05_Visualization

2020 年 7 月 19 日

1 Python for Finance Chapter 5

1.1 Visualization

1.1.1 Two-Dimensional Plotting

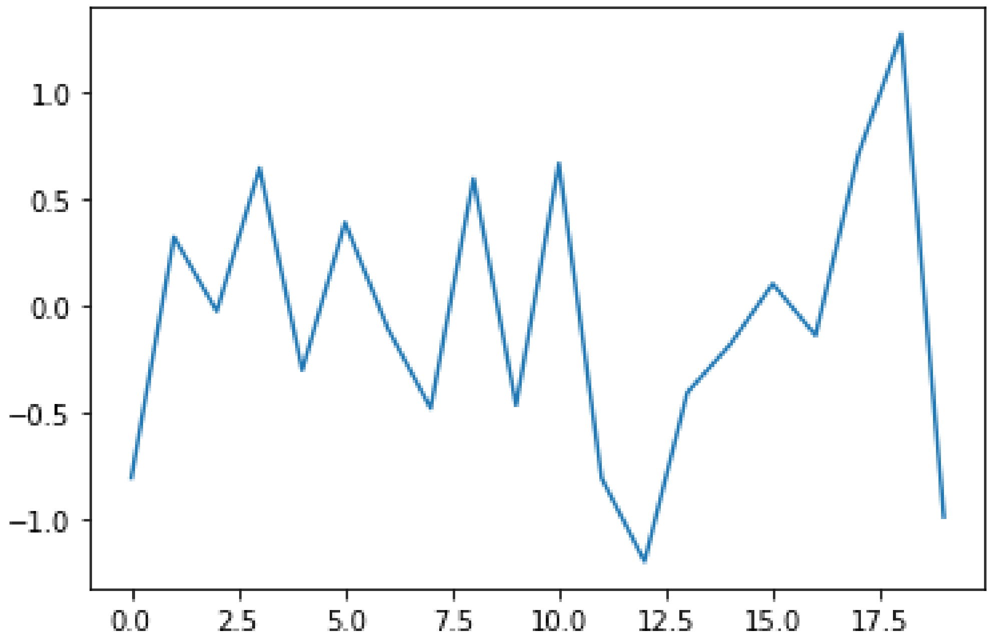
1.1.2 One-Dimensional Data Set

[1]:
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import warnings; warnings.simplefilter('ignore')
# import seaborn as sns; sns.set()
%matplotlib inline
```

[2]:
```
np.random.seed(1000)
y = np.random.standard_normal(20)
```

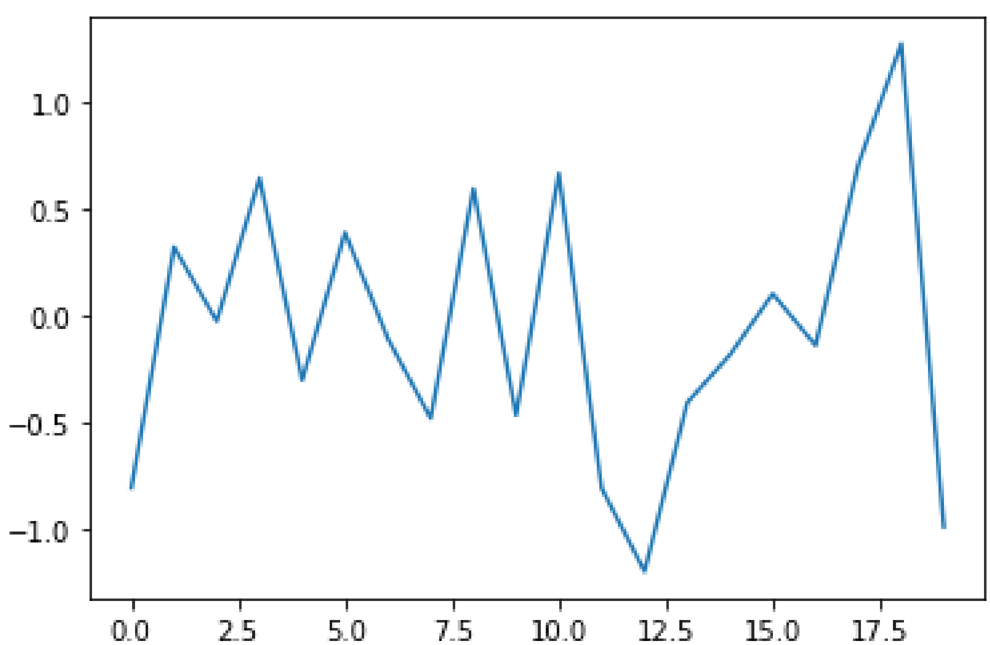
[3]:
```
x = range(len(y))
plt.plot(x, y)
# tag: matplotlib_0
# title: Plot given x- and y-values
```

[3]:
```
[<matplotlib.lines.Line2D at 0x23b9c838088>]
```



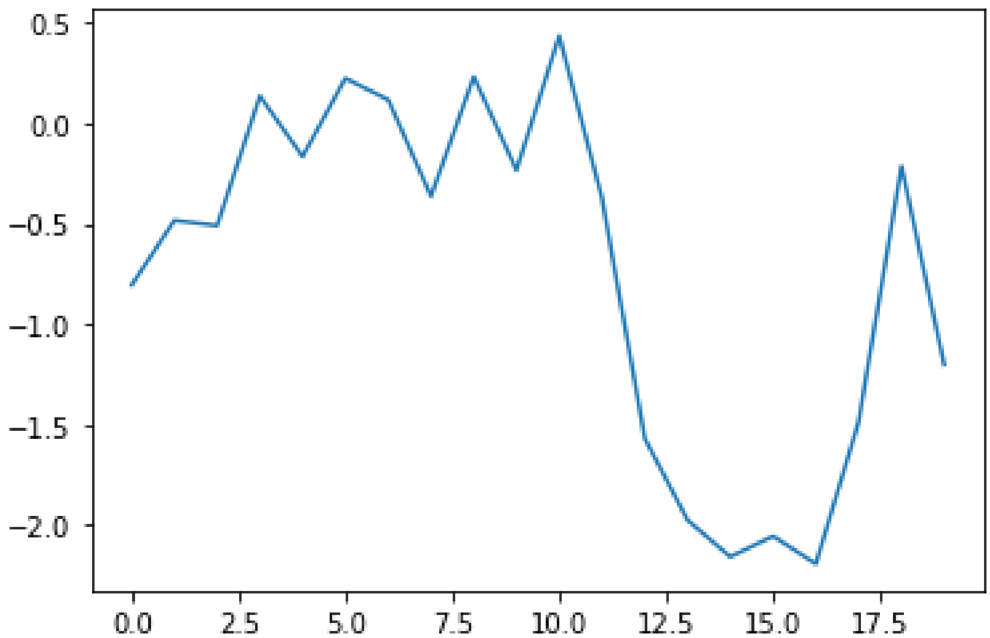
[4]:
```
plt.plot(y)
# tag: matplotlib_1
# title: Plot given data as 1d-array
```

[4]: `[<matplotlib.lines.Line2D at 0x23b99f7cb88>]`



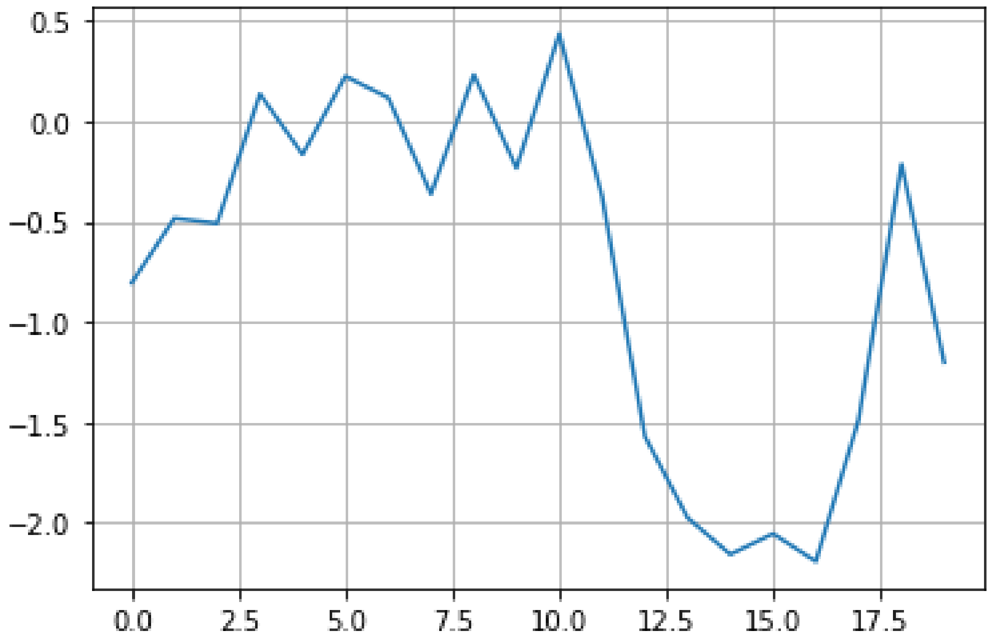
[5]:
```
plt.plot(y.cumsum())
# tag: matplotlib_2
# title: Plot given a 1d-array with method attached
```

[5]: [<matplotlib.lines.Line2D at 0x23b9d160848>]



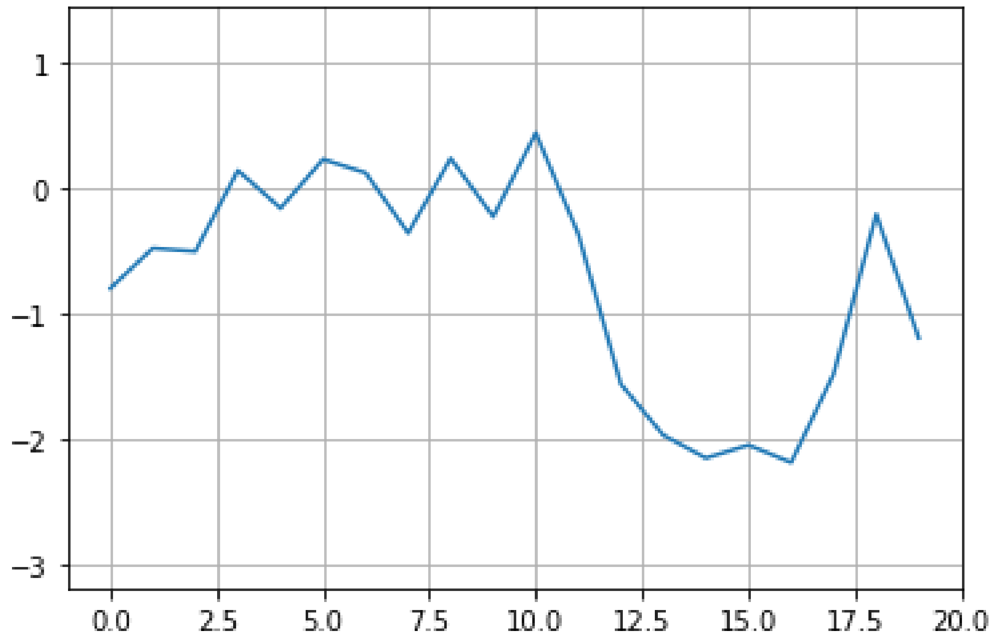
[6]:
```
plt.plot(y.cumsum())
plt.grid(True)  # adds a grid
plt.axis('tight')  # adjusts the axis ranges
# tag: matplotlib_3_a
# title: Plot with grid and tight axes
```

[6]: (-0.9500000000000001, 19.95, -2.322818663749045, 0.5655085808655865)



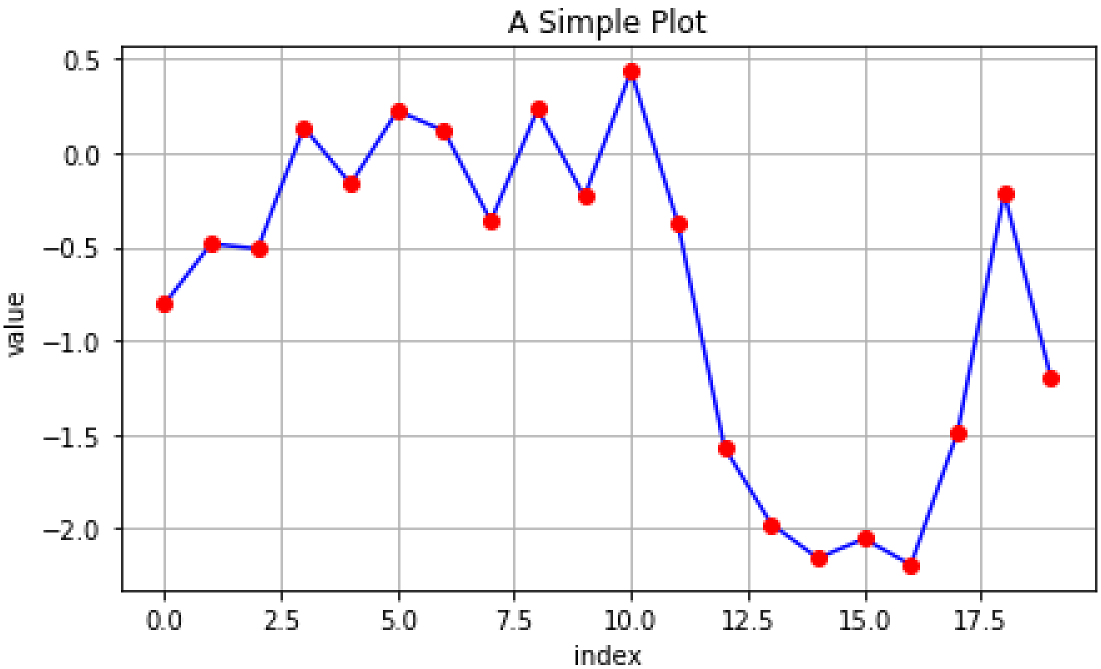
[7]:
```
plt.plot(y.cumsum())
plt.grid(True)
plt.xlim(-1, 20)
plt.ylim(np.min(y.cumsum()) - 1,
         np.max(y.cumsum()) + 1)
# tag: matplotlib_3_b
# title: Plot with custom axes limits
```

[7]: (-3.1915310617211072, 1.4342209788376488)



[8]:
```
plt.figure(figsize=(7, 4))
  # the figsize parameter defines the
  # size of the figure in (width, height)
plt.plot(y.cumsum(), 'b', lw=1.5)
plt.plot(y.cumsum(), 'ro')
plt.grid(True)
plt.axis('tight')
plt.xlabel('index')
plt.ylabel('value')
plt.title('A Simple Plot')
# tag: matplotlib_4
# title: Plot with typical labels
```

[8]:
```
