

棕榈学院

10 天 Python 训练营讲义

第十讲



2018 年 10 月 22 日-2018 年 10 月 31 日

目录

一、蜡烛图

1. 什么是蜡烛图
2. 怎么使用 `candlestick_ohlc`
3. 对图上某一个位置的数据进行标注
4. `datetime` 的用法
5. `groupby` 的用法
6. 当需要显示的数据周期是周，月，年
7. 补充知识

附：第九讲作业答案

首先我们需要载入需要的包与数据

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
import pandas as pd
```

```
import urllib.request
```

```
import json
```

```
import warnings
```

```
warnings.filterwarnings("ignore")
```

```
def import_data(timeseries, ticker, interval, size, key_name):
```

```
    if timeseries == 'TIME_SERIES_INTRADAY':
```

```
        url =
```

```
'https://www.alphavantage.co/query?function='+timeseries+'&symbol='+t  
icker+'&interval='+interval+'&outputsize='+size+'&apikey='+api
```

```
    if timeseries == 'TIME_SERIES_DAILY':
```

```
        url =
```

```
'https://www.alphavantage.co/query?function='+timeseries+'&symbol='+t  
icker+'&outputsize='+size+'&apikey='+api
```

```
    f = urllib.request.urlopen(url)
```

```
    dat = f.read()
```

```
    js = dat.decode('utf8')
```

```
    f.close()
```

```
    parse_data = json.loads(js)
```

```
    ps = parse_data[key_name]
```

```
    df = pd.DataFrame.from_dict(ps,orient = 'index') # 'column'
```

```
    df.columns = ['Open','High','Low','Close','Volume']
```

```
    return df
```

```
def data_clean(df):
```

```
    cols = df.columns
```

```
    df[cols] = df[cols].apply(pd.to_numeric, errors = 'coerce')
```

```
df.index = pd.to_datetime(df.index, format = '%Y-%m-%d')  
return df
```

```
#api = '.....'  
timeseries = 'TIME_SERIES_DAILY'  
ticker_list = ['AAPL','MSFT','GOOG','FB']  
interval = ''  
size = 'compact'  
key_name = 'Time Series (Daily)'
```

注：这里记得填入自己的 API。

```
stock_list = {}  
for ticker in ticker_list:  
    temp_dat = import_data(timeseries, ticker, interval, size, key_name)  
    stock_list[ticker] = data_clean(temp_dat)
```

