数据分析 1

数据分析

定义：

数据分析是基于商业目的，有目的的进行收集、整理、加工和分析数据，提炼有价信息的一个过程。

过程：

明确分析目的与框架、数据收集、数据处理（数据清洗、数据转换）、数据分析、数据展现和撰写报告等6个阶段。

数据类型：

- 表格型数据，其中各列可能是不同的类型（字符串、数值、日期等）。比如保存在关系型数据库中或以制表符/逗号为分隔符的文本文件中的那些数据。
- 多维数组（矩阵）。
- 通过关键列（对于SQL用户而言，就是主键和外键）相互联系的多个表。
- 间隔平均或不平均的时间序列。

Excel是最广泛的数据分析工具

为什么用Python进行数据分析

- 拥有巨大活跃的科学计算社区
- 数据科学、机器学习、学界和工业界开发重要语言
- 胶水语言，轻松集成旧有算法和系统
- 不仅适用于研究和原型构建，同时也适用于构建生产系统

重要的Python库

1、NumPy（Numerical Python）

Python科学计算的基础包

- 快速高效的多维数组对象ndarray。
- 用于对数组执行元素级计算以及直接对数组执行数学运算的函数。
- 用于读写硬盘上基于数组的数据集的工具。
- 线性代数运算、傅里叶变换，以及随机数生成。
- 成熟的C API， 用于Python插件和原生C、C++、Fortran代码访问NumPy的数据结构和计算工具。
- 对于数值型数据，NumPy数组在存储和处理数据时要比内置的Python数据结构高效得多。

2、pandas

- pandas提供了快速便捷处理结构化数据的大量数据结构和函数。
- pandas兼具NumPy高性能的数组计算功能以及电子表格和关系型数据库（如SQL）灵活的数据处理功能。它提供了复杂精细的索引功能，能更加便捷地完成重塑、切片和切块、聚合以及选取数据子集等操作
- 数据操作、准备、清洗是数据分析最重要的技能（耗时最长）

3、matplotlib

- 最流行的用于绘制图表和其它二维数据可视化的Python库
- 适合创建出版物上用的图表

4、IPython 和 Jupyter

- 执行 ➜ 探索 工作流 （探索、试错、重复）
- IPython web notebook ➜ Jupyter notebook (支持40多种编程语言)
  - Jupyter notebook支持markdown和html

5、Scipy

- 一组专门解决科学计算中各种标准问题域的包的集合。

6、scikit-learn

- scikit-learn成为了Python的通用机器学习工具包

7、statsmodels

- statsmodels包含经典统计学和经济计量学的算法

- 常用模块引用惯例

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels as sm
```

安装Anaconda

Downloads - Anaconda

Python 1991年出现，Python 2.x在2020年就会到期（包括重要的安全补丁），新项目请使用Python 3

IPython基础

```
$ ipython
In [1]: a = 5
In [2]: a
Out[2]: 5
```

pretty printed（Python对象被格式化为更易读的形式）

```
In [5]: import numpy as np
In [6]: data = {i : np.random.randn() for i in range(7)}
In [7]: data
Out[7]:
{0: -0.20470765948471295,
 1: 0.47894333805754824,
 2: -0.5194387150567381,
 3: -0.55573030434749,
 4: 1.9657805725027142,
 5: 1.3934058329729904,
6: 0.09290787674371767}
```

运行Jupyter Notebook

notebook是Jupyter项目的重要组件之一，它是一个代码、文本（有标记或无标记）、数据可视化或其它输出的交互式文档。

Python的Jupyter内核是使用IPython。

启动Jupyter

```
$ jupyter notebook
```

Jupyter Notebook

要新建一个notebook，点击按钮New，选择“Python3”或“conda[默认项]”。如果是第一次，点击空格，输入一行Python代码。然后按Shift-Enter执行。

当保存notebook时（File目录下的Save and Checkpoint），会创建一个后缀名为**.ipynb**的文件。

要加载存在的notebook，把它放到启动notebook进程的相同目录内。

```
%pwd
%ls
```

Tab补全

```
In [3]: b = [1, 2, 3]

