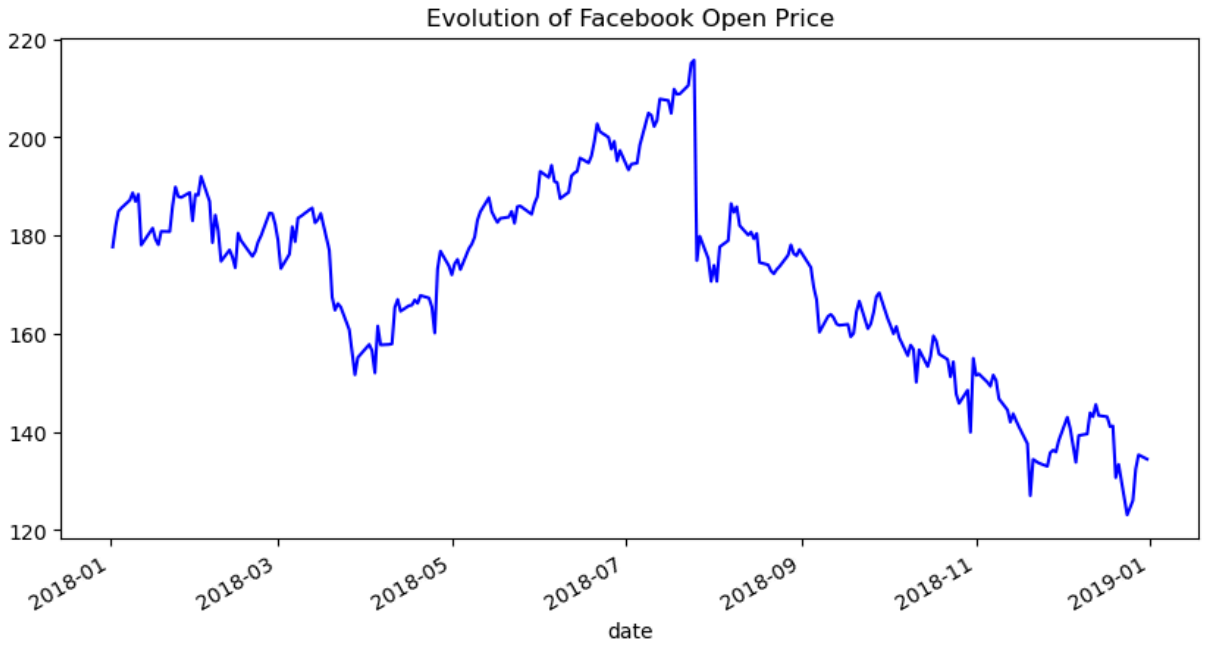
9.2 Plotting with Pandas

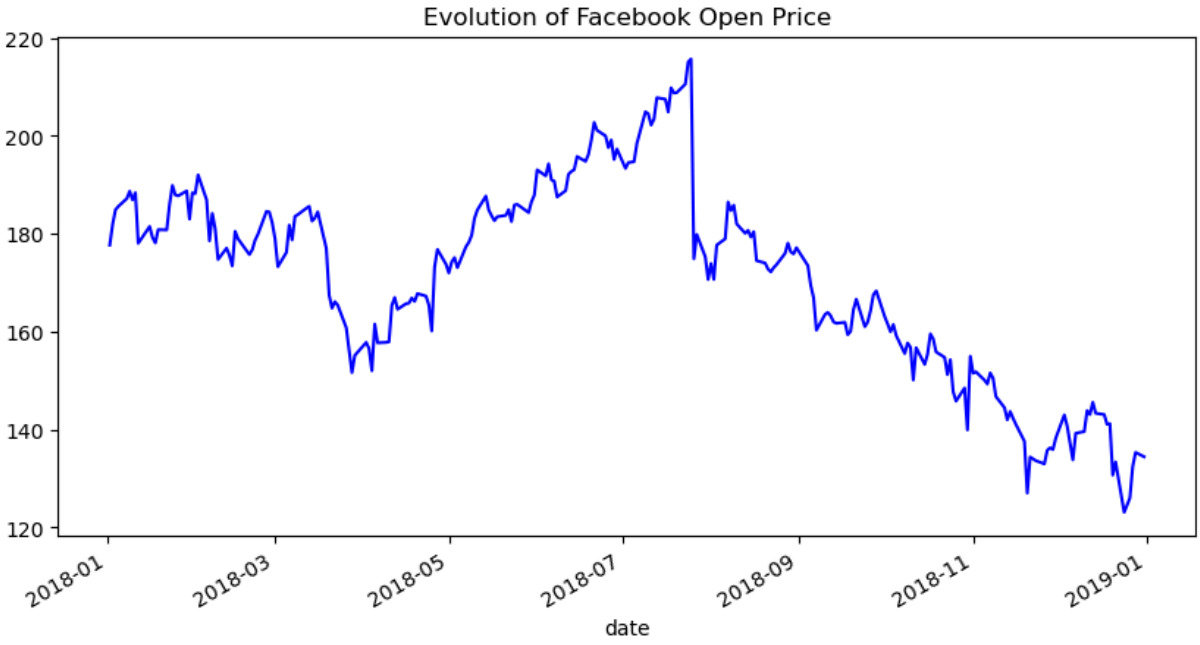
In [44]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

```
import pandas as pd
fb = pd.read_csv(
'fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
quakes = pd.read_csv('earthquakes.csv')
```

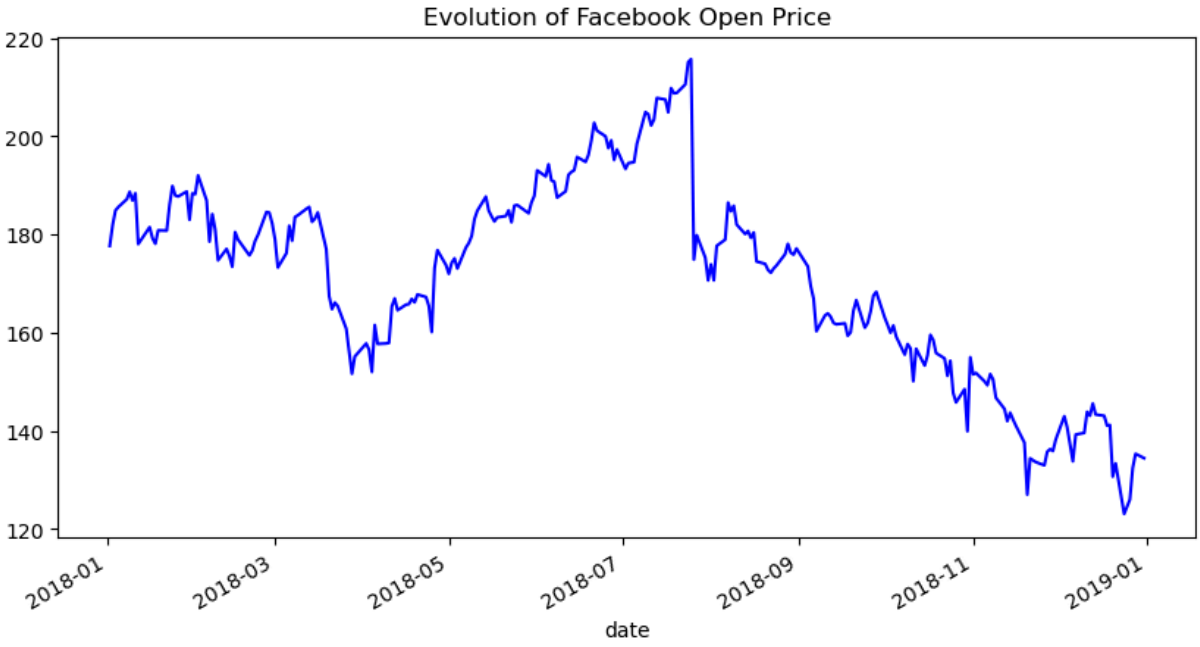
Evolution over time

In [46]:
```
fb.plot(
    kind='line',
    y='open',
    figsize=(10, 5),
    style='b-',
    legend=False,
    title='Evolution of Facebook Open Price'
)
plt.show()
```



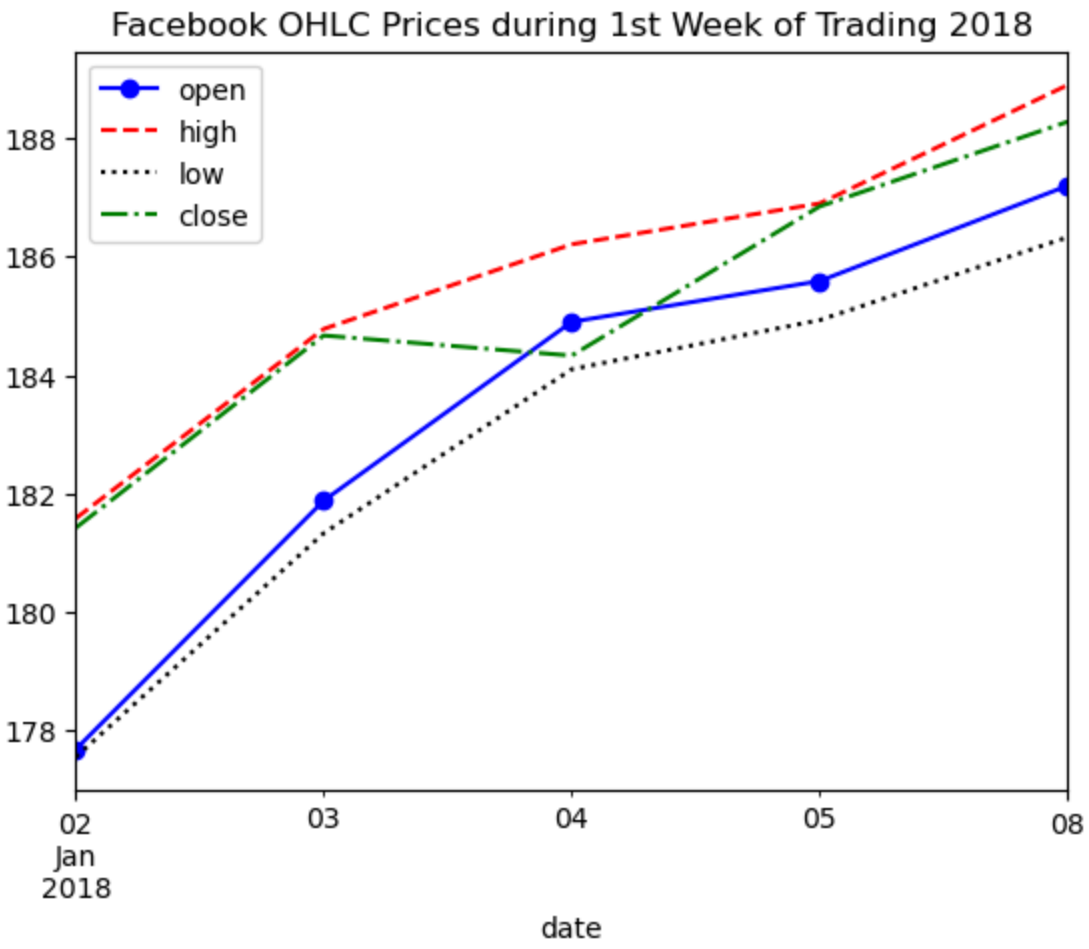


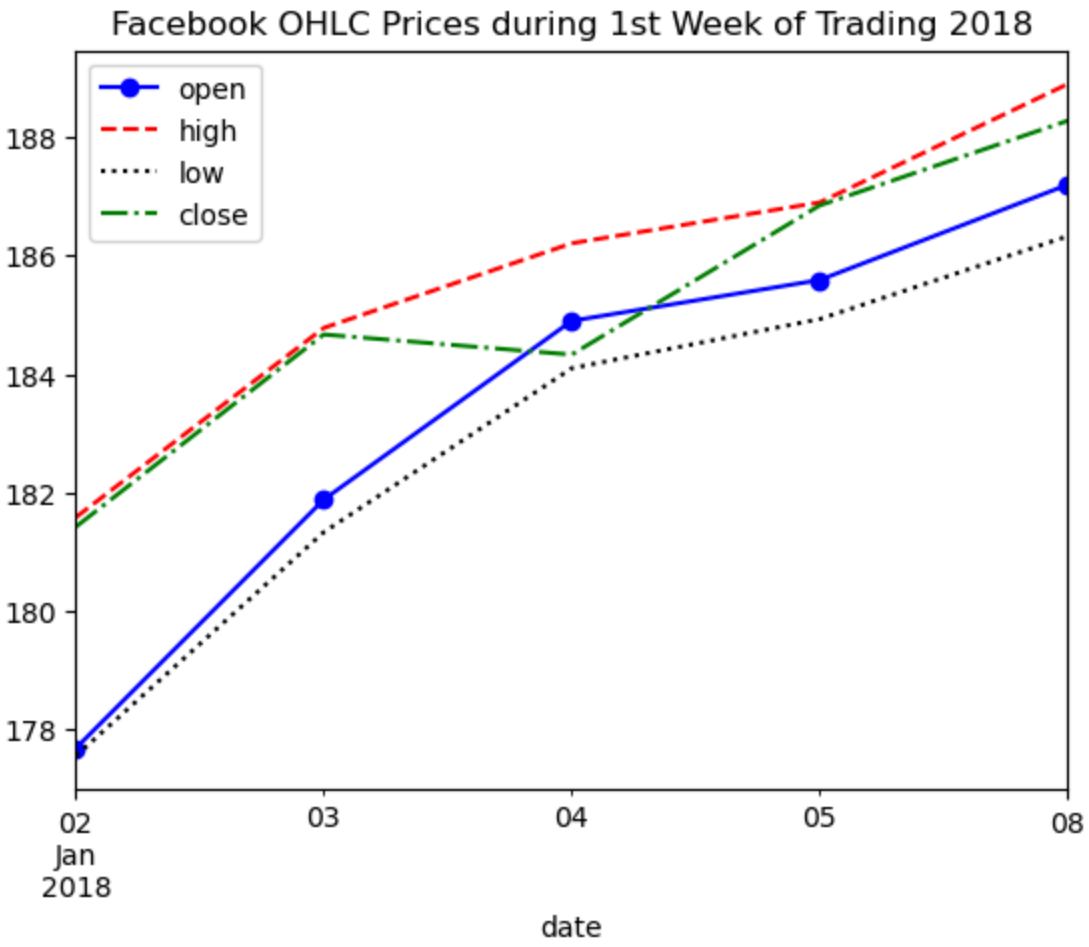
In [47]:
```
fb.plot(
    kind='line',
    y='open',
    figsize=(10, 5),
    color='blue',
    linestyle='solid',
    legend=False,
    title='Evolution of Facebook Open Price'
)
plt.show()
```



