Chapter 4 – Pandas金融场景

Frank Ziwei Zhang
School of Finance

上海对外经贸大学
SHANGHAI UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

Contents

4.1 Pandas数据结构

4.1.1序列

序列（series）是一个类似于一维数组的数据结构，不过它是由两部分构成，第1部分是标签（label）或者索引（index），第2部分是对应的数值，注意这两部分的长度必须一致。大多数NumPy的函数可以直接应用于序列的运算中。此外，可以通过第3章讨论的数组来快速生成序列。

【例4-1】沿用3.1节的例3-1，针对表4-1描绘的2018年9月3日至9月7日这5个交易日中相关股票的涨跌幅情况，演示生成包括5个交易日和股票涨跌幅情况的序列。

股票简称	9月3日	9月4日	9月5日	9月6日	9月7日
中国石油	0.3731%	2.1066%	-0.4854%	0.6098%	-0.6060%
工商银行	-0.1838%	0.1842%	-1.6544%	-0.3738%	0.3752%
上汽集团	-0.3087%	-0.0344%	-3.3391%	0.7123%	0.4597%
宝钢股份	-2.4112%	1.1704%	-2.9563%	-1.4570%	1.6129%

注意，由于 Pandas是 Python的外部第三方模块，在使用前需要导入，并且查询 Pandas的版本号信息:

```
import pandas as pd
pd.__version__
Out[2]: '1.1.3'
```

针对表4-1中2018年9月3日的数据生成序列，生成序列需要运用 Series函数，要输入涉及的索引index。首先演示手动输入生成序列，具体的代码如下：

```
return_series1 = pd.Series(data=[0.003731,-0.001838,-0.003087,-0.024112],index=['中国石油','工商银行','上汽集团','宝钢股份'])
```

return_series1 - Series

Index	0
中国石油	0.003731
工商银行	-0.001838
上汽集团	-0.003087
宝钢股份	-0.024112

其次，借助第3章例3-5中创建的数组return_array，用数组生成针对2018年9月3日数据的序列，也可以得到和上面同样的效果。具体的代码如下：

```
import numpy as np
return_list = [0.003731, 0.021066, -0.004854, 0.006098,
-0.006060, -0.001838, 0.001842, -0.016544, -
0.003738, 0.003752, -0.003087, -0.000344, -0.033391,
0.007123, 0.004597, -0.024112, 0.011704, -
0.029563, -0.014570, 0.016129]
return_array = np.array(return_list)
return_array = return_array.reshape(4,5)

return_series2 = pd.Series(data=return_array[:,0],index=['中国石油','工商银行','
上汽集团','宝钢股份'])
```

4.1.2数据框

但是，序列有一个很致命的缺点——只能有两列，其中一列是索引列，另一列是数值列，如果希望有更多的数值列，则需要用下面讨论的数据框存放数据。

数据框（Data frame）类似于Excel表结构的数据结构，设计的初衷就是将序列从原先的一维推广至多维，数据框由3部分构成：第1部分是行索引（index），第2部分是列名（columns），第3部分是取值。

同时，创建数据框可以运用DataFrame函数，该函数的参数主要有3个：第1个参数data是输入相关的数据或者变量，第2个参数 index是输入行索引，第3个参数 columns是输入列名。

4.1.2数据框

【例4-2】沿用本章例4-1的信息，将日期作为行索引，将股票名称作为列名，运Data Frame函数创建数据框，具体的代码如下:

```
date=['2018-9-3','2018-9-4','2018-9-5','2018-9-6','2018-9-7']
stock=['中国石油','工商银行','上汽集团','宝钢股份']
return_dataframe=pd.DataFrame(data=return_array.T,index=date,columns=stock)
```

return_dataframe - DataFrame

Index	中国石油	工商银行	上汽集团	宝钢股份
2018-9-3	0.003731	-0.001838	-0.003087	-0.024112
2018-9-4	0.021066	0.001842	-0.000344	0.011704
2018-9-5	-0.004854	-0.016544	-0.033391	-0.029563
2018-9-6	0.006098	-0.003738	0.007123	-0.01457
2018-9-7	-0.00606	0.003752	0.004597	0.016129

4.1.2数据框

可以把生成的数据框以Excel、CSV（逗号分隔值）、txt文本等格式导出，具体运用导出函数 to_excel, to_csv，并且需要在函数中添加导出文件存放的路径和带YUL:格式的文件名。

【例4-3】针对例4-2生成的数据框return_dataframe，依次以Excel、CSV、txt格式导出并存放在用户计算机的D盘，具体的代码如下:

```
return_dataframe.to_excel('D:/myTempData_Return.xlsx')
return_dataframe.to_csv('D:/myTempData_Return.csv')
return_dataframe.to_csv('D:/myTempData_Return.txt')
```

	A	B	C	D	E
1		中国石油	工商银行	上汽集团	宝钢股份
2	2018/9/3	0.003731	-0.001838	-0.003087	-0.024112
3	2018/9/4	0.021066	0.001842	-0.000344	0.011704
4	2018/9/5	-0.004854	-0.016544	-0.033391	-0.029563
5	2018/9/6	0.006098	-0.003738	0.007123	-0.01457
6	2018/9/7	-0.00606	0.003752	0.004597	0.016129
7					

4.1.3 外部数据导入并直接生成数据框

函数名和主要参数	具体说明
read_table(‘文件的路径’,index_col,delimiter）	文件的路径是必须输入的，它代表了拟导入的文件所存放的计算机位置； Index_col：指定某一列作为索引列,比如，index_col=0就表示将文件第1列作为索引列； delimiter：指定了数据间的分隔符，分隔符可以是空格、制表符等
read_excel(“文件的路径”, sheetname, header,index_col）	文件的路径是必须输入的； sheetname：表示需要从Excel表中导人的工作表名称，如sheetname=’Sheet1’，表示导入工作表sheetl； header：指定列名行，默认是加header=O，也就是取工作表第一行作为指定到列名行； index_col:指定某一列作为索引列，如index_col=0就表示将第一列作为索引列；
read_csv(‘文件的路径或网址’, sep,delimiter, header, index_col	文件的路径或网址是必须输入的； sep：指定分隔符，如不指定参数，则会尝试使用逗号分隔； delimiter：表示定界符，是备选分隔符，注意如果指定该参数，则sep编写参数失效； header：用法与导入Excel表是一致的； index_col：用法与导入Excel表是一致的；

4.1.3 外部数据导入并直接生成数据框

【例4-4】以从外部 Excel文件导入沪深300指数在2018年每个交易日的开盘高点位、最低点位以及收盘点位数据并且创建数据框。图4-2是拟导入的 Excel文件的部分截图。

相关的代码如下：

```
HS300_excel1 = pd.read_excel('D:\Zhangzw\Python\Python金融数据分析\RawData\第4章\沪深300指数.xlsx',sheet_name="Sheet1",header=0,index_col=0)
```

