第 3 章

Pandas 数据提取

3.1 Pandas 简介

3.1.1 Pandas 数据分析

在 Python 中存在大量处理 Excel 的第三方库。在早期,用户可以通过 xlrd 库读取 xls、xlsx 文件,利用 xlwt 库写入 xls 文件,或者利用 xlutils 修改 xls 文件。这些库都有一个共同的特点:无法批量处理 Excel 文件。随着更多第三方库的加入,用户如果需要更高效地进行批量文件处理及数据分析,则可以采用 Pandas、openpyxl 及 xlwings 结合应用,实现远比 Excel 中 VBA 更为强大的功能。例如用户可以利用 Pandas 进行数据分析、利用 Matplotlib 或 Seaborn 生成数据图表、利用 xlwings 进行文件的批量格式处理及文件的读存,实现更高层级的 BI(商业智能)。

本章主要介绍的是使用 Pandas 读取工作簿。对于刚接触 Pandas 的用户可能会有这样的疑问:什么是 Pandas? Pandas 能干什么? Pandas 数据分析或 Python 数据分析到底是怎么一回事? 简单来讲:Pandas 是 Python 的一个数据分析包,Pandas 的名称来自面板数据(Panel Data)和 Python 数据分析(Data Analysis)。Pandas 的开发者是 Wes McKinney(韦斯·麦金尼),该软件是他当时在纽约的 AQR Capital Management(AQR 资本投资管理公司)工作时,因为对 Excel 及 R 语言的数据清洗与汇总统计都感到不满意后而决定自己开发一个基于 Python 的数据分析软件。该软件于 2008 年完成了第 1 版,并于 2009 年 12 月向公众发布。可以这样说,Pandas 是 Python 数据分析的利器。正是因为 Pandas 的加入才让 Python 在数据分析领域有了一席之地。Pandas 在数据的增、删、改、查及时间序列等方面,功能十分强大。

Pandas 中主要有 Series 和 DataFrame 两种数据结构。Series:一种类似于一维数组的对象,它是由一组数据及一组与之相关的数据标签(索引)组成的。DataFrame:一张表格型的数据结构,类似 Excel、SQL 表格等二维数据结构。含有一组有序的列,每列可以是不同的值类型(数值、字符串、布尔值等),DataFrame 既有行索引,也有列索引,可以被看作由 Series 组成的字典。

注意：Series 首字母必须大写，DataFrame 采用的是驼峰式写法，D 与 F 必须大写。

Pandas 是 Python 的第三方库。按照常规约定，在使用 Pandas 之前，必须导入 Pandas 包，并以 pd 作为其别名，命令如下：

```
import pandas as pd
```

如果在 Pandas 的使用过程中同时用到 NumPy，则需同时导入 NumPy 包，并以 np 作为别名，命令如下：

```
import numpy as np
import pandas as pd
```

3.1.2 SMED 与数据分析

SMED(Single Minute Exchange of Die)是“六十秒即时换模”的简称，是精益生产中的一个术语。在以往的生产制造过程中，产品更换模具及生产换线调整过程中总是要浪费大量的时间，从而造成生产的停顿与等待，当现场管理水平跟不上时，以往的做法总会采用批量生产(Mass Production)方式来减少更换模具或生产换线的时间，其最终的结果是“停顿的时间减少了，但库存的数量却猛增了”，这种生产方式显然无法应对弹性生产的需求，直至 SMED(快速换模)得以推广后才真正将生产停顿及降低库存这两个问题一并得以解决，是对传统生产现场管理的革命性创新。

SMED 的技术可以概括为“区分内作业与外作业，将内作业转换为外作业；减少外作业时间，减少内作业时间”。如果将此技术转换到数据清洗及数据分析场合中，则可以如下理解。

(1) 区分规范数据与非规范数据、无效数据及冗余数据。

(2) 在数据分析之前将不规范的数据转换为规范数据。

(3) 在数据分析之时减少冗余，仅导入所需的数据。

(4) 优化数据分析的代码，利用更快的工具、更好的方法快速处理数据，如图 3-1 所示。

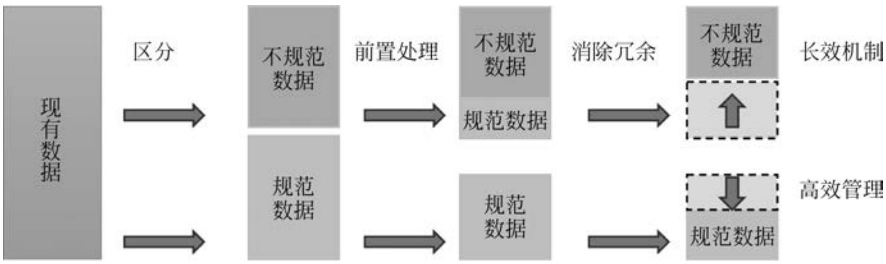


图 3-1 SMED 使用流程

Python 的代码一向以“优雅、简洁、高效”著称，Python 的数据分析利器 Pandas 的代码同样以“灵活、高效、功能强大”著称，与 SMED 的理念高度吻合，以至于用户一提起 Python 数据分析首先想到的就是 Pandas。

另外，之所以倡导在数据分析前将内作业转换为外作业是因为“在数据规范度较差的企

业，数据分析师常常将80%以上的时间用于数据清洗，而仅将不到20%的时间用于数据分析"，这是一个本末倒置的事情。

需要说明的是：Excel及其Power Query、Excel数据分析与Python数据分析、Python中的openpyxl库与xlwings库等，这些都是解决问题的工具。对问题的解决而言，工具的利用可以在最优解与次佳方案之间抉择，也可以让二者进行协同作业，从而使各工具的效能最大化。