In [49]:
```
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()
```


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

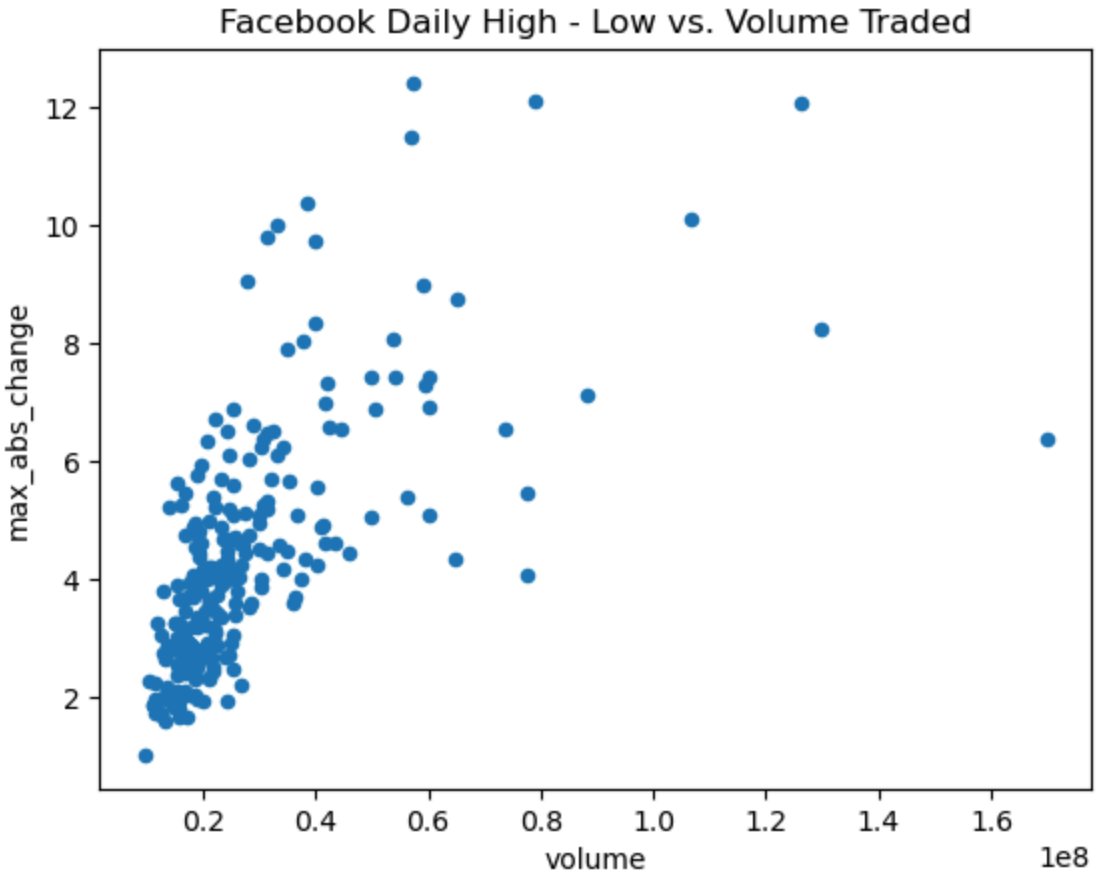


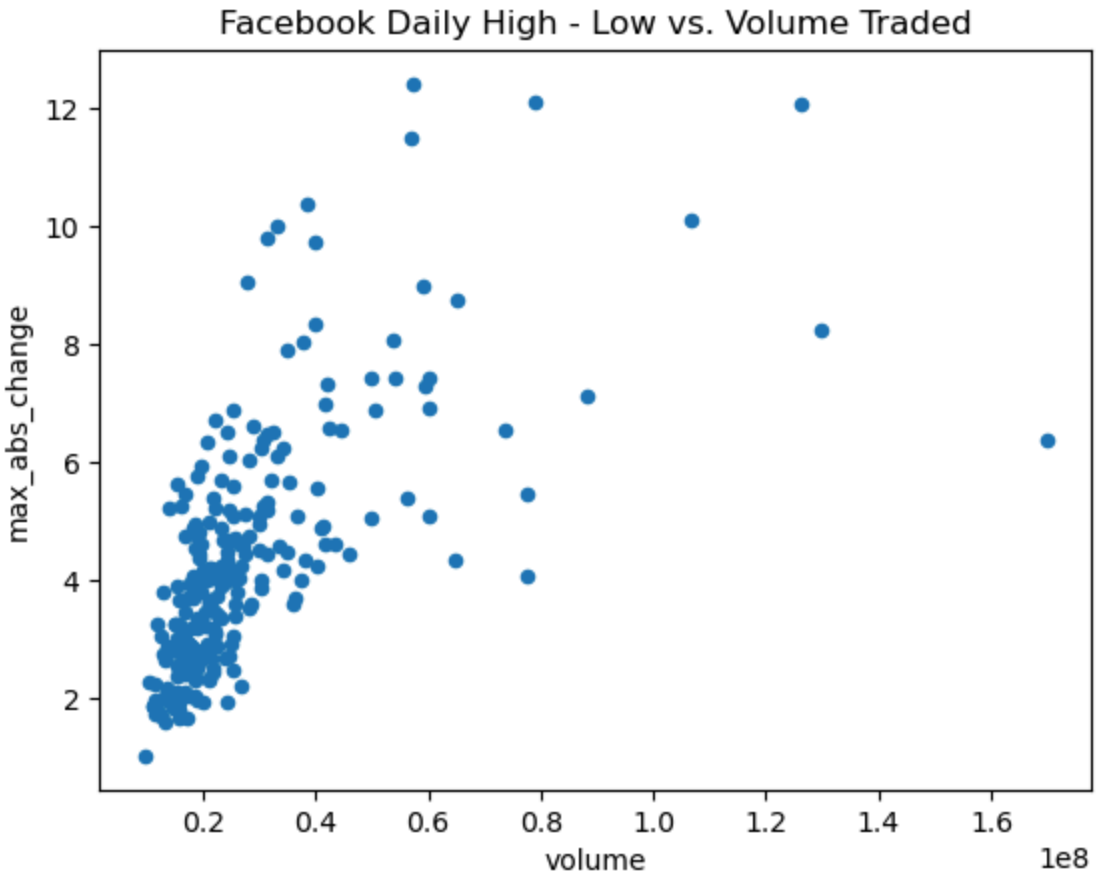
Creating subplots

In [50]:
```
fb.plot(
    kind='line',
    subplots=True,
    layout=(3,2),
    figsize=(15,10),
    title='Facebook Stock 2018'
)
plt.show()
```

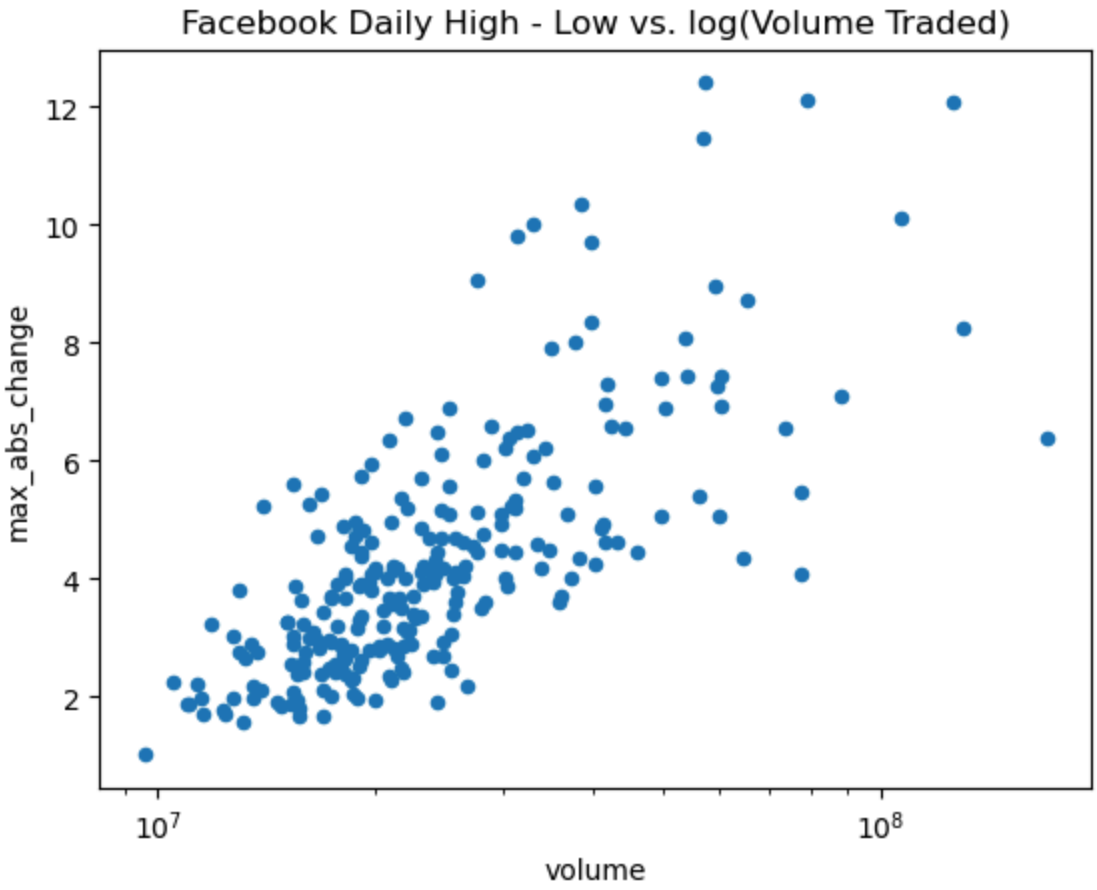
Scatter plots

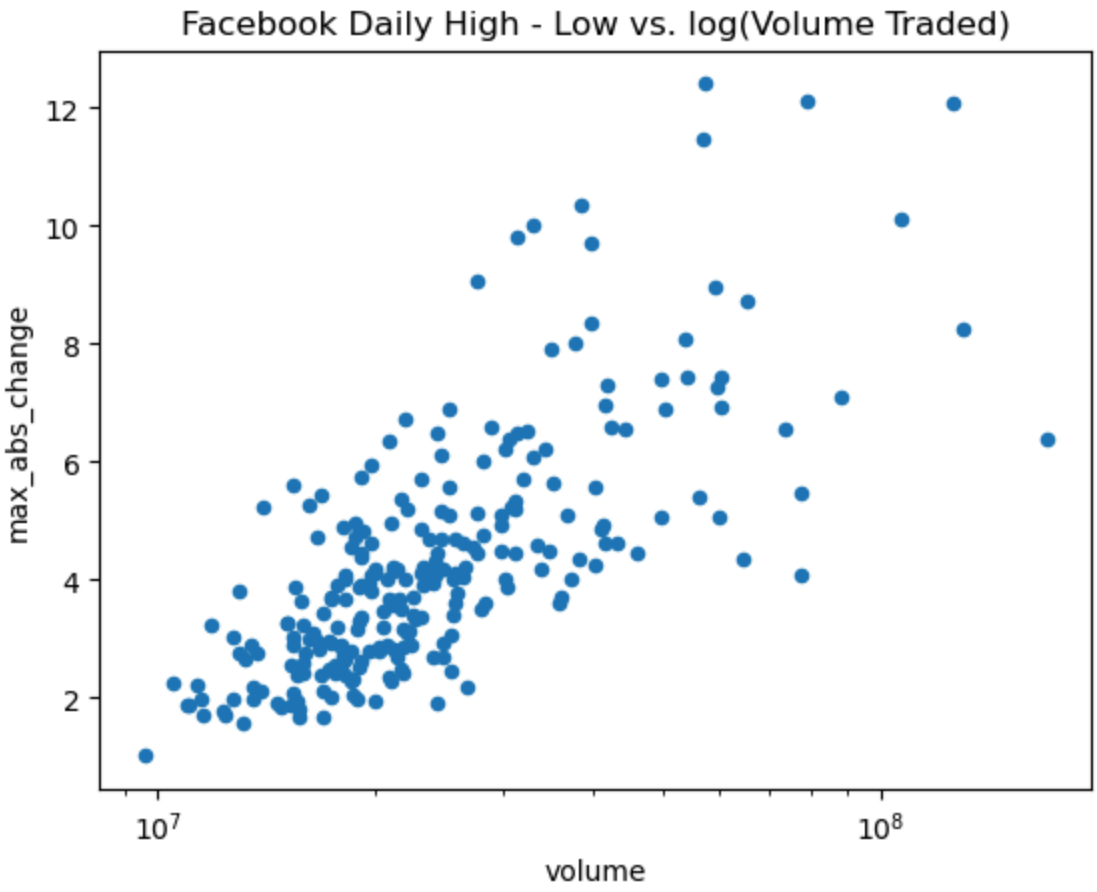
In [52]:
```
fb.assign(
    max_abs_change=fb.high - fb.low
).plot(
    kind='scatter', x='volume', y='max_abs_change',
    title='Facebook Daily High - Low vs. Volume Traded'
)
plt.show()
```


Facebook Daily High - Low vs. Volume Traded
max_abs_change
volume
1e8
2
4
6
8
10
12
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6


Facebook Daily High - Low vs. Volume Traded
max_abs_change
volume
1e8
2
4
6
8
10
12
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6

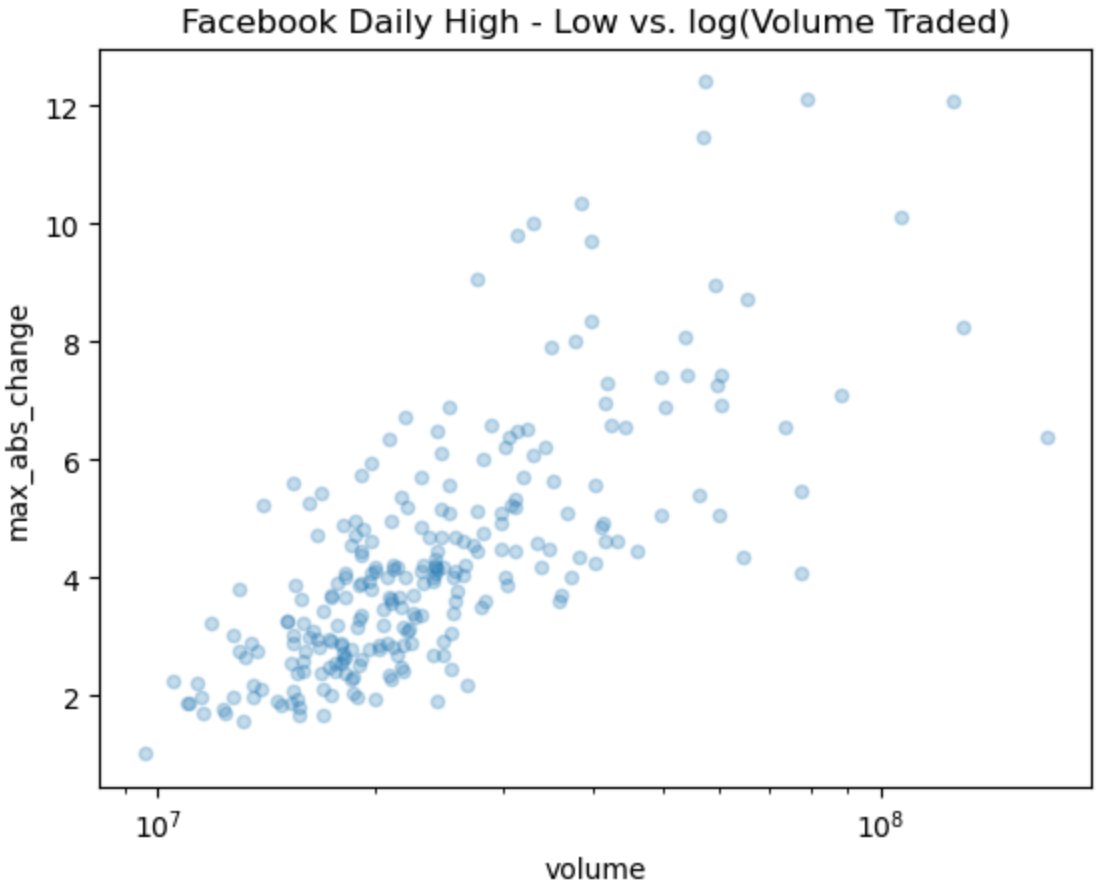
In [54]:
```
fb.assign(
    max_abs_change=fb.high - fb.low
).plot(
    kind='scatter', x='volume', y='max_abs_change',
    title='Facebook Daily High - Low vs. log(Volume Traded)',
    logx=True
)
plt.show()
```