HS300_excel1 - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2018-01-02 00:00:00	4045.21	4087.78	4045.21	4087.4
2018-01-03 00:00:00	4091.46	4140.05	4088.73	4111.39
2018-01-04 00:00:00	4114.12	4137.64	4105.89	4128.81
2018-01-05 00:00:00	4133.34	4151.28	4123.28	4138.75
2018-01-08 00:00:00	4140.85	4166.32	4127.31	4160.16
2018-01-09 00:00:00	4157.54	4191.28	4153.5	4189.3
2018-01-10 00:00:00	4187.2	4211.05	4175.14	4207.81
2018-01-11 00:00:00	4197.11	4211.8	4181.96	4205.59
2018-01-12 00:00:00	4205.14	4227.39	4199.03	4225
2018-01-15 00:00:00	4229.84	4262.93	4216.36	4225.24

4.1.3 外部数据导入并直接生成数据框

【例4-5】仅以Tushare作为例子来演示如何通过API接口在 Python中导入浦发银行（600000.SH）2018年第一季度每个交易日的开盘点位、最高点位、最低点位以及收盘点位，具体的过程如下:

首先，下载安装：

```
pip install tushare
pip install pandas  #下面两个是要用到的工具包，安装过的可以忽略
pip install lxml
```

如果是使用以前版本的，可以通过以下命令来进行升级：

```
pip install tushare –upgrade
```

其次，Tushare是一个免费开源的金融数据平台，新版本Tushare Pro更加稳定、流畅，而且将数据扩大到了股票、基金、期货、债券、外汇、行业大数据等区块链的数据。推荐大家注册一个账户，下面就是注册链接：https://tushare.pro/register?reg=397881。

4.1.3 外部数据导入并直接生成数据框

再次，获取token密匙：在刚才注册过网站的右上角点击个人主页，在接口TOKEN中我们就可以复制到token：

最后，就是设置token了

```
import tushare as ts

#方式一
ts.set_token('你刚才复制的token填在这里')
#这种方式设置token我们会吧token保存到本地，所以我们在使用的时候只需设置一次，失效之后，我们可以替换为新的token

#方式二
pro = ts.pro_api()
pro = ts.pro_api('你刚才复制的token填在这里')
这种在初始化接口的时候设置token
```

4.1.3 外部数据导入并直接生成数据框

PFYH_excel2 = pro.daily(ts_code='600000.SH', start_date='20180101', end_date='20180331')

PFYH_excel2 - DataFrame

Index	ts_code	trade_date	open	high	low	close
0	600000.SH	20180330	11.63	11.69	11.61	11.65
1	600000.SH	20180329	11.57	11.71	11.45	11.62
2	600000.SH	20180328	11.53	11.75	11.52	11.57
3	600000.SH	20180327	11.7	11.77	11.56	11.62
4	600000.SH	20180326	11.71	11.75	11.5	11.61
5	600000.SH	20180323	12	12.06	11.64	11.71
6	600000.SH	20180322	12.35	12.35	12.25	12.25
7	600000.SH	20180321	12.34	12.4	12.28	12.29
8	600000.SH	20180320	12.28	12.39	12.25	12.32
9	600000.SH	20180319	12.31	12.33	12.26	12.31

4.2 数组框的可视化

4.2.1 中文字体的可视化

由于数据框的索引、列名等可能会运用中文字体，并且在可视化的图形中也存在以中文体显示的需要。对此，可以通过输入以下的代码让系统调出中文模块：

```
from pylab import mpl  #导入子模块mpl
mpl.rcParams['font.sans-serif']=['SimHei'] #以黑体字显示中文
mpl.rcParams['axes.unicode_minus']=False #解决保存图像是负号‘_’为显示方块的问题
```

表4-3梳理了针对 Windows操作系统的中文字体对应与 Python中字体的名称，用户可以根据需要自行更换输出中文的字体格式。

Windows的字体名称	Python中对应的名称
黑体	SimHei
微软雅黑	Microsoft YaHei
微软正黑体	Microsoft JhengHei
新宋体	NSimSun
新细明体	PMingLiU
细明体	MingLiU
标楷体	DFKai-SB
仿宋	FangSong
借体	KaiTi
仿宋_GB2312	FangSong_GB2312
楷体_GB2312	KaiTi_GB2312

4.2.2 数据框可视化的函数

数据框可视化主要运用函数plot，下面就介绍该函数在运用中需要输入的主要参数。

类型参数	说 明
“line”	Line plot（折线图），模认情况下就是选择这类图形
“bar”	Vertical bar plot（条形图）
“box”	boxplot（箱线图）
“barh”	horizontal bar plot（横向条形图）
“hist”	histogram（性状图）
“kde”	Kernel Density Estimation plot（核密度估计图）
“density”	与’kde’相同
“area”	area plot（区域图）
“pie”	pie plot（饼图）
“scatter”	scatter plot（散点图）
“hexbin”	hexagonal binning plot（六边形箱图）

4.2.2 数据框可视化的函数

数据框可视化主要运用函数plot，下面就介绍该函数在运用中需要输入的主要参数。

参数kind：指图中需要显示的图形类型，可以有表4-4的图形类型参数供选择。

参数 subplots:用于判断图形中是否有子图，如果输入True就代表是，不输人（缺省情况）就表示否。

参数 sharex：如果有子图，则判断子图是否共用x轴刻度及标签，如果输入True就代表是，不输入（缺省情况）就表示否。

参数 sharey：如果有子图，则判断子图是否共用y轴刻度及标签，如果输入True就代表是，不输入（缺省情况）就表示否。

参数 layout：表示有子图的情况下，子图的行列布局情况，按照（行数，列数）的方式进行输入。比如，（2,1）就表示子图的布局是占用2行、1列。

参数 figsize：表示输出图形的尺寸大小，比如（10,8）就代表长10英寸、宽8英寸。

参数 use_index：默认用索引做x轴。

参数 title：用于生成图形的标题，并且用字符串表示。

参数 grid：表示图形是否有网格，如果选择True就代表是，不输入（缺省情况）就表示否。

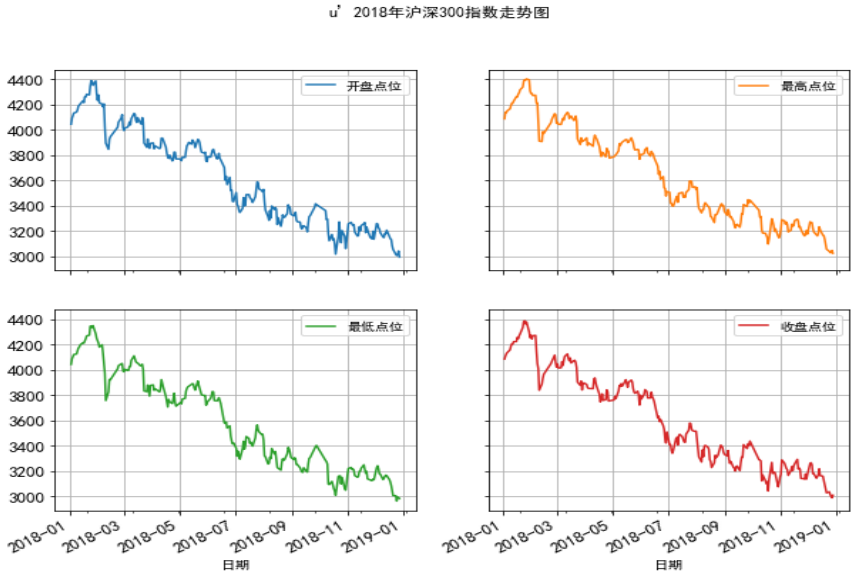
参数 fontsize：表示设置图形轴刻度的字体大小，比如 fontsize=13表示刻度字体大小是13磅。