```
stock_list.keys()  
dict_keys(['AAPL', 'MSFT', 'GOOG', 'FB'])
```

Tip：对于上面这一块内容还不太熟悉的童鞋建议再复习复习第九讲的内容~

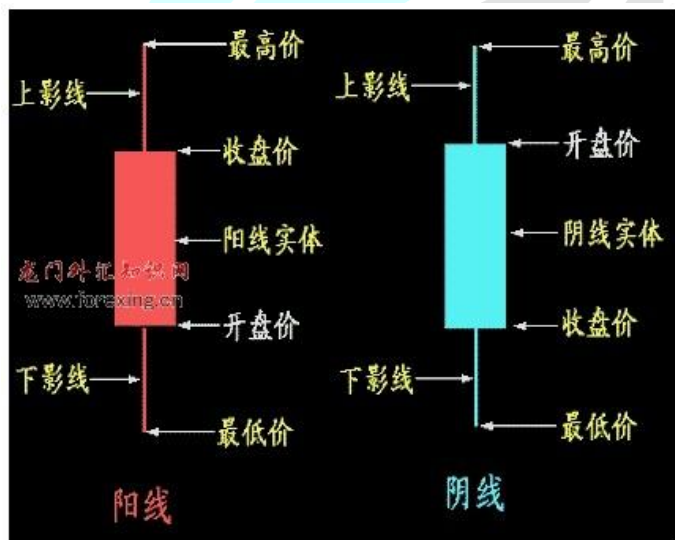
一、 蜡烛图

1. 什么是蜡烛图

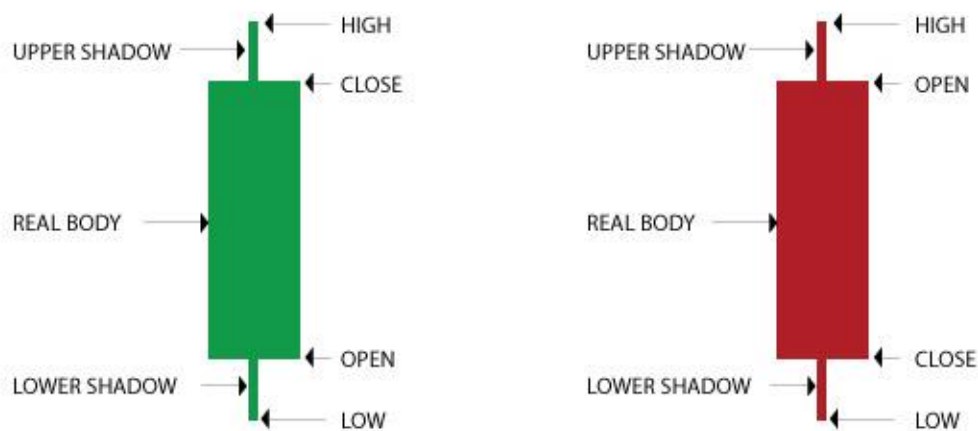
蜡烛图又叫 K 线图，起源于日本德川幕府时代。被当时日本米市的商人用来记录米市的行情与价格波动，后被引入到股市及期货市场。股市及期货市场中的 K 线图的画法包含四个数据，即开盘价、最高价、最低价、收盘价，所有的 k 线都是围绕这四个数据展开，反映大势的状况和价格信息。



K 线最上方的一条细线称为上影线，中间的一条粗线为实体。下面的一条细线为下影线。当收盘价高于开盘价，也就是股价走势呈上升趋势时，我们称这种情况下的 K 线为阳线，中部的实体以空白或红色表示。这时，上影线的长度表示最高价和收盘价之间的价差，实体的长短代表收盘价与开盘价之间的价差，下影线的长度则代表开盘价和最低价之间的差距。



注：下图为美国股市的 k 线



首先我们载入需要安装的包和函数

```
from matplotlib.dates import DateFormatter, WeekdayLocator, DayLocator,
MONDAY, date2num
from matplotlib.finance import candlestick_ohlc
import numpy as np
```

Tip : 如果你的 python 是 2.2.0 版本及以上，请运行一下代码。因为从 matplotlib 2.2.0 版本开始，matplotlib.finance 已经从 matplotlib 中剥离了，需要单独安装 mpl_finance 这个包了。

```
!pip install --user mpl_finance
```

```
from matplotlib.dates import DateFormatter, WeekdayLocator, DayLocator,
MONDAY, date2num
from mpl_finance import candlestick_ohlc
import numpy as np
```

2. 如何使用 candlestick_ohlc

当我们看到一个没有见过的函数时，首先应当去查看它的 documentation。

下面是 candlestick_ohlc 的 documentation：

https://matplotlib.org/api/finance_api.html

```
matplotlib.finance.candlestick_ohlc(ax, quotes, width=0.2, colorup='k',  
colordown='r', alpha=1.0)
```

注：

- ② 前两个是一定要被填入的参数，后面的参数不赋值也有默认的设置。
- ②第一个参数是 `ax`，指的是 `Axes`，这一个小图的位置。
- ③第二个参数是 `quotes`，里面填入的是下载下来的数据内的必要信息。我们需要把它写成 `sequence of sequences`，并且要使用 `date2num` 函数。

quotes : sequence of (time, open, high, low, close, ...) sequences

As long as the first 5 elements are these values, the record can be as long as you want (e.g., it may store volume).

time must be in float days format - see `date2num`

A sequence of sequences :

```
[(index1, open1, high1, low1, close1),(index2, open2, high2, low2,  
close2),...,(indexN, openN, highN, lowN, closeN)]
```

例：

```
data = stock_list['AAPL']
```

```
data.head()
```

	Open	High	Low	Close	Volume
2018-06-05	193.065	193.94	192.360	193.31	21565963
2018-06-06	193.630	194.08	191.920	193.98	20933619
2018-06-07	194.140	194.20	192.335	193.46	21347180
2018-06-08	191.170	192.00	189.770	191.70	26656799
2018-06-11	191.350	191.97	190.210	191.23	18308460