In [4]: b.<Tab>
b.append  b.count   b.insert  b.reverse
b.clear   b.extend  b.pop     b.sort
b.copy    b.index   b.remove
```

```
In [1]: import datetime

In [2]: datetime.<Tab>
datetime.date           datetime.MAXYEAR       datetime.timedelta
datetime.datetime       datetime.MINYEAR       datetime.timezone
datetime.datetime_CAPI  datetime.time          datetime.tzinfo
```

在变量前后使用问号？，可以显示对象的信息

```
In [8]: b = [1, 2, 3]

In [9]: b?
Type:       list
String Form:[1, 2, 3]
Length:     3
Docstring:
list() -> new empty list
list(iterable) -> new list initialized from iterable's items

In [10]: print?
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

```
def add_numbers(a, b):
    """
    Add two numbers together

    Returns
    -------
    the_sum : type of arguments
    """
    return a + b
```

然后使用?符号，就可以显示如下的文档字符串：

```
```

```
In [11]: add_numbers?
Signature: add_numbers(a, b)
Docstring:
Add two numbers together

Returns
-------
the_sum : type of arguments
File:      <ipython-input-9-6a548a216e27>
Type:      function
```

使用??会显示函数的源码：

```
In [12]: add_numbers??
Signature: add_numbers(a, b)
Source:
def add_numbers(a, b):
    """
    Add two numbers together

    Returns
    -------
    the_sum : type of arguments
    """
    return a + b
File:      <ipython-input-9-6a548a216e27>
Type:      function
```

搜索IPython的命名空间

```
In [13]: np.*load*?
np.__loader__
np.load
np.loads
np.loadtxt
np.pkgload
```

%run，%load，%paste，%cpaste 命令

%run命令运行所有的Python程序

```
In [14]: %run test3.py
```

文件中所有定义的变量（import、函数和全局变量，除非抛出异常），都可以在IPython shell中随后访问

在Jupyter notebook中，你也可以使用%load，它将脚本导入到一个代码格中

```
%load test3.py
```

%paste和%cpaste 函数。%paste 可以直接运行剪贴板中的代码

键盘快捷键

快捷键	说明
Ctrl-P 或↑箭头	用当前输入的文本搜索之前的命令
Ctrl-N 或↓箭头	用当前输入的文本搜索之后的命令
Ctrl-R	Readline 方式翻转历史搜索（部分匹配）
Ctrl-Shift-V	从剪贴板粘贴文本
Ctrl-C	中断运行的代码
Ctrl-A	将光标移动到一行的开头
Ctrl-E	将光标移动到一行的末尾
Ctrl-K	删除光标到行尾的文本
Ctrl-U	删除当前行的所有文本
Ctrl-F	光标向后移动一个字符
Ctrl-B	光标向前移动一个字符
Ctrl-L	清空屏幕

魔术命令

%timeit 测量任何Python语句，例如矩阵乘法，的执行时间

```
In [23]: foo = %pwd
In [24]: foo
```

命令	说明
%quickref	显示 IPython 的快速参考。
%magic	显示所有魔术命令的详细文档。
%debug	在出现异常的语句进入调试模式。
%hist	打印命令的输入（可以选择输出）历史。
%pdb	出现异常时自动进入调试。
%paste	执行剪贴板中的代码。
%cpaste	开启特别提示，手动粘贴待执行代码。
%reset	删除所有命名空间中的变量和名字。
%page OBJECT	美化打印对象，分页显示。
%run script.py	运行代码。
%prun statement	用 CProfile 运行代码，并报告分析器输出。
%time statement	报告单条语句的执行时间。
%timeit statement	多次运行一条语句，计算平均执行时间。适合执行时间短的代码。
%who, %who_ls, %whos	显示命名空间中的变量，三者显示的信息级别不同。
%xdel variable	删除一个变量，并清空任何对它的引用。

集成Matplotlib

IPython在分析计算领域能够流行的原因之一是它非常好的集成了数据可视化和其它用户界面库，比如matplotlib

```
%matplotlib
import matplotlib.pyplot as plt
plt.plot(np.random.randn(50).cumsum())
```