3.1.3　Excel数据的读取流程

在Pandas中，可以通过pd.read_csv()、pd.read_excel()、pd.ExcelFile.parse()、pd.read_json()、pd.read_sql()等众多方式读取来自不同数据源的数据。在本章节将详细列举使用pd.read_excel()或pd.ExcelFile.parse()将Excel文件读取到DataFrame。这两个函数及解析方法参数众多，对于初学者不易理解与记忆。考虑到Pandas读取与转换Excel数据时与Excel中的Power Query读取与转换有一些类似的地方，现采用对比学习的方式，对相关读取流程做一些简要的整理，顺带了解Power Query这门常用且易用的数据分析工具：

（1）在微软Excel的Power Query中，用户通常会通过"数据"→"获取数据"→"自文件"→"从Excel工作簿"获取来自指定路径的工作簿；或者可以通过"数据"→"来自表格/区域"，获取来自当前工作簿中的数据。相当于Pandas中pd.read_excel()或pd.ExcelFile()函数中的io参数，如图3-2所示。

(a) 数据获取方式(1)　　　　(b) 数据获取方式(2)

图3-2　利用Power Query获取Excel工作簿(1)

如果使用的是微软Power BI中的Power Query，则在启运Power BI后，通常会通过选择功能区的"主页"→"Excel工作簿"或单击报表画布区的"从Excel导入数据"获取Excel

工作簿，如图 3-3 所示。

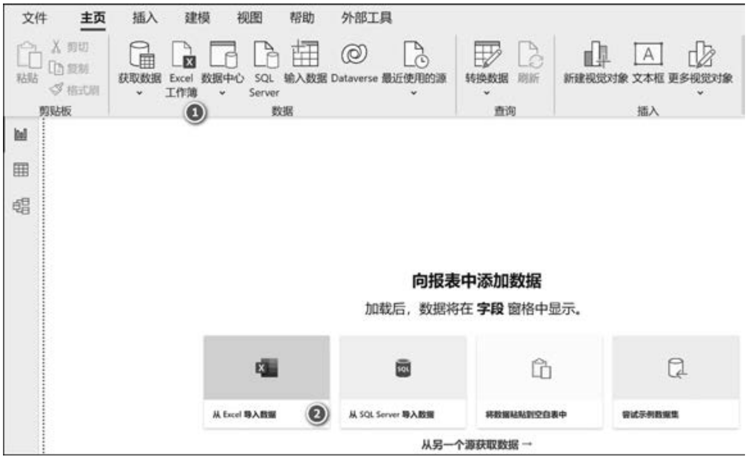


图 3-3　利用 Power Query 获取 Excel 工作簿(2)

(2) 该步骤及后续的操作步骤，在 Excel 或 Power BI 的 Power Query 中大体相似，不过 Power BI 中的 Power Query 功能更为强大。当 Power Query 读取的工作簿中包含多个工作表时，可以在导航器窗格中选择全部、部分或某工作表；该步骤相当于 Pandas 中 sheet_name 参数的应用，默认为只选择第 1 个工作表，如果为 None，则为选择全部工作表，当然也可以通过列表的方式，选择部分工作表，例如，sheet_name =[0,1, '合肥']。Power Query 导航器窗格中工作表的选择如图 3-4 所示。

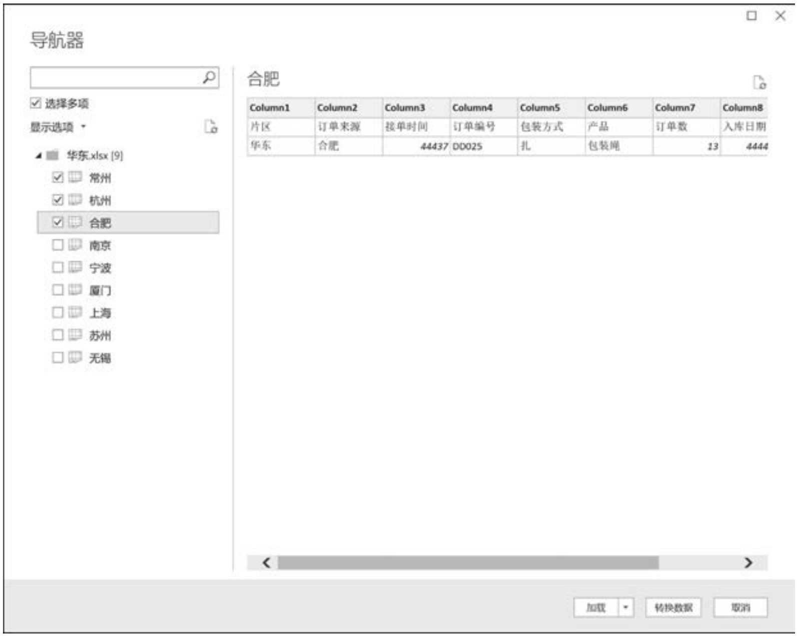


图 3-4　在 Power Query 导航器窗格中选择工作表

(3) 在 Power Query 中,在默认情况下会将数据的首行提升为标题。这相当于 Pandas 中 header 参数的应用。

(4) 在 Power Query 中,在将首行提升为标题后,系统会默认识别并自动更改数据的类型。这相当于 Pandas 中 dtype 及 converters 参数的应用。

(5) 在 Power Query 中,如果只想导入需要的列,则可以通过“选择列”或“删除其他列”的方式导入。这相当于 Pandas 中 usecols 参数的应用。

(6) 在 Power Query 中,当前面几行为空行时,可通过“主页”→“删除行”→“删除前面几行”,然后将首行提升为标题。这相当于 Pandas 中 skiprows 参数的应用,而 nrows 参数则类似于“主页”→“保留行”→“保留最前面几行”的应用;skipfooter 参数则类似于“主页”→“保留行”→“保留最后几行”的应用。其他不再一一举例。Power Query 编辑器中的图形化操作如图 3-5 所示。

