



用 Python 做统计分析

作者：陈震

组织：西南大学

时间：September 18, 2019

版本：1.00



Victory won't come to us unless we go to it. — M. Moore

目 录

1	Python 中的 Numpy 与 Pandas	1
1.1	Numpy 的常用操作	1
1.2	Pandas 的常用操作	2
1.2.1	创建、读取和存储数据	2
1.2.2	查看数据	5
1.2.3	增加、删除、修改、查询数据	7
1.2.4	数据合并	7
1.2.5	数据排序	7
1.2.6	分组汇总数据	7
2	数据的概括性度量	8
2.1	均值、方差	8
2.2	众数、中位数、分位数	8
2.3	偏态、峰态	8
3	参数估计与假设检验	9
4	方差分析	10
5	线性回归	11
6	非线性回归	12
7	主成分分析	13
8	因子分析	14
9	时间序列分析和预测	15

第 1 章 Python 中的 Numpy 与 Pandas



1.1 Numpy 的常用操作

1.2 Pandas 的常用操作

Pandas 是 Python 做统计分析时最重要的数据分析工具之一，它基于 Numpy 开发，提供了许多处理大型数据集所需的函数，可以灵活高效的处理各种数据集。

Pandas 一般使用两种数据类型：DataFrame 和 Series。其中 DataFrame 是二维数据，而 Series 是一维数据。可以这样简单理解：DataFrame 相当于 Excel 里面的一张表，而 Series 是表中的某一列。还有一个表示三维数据的类型 Panel，但我们经常使用的是前面两种数据类型。使用 Pandas 时首先到导入 pandas 包。

```
import pandas as pd
```

1.2.1 创建、读取和存储数据

1.2.1.1 创建

例如有下面的数据：

	统计学	高数	英语
张三	85	82	84
李四	68	63	90
王五	90	88	78

我们可以使用下面的代码读取到一个 DataFrame 里面：

```
df = pd.DataFrame({'统计学': [85, 68, 90], '高数': [82, 63, 88], '英语': [84, 90, 78]}, index=['张三', '李四', '王五'])
print(df)
```

从上面可以看出，DataFrame 通过一个字典类型设置各列的标题及对应数值，通过一个 index 数组设置行标题。还有一种方式是通过 numpy 的 array 数组设置数值，然后通过 columns 数组设置列标题，通过 index 数组设置行标题¹：

```
import numpy as np

df = pd.DataFrame(np.array([[85, 68, 90], [82, 63, 88], [84, 90, 78]]), columns=['统计学', '高数', '英语'], index=['张三', '李四', '王五'])
print(df)
```

一个 DataFrame 包括 columns（列标题），index（行标题）与 values（数值）三个部分，上述例子中的 columns，index，values 如下图所示：

¹如果创建时不设置行标题或列标题，系统会自动生成从 0 到行数或列数的数组作为标题

这一行是 columns

	统计学	高数
张三	84	82
李四	90	63
王五	78	88

这一列是 index

中间的是 values

1.2.1.2 读取

大部分情况下，我们要读取 Excel 里面的数据，假设上面的例子在 excel 文件‘成绩数据.xlsx’并放在电脑硬盘某个位置：

可以通过下面的代码读取文件：

```
df = pd.read_excel(r'D:\Users\chen_git\Statistics-book\datas\成绩数据.xlsx',  
                  index_col=0)
```

数据文件的地址位于‘D:\Users\chen_git\Statistics-book\datas\.xlsx’，使用 read_excel 时在文件位置字符串前面加上字母 r，这样 python 就能找到我们的数据文件了。index_col=0 表示行标题位于第一列（不然 read_excel 会把第一列内容读取到 values 里面，并默认 index 为 0, 1, 2, ...）。read_excel 会自动将 Excel 第一行的内容作为行标题。read_excel 的一般语法为：

read_excel 函数的语法

read_excel(io, sheetname=0, header=0, skiprows=None, index_col=None)

io	数据文件的地址与名字，一般为字符串
sheetname	工作簿名字，默认为 0，表示读取第一张工作簿
header	作为列名的行，默认为 0，即取第一行的值为列名
skiprows	省略指定行数的数据，从第一行开始查起
index_col	行标题所在的列

更多的语法设置可以查看官网文档：

https://pandas.pydata.org/pandas-docs/version/0.20/generated/pandas.read_excel.html

还有一种常见的数据文件类型为 csv，我们只需使用 pandas 中的函数 read_csv，它的语法与 read_excel 基本一样。但是，read_csv 可以读取网络数据库中的 csv 文件，例如下面的例子显示世界上所有国家与所属的大洲。

```
df = pd.read_csv('https://raw.githubusercontent.com/cs109/2014_data/master/countries.csv')
print(df)
```