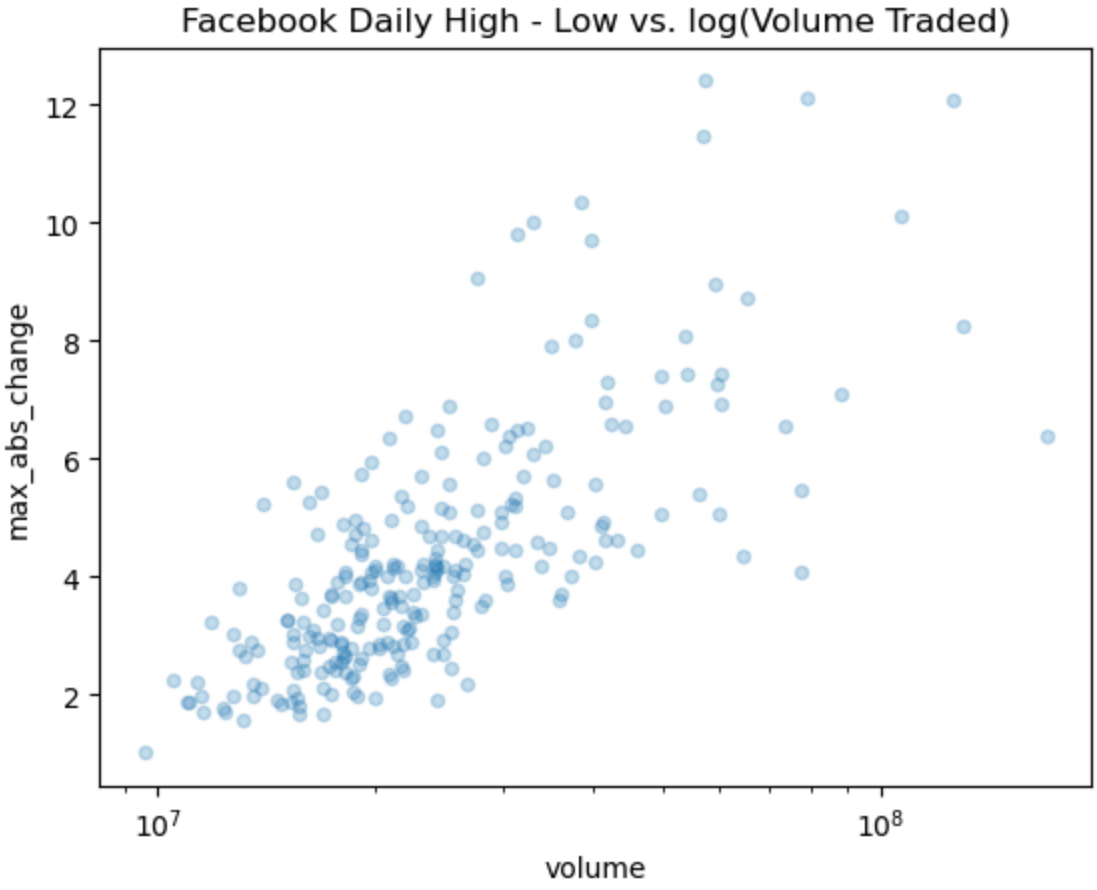




Adding Transparency to Plots with alpha

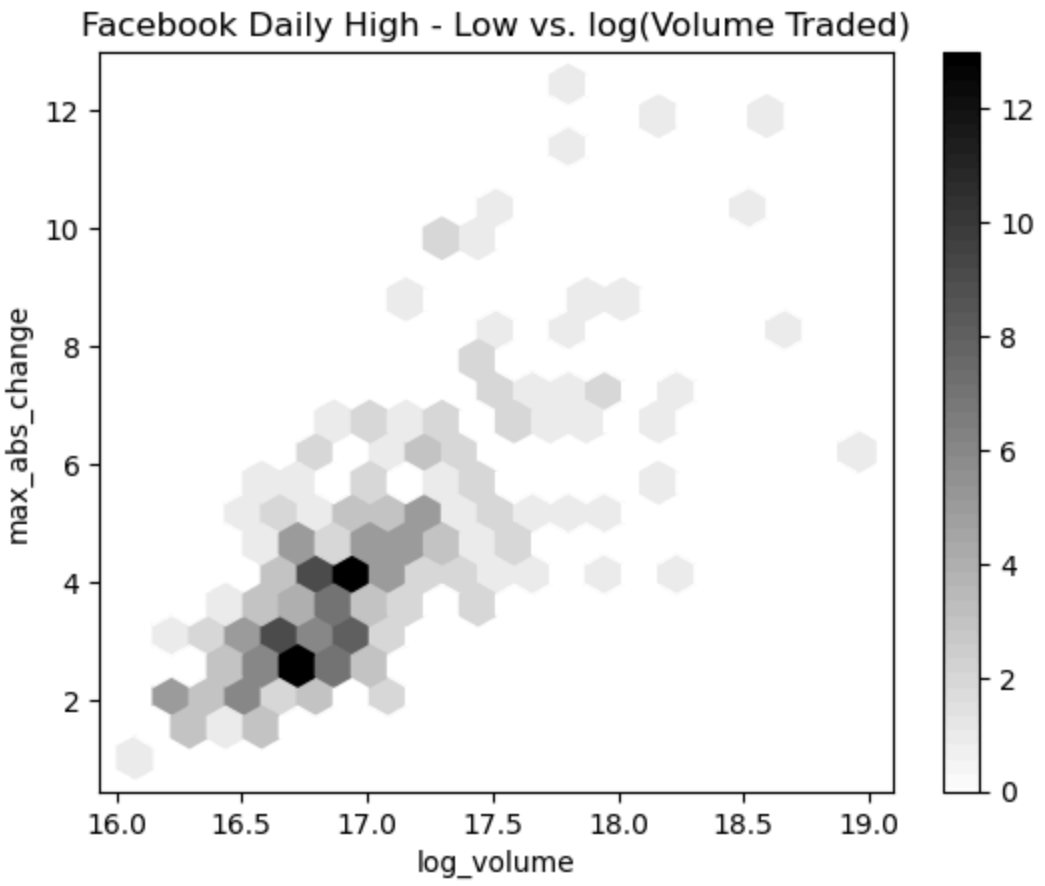
In [56]:
```
fb.assign(
    max_abs_change=fb.high - fb.low
).plot(
    kind='scatter', x='volume', y='max_abs_change',
    title='Facebook Daily High - Low vs. log(Volume Traded)',
    logx=True, alpha=0.25
)
plt.show()
```

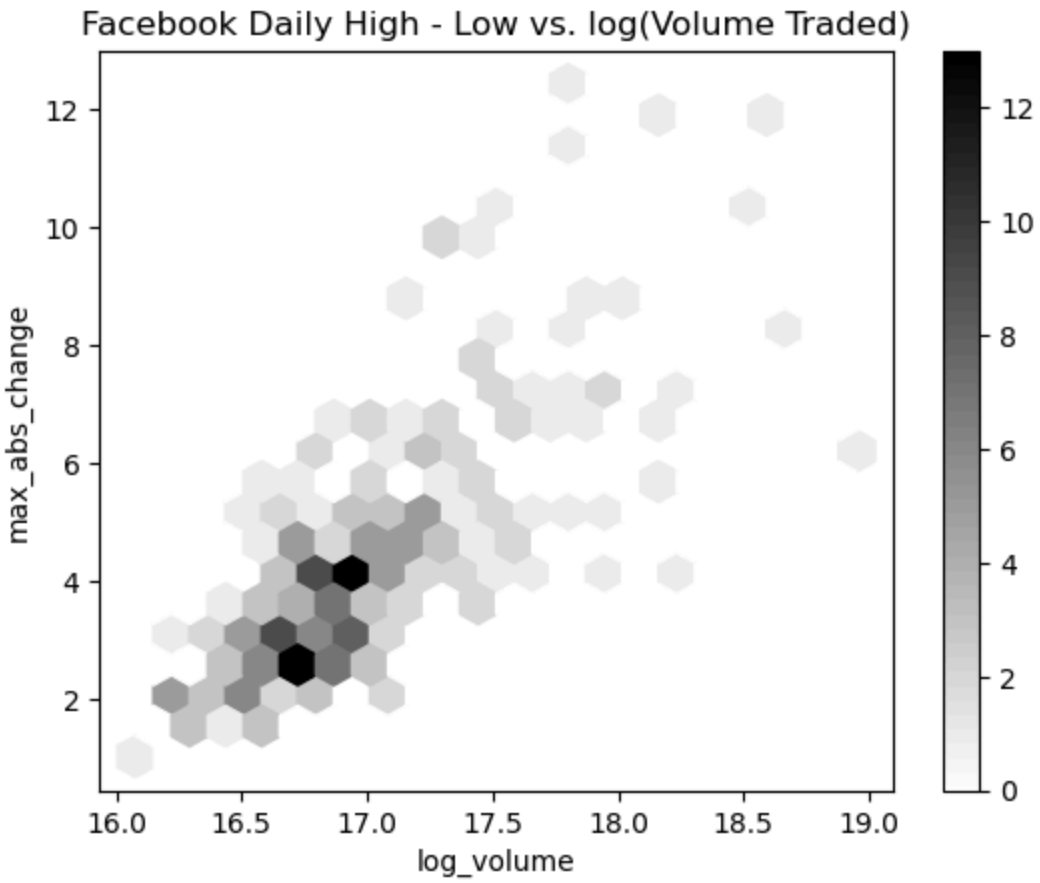

Facebook Daily High - Low vs. log(Volume Traded)
max_abs_change
volume
12
10
8
6
4
2
10^7
10^8


Facebook Daily High - Low vs. log(Volume Traded)
max_abs_change
volume
12
10
8
6
4
2
10^7
10^8

Hexbins

In [58]:
```
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 vers
)
plt.show()
```





Visualizing Correlations with Heatmaps

In [68]:
```
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')
fig.colorbar(im).set_clim(-1, 1)
labels = [col.lower() for col in fb_corr.columns]
ax.set_xticklabels([''] + labels, rotation=45)
ax.set_yticklabels([''] + labels)
plt.show
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[68], line 7
      2 fb_corr = fb.assign(
      3     log_volume=np.log(fb.volume),
      4     max_abs_change=fb.high - fb.low
      5 ).corr()
      6 im = ax.matshow(fb_corr, cmap='seismic')
----> 7 fig.colorbar(im).set_clim(-1, 1)
      8 labels = [col.lower() for col in fb_corr.columns]
      9 ax.set_xticklabels([''] + labels, rotation=45)

AttributeError: 'Colorbar' object has no attribute 'set_clim'
```

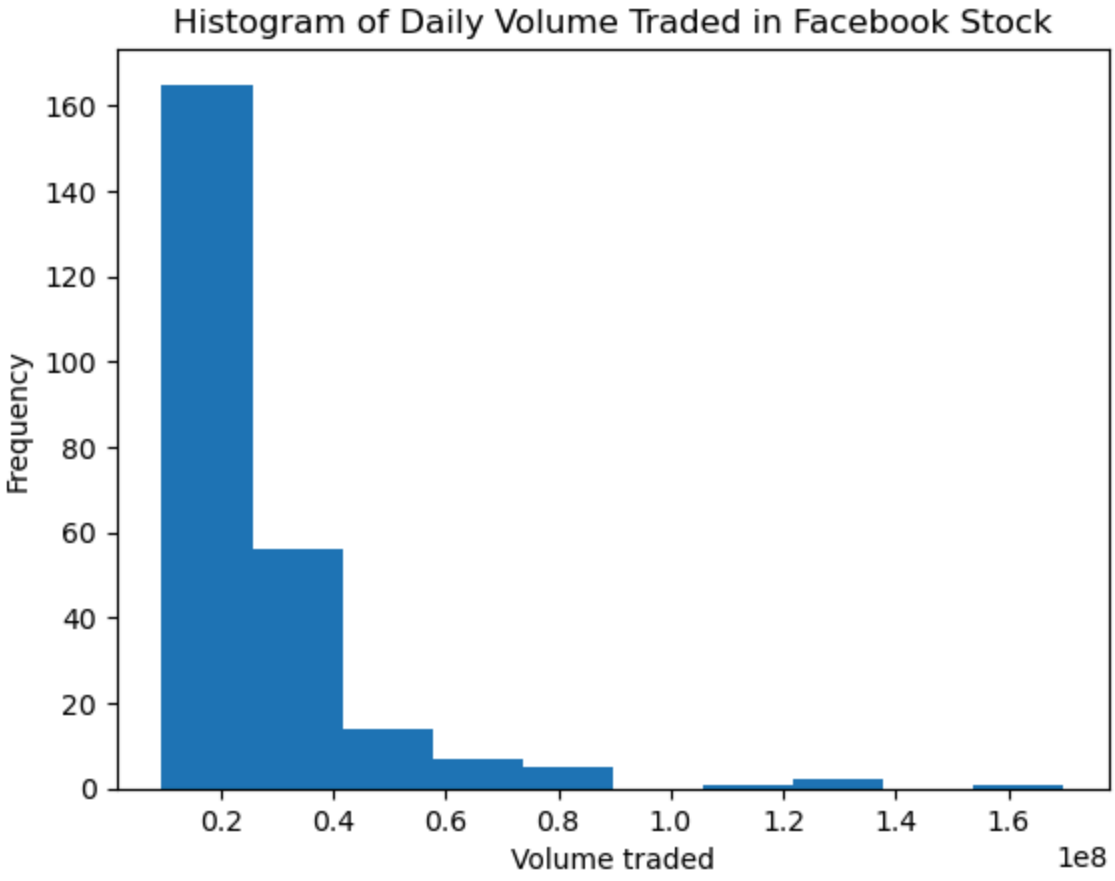
In [64]:
```
fb_corr.loc['max_abs_change', ['volume', 'log_volume']]
```

Out[64]:
```
volume        0.642027
log_volume    0.731542
Name: max_abs_change, dtype: float64
```

Visualizing distributions

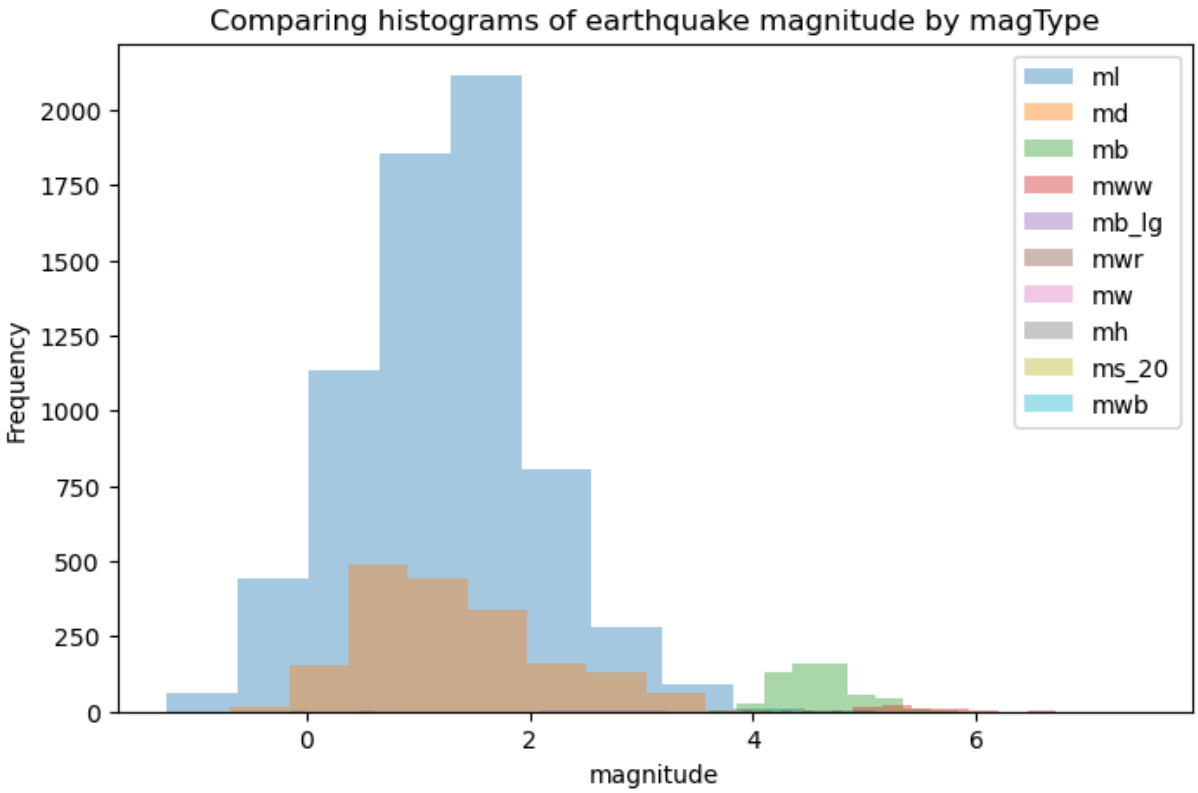
Histograms

In [67]:
```
fb.volume.plot(
    kind='hist',
    title='Histogram of Daily Volume Traded in Facebook Stock'
)
plt.xlabel('Volume traded') # label the x-axis (discussed in chapter 6)
plt.show()
```