Text(0.5, 1.0, 'A Simple Plot')
```

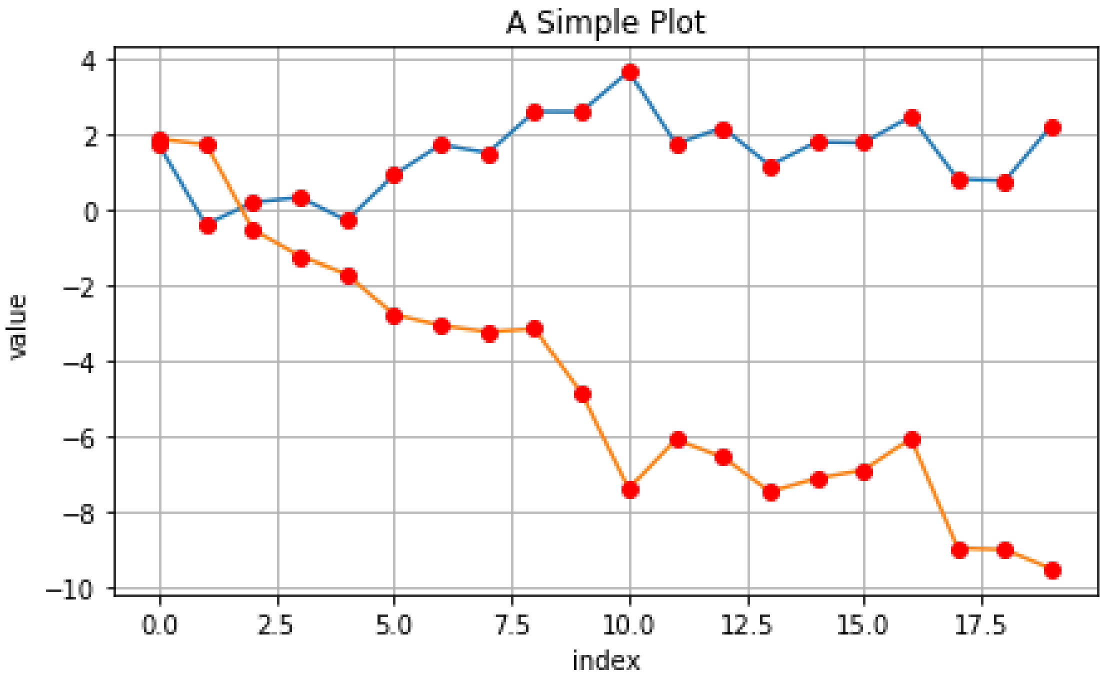


1.1.3 Two-Dimensional Data Set

[9]:
```
np.random.seed(2000)
y = np.random.standard_normal((20, 2)).cumsum(axis=0)
```

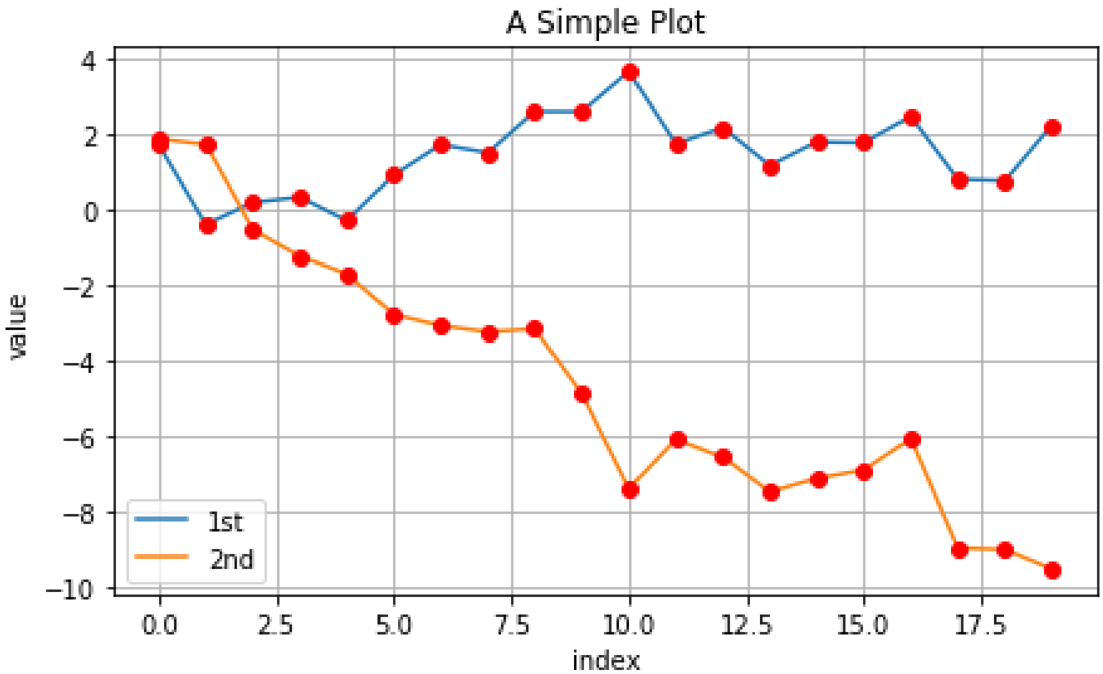
[10]:
```
plt.figure(figsize=(7, 4))
plt.plot(y, lw=1.5)
  # plots two lines
plt.plot(y, 'ro')
  # plots two dotted lines
plt.grid(True)
plt.axis('tight')
plt.xlabel('index')
plt.ylabel('value')
plt.title('A Simple Plot')
# tag: matplotlib_5
# title: Plot with two data sets
```

[10]:
```
Text(0.5, 1.0, 'A Simple Plot')
```



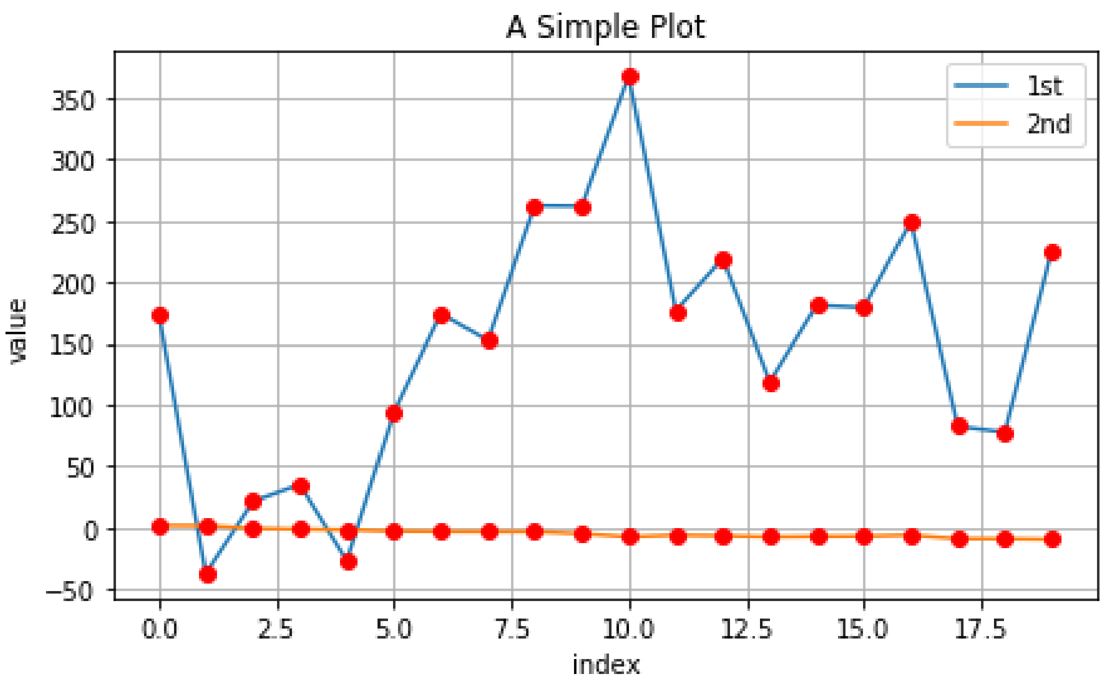
[11]:
```
plt.figure(figsize=(7, 4))
plt.plot(y[:, 0], lw=1.5, label='1st')
plt.plot(y[:, 1], lw=1.5, label='2nd')
plt.plot(y, 'ro')
plt.grid(True)
plt.legend(loc=0)
plt.axis('tight')
plt.xlabel('index')
plt.ylabel('value')
plt.title('A Simple Plot')
# tag: matplotlib_6
# title: Plot with labeled data sets
```

[11]: Text(0.5, 1.0, 'A Simple Plot')



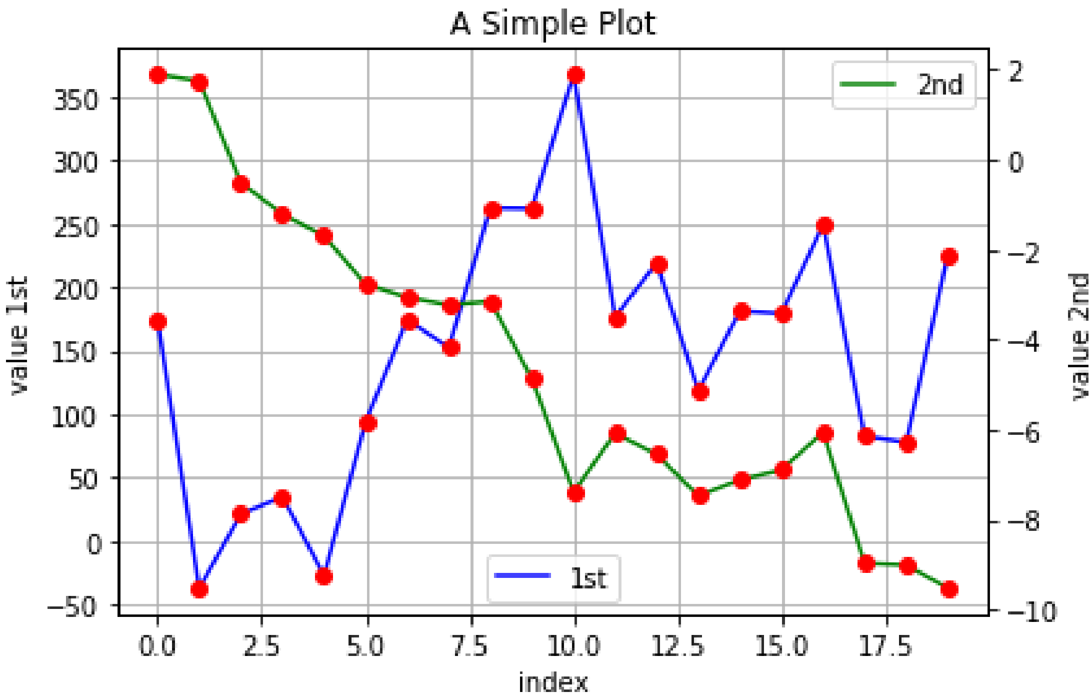
[12]:
```
y[:, 0] = y[:, 0] * 100
plt.figure(figsize=(7, 4))
plt.plot(y[:, 0], lw=1.5, label='1st')
plt.plot(y[:, 1], lw=1.5, label='2nd')
plt.plot(y, 'ro')
plt.grid(True)
plt.legend(loc=0)
plt.axis('tight')
plt.xlabel('index')
plt.ylabel('value')
plt.title('A Simple Plot')
# tag: matplotlib_7
# title: Plot with two differently scaled data sets
```

[12]:
```
Text(0.5, 1.0, 'A Simple Plot')
```



[13]:
```
fig, ax1 = plt.subplots()
plt.plot(y[:, 0], 'b', lw=1.5, label='1st')
plt.plot(y[:, 0], 'ro')
plt.grid(True)
plt.legend(loc=8)
plt.axis('tight')
plt.xlabel('index')
plt.ylabel('value 1st')
plt.title('A Simple Plot')
ax2 = ax1.twinx()
plt.plot(y[:, 1], 'g', lw=1.5, label='2nd')
plt.plot(y[:, 1], 'ro')
plt.legend(loc=0)
plt.ylabel('value 2nd')
# tag: matplotlib_8
# title: Plot with two data sets and two y-axes
```

[13]:
```
Text(0, 0.5, 'value 2nd')
```

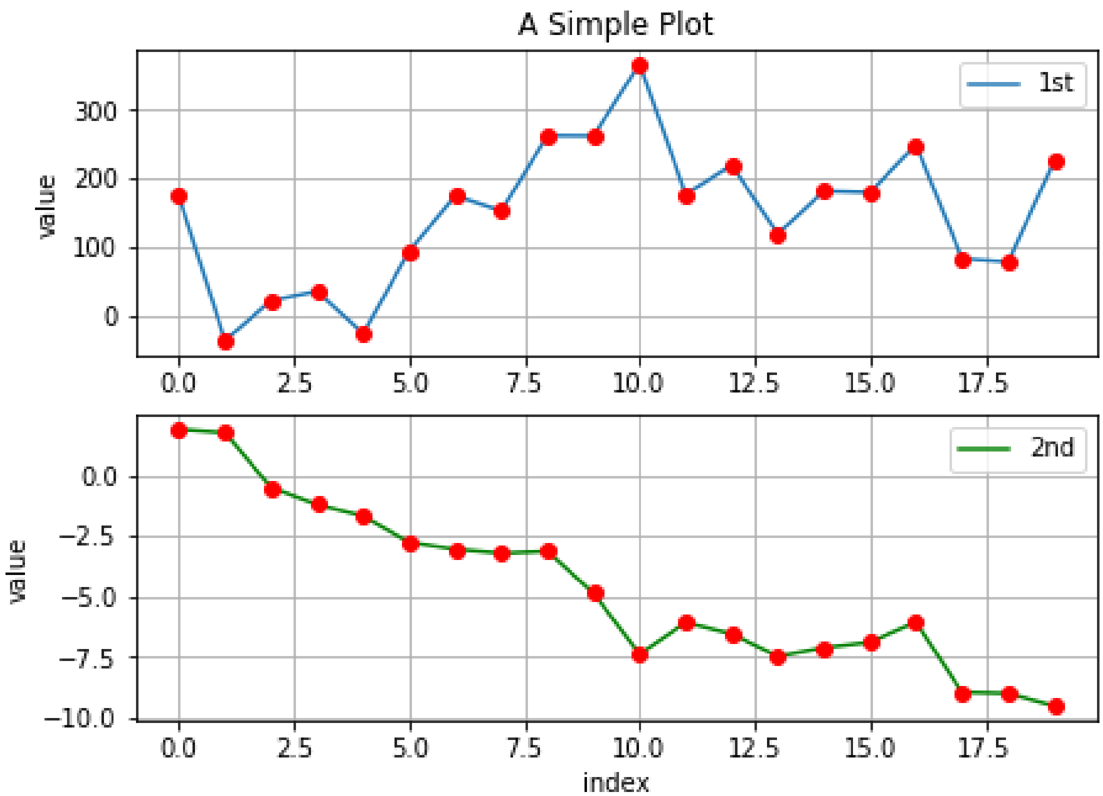


[14]:

