在本章中，我们将介绍如何运用Python进行基本的股票数据分析，并综合运用前两章所述的Numpy和matplotlib相关知识。你将学会如何获取股票数据、运用Numpy分析这些数据、并使用matplotlib制作图表将数据可视化。

3.1 使用TuShare获取股票数据

**3.1.1 TuShare简介与下载安装**

数据分析的第一步是获取源数据。从互联网上获取股票数据的方式有很多，本章所介绍的是Python爬虫库TuShare，除股票信息外，该库还囊括了宏观经济、期货乃至电影票房等数据，适合Python数据分析学习者使用。

TuShare库的下载安装与其他Python库类似，可以通过pip命令进行。如果你已经安装了Python 3.x版本，则安装Python时就已经自带了pip功能。打开命令行窗口，输入pip install tushare命令，即可自动下载并安装matplotlib。

安装完成后，可以在命令行窗口中进入Python终端会话，对之前的安装进行测试。输入import tushare命令，尝试导入TuShare。若没有出现任何错误信息提示，则说明TuShare已经成功安装，可以使用。

**3.1.2 使用TuShare获取股票历史数据**

安装成功后，输入import tushare即可导入TuShare库。通过调用TuShare中的get\_hist\_data()函数并在参数中指定股票代码、开始日期、结束日期，我们可以获取某支股票在指定日期范围内的历史股价。下面以查询中国石油（601857）从2019年1月28日到2019年3月11日的历史股价为例：

|  |
| --- |
| >>>import tushare as tu  >>>data=tu.get\_hist\_data('601857',start='2019-01-28',end='2019-03-11')  >>>data.to\_csv('d:/petro\_china.csv', header=None) |

我们调用了TuShare中的函数get\_hist\_data以获取历史股票数据，并且在调用时设置了三个参数：第一个参数为股票代码字符串，此处我们将中国石油的股票代码601857以字符串类型传递给函数；第二和第三个参数为股票历史数据的开始和结束日期，按照yyyy-mm-dd的格式，以字符串类型传递给函数，这里我们选取从2019年1月28日到2019年3月11日的股票数据。按如上所述传入参数调用get\_hist\_data()函数后，我们将函数的返回值存储在对象data中。（事实上，get\_hist\_data()函数的返回值是数据分析库pandas的DataFrame对象。对pandas感兴趣的读者可访问其官方网站浏览介绍与教程：[https://pandas.pydata.org/](https://pandas.pydata.org/pandas-docs/stable/getting_started/tutorials.html) ）

此时，输入data以显示其存储内容，窗口中将显示一张数据表，表中每列信息的含义如下：

|  |  |
| --- | --- |
| date | 日期 |
| open | 开盘价 |
| high | 最高价 |
| close | 收盘价 |
| low | 最低价 |
| volume | 成交量 |
| price\_change | 价格变动 |
| p\_change | 涨跌幅 |
| ma5 | 5日均价 |
| ma10 | 10日均价 |
| ma20 | 20日均价 |
| v\_ma5 | 5日均量 |
| v\_ma10 | 10日均量 |
| v\_ma20 | 20日均量 |
| turnover | 换手率 |

为了在关闭程序以后仍能继续使用获取的数据，我们可以将data对象的数据内容输出为csv格式进行保存，这是一种常用于保存数据的文件格式。通过调用data对象的to\_csv方法，并在参数中指定文件存储路径和名称，就能将data对象中的股票数据保存为csv文件。

3.2 分析股票数据

在获取了股票数据后就可以使用NumPy进行简单的数据分析了。首先，我们需要新建一个源代码文件，本文的后续代码均将该文件中添加。

**3.2.1 从csv源文件中导出数据**

|  |
| --- |
| **import** numpy **as** np close,volume=np.loadtxt(**"petro\_china.csv"**, delimiter=**','**, usecols=(3,5), unpack=**True**, skiprows=1) |

虽然原始数据中包含相当丰富的信息，不过目前我们暂且先研究“收盘价”和“成交量”这两个信息。在读取csv文件时，我们使用了NumPy库中的loadtxt()函数。在调用该函数时，我们还设置了若干参数：

第一个参数是要读取的文件名（如果该文件和Python源代码文件不在相同目录下，则需要给出完整路径）；

随后，delimiter参数确定了分隔符，即遇到哪个字符就自动分割数据；usecols用于指定原始数据表中哪几列数据要被导出。由于在原始数据文件中，收盘价close位于第4列，而成交量volume位于第6列，而列号又从0开始，因此我们传入一个二元组(3, 5)表示需要导出的列分别为第4和第6列；

接下来，unpack参数设置为True，表示会将导出的各列数据分开到各变量中存放；

最后，skiprows参数指定了要跳过的开头行数（注意不是行号）。由于在我们之前保存的csv文件中，第一行为表头而非数据，因此需要跳过的开头行数为1，即skiprows应取1。

**3.2.2 计算股票均价**

通过3.2.1的操作，我们从数据源csv文件中获取了中国石油从2019年1月28日到2019年3月11日的收盘价（存储在close中）和成交量（存储在volume中）。接下来就可以运用NumPy对股票数据进行分析。首先我们计算股票的各项均价：

|  |
| --- |
| *# 1. 计算成交量加权平均价* vwap = np.average(close,weights=volume) print(**'成交量加权平均价 vwap='**,vwap)  *# 2. 计算收盘时算术平均价* mean = np.mean(close) print(**'收盘时算术平均价 mean='**,mean)  *# 3. 计算收盘时加权平均价（时间与现在越近，权重越大）* t = np.arange(len(close)) twap = np.average(close,weights=t) print(**'收盘时加权平均价 twap='**,twap) |

首先计算成交量加权平均价，我们使用NumPy的average()函数进行计算。其中，权重由成交量决定，因此参数weights（权重）取volume。

接下来计算收盘时的算术平均价。通过mean函数即可计算算术平均值，因此我们调用np.mean()，并将close作为参数传入，进行计算。

第三个要计算的均值是收盘时的加权平均价，我们设定，时间距离现在越近的股价，权重越大。因此我们调用np.arange()函数创建一个数组weight。该数组长度与close相等，而且数组的每个元素依次增大1，故而满足“距现在越近，权重越大”这一要求，可以作为权重数组传递给参数weights。

经过上述计算，我们可以得到如下输出结果：

|  |
| --- |
| 成交量加权平均价 vwap= 7.72441662423  收盘时算术平均价 mean= 7.60807692308  收盘时加权平均价 twap= 7.48483076923 |

**3.2.3 计算股价极值与极值波动范围**

接下来，我们需要找出股价的最高和最低值，并计算最高值和最低值的波动范围：

|  |
| --- |
| *# 4. 获取最高价、最低价* high, low = np.loadtxt(**'petro\_china.csv'**, delimiter=**','**,usecols=(2,4),unpack=**True**, skiprows=1) print(**'该时段每日最高价：'**) print(high) *# 输出最高价* print(**'该时段每日最低价：'**) print(low) *# 输出最低价* highest = np.max(high) print(**'时段内历史最高价 highest='**, highest) lowest = np.min(low) print(**'时段内历史最低价 lowest='**, lowest) average = (highest + lowest) / 2 print(**'最值均值 average = '**,average)  *# 5. 计算最大值和最小值的波动范围* print(**'最大值波动范围='**,np.ptp(high)) print(**'最小值波动范围='**,np.ptp(low)) |

在这段代码中，我们首先仍然从数据源文件获取了每日最高价和最低价，它们分别位于csv文件的第3和第5列。将每日最高价和最低价分别存放在high和low中后，我们对其进行输出显示。接下来，通过NumPy的min()和max()函数找到历史最高价中的最大值和历史最低价中的最小值，分别保存在highest和lowest当中，并根据其计算均值average。

最后，我们调用了np.ptp()函数，分别计算high和low两个数组当中最大值和最小值的差，从而计算出历史最高价和历史最低价的波动范围。

上述代码的输出结果如下所示：

|  |
| --- |
| 该时段每日最高价：  [ 7.71 7.89 7.97 8.02 7.9 8.02 7.89 7.87 7.94 7.89 7.85 7.57  7.6 7.6 7.62 7.58 7.54 7.53 7.57 7.41 7.38 7.35 7.33 7.29  7.29 7.33]  该时段每日最低价：  [ 7.6 7.66 7.89 7.85 7.81 7.84 7.78 7.78 7.77 7.73 7.58 7.45  7.5 7.54 7.53 7.49 7.45 7.47 7.39 7.33 7.3 7.28 7.25 7.24  7.21 7.26]  时段内历史最高价 highest= 8.02  时段内历史最低价 lowest= 7.21  最值均值 average = 7.615  最大值波动范围= 0.73  最小值波动范围= 0.68 |

**3.2.4 计算股价中位数和方差**

本节将介绍如何计算股票价格的中位数和方差。由于中位数和方差计算所需的原始数据——收盘价，已经在上一节中获取并保存在close中，因此我们可以直接使用close中的数据。中位数的计算有两种方法：

|  |
| --- |
| *# 6.1 计算中位数方法一* median1 = np.median(close) print(**'中位数1 median1 = '**,median1)  *# 6.2 计算中位数方法二* sorted = np.msort(close) print(**'sorted = '**,sorted) N = len(close) middle = sorted[int((N-1)/2)] print(**'middle = '**,middle) median2 = (sorted[int(N/2)] + sorted[int((N-1)/2)])/2 print(**'中位数2 median2 = '**,median2) |

如上所示，第一种方法是直接调用NumPy的median()函数；第二种方法是先排序，再取中间的数。最终，上面代码段的输出结果如下所示：

|  |
| --- |
| median = 7.575  sorted = [ 7.25 7.27 7.27 7.32 7.35 7.36 7.41 7.45 7.51 7.52 7.53 7.57 7.57 7.58 7.59 7.66 7.71 7.8 7.83 7.83 7.85 7.87 7.88 7.91 7.94 7.98]  middle = 7.57  average\_middle = 7.575 |

计算方差的方法也有两种。第一种是直接调用NumPy函数var()进行计算；第二种是根据方差的定义去列式计算方差：

|  |
| --- |
| *# 6.3 计算方差方法一* variance1 = np.var(close) print(**'方差1 variance = '**,variance1)  *# 6.4 计算方差方法二* variance2 = np.mean((close - close.mean())\*\*2) print(**'方差2 variance\_from\_definition = '**,variance2) |

**3.2.5 计算股票收益率**

本节介绍对股票收益率三种不同指标的计算，即普通收益率、对数收益率和收益波动率。计算收益率所需的的原始数据主要是股票的收盘价，它在3.2.1节被我们保存在了变量close中。以收盘价为基础，我们首先计算每一天的收盘价相比前一天的变化量。使用NumPy的diff()函数即可计算该变化量，我们将其保存在数组diff中：

|  |
| --- |
| diff = np.diff(close)  print(**'diff ='**,diff) |

上面代码段的输出结果为：

|  |
| --- |
| diff = [-0.05 0.28 0.04 -0.11 0.04 -0.03 -0.05 0. -0.03 0.05 -0.28 -0.05 0.06 0.01 -0.02 -0.12 0.06 0.02 -0.12 -0.05 -0.01 -0.03 -0.07 0.02 0. ] |

随后我们即可根据数组diff计算普通收益率，并进一步根据普通收益率计算收益率标准差。普通收益率由每日股价相比前一日变化量与前一日股价相除获得，而收益率标准差可以通过调用np.std()函数计算出来：

|  |
| --- |
|  |

3.3 TODO:绘制分析数据统计图（七，八节）

# end