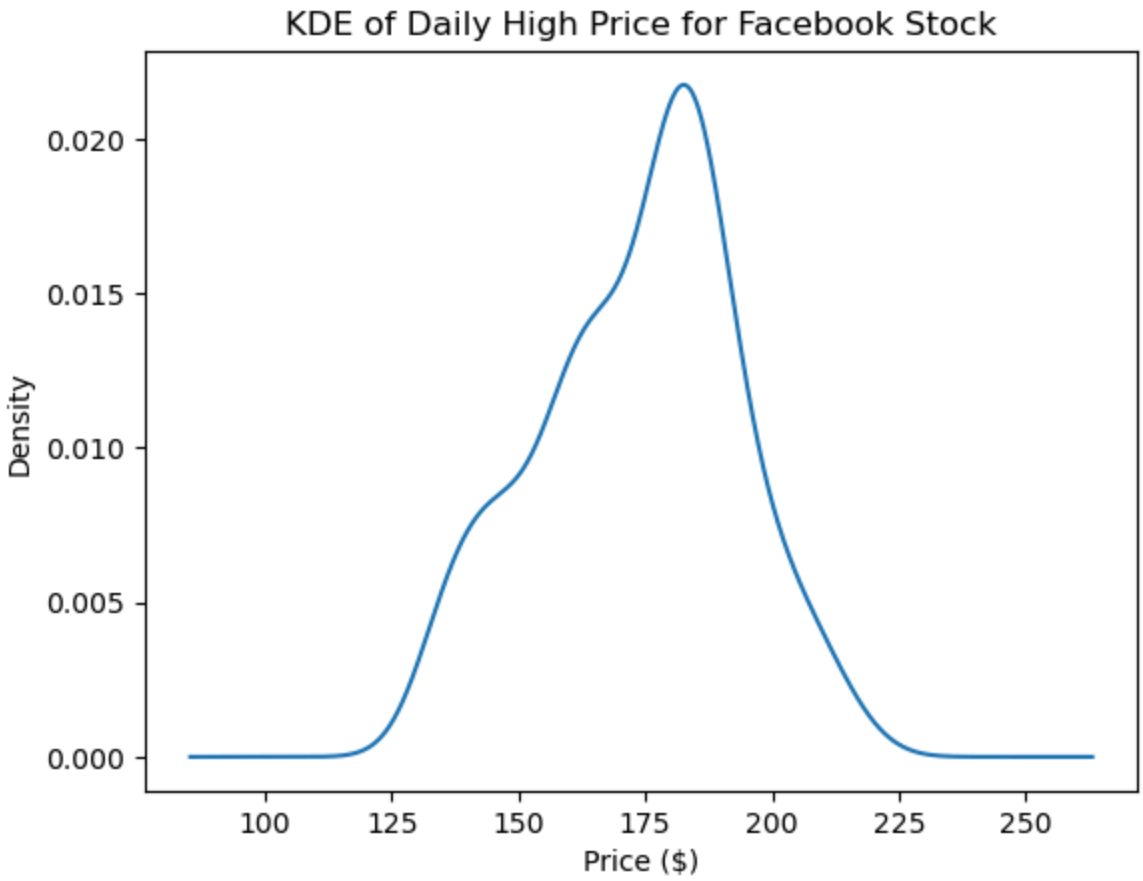
In [72]:
```
fig, axes = plt.subplots(figsize=(8, 5))
for magtype in quakes.magType.unique():
    data = quakes.query(f'magType == "{magtype}"').mag
    if not data.empty:
        data.plot(
            kind='hist', ax=axes, alpha=0.4,
            label=magtype, legend=True,
            title='Comparing histograms of earthquake magnitude by magType'
)
```

```
plt.xlabel('magnitude') # label the x-axis (discussed in chapter 6)
plt.show()
```



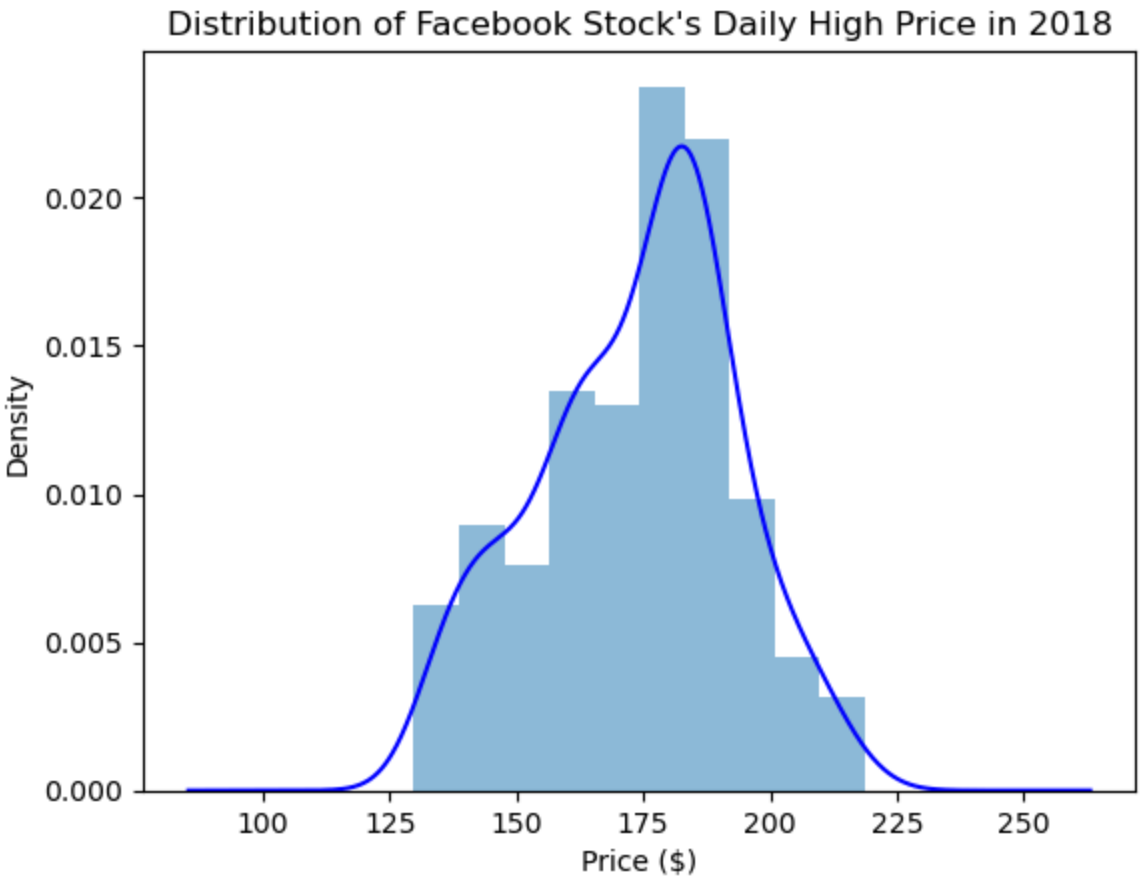
Kernel Density Estimation (KDE)

In [73]:
```
fb.high.plot(
    kind='kde',
    title='KDE of Daily High Price for Facebook Stock'
)
plt.xlabel('Price ($)') # label the x-axis (discussed in chapter 6)
plt.show()
```



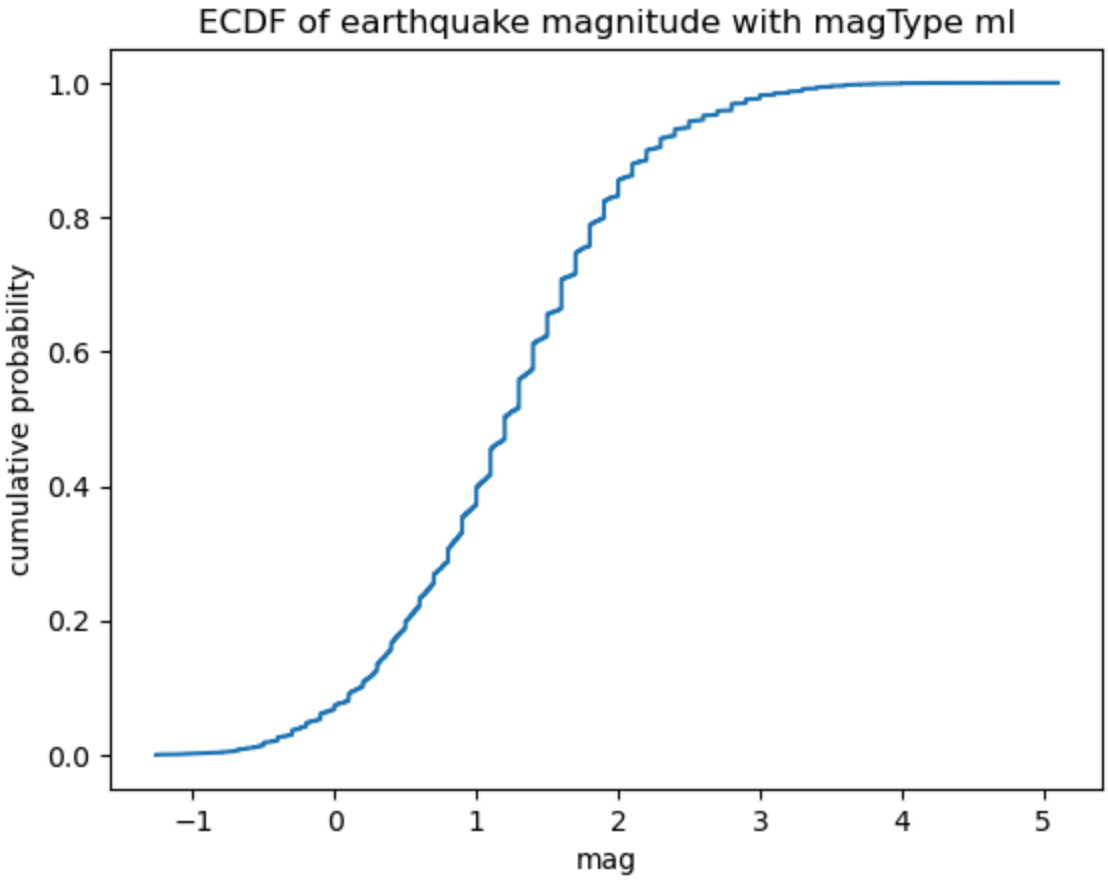
Adding to the result of plot()

In [76]:
```
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 (discussed in chapter 6)Out[15]:
plt.show()
```

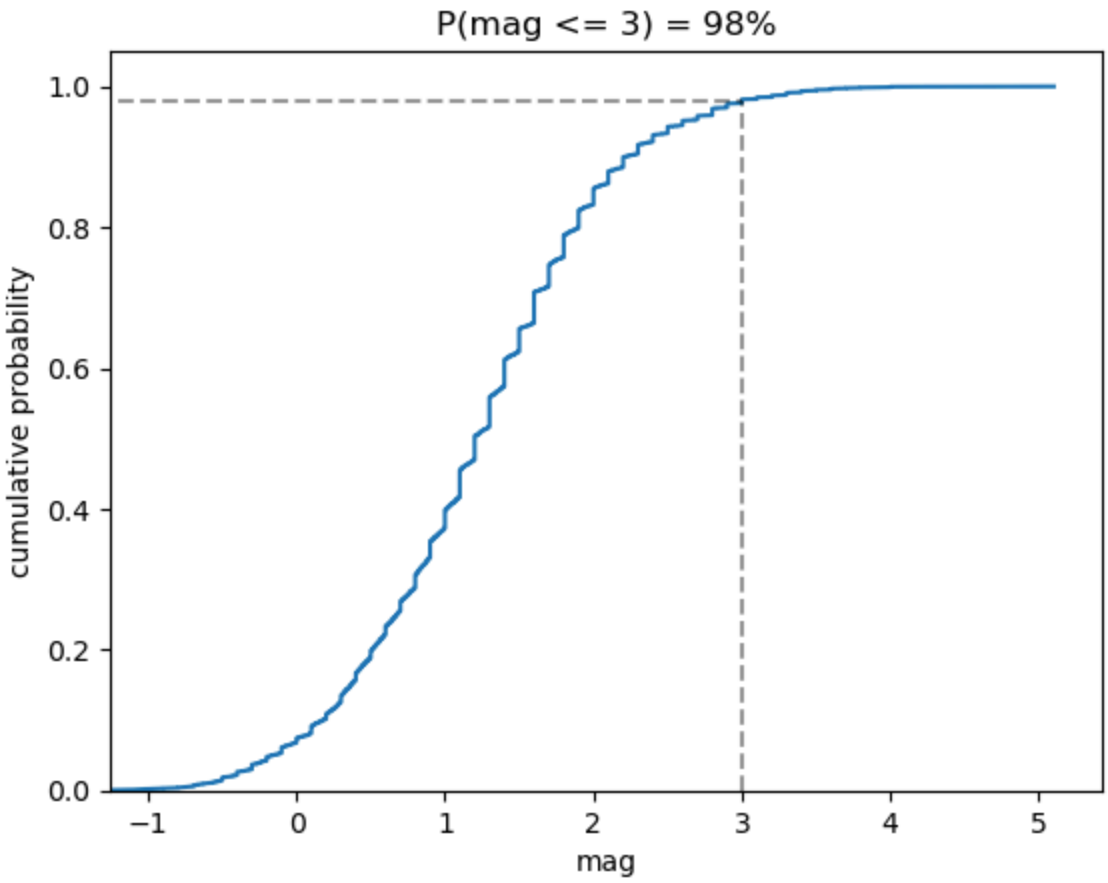


Plotting the ECDF

In [79]:
```
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')
plt.show()
```

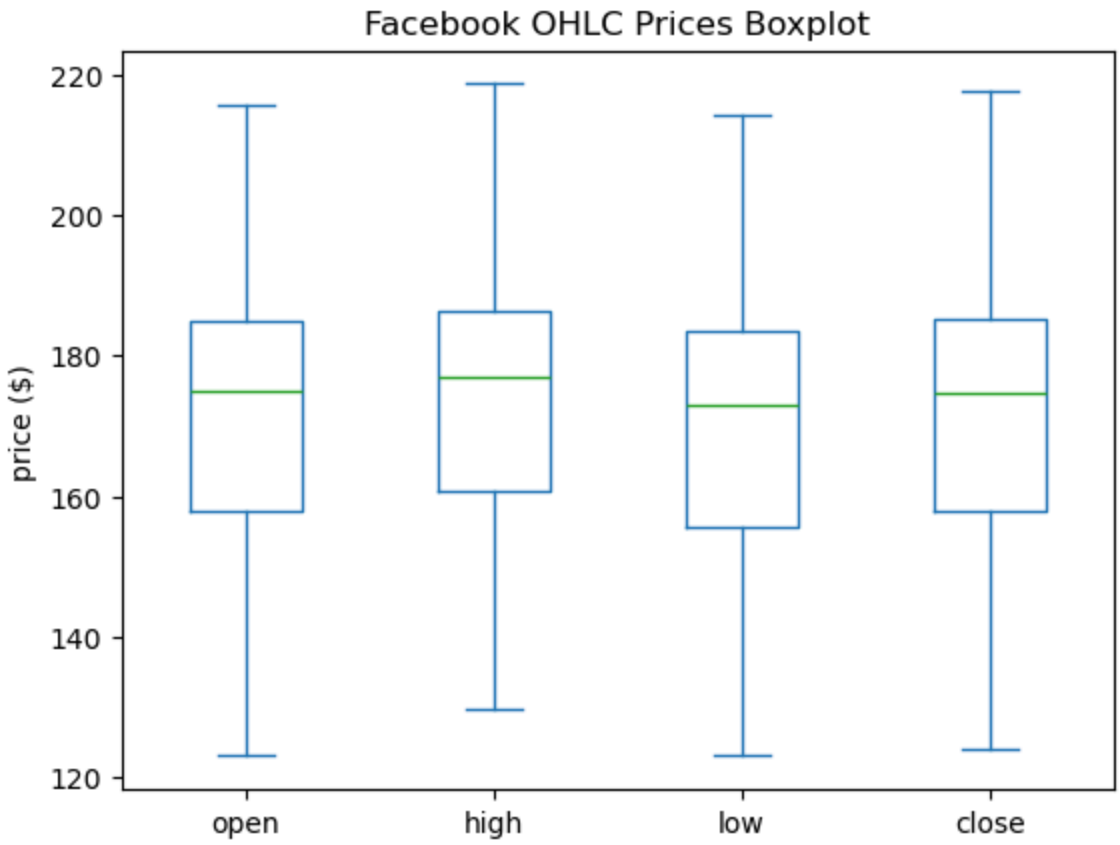


In [80]:
```
from statsmodels.distributions.empirical_distribution import ECDF
ecdf = ECDF(quakes.query('magType == "ml"').mag)
plt.plot(ecdf.x, ecdf.y)
# formatting below will all be covered in chapter 6
# axis labels
plt.xlabel('mag') # add x-axis label
plt.ylabel('cumulative probability') # add y-axis 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%')
plt.show()
```



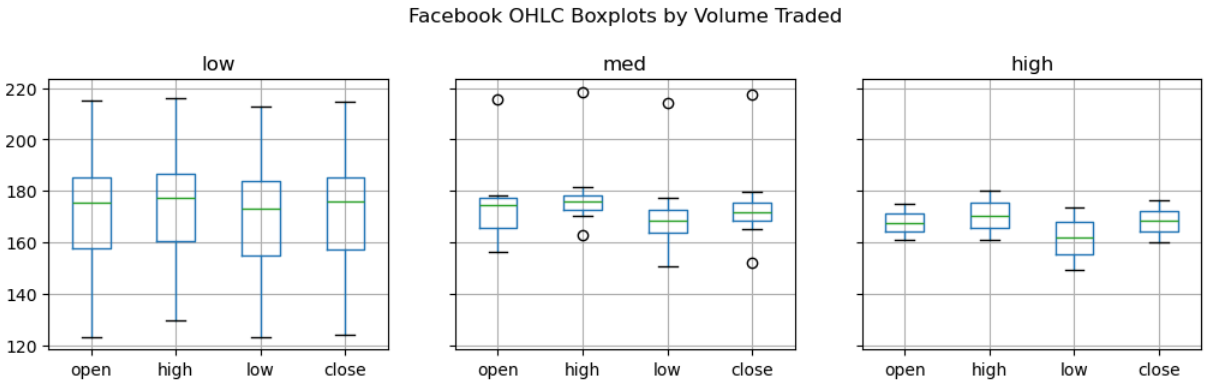
Box plots

In [82]:
```
fb.iloc[:,:4].plot(kind='box', title='Facebook OHLC Prices Boxplot')
plt.ylabel('price ($)') # label the x-axis (discussed in chapter 6)
plt.show()
```