```
plt.figure(figsize=(7, 5))
plt.subplot(211)
plt.plot(y[:, 0], lw=1.5, label='1st')
plt.plot(y[:, 0], 'ro')
plt.grid(True)
plt.legend(loc=0)
plt.axis('tight')
plt.ylabel('value')
plt.title('A Simple Plot')
plt.subplot(212)
plt.plot(y[:, 1], 'g', lw=1.5, label='2nd')
plt.plot(y[:, 1], 'ro')
plt.grid(True)
plt.legend(loc=0)
plt.axis('tight')
plt.xlabel('index')
plt.ylabel('value')
# tag: matplotlib_9
```

```
# title: Plot with two sub-plots
```

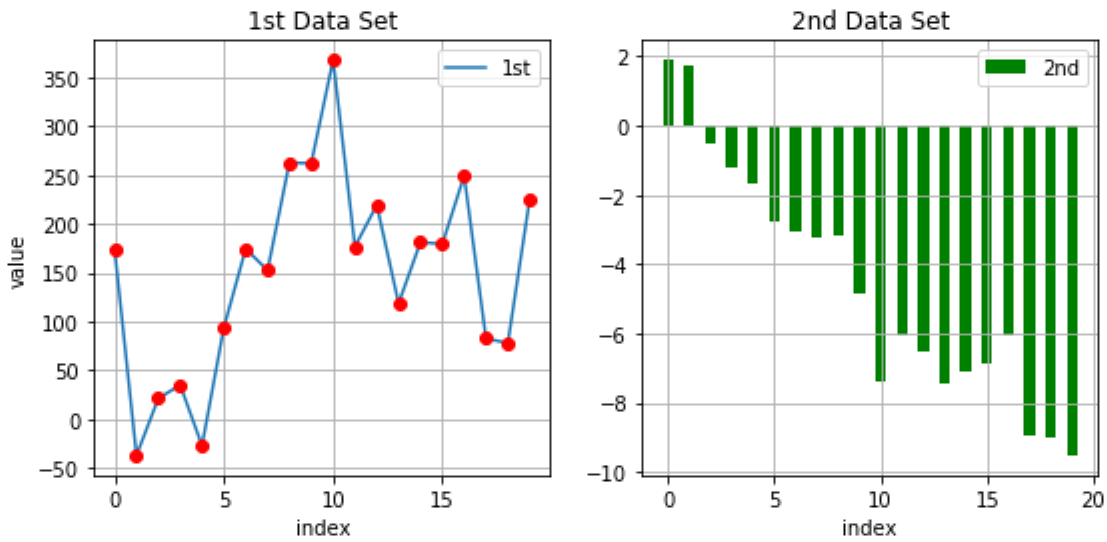
[14]: Text(0, 0.5, 'value')



[15]:
```
plt.figure(figsize=(9, 4))
plt.subplot(121)
plt.plot(y[:, 0], lw=1.5, label='1st')
plt.plot(y[:, 0], 'ro')
plt.grid(True)
plt.legend(loc=0)
plt.axis('tight')
plt.xlabel('index')
plt.ylabel('value')
plt.title('1st Data Set')
plt.subplot(122)
plt.bar(np.arange(len(y)), y[:, 1], width=0.5,
```