图 3-5 在 Power Query 中删除行

3.2 pd.read_excel()函数

pd.read_excel()函数是 Pandas 中将 Excel 读取到 DataFrame 的一个函数。该函数共有 20 多个参数,除第 1 个参数 io 外,其他参数均为可选参数。该函数适用于各类复杂应用场景,功能十分强大。它可以读取来自本地端及 URL 中的 xls、xlsx、xlsm、xlsb 等格式的文件,支持读取工作簿中单个或多个工作表;如果为多个工作表,则各工作表的名称必须存放于一个列表中,其语法如下:

```
pd.read_excel(io, sheet_name = 0, *, header = 0, names = None, index_col = None, usecols =
None, squeeze = None, dtype = None, engine = None, converters = None, true_values = None, false_
values = None, skiprows = None, nrows = None, na_values = None, keep_default_na = True, na_
filter = True, verbose = False, parse_dates = False, date_parser = None, thousands = None,
decimal = '.', comment = None, skipfooter = 0, convert_float = None, mangle_dupe_cols = True,
storage_options = None)[source]
```

3.2.1 路径参数

pd.read_excel()函数的第 1 个参数(io)可以为有效的字符串路径、Excel 文件、带 Excel 工作簿的 URL 等。第 1 个参数 io 可以是当前路径、相对路径或绝对路径。

(1) 采用当前路径表示,代码如下:

```
pd.read_excel('华东.xlsx')
```

(2) 采用相对路径表示,代码如下:

```
pd.read_excel(r'.\华东.xlsx')
```

(3) 采用绝对路径表示,代码如下:

```
pd.read_excel(r'D:\数据源\B文件\华东.xlsx')
```

在以上代码中r为转义字符。本书后续章节主要采用文件的绝对路径进行讲解。另外,本书95%以上案例演示所用的数据源为工作簿(订单表.xlsx)及其子集(华东.xlsx),子集的拆分过程详见第10章(ch10d041.py)。本章第4节和第5节示例工作簿(上海.xlsx)是工作簿(华东.xlsx)的子集。

3.2.2 引擎参数

Pandas读取Excel工作簿时所支持的引擎为xlrd(.xls格式)、openpyxl(.xlsx等最新版本格式)、odf(odf、ods、odt格式)、pyxlsb(二进制Excel文件)。Pandas在读取xls时调用xlrd,在读取xlsx时调用xlrd或openpyxl。在写入Excel时调用xlwt、openpyxl或xlswriter。

在Anaconda安装环境中,系统已自动安装并配置好了以上Pandas依赖项。在默认情况下,该参数的值将设置为None,系统会自动检测并使用适配的engine。

3.3 工作表名称

pd.read_excel()函数的sheet_name参数的默认值为0。①在Pandas中,sheet的索引值是从0开始的,0代表的是工作簿中的第1个工作表(sheet),1代表的是第2个sheet。②sheet_name的参数可以用0、1等整数表示,也可以用“Sheet1、Sheet2”等字符串表示。③当sheet_name导入的是一个列表时,可以存在整数与字符串混用的情形,例如[0,1,'Sheet3'],返回的值是一个DataFrame字典。④当sheet_name=None时,获取的是工作簿中所有工作表的内容。

3.3.1 选择工作簿中的所有工作表

当sheet_name=None时,获取的是工作簿中所有的工作表,代码如下:

```
#ch03d001.py
import pandas as pd
df = pd.read_excel(r'D:\数据源\B文件\华东.xlsx',sheet_name = None)
type(df)
```

返回的值为dict。采用列表推导式获取工作簿中所有的工作表名,代码如下:

```
[i for i in pd.read_excel(r'D:\数据源\B 文件\华东.xlsx', sheet_name = None)]
```

返回的值如下：

```
['上海', '南京', '厦门', '合肥', '宁波', '常州', '无锡', '杭州', '苏州']
```

获取字典中的指定 DataFrame，代码如下：

```
df1 = pd.read_excel(
        r'D:\数据源\B 文件\华东.xlsx',
        sheet_name = None
    )
df1['上海']
```

返回的值如图 3-6 所示。

	片区	订单来源	接单时间	订单编号	包装方式	产品	订单数	入库日期	入库数
0	华东	上海	2020-06-30	DD003	箱装	苹果醋	2	2020-07-03 00:03:21	2
1	华东	上海	2020-08-02	DD005	桶装	油漆	2	2020-08-02 09:07:18	2
2	华东	上海	2021-10-01	DD028	散装	钢化膜	4	2021-10-03 07:24:21	4

图 3-6 获取工作簿中的工作表(上海)

3.3.2 选择工作簿中指定的工作表

pd.read_excel()函数的 sheet_name 参数可为 str(字符串)、int(整数)、list(列表)。以 sheet_name=[0,1]为例，获取工作簿中的第 1 个和第 2 个工作表，代码如下：

```
df2 = pd.read_excel(
        r'D:\数据源\B 文件\华东.xlsx',
        sheet_name = [0,1]
    )
df2
```

第 2 个参数的 sheet_name=[0,1]可简写成[0,1]，读取的是第 1 个和第 2 个 sheet，生成的是 1 个字典，键为 0 和 1，值为两个 DataFrame。返回的值如下：

```
{0:  片区 订单来源  接单时间  订单编号 包装方式   产品  订单数  入库日期  入库数
0 华东   上海 2020-06-30  DD003   箱装  苹果醋    2 2020-07-03 00:03:21    2
1 华东   上海 2020-08-02  DD005   桶装    油漆    2 2020-08-02 09:07:18    2
2 华东   上海 2021-10-01  DD028   散装  钢化膜    4 2021-10-03 07:24:21    4,
1:   片区 订单来源  接单时间  订单编号 包装方式   产品  订单数  入库日期  入库数
0 华东   南京 2020-10-02  DD011   桶装    油漆    9 2020-10-02 06:22:05    9
1 华东   南京 2021-10-11  DD029     袋  老陈醋    9 2021-10-30 08:09:16    9}
```

此时可通过 df2[0]获取第 1 个 DataFrame，可通过 df2[1]获取第 2 个 DataFrame sheet_name 列表中的值可以是整型数值和字符串的混合，代码如下：

```
df3 = pd.read_excel(
```

```
    r'D:\数据源\B 文件\华东.xlsx',
    sheet_name = [0,1,'无锡']
)
df3
```