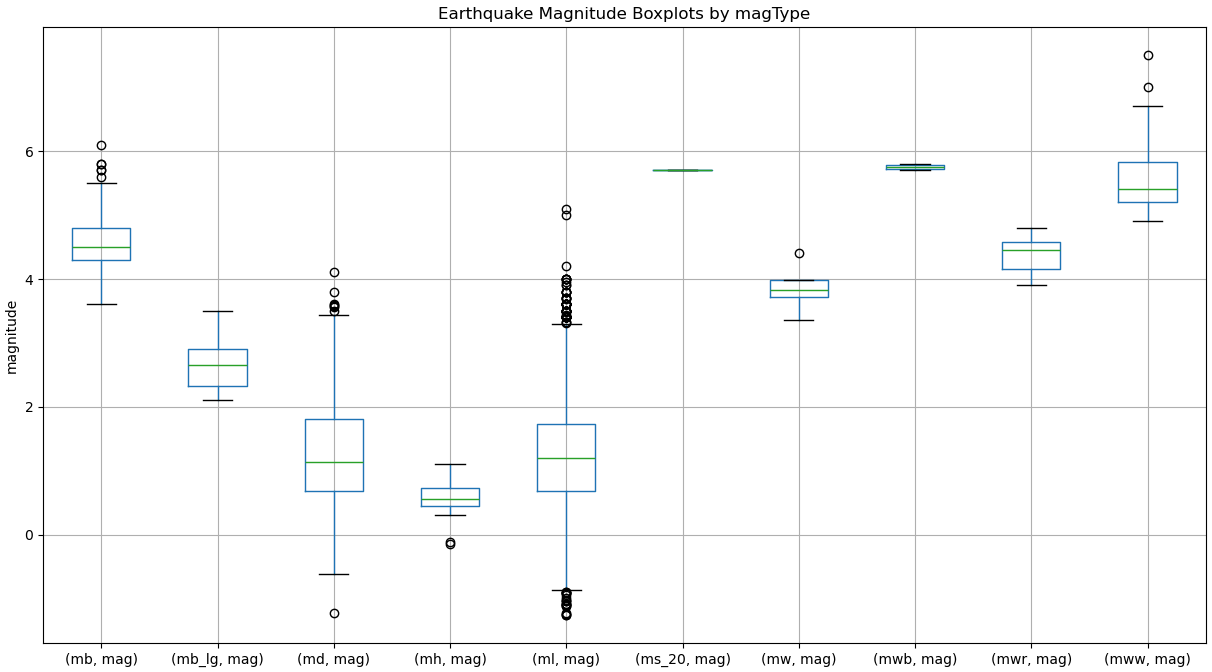
In [84]:
```
fb.assign(
    volume_bin=pd.cut(fb.volume, 3, labels=['low', 'med', 'high'])
).groupby('volume_bin').boxplot(
    column=['open', 'high', 'low', 'close'],
    layout=(1, 3), figsize=(12, 3)
)
plt.suptitle('Facebook OHLC Boxplots by Volume Traded', y=1.1)
plt.show()
```

```
C:\Users\Personal Computer\AppData\Local\Temp\ipykernel_8912\342345630.py:3: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  ).groupby('volume_bin').boxplot(
```



In [88]:
```
quakes[['mag', 'magType']].groupby('magType').boxplot(
    figsize=(15, 8), subplots=False
)
```

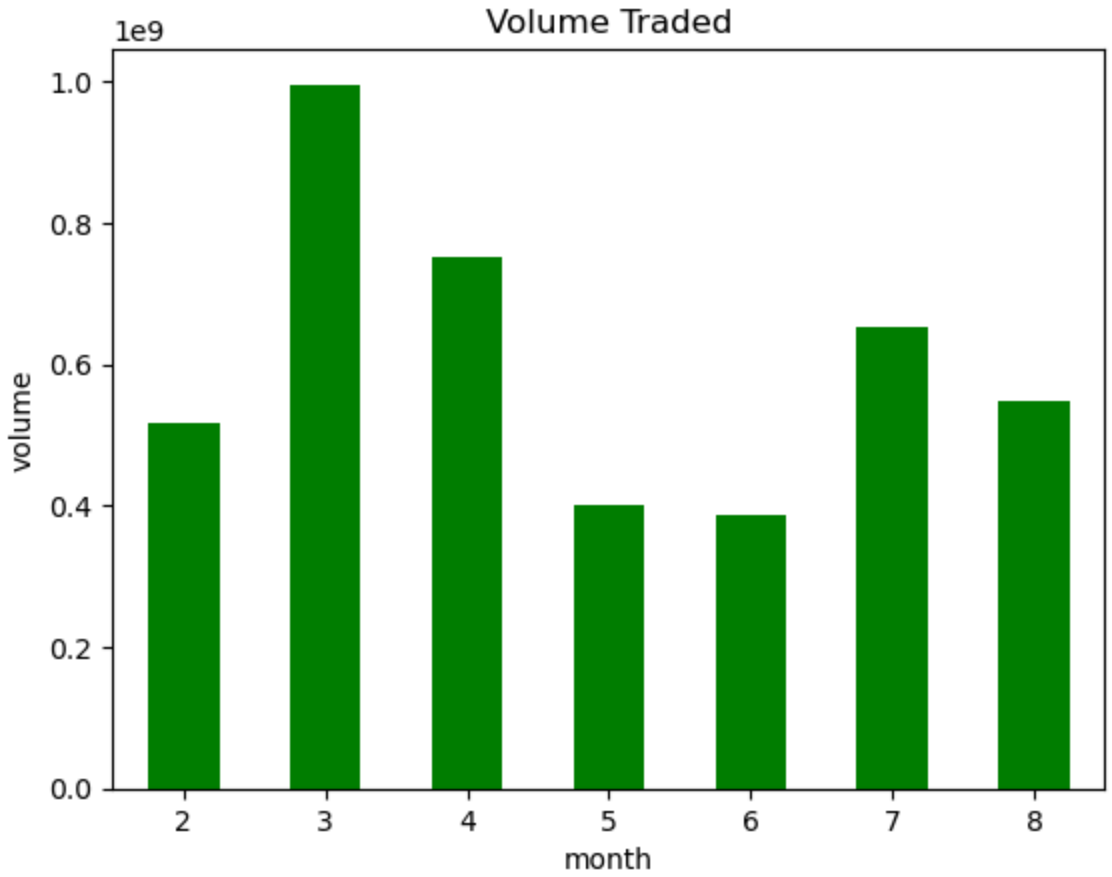
```
plt.title('Earthquake Magnitude Boxplots by magType')
plt.ylabel('magnitude') # label the y-axis (discussed in chapter 6)
plt.show()
```



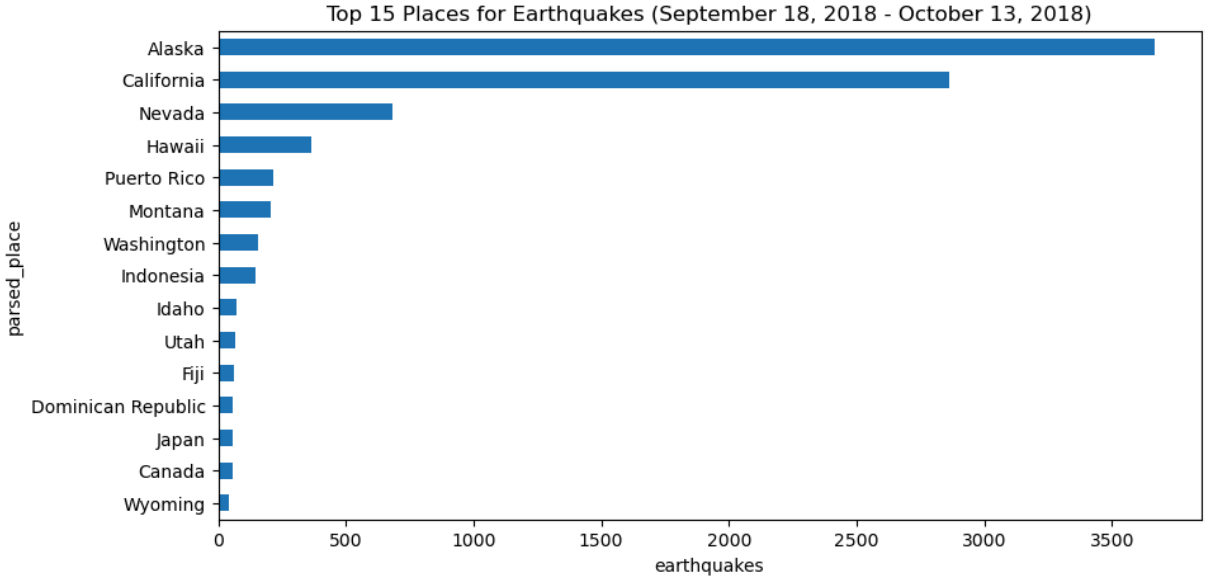
Counts and frequencies

Bar charts

In [90]:
```
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.ylabel('volume') # label the y-axis (discussed in chapter 6)
plt.show()
```

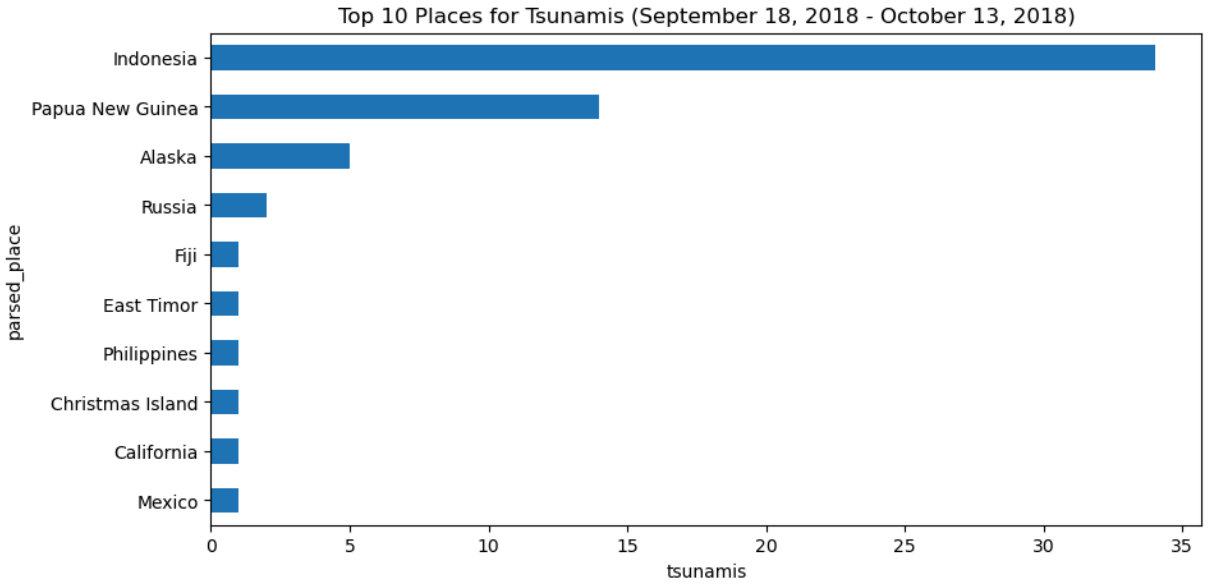


In [92]:
```
quakes.parsed_place.value_counts().iloc[14::-1,].plot(
    kind='barh', figsize=(10, 5),
    title='Top 15 Places for Earthquakes '\
'(September 18, 2018 - October 13, 2018)'
)
plt.xlabel('earthquakes') # label the x-axis (discussed in chapter 6)
plt.show()
```

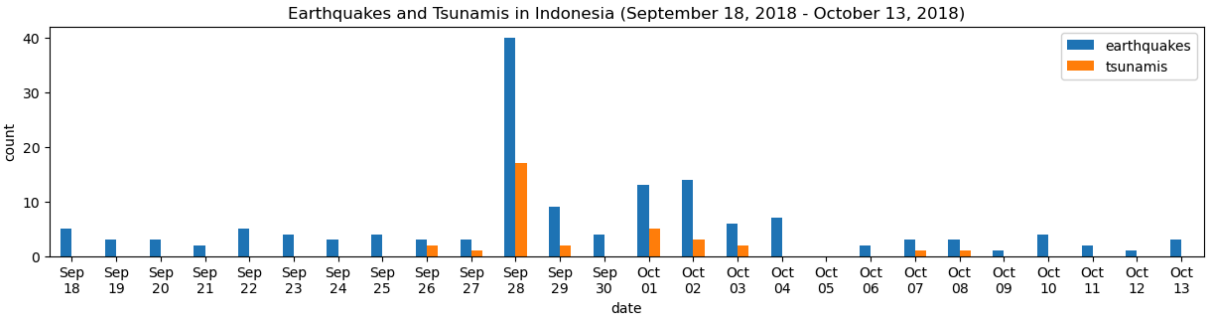


In [93]:
```
quakes.groupby('parsed_place').tsunami.sum().sort_values().iloc[-10::,].plot(
    kind='barh', figsize=(10, 5),
```

```
    title='Top 10 Places for Tsunamis '\
'(September 18, 2018 - October 13, 2018)'
)
plt.xlabel('tsunamis') # label the x-axis (discussed in chapter 6)
plt.show()
```

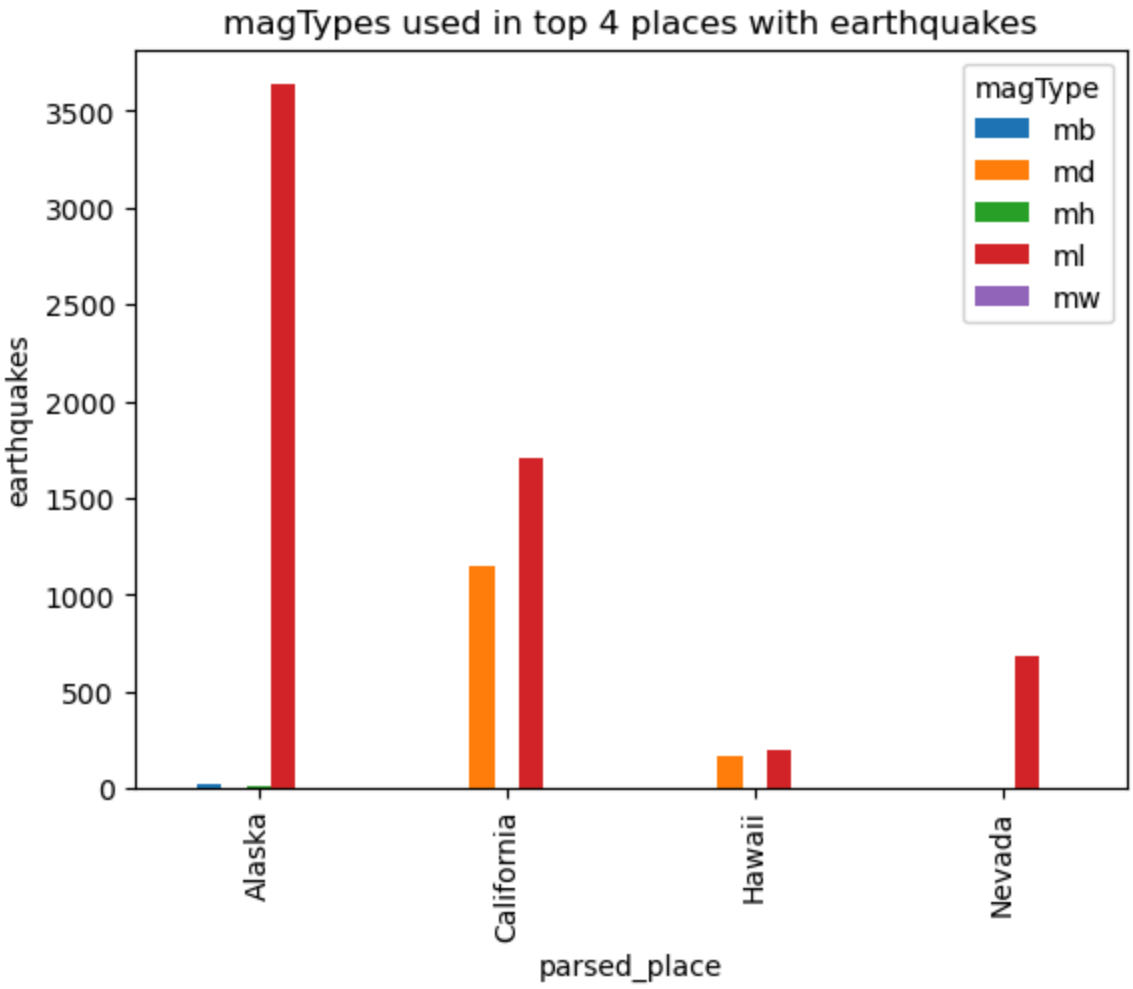


In [96]:
```
indonesia_quakes = quakes.query('parsed_place == "Indonesia"').assign(
    time=lambda x: pd.to_datetime(x.time, unit='ms'),
    earthquake=1
).set_index('time').resample('1D').sum()
indonesia_quakes.index = indonesia_quakes.index.strftime('%b\n%d')
indonesia_quakes.plot(
y=['earthquake', '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 (discussed in chapter 6)
plt.xlabel('date')
plt.ylabel('count')
plt.show()
```