```
        color='g', label='2nd')
plt.grid(True)
plt.legend(loc=0)
plt.axis('tight')
plt.xlabel('index')
plt.title('2nd Data Set')
# tag: matplotlib_10
# title: Plot combining line/point sub-plot with bar sub-plot
# size: 80
```

[15]: Text(0.5, 1.0, '2nd Data Set')



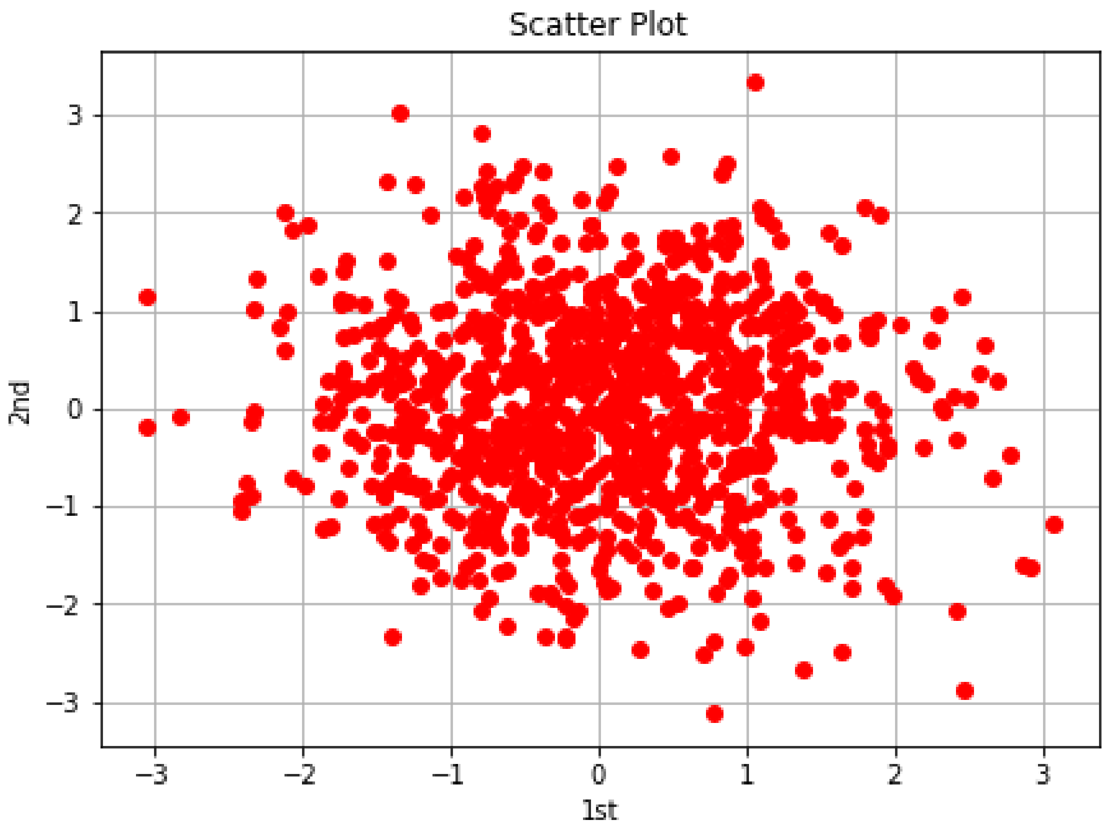
1.1.4 Other Plot Styles

[16]:
```
y = np.random.standard_normal((1000, 2))
```

[17]:
```
plt.figure(figsize=(7, 5))
plt.plot(y[:, 0], y[:, 1], 'ro')
plt.grid(True)
plt.xlabel('1st')
plt.ylabel('2nd')
plt.title('Scatter Plot')
```

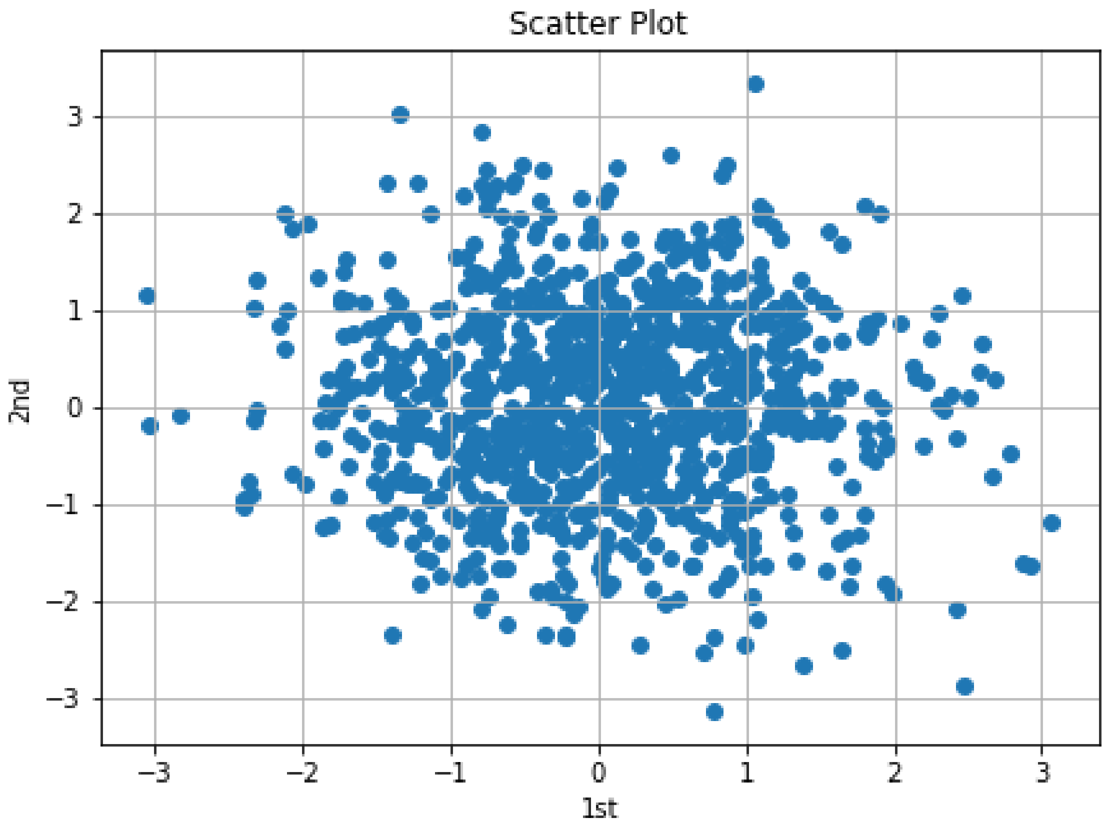
```
# tag: matplotlib_11_a
# title: Scatter plot via +plot+ function
```

[17]: Text(0.5, 1.0, 'Scatter Plot')



[18]:
```
plt.figure(figsize=(7, 5))
plt.scatter(y[:, 0], y[:, 1], marker='o')
plt.grid(True)
plt.xlabel('1st')
plt.ylabel('2nd')
plt.title('Scatter Plot')
# tag: matplotlib_11_b
# title: Scatter plot via +scatter+ function
```

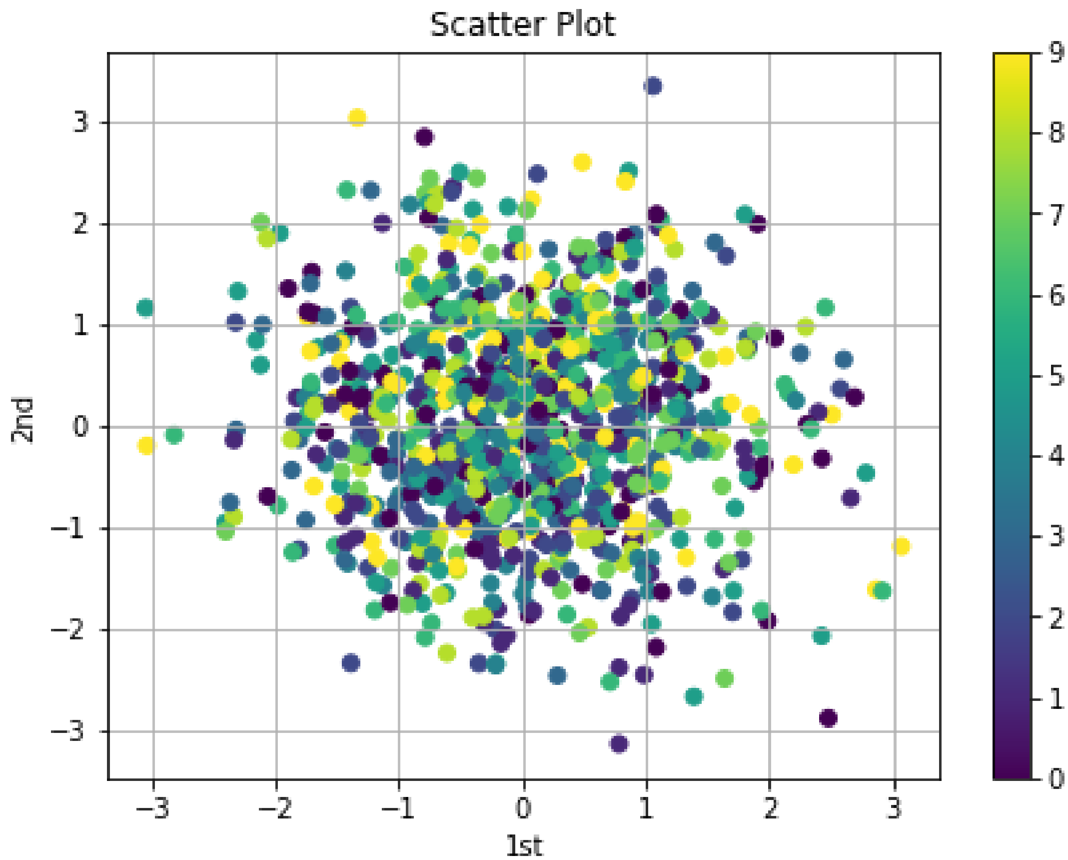
[18]: Text(0.5, 1.0, 'Scatter Plot')



[19]:
```
c = np.random.randint(0, 10, len(y))
```

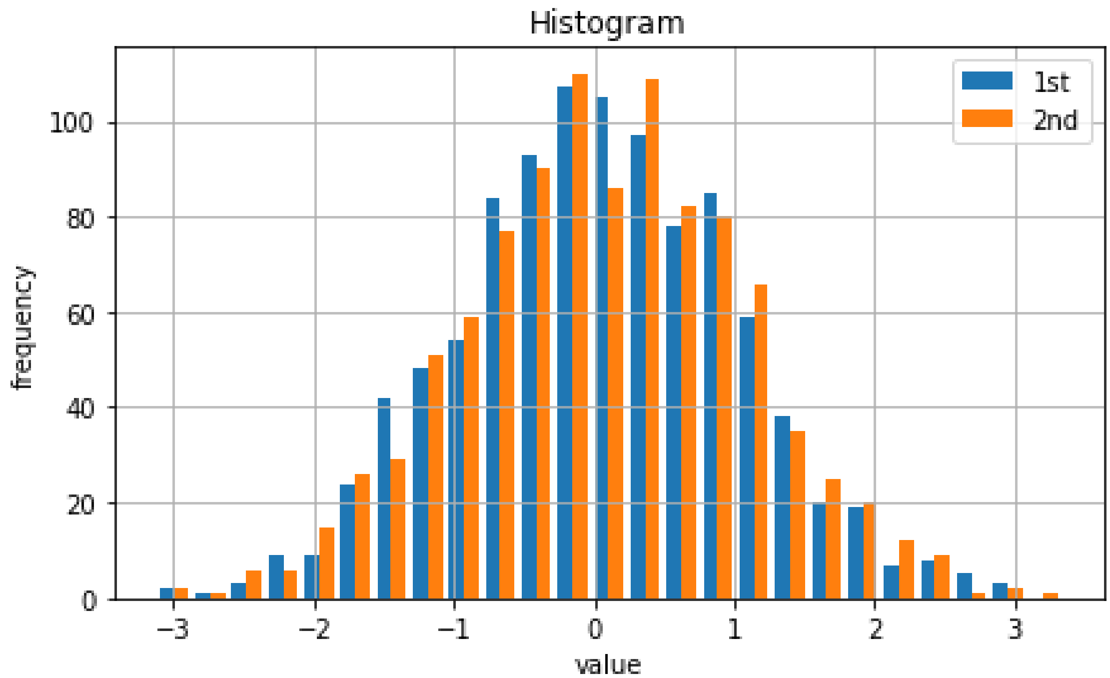
[20]:
```
plt.figure(figsize=(7, 5))
plt.scatter(y[:, 0], y[:, 1], c=c, marker='o')
plt.colorbar()
plt.grid(True)
plt.xlabel('1st')
plt.ylabel('2nd')
plt.title('Scatter Plot')
# tag: matplotlib_11_c
# title: Scatter plot with third dimension
```

[20]:
```
Text(0.5, 1.0, 'Scatter Plot')
```



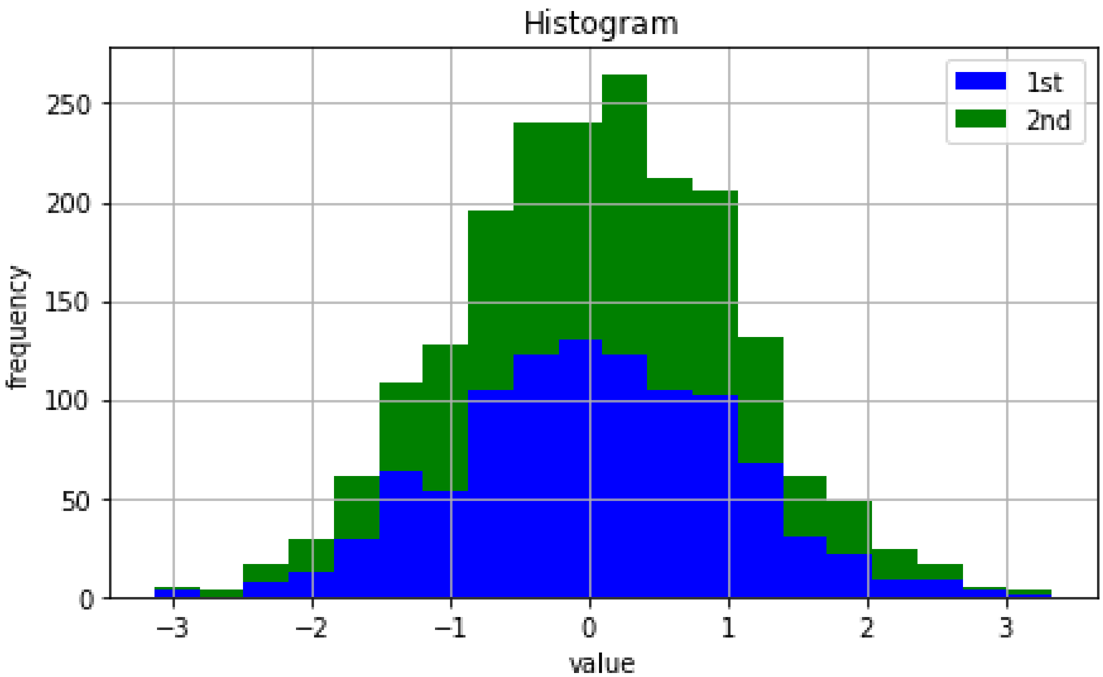
[21]:
```
plt.figure(figsize=(7, 4))
plt.hist(y, label=['1st', '2nd'], bins=25)
plt.grid(True)
plt.legend(loc=0)
plt.xlabel('value')
plt.ylabel('frequency')
plt.title('Histogram')
# tag: matplotlib_12_a
# title: Histogram for two data sets
```

[21]: Text(0.5, 1.0, 'Histogram')



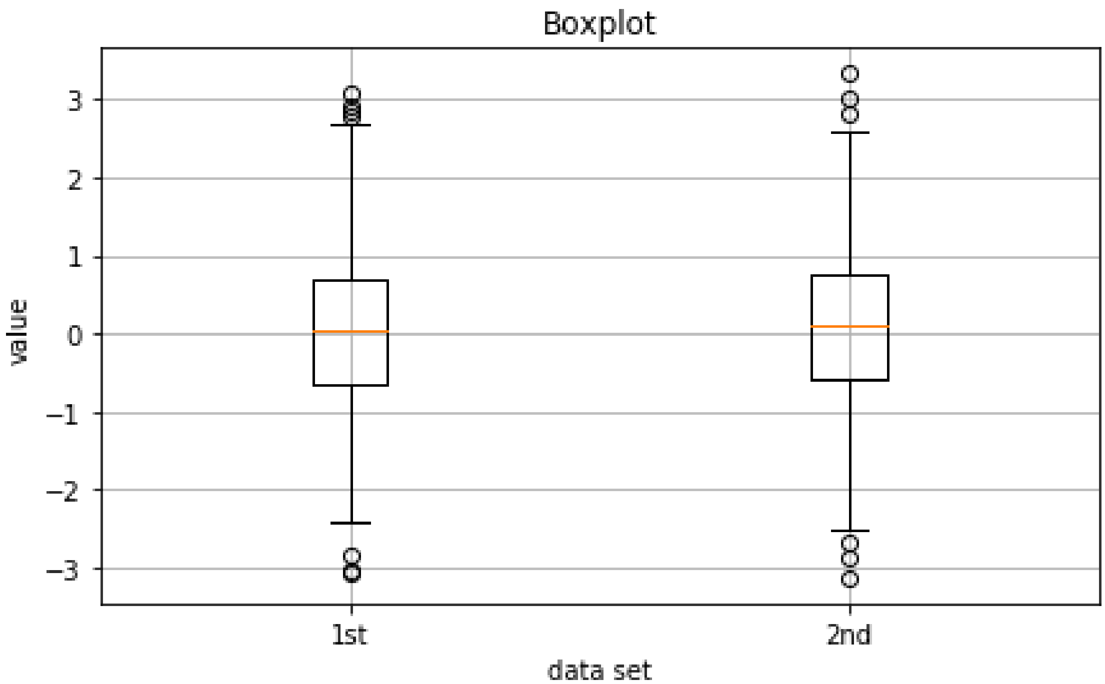