返回的是 1 个字典，键为“0、1、无锡”，内容如下：

```
{0:  片区 订单来源   接单时间   订单编号 包装方式    产品   订单数   入库日期   入库数
0 华东    上海 2020-06-30  DD003   箱装  苹果醋     2 2020-07-03 00:03:21    2
1 华东    上海 2020-08-02  DD005   桶装   油漆     2 2020-08-02 09:07:18    2
2 华东    上海 2021-10-01  DD028   散装  钢化膜     4 2021-10-03 07:24:21    4,
1:  片区 订单来源   接单时间   订单编号 包装方式    产品   订单数   入库日期   入库数
0 华东    南京 2020-10-02  DD011   桶装   油漆     9 2020-10-02 06:22:05    9
1 华东    南京 2021-10-11  DD029    袋  老陈醋     9 2021-10-30 08:09:16    9,
'无锡': 片区 订单来源   接单时间   订单编号 包装方式    产品   订单数   入库日期   入库数
0 华东    无锡 2021-07-27  DD018    膜  异形件     2 2021-07-28 06:42:49    2}
```

3.3.3　选择工作簿中具体的某一工作表

当工作簿中存在多个工作表时，如果未对工作表的位置或工作表名称进行指定，则默认导入的是第 1 个工作表。也可以通过具体的指定位置或表名获取具体的某一工作表，代码如下：

```
df4 = pd.read_excel(r'D:\数据源\B 文件\华东.xlsx',7)
df4
```

在华东.xlsx 工作簿中，第 8 个工作表的名称为“杭州”，所以也可以采用以下代码：

```
df5 = pd.read_excel(r'D:\数据源\B 文件\华东.xlsx','杭州')
df5
```

df4 及 df5 返回的值如图 3-7 所示。

	片区	订单来源	接单时间	订单编号	包装方式	产品	订单数	入库日期	入库数
0	华东	杭州	2021-07-27	DD022	散装	钢化膜	7	2021-08-03 16:31:42	7
1	华东	杭州	2021-08-03	DD020	桶装	油漆	8	2021-08-03 08:20:33	8
2	华东	杭州	2021-08-03	DD021	捆	木材	13	2021-08-04 08:58:55	13

图 3-7　获取工作簿中的工作表（杭州）

3.4　标题

pd.read_excel()函数的标题参数为 header，可设置为 int、list of int，默认值为 0。

3.4.1　提升为标题

pd.read_excel()函数的 header 参数可为 str（字符串）、int（整数）、list（列表）。在默认情况下，将第 1 行提升为标题，即 header＝0，代码如下：

```
#ch03d002.py
import pandas as pd
df1 = pd.read_excel(r'D:\数据源\B文件\上海.xlsx')
df1
```

返回的值如图 3-8 所示。

	A	B	C	D
1	华东	上海	2020-06-30 00:00:00	DD003
2	华东	上海	2020-08-02 00:00:00	DD005
3	华东	上海	2021-10-01 00:00:00	DD028

(a) 数据源

	华东	上海	2020-06-30 00:00:00	DD003
0	华东	上海	2020-08-02	DD005
1	华东	上海	2021-10-01	DD028

(b) header=0

图 3-8　将第 1 行提升为标题

3.4.2　不需要标题

如果数据源中不存在标题，则可以将 header 设置为 None(不需要标题)，代码如下：

```
df2 = pd.read_excel(r'D:\数据源\B文件\上海.xlsx',header =
None)
df2
```

	0	1	2	3
0	华东	上海	2020-06-30	DD003
1	华东	上海	2020-08-02	DD005
2	华东	上海	2021-10-01	DD028

图 3-9　不需要标题

返回的值如图 3-9 所示。

对于确实不存在列名的数据可以通过 name 参数命名列。names 参数可为 array-like，默认值为 None。利用 names 构建列名，代码如下：

```
df3 = pd.read_excel(
    r'D:\数据源\B文件\上海.xlsx',
    header = None,
    names = ['片区','订单来源','接单时间','订单编号']
)
df3
```

将以上 names 的列表改成元组也是允许的，代码如下：

```
tpl = ('片区','订单来源','接单时间','订单编号')
df4 = pd.read_excel(
    r'D:\数据源\B文件\上海.xlsx',
    header = None,
    names = tpl
)
df4
```

在日常操作时，为了防止数值被提升为标题，在 Pandas 数据导入的过程中，经常会先设置 header＝None，然后通过 names 来指定标题/列名。以上代码返回的值如图 3-10 所示。

当数据的标题位于第 3 行时，可以将第 3 行(header＝2)指定为标题，代码如下：

```
df5 = pd.read_excel(
    r'D:\数据源\B 文件\上海.xlsx',
    sheet_name = 1,
    header = 2)
df5
```

	片区	订单来源	接单时间	订单编号
0	华东	上海	2020-06-30	DD003
1	华东	上海	2020-08-02	DD005
2	华东	上海	2021-10-01	DD028

图 3-10　利用 names 参数构建列名

返回的值如图 3-11 所示。

	A	B	C	D
1				
2				
3	片区	订单来源	接单时间	订单编号
4	华东	上海	2020-06-30 00:00:00	DD003
5	华东	上海	2020-08-02 00:00:00	DD005
6	华东	上海	2021-10-01 00:00:00	DD028

(a) 数据源

	片区	订单来源	接单时间	订单编号
0	华东	上海	2020-06-30	DD003
1	华东	上海	2020-08-02	DD005
2	华东	上海	2021-10-01	DD028

(b) header=2

图 3-11　将指定的行提升为标题

3.4.3　多层索引

当 Excel 中存在合并单元格及多级标题情况时，在 Pandas 中可以利用 list of int 来构建多层索引，代码如下：

```
df6 = pd.read_excel(
    r'D:\数据源\B 文件\上海.xlsx',
    sheet_name = 2,
    header = [0,1])
df6
```