4.2.2 数据框可视化的函数

【例4-6】沿用例4-4的沪深300指数2018年的日交易数据，对数据框进行可视化，具体的代码如下：

```
HS300_excel1.plot(kind='line',subplots=True,sharex=True,sharey=True,layout=(2,2),
figsize=(10,8),title=u'年沪深300指数走势图',grid=True,fontsize=13)
```

图4-3包含了4个子图，并且子图的布局是占用2行和2列，也就是每一行有两个子图，同时每一列也有两个子图。此外，需要注意的是，由于图形标题是以中文格式输出，因此在输入代码时，除了将中文标题以字符串输入以外，还需要在字符串前面加上小写英文字u，这样的输入方式在本书后面的可视化操作中将反复用到。



4.3 数据框内部的操作

4.3.1 描述数据框的基本性质

1、index与 columns函数

对于数据框的用户而言，往往需要查看数据框的行索引名和列名是什么，就要运用index函数查看查看行索引名、columns函数查看列名。

【例4-7】沿用例4-4中从外部Excel表导入的沪深300指数2018年的日交易数据，分别查看具体的行索引名和列名，相关的代码如下：

```
HS300_excel1.index  #查看行索引名
HS300_excel1.cloumns  #查看列名
```

2、shape与describe函数

如果用户希望知道一个数据框由多少行、多少列构成，可以运用函数shape进行查询。

【例4-8】沿用例4-4中的日交易数据，查看该数据框的行数和列数，相关的代码如下：

```
HS300_excel1.shape  #查看数据框的行数和列数
```

以上输出的结果是一个元组，元组中的第1个元素代表数据框的行数（不包括列名共有243行）。第2个元素代表数据框的列数（不包括行索引共有4列）。

4.3.1 描述数据框的基本性质

【例4-9】沿用例4-4的日交易数据，查看样本量、均值、方差、最大值、最小值、分位数等涉及时间序列的统计指标，相关的代码如下：

```
HS300_excel1.describe() #查看数据框的行数和列数
```

Out[20]:

	开盘点位	最高点位	最低点位	收盘点位
count	243.000000	243.000000	243.000000	243.000000
mean	3605.910629	3633.511737	3576.081525	3605.809573
std	379.357328	378.562382	380.805563	380.883948
min	2994.795000	3024.352500	2964.875000	2990.505700
25%	3259.739050	3281.134050	3227.896000	3265.065300
50%	3521.587600	3541.782000	3478.213400	3517.656800
75%	3897.636750	3921.321600	3875.582550	3898.566550
max	4389.454700	4403.337500	4351.487100	4389.885300

4.3.2 数据框的索引与截取

函数名	功能	利用例4-4的数据作为示例
loc	通过输入行索引的方式来输出对应的数据。 注：loc是英文locate缩写	HS300_excel1.loc['2018-12-18'] #索引输出对应数据 Out[22]: 开盘点位 3139.9493 最高点位 3165.0018 最低点位 3114.4684 收盘点位 3128.4265 Name: 2018-12-18 00:00:00, dtype: float64
iloc	通过输入行号（即具体第几行的方式来输出对应的数据，并且行号0代表第一行，依次类推注。 注：iloc是英文 index locate的缩写	HS300_excel1.iloc[5] #输出第6行 Out[23]: 开盘点位 4157.5394 最高点位 4191.2843 最低点位 4153.5021 收盘点位 4189.2977 Name: 2018-01-09 00:00:00, dtype: float64

4.3.2 数据框的索引与截取

2、一般性截取

针对金融时间序列，通常情况下是希望能够截取某一时间区间内相关变量的取值情况就类似于NumPy的数组切片操作。

【例4-10】沿用例4-4的日交易数据，进行一般性截取的演示，具体的代码如下：

```
HS300_excel1[:3] #截取数据框前3行的数据
HS300_excel1[2:8] #截取数据框第3行-第8行的数据
HS300_excel1.iloc[7:13,1:3] #截取数据框第8-13行，以及2、3列的数据
```

3、条件性截取

有时候，用户希望通过设定某些选取的条件来截取相关的数据，比如针对某个列名找出大于某一数值的数据。

【例4-11】沿用例4-4的日交易数据，找出收盘点位超过4300点的相关数据，具体的代码如下:

```
HighPrice = HS300_excel1[HS300_excel1['收盘点位']>=4300] #截取数据框
```

HighPrice - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2018-01-22 00:00:00	4276.48	4338.48	4275.9	4336.6
2018-01-23 00:00:00	4346.89	4383.57	4346.79	4382.61
2018-01-24 00:00:00	4389.45	4397.82	4349.09	4389.89
2018-01-25 00:00:00	4381.98	4392.2	4336.24	4365.08
2018-01-26 00:00:00	4352.22	4403.34	4351.49	4381.3
2018-01-29 00:00:00	4387.06	4395.91	4287.11	4302.02

3、条件性截取

【例4-12】沿用例4-4的日交易数据，选取开盘点位超过4100点并且收盘点位小于4200点的相关数据，具体的代码如下：

```
HighPrice2 = HS300_excel1[(HS300_excel1['开盘点位']>=4100)&( HS300_excel1['收盘点位']<=4200)] #截取数据框
```

HighPrice2 - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2018-01-04 00:00:00	4114.12	4137.64	4105.89	4128.81
2018-01-05 00:00:00	4133.34	4151.28	4123.28	4138.75
2018-01-08 00:00:00	4140.85	4166.32	4127.31	4160.16
2018-01-09 00:00:00	4157.54	4191.28	4153.5	4189.3
2018-02-06 00:00:00	4182.33	4211.52	4131.56	4148.89
2018-02-07 00:00:00	4205.74	4212.57	4048.42	4050.5
2018-02-27 00:00:00	4120.87	4120.87	4051.81	4058.98
2018-03-12 00:00:00	4130.71	4139.53	4112.77	4127.67
2018-03-13 00:00:00	4124.33	4130.28	4087.56	4091.25

4.3.3 数据框的排序

1、按行索引大小排序

如果用户希望按照行索引的大小排序，比如，针对金融时间序列按照时间由近至远排序等，就需要运用到sort index函数，并且有一个重要的参数ascending，如果ascending=True（默认情况）则表示由小到大排序，相反如果ascending=False则表示由大到小排序。