```
[22]: plt.figure(figsize=(7, 4))
      plt.hist(y, label=['1st', '2nd'], color=['b', 'g'],
                  stacked=True, bins=20)
      plt.grid(True)
      plt.legend(loc=0)
      plt.xlabel('value')
      plt.ylabel('frequency')
      plt.title('Histogram')
      # tag: matplotlib_12_b
      # title: Stacked histogram for two data sets
```

```
[22]: Text(0.5, 1.0, 'Histogram')
```



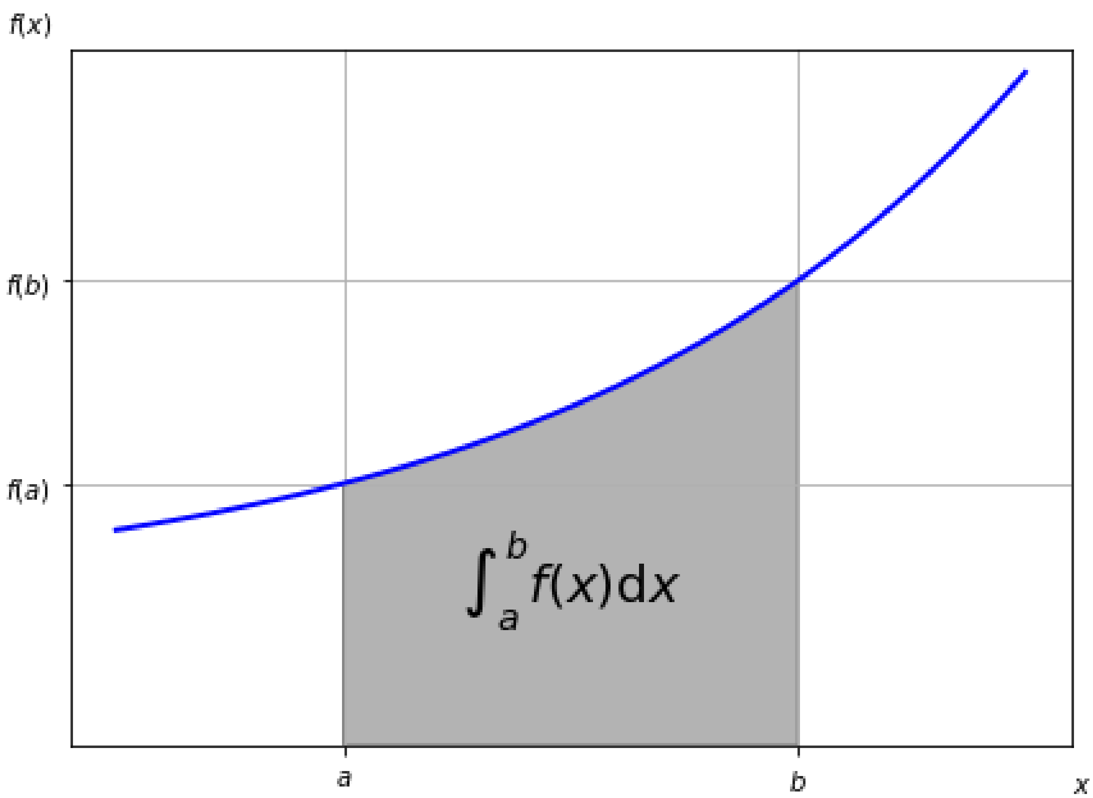
[23]:
```
fig, ax = plt.subplots(figsize=(7, 4))
plt.boxplot(y)
plt.grid(True)
plt.setp(ax, xticklabels=['1st', '2nd'])
plt.xlabel('data set')
plt.ylabel('value')
plt.title('Boxplot')
# tag: matplotlib_13
# title: Boxplot for two data sets
# size: 70
```

[23]:
```
Text(0.5, 1.0, 'Boxplot')
```



[24]:
```
from matplotlib.patches import Polygon
def func(x):
    return 0.5 * np.exp(x) + 1

a, b = 0.5, 1.5  # integral limits
x = np.linspace(0, 2)
y = func(x)

fig, ax = plt.subplots(figsize=(7, 5))
plt.plot(x, y, 'b', linewidth=2)
plt.ylim(ymin=0)

# Illustrate the integral value, i.e. the area under the function
# between lower and upper limit
Ix = np.linspace(a, b)
Iy = func(Ix)
verts = [(a, 0)] + list(zip(Ix, Iy)) + [(b, 0)]
poly = Polygon(verts, facecolor='0.7', edgecolor='0.5')
```

```python
ax.add_patch(poly)

plt.text(0.5 * (a + b), 1, r"$\int_a^b f(x)\mathrm{d}x$",
         horizontalalignment='center', fontsize=20)

plt.figtext(0.9, 0.075, '$x$')
plt.figtext(0.075, 0.9, '$f(x)$')

ax.set_xticks((a, b))
ax.set_xticklabels(('$a$', '$b$'))
ax.set_yticks([func(a), func(b)])
ax.set_yticklabels(('$f(a)$', '$f(b)$'))
plt.grid(True)
# tag: matplotlib_math
# title: Exponential function, integral area and Latex labels
# size: 60
```



1.2 Financial Plots

[25]:
```
import matplotlib
import mplfinance as mpf
print(mpf.available_styles())
```

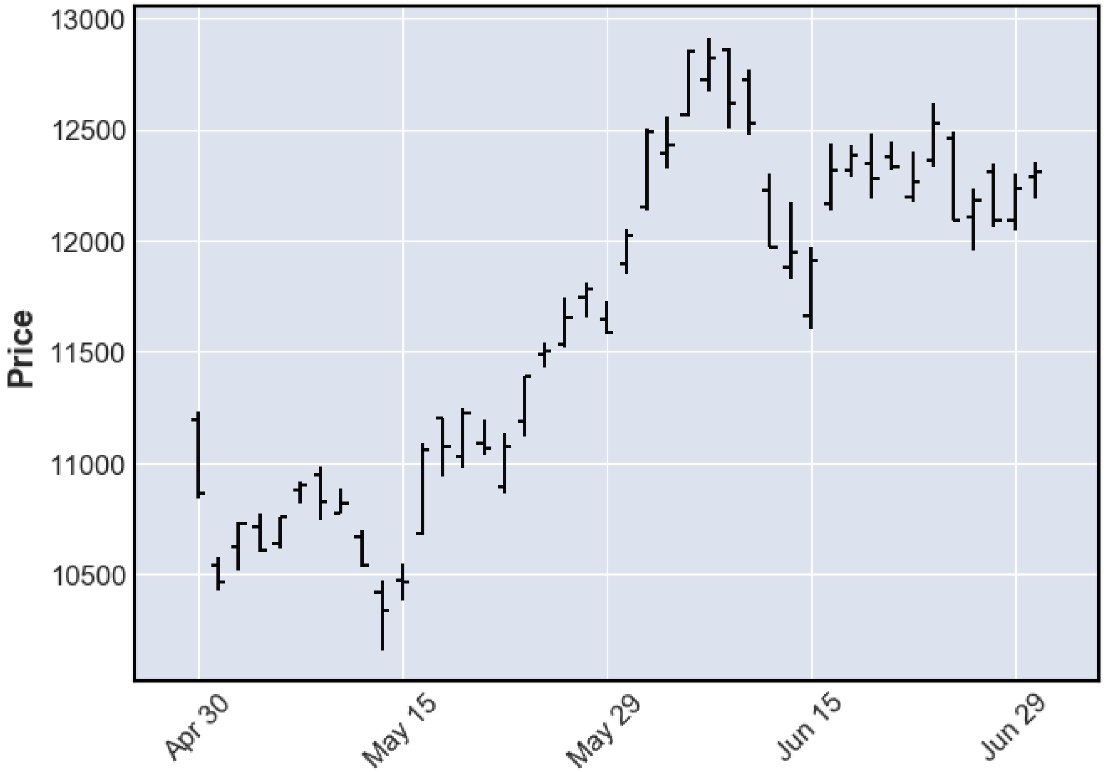
```
['binance', 'blueskies', 'brasil', 'charles', 'checkers', 'classic', 'default',
'mike', 'nightclouds', 'sas', 'starsandstripes', 'yahoo']
```

[27]:
```
import pandas_datareader as pdr
data= pdr.get_data_yahoo('^GDAXI', '2020/5/1', '2020/6/30')
```

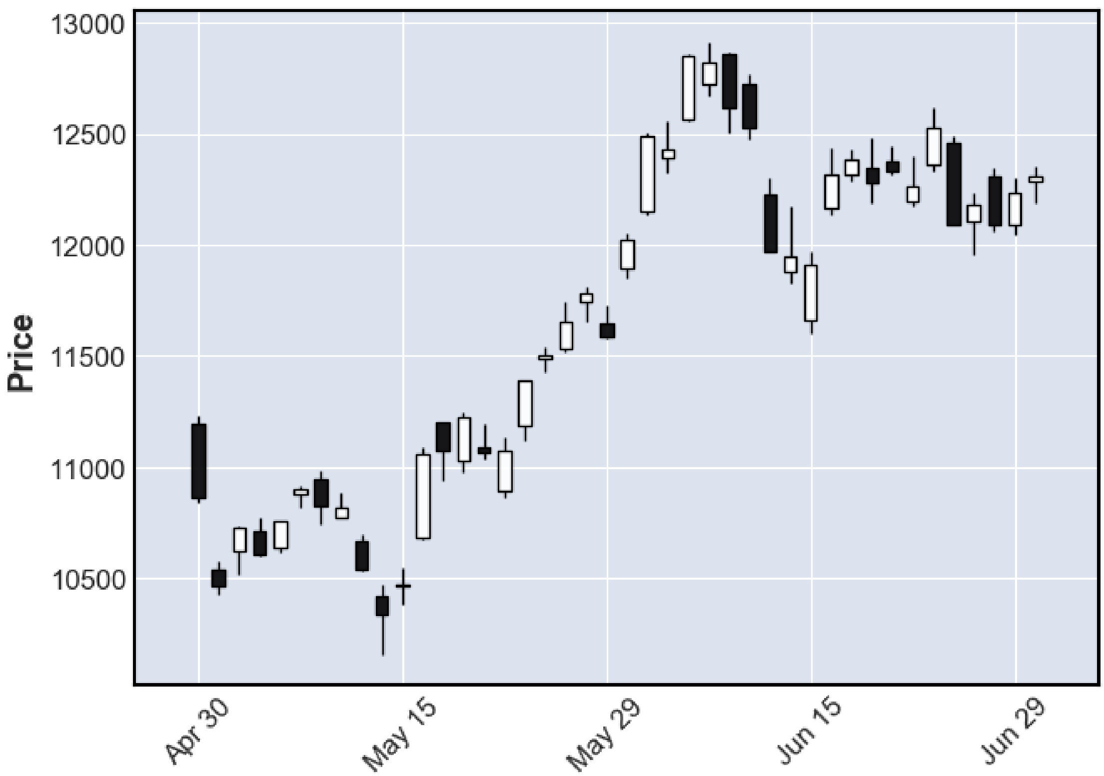
[28]:
```
data[:2]
```

[28]:
```
                    High           Low          Open         Close     Volume  \
Date
2020-04-30  11235.570312  10839.299805  11195.209961  10861.639648  162733400
2020-05-04  10578.429688  10426.059570  10543.360352  10466.799805  140425100

               Adj Close
Date
2020-04-30  10861.639648
2020-05-04  10466.799805
```

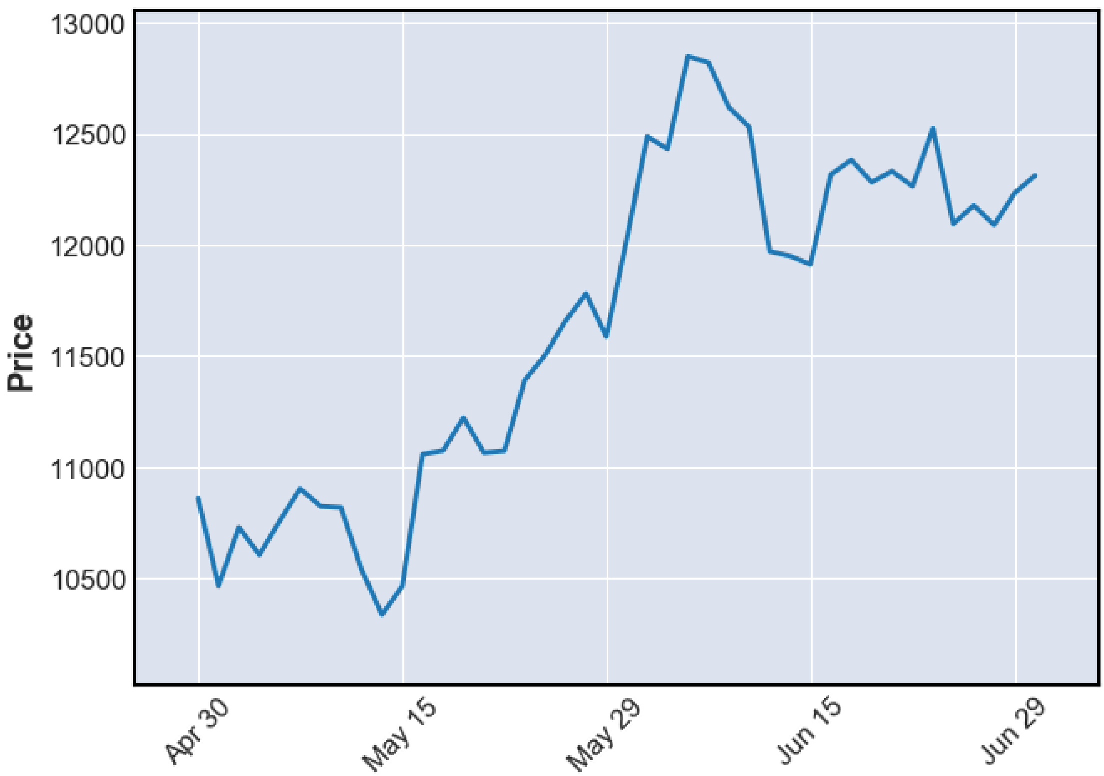
[29]:
```
mpf.plot(data)
```



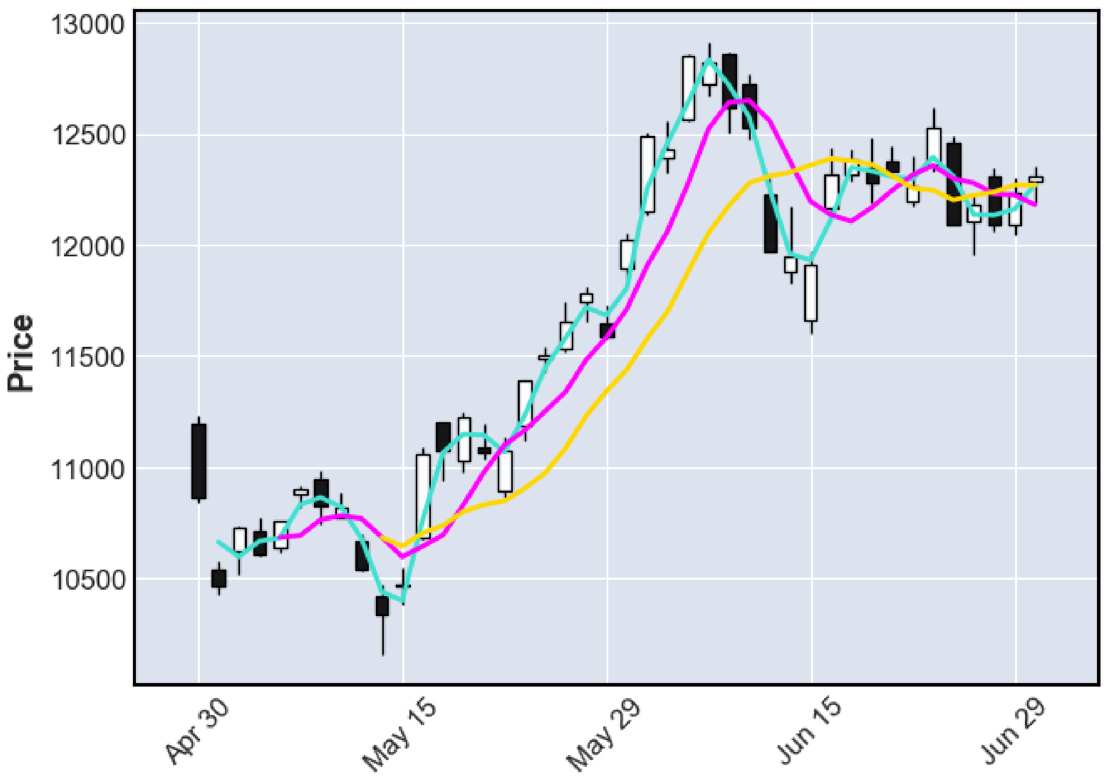

```
[30]: mpf.plot(data,type='candle')
```



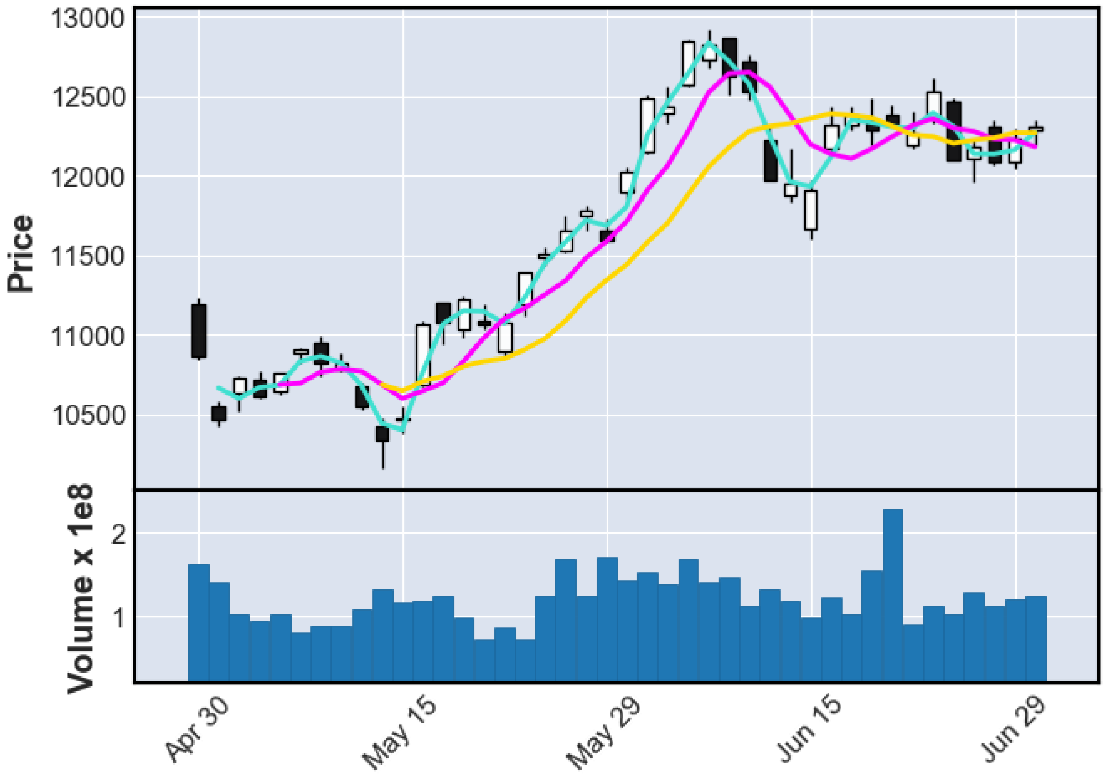
[31]:
```
mpf.plot(data, type = 'line')
```



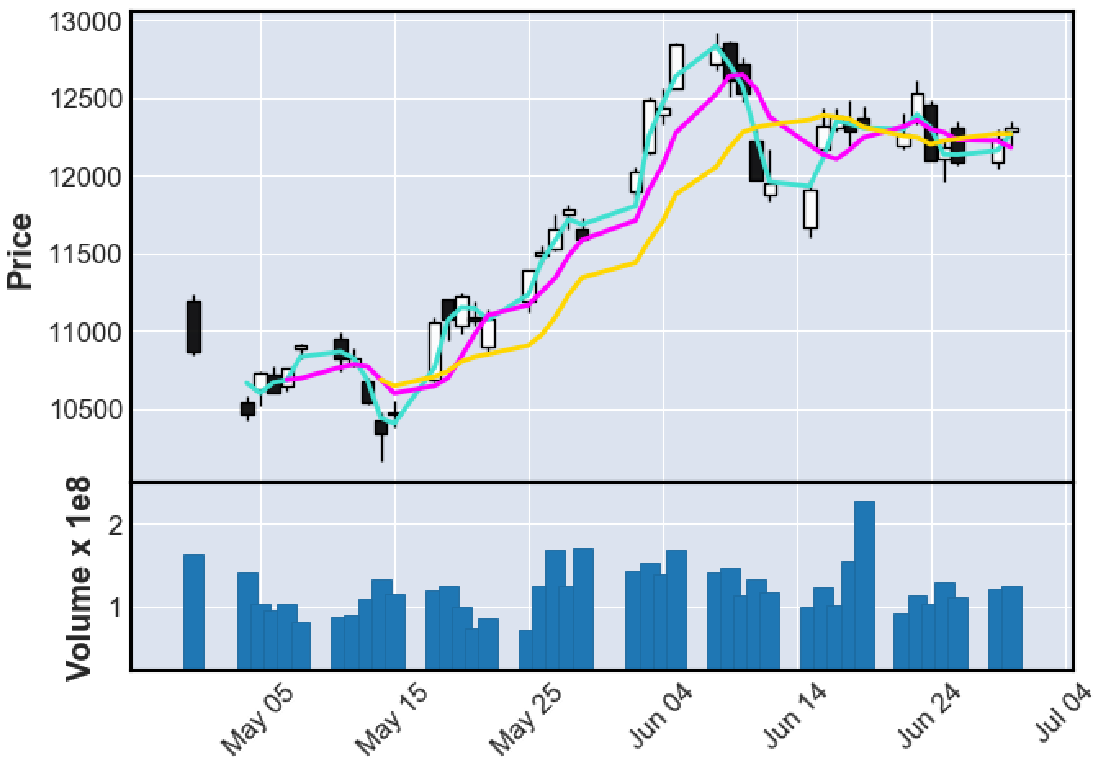

```
[32]: mpf.plot(data, type='candle', mav=(2, 5, 10)) # 绘制均线
```



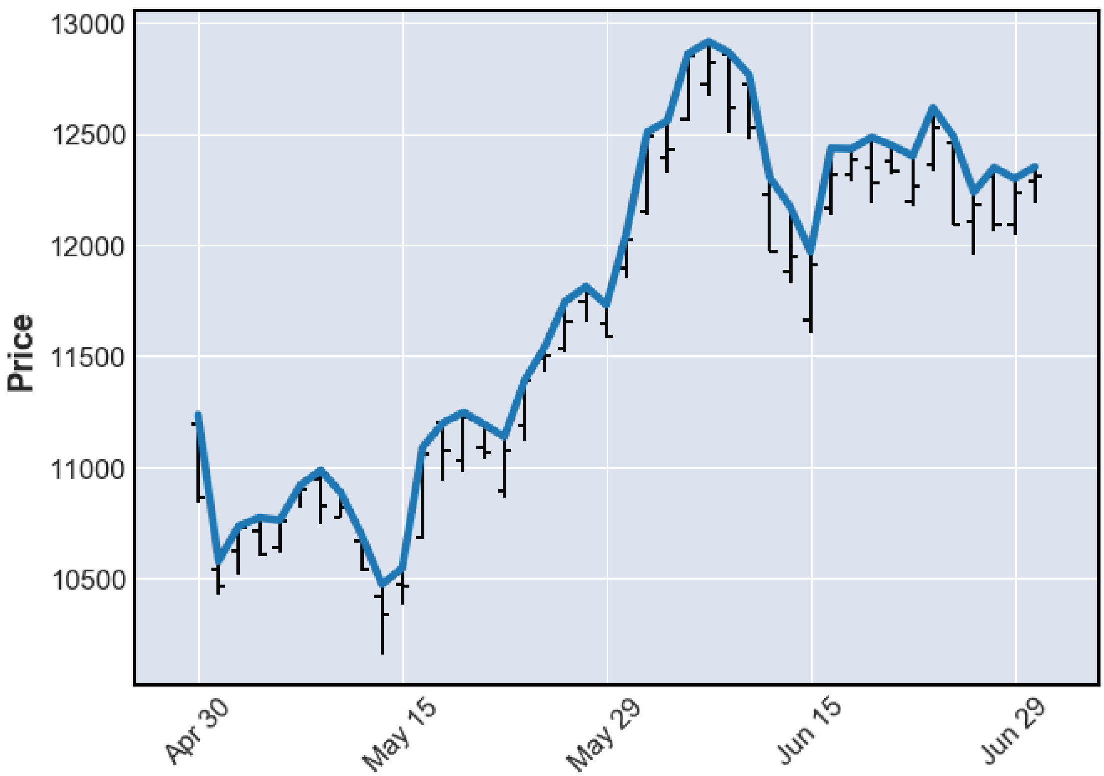
[33]:
```
mpf.plot(data, type='candle', mav=(2, 5, 10), volume=True) # 绘制成交量
```



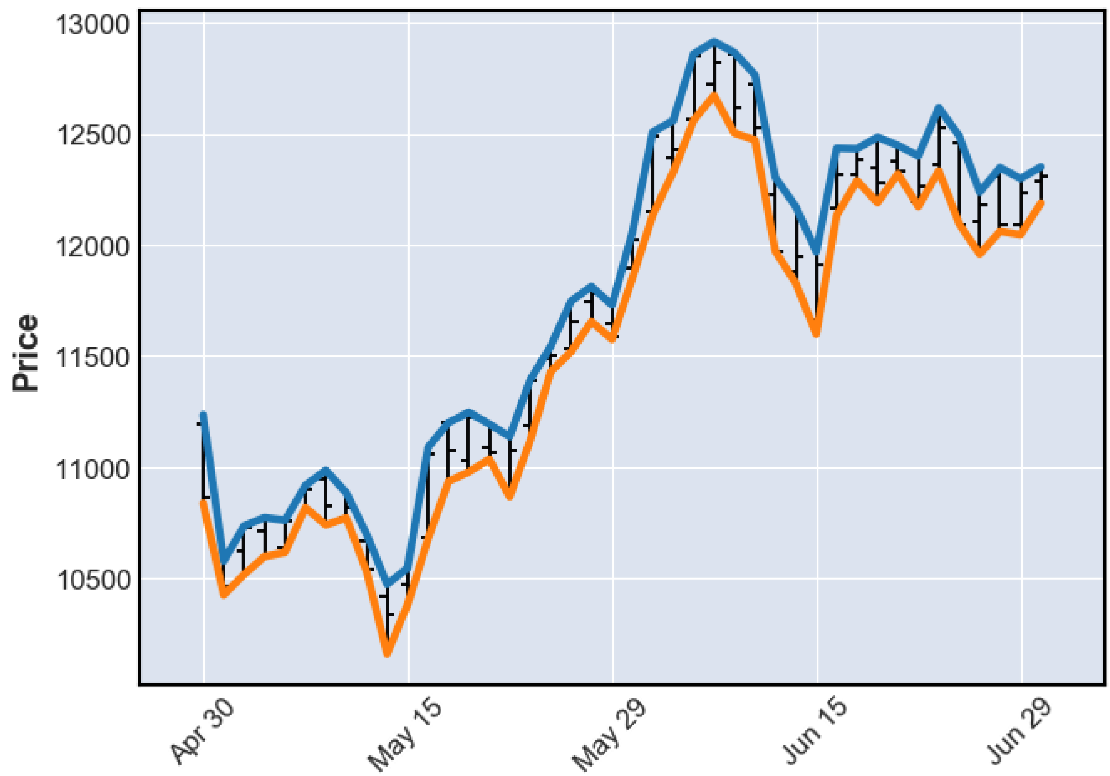
[34]:
```
mpf.plot(data,type='candle',mav=(2, 5, 10), volume=True,show_nontrading=True) #
↪显示非交易日的空白
```



[35]:
```
add_plot = mpf.make_addplot(data['High'])
mpf.plot(data, addplot=add_plot)
plt.show()  # 显示
```

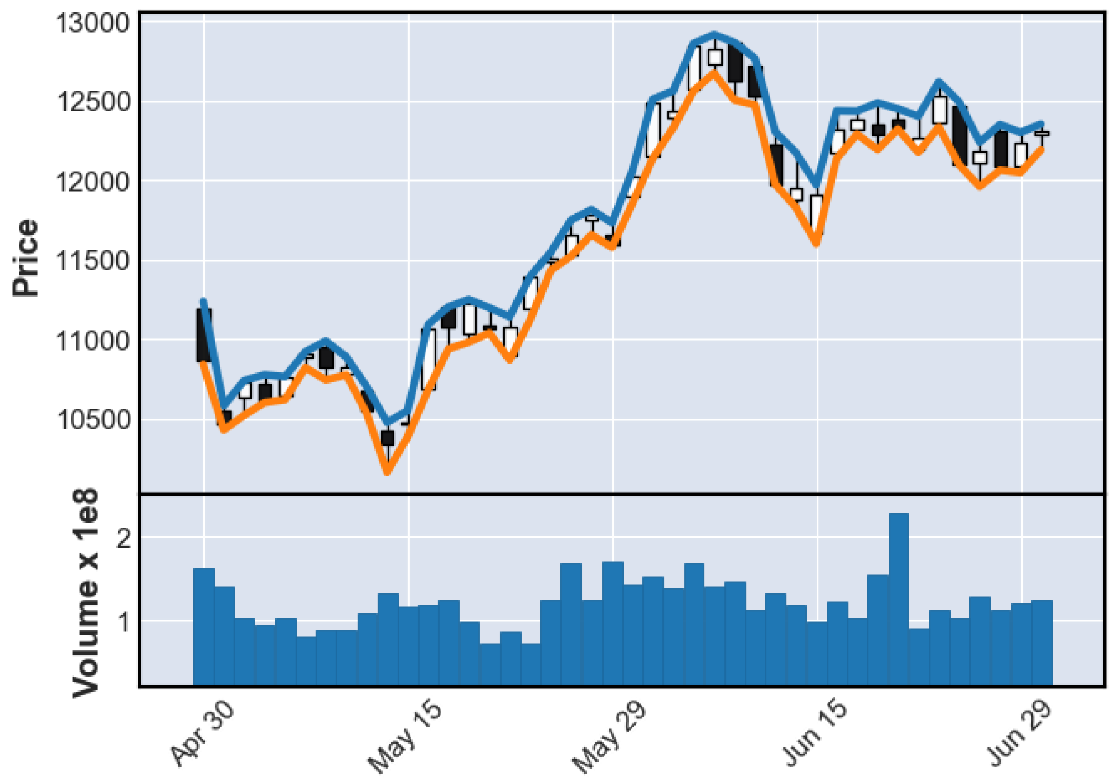


[36]:
```
add_plot = mpf.make_addplot(data[['High', 'Low']])
mpf.plot(data, addplot=add_plot)
plt.show()  # 显示
```



[37]:
```
import pandas
def data_analyze(self, data: pandas.DataFrame):
        """
        简单的数据分析，并把返回数据分析结果列表，分析的逻辑不重要，主要看如何绘制到
图形中。
        :param data:
        :return:
        """
        if data.shape[0] == 0:
            data = self.data
        s_list = []
        b_list = []
        b=-1
        for i, v in data['High'].iteritems():
            if v > data['UpperB'][i] and (b == -1 or b == 1):
                b_list.append(data['Low'][i])
```

```
                b = 0
            else:
                b_list.append(numpy.nan)  # 这里添加 nan 的目的是，对齐主图的 k
线数量
            if data['Low'][i] < data['LowerB'][i] and (b == -1 or b == 0):
                s_list.append(v)
                b = 1
            else:
                s_list.append(numpy.nan)
        return b_list, s_list

        b_list, s_list = self.data_analyze(data)
        add_plot = [
              mpf.make_addplot(b_list, scatter=True, markersize=200,␣
↪marker='^', color='y'),
              mpf.make_addplot(s_list, scatter=True, markersize=200,␣
↪marker='v', color='r'),
              mpf.make_addplot(data[['UpperB', 'LowerB']]),
              mpf.make_addplot(data['PercentB'], panel='lower', color='g',␣
↪secondary_y='auto'),]

mpf.plot(data, type='candle', addplot=add_plot, volume=True)
plt.show()  # 显示
```



1.3 3d Plotting

[38]:
```
strike = np.linspace(50, 150, 24)
ttm = np.linspace(0.5, 2.5, 24)
strike, ttm = np.meshgrid(strike, ttm)
```

[39]:
```
strike[:2]
```

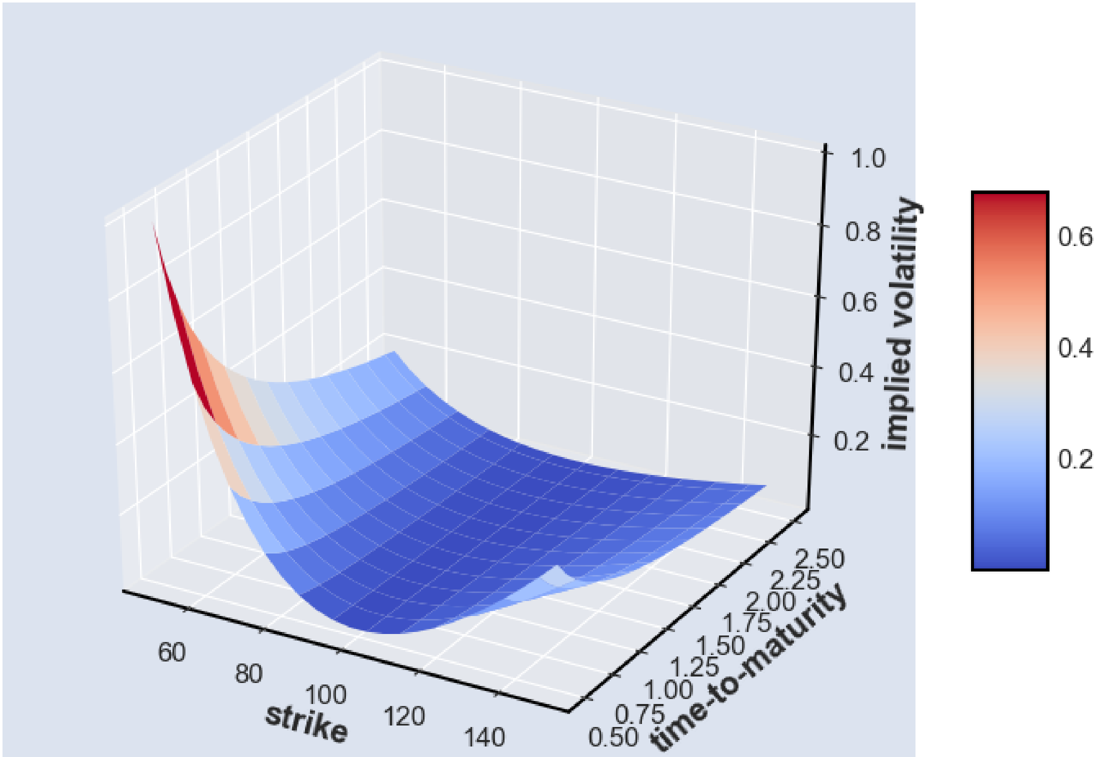
[39]:
```
array([[ 50.        ,  54.34782609,  58.69565217,  63.04347826,
         67.39130435,  71.73913043,  76.08695652,  80.43478261,
         84.7826087 ,  89.13043478,  93.47826087,  97.82608696,
        102.17391304, 106.52173913, 110.86956522, 115.2173913 ,
        119.56521739, 123.91304348, 128.26086957, 132.60869565,
        136.95652174, 141.30434783, 145.65217391, 150.        ],
       [ 50.        ,  54.34782609,  58.69565217,  63.04347826,
         67.39130435,  71.73913043,  76.08695652,  80.43478261,
         84.7826087 ,  89.13043478,  93.47826087,  97.82608696,
```

```
       102.17391304, 106.52173913, 110.86956522, 115.2173913 ,
       119.56521739, 123.91304348, 128.26086957, 132.60869565,
       136.95652174, 141.30434783, 145.65217391, 150.        ]])
```

[40]:
```
iv = (strike - 100) ** 2 / (100 * strike) / ttm
  # generate fake implied volatilities
```

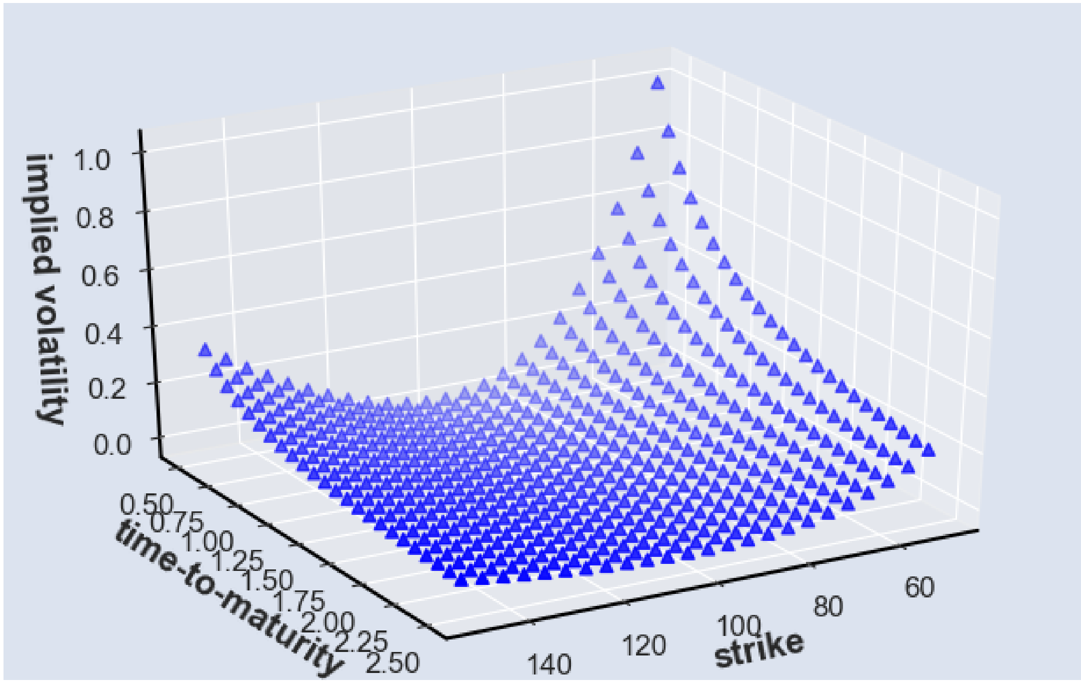
[41]:
```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(9, 6))
ax = fig.gca(projection='3d')

surf = ax.plot_surface(strike, ttm, iv, rstride=2, cstride=2,
                       cmap=plt.cm.coolwarm, linewidth=0.5,
                       antialiased=True)

ax.set_xlabel('strike')
ax.set_ylabel('time-to-maturity')
ax.set_zlabel('implied volatility')

fig.colorbar(surf, shrink=0.5, aspect=5)
# tag: matplotlib_17
# title: 3d surface plot for (fake) implied volatilities
# size: 70
```

[41]:
```
<matplotlib.colorbar.Colorbar at 0x23b9ec12408>
```



[42]:
```
fig = plt.figure(figsize=(8, 5))
ax = fig.add_subplot(111, projection='3d')
ax.view_init(30, 60)

ax.scatter(strike, ttm, iv, zdir='z', s=25,
           c='b', marker='^')

ax.set_xlabel('strike')
ax.set_ylabel('time-to-maturity')
ax.set_zlabel('implied volatility')

# tag: matplotlib_18
# title: 3d scatter plot for (fake) implied volatilities
# size: 70
```

[42]: Text(0.5, 0, 'implied volatility')


implied volatility
1.0
0.8
0.6
0.4
0.2
0.0
time-to-maturity
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
2.50
strike
140
120
100
80
60