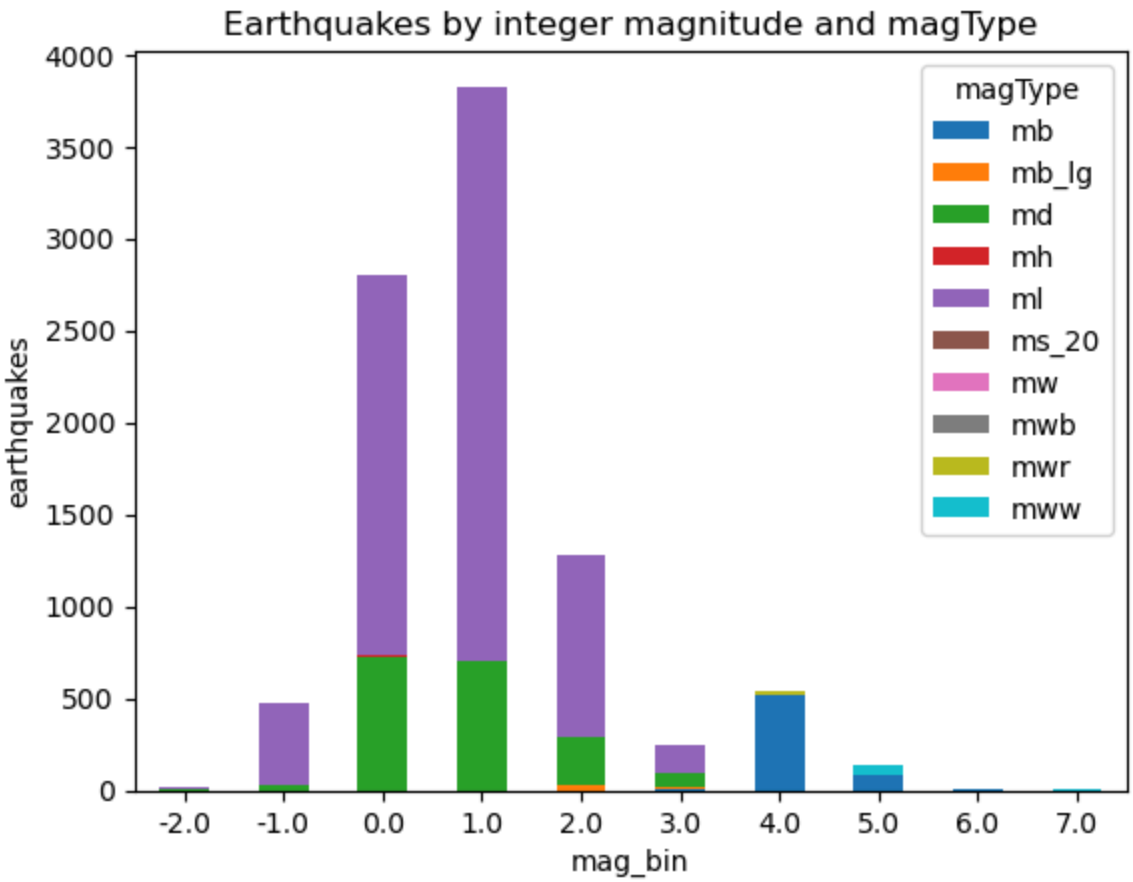
In [97]:
```
quakes[
    quakes.parsed_place.isin(['California', 'Alaska', 'Nevada', 'Hawaii'])
].groupby(['parsed_place', 'magType']).mag.count().unstack().plot.bar(
    title='magTypes used in top 4 places with earthquakes'
)
```

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



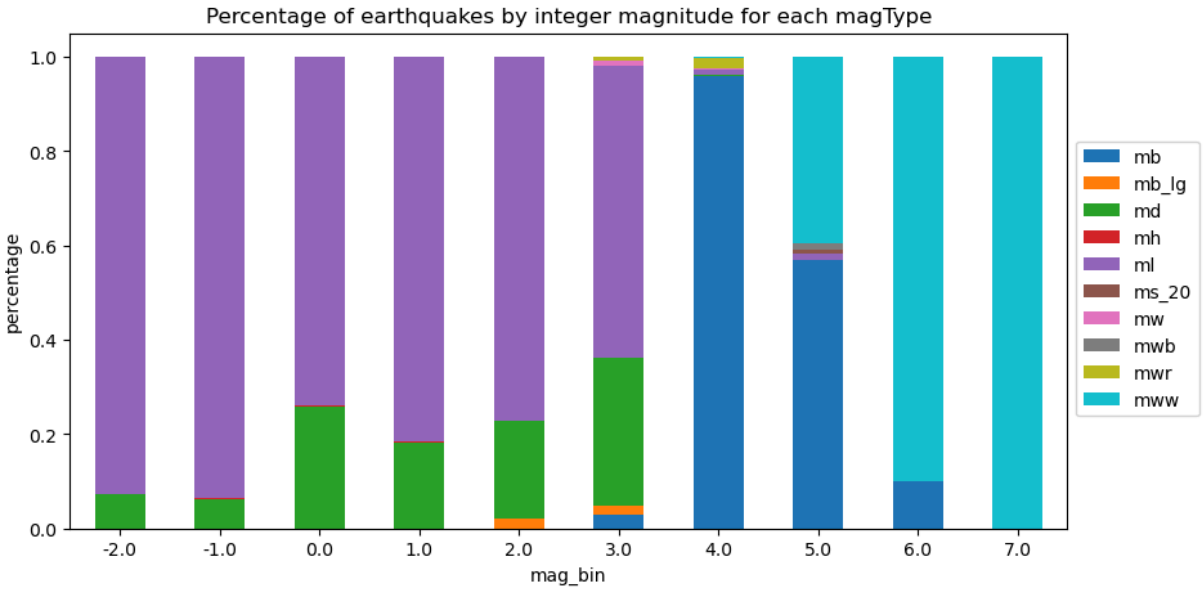
Stacked bar chart

In [98]:
```
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)
plt.show()
```



Normalized stacked bars

In [99]:
```
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)
plt.show()
```



9.3 Pandas Plotting Subpackage

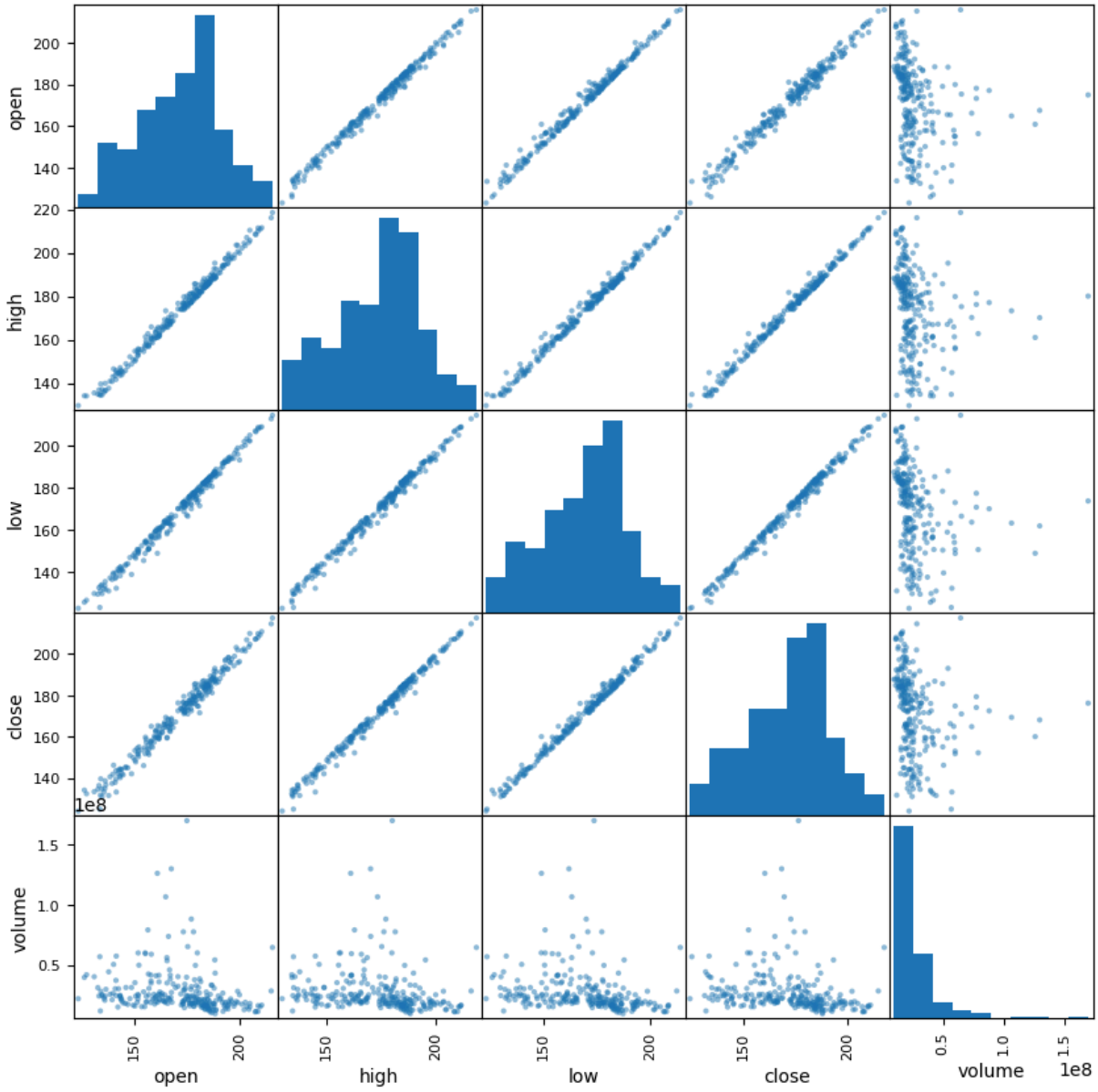
In [101…

```
%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
)
```

Scatter matrix

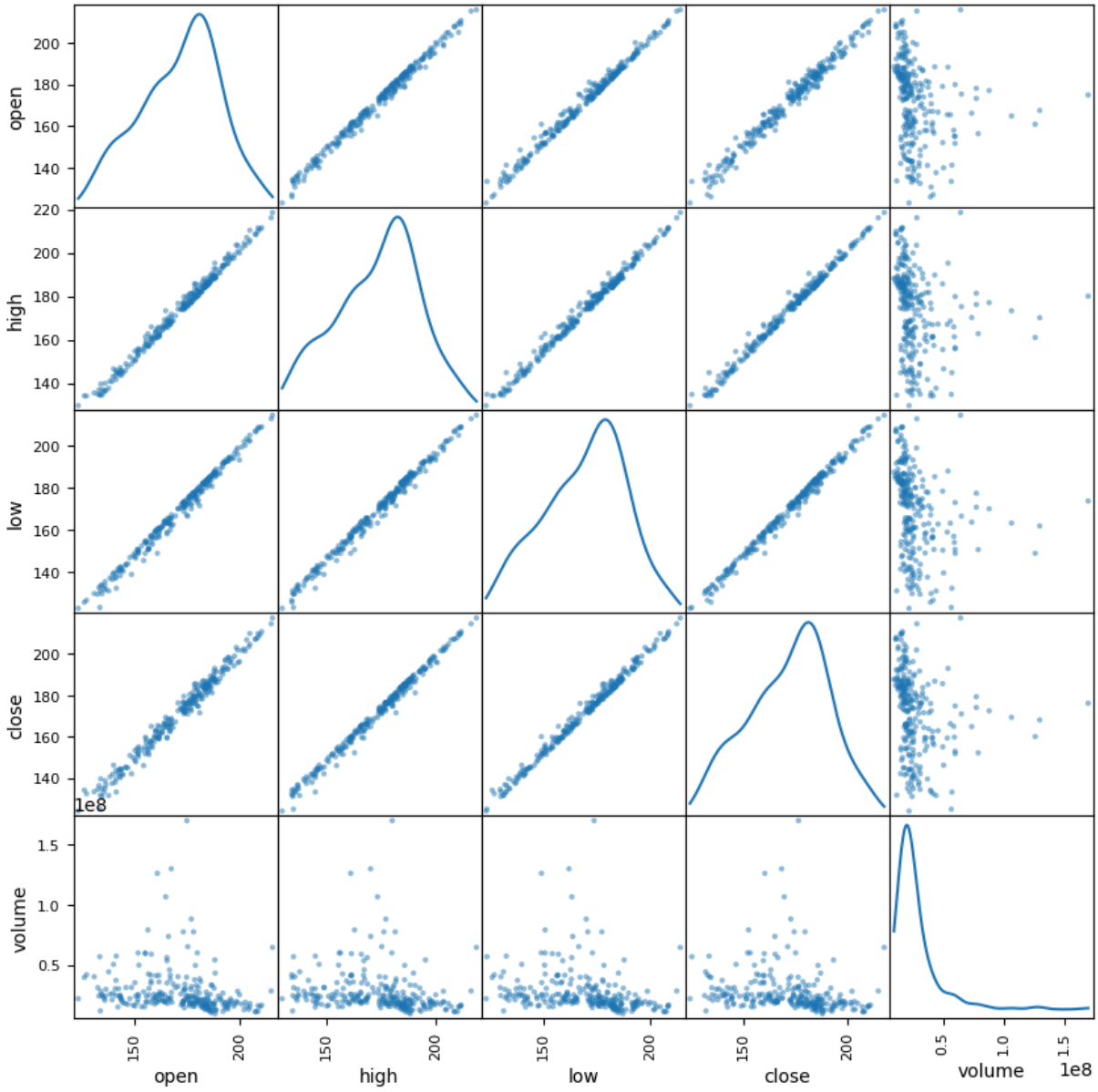
In [102…

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



In [103…

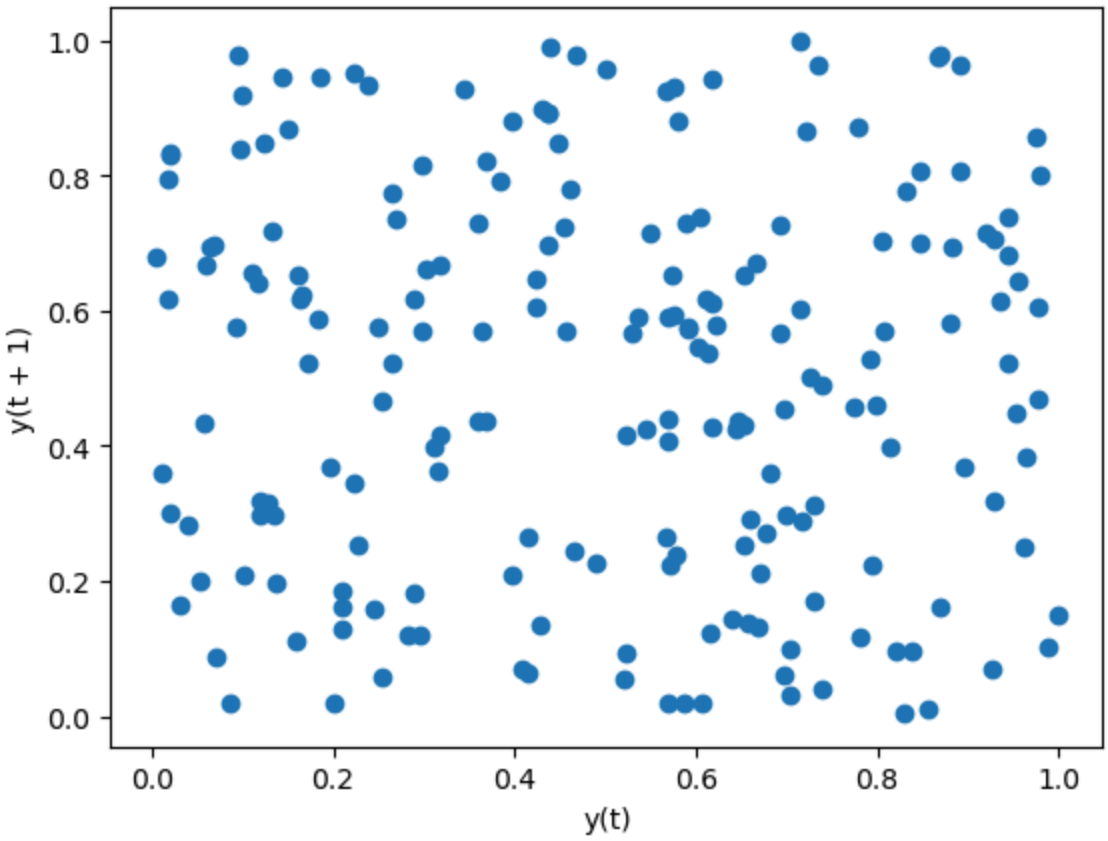
```
scatter_matrix(fb, figsize=(10, 10), diagonal='kde')
plt.show()
```