【例4-13】沿用例4-4的日交易数据，分别输出按照行索引由小到大、由大到小排序的结果，具体的代码如下：

```
AscendingPrice = HS300_excel1.sort_index(axis=0,ascending=True)
```

AscendingPrice - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2018-01-02 00:00:00	4045.21	4087.78	4045.21	4087.4
2018-01-03 00:00:00	4091.46	4140.05	4088.73	4111.39
2018-01-04 00:00:00	4114.12	4137.64	4105.89	4128.81
2018-01-05 00:00:00	4133.34	4151.28	4123.28	4138.75
2018-01-08 00:00:00	4140.85	4166.32	4127.31	4160.16
2018-01-09 00:00:00	4157.54	4191.28	4153.5	4189.3
2018-01-10 00:00:00	4187.2	4211.05	4175.14	4207.81
2018-01-11 00:00:00	4197.11	4211.8	4181.96	4205.59
2018-01-12 00:00:00	4205.14	4227.39	4199.03	4225
2018-01-15 00:00:00	4229.84	4262.93	4216.36	4225.24

2、按列名对应的数值大小进行排序

用户也可以按照数据框中，某个列名对应的数值大小进行排序，这时需要运用到sort_values函数，并且重点运用参数by=”列名”以及ascending=True（默认情况，由小到大排序）或者ascending=False（由大到小排序）。

【例4-14】沿用例4-4的日交易数据，分别输出按照收盘点位由小到大排序、最高点位由大到小排序的结果，具体的代码如下

```
AscendingPrice3 = HS300_excel1.sort_values(by='收盘点位',ascending=True)
```

AscendingPrice3 - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2018-12-27 00:00 日期	042.95	3047.23	2990.51	2990.51
2018-12-26 00:00:00	3012.87	3029.06	2996.48	3002.03
2018-12-28 00:00:00	2994.8	3024.35	2984.82	3010.65
2018-12-25 00:00:00	3006.88	3030.14	2964.88	3017.28
2018-12-21 00:00:00	3055.33	3057.4	3007.61	3029.4
2018-12-24 00:00:00	3015.5	3040.35	3007.33	3038.2
2018-10-18 00:00:00	3097.79	3097.79	3043.38	3044.39
2018-12-20 00:00:00	3083.54	3097.72	3044.29	3067.42
2018-10-29 00:00:00	3157.49	3158.81	3062.58	3076.89
2018-12-19 00:00:00	3133.74	3137.01	3086.76	3091.13

4.3.4 数据框的修改

1、修改列名

修改列名需要运用rename函数，并且输入重要的参数columns={‘原名称'：'新名称’}

【例4-15】沿用例4-4的日交易数据，将其中一个列名“收盘点位”修改为“收盘价格”，具体代码如下：

```
HS300_columnschange = HS300_excel1.rename(columns={'收盘点位':收盘价格})
```

HS300_columnschange - DataFrame

日期	开盘点位	最高点位	最低点位	收盘价格
2018-01-02 00:00:00	4045.21	4087.78	4045.21	4087.4
2018-01-03 00:00:00	4091.46	4140.05	4088.73	4111.39
2018-01-04 00:00:00	4114.12	4137.64	4105.89	4128.81
2018-01-05 00:00:00	4133.34	4151.28	4123.28	4138.75
2018-01-08 00:00:00	4140.85	4166.32	4127.31	4160.16

2、缺失值的处理

【例4-16】导入“浦发银行”“上海机场”“中国石化”这3只股票在2018年3月6日收盘价的数据并且生成新的数据框，假定在这个数据框中存在缺省值的情况，需要对这些缺省值进行处理，具体的代码如下：

```
stock = pd.read_excel('D:\Zhangzw\Python\Python金融数据分析\RawData\第4章\关于2018年3月份股票价格数据.xlsx',sheet_name="Sheet1",header=0,index_col=0)
```

stock - DataFrame

日期	浦发银行	上海机场	中国石化
2018-03-01 00:00:00	12.47	49.33	6.39
2018-03-02 00:00:00	12.41	50.35	6.33
2018-03-05 00:00:00	12.42	50.22	6.39
2018-03-06 00:00:00	12.49	nan	6.47
2018-03-07 00:00:00	12.49	49.22	6.44
2018-03-08 00:00:00	12.47	49.64	nan
2018-03-09 00:00:00	12.48	50.83	6.39
2018-03-12 00:00:00	12.5	50.09	6.45
2018-03-13 00:00:00	12.49	50.54	6.37
2018-03-14 00:00:00	12.39	49.84	6.42
2018-03-15 00:00:00	12.39	50	6.42
2018-03-16 00:00:00	12.3	49.34	6.42
2018-03-19 00:00:00	nan	50.14	6.59

4.3.4 数据框的修改

表 4-6　对例 4-16 涉及的缺失值进行处理以及相应的代码

处理缺失值的方法：删除法	处理缺失值的方法：补齐法（赋值零）
stock_dropna = stock.dropna()	stock_fillzero = stock.fillna(value=0)

stock_dropna - DataFrame

日期	浦发银行	上海机场	中国石化
2018-03-01 00:00:00	12.47	49.33	6.39
2018-03-02 00:00:00	12.41	50.35	6.33
2018-03-05 00:00:00	12.42	50.22	6.39
2018-03-07 00:00:00	12.49	49.22	6.44
2018-03-09 00:00:00	12.48	50.83	6.39
2018-03-12 00:00:00	12.5	50.09	6.45
2018-03-13 00:00:00	12.49	50.54	6.37
2018-03-14 00:00:00	12.39	49.84	6.42
2018-03-15 00:00:00	12.39	50	6.42
2018-03-16 00:00:00	12.3	49.34	6.42

stock_fillzero - DataFrame

日期	浦发银行	上海机场	中国石化
2018-03-01 00:00:00	12.47	49.33	6.39
2018-03-02 00:00:00	12.41	50.35	6.33
2018-03-05 00:00:00	12.42	50.22	6.39
2018-03-06 00:00:00	12.49	0	6.47
2018-03-07 00:00:00	12.49	49.22	6.44
2018-03-08 00:00:00	12.47	49.64	0
2018-03-09 00:00:00	12.48	50.83	6.39
2018-03-12 00:00:00	12.5	50.09	6.45
2018-03-13 00:00:00	12.49	50.54	6.37
2018-03-14 00:00:00	12.39	49.84	6.42

4.3.4 数据框的修改

处理缺失值的方法：补齐法（向前填充）

```
stock_ffill = stock.fillna(method='ffill')
```

stock_ffill - DataFrame

日期	浦发银行	上海机场	中国石化
2018-03-01 00:00:00	12.47	49.33	6.39
2018-03-02 00:00:00	12.41	50.35	6.33
2018-03-05 00:00:00	12.42	50.22	6.39
2018-03-06 00:00:00	12.49	50.22	6.47
2018-03-07 00:00:00	12.49	49.22	6.44
2018-03-08 00:00:00	12.47	49.64	6.44
2018-03-09 00:00:00	12.48	50.83	6.39
2018-03-12 00:00:00	12.5	50.09	6.45
2018-03-13 00:00:00	12.49	50.54	6.37
2018-03-14 00:00:00	12.39	49.84	6.42