返回的值如图 3-12 所示。

	A	B	C	D
1	渠道信息		订单信息	
2	片区	订单来源	接单时间	订单编号
3	华东	上海	2020-06-30 00:00:00	DD003
4	华东	上海	2020-08-02 00:00:00	DD005
5	华东	上海	2021-10-01 00:00:00	DD028

(a) 数据源

	渠道信息		订单信息	
	片区	订单来源	接单时间	订单编号
0	华东	上海	2020-06-30	DD003
1	华东	上海	2020-08-02	DD005
2	华东	上海	2021-10-01	DD028

(b) header=[0, 1]

图 3-12　构建多层索引

3.5　处理行

pd.read_excel()函数与行相关的参数有 skiprows、nrows、skipfooter。

3.5.1　跳过行

skiprows 参数可为 int、list-like 或 callable。当参数为 callable 时，参数允许为函数（含自定义函数及匿名函数）。以 skiprows＝list-like 为例，跳过前两行，代码如下：

```
#ch03d003.py
import pandas as pd
df1 = pd.read_excel(
        r'D:\数据源\B文件\上海.xlsx',
        sheet_name = 1,
        skiprows = [0,1]
)
df1
```

返回的值如图3-13所示。

	A	B	C	D
1				
2				
3	片区	订单来源	接单时间	订单编号
4	华东	上海	2020-06-30 00:00:00	DD003
5	华东	上海	2020-08-02 00:00:00	DD005
6	华东	上海	2021-10-01 00:00:00	DD028

(a) 数据源

	片区	订单来源	接单时间	订单编号
0	华东	上海	2020-06-30	DD003
1	华东	上海	2020-08-02	DD005
2	华东	上海	2021-10-01	DD028

(b) skiprows=[0, 1]

图3-13　跳过行(1)

如果需要跳过的行位于标题前及数据中,则也是允许的,代码如下:

```
df2 = pd.read_excel(
        r'D:\数据源\B文件\上海.xlsx',
        sheet_name = 3,
        skiprows = [0,1,4,6]
)
df2
```

返回的值如图3-14所示。

	A	B	C	D
1				
2				
3	片区	订单来源	接单时间	订单编号
4	华东	上海	2020/6/30	DD003
5				
6	华东	上海	2020/8/2	DD005
7				
8	华东	上海	2021/10/1	DD028

(a) 数据源

	片区	订单来源	接单时间	订单编号
0	华东	上海	2020-06-30	DD003
1	华东	上海	2020-08-02	DD005
2	华东	上海	2021-10-01	DD028

(b) skiprows=[0, 1, 4, 6]

图3-14　跳过行(2)

skiprows参数用于跳过前面的行。如果需要跳过后面的行,则需要用skipfooter参数,该参数的值为int数据类型,代码如下:

```
df3 = pd.read_excel(
        r'D:\数据源\B文件\上海.xlsx',
        sheet_name = 2,
        header = [0,1],
        skipfooter = 1
)
df3
```

返回的值如图 3-15 所示。

	A	B	C	D
1	渠道信息		订单信息	
2	片区	订单来源	接单时间	订单编号
3	华东	上海	2020-06-30 00:00:00	DD003
4	华东	上海	2020-08-02 00:00:00	DD005
5	华东	上海	2021-10-01 00:00:00	DD028

(a) 数据源

	渠道信息		订单信息	
	片区	订单来源	接单时间	订单编号
0	华东	上海	2020-06-30	DD003
1	华东	上海	2020-08-02	DD005

(b) skipfooter=1

图 3-15　跳过行(3)

3.5.2　保留行

如果需要保留数据中的最前面几行，则可以用 nrows 参数，nrows 的值为 int 数据类型。以保留最前面两行数据为例，代码如下：

```
df4 = pd.read_excel(
        r'D:\数据源\B 文件\上海.xlsx',
        sheet_name = 3,
        skiprows = [0,1,4,6],
        nrows = 2
)
df4
```

返回的值如图 3-16 所示。

Pandas 的行筛选功能相当强大，这些会放在后续章节重点讲解。

	片区	订单来源	接单时间	订单编号
0	华东	上海	2020-06-30	DD003
1	华东	上海	2020-08-02	DD005

图 3-16　保留行

3.6　选择列

pd.read_excel()函数的选择列参数为 usecols，可以采用位置索引(整型数字)或列名索引(文本字符串)方式进行列的选择。可设置为 str、list-like 或 callable，默认值为 None。当未填写 usecols 或指定 usecols＝None 时，选择所有列。

3.6.1　指定索引列

index_col 的默认值为 None，可选 str、int、list of int，生成索引。

应用举例，index_col 的值为字符串，代码如下：

```
#ch03d004.py
import pandas as pd
df1 =  pd.read_excel(
    r'D:\数据源\B 文件\订单表.xlsx',
    index_col = '片区'
)
df1.loc['西南']
```

或 index_col 的值为 int，代码如下：

```
df2 = pd.read_excel(
    r'D:\数据源\B文件\订单表.xlsx',
    index_col = 0
)
df2.loc['西南']
```

返回的值如图 3-17 所示。

	订单来源	接单时间	订单编号	包装方式	产品	订单数	入库日期	入库数
片区								
西南	成都	2020-05-31	DD002	箱装	蛋糕纸	2	2020-06-03 06:49:22	2
西南	重庆	2020-09-02	DD006	桶装	油漆	5	2020-09-02 01:14:04	4
西南	重庆	2021-08-28	DD023	桶装	油漆	11	2021-08-28 08:24:21	11

图 3-17　指定索引列(1)

index_col 的值为 list of int,代码如下:

```
df3 = pd.read_excel(
    r'D:\数据源\B文件\订单表.xlsx',
    index_col = [0,1]
)
df3.head()
```

df.head()函数用于获取前 5 条数据,head()函数的括号内可填写需获取的具体条数;df.tail()函数用于获取后 5 条数据。返回的值如图 3-18 所示。

		接单时间	订单编号	包装方式	产品	订单数	入库日期	入库数
片区	订单来源							
华北	北京	2020-04-28	DD001	箱装	蛋糕纸	2	2020-05-26 13:54:24	2
西南	成都	2020-05-31	DD002	箱装	蛋糕纸	2	2020-06-03 06:49:22	2
华东	上海	2020-06-30	DD003	箱装	苹果醋	2	2020-07-03 00:03:21	2
	宁波	2020-07-28	DD004	散装	钢化膜	1	2020-08-01 07:18:57	1
	上海	2020-08-02	DD005	桶装	油漆	2	2020-08-02 09:07:18	2

图 3-18　指定索引列(2)

3.6.2　选择列

采用字符串方式,字符串对应的是 Excel 工作表中的列字母与列范围(例如'A:D'或'A,C:D')。当默认值为 None 时解析工作表中所有的列,代码如下:

```
#ch03d005.py
import pandas as pd
df1 = pd.read_excel(
    r'D:\数据源\B文件\华东.xlsx',
    usecols = 'A:D',   #采用字符串方式
)
df1
```

或采用 list-like 方式，列表内容全为文本，代码如下：

```
df2 = pd.read_excel(
    r'D:\数据源\B文件\华东.xlsx',
    usecols = ['片区','订单来源','接单时间','订单编号',]    #可读性最好
)
df2
```

或者，列表内容全为整数。在列表中，最后一个逗号有无不影响输出，代码如下：

```
df3 = pd.read_excel(
    r'D:\数据源\B文件\华东.xlsx',
    usecols = [0,1,2,3]                #不能采用[:4]或[0:4]切片方式
)
df3
```

或事先预设的列表的字段顺序，代码如下：

```
ucl = ['片区','订单来源','接单时间','订单编号',]
df4 = pd.read_excel(
    r'D:\数据源\B文件\华东.xlsx',
    usecols = ucl
)
df4
```

或事先预设元组，代码如下：

```
tpl = ('片区','订单来源','接单时间','订单编号',)
df5 = pd.read_excel(
   r'D:\数据源\B文件\华东.xlsx',
   usecols = tpl
)
df5
```

以上代码返回的值如图 3-19 所示。

当 usecols 参数为 list-like 时，不可以采用整数与字符串混合的方式(例如[0,1,2,'订单编号'])。当采用 cols=[0,1,2,'订单编号']时，会出现的错误提示如下：

```
ValueError: 'usecols' must either be list-like of all
strings, all unicode, all integers or a callable.
```

	片区	订单来源	接单时间	订单编号
0	华东	上海	2020-06-30	DD003
1	华东	上海	2020-08-02	DD005
2	华东	上海	2021-10-01	DD028

图 3-19　选择列(1)

当采用字符串方式时，字符串可以来自不连续的区域，代码如下：

```
df6 = pd.read_excel(
       r'D:\数据源\B文件\华东.xlsx',
       usecols = 'A,C:D',
)
df6
```

返回的值如图 3-20 所示。

usecols='A, C: D'

	A	B	C	D
1	片区	订单来源	接单时间	订单编号
2	华东	上海	2020-06-30 00:00:00	DD003
3	华东	上海	2020-08-02 00:00:00	DD005
4	华东	上海	2021-10-01 00:00:00	DD028

(a) 数据源

	片区	接单时间	订单编号
0	华东	2020-06-30	DD003
1	华东	2020-08-02	DD005
2	华东	2021-10-01	DD028

(b) usecols='A, C: D'

图 3-20　选择列(2)

当 usecols 采用的是数值列表时，字符串可以来自不连续的区域，代码如下：

```
df7 = pd.read_excel(
   r'D:\数据源\B文件\华东.xlsx',
   usecols = range(1,8,2),
)
df7
```

返回的值如图 3-21 所示。

usecols=range(1, 8, 2)

	A	B	C	D	E	F	G	H	I
1	片区	订单来源	接单时间	订单编号	包装方式	产品	订单数	入库日期	入库数
2	华东	上海	2020-06-30 00:00:00	DD003	箱装	苹果醋	2	2020-07-03 00:03:21	2
3	华东	上海	2020-08-02 00:00:00	DD005	桶装	油漆	2	2020-08-02 09:07:18	2
4	华东	上海	2021-10-01 00:00:00	DD028	散装	钢化膜	4	2021-10-03 07:24:21	4

(a) 数据源

	订单来源	订单编号	产品	入库日期
0	上海	DD003	苹果醋	2020-07-03 00:03:21
1	上海	DD005	油漆	2020-08-02 09:07:18
2	上海	DD028	钢化膜	2021-10-03 07:24:21

(b) usecols=range(1, 8, 2)

图 3-21　选择列(3)

当 usecols 采用的是匿名函数时，代码如下：

```
df8 = pd.read_excel(
   r'D:\数据源\B文件\华东.xlsx',
   usecols = lambda x:～x.find('单')
)
df8
```

	订单来源	接单时间	订单编号	订单数
0	上海	2020-06-30	DD003	2
1	上海	2020-08-02	DD005	2
2	上海	2021-10-01	DD028	4

图 3-22　选择列(4)

返回的值如图 3-22 所示。

3.7　数据类型

在 Pandas 中，数据类型有 number、float、int、object、datetime、timedelta、object、category 等数据类型，也可以在代码中使用 Python 或 NumPy 等效的数据类型，或者对 float、int 数据类型进行 8 位、16 位、32 位、64 位、128 位细分指定。

3.7.1　查看数据类型

利用 Pandas 的 dtypes 属性对 DataFrame 进行属性查看，代码如下：

```
#ch03d006.py
import pandas as pd
```

```
df = pd.read_excel(
   r'D:\数据源\B文件\华东.xlsx',
   usecols = [2,3,5,6,8]
)
df.dtypes
```

返回的值如下：

```
接单时间    datetime64[ns]
订单编号            object
产品              object
订单数              int64
入库数              int64
dtype: object
```

在 Pandas 中，int 与 float 数据类型默认为 64 位。

pd.read_excel()函数可采用字典映射方式对读取的数据类型进行指定或调整，代码如下：

```
type = {"接单时间":'datetime64[ns]',"订单数":'int8','入库数':'float'}
df1 = pd.read_excel(
   r'D:\数据源\B文件\华东.xlsx',
   usecols = [2,3,5,6,8],
   dtype = type
)
df1
```

返回的值如图 3-23 所示。

	接单时间	订单编号	产品	订单数	入库数
0	2020-06-30	DD003	苹果醋	2	2.0
1	2020-08-02	DD005	油漆	2	2.0
2	2021-10-01	DD028	钢化膜	4	4.0

图 3-23　调整数据类型后的值

利用 dtypes 属性，查看 DataFrame 的数据类型，代码如下：

```
df1.dtypes
```

返回的值如下：

```
接单时间    datetime64[ns]
订单编号            object
产品              object
订单数               int8
入库数             float64
dtype: object
```

入库数的数据类型由 int64 调整为 float64。

如果对 DataFrame 中 Series 的数据类型进行调整，则可以采用 astype()方法，代码如下：

```
df1['入库数'] = df1['入库数'].astype(int)
df1.dtypes
```

返回的值如下：

```
接单时间    datetime64[ns]
订单编号            object
产品              object
订单数              int8
入库数             int32
dtype: object
```

入库数的数据类型由 float64 调整为 int32。

3.7.2 转换数据类型

pd.read_excel()函数 converters 参数的默认值为 None。采用的是字典映射方式对数据类型进行指定或对数据进行修改，字典映射方式为 dict{str，Callable}或 dict{int，Callable}，代码如下：

```
#ch03d007.py
import pandas as pd
df = pd.read_excel(
   r'D:\数据源\B文件\华东.xlsx',
   usecols =[2,3,5,6,8],
   converters = {
       "接单时间":str,
       "订单编号":lambda x:x.replace("DD","A"),
       "订单数":lambda x:x+2,
       "入库数":lambda x:float(x*3)
   }
)
df
```

或者

```
df1 = pd.read_excel(
   r'D:\数据源\B文件\华东.xlsx',
   usecols =[2,3,5,6,8],
   converters = {
       0:str,
       1:lambda x:x.replace("DD","A"),
       3:lambda x:x+2,
       4:lambda x:float(x*3)
   }
)
df1
```

返回的值如图 3-24 所示。

	A	B	C	D	E	F	G	H	I
1	片区	订单来源	接单时间	订单编号	包装方式	产品	订单数	入库日期	入库数
2	华东	上海	2020-06-30 00:00:00	DD003	箱装	苹果醋	2	2020-07-03 00:03:21	2
3	华东	上海	2020-08-02 00:00:00	DD005	桶装	油漆	2	2020-08-02 09:07:18	2
4	华东	上海	2021-10-01 00:00:00	DD028	散装	钢化膜	4	2021-10-03 07:24:21	4

(a) 数据源

	接单时间	订单编号	产品	订单数	入库数
0	2020-06-30 00:00:00	A003	苹果醋	4	6.0
1	2020-08-02 00:00:00	A005	油漆	4	6.0
2	2021-10-01 00:00:00	A028	钢化膜	6	12.0

(b) converters

图 3-24 转换数据类型

利用 DataFrame 的 dtypes 属性,查看数据类型,代码如下:

```
df1.dtypes
```

返回的值如下:

```
接单时间      object
订单编号      object
产品        object
订单数        int64
入库数      float64
dtype: object
```

接单时间的数据类型由 datetime 被调整为 object;入库数的数据类型由 int 被调整为 float。

3.7.3 解析日期时间列

parse_dates 参数用于解析日期时间列,默认值为 False。可采用 parse_dates=True、parse_dates=[列名]、parse_dates=[列位置]或 parse_dates=[[]]的索引方式。①parse_dates=True 用于尝试将索引列的文本日期时间解析为日期时间;②当索引列不是日期时间时,可采用 parse_dates=['接单时间', '入库日期']或 parse_dates=[2,7]进行解析;③当年、月、日、时、分、秒等数值型数据分别存放于各列时,可采用类似 parse_dates=[['年', '月', '日']]的二维列表形式将其合并成一列,各列的列名或位置索引均存放于内层列表中。如果需要规范拼接后的列名,则可以采用字典形式,类似于 parse_dates={'日期':['年', '月', '日']},拼接后的列名为日期。

date_parse 参数用于指定解析日期时间的格式。在一般情况下,当使用了 parse_dates 参数时,会自动调用 date_parse 参数对各类文本型日期进行格式解析。

3.8 对 Excel 的读取与解析

ExcelFile()是 Pandas 中对 Excel 表格文件进行读取操作的类,用于将 Excel 文件读取到 Pandas DataFrame 中,特别适用于含有多个 Sheet 的 Excel,支持 Excel 早期版本的 xls 格式。ExcelFile()与 WriteExcel()经常搭配使用,实现对多个 Sheet 的读写,非常方便与快捷,语法如下:

```
ExcelFile(io,storage_options = storage_options,engine = engine).parse(sheet_name = 0, header = 0,
names = None, index_col = None, usecols = None, squeeze = False, converters = None, true_values =
None, false_values = None, skiprows = None, nrows = None, na_values = None, parse_dates = False,
date_parser = None, thousands = None, comment = None, skipfooter = 0, convert_float = True,
mangle_dupe_cols = True, **kwds)
```

当 sheet_name=None 时,获取的是工作簿中所有的工作表,代码如下:

```
#ch03d008.py
```

```
import pandas as pd
df = pd.ExcelFile(r'D:\数据源\B 文件\华东.xlsx').parse()
df
```

返回的值如图 3-25 所示。

	片区	订单来源	接单时间	订单编号	包装方式	产品	订单数	入库日期	入库数
0	华东	上海	2020-06-30	DD003	箱装	苹果醋	2	2020-07-03 00:03:21	2
1	华东	上海	2020-08-02	DD005	桶装	油漆	2	2020-08-02 09:07:18	2
2	华东	上海	2021-10-01	DD028	散装	钢化膜	4	2021-10-03 07:24:21	4

图 3-25　获取工作簿中的第 1 个工作表

采用 parse()方法解析路径文件。以 sheet_name＝None 为例，代码如下：

```
df1 = pd.ExcelFile(r'D:\数据源\B 文件\华东.xlsx').parse(sheet_name = None)
dict(df1)
```

返回的值为 dict。采用列表推导式了解工作簿中所有的工作表名，代码如下：

```
[i for i in df1]
```

返回的值如下：

```
['上海', '南京', '厦门', '合肥', '宁波', '常州', '无锡', '杭州', '苏州']
```

从中不难发现，以下两种写法返回的值相同。

```
#方法一
pd.read_excel(r'D:\数据源\B 文件\华东.xlsx',sheet_name = None)
#方法二
pd.ExcelFile(r'D:\数据源\B 文件\华东.xlsx').parse(sheet_name = None)
```

因此，当 pd.read_excel()函数中与 pd.ExcelFile.parse()函数中存在相同的参数，其参数的用法基本是相同的。

3.9 存储数据

在日常使用过程中，常常会将 pd.read_excel()函数与 pd.to_excel()函数搭配使用，还会将 pd.ExcelFile()函数与 pd.ExcelWriter()函数搭配使用。df.to_excel()函数仅支持将单个 Sheet 表格写入 Excel，而 pd.ExcelWriter()函数支持将多个 Sheet 表格写入 Excel。

3.9.1 to_excel

在 Pandas 中，可以通过 df.to_csv()、df.to_excel()、pd.ExcelWriter()、df.to_json()、df.to_sql()等众多数据的写入方式。df.to_excel()方法用于将 DataFrame 写入 Excel 工作簿。如果只需将单个 DataFrame 写入 Excel 工作簿文件，则需要指定 Excel 文件名；如果

需要在 Excel 中写入多个工作表，则需要先创建 pd. ExcelWriter 对象，并在文件中指定要写入的工作表，语法如下：

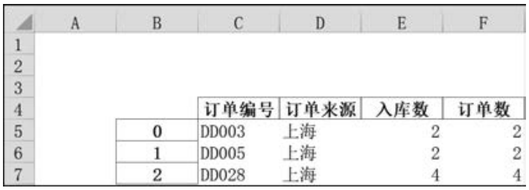
```
DataFrame.to_excel(excel_writer, sheet_name = 'Sheet1', na_rep = '', float_format = None,
columns = None, header = True, index = True, index_label = None, startrow = 0, startcol = 0,
engine = None, merge_cells = True, encoding = None, inf_rep = 'inf', verbose = True, freeze_panes =
None, storage_options = None)
```

应用举例，指定列、空值显示的值、行与列的偏移位置等，代码如下：

```
#ch03d009.py
import pandas as pd
#获取数据
df1 = pd.read_excel(r'D:\数据源\B 文件\华东.xlsx')

#存储数据
df1.to_excel(r'D:\数据源\第 3 章\ch0309\(华东)上海.xlsx',
          na_rep = '-- ',
          columns = ['订单编号', '订单来源', '入库数', '订单数'],
          startrow = 3,
          startcol = 1)
```

文件的存储路径为 D:\数据源\第 3 章\写入\(华东)上海. xlsx。返回的值如图 3-26 所示。



	A	B	C	D	E	F
1						
2						
3						
4			订单编号	订单来源	入库数	订单数
5		0	DD003	上海	2	2
6		1	DD005	上海	2	2
7		2	DD028	上海	4	4

图 3-26　将 DataFrame 写入 Excel 工作簿

3. 9. 2　ExcelWriter

pandas. ExcelWriter()函数用于在同一工作簿中保存多个工作表。应用举例，将华东. xlsx 工作簿中的苏州、杭州工作表的数据存储到苏杭. xlsx 文件中对应的工作表，代码如下：

```
df2 = pd.read_excel(r'D:\数据源\B 文件\华东.xlsx',sheet_name = '苏州')
df3 = pd.read_excel(r'D:\数据源\B 文件\华东.xlsx',sheet_name = '杭州')
with pd.ExcelWriter(r'D:\数据源\第 3 章\ch0309\苏杭.xlsx') as wt:
   df2.to_excel(wt, sheet_name = '苏州'),
   df3.to_excel(wt, sheet_name = '杭州')
```

文件被保存在苏杭. xlsx 工作簿中，该工作簿中有苏州、杭州两个工作表。在以上代码中 with 语句用于将文件的写操作包含在一个代码内，当执行完成后就会自动关闭文件对象，无须再调用 close()方法。

应用举例，批量读取工作簿(华东. xlsx)中所有工作表的数据，并批量提取列数据，代码

如下：

```
dfs = pd.read_excel(
   r'D:\数据源\B文件\华东.xlsx',
   sheet_name = None,
   usecols = ['片区','订单来源','接单时间','订单编号'],
   )

with pd.ExcelWriter(r'D:\数据源\第3章\ch0309\华东A.xlsx') as wt:
   for s, df in dfs.items():
       df.to_excel(wt,sheet_name = s,index = False)
```

或者可采用以下代码：

```
dfs = pd.read_excel(r'D:\数据源\B文件\华东.xlsx',sheet_name = None)

with pd.ExcelWriter(r'D:\数据源\第3章\ch0309\华东B.xlsx') as wt:
   for s, df in dfs.items():
       cols = df[['片区','订单来源','接单时间','订单编号']]
       cols.to_excel(wt,sheet_name = s,index = False)
```

运行以上代码,返回的值如图3-27所示。

图3-27　将DataFrame批量写入Excel工作簿