```
list1 = [1,2,3]
```

```
list2 = [3,4,5]
```

```
list3 = [9,8,7]
```

```
list(zip(list1,list2,list3))
```

```
[(1, 3, 9), (2, 4, 8), (3, 5, 7)]
```

注：

- ① zip()可以把多个列表或者多个数列里每个元素对应位置的数集合在一起。
- ② 首先这三个列表要一样长，把每个位置相同索引的数放到一起。
- ③ 这里把我们原来的数据变成 equence of sequences 的格式。

quotes =

```
list(zip(list(date2num(data.index.tolist())),data['Open'].tolist(),data['High'].tolist(),data['Low'].tolist(),data['Close'].tolist()))
```

注：

- ① 用 tolist()把整个数据都变成一个 list。
- ② 把每一列的数据分别填入。

```
fig=plt.figure(figsize=(15,9))  
ax=fig.add_subplot(1,1,1)  
candlestick_ohlc(ax,quotes,colorup='red',colordown='green',width=1*.4)  
plt.title('AAPL')  
plt.show()
```



对 x 轴进行改写，使它能够反映数据的时间：

```
mondays = WeekdayLocator(MONDAY)
alldays = DayLocator()
weekFormatter = DateFormatter('%b %d')
```

```
fig = plt.figure(figsize = (15, 9))
ax = fig.add_subplot(1,1,1)
```

```
ax.xaxis.set_major_locator(mondays)
ax.xaxis.set_minor_locator(alldays)
ax.xaxis.set_major_formatter(weekFormatter)
```

```
candlestick_ohlc(ax, quotes, colorup = 'red',colordown = 'green', width =
1*.4)
```

```
plt.title('AAPL')
plt.show()
```



注：

- ① WeekdayLocator()表示需要定位到哪一天，这里填入 MONDAY 表示显示每一周的周一
- ② DayLocator()表示在横坐标上以一天为最小的单位。
- ③ DateFormatter('%b %d')表示把日期填成什么样的形式，%b 表示用英文书写来表示月份，%d 表示用数字表示哪一天。还可以加入%Y 表示年份。例如 Oct 12,2018。
- ④ ax.xaxis.set_major_locator(mondays)，
ax.xaxis.set_minor_locator(alldays)，
ax.xaxis.set_major_formatter(weekFormatter)表示在 x 周上使用刚刚设定的条件。

3. 对图上某一个位置的数据进行标注

以标记 9 月 11 日为例：

```
dt = date2num(pd.to_datetime('2018-09-11', format = '%Y-%m-%d'))  
ax.annotate(str(data.loc['2018-09-11','High']),xy = (dt-2,227))  
ax.annotate(str(data.loc['2018-09-11','Low']),xy = (dt-2,215))
```

把它加到原来的代码中：



我们可以把这些都写成一个方程，把中间我们想要把它变成变量的部分拿出来，由此我们定义的方程可以兼顾各种不同的情况，能够通过一个方程画出不同的图形，以及有很多不同的功能。

在这里我们需要考虑的是，希望它有哪些变量。

```
def draw_candlestick(data, ticker, label_date = None, x1 = 0, x2 = 0):
    mondays = WeekdayLocator(MONDAY)
    alldays = DayLocator()
    weekFormatter = DateFormatter('%b %d') #Oct 12, 2018

    fig = plt.figure(figsize = (15, 9 ))
    ax = fig.add_subplot(1,1,1)

    ax.xaxis.set_major_locator(mondays)
    ax.xaxis.set_minor_locator(alldays)
    ax.xaxis.set_major_formatter(weekFormatter)

    quotes =
list(zip(list(date2num(data.index.tolist())),data['Open'].tolist(),data['High'].tolist(),data['Low'].tolist(),data['Close'].tolist()))

    candlestick_ohlc(ax, quotes, colorup = 'red',colordown = 'green', width
= 1)

    if label_date != None:
        dt = date2num(pd.to_datetime(label_date, format = '%Y-%m-%d'))
        ax.annotate(str(data.loc[label_date,'High']),xy = (dt-5,x1))
        ax.annotate(str(data.loc[label_date,'Low']),xy = (dt-5,x2))