处理缺失值的方法：补齐法（向后填充）

```
stock_ffill2 = stock.fillna(method='bfill')
```

stock_ffill2 - DataFrame

日期	浦发银行	上海机场	中国石化
2018-03-01 00:00:00	12.47	49.33	6.39
2018-03-02 00:00:00	12.41	50.35	6.33
2018-03-05 00:00:00	12.42	50.22	6.39
2018-03-06 00:00:00	12.49	49.22	6.47
2018-03-07 00:00:00	12.49	49.22	6.44
2018-03-08 00:00:00	12.47	49.64	6.39
2018-03-09 00:00:00	12.48	50.83	6.39
2018-03-12 00:00:00	12.5	50.09	6.45
2018-03-13 00:00:00	12.49	50.54	6.37
2018-03-14 00:00:00	12.39	49.84	6.42

4.4 数据框之间的操作

4.4.1 生成两个新的数据框

【例4-17】从外部导入沪深300指数在2016年至2017年每日交易价格的数据并且生成一个新的数据框，这个数据框待会将用于按行拼接，具体的代码如下：

```
HS300_excel2 = pd.read_excel('D:\Zhangzw\Python\Python金融数据分析\RawData\第4章\沪深300指数.xlsx',sheet_name="Sheet2",header=0,index_col=0)
#注意导入的是sheet2
```

HS300_excel2 - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2016-01-04 00:00:00	3725.86	3726.24	3468.95	3469.07
2016-01-05 00:00:00	3382.18	3518.22	3377.28	3478.78
2016-01-06 00:00:00	3482.41	3543.74	3468.47	3539.81
2016-01-07 00:00:00	3481.15	3481.15	3284.74	3294.38
2016-01-08 00:00:00	3371.87	3418.85	3237.93	3361.56
2016-01-11 00:00:00	3303.12	3342.48	3192.45	3192.45
2016-01-12 00:00:00	3214.82	3242.25	3174.55	3215.71
2016-01-13 00:00:00	3240.48	3257.3	3155.88	3155.88
2016-01-14 00:00:00	3076.64	3226.66	3072.04	3221.57
2016-01-15 00:00:00	3200.89	3216.8	3101.05	3118.73
2016-01-18 00:00:00	3068.23	3165.62	3066.73	3130.73
2016-01-19 00:00:00	3132.7	3227.91	3119.22	3223.13

4.4.1 生成两个新的数据框

【例4-18】从外部导入2018年3月东方航空、宝钢股份这两只股票的日收盘价格数据且生成一个新的数据框，该数据框将用于按列拼接，具体的代码如下：

```
stock2 = pd.read_excel('D:\Zhangzw\Python\Python金融数据分析\RawData\第4章\关于2018年3月份股票价格数据
.xlsx',sheet_name="Sheet2",header=0,index_col=0)  #注意导入的是sheet2
```

stock2 - DataFrame

日期	东方航空	宝钢股份
2018-03-01 00:00:00	7.606	9.5103
2018-03-02 00:00:00	7.4473	9.1394
2018-03-05 00:00:00	7.487	9.0443
2018-03-06 00:00:00	7.5961	9.168
2018-03-07 00:00:00	7.4771	8.9587
2018-03-08 00:00:00	7.487	8.9207
2018-03-09 00:00:00	7.5068	8.6544
2018-03-12 00:00:00	7.6258	8.759
2018-03-13 00:00:00	7.5366	8.778
2018-03-14 00:00:00	7.4076	9.0634
2018-03-15 00:00:00	7.3283	9.0729

4.4.2 函数 concat的运用

【例4-19】以例4-4和例4-17生成的两个沪深300指数的日交易价格数据框按行拼接，最终生成2016年至2018年期间沪深300指数每日交易价格的数据框，具体的代码如下:

```
HS300_new = pd.concat([HS300_excel2,HS300_excel1],axis=0)
```

我们可以用head和tail函数调看拼接后的数据开头和结尾5行：

```
HS300_new.head()
Out[7]:
```

日期	开盘点位	最高点位	最低点位	收盘点位
2016-01-04	3725.8561	3726.2446	3468.9485	3469.0662
2016-01-05	3382.1769	3518.2170	3377.2799	3478.7797
2016-01-06	3482.4064	3543.7394	3468.4666	3539.8082
2016-01-07	3481.1499	3481.1499	3284.7374	3294.3839
2016-01-08	3371.8710	3418.8508	3237.9307	3361.5632

```
HS300_new.tail()
Out[8]:
```

日期	开盘点位	最高点位	最低点位	收盘点位
2018-12-24	3015.4974	3040.3524	3007.3292	3038.1981
2018-12-25	3006.8787	3030.1418	2964.8750	3017.2815
2018-12-26	3012.8690	3029.0608	2996.4829	3002.0327
2018-12-27	3042.9491	3047.2348	2990.5057	2990.5057
2018-12-28	2994.7950	3024.3525	2984.8177	3010.6536

4.4.2 函数 concat的运用

【例4-20】运用例4-16生成的浦发银行、上海机场、中国石化这3只股票2018年3月的日收盘价形成的数据框 stock，和例4-19生成的东方航空、宝钢股份两只股票同期的日收盘价的数据框 stock2，将这两个数据框按列拼接，具体的代码如下：

```
stock_new = pd.concat([stock,stock2],axis=1)  #按列拼接
```

stock_new - DataFrame

日期	浦发银行	上海机场	中国石化	东方航空	宝钢股份
2018-03-01 00:00:00	12.47	49.33	6.39	7.606	9.5103
2018-03-02 00:00:00	12.41	50.35	6.33	7.4473	9.1394
2018-03-05 00:00:00	12.42	50.22	6.39	7.487	9.0443
2018-03-06 00:00:00	12.49	nan	6.47	7.5961	9.168
2018-03-07 00:00:00	12.49	49.22	6.44	7.4771	8.9587
2018-03-08 00:00:00	12.47	49.64	nan	7.487	8.9207
2018-03-09 00:00:00	12.48	50.83	6.39	7.5068	8.6544
2018-03-12 00:00:00	12.5	50.09	6.45	7.6258	8.759
2018-03-13 00:00:00	12.49	50.54	6.37	7.5366	8.778
2018-03-14 00:00:00	12.39	49.84	6.42	7.4076	9.0634
2018-03-15 00:00:00	12.39	50	6.42	7.3283	9.0729