	Country	Region
0	Algeria	AFRICA
1	Angola	AFRICA
2	Benin	AFRICA
3	Botswana	AFRICA
4	Burkina	AFRICA
..
189	Paraguay	SOUTH AMERICA
190	Peru	SOUTH AMERICA
191	Suriname	SOUTH AMERICA
192	Uruguay	SOUTH AMERICA
193	Venezuela	SOUTH AMERICA

[194 rows x 2 columns]

1.2.1.3 存储

存储时使用函数 to_excel 或 to_csv。例如我们将 DataFrame 存储到 E:\datas 文件夹里，并命名为 marks.xlsx：

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.array([[85, 68, 90], [82, 63, 88], [84, 90, 78]]), columns
                  =['统计学', '高数', '英语'], index=['张三', '李四', '王五'])
df.to_excel(r'E:\datas\marks.xlsx')
```


在上面的代码中，若直接写成 `df.to_excel('marks.xlsx')`，则文件存储的路径为当前工作环境所在的文件夹。

1.2.2 查看数据

1.2.2.1 概览数据

快速查看 `DataFrame` 各列数据的统计信息可以使用 `describe()` 函数，包括各列数据的非空数值数目、均值、标准差、最大值、最小值、分位数。例如上面的例子：

```
df.describe()
```

	统计学	高数	英语
count	3.000000	3.000000	3.000000
mean	83.666667	73.666667	85.333333
std	1.527525	14.364308	6.429101
min	82.000000	63.000000	78.000000
25%	83.000000	65.500000	83.000000
50%	84.000000	68.000000	88.000000
75%	84.500000	79.000000	89.000000
max	85.000000	90.000000	90.000000

`info()` 函数可以查看各列数据的类型：

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, 张三 to 王五
Data columns (total 3 columns):
统计学      3 non-null int32
高数        3 non-null int32
英语        3 non-null int32
dtypes: int32(3)
memory usage: 60.0+ bytes
None
```

`DataFrame` 的 `sample()` 函数可以从数据中随机抽取样本，小括号中用数字表示抽取的样本个数。例如，下面的代码从 `df` 里面随机抽取 2 个样本：

```
df.sample(2)
```

	统计学	高数	英语
王五	84	90	78
张三	85	68	90

另外，`head()` 函数可以显示 `DataFrame` 前五行数据，而 `tail()` 函数可以显示 `DataFrame` 最后五行数据。

1.2.2.2 查看单列、单行、单元格数据

查看某一列数据时，最简单的方式是中括号里面输入列名的方式，例如查看英语成绩那一列数据：

```
df['英语']
```

```
张三    90
李四    88
王五    78
Name: 英语, dtype: int32
```

查看某一行数据时，最简单的方式是中括号里面输入行名方式，例如查看张三哪一行的数据：

```
df['张三']
```

也可以用中括号里面跟着行索引的方式（行数：行数 + 1），即 `df[0:1]`，与上面代码显示效果一样。

```
      统计学  高数  英语
张三    85   68   90
```

若查看某个单元格，比较方便的方式是用两个中括号，每个中括号内分别跟着行名和列名，例如查看张三的高数成绩：

```
df['张三']['高数'] # 显示张三的高数成绩为 68
```

1.2.2.3 查看多列、多行数据 `iloc`

查看多行、多列数据时，一般用 `iloc` 比较方便，而且 `iloc` 不仅能查看多行多列数据，也能查看单行、单列或某个单元格数据：

例如，查看行：

```
df.iloc[1]          # 查看第 2 行数据
df.iloc[0:2]        # 查看前 2 行数据
df.iloc[[0, 2]]     # 查看第 1 行与第 3 行数据
```

查看列：

```
df.iloc[:, 1]       # 查看第 2 列数据
df.iloc[:, 0:2]     # 查看前 2 列数据
df.iloc[:, [0, 2]]  # 查看第 1 列与第 3 列数据
```

查看一块数据：


```
df.iloc[0:2, 0:2]      # 查看前 2 行, 前 2 列的一块数据  
df.iloc[[0, 2], [0, 2]] # 查看第 1、第 3 行, 第 1、第 3 列的一块数据  
df.iloc[0:2, [0, 2]]   # 查看前 3 行, 第 1、第 3 列的一块数据  
df.iloc[[0, 2], 0:2]   # 第 1、第 3 行, 前 2 列的一块数据
```

查看某个单元格:

```
df.iloc[1, 1] # 查看第 2 行, 第 2 列的单元格数据
```

1.2.3 增加、删除、修改、查询数据

1.2.4 数据合并

1.2.5 数据排序

1.2.6 分组汇总数据

第 2 章 数据的概括性度量



2.1 均值、方差

2.2 众数、中位数、分位数

2.3 偏态、峰态

第 3 章 参数估计与假设检验



第 4 章 方差分析



第 5 章 线性回归



第 6 章 非线性回归



第 7 章 主成分分析



第 8 章 因子分析



第 9 章 时间序列分析和预测

