【Python】No.3 Pandas的入门级教程

NAU Analysts 2020-02-21 09:00

以下文章来源于未生的半泽直树 ，作者日向家的栗子

未生的半泽直树

Pandas简介

在金融分析与风险管理中，很重要的一项工作是对金融时间序列（以时间作为索引的数据集合）的分析，但是NumPy的数组结构无法满足时间序列的要求。因此，在利用NumPy科学计算的基础上，作为一种包含金融时间序列的金融数据分析工具，Pandas就应运而生了。Pandas的取名是面板数（panel data)与数据分析（data analysis)的结合。Pandas纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。本篇文章就是结合具体案例讨论并演示Pandas模块的基本操作。

学习方法

在正式开始Pandas案例演示操作之前，想先介绍一下Python大神推荐的Pandas的学习方法。只有掌握正确的学习方法，才能深刻地理解所学知识，并灵活地运用，举一反三。

实现功能 培养感觉

> 在实践Pandas的过程中，不断地发现其能够替代Excel的强大之处。作为Python小白，在学习Pandas过程中，要注意培养对Pandas的感觉和兴趣，才能在试错的道路上越挫越勇！

遇到困境 前车可鉴

> 在执行代码报错时，先检查代码，自己发现错误，一般报错红色框的最后一行是核心报错信息，明确指出错误在哪里。如果自己解决不了，可以利用博客园、CSDN、 StackOverflow等网站答疑解惑,这些网站上都有前辈的经验教训。

不知甚多 翻API

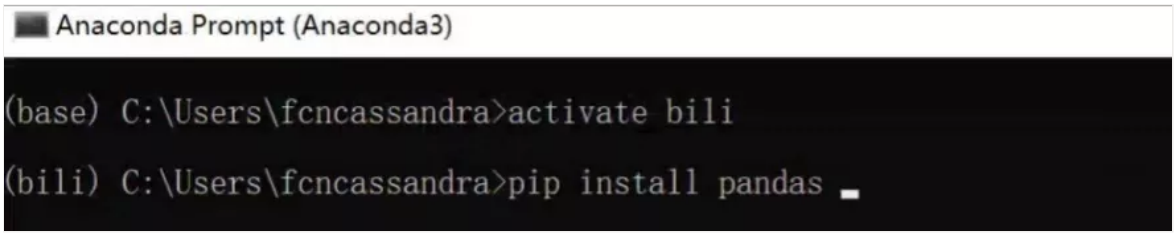
> 在深度学习Pandas过程中，会发现自己知识盲区实在太多，就可以利用Pandas中文网系统性地学习Pandas(https://www.pypandas.cn/ ）。在不断拓展知识过程中，可以将自己整理的知识漏洞和报错解决方案复制到有道云笔记和印象笔记上，方便日后的复习、巩固。

神功已成 传递他人

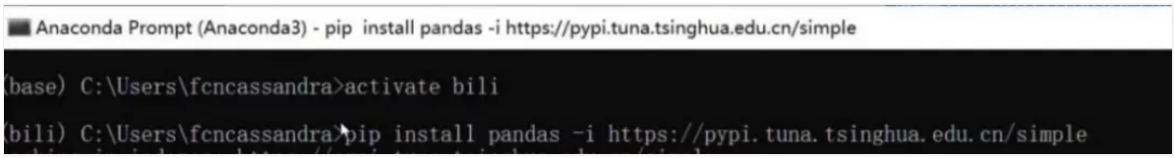
> 当小白已修炼成神功，不妨做个up主或者小老师，通过制作视频和写博文等方式，将此神功传授给后辈。在传授过程中，也会弥补自己知识漏洞，提升神功的段位。这种学习方法也叫做费曼学习法。

Pandas的安装与导入

由于Pandas是Python的外部第三方模块，在使用前需要在自己创建的环境中安装并导入Pandas，才能启用Pandas。首先打开Anaconda Prompt,在激活自己的环境后，利用pip install 命令安装Pandas。然后再打开base环境Jupyter lab,导入Pandas模块。



如果安装时网速较慢，也可以从国内的镜像源网站安装Pandas。（如下图，就是从清华大学镜像源网站https://pypi.tuna.tsinghua.edu.cn/simple 安装Pandas)



在利用import函数导入Pandas模块时，我们是以缩写的pd对模块进行命名，之后也可以用pd对模块进行调用。

```
import pandas as pd
```

正如Numpy的核心是nyarray,Pandas也是围绕着一维的序列（Series）和二维的数据框（DataFrame）两个核心数据结构展开的。

1. 创建对象

1.1 序列

Pandas中的序列（Series）是一个类似于一维数组的数据结构，它是由两部分组成，第一部分是标签（label)或者索引（index），第二部分是对应的数值（values）。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。但是，序列有一个致命的缺点，只能有两列，即索引列和数值列。

以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为例。（如图所示）

	中国石油	工商银行	上汽集团	宝钢股份
2018-09-03	0.003731	-0.001838	-0.003087	-0.024112
2018-09-04	0.021066	0.001842	-0.000344	0.011704
2018-09-05	-0.004854	-0.016544	-0.033391	-0.029563
2018-09-06	0.006098	-0.003738	0.007123	-0.014570
2018-09-07	-0.006060	0.003752	0.004597	0.016129