4.4.3 函数 merge的运用

可以运用函数merge对不同数据框按列进行拼接，注意在使用merge时，需要明确放置在左侧（left）与右侧（right）的数据框。

【例4-21】沿用例4-20中涉及的两个数据框stock、stock2，运用函数 merge按列拼接，具体的代码如下：

```
stock_new2 = pd.merge(left=stock,right=stock2,left_index=True,right_index=True)
```

4.4.4 函数 join的运用

还可以用函数join对不同数据框进行按列拼接，注意用法是“数据框.join（参数设置)”的方式进行运用。

【例4-22】沿用例4-20中涉及的两个数据框 stock、stock2，运用函数join按列拼接,具体的代码如下：

stock_new2 = stock.join(stock2,on=日期)

stock_new3 - DataFrame

日期	浦发银行	上海机场	中国石化	东方航空	宝钢股份
2018-03-01 00:00:00	12.47	49.33	6.39	7.606	9.5103
2018-03-02 00:00:00	12.41	50.35	6.33	7.4473	9.1394
2018-03-05 00:00:00	12.42	50.22	6.39	7.487	9.0443
2018-03-06 00:00:00	12.49	nan	6.47	7.5961	9.168
2018-03-07 00:00:00	12.49	49.22	6.44	7.4771	8.9587
2018-03-08 00:00:00	12.47	49.64	nan	7.487	8.9207
2018-03-09 00:00:00	12.48	50.83	6.39	7.5068	8.6544
2018-03-12 00:00:00	12.5	50.09	6.45	7.6258	8.759
2018-03-13 00:00:00	12.49	50.54	6.37	7.5366	8.778
2018-03-14 00:00:00	12.39	49.84	6.42	7.4076	9.0634
2018-03-15 00:00:00	12.39	50	6.42	7.3283	9.0729
2018-03-16 00:00:00	12.3	49.34	6.42	7.3184	8.9492
2018-03-19 00:00:00	nan	50.14	6.59	7.1597	8.6734

4.5 数组框的主要统计函数

4.5.1 静态的统计函数

表 4-7　常用静态统计函数

函数	函数含义	Python 的演示（以数据框 HS300_new 为对象）
diff	计算一阶差分 注：diff 是差分英文 difference 的缩写	HS300_diff = HS300_new.diff() HS300_diff - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2016-01-04 00:00:00	nan	nan	nan	nan
2016-01-05 00:00:00	-343.679	-208.028	-91.6686	9.7135
2016-01-06 00:00:00	100.229	25.5224	91.1867	61.0285
2016-01-07 00:00:00	-1.2565	-62.5895	-183.729	-245.424
2016-01-08 00:00:00	-109.279	-62.2991	-46.8067	67.1793
2016-01-11 00:00:00	-68.7464	-76.3701	-45.4808	-169.113
2016-01-12 00:00:00	-88.3012	-100.227	-17.9002	23.26
2016-01-13 00:00:00	25.6604	15.0421	-18.671	-59.8312
2016-01-14 00:00:00	-163.839	-30.6377	-83.8398	65.6927

4.5.1 静态的统计函数

ixmax	最大值的行索引值 注：idxmax是英文index maximum的缩写	HS300_new.idxmax() Out[7]: 开盘点位　2018-01-24 最高点位　2018-01-26 最低点位　2018-01-26 收盘点位　2018-01-24 dtype: datetime64[ns]
idxmin	最小值的行索引值 注：idxmin是英文index minimum的缩写	HS300_new.idxmin() Out[9]: 开盘点位　2016-01-29 最高点位　2016-02-29 最低点位　2016-02-29 收盘点位　2016-01-28 dtype: datetime64[ns]
kurt	峰度（四阶矩） 注：kurt是峰度英文kurtosis的缩写	HS300_new.kurt() Out[10]: 开盘点位　-0.664897 最高点位　-0.647326 最低点位　-0.679831 收盘点位　-0.655301 dtype: float64

4.5.1 静态的统计函数

max	最大值	HS300_new.max() Out[11]: 开盘点位　4389.4547 最高点位　4403.3375 最低点位　4351.4871 收盘点位　4389.8853 dtype: float64
mean	平均数	HS300_new.mean() Out[12]: 开盘点位　3499.743364 最高点位　3523.404709 最低点位　3476.860283 收盘点位　3502.227122 dtype: float64
median	中位数	HS300_new.median() Out[13]: 开盘点位　3412.6523 最高点位　3435.6366 最低点位　3398.9532 收盘点位　3421.4419 dtype: float64
min	最小值	HS300_new.min() Out[14]: 开盘点位　2855.5983 最高点位　2939.8745 最低点位　2821.2149 收盘点位　2853.7562 dtype: float64

4.5.1 静态的统计函数

pct_change	百分比变化 注:pct_change 是百分比变化英文 percentage change 的缩写	HS300_perc = HS300_new.pct_change() HS300_perc - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2016-01-04 00:00:00	nan	nan	nan	nan
2016-01-05 00:00:00	-0.0922417	-0.0558277	-0.0264255	0.00280003
2016-01-06 00:00:00	0.0296346	0.00725436	0.027	0.0175431
2016-01-07 00:00:00	-0.000360814	-0.017662	-0.0529713	-0.0693327
2016-01-08 00:00:00	-0.0313916	-0.0178961	-0.0142498	0.0203921
2016-01-11 00:00:00	-0.0203882	-0.0223379	-0.0140463	-0.0503079
2016-01-12 00:00:00	-0.0267326	-0.029986	-0.00560704	0.00728594
2016-01-13 00:00:00	0.0079819	0.0046394	-0.00588146	-0.0186059
2016-01-14 00:00:00	-0.0505601	-0.00940587	-0.0265662	0.020816
2016-01-15 00:00:00	0.0403828	-0.00305576	0.00944431	-0.0319227
2016-01-18 00:00:00	-0.0414426	-0.0159106	-0.0110689	0.00384734

quantile	分位数 其中，函数中需要设定参数 q=分位数，默认为 q=05，也就是 50%的分位数	HS300_new.quantile(q=0.2) #计算 20%分位数 Out[16]: 开盘点位 3204.2183 最高点位 3229.6237 最低点位 3181.8128 收盘点位 3209.2902 Name: 0.2, dtype: float64

4.5.1 静态的统计函数

shift	数据框移动	HS300_shift1 = HS300_new.shift(1) HS300_shift1 - DataFrame

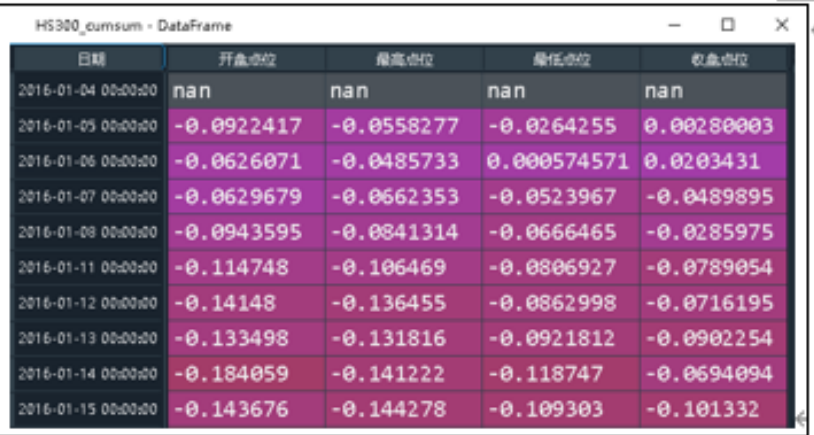
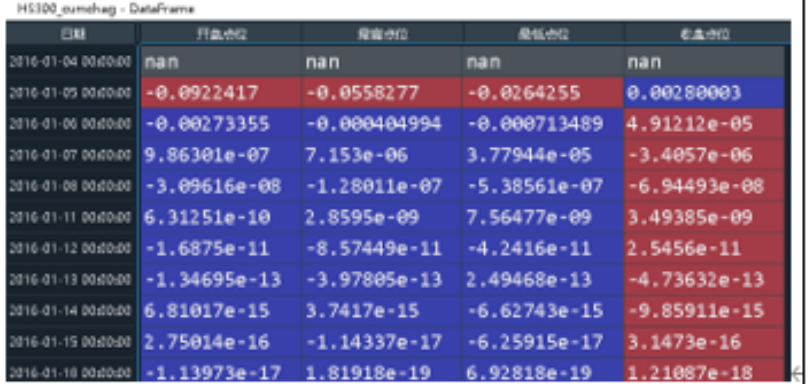
日期	开盘点位	最高点位	最低点位	收盘点位
2016-01-04 00:00:00	nan	nan	nan	nan
2016-01-05 00:00:00	3725.86	3726.24	3468.95	3469.07
2016-01-06 00:00:00	3382.18	3518.22	3377.28	3478.78
2016-01-07 00:00:00	3482.41	3543.74	3468.47	3539.81
2016-01-08 00:00:00	3481.15	3481.15	3284.74	3294.38
2016-01-11 00:00:00	3371.87	3418.85	3237.93	3361.56
2016-01-12 00:00:00	3303.12	3342.48	3192.45	3192.45
2016-01-13 00:00:00	3214.82	3242.25	3174.55	3215.71
2016-01-14 00:00:00	3240.48	3257.3	3155.88	3155.88

std	样本标准差	HS300_new.std() Out[19]: 开盘点位 336.780663 最高点位 336.231104 最低点位 337.467202 收盘点位 337.272162 dtype: float64

4.5.1 静态的统计函数

std	样本标准差	HS300_new.std() Out[19]: 开盘点位　336.780663 最高点位　336.231104 最低点位　337.467202 收盘点位　337.272162 dtype: float64
sum	求和	HS300_perc.sum() Out[20]: 开盘点位　-0.163520 最高点位　-0.174463 最低点位　-0.106405 收盘点位　-0.092767 dtype: float64
var	样本方差	HS300_new.var() Out[21]: 开盘点位　113421.215128 最高点位　113051.355192 最低点位　113884.112312 收盘点位　113752.511207 dtype: float64

4.5.1 静态的统计函数

cumsum	累积求和，也就是依次给出前 1、2、…、n 个数的和。	HS300_cumsum = HS300_perc.cumsum()
	注：cumsum 是累积求和英文 cumulative sum 的缩写	
cumprod	累积求积，依次给出前 1、2、…、n 个数的积。 注：cumprod 是累积求积英文 cumulative product 的缩写	HS300_cumchag = HS300_perc.cumprod() 

HS300_cumsum - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2016-01-04 00:00:00	nan	nan	nan	nan
2016-01-05 00:00:00	-0.0922417	-0.0558277	-0.0264255	0.00280003
2016-01-06 00:00:00	-0.0626071	-0.0485733	0.000574571	0.0203431
2016-01-07 00:00:00	-0.0629679	-0.0662353	-0.0523967	-0.0489895
2016-01-08 00:00:00	-0.0943595	-0.0841314	-0.0666465	-0.0285975
2016-01-11 00:00:00	-0.114748	-0.106469	-0.0806927	-0.0789054
2016-01-12 00:00:00	-0.14148	-0.136455	-0.0862998	-0.0716195
2016-01-13 00:00:00	-0.133498	-0.131816	-0.0921812	-0.0902254
2016-01-14 00:00:00	-0.184059	-0.141222	-0.118747	-0.0694094
2016-01-15 00:00:00	-0.143676	-0.144278	-0.109303	-0.101332

HS300_cumchag - DataFrame

日期	开盘点位	最高点位	最低点位	收盘点位
2016-01-04 00:00:00	nan	nan	nan	nan
2016-01-05 00:00:00	-0.0922417	-0.0558277	-0.0264255	0.00280003
2016-01-06 00:00:00	-0.00273355	-0.000404994	-0.000713489	4.91212e-05
2016-01-07 00:00:00	9.86301e-07	7.153e-06	3.77944e-05	-3.4057e-06
2016-01-08 00:00:00	-3.09616e-08	-1.28011e-07	-5.38561e-07	-6.94493e-08
2016-01-11 00:00:00	6.31251e-10	2.8595e-09	7.56477e-09	3.49385e-09
2016-01-12 00:00:00	-1.6875e-11	-8.57449e-11	-4.2416e-11	2.5456e-11
2016-01-13 00:00:00	-1.34695e-13	-3.97805e-13	2.49468e-13	-4.73632e-13
2016-01-14 00:00:00	6.81017e-15	3.7417e-15	-6.62743e-15	-9.85911e-15
2016-01-15 00:00:00	2.75014e-16	-1.14337e-17	-6.25915e-17	3.1473e-16
2016-01-18 00:00:00	-1.13973e-17	1.81918e-19	6.92818e-19	1.21087e-18

4.5.1 静态的统计函数

corr	相关系数 注：corr是相关系数英文correlation coefficient ）的缩写	HS300_new.corr() Out[25]: 开盘点位 最高点位 最低点位 收盘点位 开盘点位 1.000000 0.997848 0.996750 0.994180 最高点位 0.997848 1.000000 0.996429 0.997193 最低点位 0.996750 0.996429 1.000000 0.997661 收盘点位 0.994180 0.997193 0.997661 1.000000

4.5.2移动窗口与动态统计函数

时点的数据往往波动较大，因此某一时点的数据通常不能很好地表现数据本身的特性，于是就需要采用一段时间区间的数据进行描述。因此，引出了移动窗口（也称为“滑动窗口”）的概念。

简而言之，为了提升数据的可靠性，将某个点的取值扩大到包含这个点的一段区间，并且用区间进行判断，这个区间就是窗口。在Pandas中，就有一个移动窗口函数rolling，可以方便地计算带有移动窗口的数据框统计量，具体的函数形式以及主要的参数如下：