    plt.title(ticker)
    plt.show()
```

4. datetime 的用法

<https://docs.python.org/3/library/datetime.html>

`date.isocalendar()`

Return a 3-tuple, (ISO year, ISO week number, ISO weekday).

The ISO calendar is a widely used variant of the Gregorian calendar. See <https://www.staff.science.uu.nl/~gent0113/calendar/isocalendar.htm> for a good explanation.

The ISO year consists of 52 or 53 full weeks, and where a week starts on a Monday and ends on a Sunday. The first week of an ISO year is the first (Gregorian) calendar week of a year containing a Thursday. This is called week number 1, and the ISO year of that Thursday is the same as its Gregorian year.

For example, 2004 begins on a Thursday, so the first week of ISO year 2004 begins on Monday, 29 Dec 2003 and ends on Sunday, 4 Jan 2004, so that `date(2003, 12, 29).isocalendar() == (2004, 1, 1)` and `date(2004, 1, 4).isocalendar() == (2004, 1, 7)`.

通过这个函数可以返回三个信息：年份，这一天是这一年中的哪一个星期中的一天以及这一个星期中的哪一天。所以能够以此为参照，帮助我们整理以天为单位的数据。

例：

```
data['week'] = pd.to_datetime(data.index).map(lambda x : x.isocalendar()[1])
data.head(10)
```

	Open	High	Low	Close	Volume	week
2018-06-05	193.065	193.940	192.360	193.31	21565963	23
2018-06-06	193.630	194.080	191.920	193.98	20933619	23
2018-06-07	194.140	194.200	192.335	193.46	21347180	23
2018-06-08	191.170	192.000	189.770	191.70	26656799	23
2018-06-11	191.350	191.970	190.210	191.23	18308460	24
2018-06-12	191.385	192.611	191.150	192.28	16911141	24
2018-06-13	192.420	192.880	190.440	190.70	21638393	24
2018-06-14	191.550	191.570	190.220	190.80	21610074	24
2018-06-15	190.030	190.160	188.260	188.84	61719160	24
2018-06-18	187.880	189.220	187.200	188.74	18484865	25

5. groupby 的用法

<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.groupby.html>

把数据按照某一些基准，把一个大的 DataFrame 分割成几个小的 DataFrame。

例：

(结合 datetime)

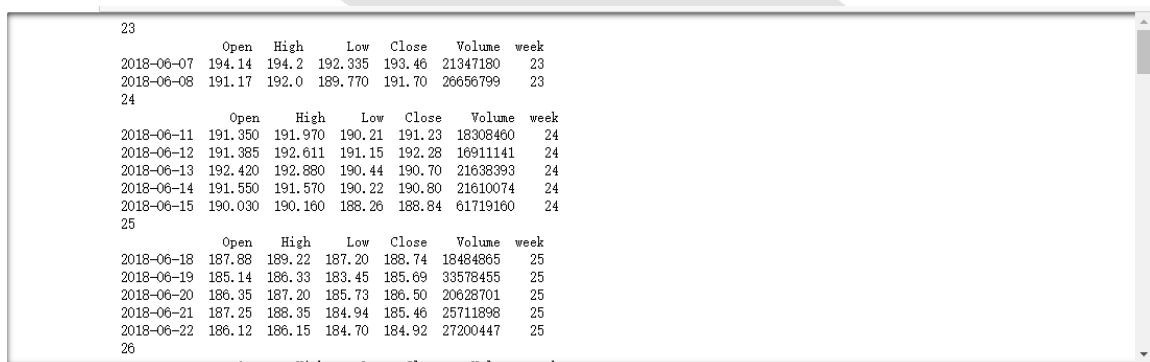
```
data['week'] = pd.to_datetime(data.index).map(lambda x : x.isocalendar()[1])
```

```
grouped = data.groupby(['week'])
```

```
for i,x in grouped:
```

```
    print(i)
```

```
    print(x)
```



	Open	High	Low	Close	Volume	week
2018-06-07	194.14	194.2	192.335	193.46	21347180	23
2018-06-08	191.17	192.0	189.770	191.70	26656799	23

	Open	High	Low	Close	Volume	week
2018-06-11	191.350	191.970	190.21	191.23	18308460	24
2018-06-12	191.385	192.611	191.15	192.28	16911141	24
2018-06-13	192.420	192.880	190.44	190.70	21638393	24
2018-06-14	191.550	191.570	190.22	190.80	21610074	24
2018-06-15	190.030	190.160	188.26	188.84	61719160	24

	Open	High	Low	Close	Volume	week
2018-06-18	187.88	189.22	187.20	188.74	18484865	25
2018-06-19	185.14	186.33	183.45	185.69	33578455	25
2018-06-20	186.35	187.20	185.73	186.50	20628701	25
2018-06-21	187.25	188.35	184.94	185.46	25711898	25
2018-06-22	186.12	186.15	184.70	184.92	27200447	25

注：i 指的是以什么标准进行分组，x 指的是以这个标准进行分组后小的 DataFrame 长成什么样子。

6. 当需要显示的数据周期是周，月，年

例：

```
plotdata = pd.DataFrame({'Open':[],'High':[],'Low':[],'Close':[]})
```

```
for i,x in grouped:
```

```
    plotdata = plotdata.append(pd.DataFrame({'Open': x.iloc[0,0],  
                                              'High': max(x.High),  
                                              'Low': min(x.Low),  
                                              'Close': x.iloc[-1,3]},  
                                index = [x.index[0]]))
```

```
draw_candlestick(plotdata, 'AAPL', label_date = None, x1 = 0, x2 = 0)
```



注：

- ① 我们把前面的代码写成方程之后仅仅是要对画的数据进行更改的话，我们只需要把变量填入函数方程就可以了。
- ② 当我们有不同的需要画不同的图时，我们需要对本身的数据进行整理，这里展示的是一种有效的数据整理方法。即当我们拿到一个数据时，数据对应了几个类型，数据的类型是打乱的。我们需要以某一行的类型为标准时，首先要有一列能够区分这些数据，在这里我们以每一行属于哪一周为标记。再使用 `grouped by`，它能把整个大的 `DataFrame` 按照标准拆分成一组一组小的 `DataFrame`，再利用一个 `for` 循环，在循环内对每一组 `DataFrame` 做一些设定的操作。
- ③ 童鞋们还可以利用上面学过的知识添加一些注释。

7.补充知识：

在股价价格信息变动分析中，有一个很重要的指数，移动平均数。移动平均数指的是指定时间段，对时间序列数据进行移动计算平均值，它是一个移动的过程。

例：

```
data['7d'] = data['Close'].rolling(window = 7, center = False).mean()
```

```
Out[53]: 2018-06-05      NaN
2018-06-06      NaN
2018-06-07      NaN
2018-06-08      NaN
2018-06-11      NaN
2018-06-12      NaN
2018-06-13    192.380000
2018-06-14    192.021429
2018-06-15    191.287143
2018-06-18    190.612857
2018-06-19    189.754286
2018-06-20    189.078571
2018-06-21    188.104286
2018-06-22    187.278571
2018-06-25    186.045714
2018-06-26    185.415714
2018-06-27    184.761429
2018-06-28    184.734286
2018-06-29    184.535714
2018-07-02    184.781429
2018-07-03    184.638571
2018-07-05    185.100000
2018-07-06    185.605714
2018-07-09    186.522857
2018-07-10    187.215714
2018-07-11    187.611429
2018-07-12    188.161429
2018-07-13    189.220000
2018-07-16    190.007143
2018-07-17    190.504286
```

注：rolling()用于计算移动平均数，windows 指的是需要几天的移动平均数。

下面是我们加入 7 天和 15 天的移动平均数后的结果



注：这些线用于判断股票价格是否有上涨趋势。

今日作业：

Q1: 延用上次作业的代码，写 `data_clean` 的函数并且使用它来清理函数。用 API 获得一个你喜欢的股票的 daily prices。把清理好的数据取名为 `data`。

Q2:

```
from matplotlib.dates import DateFormatter, WeekdayLocator, DayLocator, MONDAY, date2num
```

```
from matplotlib.finance import candlestick_ohlc
```

```
import numpy as np
```

用 `data`，仿照课上的写法，画出只有每个月数据的蜡烛图。

提示，这里 <https://docs.python.org/3/library/datetime.html> 有找 `datetime` 月份的函数。

(选做) Q3: 改写 `draw_candlestick`, 使得它有一个新的参数是 `lag`, `lag` 可以被输入 `'day'`, `'week'`, `'month'`，并且画出与之相对应的蜡烛图。

注：大家如果有兴趣的话还可以练习如何加上移动平均数~~

第九讲作业答案：

Q1：

```
def import_data(timeseries, ticker, interval, size, key_name):
```

```
    if timeseries == 'TIME_SERIES_INTRADAY':
```

```
        url =
```

```
'https://www.alphavantage.co/query?function='+timeseries+'&symbol='+ticker+'&interval='+interval+'&outputsize='+size+'&apikey='+api
```

```
    if timeseries == 'TIME_SERIES_DAILY':
```



```
url =
'https://www.alphavantage.co/query?function='+timeseries+'&symbol='+t
icker+'&outputsized='+size+'&apikey='+api
f = urllib.request.urlopen(url)
dat = f.read()
js = dat.decode('utf8')
f.close()
parse_data = json.loads(js)
ps = parse_data[key_name]
df = pd.DataFrame.from_dict(ps,orient = 'index') #'column'
df.columns = ['Open','High','Low','Close','Volume']
return df

timeseries = 'TIME_SERIES_INTRADAY'
ticker = 'AAPL'
interval = '5min'
size = 'compact'
key_name = 'Time Series (5min)'
api = 'E8AYWP6L8TGRO3OF'
apple = import_data(timeseries, ticker, interval, size, key_name)

apple.index = pd.to_datetime(apple.index, format = '%Y-%m-%d %H:%M:%S')

apple.head()
```

	Open	High	Low	Close	Volume
2018-10-25 14:10:00	219.0700	219.7300	219.0300	219.6100	238556
2018-10-25 14:15:00	219.6900	219.8000	219.5600	219.7112	230211
2018-10-25 14:20:00	219.7500	219.9300	219.6800	219.8600	228136
2018-10-25 14:25:00	219.8500	219.9300	219.5900	219.8100	230370
2018-10-25 14:30:00	219.7978	220.2600	219.7978	220.1500	270164

Q2:

```
apple['Date'] = apple.index.date
```

```
apple['Time'] = apple.index.time
```

```
apple['Volume'] = apple['Volume'].apply(pd.to_numeric, errors = 'coerce')
```

```
apple.head()
```

	Open	High	Low	Close	Volume	Date	Time
2018-10-25 14:10:00	219.0700	219.7300	219.0300	219.6100	238556	2018-10-25	14:10:00
2018-10-25 14:15:00	219.6900	219.8000	219.5600	219.7112	230211	2018-10-25	14:15:00
2018-10-25 14:20:00	219.7500	219.9300	219.6800	219.8600	228136	2018-10-25	14:20:00
2018-10-25 14:25:00	219.8500	219.9300	219.5900	219.8100	230370	2018-10-25	14:25:00
2018-10-25 14:30:00	219.7978	220.2600	219.7978	220.1500	270164	2018-10-25	14:30:00

Q3:

```
apple['Volume_pct1'] = apple['Volume'].pct_change()
```

```
apple['Volume_pct2'] = apple['Volume']/apple['Volume'].shift(1) - 1
```

```
apple['Volume_pct3'] = apple.loc[:,['Volume']].apply(lambda x: x/x.shift(1) - 1)
```

```
apple.head()
```

	Open	High	Low	Close	Volume	Date	Time	Volume_pct1	Volume_pct2	Volume_pct3
2018-10-25 14:10:00	219.0700	219.7300	219.0300	219.6100	238556	2018-10-25	14:10:00	NaN	NaN	NaN
2018-10-25 14:15:00	219.6900	219.8000	219.5600	219.7112	230211	2018-10-25	14:15:00	-0.034981	-0.034981	-0.034981
2018-10-25 14:20:00	219.7500	219.9300	219.6800	219.8600	228136	2018-10-25	14:20:00	-0.009013	-0.009013	-0.009013
2018-10-25 14:25:00	219.8500	219.9300	219.5900	219.8100	230370	2018-10-25	14:25:00	0.009792	0.009792	0.009792
2018-10-25 14:30:00	219.7978	220.2600	219.7978	220.1500	270164	2018-10-25	14:30:00	0.172740	0.172740	0.172740



扫码关注棕榈学院，解锁更多精彩课程