首先演示手动输入生成序列。生成序列需要运用Series函数，并且要输入涉及的索引index。针对表中2018年9月3日数据生成序列，代码如下：

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

执行结果：

```
return_series1

中国石油     0.003731
工商银行    -0.001838
上汽集团    -0.003087
宝钢股份    -0.024112
dtype: float64
```

1.2 数据框

Pandas的数据框（DataFrame)类似于Excel表结构的数据结构，由三部分组成：第一部分是行索引（index），第二部分是列名称（columns), 第三部分是取值。数据框也可以看成是几个序列的水平拼接的结果。

创建数据框可以运用DataFrame函数，该函数的参数主要有三个，第一个参数是data（输入相关的数据或变量），第二个参数是index（输入行索引），第三个参数是columns（输入列名）。

沿用案例，将日期作为行索引，股票作为列名，DataFrame函数手动创建数据框。

```
date = ['2018-09-03','2019-09-04','2019-09-05','2019-09-06','2019-09-07']
stock = ['中国石油','工商银行','上汽集团','宝钢股份']
return_array = (['0.003731','-0.001838','-0.003087','-0.024112'],
        ['0.021066','0.001842','-0.000344','0.011704'],
        ['-0.004854','-0.016544','-0.033391','-0.029563'],
        ['0.006098','-0.003738','0.007123','-0.01457'],
        ['0.00606','0.003752','0.004597','0.016129'])
return_dataframe=pd.DataFrame(data=return_array ,index=date,columns=stock)
```

执行结果：

```
return_dataframe
```

	中国石油	工商银行	上汽集团	宝钢股份
2018-09-03	0.003731	-0.001838	-0.003087	-0.024112
2019-09-04	0.021066	0.001842	-0.000344	0.011704
2019-09-05	-0.004854	-0.016544	-0.033391	-0.029563
2019-09-06	0.006098	-0.003738	0.007123	-0.01457
2019-09-07	0.00606	0.003752	0.004597	0.016129

除此之外，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to_excel，to_csv函数,并且需要在函数中添加导出文件存放的路径（如'C:/Users/Desktop')和带格式的文件名(如：return.xlxs)。

针对数据框return_dataframe依次以Excel、CSV、txt格式导出并存放在计算机桌面。具体代码如下：

```
return_dataframe.to_csv('C:/Users/Desktop/return.csv') #以csv格式导出
return_dataframe.to_excel('C:/Users/Desktop/return.xlsx') #以excel格式导出
return_dataframe.to_csv('C:/Users/Desktop/return.txt') #以txt格式导出
```

1.3 从外部导入并直接生成数据框

当涉及到较多数据时，一般很少直接手动输入，而是从外部导入数据。外部数据导入，主要有两种方式，一种是通过外部文件（如Excel、CSV、txt等格式）导入，一种是借助外部金融数据库与Python的API（应用程序接口，实现不同计算机软件之间的相互通信）导入数据。

外部文件导入

txt,excel,csv等格式的文件均可以导入到Python中，具体的函数及主要参数如下。

函数名和主要参数	具体说明
read_table('文件路径', 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 表中导入的工作表名称，比如 sheetname='Sheet1'，表示导入工作表 Sheet1； header：指定列名行，默认是 header=0 就是取工作表第一行作为指定列名行； index_col：指定某一列作为索引列，比如，index_col=0 就表示将第一列作为索引列
read_table('文件路径',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，如果不指定参数，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 参数失效； header：用法与导人 Excel 表是一致的； index_col：用法与导入 Excel 表是一致的

以从外部Excel文件中导入数据并创建数据框stock为例，相关代码如下：

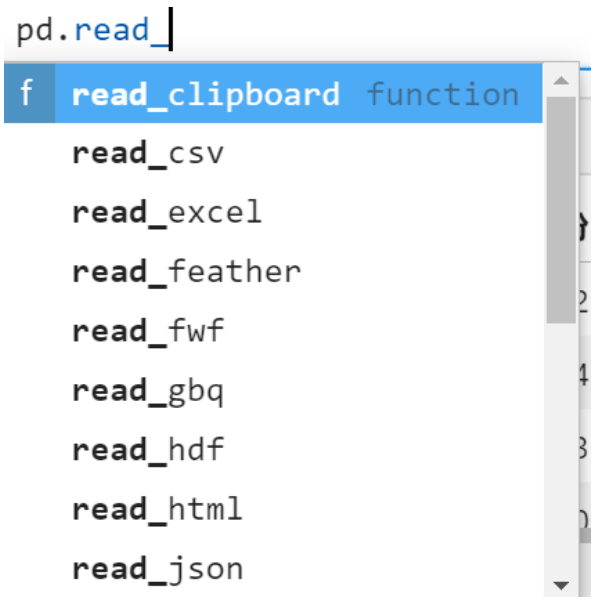
```
stock = pd.read_excel("C:/Users/Documents/工作簿1.xlsx",sheet_name='Sheet1',header=0,index_col=0)
```

执行结果：

stock

	中国石油	工商银行	上汽集团	宝钢股份
2018-09-03	0.003731	-0.001838	-0.003087	-0.024112
2018-09-04	0.021066	0.001842	-0.000344	0.011704
2018-09-05	-0.004854	-0.016544	-0.033391	-0.029563
2018-09-06	0.006098	-0.003738	0.007123	-0.014570
2018-09-07	-0.006060	0.003752	0.004597	0.016129

作为一个小白，想要了解还可以读取哪些类型的文件，就可以在输入read_时，敲击Tab键，会出来许多可读类型的文件。



除了输入文件路径导入外部文件，也可以通过读取剪切板的方式导入外部文件。首先打开文件，然后复制文件中内容，此时数据已经复制到Windows唯一的剪切板中，然后运用read_clipboard函数导入剪切板中的数据，可以获得和数据框stock一样的数据。

	中国石油	工商银行	上汽集团	宝钢股份
2018/9/3	0.003731	-0.00184	-0.00309	-0.02411
2018/9/4	0.021066	0.001842	-0.00034	0.011704
2018/9/5	-0.004854	-0.01654	-0.03339	-0.02956
2018/9/6	0.006098	-0.00374	0.007123	-0.01457
2018/9/7	-0.00606	0.003752	0.004597	0.016129

```
stock=pd.read_clipboard()
```

可以设置参数header的值，来确定columns为哪些值。header=None时，columns没有值。

外部数据库导入

No.1 wind量化接口

以wind为例演示如何通过API在Python中导入数据。

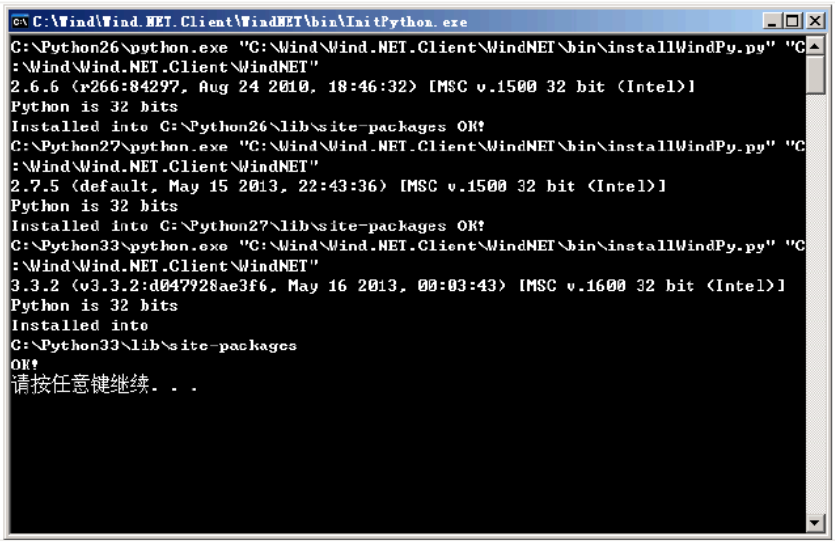
首先，需要在关闭Python环境后，打开Wind资讯终端。

然后点击'量化'选项，出现下方界面，点击'Python插件'。

在'量化'菜单下选择"修复Python插件"，或者输入'Repair Python'命令。会出现下面的界面。

注册成功对话框

Python插件安装对话框



如果按此方法安装不成功或者想要了解更多，可以参考Wind量化平台—用户手册。

本文以获取一系列资产在2019年8月1号至2019年8月8号的收盘价格为例，进行演示。

```
from WindPy import *
w.start() #启动Wind API,close是关闭API
assetlist = ['000300.SH','000905.SH']
startdate = '2019-08-01'
enddate = '2019-08-08'
data1 = w.wsd(assetlist,startdate,enddate,'')
#data1类型为instance
#wsd为日期序列的wind导入数据函数
#'close'为wind导入的指标，''中可以设置其他参数，如单位，币种等
df =pd.DataFrame(data1.Data).T
df.index = data1.Times
df.columns = assetlist
```

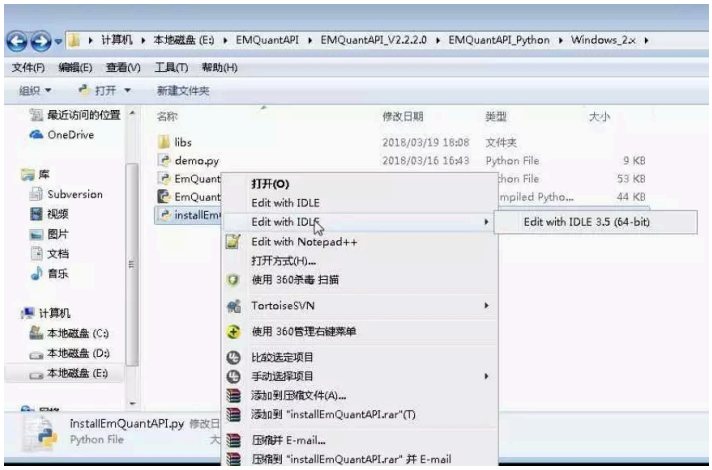
这里用了.T来转置，之所以data1需要转置，是因为wsd（）函数生成的是一个行名称是日期，列名称是股票代码的时间序列表格，但是data1.Data的表达方式：列是日期，股票代码是行，因此需要转置。把转置之后的data1赋值给数据框df,df再加上index和column。

No.2 choice量化接口

首先检查Windows系统是否安装Microsoft Visual C++ 2010可再发行组件包（可在电脑"程序和功能"中查看是否安装，若没安装:下载地址：https://www.microsoft.com/zh-CN/download/details.aspx?id=14632）

然后在choice数据量化平台官网（http://quantapi.eastmoney.com/）下载并解压压缩包EmQuantAPI_Python.zip。在python3文件夹中，找到用于注册python接口的installEMQuant API.py文件。

在命令行运行installEmQuantAPI.py注册，或者使用Anaconda编译器，请在编译器中运行installEmQuantAPI.py注册。



注册成功对话框

绑定手机号：登录Choice量化接口网站主页（http://quantapi.eastmoney.com ），点击右上角账户名-个人资料绑定手机号；或者登录Choice金融终端，进入用户中心-资料管理绑定手机号。

最后登录激活。根据所用系统环境，运行接口激活工具LoginActivator.exe（Windows），输入绑定手机号获取验证码，激活成功后生成令牌文件userInfo，使用时无需输入用户名和密码，默认从令牌中获取登录权限。一个账号多支持在十台设备上激活。如果安装遇到问题以及想要了解更多，可以查阅choice量化平台用户使用手册。

本文以获取一系列资产在2019年8月1号至2019年8月8号的收盘价格为例，进行演示。

```
from EmQuantAPI import *
c.start()
#start()为初始化登录函数，stop（）为退出登录函数
data=c.csd("300059.SZ,600425.SH","open,close","2019-08-01","2019-08-08", "")
#序列函数获取股票，指数，基金，期货等各个证券品种或组合的日频历史行情序列数据
csd(证券代码,指标,startdate,enddate,options:可选参数，为字符串='')
if (data.ErrorCode != 0):
#ErrorCode:错误代码 errorMsg:错误信息
  print ("request csd Error, ", data.ErrorMsg)
else:
  for code in data.Codes:
    for i in range(0, len(data.Indicators)):
      for j in range(0, len(data.Dates)):
        print (data.Data[code][i][j])
```

No.3 Tushare导入数据

这里我们先简单地介绍一下Tushare,它是一个免费、开源的python财经数据接口包。主要实现对股票等金融数据从数据采集、清洗加工到数据存储的过程，能够为金融分析人员提供快速、整洁和多样的便于分析的数据，为他们在数据获取方面极大地减轻工作量，使他们更加专注于策略和模研究与实现上。考虑到Python pandas包在金融量化分析中体现出的优势，Tushare返回的绝大部分的数据格式都是pandas/DataaFrame类型，非常便于用pandas/NumPy/Matplotlib进行数据分析和可视化。

Tushare的普通版网址是http://tushare.org/，目前已经不再维护，但是大部分的API可以调用，新版网址是https://tushare.pro/，下载不同的数据会有一定的积分限制，但是刚注册的用户还是可以下载基本数据。这里还是推荐大家使用新版，一方面新版数据的质量和可选择性更好，另一方面，赞助少量费用或在Tushare论坛多交流的积分提升方式，能够让整个社群更加活跃。

在导入Tushare数据之前，需要在自己创建的anaconda环境中安装Tushare的相关依赖。

```
(bili) C:\Users\fcncassandra>pip install tushare -i https://pypi.tuna.tsinghua.edu.cn/simple
(bili) C:\Users\fcncassandra>pip install lxml -i https://pypi.tuna.tsinghua.edu.cn/simple
(bili) C:\Users\fcncassandra>pip install requests -i https://pypi.tuna.tsinghua.edu.cn/simple
(bili) C:\Users\fcncassandra>pip install bs4 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

以从Tushare获取沪深300指数为例。

```
import tushare as ts
data1 = ts.get_hs300s()
#通过get_hs300s函数获取沪深300指数
```

在接下来的系列文章中，我们将会更加详细地介绍Tushare的功能和获取数据的方法。

除此之外，当前市场上还有优矿、聚宽等功能强大的量化平台，这里就不再展开介绍。

2.数据框内部操作

2.1 描述数据框的基本性质

当需要查看数据框的行索引名和列名时，就要运用index函数查看行索引名，columns函数查看列名，values函数查看值。以数据框stock为例（注意，in后面为代码，out后面为代码执行结果）：

```
in:stock.index
out:DatetimeIndex(['2018-09-03', '2018-09-04', '2018-09-05', '2018-09-06',
        '2018-09-07'],
       dtype='Datetime64[ns]', freq=None)

in: stock.columns
out:Index(['中国石油', '工商银行', '上汽集团', '宝钢股份'], dtype='object')

in: stock.values
out:array([[ 0.003731, -0.001838, -0.003087, -0.024112],
    [ 0.021066,  0.001842, -0.000344,  0.011704],
    [-0.004854, -0.016544, -0.033391, -0.029563],
    [ 0.006098, -0.003738,  0.007123, -0.01457 ],
    [-0.00606 ,  0.003752,  0.004597,  0.016129]])
```

注：数据框的values事实上就是Numpy中的array。

当需要修改列名时，可以运用rename函数，并输入参数columns={‘原列名’：‘新列名’}。以数据框stock为例：

```
in:stock.rename(columns={'宝钢股份':'鞍钢股份'})
out:中国石油   工商银行   上汽集团   鞍钢股份
2018-09-03  0.003731  -0.001838  -0.003087  -0.024112
2018-09-04  0.021066  0.001842   -0.000344  0.011704
2018-09-05  -0.004854  -0.016544  -0.033391  -0.029563
2018-09-06  0.006098   -0.003738  0.007123   -0.014570
2018-09-07  -0.006060  0.003752   0.004597   0.016129
```

想要知道一个数据框由多少行、多少列构成，可以运用shape函数进行查看。以stock为例，相关代码如下：

```
in:stock.shape
out:(5,4)
```

上图的输出结果是一个元组，元组中的第1个元素代表数据框行数（不包括列名那一行，共有5行），第2个元素代表数据框的列数（不包括行索引那一列，共有4列）。

Pandas中的info函数可以给出样本数据的相关信息概览：行数，列数，列索引，列非空值个数，列类型，内存占用。

```
in:stock.info()
out:<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 5 entries, 2018-09-03 to 2018-09-07
Data columns (total 4 columns):
中国石油   5 non-null float64
工商银行   5 non-null float64
上汽集团   5 non-null float64
宝钢股份   5 non-null float64
dtypes: float64(4)
memory usage: 200.0 bytes
```

当想要知道样本量、均值、方差、最大值、最小值、分位数等涉及序列的统计指标时，运用describe函数可以很方便地获取这些综合统计信息。

```
in:stock.describe()
out:
       中国石油     工商银行     上汽集团     宝钢股份
count 5.000000  5.000000  5.000000  5.000000
mean  0.003996  -0.003305 -0.005020 -0.008082
std   0.010901  0.007967   0.016359  0.020846
min   -0.006060 -0.016544 -0.033391  -0.029563
25%   -0.004854 -0.003738 -0.003087  -0.024112
50%   0.003731   -0.001838 -0.000344 -0.014570
75%   0.006098  0.001842  0.004597   0.011704
max   0.021066  0.003752  0.007123   0.016129
```

如上图所示，其中统计量count代表有多少个样本值，，mean代表均值，mim代表最小值，std代表标准差，25%、50%、75%代表25%、50%、75%的分位数，max代表最大值。

其实，在Pandas中也可以用一些静态统计函数来分别求出这些常见的统计值。下图就给出了一些常见的统计函数。

函数	函数含义	Python 的演示（以数据框 stock 为例）
max	最大值	stock.max()
mean	平均值	stock.mean()
min	最小值	stock.min()
median	中位数	stock.median()
quantile	分位数，其中，函数中需要设定参数 q=分位数，默认 q=0.5,也就是 50%的分位数。	stock.quantile(q=0.2)
std	标准差	stock.std()
sum	求和	stock.sum()
var	样本方差	stock.var()
cumsum	累积求和：依次给出前 1、2、…、n 个数的和	stock.cumsum()
cumprod	累积求积：依次给出前 1、2、…、n 个数的积	stock.cumprod()
corr	相关系数	stock.corr()

2.2 groupby聚合函数

groupby函数通常是根据数据框中的某一列或者多列内容进行先分组再聚合的操作。以某一列为例，先将这一列的内容按照不同维度进行拆解，再将同一维度的数据进行聚合。注意默认情况下，数值列会被聚合，而非数值列的结果会被移除。

以下表为例，进行演示。

	data1	data2	key1	key2
0	-0.278565	1.267586	a	one
1	-1.183920	-0.898350	a	two
2	0.011435	-0.207110	b	one
3	1.570595	-1.706337	b	two
4	1.149452	-1.098062	a	one

按key1进行聚合，可以分为a和b两个维度，再分别将同一维度的数据聚合。如下图所示（在这里用for循环将数据分组为a和b后再聚合的结果输出）。

```
in:for i in df.groupby('key1'):
  print(i)
out:
('a',    data1   data2 key1 key2
0 -0.278565 1.267586  a one
1 -1.183920 -0.898350  a two
4 1.149452 -1.098062  a one)
('b',    data1   data2 key1 key2
2 0.011435 -0.207110  b one
3 1.570595 -1.706337  b two)
```

一般地，为了更好地分析数据，会在groupby后会接apply、transform和其他统计函数等等。

如下图所示，就是对key1和key2聚合，并求样本量和均值。对这两列聚合，就是将数据分成了（a,one)、(a,two)、（b,one)和(b,two)这四个维度，再将同一维度的数据聚合。

```
in:df.groupby(['key1','key2']).count()
out:
           data1  data2
key1 key2
 a     one     2      2
       two     1      1
 b     one     1      1
       two     1      1

in:df.groupby(['key1','key2']).mean()
out:
              data1     data2
key1  key2
  a     one    0.435444   0.084762
        two   -1.183920  -0.898350
  b     one   0.011435   -0.207110
        two   1.570595   -1.706337
```

2.3 数据框的索引和截取

索引

对于数据框而言，分别有2种索引的方法。

第1种是运用loc函数，通过输入行索引的方式来输出对应的数据。以下面的数据框stock为例，对数据框（DataFrame）进行loc索引且只输入一列或一行，就是取出其某一序列（Seires）。

```
in:stock.loc['2018-09-03']
out:
中国石油  0.003731
工商银行  -0.001838
上汽集团  -0.003087
宝钢股份  -0.024112
Name: 2018-09-03 00:00:00, dtype: float64
```

第2种是运用iloc函数，通过输入行号（即具体第几行）的方式来输出对应的数据，注意行号0代表第一行，往下依次类推。以数据框stock为例：

```
in:stock.iloc[2] #获取2018年9月5日数据，即第三行数据
out:
中国石油  0.003731
工商银行  -0.001838
上汽集团  -0.003087
宝钢股份  -0.024112
Name: 2018- 09- 03 00:00:00, dtype: float64
```

一般性截取

数据框的截取类似于Numpy的数组切片操作，可以截取特定行或者列的数据。
截取特定行的话，可以类似切片操作。以数据框stock为例，取其前两行数据：

```
in:stock[:2]
out:
              中国石油   工商银行    上汽集团    宝钢股份
2018-09-03  0.003731  -0.001838  -0.003087  -0.024112
2018-09-04  0.021066  0.001842  -0.000344  0.011704
```

如果想要同时截取特定行和列的数据时，就需要用到iloc函数，第1个数字段表示特定的行，第2个数字段表示特定的列。注意0代表第1行或者第1列，依次往后类推。

```
in:stock.iloc[:3,1:3]#截取第1- 3行的第2，3列
out:
              工商银行   上汽集团
2018-09-03  -0.001838  -0.003087
2018-09-04  0.001842  -0.000344
2018-09-05  -0.016544  -0.033391
```

条件性截取

为了找到数据框中符合特定条件的数据，就需要运用条件性截取，比如针对某个列名找到大于某个值的数据。条件性截取可以设置多个选取条件，但是多个条件需要用英文的小括号括起来，并且不同条件之间需要用&符号连接。

以stock为例，如下图所示，获取上汽集团涨跌幅>0且<0.005的数据。

```
in:stock[(stock['上汽集团']>0)&(stock['上汽集团']<0.005)]
out:          中国石油   工商银行   上汽集团   宝钢股份
2018-09-07  -0.00606  0.003752  0.004597  0.016129
```

注意无论是索引还是截取都不是改变数据框原来的值，只是提取其中的部分数据而已。

2.4 数据框的排序

行排序

当想要对数据框的行索引大小进行排序，比如针对金融时间序列按照时间由近至远排序，就需要用到sort_index函数，且默认从小到大排序。如果想要从大到小，就需要用到参数ascending，ascending=True默认为从小到大，ascending=False则为从大到小排序。

以数据框stock为例，对日期进行从远到近排序。参数axis=0表示按行排序。

```
in:stock.sort_index(axis=0,ascending=False) #按行从大到小
out:          中国石油   工商银行   上汽集团   宝钢股份
2018-09-07  -0.006060  0.003752  0.004597  0.016129
2018-09-06  0.006098  -0.003738  0.007123  -0.014570
2018-09-05  -0.004854  -0.016544  -0.033391  -0.029563
2018-09-04  0.021066  0.001842  -0.000344  0.011704
2018-09-03  0.003731  -0.001838  -0.003087  -0.024112
```

列排序

当想要按照数据框中某个列名对应的数值大小进行排序，需要用到sort_values函数，参数by='列名'以及参数ascending=True（默认情况，从小到大）或者ascending=False(从大到小）。

如下图所示，就是对列名宝钢股份对应的值进行从大到小排序。

```
in:stock.sort_values(by='宝钢股份',ascending=False) #按列从大到小
out:          中国石油   工商银行   上汽集团   宝钢股份
2018-09-07  -0.006060  0.003752  0.004597  0.016129
2018-09-04  0.021066  0.001842  -0.000344  0.011704
2018-09-06  0.006098  -0.003738  0.007123  -0.014570
2018-09-03  0.003731  -0.001838  -0.003087  -0.024112
2018-09-05  -0.004854  -0.016544  -0.033391  -0.029563
```

2.5 缺失值的处理

由于某种特殊原因，如召开股东会，重大资产重组等，上市公司的股票数据可能存在缺失的情况。我们对数据进行处理和分析时，缺失值会是一个棘手的问题。

对缺失值，一般有两种方法：删除法和补齐法。

删除法就是运用dropna函数直接将存在缺失值的这一行数据删除。以右图的stock1为例，左图为代码和执行结果。

删除法

```
stock1_dropna=stock1.dropna()
```

```
stock1_dropna
```

	上海机场	浦发银行
2018/3/1	12.47	49.33
2018/3/2	12.41	50.35
2018/3/5	12.42	50.22
2018/3/6	12.49	NaN
2018/3/7	12.49	49.22

	上海机场	浦发银行
2018/3/1	12.47	49.33
2018/3/2	12.41	50.35
2018/3/5	12.42	50.22
2018/3/7	12.49	49.22

补齐法

补齐法，就是运用fillna函数将缺失值进行补充。对缺失值的补充，可以分为赋值零、向前填充和向前填充。

No.1 赋值零

运用fillna函数，并输入参数value=0的将缺失值赋值为零。以stock1为例。

```
in:stock1_fillzero=stock1.fillna(value=0)
  stock1_fillzero
out:     上海机场 浦发银行
2018/3/1  12.47  49.33
2018/3/2  12.41  50.35
2018/3/5  12.42  50.22
2018/3/6  12.49   0.00
2018/3/7  12.49  49.22
```

No.2 向前填充

运用fillna函数，输入参数method='ffill'，用缺失值所在列的前一个非NaN值进行补齐。以stock1为例。

```
in:stock1_ffill=stock1.fillna(method='ffill')
  stock1_ffill
out:     上海机场 浦发银行
2018/3/1  12.47  49.33
2018/3/2  12.41  50.35
2018/3/5  12.42  50.22
2018/3/6  12.49  50.22
2018/3/7  12.49  49.22
```

No.3 向后填充

运用fillna函数，输入参数method='bfill'用缺失值所在列的后一个非NaN值进行补齐。以stock1为例。

```
in:stock1_bfill=stock1.fillna(method='bfill')
  stock1_bfill
out:     上海机场 浦发银行
2018/3/1  12.47  49.33
2018/3/2  12.41  50.35
2018/3/5  12.42  50.22
2018/3/6  12.49  49.22
2018/3/7  12.49  49.22
```

3.数据框间的操作

对数据进行分析时，为了便于分析，往往会需要将不同的数据框进行拼接合并。拼接分为两类：按行上下拼接和按列左右拼接。在Pandas中，concat、merge和join主要用于数据框的拼接。本文以stock1，stock2和stock3为例进行演示。

stock2

	东方航空
2018/3/1	7.6060
2018/3/2	7.4473
2018/3/5	7.4870
2018/3/6	7.5961
2018/3/7	7.4771

stock3

	上海机场	浦发银行
2018/3/8	12.47	49.64
2018/3/9	12.48	50.83
2018/3/12	12.50	50.09

concat

concat 函数既可以用于行拼接，也可以用于列拼接。concat函数的具体用法为：

concat([数据框1，数据框2,...，数据框n],axis=0或1)

注意：axis=0表示按行拼接，axis=1表示按列拼接。

以stock1和stock3按行拼接，代码如下：

```
in: newstock=pd.concat([stock1,stock3],axis=0)
    newstock
out:     上海机场   浦发银行
2018/3/1  12.47  49.33
2018/3/2  12.41  50.35
2018/3/5  12.42  50.22
2018/3/6  12.49    NaN
2018/3/7  12.49  49.22
2018/3/8  12.47  49.64
2018/3/9  12.48  50.83
2018/3/12 12.50  50.09
```

以stock1和stock2按列拼接为例，代码如下：

```
in: newstock1=pd.concat([stock1,stock2],axis=1)
    newstock1
out:     上海机场 浦发银行 东方航空
2018/3/1  12.47  49.33  7.6060
2018/3/2  12.41  50.35  7.4473
2018/3/5  12.42  50.22  7.4870
2018/3/6  12.49    NaN  7.5961
2018/3/7  12.49  49.22  7.4771
```

merge

merge函数是对不同数据框按列进行拼接，需要明确放置在左侧（left)和右侧（right）的数据框。函数merge中的参数left_index表示是否按照左侧数据框的行索引拼接，True为是，False为否。参数right_index则代表是否按照右侧数据框的行索引拼接，用法和left_index一致。由于在本例题中，两个数据框的行索引完全相同，所以两个数据框的行索引都采取进行拼接。以stock1和stock2为例，代码如下：

```
in: pd.merge(left=stock1,right=stock2,left_index=True,right_index=True)

out:     上海机场 浦发银行 东方航空
2018/3/1  12.47  49.33  7.6060
2018/3/2  12.41  50.35  7.4473
2018/3/5  12.42  50.22  7.4870
2018/3/6  12.49  NaN    7.5961
2018/3/7  12.49  49.22  7.4771
```

join

join是对不同数据框进行按列拼接，用法为：数据框.join(参数设置)。join函数中的参数 on表示按照几个或者某几个索引进行合并，默认是依据两个数据框的行索引进行拼接。以stock1和stock2为例（结果同concat图2），代码如下：

```
in: stock1.join(stock2)
out:    上海机场 浦发银行 东方航空
2018/3/1  12.47  49.33  7.6060
2018/3/2  12.41  50.35  7.4473
2018/3/5  12.42  50.22  7.4870
2018/3/6  12.49    NaN  7.5961
2018/3/7  12.49  49.22  7.4771
```

4.相关重要函数

lambda函数

lambda函数也叫匿名函数，即没有具体名称的函数，它允许快速定义单行函数，可以用在任何需要函数的地方。lambda表达式是起到一个函数速写的作用。允许在代码内嵌入一个函数的定义。

lambda的表达式：lambda 参数：操作参数

由于lambda表达式是固定的，其本质上只有一种用法，那就是定义一个lambda函数。但事实上，根据这个lambda函数应用场景的不同，可以将lambda函数的用法扩展为以下几种：

No.1 赋值给变量

将lambda函数赋值给一个变量，通过这个变量间接调用该lambda函数。举个例子，将三个数相加。

```
in: ibd = lambda x,y,z : x+y+z
    print(ibd(1,2,3))
out:6
```

No.2 作为函数返回值

将lambda函数作为其他函数的返回值，返回给调用者。函数的返回值也可以是函数。当lambda作为函数返回值时，lambda函数实际上是定义在某个函数内部的函数，称之为嵌套函数，或者内部函数。

以关键字def定义函数，lambda作为返回值为例。这里定义了一个mint函数，返回了一个lambda表达式。其中lambda表达式获取到了上层def作用域的变量名x的值。a是mint函数的返回值。a(2)，即是调用了mint返回的lambda表达式。

```
in: def mint(x):
        return lambda y:x**y
    a = mint(22)
    print(a(2))
out:484
```

No.3 作为函数参数

将lambda函数作为参数传递给其他函数。部分Python内置函数接收函数作为参数，如map函数、sorted函数、reduce函数等。

这里，以reduce函数举例，求n=9的阶乘。reduce函数在库functools中，需要先导入。简单介绍一下reduce函数，其语法是：reduce(函数，序列)，这里函数参数是一个有两个参数的函数，reduce依次从序列中取一个元素，和上一次调用函数的结果共同做参数，再次调用函数。

```
in: import pandas as pd
    from functools import reduce
    n=9
    print(reduce(lambda x,y:x*y,range(1,n+1)))
out:362800
```

def 也是自定义函数的一种方法。
运用def语法时，函数的基本框架如下：

def 函数名（参数）：
函数主体
return 返回对象

def定义函数与lambda的区别在于：

1. def创建的函数是有名称的，而lambda没有。
2. lambda的后面只能有一个表达式，def则可以有多个。
3. 像if、for、print等语句不能用于lambda中，def可以。
4. lambda一般用来定义简单的函数，而def可以定义复杂的函数。

No.4 pandas的lambda函数

DataFrame中有apply函数，可以对DataFrame进行函数式操作，加快我们处理数据的效率。下面我们先生成一个DataFrame对象：

```
from pandas import DataFrame
import pandas as pd
import numpy as np
df = DataFrame(np.random.randn(4,4),columns=['a','b','c','d'])
df
```

	a	b	c	d
0	-1.420716	-0.207620	0.062200	-0.701441
1	0.493224	-0.118420	-0.794120	-1.085717
2	1.085720	0.032872	0.799283	0.680725
3	-1.293762	0.815264	1.698777	-0.312940

接着，我们来计算，每一列即a, b, c, d四列数字的最大值和最小值之差。我们可以编写如下代码，其中**lambda x** 就表示从第一列开始直到最后一列进行某种函数操作。这里的每列的数据类型正是Pandas的

Series对象。然后我们利用**x.max() - x.min()** 让每个Series的最大值减去了最小值。所以最下面的输出的最左边的索引就是我们的列名：a, b, c, d。

```
df.apply(lambda x:x.max()-x.min())
```

```
a    2.506436
b    1.022884
c    2.492897
d    1.766442
dtype: float64
```

如果我们想让每一行的最大值减去最小值，就需要调整我们读取的方向，从原来从左到右的读取列Series，变成从上到下读取行Series，执行这种操作需要加 axis=1。

```
df.apply(lambda x:x.max()-x.min(), axis=1)
```

```
0    1.482916
1    1.578941
2    1.052848
3    2.992539
dtype: float64
```

序列（Series）也可以进行lambda函数操作，只是无需指定axis，因为序列只有从上到下的方向，每次处理的x也就只有一个值。

时间数据处理

Python中有处理时间的datetime、time库，在Pandas中，datetime函数和to_datetime函数也有着类似的功能。由于Pandas对时间数据的处理有很多操作，在这里就简单介绍将非时间类型的数据，如字符串、Unicode等数据类型，转换成时间类型。

在我们导入外部文件数据时，时间、日期在Python并不是时间类型，很有可能是Unicode或者string等，因此我们在对时间数据进行其他操作时，就需要进行时间类型转换。

to_datetime函数可以将字符串型的日期转换成时间类型。to_datetime的语法：

to_ datetime(dateString, format)

datetime有年月日时分秒，其中年月日是必须要输入的参数，时分秒可以不输入，默认为零。因此，字符串型日期dataString必须包括年月日。

下图是datetime 格式的定义。

属性	注释
%Y	代表年份
%m	代表月份
%d	代表日期
%H	代表小时
%M	代表分钟
%S	代表秒

以转换读取clipboard形成的数据框stock1的日期格式为例:

```
in: type(stock1.index[0])
out:str

in: pd.to_datetime(stock1.index[0],format='%Y/%m/%d')
out:Timestamp('2018-03-08 00:00:00')
```

结语

总之，Pandas中纳入了大量能使我们快速便捷地处理数据的函数和方法。这篇文章也仅仅是浅尝辄止，粗略地介绍Pandas，想要系统性地了解、把握它，就需要利用Pandas中文网、CSDN论坛、计算机书籍等工具来进一步学习。希望这篇文章能够助小白们早日打怪升级，成为大佬！如果您是高段位码农，您可以在评论区留下您的真知灼见，期待Next Innovation!

本期撰写人：王佳妮
南京审计大学2018级CFA2班
邮箱：1772527606@qq.com
公众号：未生的半泽直树