数据框或序列.rolling(window=窗口数,axis=0或1）.统计量函数（axis=0或1）

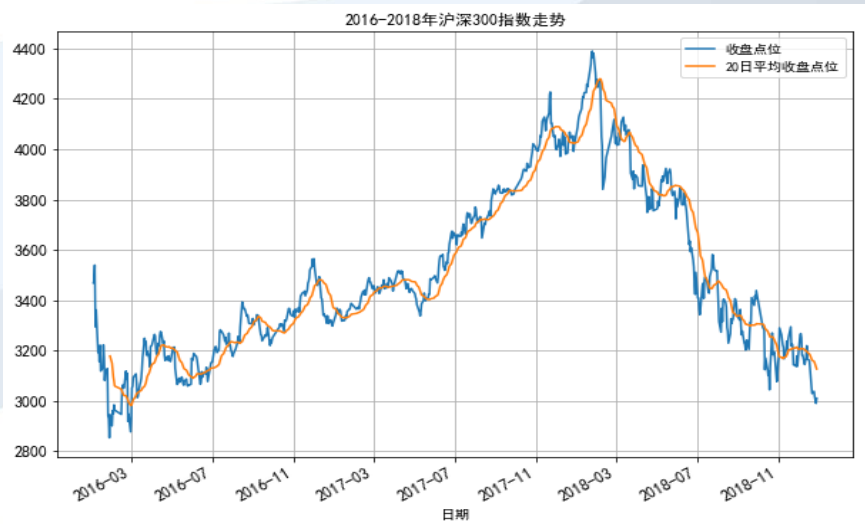
注意，函数 rolling中的参数axis=0或1的含义依然是选择0表示按列实现函数（默认值），1则是按行来实现函数。

下面，主要介绍金融时间序列中最重要的3个移动统计量，分别是移动平均、移动标准差（移动波动率）、移动相关系数。

4.5.2移动窗口与动态统计函数

【例4-23】以例4-19生成的数据框作为分析对象，生成收盘点位的20日均值，将该平均值与每日的收盘点位进行可视化（见图4-4），具体的代码如下：

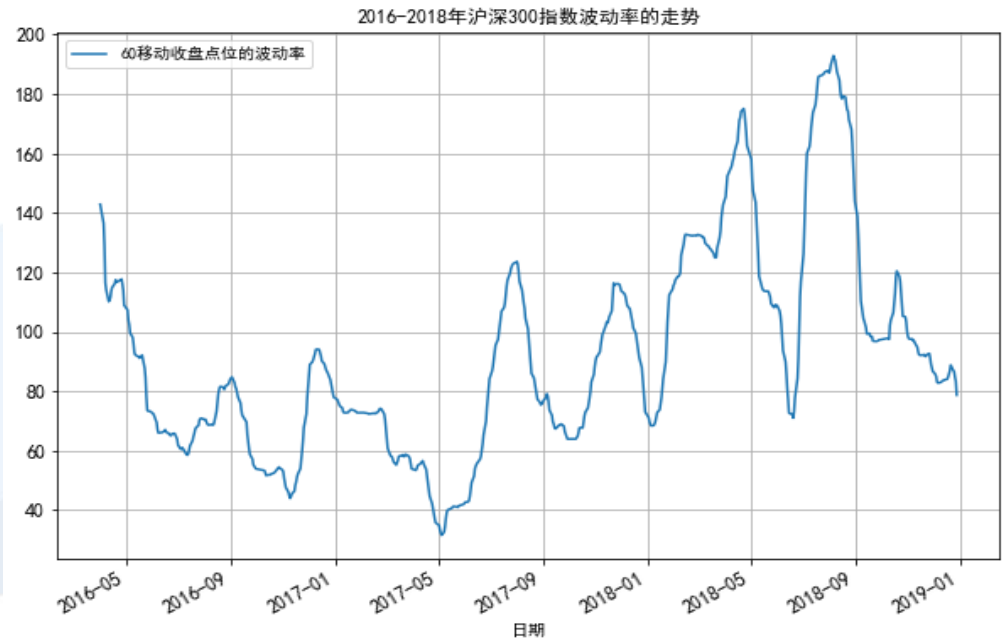
```
HS300_meanclose = HS300_new['收盘点位'].rolling(window=20).mean() #生成一个20日平均收盘点位的序列
HS300_meanclose = HS300_meanclose.to_frame()  #将序列变成数据框
HS300_meanclose = HS300_meanclose.rename(columns={'收盘点位':'20日平均收盘点位'})
HS300_close = HS300_new['收盘点位'].to_frame()#生成每日收盘点位的序列
HS300_new1 = pd.concat([HS300_close,HS300_meanclose],axis=1) #生成包括每日收盘点位，20日平均收盘点位的新数据框
HS300_new1.plot(figsize=(10,7),title='2016-2018年沪深300指数走势',grid=True,fontsize=12)
```



4.5.2移动窗口与动态统计函数

【例4-24】以例4-19生成的数据框作为分析对象，生成60天时间窗口的沪深300指数收盘点位的移动波动率（移动标准差），并且进行可视化（见图4-5），具体的代码如下：

```
HS300_rollingstd  = HS300_new[收盘点位].rolling(window=60).std()
HS300_rollingstd  = HS300_rollingstd.to_frame()
HS300_rollingstd  = HS300_rollingstd.rename(收盘点位:60移动收盘点位的波动率)
HS300_rollingstd.plot(figsize=(10,7),title=u'2016-2018年沪深300指数波动率的走势',grid=True,fontsize=12)
```



4.5.2移动窗口与动态统计函数

【例4-25】以例4-19生成的数据框作为分析对象，生成30天时间窗口的沪深300指数开盘点位、最高点位、最低点位以及收盘点位之间的移动相关系数，具体的代码如下：

```
HS300_rollingcorr  = HS300_new.rolling(window=30).corr()
HS300_rollingcorr  = HS300_rollingcorr.dropna()
```

HS300_rollingcorr - DataFrame

	日期	Index	开盘点位	最高点位	最低点位	收盘点位
0	2016-02-19 00:00:00	开盘点位	1	0.980264	0.955704	0.903957
1	2016-02-19 00:00:00	最高点位	0.980264	1	0.971737	0.953592
2	2016-02-19 00:00:00	最低点位	0.955704	0.971737	1	0.972716
3	2016-02-19 00:00:00	收盘点位	0.903957	0.953592	0.972716	1
4	2016-02-22 00:00:00	开盘点位	1	0.973813	0.952086	0.900403
5	2016-02-22 00:00:00	最高点位	0.973813	1	0.967829	0.955815
6	2016-02-22 00:00:00	最低点位	0.952086	0.967829	1	0.969874
7	2016-02-22 00:00:00	收盘点位	0.900403	0.955815	0.969874	1
8	2016-02-23 00:00:00	开盘点位	1	0.974454	0.949938	0.894477
9	2016-02-23 00:00:00	最高点位	0.974454	1	0.961305	0.947445

Your appreciation makes me a miracle. Thank you!

Frank Ziwei Zhang
18117228563
frank8027@163.com

上海对外经贸大学
SHANGHAI UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS