07-pandas数据加载、存储文件

预习重点

> - 如何读取csv格式数据
> - 如何读取excel格式数据
> - 问题： 遇到大文件，直接读取容易卡死，怎么办?

常见文件读取函数：

函数	说明
read_csv	从文件、URL、文件型对象中加载带分隔符的数据。默认分隔符为逗号
read table	从文件、URL、文件型对象中加载带分隔符的数据。默认分隔符为制表符(t)
read fwf	读取定宽列格式数据(也就是说，没有分隔符)

read_clipboard	读取剪贴板中的数据，可以看做read_table的剪贴板版。再将网页转换为表格时很有用
read excel	从Excel XLS或XLSX file读取表格数据
read hdf	读取pandas写的HDF5文件
read html	读取HTML文档中的所有表格
read_json	读取JSON(JavaScript Object Notation)字符串中的数据
read_msgpack	二进制格式编码的pandas数据
read pickle	读取Pythonpickle格式中存储的任意对象
read_sas	读取存储于SAS系统自定义存储格式的SAS数据集
read_sql	(使用SQLAlchemy)读取SQL查询结果为pandas的DataFrame
read_stata	读取Stata文件格式的数据集
read_feather	读取Feather二进制文件格式

1 文件读取之read_csv()

read_csv() 是 Pandas 库中的函数，用于从 CSV 文件中读取数据并创建一个 DataFrame。以下是一些常用参数的说明：

1. **filepath_or_buffer**：这是必选参数，指定要读取的 CSV 文件的路径或 URL。可以是一个字符串，也可以是类似文件对象的可迭代对象（例如文件句柄）。
2. **sep**：指定 CSV 文件中的字段分隔符。默认值是逗号（**,**），但您可以根据文件的实际分隔符来设置这个参数，例如制表符（**\t**）或分号（**;**）。
3. **delimiter**：**sep** 的别名，用于指定字段分隔符。
4. **header**：指定用作列名的行号。通常为整数，如果没有列名行，可以设置为 **None**。默认值为 0，表示第一行包含列名。
5. **names**：如果没有列名行，您可以通过此参数提供列名的列表。与 **header=None** 一起使用。
6. **index_col**：用于指定哪一列作为 DataFrame 的索引列。可以是列名、列号或多列的组合。
7. **usecols**：指定要读取的列，可以是列名的列表或列号的列表。用于仅读取感兴趣的列，而不是整个文件。
8. **skiprows**：指定要跳过的行数，通常用于跳过文件中的标题或注释行。
9. **nrows**：指定要读取的行数，可以用于限制读取的数据量。
10. **dtype**：用于指定列的数据类型的字典，可以帮助优化内存使用和数据类型转换。
11. **encoding**：指定 CSV 文件的字符编码，常见的编码包括 "utf-8"、"ISO-8859-1" 等。
12. **parse_dates**：指定要解析为日期时间的列，可以是列名或列号的列表。
13. **date_parser**：自定义日期时间解析函数，用于将字符串解析为日期时间对象。
14. **na_values**：指定要视为缺失值的标记值列表，例如 "NA"、"N/A"、"null"

等。

15. **skipinitialspace**：指定是否跳过字段值前的空白字符，默认为 False。
16. **skip_blank_lines**：指定是否跳过空白行，默认为 True。
17. **quotechar**：指定用于引用字段值的字符，通常为双引号或单引号。
18. **compression**：指定文件的压缩格式，例如 "gzip"、"bz2"、"xz" 等。
19. **thousands**：用于指定千位分隔符的字符。

1) 文件路径

```python
pd.read_csv(r'D:\exa5.csv',encoding='ANSI')
```

2)分隔符 sep

```python
pd.read_csv(path,sep='\t',encoding='ANSI') # 不能分隔
```

4) index_col(索引列)

```python
# 指定第0列为索引列
pd.read_csv(path,index_col=0,encoding='ANSI')
```

5) usecols(使用部分列的数据)

```python
pd.read_csv(path,usecols=['股票1','股票2'],encoding='ANSI')
```

6) dtype(数据类型)

```python
# 读取时指定元素类型
pd.read_csv(path,dtype=np.float64,encoding='ANSI')
```

7) parse_dates

```python
# 将日期列转成日期类型
data7 = pd.read_csv(r"E:\exa5.csv",parse_dates=['日期'],encoding='ANSI')
data7.info()
```

2 读取之read_table()

参数同read_csv()， **read_csv()** 和 **read_table()** 之间的区别主要是函数名称，默认分隔符的历史设置。在当前版本的 Pandas 中，它们几乎可以互相替代，只需根据实际数据文件的分隔符来设置 **sep** 参数即可。

read_table()读取csv

```python
pd.read_table(r"D:\股票数据.csv",sep=',')
```

3 read_excel()

```python
data1  = pd.read_excel(r'E:\exa10.xlsx',sheet_name='股票')
```

4 json 读取

```python
t1 = time.time()
jsond = pd.read_json('D:/xiyun/股票数据.json')
t2 = time.time()
print(t2-t1)
jsond
```

网页的json格式数据，test.json

```python
[
    {
    "id": "A001",
    "name": "兮云",
    "url": "www.baidu.com",
    "likes": "吃"
    },
    {
    "id": "A002",
    "name": "Google",
    "url": "www.google.com",
    "likes": "睡"
    },
    {
    "id": "A003",
    "name": "淘宝",
    "url": "www.taobao.com",
    "likes": "玩"
    }
]
```

实战

```python
import pandas as pd
# 读取json格式，会自动转换为DataFrame格式
df = pd.read_json('test.json')
print(type(df))

# 转换成字符串
dd = df.to_string()
print(type(dd))    # <class 'str'>
```

5 数据保存

保存csv

```python
df.to_csv('股票数据1.csv',header=True,index=True,mode='w')
```

保存excel

```python
df.to_excel('股票数据.xlsx',sheet_name='股票涨跌幅百分比',index=True,header=True)
```

保存json

```python
df.to_json('股票数据4.json')
```