Lag plot

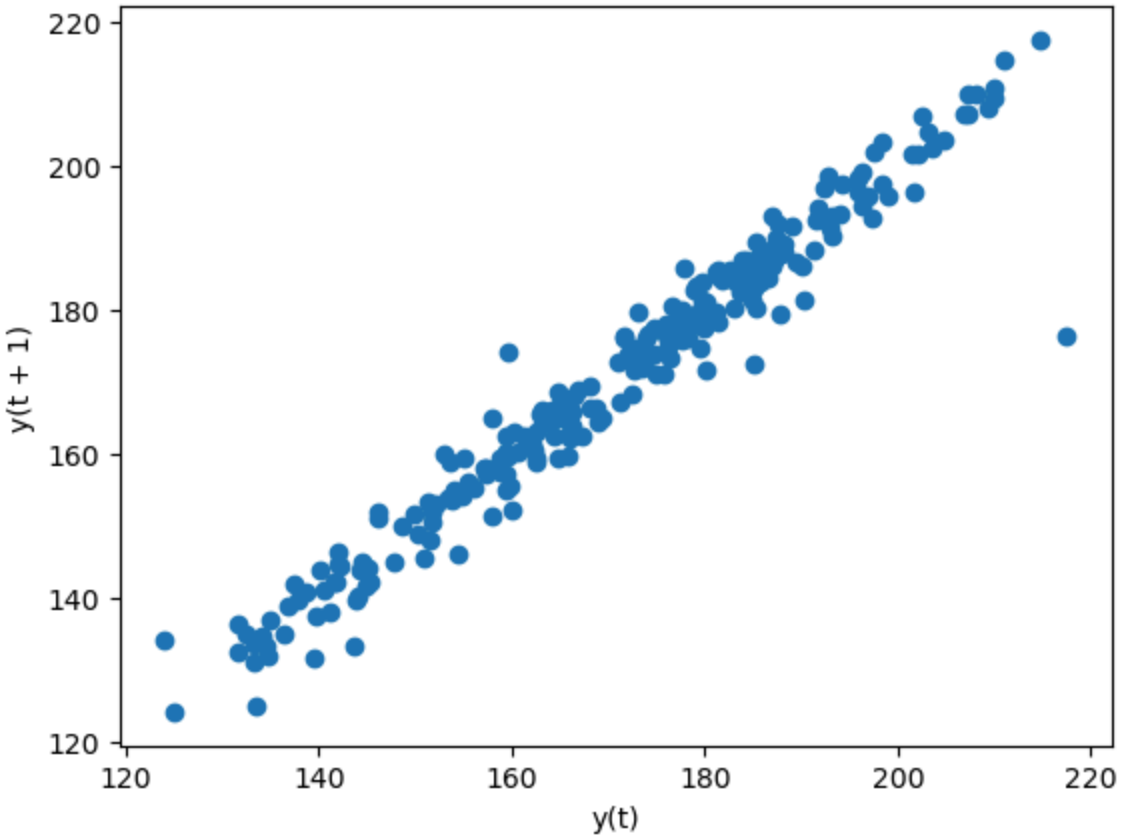
In [104…

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

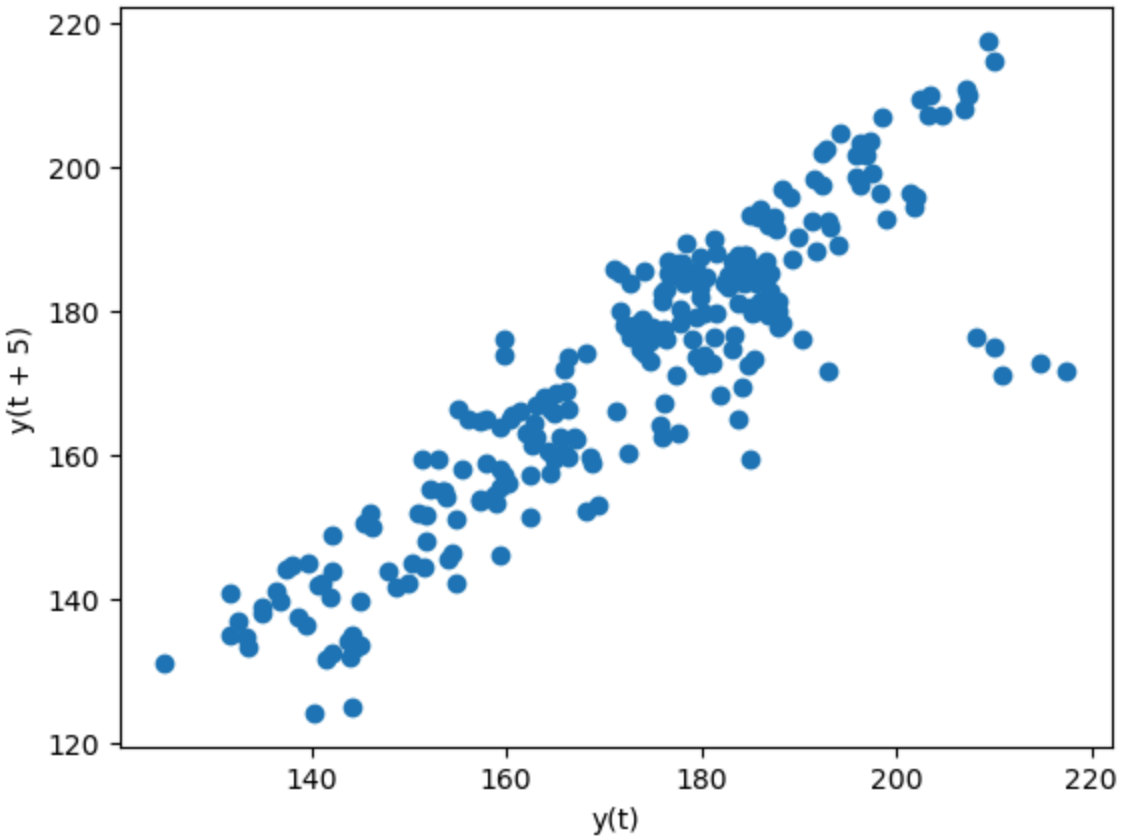


In [107…

```
lag_plot(fb.close)
plt.show()
```

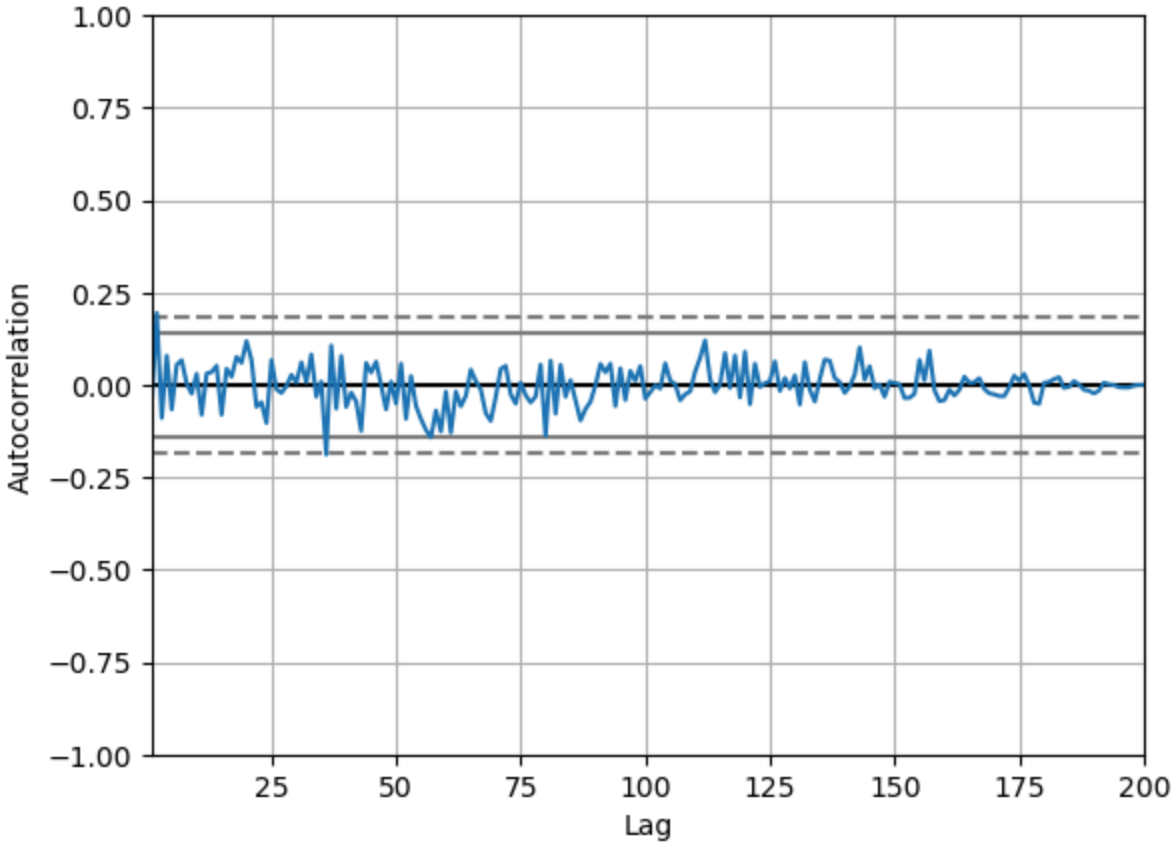


In [108…
```
lag_plot(fb.close, lag=5)
plt.show()
```



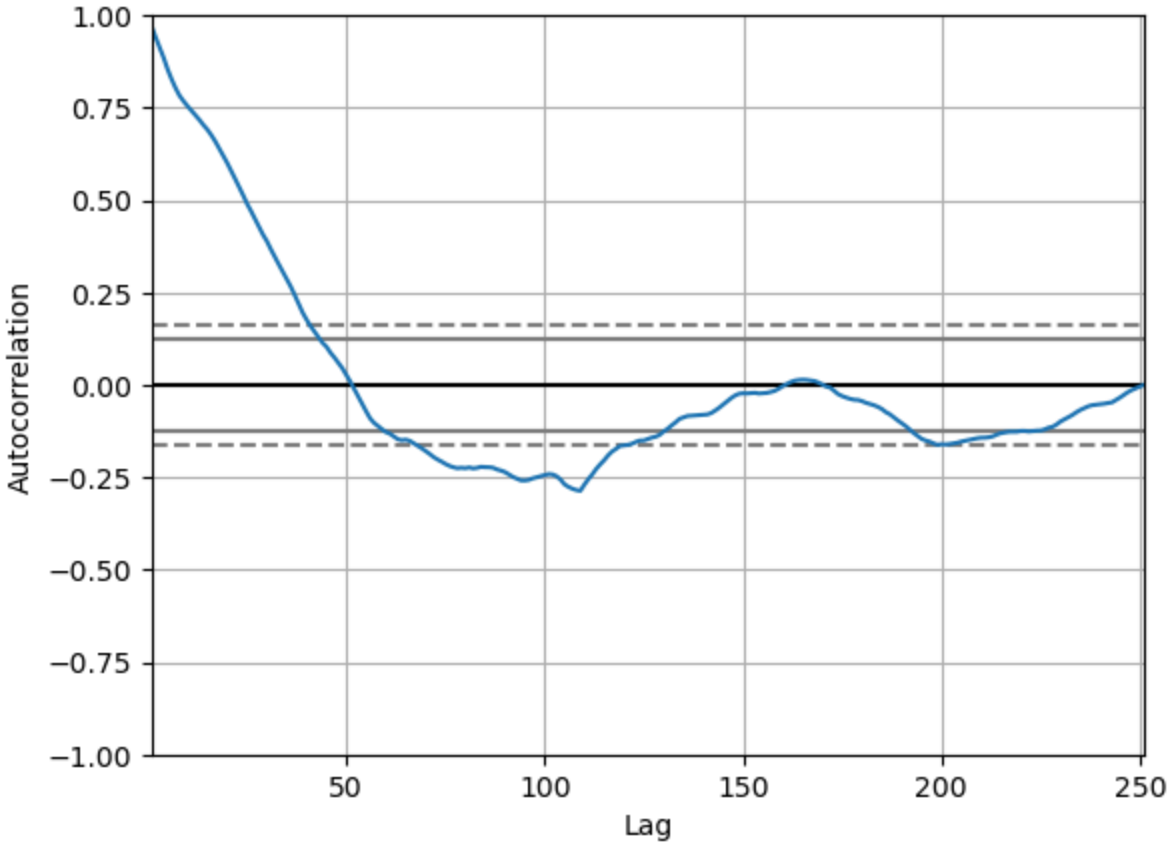
Autocorrelation plots

In [109…
```
from pandas.plotting import autocorrelation_plot
np.random.seed(0) # make this repeatable
autocorrelation_plot(pd.Series(np.random.random(size=200)))
plt.show()
```



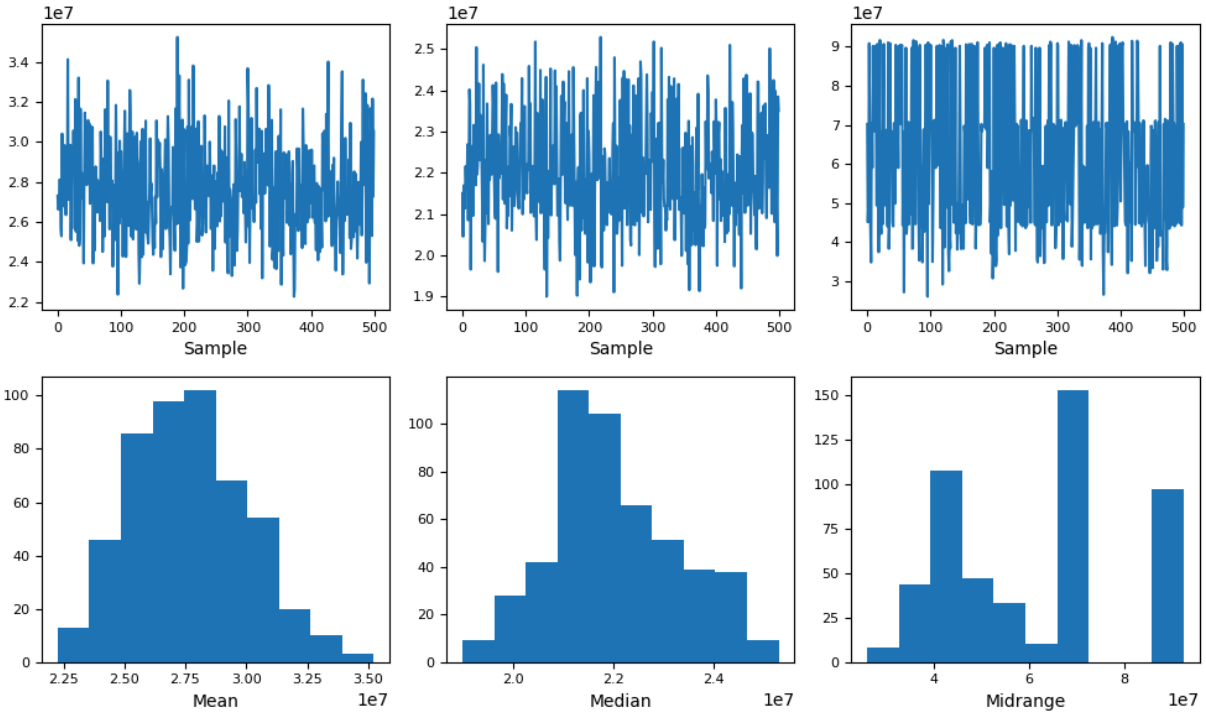
In [110…

```
autocorrelation_plot(fb.close)
plt.show()
```



Bootstrap plot

In [111…
```
from pandas.plotting import bootstrap_plot
fig = bootstrap_plot(fb.volume, fig=plt.figure(figsize=(10, 6)))
plt.show()
```



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.

In [113…
```
import matplotlib.pyplot as plt
import pandas as pd
```

In [115…
```
fb = pd.read_csv('fb_stock_prices_2018.csv')
fb.head()
```

Out[115…

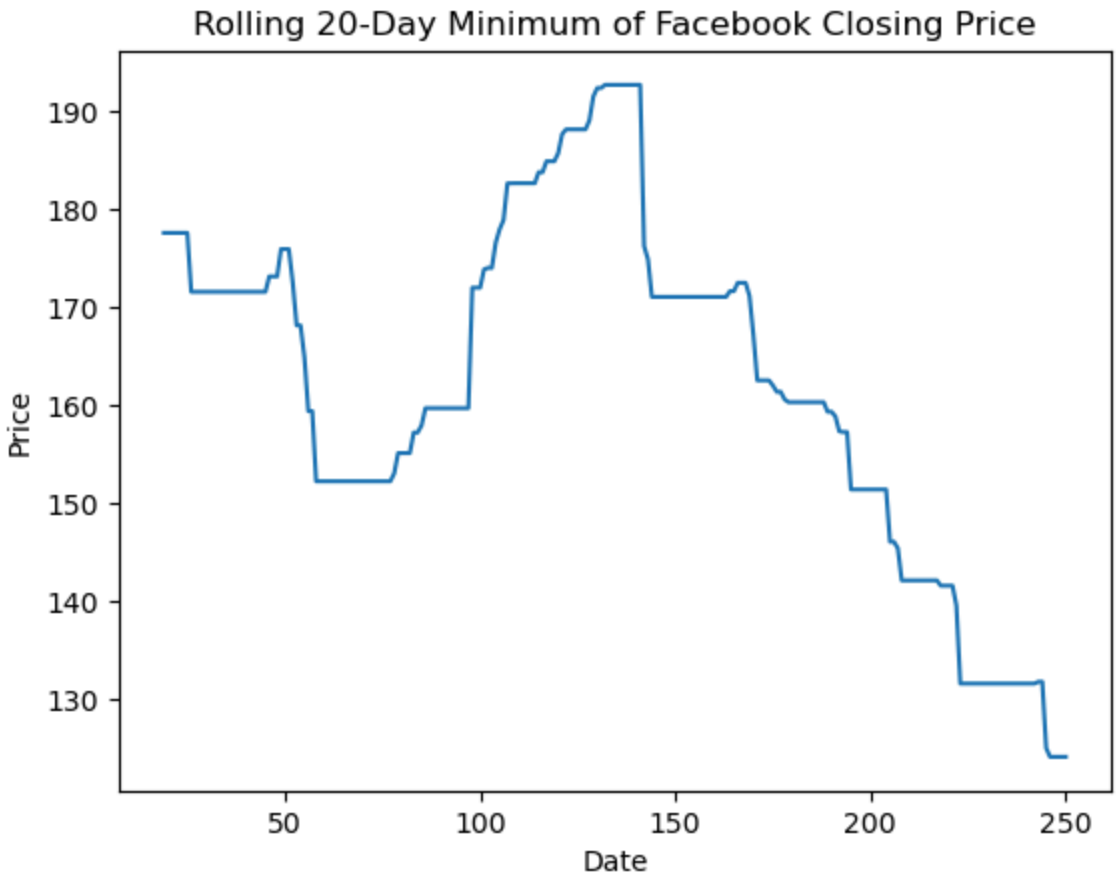
	date	open	high	low	close	volume
0	2018-01-02	177.68	181.58	177.5500	181.42	18151903
1	2018-01-03	181.88	184.78	181.3300	184.67	16886563
2	2018-01-04	184.90	186.21	184.0996	184.33	13880896
3	2018-01-05	185.59	186.90	184.9300	186.85	13574535
4	2018-01-08	187.20	188.90	186.3300	188.28	17994726

In [117…

```
# use rolling
fb_rollmin = fb['close'].rolling(window=20).min()

# use plot and add title
fb_rollmin.plot(title='Rolling 20-Day Minimum of Facebook Closing Price')

# add labels
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```



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

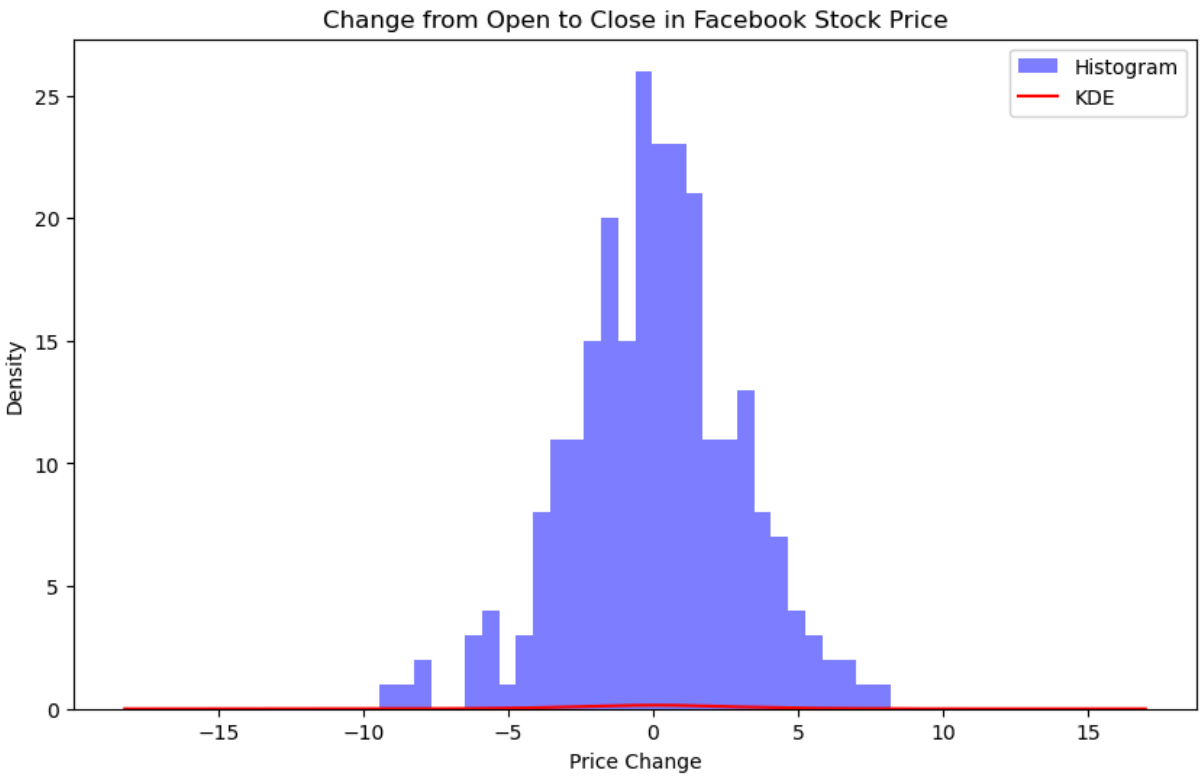
In [118…

```
fb_change = fb['close'] - fb['open']

# customization of histogram and plotting
plt.figure(figsize=(10, 6))

# use plot
fb_change.plot(kind='hist', bins=30, alpha=0.5, label='Histogram', color='blue')
fb_change.plot(kind='kde', label='KDE', color='red')

# add labels and title
plt.title('Change from Open to Close in Facebook Stock Price')
plt.xlabel('Price Change')
plt.legend()
plt.show()
```



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

In [119…

```
eq = pd.read_csv('earthquakes-1.csv')
eq.head()
```

Out[119...

	mag	magType	time	place	tsunami	parsed_place
0	1.35	ml	1539475168010	9km NE of Aguanga, CA	0	California
1	1.29	ml	1539475129610	9km NE of Aguanga, CA	0	California
2	3.42	ml	1539475062610	8km NE of Aguanga, CA	0	California
3	0.44	ml	1539474978070	9km NE of Aguanga, CA	0	California
4	2.16	md	1539474716050	10km NW of Avenal, CA	0	California

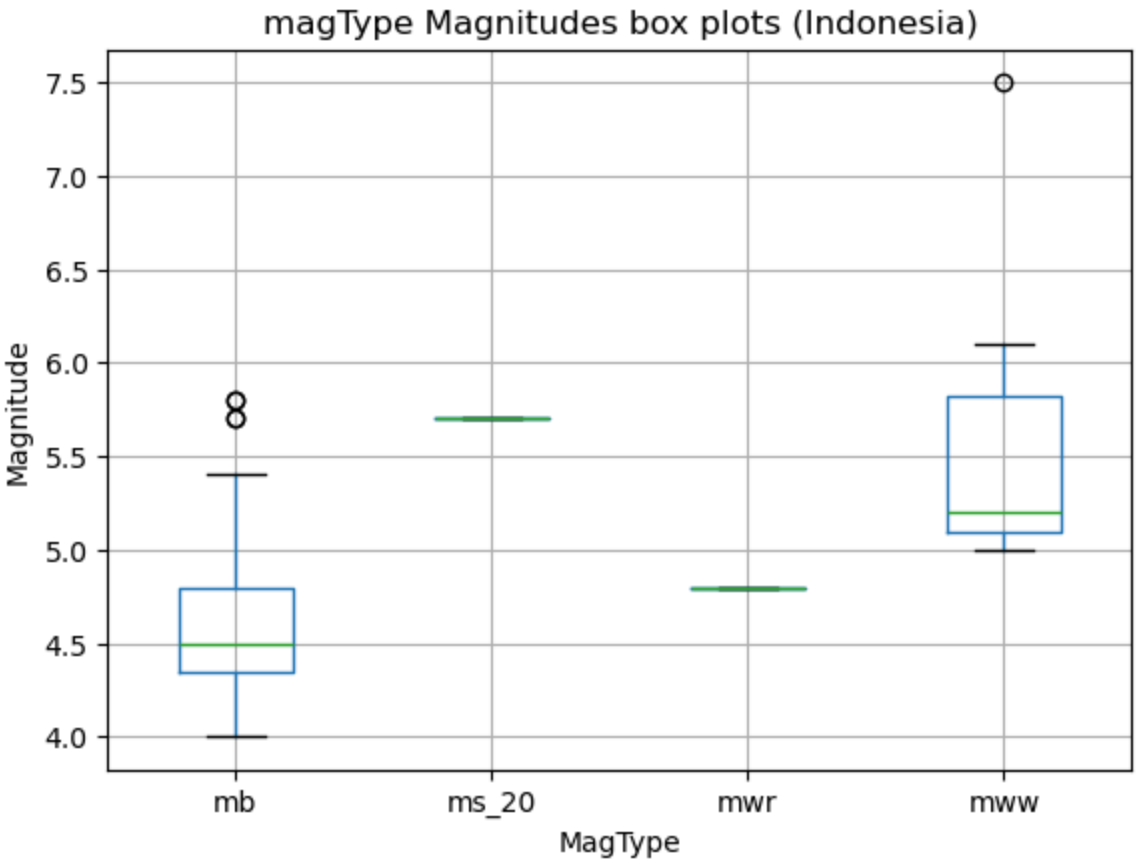
In [122...

```
# filter to indonesio only
indo = eq[eq['parsed_place'] == 'Indonesia']

# customization of box plot
plt.figure(figsize=(10, 6))
indo.boxplot(column='mag', by='magType')

# add title and labels
plt.title('magType Magnitudes box plots (Indonesia)')
plt.suptitle('')  # Remove default title
plt.xlabel('MagType')
plt.ylabel('Magnitude')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



4. Make a line plot of the difference between the weekly maximum high price and the weekly minimum low price for Facebook. This should be a single line.

In [126…
```
fb.head(3)
```

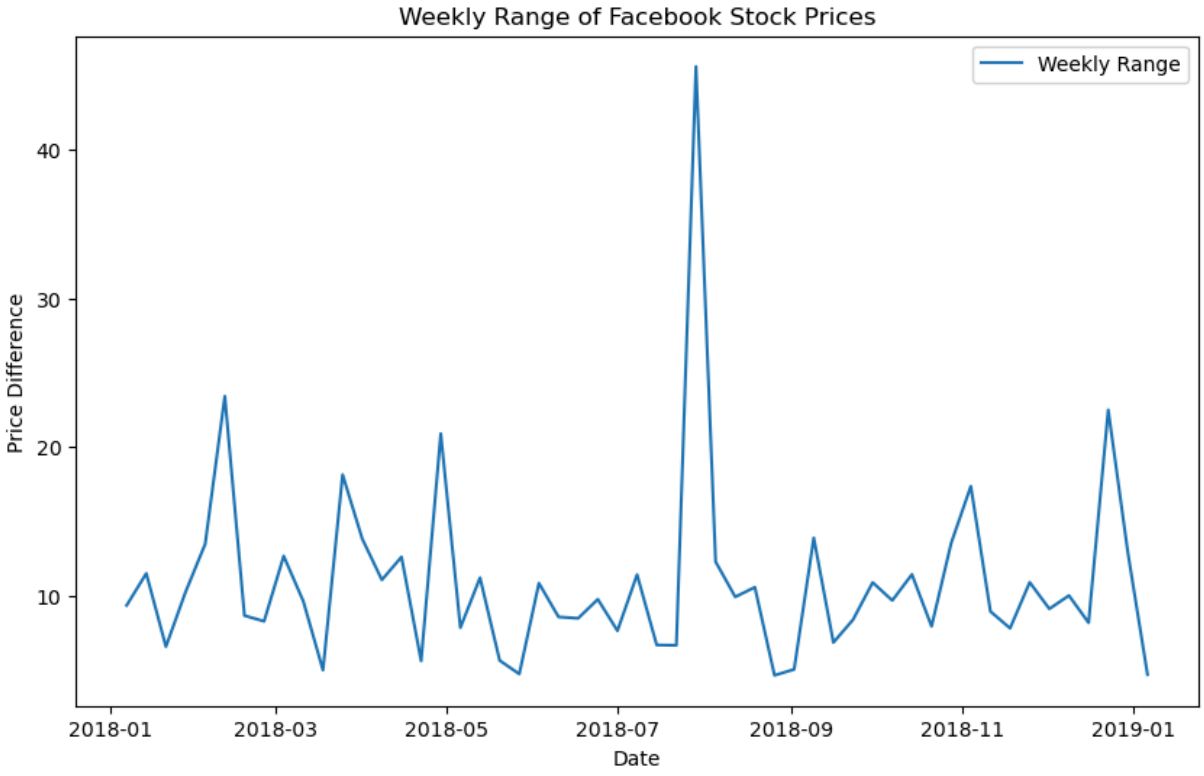
Out[126…

	date	open	high	low	close	volume
0	2018-01-02	177.68	181.58	177.5500	181.42	18151903
1	2018-01-03	181.88	184.78	181.3300	184.67	16886563
2	2018-01-04	184.90	186.21	184.0996	184.33	13880896

In [130…
```
# use resample in order to group data into intervals like weekly for example in thi
diff = fb.resample('W').agg({'high': 'max', 'low': 'min'})
diff['range'] = diff['high'] - diff['low']

# custom of figure for the plot display
plt.figure(figsize=(10, 6))
plt.plot(diff.index, diff['range'], label='Weekly Range')

# add title and labels
plt.title('Weekly Range of Facebook Stock Prices')
plt.xlabel('Date')
plt.ylabel('Price Difference')
plt.legend()
plt.show()
```



Using matplotlib and pandas, create two subplots side-by-side showing the effect that after-hours trading has had on Facebook's stock price:

- The first subplot will contain a line plot of the daily difference between that day's opening price and the prior day's closing price (be sure to review the Time series section of Aggregating Pandas DataFrames for an easy way to do this).
- The second subplot will be a bar plot showing the net effect this had monthly, using resample().
- Bonus #1: Color the bars according to whether they are gains in the stock price (green) or drops in the stock price (red).
- Bonus #2: Modify the x-axis of the bar plot to show the threeletter abbreviation for the month.

In [ ]:

Summary/Conclusion:

Today, I have learned the basic use as well as the advance use of matplot and pandas in order to output a visualization. The parts that were difficult is that when error came up and you may want to re-read the documents again and had to modify. I did not know that both matplot and pandas

have features like these. Although, I do know that they're meant for visualization but I am not aware that they can be used so advanced like these and I am sure there are much more advance. I do hope I get to understand them. If I am finally able to write these codes without looking back to documents then I'll be unstoppable.

In [ ]: