

Title: W1 SQL入门 分析城市变暖趋势

Tags: 数据分析初级, SQL, 实战项目

# W1 SQL入门 分析城市变暖趋势

- [W1 SQL入门 分析城市变暖趋势](#)
- [学习地图](#)
- [/目标1/: SQL和移动平均值](#)
- [/目标2/: 浏览项目内容](#)
- 彩蛋
  - [如何适应节奏12周毕业](#)

## 学习地图

同学们，我们的纳米学位从明天就要开始了，我先介绍下咱们课程的安排。这个课程由4个真金白银的项目组成，说是真金白银，原因是这4个项目：

1. 从实际工作场景中变化得来；
2. 这4个项目涵盖了数据分析基础的4个最重要知识点；
3. 这4个项目，必须认真完成提交，并且通过评审老师的审阅才算通过。

为了能够保证大家的学习效果，我们这4个项目都是按照3个阶段设计的：

1. 第一阶段，攻克所有重难点，完成项目梗概（高强度学习阶段）；
2. 第二阶段，提交和修改（提交阶段）；
3. 第三阶段，进度慢点同学的查漏补缺（灵活假期），这样前紧后松的节奏来安排的。

如此循环4次，我们就可以毕业了！（此处可以有掌声！）。为了大家能够适应Uda的学习节奏，和互相认识（撩），咱们第一个项目比较简单，在接下来的这一周中，我们将完成：**项目1：探索未来气候发展趋势**

在这个项目中，我们将分析本地和全球的气温数据，并比较你居住地的气温走向与全球气温走向。你的任务是让数据可视化，描述全球气温走向和最接近你居住地的大城市气温走向之间的相似性与差异。

另外，为了能够顺利完成项目1，请准备好以下环境：

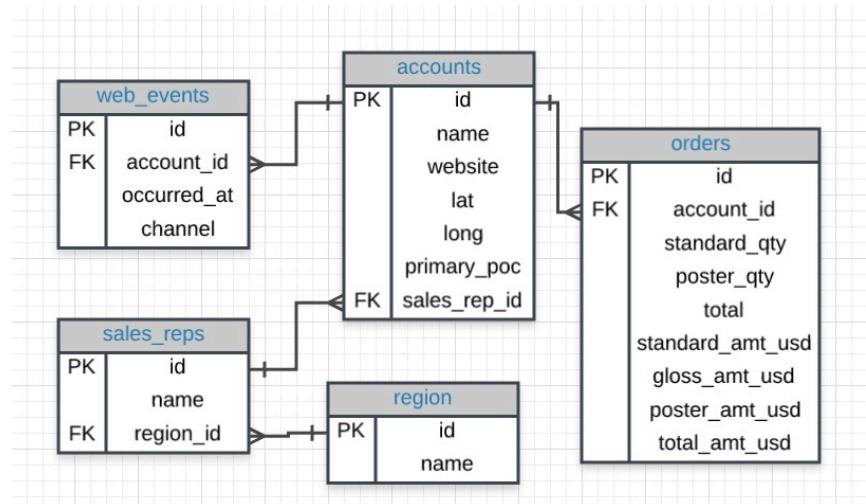
1. Chrome浏览器（和优达兼容性最好）

- 质量稳定的宽带（有时教师刷不出是网络问题，如果遇到的话可以试试手机4g开热点连接）
- Office软件（MS的Excel或者google docs 「如果能访问的话」

# /目标1/: SQL和移动平均值

所有部分包括（1-48）小节（不用害怕，每节很短的，2天差不多就看完了），记得完成所有的练习。对应本周/SQL初探/的内容，重点内容如下，其中/xx/x为对应相关视频小结，便于大家回看复习。

- Spreadsheets - 电子表格（比如Excel）
- ERD - Entity Relationship Diagram / 实体关系图。（了解）大家注意其中有PK和FK，其实这是对右侧column的限制（比如说所有id都是PK，是Primary Key，说明这个列是主键，几种column的标识如下：
  - 主键约束 SQL中constraint PK\_字段 primary key(字段),
  - 唯一约束 constraint UK\_字段 unique key(字段),
  - 默认约束 constraint DF\_字段 default('默认值') for 字段,
  - 检查约束 constraint CK\_字段 check(约束。如：len(字段)>1),
  - 主外键关系 constraint FK\_主表\_从表 foreign(外键字段) references 主表(主表主键字段)



- /2/SQL - a language used to interact with a database / 用于与数据库交互的语言
- /28/派生列：Derived Column。我们将现有的列组合，生成的新列称为派生列。在生成以后，可以用AS为这列起名（否则筛选出的结果这列名字是？Column这样的）

```
1 | SELECT standard_qty / (standard_qty + gloss_qty + poster_qty) AS stand_ratio
```

- /31/逻辑运算符：使用WHERE时，后面的运算符有两种。第一种是表示大小判断的算术运

算符，第二种是可以用于进行文本之间判断的逻辑运算符，逻辑运算符包括：

- LIKE 表示符合通配符规则的都选出来。其中%表示其他可能的数字。
- IN 表示按照后面的值精确匹配（比如过滤出特定顾客的订单），可以是 (x, y) 这样的多值。

```
1 | WHERE name LIKE '%one%';
2 | WHERE channel IN ('organic', 'adwords');
```

- /47/SQL语句的顺序是：SFWOL（每行缩写）

```
1 | SELECT col1, col2
2 | FROM table1
3 | WHERE col3 > 5 AND col4 LIKE '%os%'
4 | ORDER BY col5
5 | LIMIT 10;
```

- /47/SQL语句为什么要大写：其实Select、SELECT、SeleCT这样都是能正常运行的。但是大家想过没有，为什么都可以运行？因为SQL语句在执行的时候会先把语句都转换成大写的。如果写了小写的，会在执行时候先进行转换。增加执行的时间，当语句很多的时候影响执行效率。一般大公司会对此做要求。而且在很长代码的时候大写更容易辨认。
- /Plus/数据库的一致性：本节中提到了SQL之所以能够高效的处理数据库中的数据，是因为数据库的列都是同一类别的数据（也叫Feature）。对于数据库做一点扩展：数据一致性就是数据保持一致，在分布式系统中，可以理解为多个节点中数据的值是一致的。
  - 强一致性：当更新操作完成之后，任何多个后续进程或者线程的访问都会返回最新的更新过的值。这种是对用户最友好的，就是用户上一次写什么，下一次就保证能读到什么。根据 CAP 理论，这种实现需要牺牲可用性。
  - 弱一致性：系统并不保证续进程或者线程的访问都会返回最新的更新过的值。系统在数据写入成功之后，不承诺立即可以读到最新写入的值，也不会具体的承诺多久之后可以读到。
  - 最终一致性：弱一致性的特定形式。系统保证在没有后续更新的前提下，系统最终返回上一次更新操作的值。在没有故障发生的前提下，不一致窗口的时间主要受通信延迟，系统负载和复制副本的个数影响。DNS 是一个典型的最终一致性系统。
  - 其实一致性是缘由CAP定理：一致性（Consistency）、可用性（Availability）和分区耐受性（Partition tolerance），3个属性只可能同时满足2个。
- /7/SQL代码是Statements，由clauses组成(SELECT、FROM可以简单的理解一个大写的是一个clauses，很多clauses组成了Statements)。
- /Plus/配置自己的SQL环境：
  - SQL是Structured Query Language的缩写，是人与关系型数据库交互的通用语言。
  - 不同的关系型数据库的代码会有一些区别。

- sqlite是一个轻量化的关系型数据库，下载后，在命令行调用就可以进入（和Uda的工作空间相同了），下载地址：<https://sqlite.org/download.html>
- python和数据库。有很多操作数据库的接口，比如sqlite3是用来操作sqlite库的。其实python一般不直接操作数据库，而是用一个orm框架作为中间层，用操作对象的方法来操作数据库，避免直接写sql语句，这样比较方便，也可以防止sql注入攻击。  
sqlalchemy是比较常用的orm，另外一些web框架也会提供自己的orm，比如django自带的就很好用[/廖雪峰的一篇orm介绍/](#)
- 移动平均数是干什么的？
  - 统计中的移动平均法则对动态数列的修匀的一种方法，是将动态数列的时距扩大。所不同的是采用逐期推移简单的算术平均法，计算出扩大时距的各个平均是，这一些列的推移的序时平均数就形成了一个新的数列，通过移动平均，现象短期不规则变动的影响被消除如果扩大的时距能与现象周期波动的时距相一致或为其倍数，就能进一步削弱季节变动和循环变动的影响，更好的反应现象发展的基本趋势。
  - 就是说为了减小波动。比如像项目中这种看长时间每年平均气温的需求。或者大家可以想想股票如果补仓了，那么所持有的所有股票，的平均值就变化了，这平均值一般是下一步决策的依据。
  - 移动平均（Moving Average, MA），是技术分析中一种分析时间序列数据，最简单、最常用的分析工具之一，如股价（如开盘价、收盘价、最高价、最低价）、回报或交易量等，抚平短期波动，反映长期趋势或周期，在变化无常的市场中能发挥特别的作用。移动平均线也是其它许多技术指标的基石。
  - 可以参考：[/移动平均数简介/](#)

## /目标2/：浏览项目内容

这部分请大家看一下项目的内容。当理解项目背景之后，大家能够知道完成项目的3个步骤：

- step1：使用SQL语言过滤并下载csv文件
  - 首先我们要选择需要那个城市，数据中有个表提供了可选，为了能够快速浏览中国都有那个城市，我们还可以用ORDER BY来排序，这样中国的城市就都在一起了：

```

1 | SELECT *
2 | FROM city_list
3 | ORDER BY country;
```

输出就是这个样地，我们就能知道有那个城市可以选了，选一个你喜欢的就好

1	Jilin	China
2	Kunming	China

3	Xuzhou	China
4	Xian	China

那么我们来选，别忘了地点和时间两个限定条件

！特别注意！数据要选30个比较合适（适合使用正态分布的起始数据数，这个后面会讲，大家明白太小了不好使）！

```
1 | SELECT *
2 | FROM city_data
3 | WHERE city = 'Shanghai' AND year > '2010'
4 | ORDER BY year;
```

- step2：使用spreadsheet工具打（就是excel或者google表单）打开csv文件制作可视化图表。
  - 此处注意按照要求是两个csv文件，可以把两个文件内容copy到1个文件，注意列名要有区分，方便出图。
  - !一定要另存为xlsx（google就是google的格式）， csv文件是不能存图的信息的！
- step3：将报告生成pdf文件（另存为pdf即可）并提交项目
  - 这里按照项目要求，把内容写到word（或者google doc），之后另存为pdf，就可以完成了。
  - 一定注意不能有中文名字，还有认真看项目要求，不要拉问题！

## 彩蛋

### 如何适应节奏12周毕业

写下这些絮叨的助教大大是一个（自认为）又帅又认真的大龄IT男，他用了8个月0基础完成了数据分析初级，高级，还参与了新Python课程的测试。如果用1个字对这段经历做总结，那就是：充实（，，，2个字了）。长话短说，自我突破是很痛苦的，Uda的这门课很好的特色小结如下：

1. 第一，能够用得到。要知道，学得再好，用不起来也白搭。数据分析的场景越来越多，数据集也非常多，还有各种竞赛，如果想用起来，大环境已经非常成熟了。而且在工作也确实有很多机会使用，比如之前自己用excel公式做了半自动化的报告就觉得很好了，学了课程才会了python这种更加风骚的操作。

2. 第二，学习曲线科学。不要会错意，对于初入门的小白来讲，课程还是挺难的。来听听想毕业都要会什么：Python语言、SQL语言、统计学、数据分析思维、A/B测试、Anaconda环境、Sublime IDE……。害不害怕？好了，不哭了……要知道Uda的学费没白交。Uda收了钱以后，还是做了很多努力平滑大家的学习曲线的：
  - i. 课程是项目制。4个项目就要交4个报告，还有非常专业（…严格的）评审老师。
  - ii. 微信通关群。在学习的过程中，每个项目有通关群，群里有助教老师耐心的解决问题。
  - iii. 论坛和1对1辅导。如果涉及代码比较复杂还可以发论坛（好多问题论坛里都有的……忍不住剧透了）。最后还有大招每周1次1对1电话辅导。
  - iv. Uda在非常努力的提供反馈，甚至是不厌其烦的催促你在学，班主任小姐姐助教大大是非常认真的。
3. 第三，自我的突破。之前在知乎周刊看到一句话：。觉得会写代码又拽又神秘。也陆续买书瞎看，进展可以说…顶Uda一周的学习内容吧。在这次学习期间，不但一下入门了Python, R, JavaScript（用于数据分析的差不多了，目标明确，我是要用代码来分析数据的！）还学会了怎么应对统计分析、数据可视化、探索性数据分析、机器学习等等实际工作的实现。接下来就是自己积累了。

作为第一天入坑，同学们除了有些慌张，是不是还有点小激动？别忘了大家承诺的时间，一切准备就绪，就等你到达战场了，加油！

PS：作为准备进入数据分析世界的人们，以下内容将会培养你的基础技能，请大家长期积累：

- /在线学习第一课/ 中的：如何搜索和如何提问两部分
- [/Uda总结的资源/](#)
- [/Anaconda和Jupyter Notebook/](#)除了学习，安装本地资源更好
- Python基础学习：请学习py3版本。简单的可选：[/廖雪峰的免费课程/](#)

Title: W2 Python基础 [项目：探索共享单车用户行为规律1/4]

Tags: 数据分析初级, 实战项目

# W2 Python基础 [项目：探索共享单车用户行为规律1/4]

---

- [W2 Python基础 \[项目：探索共享单车用户行为规律1/4\]](#)
- [/学习地图/](#)
  - [/ 项目路径](#)
  - [/ 项目推进](#)
  - [/ 环境推荐](#)
  - [/ Atom软件扩展的安装](#)
  - [学习计划](#)
- [/目标1/: 数据类型和运算符](#)
- [/目标2/: 控制流](#)
  - [/ 17.Zip和Enumerate](#)
- [/目标3/: 函数](#)
  - [/ 5.变量作用域](#)
  - [/8.文档](#)
  - [/ 14.迭代器和生成器（选学）/](#)
- [/目标4/: 脚本编写](#)
  - [/ 8.在脚本中接受原始输入/格式化字符串](#)
  - [/13.处理错误](#)
  - [/ 17.读写文件](#)
  - [/ 18.读写文件](#)
  - [/20.导入本地模块](#)
  - [/ 26.第三方库](#)
  - [/ 28.在线资源](#)
  - [项目内容](#)
- [/彩蛋/](#)

## /学习地图/

---

本周开始，我们要进行第二个项目：探索共享单车用户行为规律的学习了。

- 项目说明:在此项目中，你将利用 Python 探索与以下三大美国城市的自行车共享系统相关的数据：芝加哥、纽约和华盛顿特区。你将编写代码导入数据，并通过计算描述性统计数据回答有趣的问题。你还将写一个脚本，该脚本会接受原始输入并在终端中创建交互式体验，以展现这些统计信息。
- 练习目标：数据分析过程、数据整理、探索和可视化
- 项目概述：请先看下这一页，对项目有个感触[/项目简介/](#)

## / 项目路径

---

项目名称：探索共享单车用户行为规律

项目时间：4周

- 第1周：Python基础内容 - Python基础（数据类型和运算符、控制流、函数、脚本编写）
- 第2周：Python数据处理内容 - Python数据分析（Numpy、Pandas库）
- 第3周：完成项目 - 项目：探索美国共享单车数据
- 第4周：通过项目 - 修改项目、查缺补漏、整理笔记

## / 项目推进

---

**Week1要求（本周任务）：**

- 完成探索美国共享单车数据项目的1-3节内容
- 搭建本地anaconda环境（Python3版本）确保Spyder可以使用
- 下载bikeshare-new-2.zip项目文件。如果教室里面不能下载，请下载：[/我搬的砖/](#)

## / 环境推荐

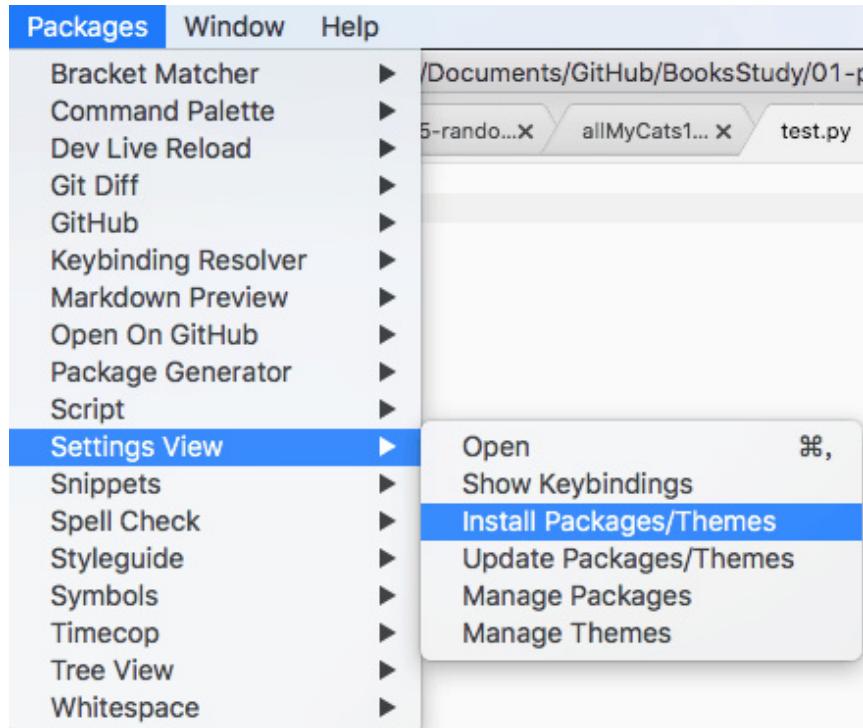
---

另外，为了能够顺利完成项目2，请准备好以下环境：

1. （非必须但推荐）本地Anaconda环境，安装后可以运行：Python3、Jupyter Notebook、Spyder/[Anaconda的和Jupyter Notebook安装配置说明/](#)、[/Spyder的简单教程/](#)
2. 如果想自己安装也可以选择Atom和Sublime两个，和Spyder是一样的，但要单独安装，有点麻烦。当然，有经验的同学用自己习惯的就好了
3. 在本周第4部分也有Atom的使用教程，可以先看或者顺序看：[/配置Python编程环境/](#)

## / Atom软件扩展的安装

---

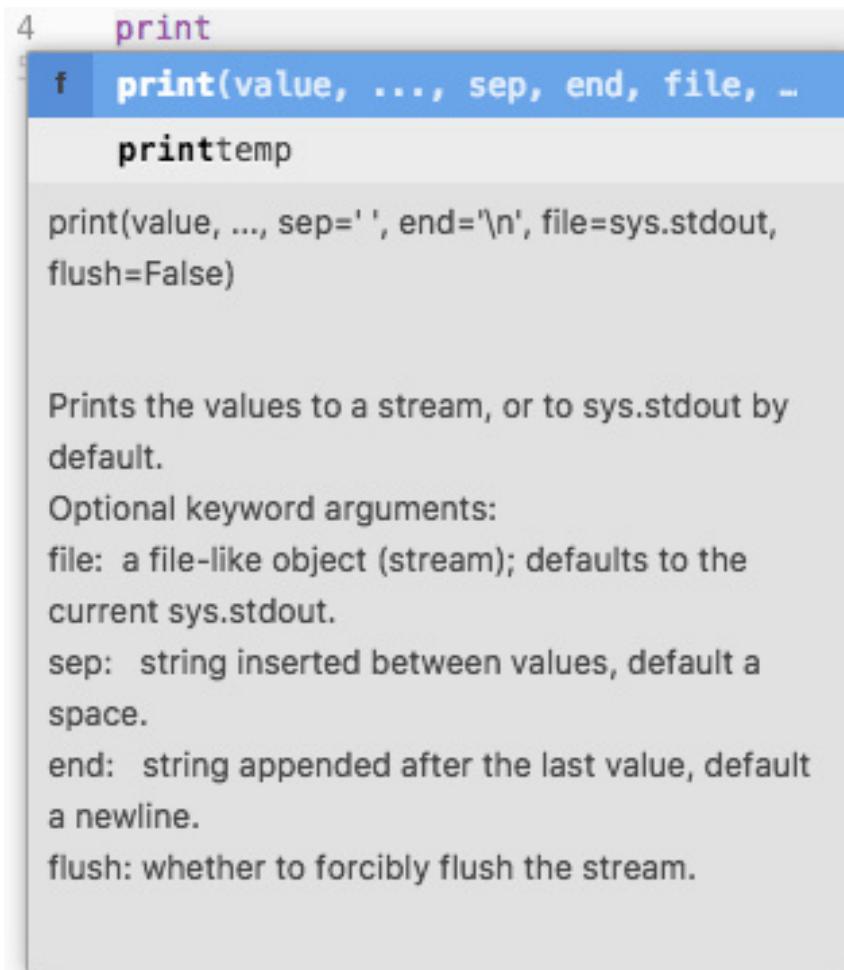


搜索并安装（配置也在packages的下面），安装时可能会需要安装这个扩展依赖的别的扩展，一路确认就好了。

The screenshot shows the Atom package manager interface. On the left, there's a sidebar with icons for Core, Editor, URI Handling, Keybindings, Packages (which is selected and highlighted in blue), Themes, and Updates. Below the sidebar is a button labeled '+ Install'. On the right, the main area has a title '+ Install Packages'. It includes a note: '② Packages are published to atom.io and are installed to /Users/mengfanchun/.atom/packages'. There are two search results: 'pylint' and 'linter-sass-lint'. The first result, 'linter-pylint 2.1.1', is described as 'Lint python on the fly, using pylint' and is developed by 'AtomLinter'. It has 115,834 installations. The second result, 'linter-sass-lint 1.8.3', is described as 'Atom Linter plugin to lint your Sass/SCSS with pure node sass-lint' and is also developed by 'AtomLinter'. It has 205,388 installations. Both results have 'Install' buttons.

推荐扩展：script 使用command + i 可以在atom中运行py

推荐扩展：pylint 可以对输入的语句做详细提示，当你输入后会这样：



## 学习计划

看着还挺丰富的对吧，还记得我们完成项目的3个阶段么？

1. 第一阶段，攻克所有重难点，完成项目梗概（高强度学习阶段）；
2. 第二阶段，提交和修改（提交阶段）；
3. 第三阶段，进度慢点同学的查漏补缺（灵活假期），这样前紧后松的节奏来安排的。

这4周我们讲完成/4. Python入门/部分的所有内容，拆分后的每周工作是这样的：

时间	学习重点	对应内容（按照名字找就好了）
第1周	Python基础内容	数据类型和运算符、控制流、函数、脚本编写
第2周	Python数据处理内容	Numpy & Pandas - 第一、二部分
第3周	运用前2周的知识完成项目	项目：探索美国共享单车数据
第4周	项目修改与通过	修改项目、查缺补漏、休息调整

每周的时间还是按照这个进度，注意周六的时间是大家Classin视频讨论的时间，其他的可以灵活调配，记住目标不要拖过一周为妙，加油！

学习时间	学习资源	学习重点
周2	/助教/发布当周导学	浏览导学文件内容开始学习
周3、周4	/Uda/线上内容	学习Uda Classroom内容
周5	/助教/1v1预约	难点可预约1v1语音指导
周6 20:30-21:30	/助教/视频讲解	讲解本周导学内容、回答疑难问题
周7	/小结/本周总结	总结、笔记、思考
周1	/选学/自主学习修养	自主学习（选学部分）或调休

## /目标1/：数据类型和运算符

此处和试学项目的内容相同，可以略过有不熟悉的再回看。

## /目标2/：控制流

此处就不是选学了，对于任何一种编程语言，控制流（循环）是非常重要的，请按以下顺序完成：

- 完成课程1-16内容
- 完成试学项目的选看内容 <https://github.com/mengfanchun2017/DAND-Basic-P0/blob/master/day6-guide.md>
- 完成课程17-22部分，之后看下面扩展

## / 17.Zip和Enumerate

- ```

1 | - 我们先来说zip，就是拉链的意思，可以把进行组合和拆分，课程里的例子非常详细。其实zip
2 | 函数可以进行行列转换，感兴趣的话可以看这个: https://blog.csdn.net/shomy\_liu/article/details/46968651
3 |
    - 接下来是Enumerate函数，其实就是简化for的一个内置函数，让你的循环更漂亮！
    - 最后是List Comprehension列表推导式：可以把一个设定初始空值，用条件循环填充的循环
      ，简化成一个赋值语句，对比以下，注意else语句的位置是在前面：

```

```

1 | #for loop:
2 | capitalized_cities = []

```

```

3 | for city in cities:
4 |     capitalized_cities.append(city.title())
5 |
6 | #list comprehension:
7 | capitalized_cities = [city.title() for city in cities]
8 |
9 | #list comprehension else:
10 | squares = [x**2 if x % 2 == 0 else x + 3 for x in range(9)]

```

书中练习的解释（要先做完在看呦）：

```

1 | #练习1
2 | names = ["Rick Sanchez", "Morty Smith", "Summer Smith", "Jerry Smith", "Beth Smith"]
3 | first_names = [name.split()[0].lower() for name in names]
4 | ##此处split会把Rick Sanchez分解为Rick和Sanchez，通过[0]就是选定了Rick
5 | print(first_names)
6 | #['rick', 'morty', 'summer', 'jerry', 'beth']
7 |
8 |
9 | #练习3
10 | scores = {
11 |     "Rick Sanchez": 70,
12 |     "Morty Smith": 35,
13 |     "Summer Smith": 82,
14 |     "Jerry Smith": 23,
15 |     "Beth Smith": 98
16 | }
17 |
18 | passed = [name for name, score in scores.items() if score >= 65]
19 | ##scores.items()就是把最前面的scores字典拆分成index (人名, Rick Sanchez) 和val
20 | ue (70)，这样的话就赋值给前面的name, score
21 | print(passed)
22 |

#输出
['Beth Smith', 'Summer Smith', 'Rick Sanchez']

```

## /目标3/：函数

### / 5. 变量作用域

如果遇到UnboundLocalError说明函数中对函数外定义的变量进行了修改。Python 不允许函数修

改不在函数作用域内的变量。这个原则仅适用于整数和字符串，列表、字典、集合、类中可以在子程序中（子函数）通过修改局部变量达到修改全局变量的目的。

## /8. 文档

docstrings的用途请见 <https://github.com/mengfanchun2017/DAND-Basic-P0/blob/master/day3-guide.md> 最后一段。docstrings的显示有两种方法，注意这两种方式都不用知道函数要求的参数是什么：

```
1 | help(functions)
2 | print(functions.__doc__)
```

1 | - 单引号和双引号都是OK的

- /11Lambda表达式/

- 当一个简单函数只会使用一次的时候，可以使用匿名函数的方式进行表达。比如下面这个例子，double为赋值对象，涉及2个参数x, y（就是lambda后面跟的），计算的时候吧两个参数乘积为结果（函数的内容就是冒号后面的东西）
- 这里重点要讲下map()函数，map函数的作用就是根据函数，对指定的序列做计算
  - 语法是这样的（注意先是要怎么处理数据的函数，后是要处理的数）：

```
1 | #map函数
2 | map(function, iterable, ...)
```

1 | 当然这个function也是可以使用lambda一次完成的，两种方式对比如下：

```
1 | #使用函数
2 | def square(x) :
3 |     return x ** 2
4 | map(square, [1,2,3,4,5])
5 | #使用lambda
6 | map(lambda x: x ** 2, [1, 2, 3, 4, 5])
7 | #两种方式的结果是相同的:
8 | [1, 4, 9, 16, 25]
9 |
10 | #注意map是可以有多个输入的
11 | map(lambda x, y: x + y, [1, 3, 5, 7, 9], [2, 4, 6, 8, 10])
12 | #结果为
13 | [3, 7, 11, 15, 19]
```

接下来我们就能看懂教室里的例子了：

```
1 | numbers = [
2 |     [34, 63, 88, 71, 29],
3 |     [90, 78, 51, 27, 45],
4 |     [63, 37, 85, 46, 22],
5 |     [51, 22, 34, 11, 18]
6 | ]
7 | #首先numbers是一个嵌套的列表，有4个元素，每个元素（每行）又包括4个元素
8 |
9 | averages = list(map(lambda x :sum(x)/len(x),numbers))
10 | #此处的list是将map生成的4行平均数存为一个列表
11 | #lambda的内容是: sum(num_list)/len(num_list)，用每个元素的加和除以每个元素内部
12 | 的个数
#最后numbers是输入
```

filter函数和map类似，请看这个说明：<http://www.runoob.com/python/python-func-filter.html>

## / 14.迭代器和生成器（选学） /

这部分有点绕，选学，想看更详细的可以参考：<https://www.zhihu.com/question/20829330>

## / 目标4/：脚本编写

### / 8.在脚本中接受原始输入/格式化字符串

这里出现了个有点奇怪的print函数：

```
1 | name = input("Enter your name: ")
2 | print("Hello there, {}!".format(name.title()))
```

有点奇怪啊，不就是让用户输入个名字，不是应该这样的么？

```
1 | name = input("Enter your name: ")
2 | print("Hello there, ", name, "!")
```

对的，实际上输出是一样的，后面这个我们比较熟悉，把字符和变量串在一起输出。但是观察上下对比，是不是上面的这个比较简单呢？这种新的方法叫做：**print格式化字符串**。大家对比观察一下，其实就是在print里面放了个{}，并在后面加了个.format(name.title())。这个语句的意思是，

打印到{}的时候，把后面这个.format()里的东西打印出来，name.title就是把输入的name的第一个字母改为大写。默认{} {} {}...会按照后面.format(a, b, c)来替换,但也可以指定。比如{0}指定的是a, {1}指定的是b, 以此类推。举个例子就知道了：

```
1 ##format method
2 print('---test1:---')
3 print('I am lucky to eat {} {} {} {}'.format(4, 'eggs', 1, 'spam'))
4 print('---test2:---')
5 print('I am lucky to eat {} {} {} {}'.format(4, 'eggs', 1, 'spam'))
6 print('---test3:(option)---')
7 print('I am lucky to eat {:.2f} {} {} {}'.format(4, 'eggs', 1, 'spam'))
8 ')
9 print('---test4:(option)---')
print('I am lucky to eat {:.2f} {}:{}^20 {} {}'.format(4, 'eggs', 1, 'spam'))
```

输出是这样的：其中test3, 4是更为复杂的应用，感兴趣的话看这两个链接：[https://blog.csdn.net/i\\_chaoren/article/details/77922939](https://blog.csdn.net/i_chaoren/article/details/77922939)  
<https://www.cnblogs.com/wilber2013/p/4641616.html>

```
1 ---test1:---
2 I am lucky to eat 4 eggs 1 spam!
3 ---test2:---
4 I am lucky to eat 1 eggs spam 4!
5 ---test3:(option)---
6 I am lucky to eat 1.00 eggs spam 4!
7 ---test4:(option)---
8 I am lucky to eat 1.00 #####eggs##### spam 4!
```

print再扩展一点，如果看到这样的家伙 + variable + ，是一种在字符串中加变量的方法，其实和逗号分隔是等价的：

```
1 print('hi ' + name + ' !')
2 print("hi " + name + " !")
3 print('hi', name, '!')
4 print("hi", name, "!")
5 #这4个的输出都是一样的：（注意下代码前后的空格是不一样的，所以前面的也会用到）
6 hi handsome !
```

其实普通的就print，需要处理复杂的就.format (.format还有一种简化写法，不管他）。print就都OK了，详细的带入列表和字符串的方法：<https://stackoverflow.com/questions/17153779/how-can-i-print-variable-and-string-on-same-line-in-python>

这一节后面还有个eval是把用户输入的内容当作python代码处理。扩展下也可以这样使用，把str字符串转化为响应的内容(>>>是输入的代码），大家注意a和b的type是不一样的：

```
1 >>> a = "{1: 'a', 2: 'b'}"
2 >>> type(a)
3 <type 'str'>
4
5 >>> b = eval(a)
6 >>> print b
7 {1: 'a', 2: 'b'}
8 >>> type(b)
9 <type 'dict'>
```

## /13. 处理错误

大家理解try、except、else、finally4个语句的执行条件就好了。

- try：这是 try 语句中的唯一必需子句。该块中的代码是 Python 在 try 语句中首先运行的代码。
- except：如果 Python 在运行 try 块时遇到异常，它将跳到处理该异常的 except 块。
- else：如果 Python 在运行 try 块时没有遇到异常，它将在运行 try 块后运行该块中的代码。
- finally：在 Python 离开此 try 语句之前，在任何情形下它都将运行此 finally 块中的代码，即使要结束程序，例如：如果 Python 在运行 except 或 else 块中的代码时遇到错误，在停止程序之前，依然会执行此finally 块。
- 对于/15/的例子做个简单的说明（以注释方式）：

```
1 def create_groups(items, num_groups):
2     # 定义函数，2个输入items（多少个东西），分成num_groups(分成多少个组)
3     try:
4         size = len(items) // num_groups
5         # 上来是计算每组大小
6     except ZeroDivisionError:
7         print("WARNING: Returning empty list. Please use a nonzero number")
8         return []
9     # 但是当发生ZeroDivisionError时（除数为0的时候的错误），就显示错误并返回空值
10    else:
11        groups = []
12        for i in range(0, len(items), size):
13            # 是说从0到items的最大数（就是len(items)这个的结果），按照size（在try语句
14            中得出的每组大小）进行循环
```

```
16         groups.append(items[i:i + size])
17         #每一个循环，把当前的组存追加存放到groups里面
18     return groups
19 #如果没报错，就是按照上面这一段把每组都有什么写到groups里面
20 finally:
21     print("{} groups returned.".format(num_groups))
22     #无论怎么处理的，都打印一行提示，使用的是格式化字符串的方式
23
24 print("Creating 6 groups...")
25 for group in create_groups(range(32), 6):
26     print(list(group))
27 print("\nCreating 0 groups...")
28 for group in create_groups(range(32), 0):
29     print(list(group))
```

输出是这样的：

```
1 Creating 6 groups...
2 6 groups returned.
3 [0, 1, 2, 3, 4]
4 [5, 6, 7, 8, 9]
5 [10, 11, 12, 13, 14]
6 [15, 16, 17, 18, 19]
7 [20, 21, 22, 23, 24]
8 [25, 26, 27, 28, 29]
9 [30, 31]
10
11 Creating 0 groups...
12 WARNING: Returning empty list. Please use a nonzero number.
13 0 groups returned.
```

因为函数输出的是一个嵌套的列表，所以要用上面的方式把列表的每一组元素显示出来，我们加一句print就能看明白了：

```
1 | print(create_groups(range(38), 3))
```

输出是这样的：

```
1 | 3 groups returned.
2 | [range(0, 12), range(12, 24), range(24, 36), range(36, 38)]
```

另外，即是except有报错输出，也是可以通except as的方式访问并输出的，这一块知道就行了。

## / 17.读写文件

注意append和write区别， 和with的用法

- 追加文件是用f.append(), 使用f.write()将会覆盖文件
- 为了避免忘记f.close()关闭一个文件， 可以使用with的方式：

```
1 with open('my_path/my_file.txt', 'r') as f:  
2     file_data = f.read()  
3 #和下面的语句是一样的  
4 f = open('my_path/my_file.txt', 'w')  
5 file_data = f.read()  
6 f.close()
```

## / 18.读写文件

注意这个地方有个新的.split(',')方法， 用处是把line里面的内容用， split开， 关于readline相关的用法， 总结为以下几个例子：

```
1 print('{0:-^30}'.format('print read'))  
2 #print函数会在结尾自动加入换行  
3 with open('print.format.py') as song:  
4     print(song.read(1))  
5     print(song.read(8))  
6     print(song.read(8))  
7     #print(song.read())  
8  
9 print('{0:-^30}'.format('print read end none'))  
10 with open('print.format.py') as song:  
11     print(song.read(1), end = '')  
12     print(song.read(8), end = '')  
13     print(song.read(8))  
14     #print(song.read())  
15  
16 print('{0:-^30}'.format('print readline'))  
17 with open('print.format.py') as song:  
18     print(song.readline())  
19     print(song.readline())  
20     print(song.readline(1), end = '\n\n')  
21     #可以看出readline()是每次读取一行 (/n换行跟随上一行， 不会算成下一行)  
22     #如果readline(x)，就是读出这行的x个字符  
23     #这种方式时结尾的/n不会打印
```

```

24     #结尾\n\n 才会换行，一个的话会追加到x个字符后面
25
26 print('{0:-^30}'.format('print readlines'))
27 with open('print.format.py') as song:
28     print(song.readlines(), end = '\n\n')
29     #readlines是把所有行读入到一个列表中
30
31 print('{0:-^30}'.format('for line in file'))
32 test_lines = []
33 with open('print.format.py') as song:
34     #line可以替换成i, 只不过line比较明确
35     #这里重点是, 对于open的文件, for 循环是每次循环一行
36     for line in song:
37         test_lines.append(line.strip())
38     print(test_lines, end = '\n\n')
39
40 print('{0:-^30}'.format('split lines'))
41 def create_cast_list(filename):
42     cast_list = []
43     #use with to open the file filename
44     with open("circus.csv") as f:
45         for line in f:
46             name=line.split(',')[0]
47             cast_list.append(name.strip())
48     #下面的是最后一个循环的输出, 输出做对比就明白很多了:
49     print('{0:-^30}'.format('under is split testing'))
50     print('originial: {0:#^20}'.format(line), end = '')
51     nameall = line.split(',')
52     nameallfirst = line.split(',')[0]
53     print(name)
54     print(nameall)
55     print(nameallfirst)
56     #use the for loop syntax to process each line
57     #and add the actor name to cast_list
58
59     return cast_list
60
61 cast_list = create_cast_list('circus.csv')
62 for actor in cast_list:
63     print(actor)

```

结果如下，感兴趣的可以自己研究下：

```

1 -----print read-----
2 #

```

```

3 #format
4 methord
5
6 -----print read end none-----
7 ##format methord
8
9 -----print readline-----
10 ##format methord
11
12 print('---test1:---')
13
14 p
15
16 -----print readlines-----
17 ['##format methord\n', "print('---test1:---')\n", "print('I am lucky to eat {} {} {} {}!'.format(4,'eggs', 1, 'spam'))\n", '\n', "print('---test2:---')\n", "print('I am lucky to eat {2} {1} {3} {0}!'.format(4,'eggs', 1, 'spam'))\n", '\n', "print('---test3:(option)---')\n", "print('I am lucky to eat {2:.2f} {1} {3} {0}!'.format(4,'eggs', 1, 'spam'))\n", '\n', "print('---test4:(option)---')\n", "print('I am lucky to eat {2:.2f} {1:#^20} {3} {0}!'.format(4,'eggs', 1, 'spam'))\n"]
24
25 -----for line in file-----
26 ['##format methord', "print('---test1:---')", "print('I am lucky to eat {} {} {} {}!'.format(4,'eggs', 1, 'spam'))", '', "print('---test2:---')", "print('I am lucky to eat {2} {1} {3} {0}!'.format(4,'eggs', 1, 'spam'))", '', "print('---test3:(option)---')", "print('I am lucky to eat {2:.2f} {1} {3} {0}!'.format(4,'eggs', 1, 'spam'))", '', "print('---test4:(option)---')", "print('I am lucky to eat {2:.2f} {1:#^20} {3} {0}!'.format(4,'eggs', 1, 'spam'))"]
33
34 -----split lines-----
35 ----under is split testing----
36 originial: The Fred Tomlinson Singers, Amantillado Chorus / ... (7 episodes, 1969-1973)
The Fred Tomlinson Singers
['The Fred Tomlinson Singers', ' Amantillado Chorus / ... (7 episodes', ' 1969-1973)\n']
The Fred Tomlinson Singers
Graham Chapman
Eric Idle
Terry Jones
Michael Palin
Terry Gilliam
John Cleese
Carol Cleveland

```

## /20.导入本地模块

接下来讲解下if main这一块：

- if **name == '\_\_main\_\_'** 简单的理解就是：如果模块是被直接运行的，则代码块被运行，如果模块是被导入的，则代码块不被运行
- 是为了在导入时候不运行（被调用才运行）的限制
- 详细说明：<http://blog.konghy.cn/2017/04/24/python-entry-program/>
- /23标准库/
- 这里介绍了random标准库的两个用法，总结如下：

```
1 import random
2 word_list = ['tattoo', 'happy', 'apple', 'ios', 4]
3
4 def generate_password():
5     return str(random.choice(word_list)) + str(random.choice(word_list))
6     + str(random.choice(word_list))
7     #增加了str确保如果wordlist里面有4这样的数字可以转化为字符
8 print(generate_password())
9
10 def generate_password2():
11     return ''.join(random.sample(word_list, 5))
12     #join方式就不能加str，要求wordlist都是字符，但是既然是wordlist就应该都保证是
13     #而不是在写处理代码时候再额外处理不推进str的方式
14 print(generate_password2())
```

## / 26.第三方库

在安装python或者anaconda后，可以使用： `pip install package_name` 来安装需要的包。推荐的一些安装包很实用，[/链接归档/](#)。当然也可以将需要的包放在一个文件中，批量安装 `pip install -r requirements.txt`， requirements.txt文件示例如下：

```
1 beautifulsoup4==4.5.1
2 bs4==0.0.1
3 pytz==2016.7
```

## / 28. 在线资源

一定要看！提升软能力！

### 项目内容

本首是Python项目的第1周，主要是理解项目和准备项目文件，请大家做到以下几点：

- 完成/项目：探索美国共享单车数据/的1-3节内容
- 搭建本地anaconda环境（Python3版本）确保Spyder可以使用
- 下载bikeshare-new-2.zip项目文件。如果教室里面不能下载，请尝试下载下面的链接：<https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project1/Project1Files/bikeshare-new-2.zip>
- 整理并保存好好提交项目1的文件目录，并用spyder打开项目文件浏览

### / 彩蛋 /

大家第1个项目，过了没！爽不爽啊！本周开始我们将会完成初级数据分析中最难搞也最能让你成长的项目了。希望大家能够按照本周导学的内容，先学习完内容，在准备好项目的环境和文件。

其中学习的内容主要是4部分，大家可以每天搞一个，这样周六咱们视频讲解的时候，就是复习，而不是第一次听了。本周虽然没有项目提交，但是内容还是比较多的，如果同学是编程新手的话，一定记得多看、多试、多问，大家一起加油，给项目2做个好的开端！另外对于试学项目的导学大家也可以回顾下：[/试学导学内容/](#)

Title: W3 Pandas和Numpy [项目：探索共享单车用户行为规律2/4]

Tags: 数据分析初级

# W3 Pandas和Numpy [项目：探索共享单车用户行为规律2/4]

---

- W3 Pandas和Numpy [项目：探索共享单车用户行为规律2/4]
- /学习地图/
  - / 项目路径
  - / 项目推进
  - / 重点提示
- /目标1/: Numpy & Pandas 第一部分
  - \*/ 4.Numpy和Pandas中的一维数组
  - \*\*/ 5.NumPy 数组
  - \*\*/ 8.练习：计算整体完成率
  - \*\*/ 9.标准化数据/学会调用numpy的功能
  - \*/ 10.练习：NumPy索引数组
  - \*/ 12.练习：原地与非原地
  - \*\*\*/ 13.练习：Pandas Series
  - \*\*\*/ 14.练习：Series索引
  - \*\*/ 17.练习：Pandas Series apply()
  - \*/ 18.练习：在Pandas中绘图
- /目标2/: Numpy & Pandas 第二部分
  - \*\*\*/ 3.练习：二维Numpy数组
  - \*/ 4.练习：Numpy轴
  - \*/ 5.NumPy和Pandas数据类型
  - \*\*\*/ 6.练习：访问DataFrame元素
  - \*/ 7.将数据加载到 DataFrame 中
  - \*/ 8.练习：计算相关性
  - \*/ 9.Pandas 轴名
  - \*/ 10.练习：DataFrame向量化运算
  - \*\*/ 11.练习：DataFrame applymap()
  - \*/ 12.13.练习：DataFrame apply()
  - \*/ 14.练习：向Series添加DataFrame
  - \*/ 15.练习：再次归一化每一列

- \*\*\*/ 16.练习：Pandas groupby()
- \*\*\*/ 17.练习：每小时入站和出站数
- \*/ 18.练习：合并Pandas DataFrame
- \*/ 19.练习：使用DataFrame绘制图形
- \*/ 20.三维数组
- /彩蛋/

## /学习地图/

---

上周我们进行了Python基础内容的学习，对于没有太多基础的同学来讲可能有些头疼，本周开始我们将会完成数据分析中常用的dataframe数据结构的学习。这一周的重点是学习Python中的NumPy和Pandas两个第三方库。希望大家能够按照本周导学的内容，先学习完内容，再准备好项目的环境和文件。

## /项目路径

---

项目名称：探索共享单车用户行为规律

项目时间：4周

- 第1周：Python基础内容 - Python基础（数据类型和运算符、控制流、函数、脚本编写）
- **第2周：Python数据处理内容 - Python数据分析（Numpy、Pandas库）**
- 第3周：完成项目 - 项目：探索美国共享单车数据
- 第4周：通过项目 - 修改项目、查缺补漏、整理笔记

## /项目推进

---

**Week1要求（已完成）：**

- 完成探索美国共享单车数据项目的1-3节内容
- 搭建本地anaconda环境（Python3版本）确保Spyder可以使用
- 下载bikeshare-new-2.zip项目文件。如果教室里面不能下载，请下载：[/我搬的砖/](#)

**Week2要求（本周任务）：**

- 能够打开项目文件（用Uda线上的可以，但建议可以试试用Spyder本地打开项目文件浏览）
- 了解项目文件中有几个函数，函数名和输入是什么（其他的看不懂没关系）

## / 重点提示

- 每周任务会拆分成不同目标，对应的是课程中的Lesson，每个Lesson中的小节将通过“//编号.内容”进行对应
- 每小节前面将会放置1到3个星，代表与项目的相关性（也在一定程度上代表重要性）从低到高。请优先学习高优先级的任务。1颗星的是选学内容，请大家本周任务全部完成后再学习。
- 每周3发布的导学文件，建议在周3到周6学习完成。
- 每周6的20:30-21:30时间为优达日现场讲解，主要内容是1) 答疑、2) 讲解本周导学文件。
- 如果同学是编程新手的话，一定记得多看、多试、多问，大家一起加油。

## / 目标1/： Numpy & Pandas 第一部分

不要怂，就是肝！ 对应Numpy 和 Pandas这两部分（尤其是后者）可是数据分析最重要的第三方库，当你找我了之后，你就会爱上这只大熊猫了！

### \*/ 4.Numpy和Pandas中的一维数组

了解下哦使用pandas读入csv文件比直接使用unicodecsv读入要快就好了。

### \*\*/ 5.NumPy 数组

Numpy中的Arrays和list列表很像，区别如下：

- Arrays中的元素要都是同一类别（数字，字符只能有一种）
- 可以使用Numpy中的很多函数，比如平均值mean(), 标准差std()。虽然list也可以使用这些函数，但Arrays运行起来更快
- Arrays可以扩展到多维
- 代码练习中的 if True: 就会执行后面一小段程序，如果是 if False: 就不会执行，是一种测试的习惯，大家可以更改True 或 False 来决定运行那一块代码。对代码def那里的说明如下：

```
1 | def max_employment(countries, employment):  
2 | #定义了一个函数max_employment,需要输入两个参数countries和employment  
3 |  
4 |     max_country = None  
5 |     max_value = 0
```

```
6     #将最大城市设为空， 将最大值设为0
7     for i in range(len(countries)):
8         country = countries[i]
9         country_employment = employment[i]
10    #开始循环， 对countries中的国家， 循环读取国家名和就业率
11    if country_employment > max_value:
12        max_country = country
13        max_value = country_employment
14    #如果就业率比最大值还大， 就把新发现的作为新的最大值和拥有最大值的国家
15
16    return (max_country, max_value)
17    #函数结束， 返回两个找到的最大值和对应国家
```

## \*\*/ 8.练习：计算整体完成率

按位与与逻辑与， 此处只需要看看练习， 了解numpy的向量化运算就可以了。感兴趣的可以看下面3个链接：

- 区别：<http://www.cnblogs.com/wudongyang/p/4340003.html>
- numpy逻辑或说明：[https://docs.scipy.org/doc/numpy/reference/generated/numpy.logical\\_or.html](https://docs.scipy.org/doc/numpy/reference/generated/numpy.logical_or.html)
- numpy逻辑或说明：[https://docs.scipy.org/doc/numpy/reference/generated/numpy.logical\\_not.html](https://docs.scipy.org/doc/numpy/reference/generated/numpy.logical_not.html)

## \*\*/ 9.标准化数据/学会调用numpy的功能

首先说明一下为什么我们要标准化呢？因为在很多数据进行比较的时候，范围、比例等都会有很大的变化，而这些体现在数值上来说，人根本就没有什么感觉（看到就发懵了，不好判断）。那么我们怎么办来比较不同的数据呢： 我们把大家按照比例缩放，都放到一个区域中呗，比如所有的数据都在0-1之间，那么我看到接近1的就知道很高了，接近0的就很低了，其他的不用考虑。前面的说明只是一个粗浅的举例，这种方式有很规则和应用。

细抠的话还有标准化和归一化等等细节，在机器学习中接触比较多，大家有兴趣看链接：<https://blog.csdn.net/fonthrone/article/details/74067064>

那么接下来这部分的例子就是要衡量各个国家的就业率，通过标准化，我们能够看出不同国家做的好坏，代码解释如下：

```
1 #其实这里的函数没有必要， 只用一行就ok了
2 def standardize_data(values):
```

```
3     standardize_values = (values - values.mean()) / values.std()
4     #这行就是进行表转化转换，注意里面的变量使用的是values，这是一个numpy的阵列，所以
5     #输出也是一个阵列（每个元素都计算一遍，再存入到阵列中）
6     #但是其中values.mean()是一个阵列所有元素的均值，是一个固定值；同样values.std()
7     #是标准差，也是一个固定值。
8     #标准差这里我们第四个项目再讲，大家看明白是怎么计算就好了
9     return standardize_values
```

关于numpy强大的数学运算功能，请大家参见：[https://www.tutorialspoint.com/numpy/numpy\\_statistical\\_functions.htm](https://www.tutorialspoint.com/numpy/numpy_statistical_functions.htm)

## \*/ 10.练习： NumPy索引数组

了解Array的索引数组使用方式，可以在一个数组后面加 [条件] 对数组进行过滤。

## \*/ 12.练习： 原地与非原地

了解list和numpy array在对于元素修改的不同之处。

## \*\*\*/ 13.练习： Pandas Series

从这节开始我们将会学习Pandas库，Pandas库里的Series和之前学习的Numpy Array是类似的。**Pandas是Python处理数据最重要的库**，以下几节需要多看几遍。相比之下，有更多扩展性，比如说：

Series.describe()方法，对于Series的数据内容自动计算平均值、标准偏差、中位数和一些其他的统计数据。做总体描述，便于观察数据情况。

Series包括了Array的大部分功能，总结如下（假设这个Series命名为S）：

- 访问元素 Accessing Elements - S[0], S[3:7]
- 循环 Looping - for x in S
- 功能计算 Convenient Functions - S.mean(), S.max()
- 向量运算 Vectorized Operation - S1 + S2

此处的练习还用到了上周的逻辑判断的内容，讲解一下函数部分：

```
1 def variable_correlation(variable1, variable2):
2     #此处定义了函数，两个变量，就是life_expectancy和gdp了
3     both_above = (variable1 > variable1.mean()) & \
```

```

4     (variable2 > variable2.mean())
5 #显示看都大于平均值的。首先看两个小括号，第一个是用变量1的每个值和变量1的平均值比
6 起 (.mean()就是求平均值)，第二个小括弧是比较变量2的。两个都比较好了之后，在做与操作
7 (就是说都是True的话，结果就为True)，其中 \ 是python中的换行，为了不超过一行79个字
8 符的限制
9 both_below = (variable1 < variable1.mean()) & \
10    (variable2 < variable2.mean())
11 #同上，改为判断是不是都低于平均值
12
13 is_same_direction = both_above | both_below
14 #接下来，根据要求吧都高于和都低于的进行与操作，就能看出同方向变化的量是谁了（这里
15 的意思是看看两个指标是不是相关啊）
16 num_same_direction = is_same_direction.sum()
17 #接着求出有多少个是相关的，运用的是.sum()就是求和，因为True是1所以所有True的加
18 起来就是17了
19 num_different_direction = len(variable1) - num_same_direction
20 #最后，那么不相关的就是用总数减去相关的就好了，len求总数（就是看这个序列有多少个
元素）

    return (num_same_direction, num_different_direction)
#函数的最后，返回这两个值17个相关，3个不相关。感觉预期寿命和GDP还是有关系的对不对
。 （这个练习可以当成一个mini项目了）

```

## \*\*\*/ 14.练习：Series索引

这节里面一句话亮了：**Numpy arrays**是增强版的**Python lists**，**Pandas series**像是**Python lists**与**dictionary**的结合。**lists**列表是根据顺序位置获取值的，而**dictionary**字典是通过**key-value**对应来获取值的，而在**series**中这两种方式都可以，对应的是**iloc**和**loc**方式。

这里面要注意的一个是里面使用的**argmax()**已经被**idxmax()**替代了，代码解释如下：

```

1 def max_employment(employment):
2     employment_pd = pd.Series(employment, index = countries)
3     #将employment的数据转换为Series，索引使用countries列表中的数据
4     max_country = employment_pd.idxmax()
5     #使用idmax找到最大的索引
6     max_value = employment_pd.loc[max_country]
7     #使用.loc[index]将最大索引对应的数值赋给max_value
8     return (max_country, max_value)
9     #返回max_country与max_value

```

## \*\*/ 17.练习：Pandas Series apply()

- 本节大家记住Panda series和Python lists在类似的，可以使用for循环。
- 并且学了一个.apply()的方法。可以将()的函数作用于所有serie的元素中，并生成新的serie。
- 还有做练习的时候用到一个.split方法，是将输入分隔使用的，我将在下面代码中用注释讲解，感兴趣的也可以看官方文档：<https://docs.python.org/2/library/stdtypes.html#str.split>
- 中文介绍文档：<https://my.oschina.net/hkmax/blog/146553>

```
1 def reverse_names(names):  
2     splited = names.split()  
3     #首先使用split将名字分为两个名字比如‘Lao Meng’就会变成['Lao', 'Meng']  
4     #当变成一个列表之后呢，就可以使用[索引号]分别选择‘Lao’和‘Meng’了  
5     new_names = splited[1] + ' ' + splited[0]  
6     #splited[1]代表‘Meng’， splited[0]代表‘Lao’  
7     #通过对new_names赋值，改变了名字的前后顺序  
8     #注意中间加了一个‘ ’空格  
9     return new_names  
10    #最后函数返回new_names新的名字  
11  
12    print(names.apply(reverse_names))  
13    #使用apply调用刚刚的函数，那么每个series中的值都会被这个函数处理一遍
```

## \*/ 18.练习：在Pandas中绘图

完全作为选修内容，在项目2中不要求。大家知道import的seaborn是一个用于绘图的库就好了。一个中文的说明：[http://blog.mazhangjing.com/2018/03/29/learn\\_seaborn/#233-多变量多图形拟合叠加](http://blog.mazhangjing.com/2018/03/29/learn_seaborn/#233-多变量多图形拟合叠加)

## /目标2/：Numpy & Pandas 第二部分

### \*\*\*/ 3.练习：二维Numpy数组

这个练习一定要认真，因为扩展到了2维。对于二维，可以这样理解：

- 列表是[item1,item2]
- 那么当列表中每个元素又是一个列表的时候呢，就会变成这样
- 嵌套列表[[item1a,item1b],[item2a,item2b]]

- 如果我们转换下形势，就变成这样了（有了 $2 \times 2$ 矩阵的感觉了）
  - $\begin{bmatrix} \text{item1a}, \text{item1b} \\ \text{item2a}, \text{item2b} \end{bmatrix}$
- 也就能对应下面这个表格了：

| item1a | item1b |
|--------|--------|
| item2a | item2b |

这样的话，我们就能理解练习中的二维数组：

```

1 | ridership = np.array([
2 |     [0, 0, 2, 5, 0],
3 |     [1478, 3877, 3674, 2328, 2539],
4 |     [1613, 4088, 3991, 6461, 2691],
5 |     [1560, 3392, 3826, 4787, 2613],
6 |     [1608, 4802, 3932, 4477, 2705],
7 |     [1576, 3933, 3909, 4979, 2685],
8 |     [95, 229, 255, 496, 201],
9 |     [2, 0, 1, 27, 0],
10 |    [1438, 3785, 3589, 4174, 2215],
11 |    [1342, 4043, 4009, 4665, 3033]
12 | ])

```

实际上是这个意思：

| 天\车站 | 车站1  | 车站2  | 车站3  | 车站4  | 车站5  |
|------|------|------|------|------|------|
| 第1天  | 0    | 0    | 2    | 5    | 0    |
| 第2天  | 1478 | 3877 | 3674 | 2328 | 2539 |
| 第3天  | 1631 | 4088 | 3991 | 6461 | 2691 |
| ...  | ...  | ...  | ...  | ...  | ...  |

练习中的代码解释如下：

```

1 | def mean_riders_for_max_station(ridership):
2 |     max_station = ridership[0,:].argmax()
3 |     # 通过argmax检查第1行那个值最大，如果想看看中间输出，把下面行#去掉打印看看
4 |     #print(ridership[0,:])
5 |
6 |     # [0,:]代表选中第一天的所有车站出入量，就是从[0 0 2 5 0]中选

```

```
7     # 结果是第4个元素5最大，所以max_station = 3
8     # argmax就是求最大元素的位置，注意是从0开始，所以第四个元素是3
9     mean_for_max = ridership[:,max_station].mean()
10    # 那么接下来要算这个车站的平均数，把这个车站所有的数量取平均
11    # [:,max_station]带入3就是[:,3]就是说第4列的所有值
12    # : 的意思是选中所有内容
13    overall_mean = ridership.mean()
14    # 这行代码是计算整体的客流量平均值
15    return (overall_mean, mean_for_max)
16
17 print(mean_riders_for_max_station(ridership))
18
19 # 扩展， argmax可以加axis参数，代表计算2d数组中横向切分和纵向切分的最大值位置（下一节
20 # 将会详细讲）
21 # 下面行将会返回每一列的最大值在那里
# print(ridership.argmax(axis = 0))
```

## \*/ 4.练习：Numpy轴

本节是上节的扩展，讲解了使用axis对二维数组的列进行操作的方法。看明白视频即可。

## \*/ 5.NumPy和Pandas数据类型

NumPy的多维数组，和Pandas的DataFrame是比较类似的，都是2维的数据结构。DataFrame是数据分析的重要数据结构，在特性上比较利于数据分析，这节要求看懂视频。比如说这两个区别：

- DataFrame里面的不同列可以是不同类型的数据，而2维的Array则只能是一个数据类型。
- DataFrame可以使用像字典的key value键值对应的结构。

## \*\*\*/ 6.练习：访问DataFrame元素

记住DataFrame是Pandas库里面的多维数据结构，二期将会在数据分析大量使用，本节练习请一定认真完成。

```
1 import pandas as pd
2
3 # Subway ridership for 5 stations on 10 different days
4 ridership_df = pd.DataFrame(
5     data=[[ 0, 0, 2, 5, 0],
```

```
6      [1478, 3877, 3674, 2328, 2539],  
7      [1613, 4088, 3991, 6461, 2691],  
8      [1560, 3392, 3826, 4787, 2613],  
9      [1608, 4802, 3932, 4477, 2705],  
10     [1576, 3933, 3909, 4979, 2685],  
11     [ 95, 229, 255, 496, 201],  
12     [ 2, 0, 1, 27, 0],  
13     [1438, 3785, 3589, 4174, 2215],  
14     [1342, 4043, 4009, 4665, 3033]],  
15     index=['05-01-11', '05-02-11', '05-03-11', '05-04-11', '05-05-11',  
16         '05-06-11', '05-07-11', '05-08-11', '05-09-11', '05-10-11'],  
17     columns=['R003', 'R004', 'R005', 'R006', 'R007'])  
18 )  
19 # 用index定义行名，用columns定义列名  
20  
21 # Change False to True for each block of code to see what it does  
22  
23 # DataFrame creation  
24 if 0:  
25     # You can create a DataFrame out of a dictionary mapping column name  
26     s to values  
27     df_1 = pd.DataFrame({'A': [0, 1, 2], 'B': [3, 4, 5]})  
28     print(df_1)  
29  
30     # You can also use a list of lists or a 2D NumPy array  
31     df_2 = pd.DataFrame([[0, 1, 2], [3, 4, 5]], columns=['A', 'B', 'C'])  
32     print(df_2)  
33  
34  
35 # Accessing elements  
36 if 0:  
37     print(ridership_df.iloc[0])  
38     print(ridership_df.loc['05-05-11'])  
39     print(ridership_df['R003'])  
40     print(ridership_df.iloc[1, 3])  
41     # .iloc[x]是按位置定位行  
42     # .loc['x']是按照名字定位行  
43     # [colname]是选择列  
44     # .iloc[x,y]是定位行和列（也就是1个元素了）  
45  
46 # Accessing multiple rows  
47 if 0:  
48     print(ridership_df.iloc[1:4])  
49     # [x:y]表示二维表中的一个范围  
50  
51 # Accessing multiple columns
```

```

52 if 0:
53     print (ridership_df[['R003', 'R005']])
54     # 此处是选择多列
55
56 # Pandas axis
57 if 0:
58     df = pd.DataFrame({'A': [0, 1, 2], 'B': [3, 4, 5]})
59     print (df.sum())
60     # 按照列做汇总
61     print (df.sum(axis=1))
62     # 按照行做汇总
63     print (df.values.sum())
64     # 把所有值做汇总
65
66 def mean_riders_for_max_station(ridership):
67     """
68         Fill in this function to find the station with the maximum riders on
69         the
70         first day, then return the mean riders per day for that station. Als
71         o
72         return the mean ridership overall for comparsion.
73
74     This is the same as a previous exercise, but this time the
75     input is a Pandas DataFrame rather than a 2D NumPy array.
76     """
77     overall_mean = ridership.values.mean()
78     # DataFrame使用.values.mean()计算所有的平均值
79     max_station = ridership_df.iloc[0].argmax()
80     # 此处和arrays不同，使用iloc[0].argmax()找出在0行最大的列值在那里
81     mean_for_max = ridership[max_station].mean()
82     # 再把这个列的所有值做平均，就能得出这个车站的平均值了
83
84     return (overall_mean, mean_for_max)
85
86 print(mean_riders_for_max_station(ridership_df))

```

## \*/ 7.将数据加载到 DataFrame 中

本节看一遍视频，回顾下：

- pd.readcsv('filename.csv')
- pd.head()
- pd.describe()

## \*/ 8.练习：计算相关性

本节内容在项目2里不要求，在项目4中会有涉及（有精力的细究）本节的简要说明：

- **相关性分析：** 数据分析的一个重要使用场景就是看两个参数之间有没有相关性。比如说同一个车站一周的人多少变化，和是否上班日有关系么？还是和是否下雨有关？又或者和温度有没有关系？
- **相关性分析工具：** 在统计学中有一个工具是计算两个相关性的叫做 Pearson's R。链接课程里面有。还记得前面计算 预期寿命 和 GDP 之间的关系时。通过计算，我们知道两个指标同向（都高于均值，或都低于均值）的有17个，而不同向的只有3个。这个就和R的这种方式类似。

```
1 | def correlation(x, y):  
2 |     std_x = (x - x.mean()) / x.std(ddof=0)  
3 |     std_y = (y - y.mean()) / y.std(ddof=0)  
4 |     # 将x, y做标准化（见之前的知识）  
5 |  
6 |     return (std_x * std_y).mean()  
7 |     # 返回r相关性指标
```

## \*/ 9.Pandas 轴名

本节是对DataFrame轴使用的扩展，除了使用 axis='index' 和 axis='column' 之外，也可以使用 axis=0 和 axis=1 来定义x轴或者y轴。了解即可。

## \*/ 10.练习： DataFrame向量化运算

这节里面有句亮点，我一定要放出来： 我觉得函数说明晦涩难懂，所以我决定直接试一下函数，以了解它的用处。因为用了dataframe的.shift()方法，只用一行就解决问题了！不用去写循环了！

```
1 | def get_hourly_entries_and_exits(entries_and_exits):  
2 |     return entries_and_exits - entries_and_exits.shift(1)  
3 |     # .shift()这个方法就是把所有行错后，只用一行代码就搞定了！！！  
4 |     # shift的文档和参数如下：http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.shift.html
```

## \*\*/ 11.练习： DataFrame applymap()

还记得numpy array的.apply()么， pandas dataframe有一个更加厉害的方法.applymap()， 练习代码解释如下：

```
1 def convert_grades(grades):
2     if grades >= 90:
3         return 'A'
4     elif grades >= 80:
5         grades = 'B'
6         return grades
7     #注意if和elif的写法，都是可以的，但第1种直接return的更加简洁
8     elif grades >= 70:
9         return 'C'
10    elif grades >= 60:
11        return 'D'
12    else:
13        return 'F'
14
15 print(grades_df.applymap(convert_grades))
16 # 如果不使用.apply, 而直接把df作为输入，就会报错：
17 # ValueError: The truth value of a DataFrame is ambiguous.
18 # 因为不知道要怎么处理df中的那么多数据，apply就明确的说每个都要处理
19 print(grades_df)
20 # 注意虽然上面做了转换但是没有写入到grades_df，所以打印df还是以前的值
```

## \*/ 12.13.练习： DataFrame apply()

此节选学。其实DataFrame也有.apply()方法， 和Series作用相同， 使用场景是那些使用.applymap()会对结果产生影响的情况（因为apply可以对指定列或行做操作， 不是像applymap那样对所有的元素）。

课程中对于ddof=0 的参数说明的非常详细：

计算得出的默认标准偏差类型在 numpy 的 .std() 和 pandas 的 .std() 函数之间是不同的。默认情况下， numpy 计算的是总体标准偏差， ddof = 0。另一方面， pandas 计算的是样本标准偏差， ddof = 1。如果我们知道所有的分数，那么我们就有了总体——因此，要使用 pandas 进行归一化处理，我们需要将“ddof”设置为 0。

练习13对于apply()做了扩展， apply(np.mean)， apply(np.max)可以作用于行或者列，并把结果存为一个Series（相当于从2维变成了1维）。

## \*/ 14.练习：向Series添加DataFrame

此节选学。可以作为向量化运算的扩展，练习完全是运行一遍哥哥if代码框就好了，不用自己写。

## \*/ 15.练习：再次归一化每一列

此节选学。挺复杂的一节，甚至可以放到毕业后复盘再回来看。引入了.sub().div()两个方法。

## \*\*\*/ 16.练习：Pandas groupby()

.groupby()方法的用处是根据需要把数据集拆分为子集，比如说练习中的是将整个数据拆分为周1到周7，一同7个子集。之后就可以得到周1到周7的每天均值，进行数据分析。

练习中的if False: 后面的代码框不会运行，当改成if True: 之后就会运行，这个Uda练习的一个特点，比较方便进行测试。另外if 0:和if 1:也是一样的效果。

```
1 filename = 'nyc-subway-weather.csv'  
2 #因为下载的csv文件名是-而不是_，将代码中的文件名改为上面  
3 #为了简化，将文件放到与本py文件一个目录就可执行  
4 subway_df = pd.read_csv(filename)  
5 print(subway_df.head())  
6 #使用.head()检查数据的前5行  
7 print(subway_df.groupby('day_week').mean())  
8 #根据'day_week'也就是一周7天，对数据集进行拆分，并求各个拆分组的均值（就是把所有数据  
9 按照周一到周日分成7份，并对每份求均值）。可以考察每周到底哪一天系统运作比较忙。  
10 print(subway_df.groupby('day_week').mean()[['ENTRIESn_hourly']])  
#对上面的输出中的['ENTRIESn_hourly']内容单独展示
```

## \*\*\*/ 17.练习：每小时入站和出站数

本部分为本周最后的必学部分，就是把这周的东西放在一起，最后输出每小时的初入站人数。里面的内容前面分步都接触过，整合起来，成就感有没有！

```
1 import numpy as np  
2 import pandas as pd  
3  
4 values = np.array([1, 3, 2, 4, 1, 6, 4])  
5 example_df = pd.DataFrame({
```

```

6     'value': values,
7     'even': values % 2 == 0,
8     'above_three': values > 3
9 }, index=['a', 'b', 'c', 'd', 'e', 'f', 'g'])
10
11 #print(example_df)
12
13 # Change False to True for each block of code to see what it does
14
15 # Standardize each group
16 if 0:
17     def standardize(xs):
18         return (xs - xs.mean()) / xs.std()
19     grouped_data = example_df.groupby('even')
20     print(grouped_data)
21     print()
22     #只打印groupby的输出是一个提示，因为没有指定要打印分组之后的什么东西
23     print(grouped_data['value'].apply(standardize))
24     #使用.apply将grouped_data分组后的value的值进行标准化计算
25
26 # Find second largest value in each group
27 if 1:
28     def second_largest(xs):
29         sorted_xs = xs.sort_values(inplace=False, ascending=False)
30         #如果报错'Series' object has no attribute 'sort'
31         #是因为pandas版本已经比实例中的高
32         #将sort改为sort_values
33         return sorted_xs.iloc[1]
34         # 先对输出排序，再使用.iloc[1]输出排序后的第二个元素
35     grouped_data = example_df.groupby('even')
36     print(grouped_data['value'].apply(second_largest))
37
38 # --- Quiz ---
39 # DataFrame with cumulative entries and exits for multiple stations
40 ridership_df = pd.DataFrame({
41     'UNIT': ['R051', 'R079', 'R051', 'R079', 'R051', 'R079', 'R051', 'R0
42 79', 'R051'],
43     'TIMEn': ['00:00:00', '02:00:00', '04:00:00', '06:00:00', '08:00:00'
44 , '10:00:00', '12:00:00', '14:00:00', '16:00:00'],
45     'ENTRIESn': [3144312, 8936644, 3144335, 8936658, 3144353, 8936687, 3
46 144424, 8936819, 3144594],
47     'EXITSn': [1088151, 13755385, 1088159, 13755393, 1088177, 13755598
48 , 1088231, 13756191, 1088275]
49 })
50
51 def get_hourly_entries_and_exits(entries_and_exits):

```

```
52     return entries_and_exits - entries_and_exits.shift(1)
53     # 使用之前的shift方法
54
55     print(ridership_df.groupby('UNIT')[['ENTRIESn', 'EXITSn']].apply(get_hourly_entries_and_exits))
56     # 按照UNIT进行分组，并调用函数计算每小时的值
57     # 注意如果不指定对['ENTRIESn', 'EXITSn']做操作的话会报错
58     # 因为其他列包含数值，使用.apply()会报错的
```

## \*/ 18.练习：合并Pandas DataFrame

此节选学。合并的相关操作，引入了.merge()方法，可以遇到再研究。

## \*/ 19.练习：使用DataFrame绘制图形

此节选学。项目2中对图形不要求。

## \*/ 20.三维数组

此节不学！！！ 这节是谁写的，出来，我保证不打死你。用不到，而且链接中的Panel也要被丢弃了。这节就一个目的：吓唬你！ 实际上你的任务已经完成。

# /彩蛋/

- 项目2中的py3版本文件（有的练习是py2环境的，当本地做遇到问题可以参考）命名规则  
lx\_y\_z.py(x是课程号，y是小节号，z是小节内容简称。[/下载链接/](#)
- 官方数据结构说明：<http://pandas.pydata.org/pandas-docs/stable/dsintro.html#panel>
- 官方DataFrame说明（也有所有方法的汇总，超级有用）：<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html>
- 控制流总结：[https://bop.mol.uno/09.control\\_flow.html](https://bop.mol.uno/09.control_flow.html)

Title: W3 Plus Dataframe简介

Tags: 数据分析, 初级

# W3 Plus Dataframe简介

---

## / dataframe的由来

Python中的dataframe数据结构，是pandas第三方库中的提供的一种数据结构。超详细的官方文档：<http://pandas.pydata.org/pandas-docs/stable/merging.html#database-style-dataframe-joining-merging>

## / 为什么要用dataframe这种数据结构

- 结合了python list的序列特性，和dictionary的键-值对应特性，更加灵活
- pandas使用Cpython完成，速度更快
- pandas序列中的每一列都可以是不同的数据类型（numpy的array所有列必须相同）
- 有众多的方法可以计算统计数字，比如：.mean()计算平均数 .max()计算最大值 .min()计算最小值 .mode()计算众数
- 对于dataframe数据，有很多方便的检索信息比如：.info()看整体信息 .head()和.tail()看数据头尾信息 .describe()看描述信息
- 可以设置列名和行名（想想看你excel中的表单，没有列名和行名将是多么的痛苦），numpy的array就不可以的
- 拥有丰富读写功能，比如读入csv文件，而且同样是Cpython的原因，所以会快很多
- pandas dataframe的结构和R语言中的dataframe（高级课程有学呦）结构类似，学一个送半个，要不要了解一下
- 据说pypy因为使用了jit技术，比cpython还快5倍（有空要了解一下）

简单的说：datafram能处理复杂的数据类型、具备众多数据分析和统计的内置方法、并且处理速度超快！那么我们不用他用谁捏？

## / dataframe入门

## // dataframe啥样子

- week3导学中的：{6.练习：访问DataFrame元素}
- spyder展示

## / dataframe.applymap()介绍

- week3导学中的：{11.练习：DataFrame applymap()}

## / dataframe使用

---

### // dataframe简介

比较全面的介绍：

- <https://www.infoworld.com/article/3257599/analytics/introducing-pandas-dataframe-for-python-data-analysis.html>
- <https://www.infoworld.com/article/3257599/analytics/introducing-pandas-dataframe-for-python-data-analysis.html?nsdr=true&page=2>
- <https://www.infoworld.com/article/3257599/analytics/introducing-pandas-dataframe-for-python-data-analysis.html?nsdr=true&page=3>

### // 官方文档

官方文档是最全面的一手信息，要慢慢培养看的感觉，虽然太多会比较蒙！

<http://pandas.pydata.org/pandas-docs/stable/merging.html#database-style-dataframe-joining-merging>

### // pandas里的众多方法e.g.

to\_datetime 将数据格式转换成时间格式，之后可以进行各种操作的呦！官方文档：<http://pandas.pydata.org/pandas-docs/stable/merging.html#database-style-dataframe-joining-merging>

### // pandas里导入的库（来自小艾神助攻）

- python的import一般就是3类，python标准库，开源库，还有自己编写的模块
- 标准库可以参考 <https://docs.python.org/3/library/index.html>
- 第三方库可以参考 <https://github.com/jobbole/awesome-python-cn/>
- 自己可以编写的模块可以参考 你自己，就是把你要用的一堆函数和文件打包，就叫模块了



Title: 项目实战 [项目: 探索共享单车用户行为规律3/4]

Tags: 数据分析初级, 实战项目

# W3 项目实战 [项目: 探索共享单车用户行为规律3/4]

---

- [W3 项目实战 \[项目: 探索共享单车用户行为规律3/4\]](#)
- [/学习地图/](#)
  - [/ 项目路径](#)
  - [/ 项目推进](#)
- [/目标1/: 分析项目模板文件](#)
  - [/ 1.拆分项目功能](#)
  - [/ 2.分析项目模板文件](#)
- [/目标2/: 完成分模块练习](#)
  - [/ 5.6.计算最受欢迎的开始时间](#)
  - [/ 7.8.显示用户类型细分](#)
  - [/ 9.10.数据加载与过滤](#)
- [/目标3/: 按照函数完成项目文件](#)
  - [/ 2.get\\_filters\(\) 函数](#)
  - [/ 3.load\\_data\(city, month, day\) 函数](#)
  - [/ 4.time\\_stats\(df\) 函数](#)
  - [/ 5.station\\_stats\(df\) 函数](#)
  - [/ 6.trip\\_duration\\_stats\(df\) 函数](#)
  - [/ 7 usr\\_stats\(df\) 函数](#)

## /学习地图/

---

同学们，项目2已经到了第3周，前两周我们学习了python的基础知识和dataframe数据结构。这周我们将开始准备完成项目文件，我们将会在这周过一遍项目中的难点，讲解项目中的机构，希望同学们能通过本周学习尽快提交项目（如果没能提交也不用太着急，下周是回顾、扩展和修改项目的时间）。

## / 项目路径

---

项目名称：探索共享单车用户行为规律

项目时间：4周

- 第1周：Python基础内容 - Python基础（数据类型和运算符、控制流、函数、脚本编写）
- 第2周：Python数据处理内容 - Python数据分析（Numpy、Pandas库）
- **第3周：完成项目 - 项目：探索美国共享单车数据**
- 第4周：通过项目 - 修改项目、查缺补漏、整理笔记

## / 项目推进

---

**Week1要求（已完成）：**

- 完成探索美国共享单车数据项目的1-3节内容
- 搭建本地anaconda环境（Python3版本）确保Spyder可以使用
- 下载bikeshare-new-2.zip项目文件。如果教室里面不能下载，请下载：[/我搬的砖/](#)

**Week2要求（已完成）：**

- 能够打开项目文件（用Uda线上的可以，但建议可以试试用Spyder本地打开项目文件浏览）
- 了解项目文件中有几个函数，函数名和输入是什么（其他的看不懂没关系）

**Week3要求（肝！少年！）：**

- 浏览本导学文件
- 做完项目中的练习
- 根据导学文件，按照函数一个个，一步步的完成项目
- 提交最好了！

## / 目标1/：分析项目模板文件

---

### / 1.拆分项目功能

---

要完成项目，不要一下就掉到代码中去，我们首要明白的是我们要解决什么问题，我们有什么数据，我们怎么拆分实现不同的功能。

项目的说明页面非常详细：[/项目说明页面/](#)

数据文件如下：

- 我们有3个城市的共享单车数据（由Motivate提供）
- 数据集包括以下内容：

起始时间 Start Time (例如 2017-01-01 00:07:57)

结束时间 End Time (例如 2017-01-01 00:20:53)

骑行时长 Trip Duration (例如 776 秒)

起始车站 Start Station (例如百老汇街和巴里大道)

结束车站 End Station (例如塞奇威克街和北大道)

用户类型 User Type (订阅者 Subscriber/Registered 或客户 Customer/Casual)

性别 Gender (只有纽约和芝加哥数据有)

出生年份 Birth Year(只有纽约和芝加哥数据有)

## 项目目标（问题）如下：

- 通过分析收到的数据文件，得出共享单车运行数据中的趋势，辅助进行商业决策
- 本次分析过程基于收到的数据，属于探索性的分析阶段，目的在于根据不同特征的共享单车的趋势数据
- 本次分析不涉及分析两个变量中的相关关系（项目4就有了，不要着急学霸们）
- 根据项目模版，需要回答一下问题：

起始时间 (Start Time 列) 中哪个月份最常见？

起始时间中，一周的哪一天（比如 Monday, Tuesday）最常见？ 提示：可以使用 `datetime.weekday()` (点击查看文档)

起始时间中，一天当中哪个小时最常见？

总骑行时长 (Trip Duration) 是多久，平均骑行时长是多久？

哪个起始车站 (Start Station) 最热门，哪个结束车站 (End Station) 最热门？

哪一趟行程最热门（即，哪一个起始站点与结束站点的组合最热门）？

每种用户类型有多少人？

每种性别有多少人？

出生年份最早的是哪一年、最晚的是哪一年，最常见的是哪一年？

## 如何完成项目：

- 首先看下3、11、12节的完成指南
- 接下来要决定是在13节项目空间完成并提交，还是下载文件到本地再在15节提交（需要本地环境）！**注意2选1，只在一个地方提交就可以了！**
- 如果是在13节处完成项目，请参看14节的说明
- 如果是本地完成，下载的数据和模版文件链接（2018年7月版本）：<https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project2/bikeshare-new-2.zip>
- 第4节是检查数据的基本语句，选修！我们再项目4再讲相关内容。

## / 2. 分析项目模板文件

对于一个工程文件（很大很长的代码文件），可能有上千行代码，如果从头看起来会让人不知所措，实际的代码文件处理起来是这样的（此处由小ai的神助补充）：

其实程序员这个工种，写代码的时间应该是不多的，从头开始写一个项目的情况也很少见，大部分时间是在看别人写的代码，然后在里面加一点点新的东西。所以看代码的能力是很重要的。

看代码的第一步其实就是找到程序的入口。其实在大多数语言里面，都是main()这个函数，这也是为什么它叫main的原因。

这个项目只有一个文件，你会有时间读完它。但是如果给你1000万行的一个项目，你肯定不能一个文件一个文件来读了。所以咱们都是要找到一个程序的主入口。先看看咱们这个程序的功能是啥，这样就抓住了主线。

我记得做考研英语阅读的时候也是，我一般都先看最后一段，然后看每一段的第一句，这样就抓住了文章的主旨，不会因为看不懂几个句子和单子就蒙圈了。

读代码也是，从main函数一路看下去，每个函数是什么功能，从它的名字就可以看个大概，了解程序的主要功能和流程。这样你就不会因为看不懂某个函数的实现而陷入迷茫。

一般程序里面只有main函数里面可能会写一个死循环，其他地方最好还是不要写while true，万一忘记break，程序就挂死了

总结下来就是：读代码能力非常重要，从main函数开始一路看下去，可以避免看不懂某个函数而陷入迷茫（看来看不懂别人的函数也是会发生啊）。

```
1 - /1/ prepare block
2 -- imports # 导入需要第三方库
3 -- CITY_DATA # 定义CITY_DATA包括3个城市的文件
4 --- # CITY_DATA包括3城市chicago, new york city, washington
5 --- # 采用字典结构存储数据chicago': 'chicago.csv'
6
7 - /2T/ def get_filters()
8 -- # 函数输入参数为空
9 -- # 此函数是和用户互动得到数据筛选输入的
10 -- # 通过input方式请用户输入要求
11 --- /T2.1/ # 那个城市?
12 --- /T2.2/ # 那个月份(只有1-6月)?
13 --- /T2.3/ # 周几?
14
15 - /3T/ def load_data(city, month, day)
```

```
16 -- # 函数输入为city, month, day
17 -- # 此处没有todo但是要写加载数据
18 --- /T3.1/ 加载数据 (和练习一毛一样的)
19 -- # return df 所以要结合别的调用使用
20
21 - /4T/ def time_stats(df)
22 -- # 函数输入为df (过滤后的数据)
23 -- # 使用print通知用户进行将要完成的事情
24 -- # 计算不同颗粒度时间内容最繁忙时段
25 --- /T4.1/ #月份
26 --- /T4.2/ #日
27 --- /T4.3/ #小时
28 -- # 使用 time函数报告运行所用时间
29
30 - /5T/ def station_stats(df)
31 -- # 函数输入为df (过滤后的数据)
32 -- # 使用print通知用户进行将要完成的事情
33 -- # 计算不同内容最繁忙车站
34 --- /T5.1/ #开始
35 --- /T5.2/ #结束
36 --- /T5.3/ #综合考虑
37 -- # 使用 time函数报告运行所用时间
38
39 - /6T/ def trip_duration_stats(df)
40 -- # 函数输入为df (过滤后的数据)
41 -- # 使用print通知用户进行将要完成的事情
42 -- # 计算行程数据
43 --- /T6.1/ #总旅行时间
44 --- /T6.2/ #平均旅行时间
45 -- # 使用 time函数报告运行所用时间
46
47 - /7T/ def user_stats(df)
48 -- # 函数输入为df (过滤后的数据)
49 -- # 使用print通知用户进行将要完成的事情
50 -- # 计算用户数据
51 --- /T7.1/ #用户数计算
52 --- /T7.2/ #用户性别计算
53 --- /T7.3/ #用户其他信息计算
54 -- # 使用 time函数报告运行所用时间
55
56 - /8-FirstLook! / def main()
57 -- # 主函数是调用其他函数的主程序
58 -- # 不用编辑但逻辑要弄通
59 -- # while True 是一种在测试时常用的方式
60 --- # 当不想执行的时候改为 while False
61 --- # 1和0等价
```

```
62 --- # 此处可以删掉，也可不管
63 -- # 此处以下是通过调用6个函数完成所有功能
64 --- city, month, day = get_filters() # 赋值
65 --- df = load_data(city, month, day) # 生成df数据
66 --- time_stats # 处理时间数据
67 --- station_stats(df) # 处理车站数据
68 --- trip_duration_stats(df) # 处理旅程数据
69 --- user_stats(df) # 处理旅客数据
70 -- # 通过restart再运行一遍程序功能
71
72 - /9/ if __name__ == "__main__":
73 -- # 库文件标准结尾
74 -- # 能够保证本文件作为第三方库的被导入时只运行要求的函数
75 -- # 详细请参见week2导学后面扩展内容
```

根据以上拆分，我们发现其实要想完成项目，我们只用把2、4、5、6、7部分完成就可以了（在数字编号后面有个T的部分）。本导学所有内容将会按照以上项目代码框进行讲解。其中实现方法只是讲解一种可选方式，请同学们理解后自己完成。

## /目标2/：完成分模块练习

在完成项目之前，Uda非常贴心的把要用到的3个重点做了3组讲解和练习，对应5-10节我们来讲一下。这块也不白学，3个练习对应的内容调整一点点就可以写进项目文件中了。

注意：本节练习和项目全部使用Pandas Dataframe数据结构完成！

### / 5.6.计算最受欢迎的开始时间

本代码基于Uda课程中的工作区完成。

```
1 import pandas as pd
2
3 filename = 'chicago.csv'
4
5 # load data file into a dataframe
6 df = pd.read_csv(filename)
7 # 使用.read_csv()方法读入数据
8
9 # 首先我们读入的Start Time这一列默认是字符格式
10 # 我们使用type检查Start Time列的第一个元素
11 # 可以看到转换后的类型是<class 'str'>
12 print('original data type:')
```

```

13 print(type(df['Start Time'][0]))
14
15 # 为了能够计算时间要变换为pandas里面的时间格式
16 # 就是使用.to_datetime()这个方法转换格式
17 df['Start Time'] = pd.to_datetime(df['Start Time'])
18 # 下面再次检查类型输出变为 <class 'pandas.tslib.Timestamp'>
19 print('\nconverted data type:')
20 print(type(df['Start Time'][0]))
21
22 # 接下来对时间分析就可以随心所欲了
23 # 比如把Start Time的小时数拆出来, 存为新的列
24 # 方式就是在需要处理的数据后面加.dt.hour
25 df['hour'] = df['Start Time'].dt.hour
26 # 两个的区别如下
27 print('\noriginal time: ')
28 print(df['Start Time'].head())
29 print('\nonly hour: ')
30 print(df['hour'].head())
31
32 # 之后我们就可以使用.mode()查看众数 (出现最多的数是什么样子了)
33 popular_hour = df['hour'].mode()
34 print('\nMost Popular Start Hour:', popular_hour)
35 # 结果的输出是这样的:
36 # Most Popular Start Hour: 0      17
37 # dtype: int64
38 # 意思是众数发现了一个(可以并列的)众数
39 # 第0索引的众数是17
40 # 数据类型是整数
41
42 # 太丑了对不对, 所以我们直接要求把这第0个位置的结果显示出来
43 popular_hour = df['hour'].mode()[0]
44 print('\n(pretty) Most Popular Start Hour:', popular_hour)

```

输出是这个样子滴:

```

1 original data type:
2 <class 'str'>
3
4 converted data type:
5 <class 'pandas.tslib.Timestamp'>
6
7 original time:
8 0 2017-05-29 18:36:27
9 1 2017-06-12 19:00:33
10 2 2017-02-13 17:02:02

```

```
11 | 3 2017-04-24 18:39:45
12 | 4 2017-01-26 15:36:07
13 | Name: Start Time, dtype: datetime64[ns]
14 |
15 | only hour:
16 | 0 18
17 | 1 19
18 | 2 17
19 | 3 18
20 | 4 15
21 | Name: hour, dtype: int64
22 |
23 | Most Popular Start Hour: 0      17
24 | dtype: int64
25 |
26 | (pretty) Most Popular Start Hour: 17
```

## / 7.8.显示用户类型细分

本代码基于Uda课程中的工作区完成。

```
1 import pandas as pd
2
3 filename = 'chicago.csv'
4
5 df = pd.read_csv(filename)
6
7 # 就是使用.value_counts()对一列的值做统计输出
8 user_types = df['User Type'].value_counts()
9
10 print(user_types)
```

输出是这样的，可以看到User Type这列一共有两种值，和每种值的个数：

```
1 Subscriber    330
2 Customer      70
3 Name: User Type, dtype: int64
```

## / 9.10.数据加载与过滤

这块其实是把5.6.节的东西运用到项目文件中。

```
1 import pandas as pd
2
3 CITY_DATA = { 'chicago': 'chicago.csv',
4               'new york city': 'new_york_city.csv',
5               'washington': 'washington.csv' }
6 # 这是文件列表，根据不同的输入选择不同的文件
7
8 def load_data(city, month, day):
9     # 定义函数，根据用户输入的3个变量处理数据
10    # city是要处理那个城市
11    # month、day是要处理的月份和天输入
12
13    df = pd.read_csv(CITY_DATA[city])
14    # 根据输入的city读入csv文件
15
16    df['Start Time'] = pd.to_datetime(df['Start Time'])
17    # 将时间从字符格式转为datetime格式
18
19    df['month'] = df['Start Time'].dt.month
20    df['day_of_week'] = df['Start Time'].dt.weekday_name
21    # 将month和weekday抽离出来建立新的列
22    # 是为了后面根据输入进行数据筛选时候的参考
23
24    if month != 'all':
25        # 如果月份要求不是all，则按照要求筛选数据
26
27        months = ['january', 'february', 'march', 'april', 'may', 'june']
28    ]
29        # 这是可选的所有月份
30    month = months.index(month) + 1
31        # 定义month变量等于输入月份的索引加1
32        # 为什么要加1呢？
33        # 因为索引是从0开始，用户输入是从1开始
34
35    df = df[df['month'] == month]
36        # 最后对df做更新
37        # 右边df['month'] == month是一种过滤
38        # 左边是month列的所有内容（1-6月）
39        # 右边是刚刚生成的要筛选的month变量（由用户输入）
40        # 中间是判断 ==
41        # 结果就是满足右边条件的数据被重新写入df里
42
43    if day != 'all':
44        # week的筛选方式同上
45        df = df[df['day_of_week'] == day.title()]
```

```
46  
47     return df  
      #最终返回筛选过的数据
```

## /目标3/：按照函数完成项目文件

根据上一节的模版文件分析，我们只用完成2、3、4、5、6、7部分的填空题就好了，在这之前，我们分析下最后8的main文件是怎么把这几部分串接起来使用的：

```
1 def main():  
2     while True:  
3         # 当条件为真时，执行。因为是True所以总是执行这个循环  
4         # 除非遇到break才会停止  
5         # 本段代码最后有个是否要再分析一遍的提问  
6         # 如果回答的不是yes，将会终止循环  
7             city, month, day = get_filters()  
8             # 调用函数，得到city, month, day的输入  
9             df = load_data(city, month, day)  
10            # 根据输入的内容对数据进行过滤，生成df函数  
11            time_stats(df)  
12            # 调用函数，对时间进行处理  
13            station_stats(df)  
14            # 调用函数，对车站进行处理  
15            trip_duration_stats(df)  
16            # 调用函数，对旅程进行处理  
17            user_stats(df)  
18            # 调用函数，对旅程进行处理  
19  
20            restart = input('\nWould you like to restart? Enter yes or no.\n')  
21        )  
22        if restart.lower() != 'yes':  
23            break  
24        # 此处的仪式是如果yes再循环一遍  
        # 如果不是，则结束
```

大概的输出是这样的，注意 > 后面都是我输入的：

```
1 Hello! Let's explore some US bikeshare data!  
2  
3 q1/3: which city do you want to know? chicago,  
4          york city or washington? > chicago  
5  
new
```

```
6 | q2/3: which month do you want to know? choose from all,  
7 | january, february, ... , june. > january
```

```
q3/3: which day do you want to know? choose from all,  
monday, ... , sunday. > all
```

## / 2.get\_filters() 函数

这个函数的作用就是让用户输入city, month, day3个参数，在输入的时候要注意判断是否输入有效。在项目模版中的docstring（就是开始“”包围起来的内容）解释的很详细（为了便于理解，官方模版的注释都已经删除，请结合两个文件一起学习）：

```
1 | def get_filters():  
2 |  
3 |     print('Hello! Let\'s explore some US bikeshare data!')  
4 |     # 先显示一行输出，让用户知道在做什么任务  
5 |     # TO DO: get user input for city (chicago, new york city, washington  
6 | ). HINT: Use a while loop to handle invalid inputs  
7 |  
8 |     city = input('q1/3: which city do you want to know? chicago, new yor  
9 | k city or washington? > ')  
10 |    # 第一个是获得city的输入，输入可选在模版提示中  
11 |    # 使用city = inpyt(xxx)，注意xxx是在输入时提示的语句  
12 |  
13 |    month = input('q2/3: which month do you want to know? choose from al  
14 | l, january, february, ... , june. > ')  
15 |    # 同样的获得month输入，输入可选在模版提示中  
16 |  
17 |    day = input('q3/3: which day do you want to know? choose from all, m  
18 | onday, tuesday, ... sunday. > ')  
19 |    # 同样的获得day输入，输入可选在模版提示中  
20 |  
21 |    print('-'*40)  
# 打印一行 - - - - 作为分隔  
# *40的意思就是把前面的家伙重复40遍  
return city, month, day  
# 返回city, month, day 3个参数就达到目标了
```

那么我们搞完了么，没呢！大家注意到提示里面有这么一句了么：HINT: Use a while loop to handle invalid inputs。也是就是说如果用户输入的不对，我们就没法筛选数据，也就没办法完成任务。所以我们要加一块对用户输入做判断的。那么我们有3个参数要让用户输入，这里有两种方式进行判断：

- 方式1:3个参数都输入完成了，总体做判断，如果那个是错的，就都要求重新输入。好处是代码简单，效率高点。坏处是用户麻烦，不小心输错了1个还要都重来一遍。
- 方式2:每输入一个参数就进行判断，如果输入错了就赶紧重新输入这1个参数。好处是用户方便些，坏处是代码复杂一点点。
- 选择：都可以的，我个人喜欢方式2，因为用户方便为优先。

那么我们怎么来写这个循环呢，按照提示我们可以把3处输入包在while循环中，如果输入的是合法选项，我们就pass，如果是错的就打回去重新输入。另外需要之处py3就使用inpyt就行了，这个和py2中的raw\_input是一样的。使用月份筛选的方法做个例子(实现1)：

```

1 def get_filters():
2
3     mon_option = ['all', 'january', 'february', 'march', 'april', 'may',
4     'june']
5     # 先是建立备选名单
6
7     month = ''
8     # 给变量定义为空
9     # 因为第一次要比较，所以要先赋值再使用
10
11    while month not in mon_option:
12        # 使用while循环，如果变量month没在备选名单mon_option里的话就执行
13        # 相当于输入不对就一直循下面的inpyt让用户输入
14        # 第一次的时候month是空值，所以也会执行
15        month = input('q2/3: which month do you want to know? choose from al
l, january, february, ... , june. > ')
            # 赋值语句，输入提示尽量人性化便于用户理解

```

这种先给month赋值为空的方式有点啰嗦，像我这样一个有要求的男人是不会这样妥协的，于是有了（实现2）：

```

1 while True:
2     # 总是循环执行，直到遇到break
3     month = input('q2/3: which month do you want to know? choose from al
4     l, \
5                     january, february, ... , june. > ')
6     # \ 是代码换行的意思，python的编程规范是每行不超过79个字符
7     if month in mon_option:
8         break
9     # 当经过判断 month 变量在可选范围内，就break结束while循环

```

方法2的问题是while True这种循环判断方式总是成立，但是当你知道自己在写什么的时候就没有问题了。所以无论是那种方式都是编程风格的选择，都是可以的。这里请大家按照例子把自己的

def\_filters完成。

## / 3.load\_data(city, month, day) 函数

在上一节，我们收集到了用户的输入，就可以将原始数据文件按照用户的要求。大家发现了没有，这里的代码和 **目标2 {9.10.数据加载与过滤}** 是一模一样的。要注意回顾一下的是这个函数在主函数main()中是怎么调用的就好了：

```
1 | city, month, day = get_filters()
2 | # 调用函数，得到city, month, day的输入
3 | df = load_data(city, month, day)
4 | # 将这3个变量输入到load_data函数，得到我们需要的数据df
```

## / 4.time\_stats(df) 函数

大家注意到了小括弧中的df没，这说明这个函数是要输入的，而这个输入就是df这个数据集，就是上一节在main中调用得出的。接下来我们先对时间做处理，主要的内容也是和 **目标2：{5.6.计算最受欢迎的开始时间}** 是一样的：

```
1 | def time_stats(df):
2 |     print('\nCalculating The Most Frequent Times of Travel...\n')
3 |     # 先是打印一行提示
4 |     start_time = time.time()
5 |     # 此处可以忽略，是调用time函数用于计算代码的执行时间
6 |
7 |     # 为了能够计算时间要变换为pandas里面的时间格式
8 |     # 就是使用.to_datetime()这个方法转换格式
9 |     df['Start Time'] = pd.to_datetime(df['Start Time'])
10 |
11    # TO DO: display the most common month
12    df['month'] = df['Start Time'].dt.month
13    max_month = df['month'].mode()[0]
14
15    # TO DO: display the most common day of week
16    df['day'] = df['Start Time'].dt.day
17    max_day = df['day'].mode()[0]
18
19    # TO DO: display the most common start hour
20    df['hour'] = df['Start Time'].dt.hour
21    max_hour = df['hour'].mode()[0]
```

```
23     print("\nThis took %s seconds." % (time.time() - start_time))
24 # 后面的time.time() - start_time就是用当前时间剪去开始运行时的时间
25 # 可以看出这段代码运行了多长时间
26 print('-'*40)
27 print('max freq month is:')
28 print(max_month)
29 print('max freq day is:')
30 print(max_day)
31 print('max freq hour is:')
32 print(max_hour)
33 #以上是3个频率的输出
```

当运行后，main()函数会调用time\_stats(df)函数，输出是这个样子的（因为有使用了time方法，所以会输出执行时间）：

```
1 This took 0.04229283332824707 seconds.
2 -----
3 max freq month is:
4 1
5 max freq day is:
6 21
7 max freq hour is:
8 17
```

## / 5.station\_stats(df)函数

与4节大同小异，请同学们自己完成。提示如下：

```
1 # 问题3是问那个车站组合出现最多
2     # 为了回答这个问题，我们可以在数据新建1列
3     # 例子命名为combine_sation
4     # 由Start Station的数据加上End Station合并
5     # 为了便于识别，我加上了 --- 作为标签
6     # 再之后就一样是使用mode()进行统计了
7     df['combine_station'] = df['Start Station'] + ' --- ' + df['End Stat
8     ion']
      max_combine = df['combine_station'].mode()[0]
```

输出是这个样子的：

```
1 max freq start station is:
2 Streeter Dr & Grand Ave
```

```
3 max freq end station is:  
4 Streeter Dr & Grand Ave  
5  
6 max freq combine statuion is:  
7 Lake Shore Dr & Monroe St --- Streeter Dr & Grand Ave
```

## / 6.trip\_duration\_stats(df)函数

与4节大同小异，请同学们自己完成。提示如下：

```
1 # 使用.sum()方法求和  
2     total_trip_time = df['Trip Duration'].sum()  
3  
4 # 使用.mean()方法求平均值  
5     mean_trip_time = df['Trip Duration'].mean()
```

输出是这样的：

```
1 total trip time is:  
2 280871787  
3 mean trip time is:  
4 936.23929
```

## / 7 usr\_stats(df)函数

这部分与{7.8.显示用户类型细分}方法相同，还扩展了一个性别的统计。之后使用min、max统计了最大最小值，提示如下：

```
1 # 使用.value_counts方法进行统计  
2     user_types = df['User Type'].value_counts()  
3  
4 # 同样使用.value_counts方法进行统计  
5     gender_types = df['Gender'].value_counts()  
6  
7 # 计算最大最小值使用的是.min方法和.max方法  
8     earliest = df['Birth Year'].min()  
9     recent = df['Birth Year'].max()  
10    common = df['Birth Year'].mode()[0]
```

输出是这样的：

```
1 Calculating User Stats...
2
3 user types is:
4 Subscriber    238889
5 Customer      61110
6 Dependent      1
7 Name: User Type, dtype: int64
8
9 gender types is:
10 Male        181190
11 Female       57758
12 Name: Gender, dtype: int64
13
14 recent year of birth is:
15 1899.0
16
17 recent year of birth is:
18 2016.0
19
20 common year of birth is:
21 1989.0
```

Title: 项目实战 [项目: 探索共享单车用户行为规律4/4]

Tags: 数据分析初级, 实战项目

# W3 项目实战 [项目: 探索共享单车用户行为规律4/4]

---

- W3 项目实战 [项目: 探索共享单车用户行为规律4/4]
- /学习地图/
  - / 项目路径
  - / 项目推进
  - / 重点提示
- /目标1/: 项目整体回顾
  - / 2.项目分析思路
  - / 3.项目解答路径
    - // 能力1:Python基础
    - // 能力2:针对项目的练习
    - // 能力3:完成项目
  - // 4.数据分析工具汇总
- /目标2/: 知识点梳理
  - / Week1的必学模块
  - / Week2的必学模块
- /目标3/: 项目文件优化 (看懂即可)
  - / 1.gender数据不完整
  - / 2.输出优化
  - / 3.使用循环获得用户输入
  - / 4.输出年为整数
  - / 5.允许输入大写字母

## /学习地图/

---

同学们, 恭喜大家, 这周是项目2的最后一周, 经过前3周的努力, 同学们应该都在努力的提交项目中吧。本周我们将对项目2的重点知识做回顾和扩展。总结好项目2, 并为下周项目3开始打好基础。

# / 项目路径

---

项目名称：探索共享单车用户行为规律

项目时间：4周

- 第1周：Python基础内容 - Python基础（数据类型和运算符、控制流、函数、脚本编写）
- 第2周：Python数据处理内容 - Python数据分析（Numpy、Pandas库）
- 第3周：完成项目 - 项目：探索美国共享单车数据
- 第4周：通过项目 - 修改项目、查缺补漏、整理笔记

# / 项目推进

---

**Week1要求（已完成）：**

- 完成探索美国共享单车数据项目的1-3节内容
- 搭建本地anaconda环境（Python3版本）确保Spyder可以使用
- 下载bikeshare-new-2.zip项目文件。如果教室里面不能下载，请下载：[/我搬的砖/](#)

**Week2要求（已完成）：**

- 能够打开项目文件（用Uda线上的可以，但建议可以试试用Spyder本地打开项目文件浏览）
- 了解项目文件中有几个函数，函数名和输入是什么（其他的看不懂没关系）

**Week3要求（已完成）：**

- 浏览本导学文件
- 做完项目中的练习
- 根据导学文件，按照函数一个个，一步步的完成项目
- 提交最好了！

**Week4要求（本周任务）：**

- 项目通关
- 知识点回顾
- 整理笔记

# / 重点提示

---

- 本周任务是项目通过！所以要尽早提交，因为评审老师会审阅反馈的！
- 目标1、目标2、目标3对应的是 1) 整体项目回顾； 2) 知识点梳理； 3) 项目文件优化（选

学)

- 请认真看本导读文件，如果同学是编程新手的话，一定记得多看、多试、多问，大家一起加油

# /目标1/：项目整体回顾

---

## / 2.项目分析思路

---

要完成项目，不要一下就掉到代码中去，我们首要明白的是我们要解决什么问题，我们有什么数据，我们怎么拆分实现不同的功能。

接下来是查看项目文件，先从最后的 main() 函数看整个文件是怎么组织的，再根据里面调用的方式去看需要你处理的文件，因为在实际的工程中的是有很多合作的，读大量代码并且在你负责的地方写代码是一般的工作场景。详细的可以参见我们：[/小艾的神助攻解释/](#)

由于项目文件不是十分长，在上面的链接那节后面还为大家分析了下整个文件的结构，了解了项目中调用的函数便于后面逐条完成项目目标。

## / 3.项目解答路径

---

总体来讲，要完成项目可以按照下面的3个步骤进行推进：

### // 能力1:Python基础

- 本部分主要是通过 Project2 week1, Project week2的基础内容的学习了解所需的数据编程知识
- week1导学链接：<https://github.com/mengfanchun2017/DAND-Basic/blob/master/week1-guide.md>
- week2导学链接：<https://github.com/mengfanchun2017/DAND-Basic/blob/master/week2-guide.md>

### // 能力2:针对项目的练习

- 首先了解项目，除了项目的描述页面之外还可以自己查查资料，提高兴趣，共享单车这个事在中国还是很熟悉的
- 之前去米国旅游，30分钟就要大改7.5刀啊，小黄、小蓝和小橙相当的实惠啊
- 当然也有些爆料的，比如这篇：[https://m.sohu.com/a/240046942\\_161623/?pvid=000115\\_](https://m.sohu.com/a/240046942_161623/?pvid=000115_)

3w\_a (ps: 谁要知道不到200能买辆小蓝, 请通知我)

- 这部分最终要的是把项目里面的几个练习做完, 和完成项目文件相关度超高。在week4的导学中有详细讲解: <https://github.com/mengfanchun2017/DAND-Basic/blob/master/week4-guide.md>

## // 能力3:完成项目

- 独立完成一个项目是非常不容易的, 即使有助教的帮助, 自己也要付出很多努力, 但是这个过程十分值得
- 即使细节都知道了, 从无到有, 从文件到提交也是不能出现一点错误
- 而且还有适应调试出错的过程, 当你郁闷的时候切记一点: **这很正常, 报错信息能够提供很多线索, 尝试着解决问题是非常重要的能力!**

## // 4.数据分析工具汇总

另外, 为了能够顺利完成项目2, 和以后的几个数据分析项目, 我们还学习准备了本地环境:

**Anaconda / Jupyter Notebook / Spyder:**

1. (非必须但推荐) 本地Anaconda环境, 安装后可以运行: Python3、Jupyter Notebook、Spyder
2. Anaconda的安装和Jupyter Notebook的安装配置请见可选内容: <https://classroom.udacity.com/nanodegrees/nd002-cn-basic/parts/91b5b867-4a7f-49c5-b658-57521a8de12d>
3. spyder的简单教程: <https://blog.csdn.net/LucyGill/article/details/78068985>
4. 注意一点, 本地环境并不是必须的, 所有任务都可以在uda平台完成。请一定记住: **按照计划完成项目优先! ! ! 等过关了之后, 大家再回来复盘和抠细节。**

## /目标2/: 知识点梳理

大家还记得开学时候Uda的两个原则么? 本质是保持住项目进度, 循环往复的螺旋上升, 不要去扣不影响项目完成的细节, 等通关了再来复盘就好了! 切记咬紧学习进度!

**最快学习原则:**

- Need to know Principle 按需知情原则
- Competence without comprehension 无需理解即可完成能力

## / Week1的必学模块

---

- 数据类型和运算符：和试学项目是差不多的，Python基础
- 控制流：<https://github.com/mengfanchun2017/DAND-Basic/blob/master/week2-guide.md#%E7%9B%AE%E6%A0%872%E6%8E%A7%E5%88%B6%E6%B5%81> 这个部分，对于两种循环的区别可以参照之前试学的导学：<https://github.com/mengfanchun2017/DAND-Basic-P0/blob/master/day6-guide.md>
- 函数：<https://github.com/mengfanchun2017/DAND-Basic/blob/master/week2-guide.md#%E7%9B%AE%E6%A0%872%E6%8E%A7%E5%88%B6%E6%B5%81>

## / Week2的必学模块

---

- Pandas Series：先了解Pandas的一维结构是怎样的，和它的特点（其实Dataframe就是嵌套2维的Series）。<https://github.com/mengfanchun2017/DAND-Basic/blob/master/week3-guide.md#13%E7%BB%83%E4%B9%A0pandas-series>
- Series索引：了解Pandas的Series和Dataframe是可以进行索引操作的 <https://github.com/mengfanchun2017/DAND-Basic/blob/master/week3-guide.md#14%E7%BB%83%E4%B9%A0series%E7%B4%A2%E5%BC%95>
- 二维Numpy数组：此处虽然不是Pandas的Dataframe，但是结构是类似的，逻辑搞懂即可。<https://github.com/mengfanchun2017/DAND-Basic/blob/master/week3-guide.md#3%E7%BB%83%E4%B9%A0%E4%BA%8C%E7%BB%B4numpy%E6%95%B0%E7%BB%84>
- 访问DataFrame元素：重点中的重点！对Dataframe要上手操作了，练习也不要放过！ <https://github.com/mengfanchun2017/DAND-Basic/blob/master/week3-guide.md#6%E7%BB%83%E4%B9%A0%E8%AE%BF%E9%97%AEdataframe%E5%85%83%E7%B4%A0>
- 练习：每小时入站和出站数。相当于一个小项目，把上面讲的都串起来了，不要错过。<https://github.com/mengfanchun2017/DAND-Basic/blob/master/week3-guide.md#17%E7%BB%83%E4%B9%A0%E6%AF%8F%E5%B0%8F%E6%97%B6%E5%85%A5%E7%AB%99%E5%92%8C%E5%87%BA%E7%AB%99%E6%95%B0>

## /目标3/：项目文件优化（看懂即可）

---

对于上周week4导学中的项目讲解，其实已经达到了项目通过的水平，但是编程这件事，我们还是可以尽量优化一下，对比上周的项目讲解，我们还有这几个地方可以改进：

1. gender数据不完整：3个城市的的数据文件中，washington是没有性别数据的，选中的话会报错虽然可以使用try/except优化输出

2. 对于输出显示并不友好，看着有点晕，调整输出显示
3. 对于用户的3个输入，也可以写一个小函数调用3回。3次区别并不大，当输入比较多的时候（或者要从很多种备选中选择的时候）会比较简洁
4. 年统计输出为1988.0的问题处理（年么，还是输出1988好看对不对）
5. 对于用户输入的友好响应（如果用户输入ChiCago，也能正确完成分析）
6. 上周的文件有不符合pep编程规范的地方（Atom中的Pylint可以提示），需要修改更加专业（本小节请大家自己研究）。关于Pep请参考之前的一个介绍：<https://github.com/mengfanchun2017/DAND-Basic-P0/blob/master/day5-guide.md>

## / 1.gender数据不完整

使用try/except可以对出错进行判断并处理，不会造成程序停止：

```
1 | try:
2 |     gender_types = df['Gender'].value_counts()
3 | except KeyError:
4 |     print('>>>The city you choose do not have /Gender/ data:')
5 | else:
6 |     print('\n>>>gender types is:')
7 |     print(gender_types)
8 | finally:
9 |     pass
```

具体的try/except用法在week2导学中有讲解，摘录如下：

- try：这是 try 语句中的唯一必需子句。该块中的代码是 Python 在 try 语句中首先运行的代码。
- except：如果 Python 在运行 try 块时遇到异常，它将跳到处理该异常的 except 块。
- else：如果 Python 在运行 try 块时没有遇到异常，它将在运行 try 块后运行该块中的代码。
- finally：在 Python 离开此 try 语句之前，在任何情形下它都将运行此 finally 块中的代码，即使要结束程序，例如：如果 Python 在运行 except 或 else 块中的代码时遇到错误，在停止程序之前，依然会执行此finally 块。

举个 lesson2 13节的例子做个简单的说明（以注释方式）：

```
1 | def create_groups(items, num_groups):
2 | #定义函数，2个输入items（多少个东西），分成num_groups(分成多少个组)
3 |     try:
4 |         size = len(items) // num_groups
5 |         #上来是计算每组大小
6 |     except ZeroDivisionError:
```

```

7     print("WARNING: Returning empty list. Please use a nonzero number")
8 r.")
9     return []
10    #但是当发生ZeroDivisionError时（除数为0的时候的错误），就显示错误并返回空值
11 else:
12     groups = []
13     for i in range(0, len(items), size):
14         #是说从0到items的最大数（就是len(items)这个的结果），按照size（在try语句
15         中得出的每组大小）进行循环
16         groups.append(items[i:i + size])
17         #每一个循环，把当前的组存追加存放到groups里面
18     return groups
19 #如果没报错，就是按照上面这一段把每组都有什么写到groups里面
20 finally:
21     print("{} groups returned.".format(num_groups))
22 #无论怎么处理的，都打印一行提示，使用的是格式化字符串的方式
23
24 print("Creating 6 groups...")
25 for group in create_groups(range(32), 6):
26     print(list(group))
27 print("\nCreating 0 groups...")
28 for group in create_groups(range(32), 0):
29     print(list(group))

```

输出是这样的：

```

1 Creating 6 groups...
2 6 groups returned.
3 [0, 1, 2, 3, 4]
4 [5, 6, 7, 8, 9]
5 [10, 11, 12, 13, 14]
6 [15, 16, 17, 18, 19]
7 [20, 21, 22, 23, 24]
8 [25, 26, 27, 28, 29]
9 [30, 31]
10
11 Creating 0 groups...
12 WARNING: Returning empty list. Please use a nonzero number.
13 0 groups returned.

```

## / 2. 输出优化

这里比较偏向个性化，可以使用print较复杂一点的方式，除了之前介绍的格式化字符串方法，还

可以使用print的.center(),.ljust()方式进行行级排版。不是很复杂，有兴趣的同学可以搜索一下。  
最后我的输出是这样的：

```
1 ##Hello! Let's explore some US bikeshare data!##
2
3 -----Step1 : Get input-----
4
5 q1/3: which city do you want to know?
6 option:<chicago,new york city,washington>
7 washington
8
9 q2/3: which month do you want to know?
10 option:<all,january,february,march,april,mai,june>
11 jan
12 ---warning: I do not have data about that month.
13 ---Or you type a wrong name
14 ---Input Again
15
16 q2/3: which month do you want to know?
17 option:<all,january,february,march,april,mai,june>
18 january
19
20 q3/3: which day do you want to know?
21 option:<all,monday,tuesday,wednesday, ... ,sunday>
22 all
23
24 >>>>>>>>>Got Inputs:>>>>>>>>
25 >>>city requirement: washington
26 >>>month requirement: january
27 >>>day requirement: all
28
29 -----Step2 : Computing-----
30
31 ...Calculating The Most Frequent Times of Travel...
32
33 (Took 0.04522085189819336 seconds.)
34 >>>max freq month is: 1
35 >>>max freq day is: 12
36 >>>max freq hour is: 17
37
38 ...Calculating The Most Popular Stations and Trip...
39
40 (This took 0.025053977966308594 seconds.)
41 >>>max freq start station is:
42 Columbus Circle / Union Station
```

```
43
44 >>>max freq end station is:
45 Columbus Circle / Union Station
46
47 >>>max freq combine statuion is:
48 Columbus Circle / Union Station --- 8th & F St NE
49
50 ...Calculating Trip Duration...
51
52 (Took 0.0006821155548095703 seconds.)
53 >>>total trip time is:
54 26948288.554999996
55 >>>mean trip time is:
56 896.6921290719757
57
58 ...Calculating User Stats...
59
60 >>>user types is:
61 Subscriber      26246
62 Customer        3807
63 Name: User Type, dtype: int64
64 >>>The city you choose do not have /Gender/ data:
65 >>>The city you choose do not have /Year/ data:
66 (Took 0.006235837936401367 seconds.)
67
68
69 ### Proceeding complete.
70 ### Would you like to restart?
71 ### Enter yes to restart or any key to quit.
```

## / 3. 使用循环获得用户输入

我们先写一个函数，实现检查用户输入是不是在指定的可选范围内：

- 函数的输入是3个变量
- 变量1是输入的提示词
- 变量2是当输入有错的时候的提示词（不在列表中）
- 变量3是进行判断的列表

```
1 def phrase_input(input_prompt,err_prompt,option_list):
2
3     user_input = input(input_prompt)
4     while user_input not in option_list:
```

```
5     user_input = input(err_prompt)
6     return user_input
```

之后就可以在filter\_data()函数中调用3次，生成输出了，修改之后的代码和解释如下：

```
1 def get_filters():
2
3     city_option = CITY_DATA.keys()
4     #直接指定城市的话不太灵活，可以根据CITY_DATA的keys () 进行判断
5     #.keys()方法就是吧CITY_DATA中的key输出成一个列表，和下面这句等价：
6     #city_option = ['chicago', 'new york city', 'washington']
7     month_option = ['all', 'january', 'february', 'march', 'april', 'may'
8     ,
9         'june']
10    day_option = ['all', 'monday', 'tuesday', 'wednesday',
11                  'thursday', 'friday', 'saturday', 'sunday' ]
12
13    #将city、month、day、err的输出分离出来
14    city_prompt = 'q1/3: which city do you want to know? \
15                           \noption:<chicago,new york city,washington> \n>>> '
16    month_prompt = 'q2/3: which month do you want to know? \
17                           \noption:<all,january,february,march,april,may,jun
18 e>\n                           \n>>> '
19
20    day_prompt = 'q3/3: which day do you want to know? \
21                           \noption:<all,monday,tuesday,wednesday, ...,sunday>
22 \
23                           \n>>> '
24
25    err_prompt = '---warning: I do not have that data.\n
26                           \n---Or you type a wrong name\n---Input Again\n>>> '
27
28    #调用3次函数，将结果负责给city、month、day三个变量
29    city = phrase_input(city_prompt,err_prompt,city_option)
30    month = phrase_input(month_prompt,err_prompt,month_option)
31    day = phrase_input(day_prompt,day_option)
32
33    #输出结果
34    str_got_input = 'Got Inputs:'
35    print(' ')
36    print(str_got_input.center(30,'>'))
37    #str有.ljust .center .rjust等方式不使用变量的话很方便
38    #http://www.tutorialspoint.com/python/string_ljust.htm
39    print('>>>city requirement:',city)
40    print('>>>month requirement:',month)
41    print('>>>day requirement:',day)
```

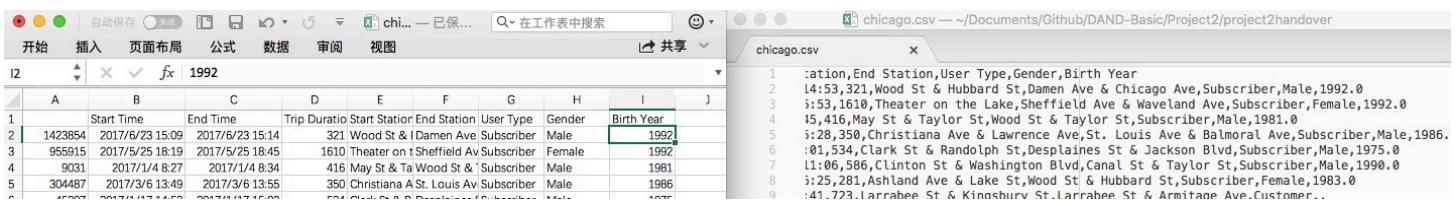
```
return city, month, day
```

## / 4.输出年为整数

整体函数的输出中有一段是这样的：

```
1 >>>earliest year of birth is:  
2 1899.0  
3 >>>recent year of birth is:  
4 2016.0  
5 >>>common year of birth is:  
6 1989.0
```

那么为什么年份的结果会是1899.0这种方式呢，其实这是因为原始数据中都是1899.0这样的，所以在被读入的时候会变成float，所以输出就是小数了。但是如果使用excel打开csv的话，这一列是整数，所以容易让人迷惑：



|   | A       | B               | C               | D             | E                         | F           | G         | H      | I          | J |
|---|---------|-----------------|-----------------|---------------|---------------------------|-------------|-----------|--------|------------|---|
| 1 |         | Start Time      | End Time        | Trip Duration | Start Station             | End Station | User Type | Gender | Birth Year |   |
| 2 | 1423854 | 2017/6/23 15:09 | 2017/6/23 15:14 | 321           | Wood St & Damen Ave       | Subscriber  | Male      | 1992   |            |   |
| 3 | 955915  | 2017/5/25 18:19 | 2017/5/25 18:45 | 1610          | Theater on t Sheffield Av | Subscriber  | Female    | 1992   |            |   |
| 4 | 9031    | 2017/1/4 8:27   | 2017/1/4 8:34   | 416           | May St & Ta Wood St &     | Subscriber  | Male      | 1981   |            |   |
| 5 | 304487  | 2017/3/6 13:49  | 2017/3/6 13:55  | 350           | Christiana A St. Louis Av | Subscriber  | Male      | 1986   |            |   |

chicago.csv — ~/Documents/Github/DAND-Basic/Project2/project2handover

```
1 :ation,End Station,User Type,Gender,Birth Year  
2 L4:53,321,Wood St & Hubbard St,Damen Ave & Chicago Ave,Subscriber,Male,1992.0  
3 i:53,1610Theater on the Lake,Sheffield Ave & Waveland Ave,Subscriber,Female,1992.0  
4 15,416,May St & Taylor St,Wood St & Taylor St,Subscriber,Male,1981.0  
5 i:28,350,Christiana Ave & Lawrence Ave,St. Louis Ave & Balmoral Ave,Subscriber,Male,1986.  
6 :01,534,Clark St & Randolph St,Desplaines St & Jackson Blvd,Subscriber,Male,1975.0  
7 L1:06,586,Clinton St & Washington Blvd,Canal St & Taylor St,Subscriber,Male,1990.0  
8 i:25,281,Ashland Ave & Lake St,Wood St & Hubbard St,Subscriber,Female,1983.0  
9 :41,723,Larrahee St & Kinnshury St,Larrahee St & Armitage Ave,Customer..
```

比较简单的解决方式是在输出那里加一个int转换成整数：

```
1 print('\n>>>earliest year of birth is:')  
2 print(int(earliest))  
3 print('\n>>>recent year of birth is:')  
4 print(int(recent))  
5 print('\n>>>common year of birth is:')  
6 print(int(common))
```

也可以使用.to\_datetime()将这一列转换成pandas中的时间格式，但是需要加参数，因为原本是float格式，不加参数会转换出奇怪的结果，有兴趣的同学可以自己研究下。其实pandas中的readcsv是可以定义列格式的，不过使用比较复杂也容易出错（要定义每一列的格式），了解下就好了。

## / 5.允许输入大写字母

如果用户输入了大写字母，按照之前的文件还是算错误的，因为判断的文件列表中所有城市是小

写，那么我们有没有方式避免让用户再次输入并且问候我们的亲人呢？有的，可以使用字符串方法，将输入变成小写（基于之前的循环获得输入方法）：

```
1 | user_input = input(input_prompt).lower()
```

当然，如果在获得user\_input时候不转换，在以后需要判断时、调用文件时转换也可以的，只不过：

- 首先，存储了用户的原始 WaSHINTON 输入没有意义
- 其次，在使用时候都要转换，复杂容易忘记加.lower()

这里的输出如下，注意输入和反馈输入washington是不一样的：

```
1 | -----Step1 : Get input-----
2 |
3 | q1/3: which city do you want to know?
4 | option:<chicago,new york city,washington>
5 | waShington
6 |
7 | q2/3: which month do you want to know?
8 | option:<all,january,february,march,april,may,june>
9 | all
10 |
11 | q3/3: which day do you want to know?
12 | option:<all,monday,tuesday,wednesday, ... ,sunday>
13 | all
14 |
15 | >>>>>>>Got Inputs:>>>>>>>
16 | >>>city requirement: waShington
17 | >>>month requirement: all
18 | >>>day requirement: all
```

Title: W6 数据分析方法论 [项目: 探索数据集项目1/4]

Tags: 数据分析初级, 实战项目

# W6 数据分析方法论 [项目: 探索数据集项目1/4]

---

- W6 数据分析方法论 [项目: 探索数据集项目1/4]
- /学习地图/
  - / 项目路径
  - / 项目推进
- /目标1/: 数据分析过程
  - \*\*/ 2.课程概述
  - \*\*/ 3.数据分析的应用
  - \*\*\*/ 5.数据分析过程概述
  - \*\*/ 6.数据分析过程练习
  - \*/ 9.提问 10.数据集问题
  - \*/ 11.数据整理和EDA
  - \*\*\*/ 13.阅读csv文件
  - \*\*|14 评估和理解
  - / 15.评估和理解练习
  - \*\*\*/ 17.清理示例
  - \*\*/ 18.清理练习
  - \*\*/ 20.使用Pandas绘图
  - \*\*/ 23.得出结论示例
  - \*\*/ 24.练习: 得出结论
  - \*/ 26.传达结果示例
  - \*/27 传达结果练习
- /目标2/: 案例研究1、2 (见本周附加导学文件)

## /学习地图/

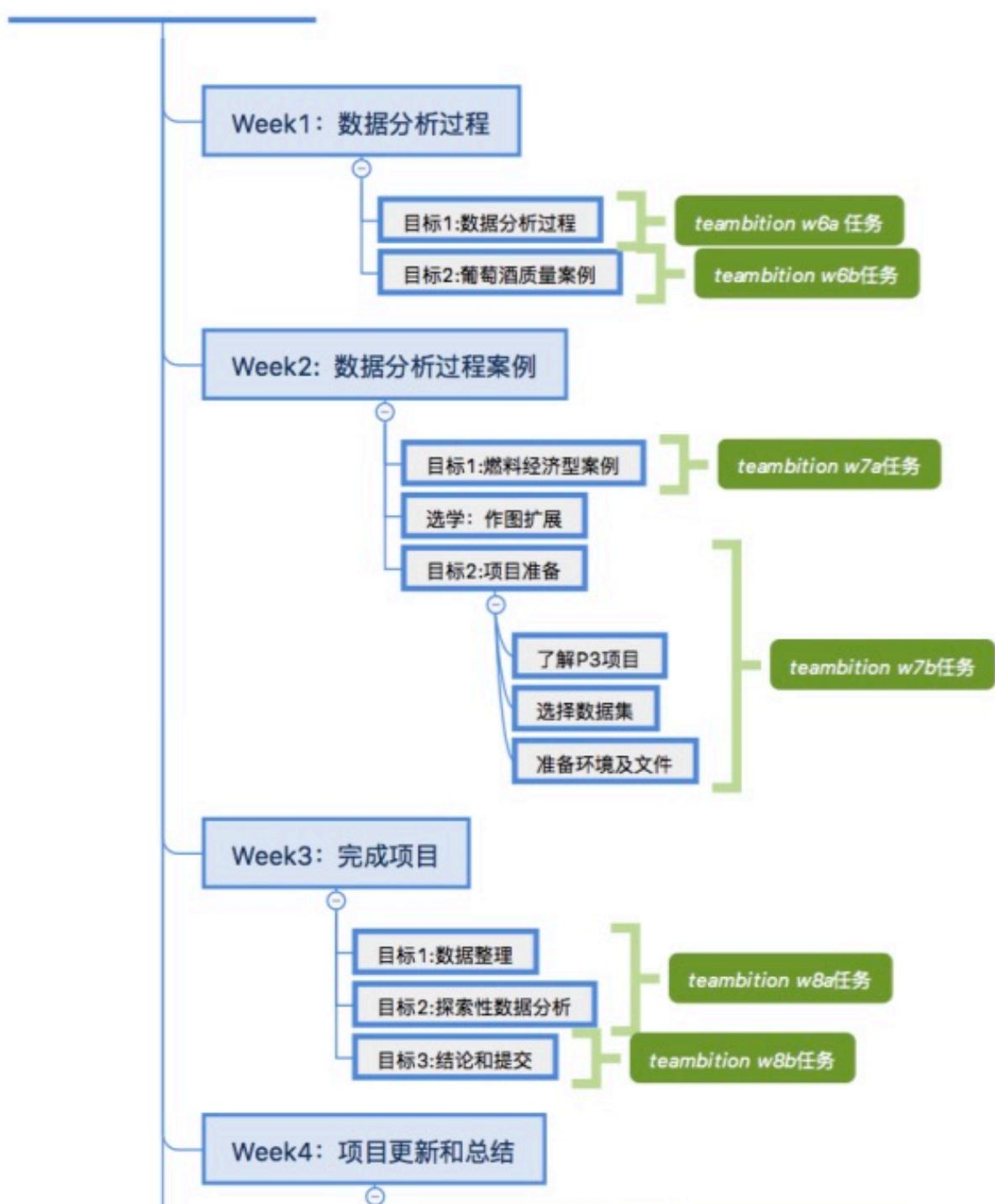
---

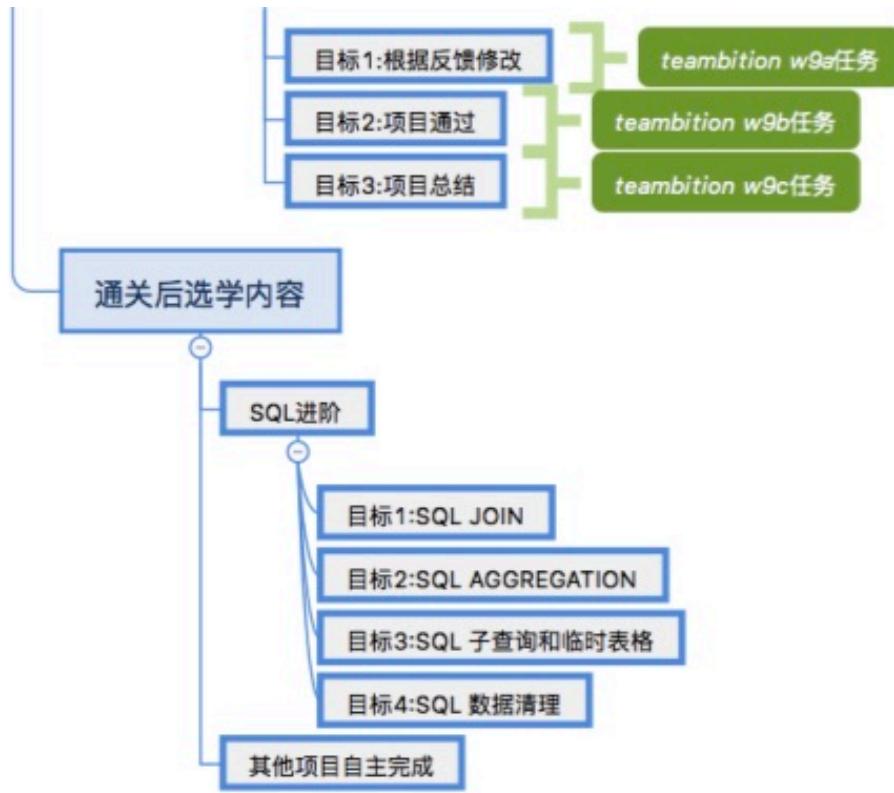
本周是数据分析课项目3的Part1, 是项目3的基础内容。课程将会通过几个小的数据贯穿讲解数据分析过程, 新加入的知识点是怎样进行数据清理, 因为现实世界的数据往往有很多瑕疵, 而要想得到良好的分析结果, 就要把这些瑕疵处理掉。大家加油!

数据分析入门是数据分析微学位的第3个项目，项目时间为4周，在完成学习之后。可以从5个数据集中选取一个进行数据分析的项目提交。其中4个数据使用Python处理，是重点。如果习惯使用SQL进行处理，也有一个足球的数据集和相关内容（建议优先完成Python的，SQL的可以以后回来复盘再学）。这个项目的学习过程就比较完全，在课程中会串讲一遍知识点（第一遍讲解），之后是红酒的案例分析（一遍运用中讲解），最后是燃料经济学的案例分析（又是一遍运用中讲解）。在这3个部分都学习完成后，就可以着手选择项目完成了。需要注意，本部分不包括统计学的内容，将会在下一个部分引入。具体每周内容和知识点汇总见后面小节。

## / 项目路径

### 探索数据集





## / 项目推进

**Week1要求（本周任务）：**

- 完成项目简介部分
- 这次项目中有4个数据可以选择（新手不建议SQL足球那个），请根据项目说明，选一个可心的呦/[项目数据说明/](#)

## /目标1/: 数据分析过程

### \*\*/ 2.课程概述

着节是项目3的开始，看完之后应该对整体的课程设计和需要的SQL知识（项目1）、python知识（项目2）有所了解。并且提供了不少额外资源的链接。

此处最后一个链接是一本超好的书，可以陪你走到微学位完成，请不要错过（鸟文的，中文的也有），如果链接不可用，可以试试我搬运的：

### \*\*/ 3.数据分析的应用

---

提供了5个链接，对于数据分析能干什么做了说明：

- 百万数据告诉你第一次约会用来了解对方的最佳问题
- 看看沃尔玛如何使用大数据分析来增加销量
- 你还可以了解Bill James 如何将数据分析应用于棒球
- 数据分析如何帮助设计药物
- 这篇Facebook 博客(需科学上网) 和另一篇文章 介绍如何用数据分析社交媒体上的意识形态

## \*\*\*/ 5.数据分析过程概述

---

1. 提问
  - 好奇心是最好的老师
2. 整理数据
  - Gather 收集
  - Assess 评估
  - Clean 清理
3. 执行EDA (探索性数据分析)
  - 包括画图进行观察
4. 得出结论 (或做出预测)
  - 通过机器学习得出
  - 通过推断统计学得出 (项目4的重点呦，我们到时候再学习)
5. 传达结果
  - 一图胜千言，交出图来！

## \*\*/ 6.数据分析过程练习

---

又是一份共享单车的数据！这节请看看，就当复习了。

## \*/ 9.提问 10.数据集问题

---

在第9节介绍了一份肿瘤数据：

<https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29>

原始数据是.data格式，第10节练习中的工作空间中可以导入服务器端的csv文件对数据进行观察和提问。

## \*/ 11.数据整理和EDA

---

本节讨论了数据分析流程中的第2步：Wrangling（数据整理）和第3步：EDA（探索性数据分析）之间的关系和交互过程。本部分说明包含了几个子类的说明，将相应节号整合加入便于理解：

### Wrangling：主要解决收集数据工作

- Gather收集数据 |**12 收集数据** 可以下载、从API获取、网页获取（爬虫）或者使用公司的数据库。
- Assess评估数据，见**14节练习**
- Clean

### EDA：主要解决探索数据工作

- Explor探索数据
- Augment增强（好奇怪的名词啊）应该是就是根据探索的方式对一些点深入研究的意思

## \*\*\*/ 13. 阅读csv文件

本节对pandas.read\_csv()做了超级详细的扩展，推荐在Uda工作空间中完成（因为文件是在Uda工作空间中，不建议在本地做）。我们抓个其中的电厂的例子做说明，如果只是简单的读入csv文件，输入输出是这样子的：

```
df_powerplant = pd.read_csv('powerplant_data.csv')
df_powerplant.head(3)
```

|   | AT    | V     | AP      | RH    | PE     |
|---|-------|-------|---------|-------|--------|
| 0 | 8.34  | 40.77 | 1010.84 | 90.01 | 480.48 |
| 1 | 23.64 | 58.49 | 1011.40 | 74.20 | 445.75 |
| 2 | 29.74 | 56.90 | 1007.15 | 41.91 | 438.76 |

原始输出，根本看不出是什么意思，我们可以查看下数据说明：

Features consist of hourly average ambient variables

- Temperature (T) in the range 1.81°C and 37.11°C,
- Ambient Pressure (AP) in the range 992.89-1033.30 milibar,
- Relative Humidity (RH) in the range 25.56% to 100.16%
- Exhaust Vacuum (V) in the range 25.36-81.56 cm Hg
- Net hourly electrical energy output (EP) 420.26-495.76 MW

那么我们就可以在读取csv的时候把列的名称做扩展了说明，这样看就好多了：

```

df_powerplant = pd.read_csv('powerplant_data.csv',
                            names = ['Temperature', 'Pressure',
                                      'Humidity', 'Vacuum', 'Output'])
df_powerplant.head(3)

```

|   | Temperature | Pressure | Humidity | Vacuum | Output |
|---|-------------|----------|----------|--------|--------|
| 0 | AT          | V        | AP       | RH     | PE     |
| 1 | 8.34        | 40.77    | 1010.84  | 90.01  | 480.48 |
| 2 | 23.64       | 58.49    | 1011.4   | 74.2   | 445.75 |

那么接下来我们就可以使用`.to_csv()`方法把数据集存储为新的csv文件就可以了，这里注意有个`index`的问题要考虑下：

|   | Unnamed: 0 | Temperature | Pressure | Humidity | Vacuum | Output |
|---|------------|-------------|----------|----------|--------|--------|
| 0 | 0          | AT          | V        | AP       | RH     | PE     |
| 1 | 1          | 8.34        | 40.77    | 1010.84  | 90.01  | 480.48 |
| 2 | 2          | 23.64       | 58.49    | 1011.4   | 74.2   | 445.75 |
| 3 | 3          | 29.74       | 56.9     | 1007.15  | 41.91  | 438.76 |
| 4 | 4          | 19.07       | 49.69    | 1007.22  | 76.79  | 453.09 |

这个 `Unnamed: 0` 是什么？`to_csv()` 默认保存索引，除非指定不保存。如需忽略索引，必须提供参数 `index=False`

```
df_powerplant.to_csv('powerplant_data_edited.csv', index=False)
```

```

df = pd.read_csv('powerplant_data_edited.csv')
df.head()

```

|   | Temperature | Pressure | Humidity | Vacuum | Output |
|---|-------------|----------|----------|--------|--------|
| 0 | AT          | V        | AP       | RH     | PE     |
| 1 | 8.34        | 40.77    | 1010.84  | 90.01  | 480.48 |
| 2 | 23.64       | 58.49    | 1011.4   | 74.2   | 445.75 |
| 3 | 29.74       | 56.9     | 1007.15  | 41.91  | 438.76 |
| 4 | 19.07       | 49.69    | 1007.22  | 76.79  | 453.09 |

里面新出来的这个`index`是行的名字，扩展看懂下面这个就行了：

```
df = pd.read_csv('student_scores.csv', index_col='Name')
df.head()
```

|       | ID    | Attendance | HW   | Test1 | Project1 | Test2 | Project2 | Final |
|-------|-------|------------|------|-------|----------|-------|----------|-------|
| Name  |       |            |      |       |          |       |          |       |
| Joe   | 27604 | 0.96       | 0.97 | 87.0  | 98.0     | 92.0  | 93.0     | 95.0  |
| Alex  | 30572 | 1.00       | 0.84 | 92.0  | 89.0     | 94.0  | 92.0     | 91.0  |
| Avery | 39203 | 0.84       | 0.74 | 68.0  | 70.0     | 84.0  | 90.0     | 82.0  |
| Kris  | 28592 | 0.96       | 1.00 | 82.0  | 94.0     | 90.0  | 81.0     | 84.0  |
| Rick  | 27492 | 0.32       | 0.85 | 98.0  | 100.0    | 73.0  | 82.0     | 88.0  |

```
df = pd.read_csv('student_scores.csv', index_col=['Name', 'ID'])
df.head()
```

|       |       | Attendance | HW   | Test1 | Project1 | Test2 | Project2 | Final |
|-------|-------|------------|------|-------|----------|-------|----------|-------|
| Name  | ID    |            |      |       |          |       |          |       |
| Joe   | 27604 | 0.96       | 0.97 | 87.0  | 98.0     | 92.0  | 93.0     | 95.0  |
| Alex  | 30572 | 1.00       | 0.84 | 92.0  | 89.0     | 94.0  | 92.0     | 91.0  |
| Avery | 39203 | 0.84       | 0.74 | 68.0  | 70.0     | 84.0  | 90.0     | 82.0  |
| Kris  | 28592 | 0.96       | 1.00 | 82.0  | 94.0     | 90.0  | 81.0     | 84.0  |
| Rick  | 27492 | 0.32       | 0.85 | 98.0  | 100.0    | 73.0  | 82.0     | 88.0  |

## \*\*|14 评估和理解

本节的重点是对dataframe的一些信息的获取，可以用于评估所收到数据情况，有以下几个小点：

- 通过.shape检查数据的维度信息：下面的输出是说df是个有569行和32列的二维数据：

```
# 返回数据框维度的元组
df.shape
```

```
(569, 32)
```

- 通过.dtypes检查各列的类型：（注意diagnosis本身是str字符串格式，这是由于str在dataframe中以对象方式呈现，如果深入到一个元素使用type(df['diagnosis'][0])检查的话还是会显示str）

```
# 返回列的数据类型
df.dtypes
```

```
id                  int64
diagnosis          object
radius_mean        float64
```

- 通过.info()检查所有列的数量
- 通过.describe()检查数据集每列的统计学信息：（一个8个）

```
# 返回每列数据的有效描述性统计
df.describe()
```

|              | <b>id</b>    | <b>radius_mean</b> | <b>texture_mean</b> | <b>perimeter_mean</b> | <b>area_mean</b> | <b>smoothness_mean</b> |
|--------------|--------------|--------------------|---------------------|-----------------------|------------------|------------------------|
| <b>count</b> | 5.690000e+02 | 569.000000         | 548.000000          | 569.000000            | 569.000000       | 521.000000             |
| <b>mean</b>  | 3.051467e+07 | 14.113021          | 19.293431           | 91.877909             | 653.288576       | 0.096087               |
| <b>std</b>   | 1.250417e+08 | 3.506148           | 4.327287            | 24.162787             | 349.476899       | 0.013924               |
| <b>min</b>   | 8.670000e+03 | 6.981000           | 9.710000            | 43.790000             | 143.500000       | 0.052630               |
| <b>25%</b>   | 8.691040e+05 | 11.700000          | 16.167500           | 75.170000             | 420.300000       | 0.086050               |
| <b>50%</b>   | 9.060240e+05 | 13.370000          | 18.785000           | 86.340000             | 551.100000       | 0.095780               |
| <b>75%</b>   | 8.910251e+06 | 15.780000          | 21.825000           | 103.800000            | 782.700000       | 0.104800               |
| <b>max</b>   | 9.113205e+08 | 28.110000          | 39.280000           | 188.500000            | 2501.000000      | 0.163400               |

8 rows × 31 columns

- 通过.columns .index可以看到数据集的行列信息，注意columns的输出是一个列表，所以可以遍历访问：

```
print(df.index)
```

```
RangeIndex(start=0, stop=569, step=1)
```

```
print(df.columns)
```

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave_points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_SE', 'texture_SE', 'perimeter_SE', 'area_SE', 'smoothness_SE',
       'compactness_SE', 'concavity_SE', 'concave_points_SE', 'symmetry_SE',
       'fractal_dimension_SE', 'radius_max', 'texture_max', 'perimeter_max',
       'area_max', 'smoothness_max', 'compactness_max', 'concavity_max',
       'concave_points_max', 'symmetry_max', 'fractal_dimension_max'],
      dtype='object')
```

```
# 查看每列的索引号和标签
```

```
for i, v in enumerate(df.columns):
    print(i, v)
```

```
0 id
1 diagnosis
2 radius_mean
3 texture_mean
```

- 通过.loc[] .iloc[]选择所需数据，第一种是key value对应，第二种是通过索引。简单的说是这样选：[行范围,列范围]具体看例子：

```
# 选择从 'id' 到最后一个均值列的所有列
df_means = df.loc[0:10,'id':'fractal_dimension_mean']
df_means.head()
```

|   | id       | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean |
|---|----------|-----------|-------------|--------------|----------------|-----------|-----------------|
| 0 | 842302   | M         | 17.99       | NaN          | 122.80         | 1001.0    | 0.11840         |
| 1 | 842517   | M         | 20.57       | 17.77        | 132.90         | 1326.0    | 0.08474         |
| 2 | 84300903 | M         | 19.69       | 21.25        | 130.00         | 1203.0    | 0.10961         |
| 3 | 84348301 | M         | 11.42       | 20.38        | 77.58          | 386.1     | NaN             |
| 4 | 84358402 | M         | 20.29       | 14.34        | 135.10         | 1297.0    | 0.10031         |

```
# 用索引号重复以上步骤
df_means = df.iloc[0:10,:12]
df_means.head()
```

|   | id       | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean |
|---|----------|-----------|-------------|--------------|----------------|-----------|-----------------|
| 0 | 842302   | M         | 17.99       | NaN          | 122.80         | 1001.0    | 0.11840         |
| 1 | 842517   | M         | 20.57       | 17.77        | 132.90         | 1326.0    | 0.08474         |
| 2 | 84300903 | M         | 19.69       | 21.25        | 130.00         | 1203.0    | 0.10961         |
| 3 | 84348301 | M         | 11.42       | 20.38        | 77.58          | 386.1     | NaN             |
| 4 | 84358402 | M         | 20.29       | 14.34        | 135.10         | 1297.0    | 0.10031         |

- 挑选不连续的列：在选择时可以用列表挑选需要的列，两种实现方式（后面的用了变量方便修改）：

```
# 选择从任意列 (列表方式)
cols = ['id', 'fractal_dimension_mean']
df_means = df.loc[0:10, ['id', 'fractal_dimension_mean']]
df_means.head(20)
```

|    | id       | fractal_dimension_mean |
|----|----------|------------------------|
| 0  | 842302   | 0.07871                |
| 1  | 842517   | 0.05667                |
| 2  | 84300903 | 0.05999                |
| 3  | 84348301 | 0.09744                |
| 4  | 84358402 | 0.05883                |
| 5  | 843786   | 0.07613                |
| 6  | 844359   | 0.05742                |
| 7  | 84458202 | 0.07451                |
| 8  | 844981   | 0.07389                |
| 9  | 84501001 | 0.08243                |
| 10 | 845636   | 0.05697                |

```
# 用索引号重复以上步骤
cols = [1,10,12]
df_means = df.iloc[0:10,cols]
df_means.head(20)
```

- 但是上面的方法不能在不连续的列中混入范围的选择，如果列很多的时候要这样处理：

```
# 使用iloc选择复杂筛选所需列
df_means = df.iloc[0:10,np.r_[1,10:12]]
```

|   | diagnosis | symmetry_mean | fractal_dimension_mean |
|---|-----------|---------------|------------------------|
| 0 | M         | 0.2419        | 0.07871                |
| 1 | M         | 0.1812        | 0.05667                |
| 2 | M         | 0.2069        | 0.05999                |
| 3 | M         | 0.2597        | 0.09744                |
| 4 | M         | 0.1809        | 0.05883                |
| 5 | M         | 0.2087        | 0.07613                |
| 6 | M         | NaN           | 0.05742                |
| 7 | M         | 0.2196        | 0.07451                |
| 8 | M         | 0.2350        | 0.07389                |
| 9 | M         | 0.2030        | 0.08243                |

- 官方文档：<https://pandas.pydata.org/pandas-docs/stable/indexing.html>

## / 15.评估和理解练习

通过练习有一点扩展，总结如下：

- 通过.xx选择df中的列，一下两种方式是等价的：

```
df['education'].unique()
```

```
array(['Bachelors', 'HS-grad', '11th', 'Masters', '9th',
       'Some-college', 'Assoc-acdm', 'Assoc-voc', '7th-8th',
       'Doctorate', 'Prof-school', '5th-6th', '10th', '1st-4th',
       'Preschool', '12th'], dtype=object)
```

```
df.education.unique()
```

```
array(['Bachelors', 'HS-grad', '11th', 'Masters', '9th',
       'Some-college', 'Assoc-acdm', 'Assoc-voc', '7th-8th',
       'Doctorate', 'Prof-school', '5th-6th', '10th', '1st-4th',
       'Preschool', '12th'], dtype=object)
```

- 通过.unique()筛选唯一的值，可以用len直接求出有几个，其实也可以使用.nunique()直接得出：

```
df['education'].nunique()
```

```
16
```

```
len(df.education.unique())
```

```
16
```

15317154270712.jpg)

- 通过.value\_counts()统计一列中的数值个数：

```
df.age.value_counts()
```

```
36    898
31    888
34    886
23    877
```

- 通过.isnull()统计缺失值。在df中有isnull的方法可以检查缺失值（也有notnull）。要在后面加个sum()就可以检查个数了(当然也可以对所有列做处理)：

```
df.age.isnull().sum()
```

```
0
```

```
df.isnull().sum()
```

```
age          0  
workclass    1836  
fnlwgt        0  
education     0  
education-num 0  
marital-status 0  
occupation   1843  
relationship   0  
race          0  
sex           0  
capital-gain   0  
capital-loss   0  
hours-per-week 0  
native-country  583  
income         0  
dtype: int64
```

- 注意`.isnull()`是对值做判断，所以无论是否为空都有一个结果，所以对`isnull()`做`value_count()`也是可以的，注意两种对比：

```
df.workclass.notnull().value_counts()
```

```
True      30725  
False     1836  
Name: workclass, dtype: int64
```

```
df.workclass.isnull().sum()
```

```
1836
```

- 通过`.quantile()`查询相应百分位的值。如果要多个要放在一个列表中，并且小数点前的0可以省略：

```
df.age.quantile(0.25)
```

```
28.0
```

```
df.age.quantile([.25,.50,.75])
```

```
0.25    28.0  
0.50    37.0  
0.75    48.0  
Name: age, dtype: float64
```

## \*\*\*/ 17.清理示例

总算来了，清理数据是本周的重点，大家这一节一定要好好学。首先我们要知道清理数据常见的

3种情况：

1. 缺失值
2. 冗余数据
3. 数据类型错误

我们先从缺失值下手处理这个问题，按照书中的例子，对于duration这样的数字组成的数据列，一种方法是直接把平均值填充到NaN（标示空值的位置）：

```
1 # 方法1, 先定义mean变量, 再使用fillna(mean)将空值填充为mean的值
2 mean = df['view_duration'].mean()
3 df['view_duration'] = df['view_duration'].fillna(mean)
4 # # 此处要用 = 将后面处理后的数据写入原数据
5
6 # 方法2, 在fillna中增加inplace参数, 表示替换
7 # # 并且直接把mean的计算融入fillna的参数中
8 df['view_duration'].fillna(df['view_duration'].mean(), inplace=True)
```

接下来我们处理数据重复的问题：

```
1 # 使用.duplicated()处理
2 # # 注意.duplicated()输出是一个个True / False的列表, 所以要想知道有多少个, 需要使用sum()或者.sum()进行统计
3 df.duplicated().sum()
4
5 # 最后使用.drop_duplicateds()清理数据, 同样可以使用inplace = True进行替换
6 df.drop_duplicateds(inplace=True)
7 # # 另外如果只是对一列中的重复值统计和去掉的话使用subset
8 # # df.drop_duplicateds(subset=['colname'], inplace=True)
9 # # 另外也可以使用keep参数定义保留那一个重复数据
10 # # https://stackoverflow.com/questions/23667369/drop-all-duplicate-rows-in-python-pandas
```

最后的改变数据格式的方式我们已经在项目2中使用过了，还记得 pd.to\_datetime() 这个方法么？详情请见week4导学内容。如果你将转换之后的数据保存为csv下次打开后还不是datetime格式，因为csv无法数据类型。但是这种情况可以通过在读取csv的时候使用parse\_dates参数解决。感兴趣的请戳（选学）：<https://stackoverflow.com/questions/17465045/can-pandas-automatically-recognize-dates>

## \*\*/ 18.清理练习

对于清理，还有一个小练习，建议有精力的不要放过，同样由于csv文件的原因请在工作空间完

成，有2点做个扩展：

```
1 # 扩展1：重命名列
2 # # 可以直接通过赋值重命名df的列，但元素数要一致
3 # # 首先定义一个空列
4 new_labels = []
5 # # 之后用一个循环把原先列中带_mean的列名都改为不带_mean的
6 for col in df.columns:
7     print(col)
8     if '_mean' in col:
9         # 判断如果带_mean
10        new_labels.append(col[:-5])
11        # 那么就把不包括最后5个字母的列名存为新列名
12    else:
13        new_labels.append(col)
14        # 如果没有_mean的话就保持原名字
15
16 print(new_labels)
17 # 检查下是否改好了
18
19 df.columns = new_labels
20 # 将新的列名赋值
21 df.head()
22 # 检查下是不是改过来了，大功告成！
```

```
1 # 扩展2：用均值填充缺失值
2 # # 使用循环将除了前2列之外的进行均值填充处理
3 # # 第一列是用户id，第二列是诊断评级
4 for i in df.columns[2:]:
5     # 使用df.columns遍历列
6     # 2：表示的是从第2列到最后一列（排除了0、1列）
7     df[i].fillna(df[i].mean(), inplace = True)
8     # # 使用上节的方法完成填充
9 # 用 info() 确认修改
10 df.info()
11 # 这一次就可以看到所有列的飞空数字是一样的了（info显示每列后面的是非空数据）
```

后面的去掉重复数据方面，就是一样的了，使用这个数据我们可以观察一下，`duplicated`是找到所有列都一样的，如果只看一列，数据是不同的（正好有分类信息，会重复很多），对比如下：

```
df.duplicated().sum()  
5  
  
df.duplicated(subset=['id']).sum()  
7
```

```
df.duplicated(subset=['diagnosis']).sum()  
567
```

## \*\*/ 20.使用Pandas绘图

Pandas中的绘图功能其实是封装了matplotlib中的功能，所以呢就不用再import一遍了。简单的例子如下（基于已经导入的df数据）：

```
1 # 为了能够在jupyter中显示图形要增加下面这句:  
2 % matplotlib inline  
3  
4 # 可以直接在数据集上调用，会每个列出一个图，比如  
5 # # .plot() 折线图  
6 # # .hist() 直方图  
7 # # 可以在()中定义数据的大小  
8 df.hist(figsize=(8,8));  
9 # # 默认会把图形建立的一些信息一起输出，可以在结尾加上;隐藏信息，只出现图  
10 # # .hist()和.plot(kind='hist')是相同的  
11  
12 # 对于education这样的分类的列，可以使用.value_counts()先把各列数据统计出来，再画图  
13 df['education'].value_counts().plot(kind='bar')  
14  
15 # 放大招！有个.scatter_matrix()可以让你看到每两个变量之间散点的关系，便于进行初步观  
16察  
17 # # 注意使用方法是将数据集作为参数输入  
18 # # 还会展示每个变量的直方图  
19 pd.plotting.scatter_matrix(df, figsize=(15,15));  
20 # # 如果只想看特定的两个变量，这样写：  
df.plot(x='compactness', y='concavity', kind='scatter')
```

## \*\*/ 23.得出结论示例

这里呢大家明白筛选是怎么回事就好（在项目2用过呦），最后的把两个图放在一起显示的代码本周不要求。

```
1 # 这里是使用过滤将df中满足条件诊断为M的存为新的数据df_m
2 df_m = df[df['diagnosis'] == 'M']
```

## \*\*/ 24.练习：得出结论

本节是对上一节可视化的练习，后面需要使用.idmin()方法，了解一下：

```
# 使用.min求最小值
df.loc[:, 'storeC'].min()
```

927

```
# 使用.idxmin求最小值的位置
df.loc[:, 'storeC'].idxmin()
```

9

```
# 检查结果
df.storeC.iloc[9]
```

927

根据同学的提问做个扩展，这里有几个问题是说最大？最后一个月？最后3个月这样的，如果看了数据的话，可以直接把值输入，但是如果换了数据我们怎么办？我们要写一段代码能够处理所有这样类型的数据就帅气了，几个代码的解答：

```
1 # 那么我们继续探索，怎么发现最大年，月与日呢？
2 # 为了方便以后使用我们定几个变量max_year, max_month, max_day
3
4 max_year = df['week'].dt.year.max()
5 # 对于year，我们直接.dt.year后面跟一个max()就可以了
6
7 max_month = df[df['week'].dt.year == max_year]['week'].dt.month.max()
8 # 这里稍微麻烦些，我要确定在最大年的最大月是多少
9 # 如果不加限制的话，肯定输出12月，就没有意义了
10 # 还记得dataframe的filter用法么？
11 # df[中间是过滤条件]
12 # 于是我们用[df['week'].dt.year == max_year]表示年做个限定
13 # 最后使用dt.month.max()求出最大月
14
15 max_day = df[(df['week'].dt.year == max_year) &
16                 (df['week'].dt.month == max_month)]['week'].dt.day.max()
17 # 与上面大致相同，就是要加入两个过滤条件
18 # [(条件1) & (条件2)]
19
```

```

20 # 最后检查下是否和数据一致
21 print(max_year, max_month, max_day)
22 print(df.tail(1))
23
24 # 根据这个规则就可以选中这一列了
25 df[(df['week'].dt.year == max_year) &
26     (df['week'].dt.month == max_month) &
27     (df['week'].dt.day == max_day)]

```

|  | week | storeA     | storeB | storeC | storeD | storeE |      |
|--|------|------------|--------|--------|--------|--------|------|
|  | 199  | 2018-02-25 | 7560   | 6904   | 3569   | 5045   | 2585 |

```

1 # 最后一个月的总销售额
2 last_month = df[(df['week'].dt.year == max_year) &
3                   (df['week'].dt.month == max_month)]
4 print(last_month)
5 # 检查没有问题，我们用sum求和
6 last_month.sum().sum()
7 # 第一个sum是没列求和（每个store的），第二个是把A-Estore的再求和

```

输出是这样的：

```

1 storeA    5865.480
2 storeB    6756.710
3 storeC    4942.105
4 storeD    5431.405
5 storeE    2580.025
6 dtype: float64

```

同样的，对于指定日期和店的销售额：

```

1 # 2016 年 3 月 13 日的销售额
2 df[df['week'] == '2016-03-13'].sum()[1:].sum()
3 # 直接filter中输入要的日期就好了
4 # 注意这种过滤和上面的year\month\day的过滤不同
5 # 当第一次sum以后，会有week这行信息
6 # 如果在sum会报错，因为这行不能sum
7 # 加一个[1:]把week这行去掉，就好了
8
9 # C 店销售额最低的一周
10 print(df['storeC'].idxmin())
11 # 先使用idxmin找到最小值的索引
12 print(df.iloc[df['storeC'].idxmin()])
13 # 再打印出来

```

```
14 # 可以使用.min检查值对不对
15 print(df['storeC'].min())
16 # 最后要求的是最低一周
17 print(df.iloc[df['storeC'].idxmin()]['week'])
18 # 就是独立输出week的值了
```

最后比较好玩的是最后3个月的销售额，写了个函数可以扩展成n个月的：

```
1 # 最后3个月的
2 # 简单的就是把3个sum.sum进行加和
3
4 def lastmonths(n):
5     # 首先是计算出来最后一个月的
6     current_year = max_year
7     current_month = max_month
8     total = 0
9     while n > 0:
10         total = total + df[(df['week'].dt.year == current_year) & (df['w
11 eek'].dt.month == current_month)].sum().sum()
12         # 定义下total是由每次循环的总数加和得来的
13         # 注意条件已经变成了current_year和current_month
14         if current_month == 1:
15             current_month = 12
16             current_year = current_year - 1
17             # 这里要做个判断，如果是1月了，就要变化到前一年的12月
18         else:
19             current_month = current_month - 1
20             # 否则的话只月数少1
21             n = n-1
22             # n计数减1
23         return total
24         # 返回总数
25
26 lastmonths(5)
# 调用函数，想看几个月看几个月
```

## \*/ 26. 传达结果示例

这节一路看完就Ok了，新出现的内容有：

- 定义可视化图形的index，这样两个前后相关的可视化图形在展示上比较统一

```
1 | ind = df_a['education'].value_count().index
```

```
2 # 这句话就是创建了ind变量，这个变量等于后面那个index的输出
3 # 这个输出是对df_a['education']进行统计得出的，类别最多的排序在前面（使用的就是.value_count()方法）
4
5 df_a['education'].value_counts()[ind].plot(kind='bar');
6 # 这句的意思是我要画个图.plot(kind='bar')
7 # 图的内容就是df_a['education'] 的值分类
8 # 这个分类按照从多到少排列（使用的就是.value_count()方法）
9 # 但是我又想了想，还是我指定一下排列顺序吧，现在改为使用ind排列（刚生成的，其实df_a不加这个结果一样的，因为本身就是自己的顺序么）
10
11 df_b['education'].value_counts()[ind].plot(kind='bar');
12 # 但是这行加[ind]就有用了，因为这里如果不加就回按照 df_b生成的从大到小排列了，两个
13 # 图各说各话回很乱的
# 所以引入以df_a顺序生成的[ind]，就能做到排序相同了
```

## \*/27 传达结果练习

此处有的部分和24相同，大家注意在生成一个序列值的地方可以使用做图显示，能够更好的传达自己想法就可以了。

## /目标2/：案例研究1、2（见本周附加导学文件）

Title: W6 Plus 酒品质分析案例

Tags: 数据分析初级, 实战项目

# W6 Plus 酒品质分析案例

---

- [W6 Plus 酒品质分析案例](#)
- [/目标2/: 酒品质分析案例](#)
  - [\\*/ 6.理解numpy库为什么很快](#)
  - [\\*/ 7.如何用numpy增加一列数据](#)
  - [\\*/ 9.重命名列](#)
  - [\\*/ 10.附加数据续 \(写入CSV的index参数\)](#)
  - [\\*\\*/ 11.使用可视化探索数据](#)
  - [\\*/ 12.Pandas Groupby](#)
  - [\\*/ 13.使用Groupby得出结论 \(由值生成分类数据\)](#)
  - [\\*\\*/ 14.15.Pandas Query](#)
  - [\\*\\*/ 16.17.类型和质量图](#)
  - [\\*/ 18.Matplotlib示例](#)
  - [\\*/ 19.20.Matplotlib绘图](#)

## /目标2/: 酒品质分析案例

---

案例1是关于白酒和红酒品质的分析。其中新接触的一些数据处理方式会在以下内容中介绍。数据文件分为白酒和红酒两个文件: winequality-red.csv, winequality-white.csv。数据来源依然是UCI的数据集: <https://archive.ics.uci.edu/ml/datasets/Wine+Quality>

请注意: 为了对应课程中的小节, 每个二级标题中的|x 的x对应课程中的小节编号。

### \*/ 6.理解numpy库为什么很快

---

再次强调了python原生和numpy (pandas里的数学运算很多是从numpy中来的) 后者更快.{6}中差了100倍! 这种原因是C没有自动垃圾回收动态数据什么的功能 (而这些功能是要消耗性能的), 具体解释: [https://stackoverflow.com/questions/418914/why-is-c-so-fast-and-why-aren't-other-languages-as-fast-or-faster](https://stackoverflow.com/questions/418914/why-is-c-so-fast-and-why-aren-t-other-languages-as-fast-or-faster)

## \*/ 7.如何用numpy增加一列数据

在{7}中介绍了在numpy中增加列的方法：

```
1 | color_red = np.repeat('red', red_df.shape[0])
2 | # 通过np.repeat方法制作一个列表，列表的内容是第一个参数'red'，列表的长度是第二个参数
3 | red_df.shape[0]
4 | # 其实shape[0]就是数据的行数（在p2中有讲shape）
5 |
6 | print(color_red)
7 | # 那么我么检查下是不是生成了一个列表
8 |
9 | print(len(color_red))
#再检查下列表的长度
```

输出是这个样子滴：

```
1 | ['red' 'red' 'red' ... 'red' 'red' 'red']
2 | 1599
```

接下来是把生成的颜色数据加入到数据中：

```
1 | white_df['color'] = color_white
2 | # 新列的名字叫'color'
3 | white_df.head()
4 | # 我们再来看看white_df的数据就会发现有一新列叫'color'
```

再之后是用append方法把红酒和白酒的数据搞在一起，因为这时候已经有了color这列的信息区分红酒和白酒了，所以呢就可以集合在一起了：

```
1 | wine_df = red_df.append(white_df)
```

其实呢，我个人更偏爱直接使用pandas的方式增加color列，一下方法是等价的：

```
1 | red_df['color'] = 'red'
2 | # 多么简洁！！！我爱大熊猫！！！
```

## \*/ 9.重命名列

接下来发现了数据很奇怪，根据检查white和red的列发现是有一列名字不同，那么我们来修改

(ps: 这节看下就好, 如果是本地下载的数据两列名字是一样的) :

```
1 # 经过尝试列名是不可变的, 只能重新分配一个列表
2 new_labels = list(red_df.columns)
3 # 先是把原先的列名搞出来
4 new_labels[6] = 'total_sulfur_dioxide'
5 # 将其中的第7个列的名称改对
6 red_df.columns = new_labels
7 # 将新的列名读入red_df中
8
9 # 好麻烦有没有, 我们可以使用rename的方法完成:
10 red_df = red_df.rename(columns = {'total_sulfer-dioxide':'total_sulfur_dioxide'})
11
12 # 高兴到飞起有木有! rename的链接:
# https://stackoverflow.com/questions/20868394/changing-a-specific-column-name-in-pandas-dataframe
```

## \*/ 10.附加数据续 (写入CSV的index参数)

这里大家请回顾一下之前讲的讲数据写入csv时候的index参数, 如果不加index = False, 默认会加一列作为行号, 对比如下:

```
wine_df.to_csv('winequality_edited.csv', index=False)
check1_df = pd.read_csv('winequality_edited.csv')
check1_df.head(3)
```

|   | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH   | sulphates | alc  |
|---|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|---------|------|-----------|------|
| 0 | 7.4           | 0.70             | 0.00        | 1.9            | 0.076     | 11.0                | 34.0                 | 0.9978  | 3.51 |           | 0.56 |
| 1 | 7.8           | 0.88             | 0.00        | 2.6            | 0.098     | 25.0                | 67.0                 | 0.9968  | 3.20 |           | 0.68 |
| 2 | 7.8           | 0.76             | 0.04        | 2.3            | 0.092     | 15.0                | 54.0                 | 0.9970  | 3.26 |           | 0.65 |

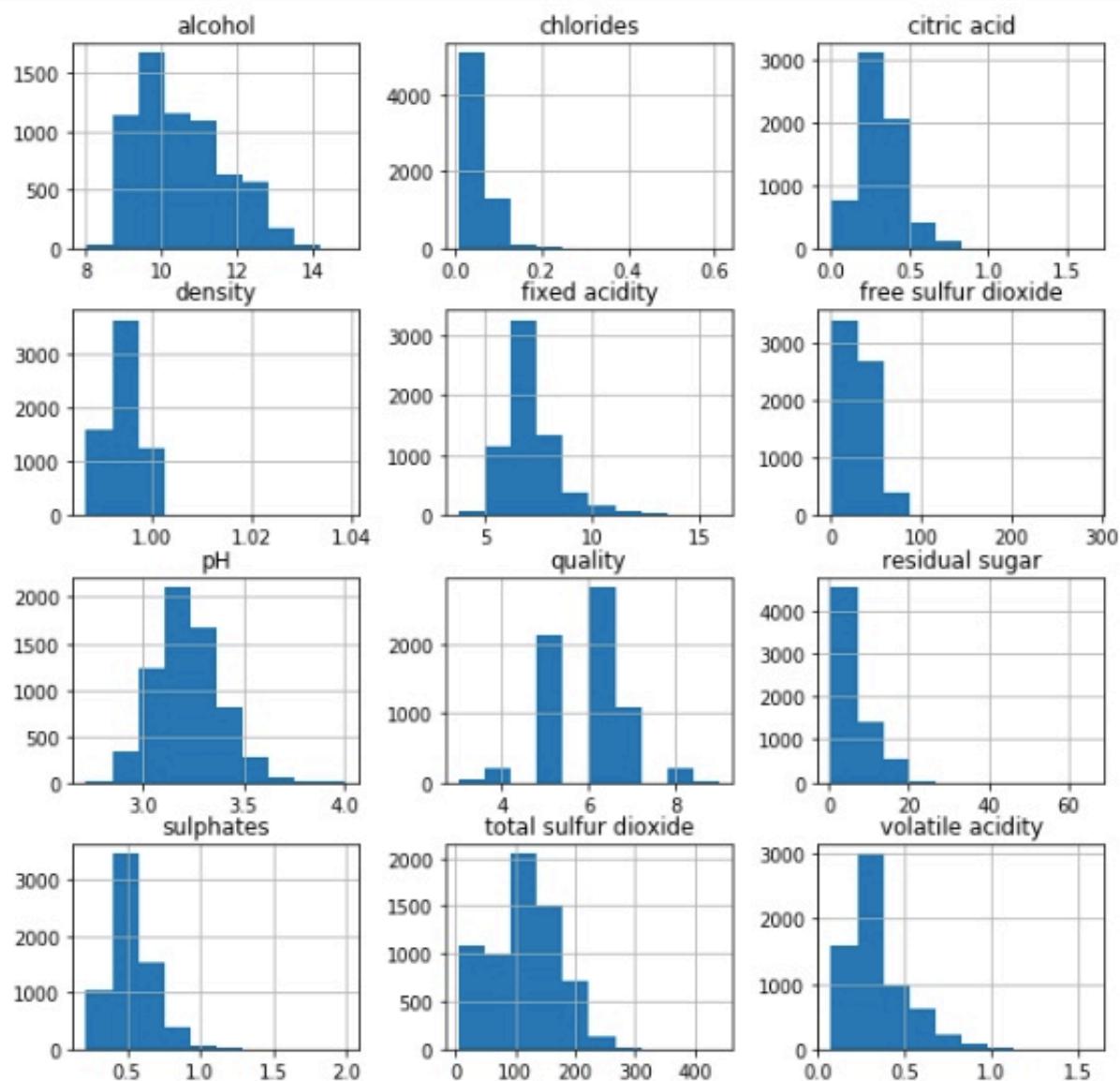
```
wine_df.to_csv('winequality_edited.csv')
check2_df = pd.read_csv('winequality_edited.csv')
check2_df.head(3)
```

|   | Unnamed: 0 | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH   | su |
|---|------------|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|---------|------|----|
| 0 | 0          | 7.4           | 0.70             | 0.00        | 1.9            | 0.076     | 11.0                | 34.0                 | 0.9978  | 3.51 |    |
| 1 | 1          | 7.8           | 0.88             | 0.00        | 2.6            | 0.098     | 25.0                | 67.0                 | 0.9968  | 3.20 |    |
| 2 | 2          | 7.8           | 0.76             | 0.04        | 2.3            | 0.092     | 15.0                | 54.0                 | 0.9970  | 3.26 |    |

## \*\*/ 11.使用可视化探索数据

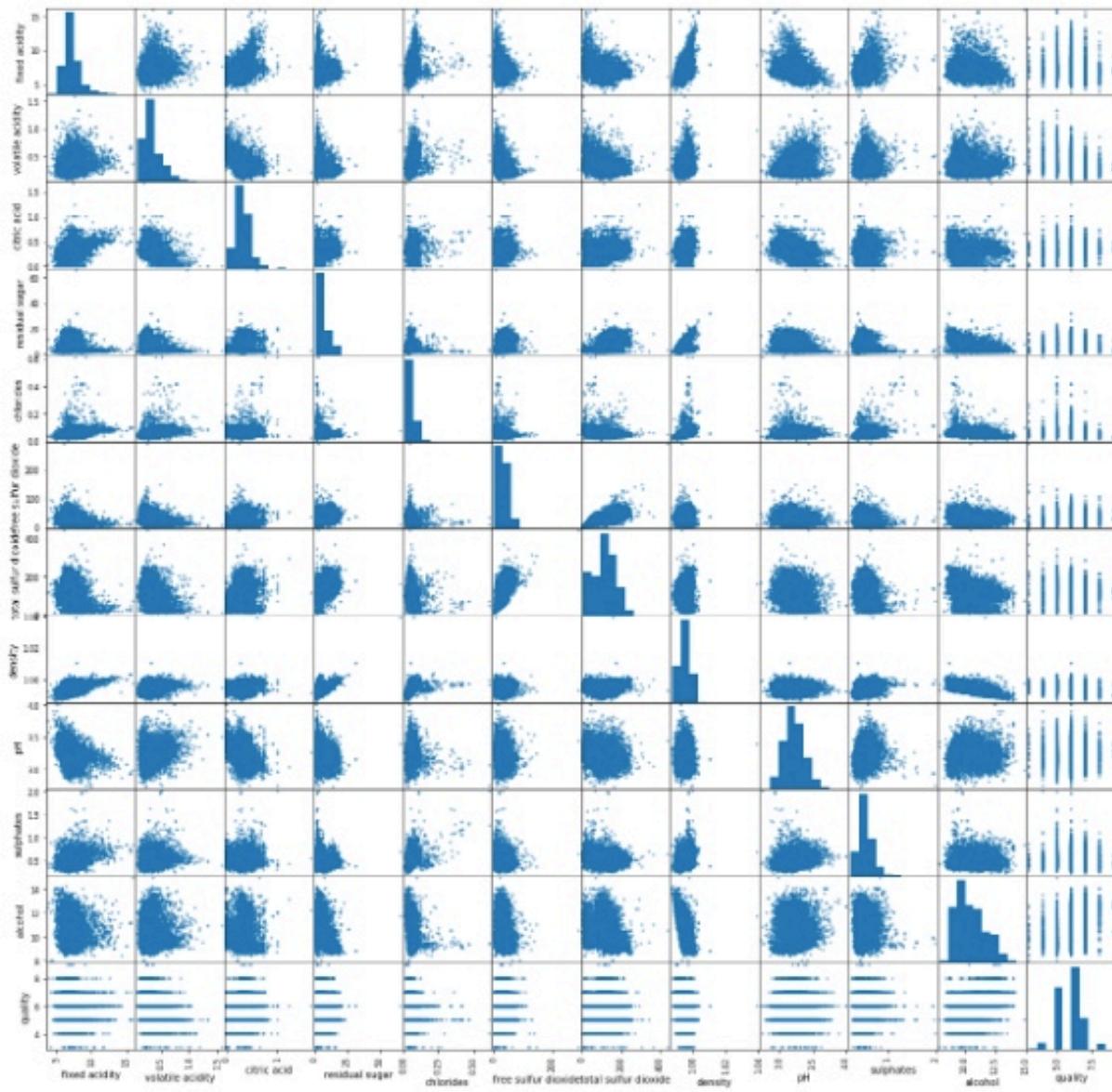
这里就是把本周课程中的作图方法使用一遍：首先是所有列的hist（我把figsize调整了下）：

```
wine_df.hist(figsize=(10,10));
```



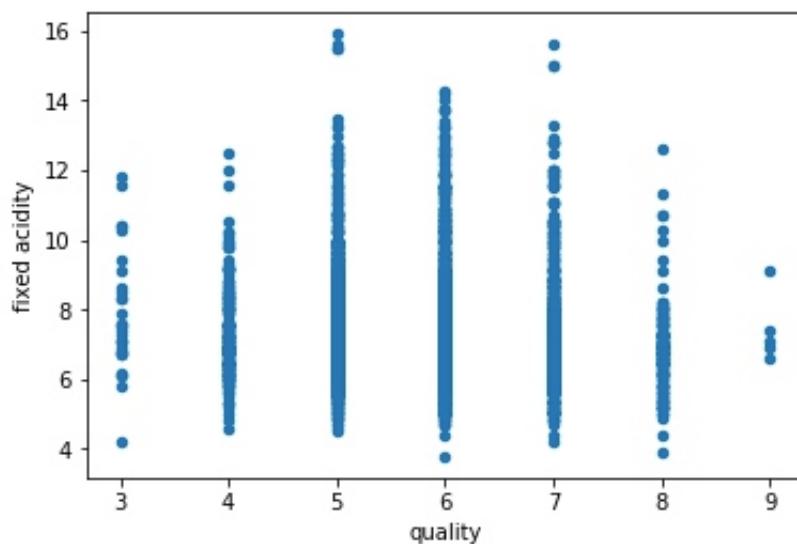
接下来是散点图，我想到了那个scatter\_matrix大招，结果由于参数太多了，执行超慢，而且太小了啥也看不出来（来个大显示器？给报销么...），放弃：

```
pd.plotting.scatter_matrix(wine_df, figsize=(20,20));
```



最后那就只能两个两个对比了：

```
wine_df.plot(x='quality', y='fixed acidity', kind='scatter');
```

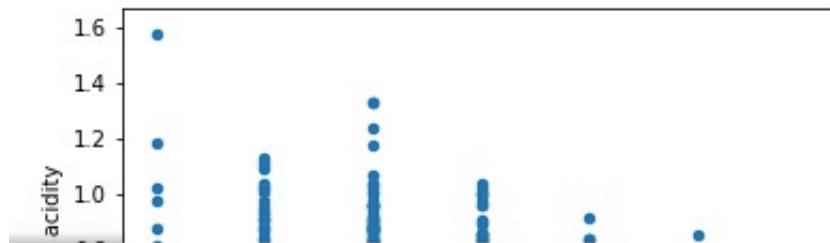
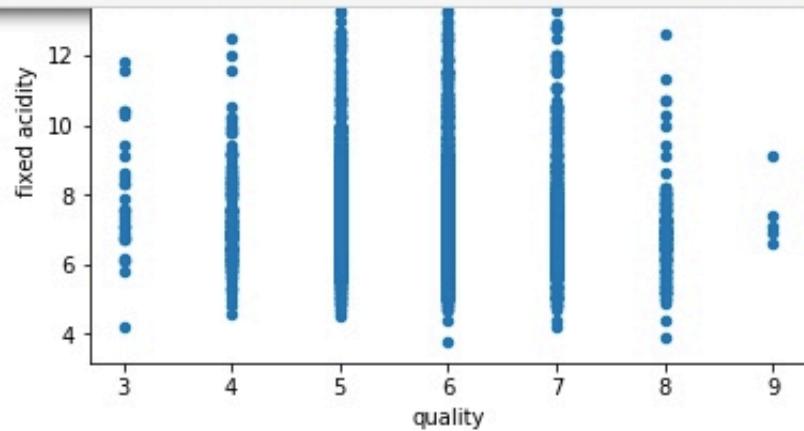


但是有这么多要对比的，肿么办呢，我来写个循环（注意输出比较多是可以滚动看的）：

```
def wine_scater(df, x, y):
    return df.plot(x, y, kind='scatter')

x = 'quality'
y = wine_df.columns[:-2]
# 通过print(wine_df.columns)观察列
# 除了最后的quality是自己, color不是数字要排除, 其他的都可以比较的
# 所以使用上面的[:-2]表示从开始到倒数第二个元素

for i in y:
    wine_scater(wine_df, x, i)
# 因为y是一个列表, 我们就可以用for in将每一个指标循环打印出来了
```



## \*/ 12.Pandas Groupby

这一节我们来使用实际数据复习下groupby函数。在上一节我们试图用散点图的方式寻找和质量相关的参数（地毯式），这一节我们换个想法：我们把数据按照质量划分成小组，看看各组之间的各项指标有什么变化：

```
wine_df.groupby('quality').mean()
```

|                | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density  |
|----------------|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|----------|
| <b>quality</b> |               |                  |             |                |           |                     |                      |          |
| 3              | 7.853333      | 0.517000         | 0.281000    | 5.140000       | 0.077033  | 39.216667           | 122.033333           | 0.995744 |
| 4              | 7.288889      | 0.457963         | 0.272315    | 4.153704       | 0.060056  | 20.636574           | 103.432870           | 0.994833 |
| 5              | 7.326801      | 0.389614         | 0.307722    | 5.804116       | 0.064666  | 30.237371           | 120.839102           | 0.995849 |
| 6              | 7.177257      | 0.313863         | 0.323583    | 5.549753       | 0.054157  | 31.165021           | 115.410790           | 0.994558 |
| 7              | 7.128962      | 0.288800         | 0.334764    | 4.731696       | 0.045272  | 30.422150           | 108.498610           | 0.993126 |
| 8              | 6.835233      | 0.291010         | 0.332539    | 5.382902       | 0.041124  | 34.533679           | 117.518135           | 0.992514 |
| 9              | 7.420000      | 0.298000         | 0.386000    | 4.120000       | 0.027400  | 33.400000           | 116.000000           | 0.991460 |

但是我又觉得这种把红酒白酒混合在一起的方式有点太混了，于是决定在groupby中多加一个color参数（注意多个参数要写成一个列表），就可以区分出来了：

```
wine_df.groupby(['quality','color']).mean()
```

|                      | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density |
|----------------------|---------------|------------------|-------------|----------------|-----------|---------------------|----------------------|---------|
| <b>quality color</b> |               |                  |             |                |           |                     |                      |         |
| 3 red                | 8.360000      | 0.884500         | 0.171000    | 2.635000       | 0.122500  | 11.000000           | 24.900000            | 0.9     |
| 3 white              | 7.600000      | 0.333250         | 0.336000    | 6.392500       | 0.054300  | 53.325000           | 170.600000           | 0.9     |
| 4 red                | 7.779245      | 0.693962         | 0.174151    | 2.694340       | 0.090679  | 12.264151           | 36.245283            | 0.9     |
| 4 white              | 7.129448      | 0.381227         | 0.304233    | 4.628221       | 0.050098  | 23.358896           | 125.279141           | 0.9     |
| 5 red                | 8.167254      | 0.577041         | 0.243686    | 2.528855       | 0.092736  | 16.983847           | 56.513950            | 0.9     |
| 5 white              | 6.933974      | 0.302011         | 0.337653    | 7.334969       | 0.051546  | 36.432052           | 150.904598           | 0.9     |
| 6 red                | 8.347179      | 0.497484         | 0.273824    | 2.477194       | 0.084956  | 15.711599           | 40.869906            | 0.9     |
| 6 white              | 6.837671      | 0.260564         | 0.338025    | 6.441606       | 0.045217  | 35.650591           | 137.047316           | 0.9     |
| 7 red                | 8.872362      | 0.403920         | 0.375176    | 2.720603       | 0.076588  | 14.045226           | 35.020101            | 0.9     |

## \*/ 13. 使用Groupby得出结论（由值生成分类数据）

当使用groupby的时候，可以在后面指定要考察的列（而不是所有列），并且可以加上as\_index = False指定不以groupby作为index（会按数字排列）：

```
wine_df.groupby('color', as_index = False)['quality'].mean()
```

| color | quality        |
|-------|----------------|
| 0     | red 5.636023   |
| 1     | white 5.877909 |

接下来的问题2有一点点复杂，我梳理一下：

- 首先数据中的pH是数值
- 在我们要考察pH level和酒质量的关系
- 既然是level那么我们就要把所有数据分为几大列，比如可以这样定义：
- 酸度水平：
  - H 高: 最低 25% 时的 pH 值
  - MH 中等偏高: 25% - 50% 时的 pH 值
  - M 中: 50% - 75% 时的 pH 值
  - L 低: 最高 75% 时的 pH 值
- 肚么办，首先我们要在数据中创建一个新列来记录酸度水平，就是acidity\_levels
- 那么我们把酸度水平的数据填上呢，pandas有个cut函数可以干这个事（把数值按照百分比归类到对应的level上）
- 代码如下，请认真读注释：

```
1 # 用 Pandas 描述功能查看最小、25%、50%、75% 和 最大 pH 值
2 wine_df['pH'].describe()
3
4 # 将min到max的5个值用做“分割”点
5 bin_edges = [2.72, 3.11, 3.21, 3.32, 4.01]
6
7 # 当然也可以偷懒了，先确定要的5个是describe输出的倒数5个，用[3:]选中
8 # 因为是series结构（有min : 2.72这种对应关系）
9 # 所以用list把输出的value（就是2.72这些值）输出成列表
10 bin_edges_smart = list(wine_df['pH'].describe()[3:].values)
11
12 # 检查下
13 print(bin_edges_smart)
14
15 # 接下来对每个酸度水平类别进行命名
16 bin_names = ['H', 'MH', 'M', 'L']
17
18 # 之后就可以创建 acidity_levels 列了
19 wine_df['acidity_levels'] = pd.cut(wine_df['pH'], bin_edges, labels=bin_
20 names)
21 # 这里注意cut方法的3个参数用法第一个参数是要比较的列wine_df['pH']，第二个是比较的区
```

```
22 | 分点就是那5个值，第三个是分类的名称
```

```
# 检查该列是否成功创建
```

```
wine_df.head()
```

输出是：

```
1 count      6497.000000
2 mean       3.218501
3 std        0.160787
4 min        2.720000
5 25%        3.110000
6 50%        3.210000
7 75%        3.320000
8 max        4.010000
9 Name: pH, dtype: float64
10
11 [2.72, 3.11, 3.21, 3.32, 4.01]
```

```
1 wine_df.head(5)
```

| residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH   | sulphates | alcohol | quality | color | acidity_levels |
|----------------|-----------|---------------------|----------------------|---------|------|-----------|---------|---------|-------|----------------|
| 1.9            | 0.076     | 11.0                | 34.0                 | 0.9978  | 3.51 | 0.56      | 9.4     | 5       | red   | L              |
| 2.6            | 0.098     | 25.0                | 67.0                 | 0.9968  | 3.20 | 0.68      | 9.8     | 5       | red   | MH             |
| 2.3            | 0.092     | 15.0                | 54.0                 | 0.9970  | 3.26 | 0.65      | 9.8     | 5       | red   | M              |
| 1.9            | 0.075     | 17.0                | 60.0                 | 0.9980  | 3.16 | 0.58      | 9.8     | 6       | red   | MH             |
| 1.9            | 0.076     | 11.0                | 34.0                 | 0.9978  | 3.51 | 0.56      | 9.4     | 5       | red   | L              |

那么最后就可以回答问题了：

```
1 wine_df.groupby(['acidity_levels'])['quality'].mean()
```

```
acidity_levels
H      5.783343
MH     5.784540
M      5.850832
L      5.859593
Name: quality, dtype: float64
```

## \*\*/ 14.15.Pandas Query

就是如何过滤你要的数据，之前是使用filter的方法，也可以使用query的方法，代码对比如下：

```
1 # selecting malignant records in cancer data
2 df_m = df[df['diagnosis'] == 'M']
3 df_m = df.query('diagnosis == "M"')
4
5 # selecting records of people making over $50K
6 df_a = df[df['income'] == '>50K']
7 df_a = df.query('income == ">50K"')
```

要想回答究竟含量高是否更高评分，我们还是要把数据分为高和低，就是按照50%为区分。这里我们使用query的方法实现：

```
1 print(wine_df['alcohol'].median())
2 # 如果使用众数mode的话会相差超多，使用mean还凑合，使用median中位数最好
3
4 # 选择酒精含量小于平均值的样本
5 low_alcohol = wine_df.query('alcohol < 10.49')
6
7 # 选择酒精含量大于等于平均值的样本
8 high_alcohol = wine_df.query('alcohol >= 10.49')
9
10 # 确保这些查询中的每个样本只出现一次
11 print(low_alcohol.shape[0])
12 print(high_alcohol.shape[0])
13 wine_df.shape[0] == low_alcohol.shape[0] \
+ high_alcohol.shape[0]
14 # 左边是全数据量，右边是拆分后的数据量和
15 # 也顺便检查下两个分组的数量
16
17 # 回答问题
18 # 这次问题的回答是为了使用query，比cut略显麻烦
19 mean_quality_high = high_alcohol['quality'].mean()
20 mean_quality_low = low_alcohol['quality'].mean()
21 print(mean_quality_high)
22 print(mean_quality_low)
```

接下来糖分的内容一模一样的方法，就不赘述了。

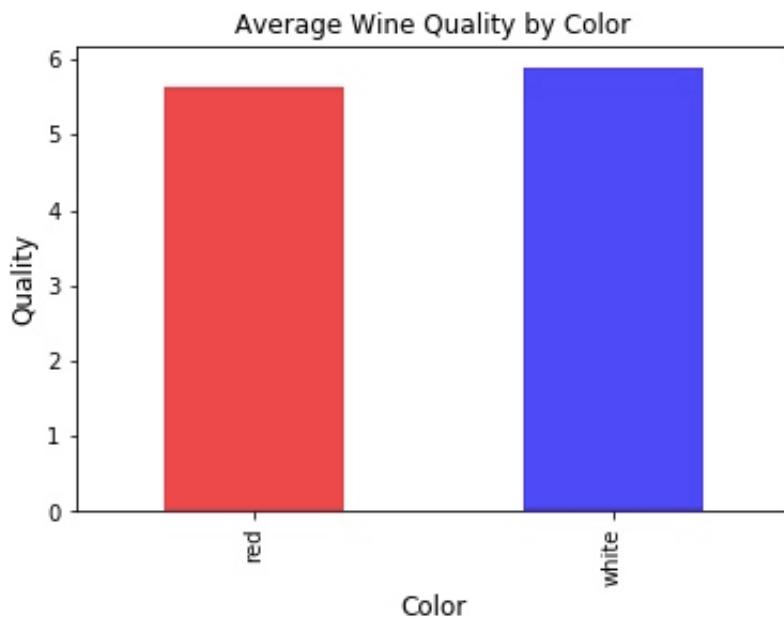
## \*\*/ 16.17.类型和质量图

这两节就可以画图了，我们还是用回groupby方法

```
1 | wine_df.groupby('color')['quality'].mean().plot(kind='bar', title='Average Wine Quality by Color', color = ['red', 'white'], alpha = .7)
```

但是为了好看，我还想引入x和y的title，代码说明：

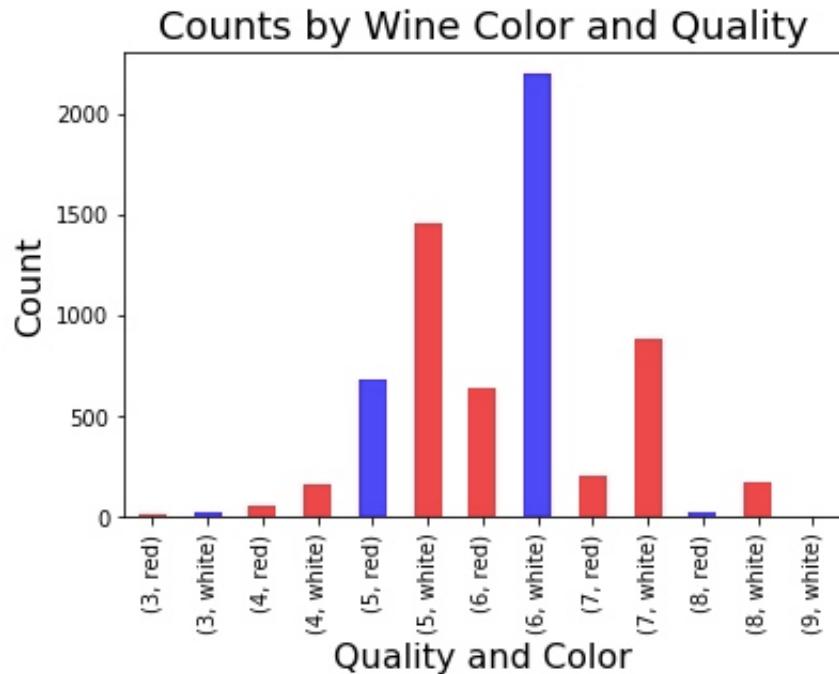
```
1 # 接下来我们要美化一下（美颜相机了解一下？）
2 # 在文档前面加上import matplotlib.pyplot as plt
3 # 这种方式是把matplotlib第三方库众的pyplot模块引入
4 # 有时候库内容很多这样可以只引入库中需要的部分
5 # 这个模块是要定义图形x和y轴时候要用的pandas中并没有集成这个功能
6 # 一同引入的还有seaborn
7
8
9 wine_df.groupby('color')['quality'].mean() \
10 .plot(kind='bar',
11       title='Average Wine Quality by Color',
12       color = ['red', 'blue'], alpha = .7);
13
14 plt.xlabel('Color', fontsize=12)
15 plt.ylabel('Quality', fontsize=12)
16 # 有了这两行就可以看到两个轴的标签了
17 # 注意这两行要放在最下面，因为plot会输出xlabel所以要在最后覆盖一下
```



接下来，我们看看能不能再加点东西，就平均值对比数据太少了，我们看看每个质量级别的对比吧(使用counts和groupby，示例使用pH数据)：

```
1 counts = wine_df.groupby(['quality', 'color']).count()['pH']
2 # 随便选了pH列，可以写循环输出所有的
3 counts.plot(kind='bar', color = ['red', 'blue'], alpha = .7)
4 plt.xlabel('Quality and Color', fontsize = 16)
5 plt.ylabel('Count', fontsize = 16)
6 plt.title('Counts by Wine Color and Quality', fontsize = 18)
```

7 | # 我把title也拆出来了为了能够指定大小，这样plot也比较短



ps:seaborn可以画出超帅气的图来，我们将在下周进行讲解，有兴趣的可以看下官方文档和样例库：

- <https://seaborn.pydata.org/>
- <https://seaborn.pydata.org/examples/index.html>

## \*/ 18. Matplotlib示例

{18}建议大家浏览并跑一遍工作空间中的文件，主要多了介绍怎么为轴的标签指定名字（比如说不是显示3、4、5而是显示质量3、质量4、质量5）

## \*/ 19.20. Matplotlib绘图

此处作为扩展了解一下作图的更复杂操作，了解即可。

Title: W7 汽车燃料经济性案例 [项目：探索数据集项目2/4]

Tags: 数据分析初级，实战项目

# W7 汽车燃料经济性案例 [项目：探索数据集项目2/4]

---

- [W7 汽车燃料经济性案例 \[项目：探索数据集项目2/4\]](#)
- [/学习地图/](#)
- [/目标1/：数据分析过程燃料经济性案例](#)
  - [/ 5.数据评估](#)
  - [/ 6.清理列标签](#)
  - [/ 7.过滤、丢空、去重](#)
  - [/ 9.修正数据1](#)
  - [/ 10.修正数据类型2](#)
  - [/ 12.使用可视化探索数据](#)
  - [/ 13.结论和可视展示 Q1](#)
  - [/ 13.结论和可视展示 Q2](#)
- [/目标2/：了解Python画图的扩展（本周扩展导学中讲解）](#)

## /学习地图/

---

本周依然是数据清理的案例讲解，数据清理是数据分析中最耗时的部分，请认真学习并自己跑明白代码。

## /目标1/：数据分析过程燃料经济性案例

---

今天的案例是根据环保局发布的数据对燃料经济性做分析。

- 燃料经济性的介绍：[https://en.wikipedia.org/wiki/Fuel\\_economy\\_in\\_automobiles](https://en.wikipedia.org/wiki/Fuel_economy_in_automobiles)
- 数据下载链接：<https://www.fueleconomy.gov/feg/download.shtml/>
- 数据Feature的说明：<https://www.fueleconomy.gov/feg/EPAGreenGuide/GreenVehicleGuideDocumentation.pdf>
- 本次案例是对比2008年和2018年汽车的能源消耗数据

- ! 注意！链接下载的是txt文件，而且是使用tab分割的。建议本案例在workspace上完成。  
如果需要自己下砸的话读入文件要注意。
- 我传送的源数据链接（最新版和working space中的有些不同）
  - [https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/files/all\\_alpha\\_08.txt](https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/files/all_alpha_08.txt)
  - [https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/files/all\\_alpha\\_18.txt](https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/files/all_alpha_18.txt)

## / 5.数据评估

注意出了import外，自己下载文件本地做的readcsv是这样的：

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5 df08 = pd.read_csv('all_alpha_08.txt', sep = '\t')
6 df18 = pd.read_csv('all_alpha_18.txt', sep = '\t')
7 # 注意原文件链接为xlsx、zip和txt（不是csv， csv课程中没给出链接）
8 # 使用read_csv读入，发现没有分列，看样子是tab分割
9 # 加个 sep = '\t'指定使用tab分割解决
10 # (mac 遇到奇怪问题下载的文件都很奇怪，可能和单位网有关)

```

除了检查na数据的数量，我们还可以检查下比例,结果看着比较帅气和整洁：：

```

1 # 看下占的百分比
2 round(nuchek / df08.shape[0], 3)
3 # round(float,n) 的作用是把float小数，保留到小数点后n位
4 # 当然了 nuchek是求出的丢失值的数量， df08.shape[0]是所有项，案例1有讲

```

|   |       |       |
|---|-------|-------|
| 1 | Model | 0.000 |
| 2 | Displ | 0.000 |
| 3 | Cyl   | 0.090 |
| 4 | Trans | 0.090 |
| 5 | Drive | 0.042 |

非空唯一值的解法，我比较懒使用循环，而且使用unique：

```

1 for i in df08.columns:
2     print(i, end = ' : ')
3     print(df08[i].nunique())

```

```
4 | # 可以使用格式化字符串来更好的显示, 略过
```

```
1 | Model : 436
2 | Displ : 47
3 | Cyl : 8
4 | Trans : 14
5 | Drive : 2
```

## / 6.清理列标签

对于判断两个数据集的列是否一样:

```
1 | # 先检查一下有没有列不同的
2 | df08.columns == df18.columns
3 | # 输出是个boolean的列表, 比较直观
4 |
5 | # 也可以用这个, 比较安静的, 因为加了.all()所以只会出一个True或者False
6 | (df08.columns == df18.columns).all()
```

```
1 | array([ True,  True,  True,  True,  True,  True, False,  True,  True, Tr
2 | ue,  True,  True,  True,  True])
3 |
4 | True
```

那么到底哪列不同呢, 我来写个循环告诉我 (懒) :

```
1 | # 把08列的名字得出来
2 | for i in df08.columns:
3 |     if i not in df18.columns:
4 |         print(i)
5 |
6 | # 把18列的名字得出来
7 | for i in df18.columns:
8 |     if i not in df08.columns:
9 |         print(i)
```

```
1 | Sales Area
2 | Cert Region
```

那么我们来替换一下, 而且顺手把不同形式的列名修改一下 (有的是空格, 有的是下划线) :

```

1 # 将08的替换为18的
2 df08.rename(columns =
3             lambda x: x.replace('Sales Area', 'Cert Region'),
4             inplace = True)
5 # 这里使用的是lambda x:
6 # 就是对columns执行后面的操作 x.replace
7 # 而x.replace的操作是把后面小括弧中的前面替换成后面
8
9 # 还有就是把列中的空格替换成下滑线
10 # 因为很多处理要对空格判断，不要有比较好
11 # 全部变为小写是习惯，了解一下吧
12 df08.rename(columns=lambda x:
13               x.strip().lower().replace(" ", "_"),
14               inplace=True)
15 # .strip是去除单词首尾的空格
16 # .lower是变为小写
17 # .replace是把小括弧里的做替换
18 df08.columns

```

着节是项目3的开始，看完之后应该对整体的课程设计和需要的SQL知识（项目1）、python知识（项目2）有所了解。并且提供了不少额外资源的链接。

此处最后一个链接是一本超好的书，可以陪你走到微学位完成，请不要错过（鸟文的，中文的也有），如果链接不可用，可以试试我搬运的：

## / 7.过滤、丢空、去重

在案例1中有讲，注意unique、drop、isnull、dropna等的用法

## / 9.修正数据1

是这个项目的重点，之前内容没有讲过，我们先来看下cyl的数据其实是这样的：(6 cyl) 其实就是发动机的缸数，越大越腻害，而且只有偶数整数。所以我们要把它变成6，注意输出的上下对比：

```

1 # 使用str的extract检索里面的数字
2 print(df08['cyl'].value_counts())
3 df08['cyl'] = df08['cyl'].str.extract('(\d+)').astype(int)
4 # (\d+)是匹配任意数字的意思，有兴趣的可以看看reg或者回顾week2格式化字符串
5 print(df08['cyl'].value_counts())
6 type(df08['cyl'][0])

```

```
1 (6 cyl)    836
2 (4 cyl)    582
3 (8 cyl)    516
4 (5 cyl)    113
5 (12 cyl)   60
6 (10 cyl)   29
7 (2 cyl)    4
8 (16 cyl)   2
9 Name: cyl, dtype: int64
10
11 6    836
12 4    582
13 8    516
14 5    113
15 12   60
16 10   29
17 2    4
18 16   2
19 Name: cyl, dtype: int64
```

结果2018年的文件到不是字符，但是小数，难不倒我们，使用astype：

```
1 print(df18['cyl'].value_counts())
2 df18['cyl'] = df18['cyl'].astype(int)
3 #貌似astype不能使用inplace参数
4 print(df18['cyl'].value_counts())
5 type(df18['cyl'][0])
```

```
1 4.0      1210
2 6.0      843
3 8.0      418
4 12.0     51
5 3.0      42
6 10.0     16
7 2.0      4
8 5.0      4
9 16.0     2
10 Name: cyl, dtype: int64
11 4      1210
12 6      843
13 8      418
14 12     51
15 3      42
16 10     16
```

```
17 5      4
18 2      4
19 16     2
20 Name: cyl, dtype: int64
```

## / 10.修正数据类型2

到了这里，发现有的是混合动力车，使用能源、里程、污染什么的使用 a\b 记录的，据说设计这个数据结构的人已经被数据分析师砍死了，悲剧啊，但是还要处理下。有些复杂，看懂即可：

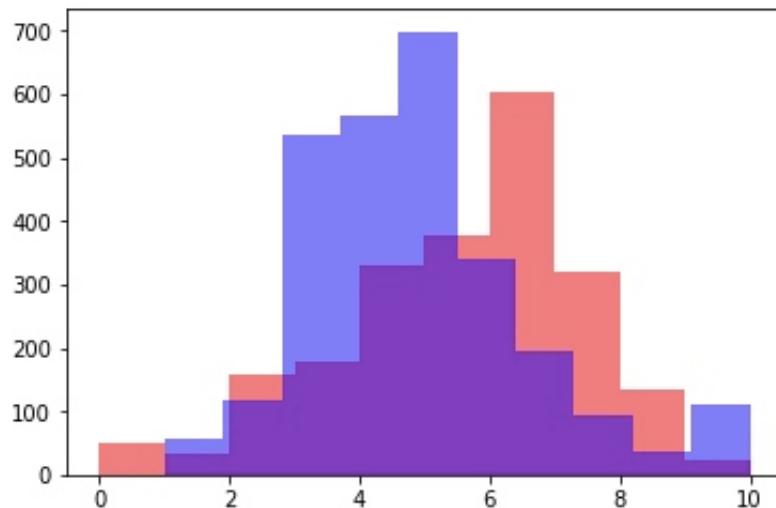
```
1 # 找出含有 / 的部分
2 # 也可以使用query方式
3 hb_08 = df08[df08['fuel'].str.contains('/')]
4 print(hb_08.shape)
5
6 # 拆分为两列，使用copy否则会修改原数据
7 # create two copies of the 2008 hybrids dataframe
8 df1 = hb_08.copy() # data on first fuel type of each hybrid vehicle
9 df2 = hb_08.copy() # data on second fuel type of each hybrid vehicle
10
11 # 确定要拆分的列
12 # columns to split by "/"
13 split_columns = ['fuel', 'air_pollution_score', 'city_mpg', 'hwy_mpg', 'cmb_mpg', 'greenhouse_gas_score']
14
15 # apply split function to each column of each dataframe copy
16 for c in split_columns:
17     df1[c] = df1[c].apply(lambda x: x.split('/')[0])
18     df2[c] = df2[c].apply(lambda x: x.split('/')[1])
19         # lambda对每个带/的进行分割，分别赋值第1个和第2个拆分元素
20 print(df1.head(3))
21 # 看下是否成功
22
23
24 dfnewrows = df1.append(df2)
25 print(df1.shape)
26 print(dfnewrows.shape)
27 dfnewrows.head(3)
28 # 检查数量
29
30 # 更新df08
31 print(df08.shape)
32 print(hb_08.index)
33 # 检查下要删除的行
```

```
34 df08.drop(hb_08.index, inplace = True)
35 df08 = df08.append(dfnewrows, ignore_index = True)
36 # append 没有inplace参数
37 df08.shape
38
39 # 现在可以转换了(之前由于 \ 的原因, 转换会报错)
df08.air_pollution_score = df08.air_pollution_score.astype(float)
```

## / 12. 使用可视化探索数据

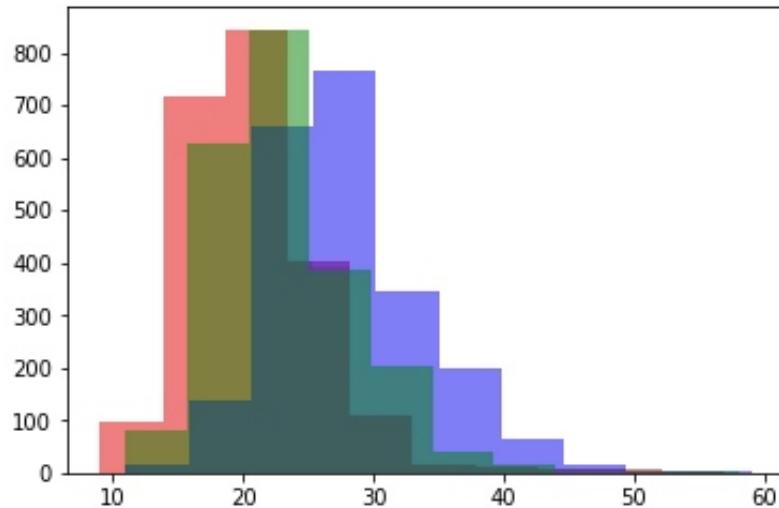
画图画图画画图，首先看看环保，，，，温室气体指标评分，10年得分降低了，到底是低好还是高好呢，需要大家去数据源自己研究，我加了个透明度，看着还不错：

```
1 p08=plt.hist(df08['greenhouse_gas_score'],color='r', alpha = 0.5)
2 p18=plt.hist(df18['greenhouse_gas_score'],color='b', alpha = 0.5)
3 plt.show()
4 # 哇偶，可以看出18年的greenhouse_gas_score有所下降！
5 # 企鹅宝宝，，，
```



再看看city、hwy、cmb的里程续航啥关系：

```
1 city=plt.hist(df08['city_mpg'],color='r', alpha = 0.5)
2 hwy=plt.hist(df08['hwy_mpg'],color='b', alpha = 0.5)
3 cmb = plt.hist(df08['cmb_mpg'],color='g', alpha = 0.5)
4 plt.show()
5 # 从数据可以推断出city是在城市的速度 (miles per hour)
6 # hwy是高速的, cmb是联合的
```



## / 13. 结论和可视展示 Q1

开始回答问题喽，对于这个案例，都做到这了，给个福利，简单扩展下，便于你去撩人，假装有知识、有担当、有爱心、有态度的上进青年一枚：

- 这部分feature pdf中没有特殊说明，bing出来：
- CNG 压缩天然气
- diesel 柴油
- Gasoline 汽油
- ethanol 乙醇
- gas 天然气

瞎想的（可以忽悠不懂的）

1. CNG是压缩天然气，另外一种是液化天然气LNG（C代表compressed，L代表liquid，LNG可以制造CNG，NG是不经压缩的gas）。后者更环保因为是液态，行驶也更远。但是前一种更加普遍（因为可以使用现在的汽油汽车改装，难道混合动力的都是这个货？）
2. 乙醇更环保，天然气是混合物，里面可能包含一些S,N这些元素。当这些元素燃烧时会产生污染气体。乙醇燃烧的产物是二氧化碳和水。再者乙醇是可再生资源，所以目前有很多国家在汽油中添加乙醇，已减少汽油的使用量。
3. 在18年的数据中还有电力汽车（我的特斯拉啊，看着就帅！，，，，这模型还挺重的）马克思，，哦不马斯克的Solar City了解一下。
4. 综上所述，由于各年的value不同，我们就分为两个阵营进行比较（谁说要加权来的，你出来，我保证，，，不打死你，，，）Gasoline和diesel作为传统能源，其他都粗暴的归为清洁能源。

言归正传，我们先看看各使用能源分类，再算算2008到2018的比例是否有上升，就知道大家是否在努力了

```
1 | df08.groupby(['fuel']).count()
```

|          | model | displ | cyl  | trans | drive | cert_region | veh_class | air_pollution_score |
|----------|-------|-------|------|-------|-------|-------------|-----------|---------------------|
| fuel     |       |       |      |       |       |             |           |                     |
| CNG      | 2     | 2     | 2    | 2     | 2     | 2           | 2         | 2                   |
| Gasoline | 2067  | 2067  | 2067 | 2067  | 2067  | 2067        | 2067      | 2067                |
| diesel   | 7     | 7     | 7    | 7     | 7     | 7           | 7         | 7                   |
| ethanol  | 66    | 66    | 66   | 66    | 66    | 66          | 66        | 66                  |
| gas      | 65    | 65    | 65   | 65    | 65    | 65          | 65        | 65                  |

```
1 | cleanlist = ['cng', 'ethanol', 'gas', 'electricity']
2 | # 先制定清洁能源的备选
3 |
4 | def cleanratio(df):
5 |     cnumber = 0
6 |     fuel_list = list(set(df['fuel'].values))
7 |
8 |     for i in fuel_list:
9 |         if i.lower() in cleanlist:
10 |             # 本例子中i不要变化，因为后面还要根据i来匹配
11 |             cnumber = cnumber + df[df['fuel'] == i].shape[0]
12 |     return round(cnumber / df.shape[0], 4)
13 |     # 使用round控制小数的精度: round(你要显示的数, 你要显示的小数位数)
14 |
15 | print(cleanratio(df08))
16 | print(cleanratio(df18))
17 | # 可见从6%增长到了9%
```

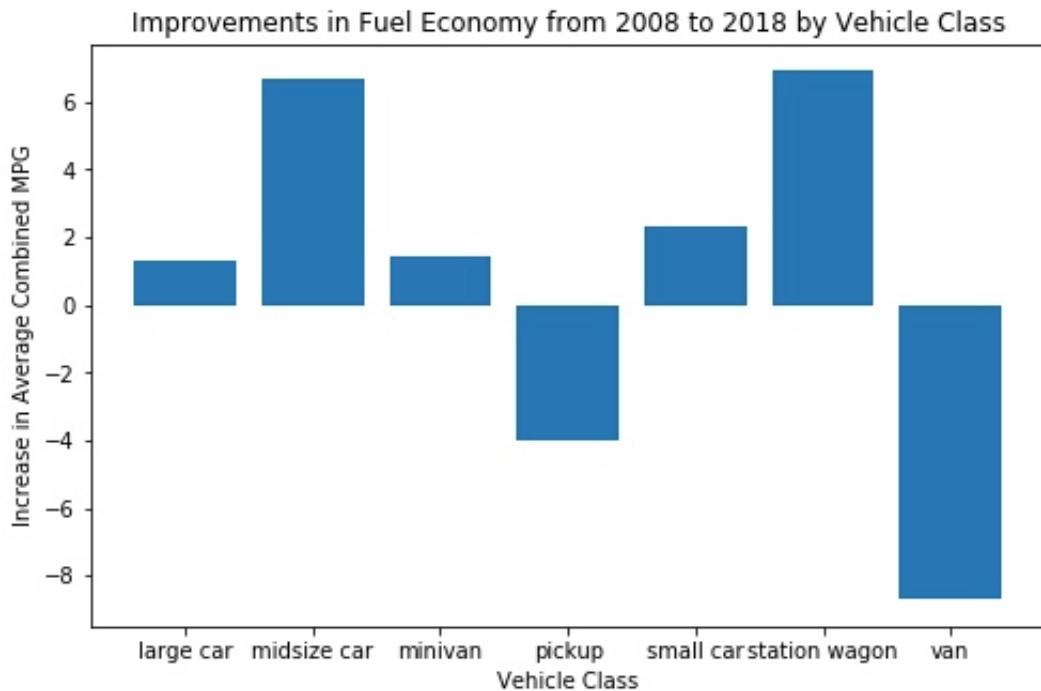
```
1 | 0.0603
2 | 0.0924
```

## / 13. 结论和可视展示 Q2

这个问题也不错的：各车辆类别（veh\_class）在燃料经济性方面的改进（mpg 的增长）是多少？其实还是使用groupby进行数据的真理，再引入mean看每个种类的平均值就可以了，看懂即可：

```
1 | # only plot the classes that exist in both years
```

```
2 inc.dropna(inplace=True)
3 plt.subplots(figsize=(8, 5))
4 plt.bar(inc.index, inc)
5 plt.title('Improvements in Fuel Economy from 2008 to 2018 by Vehicle Clas
6 ss')
7 plt.xlabel('Vehicle Class')
8 plt.ylabel('Increase in Average Combined MPG');
# 美化美化，有兴趣不？
```



后面的问题就可以不看了，有个merge把两个数据融合的内容，但例子有点不好理解，大家～到此为止就可以了！加油加油！又从头到尾看过了一个数据分析，记得去喷一下呦。

## /目标2/：了解Python画图的扩展（本周扩展导学中讲解）

Title: W7 Plus Matplotlib简介

Tags: 数据分析初级, 实战项目

# W7 Plus Matplotlib简介

---

- [W7 Plus Matplotlib简介](#)
- [/目标2/: 了解Python画图的扩展包](#)
  - [/ Matplotlib是啥东东](#)
  - [/ 画图的组织方式](#)
  - [/ 画图的细节](#)
  - [/ 做图标准代码模版](#)
  - [/ 使用Pandas做图](#)
  - [/ 后续学习](#)
  - [/ 资料](#)

## /目标2/: 了解Python画图的扩展包

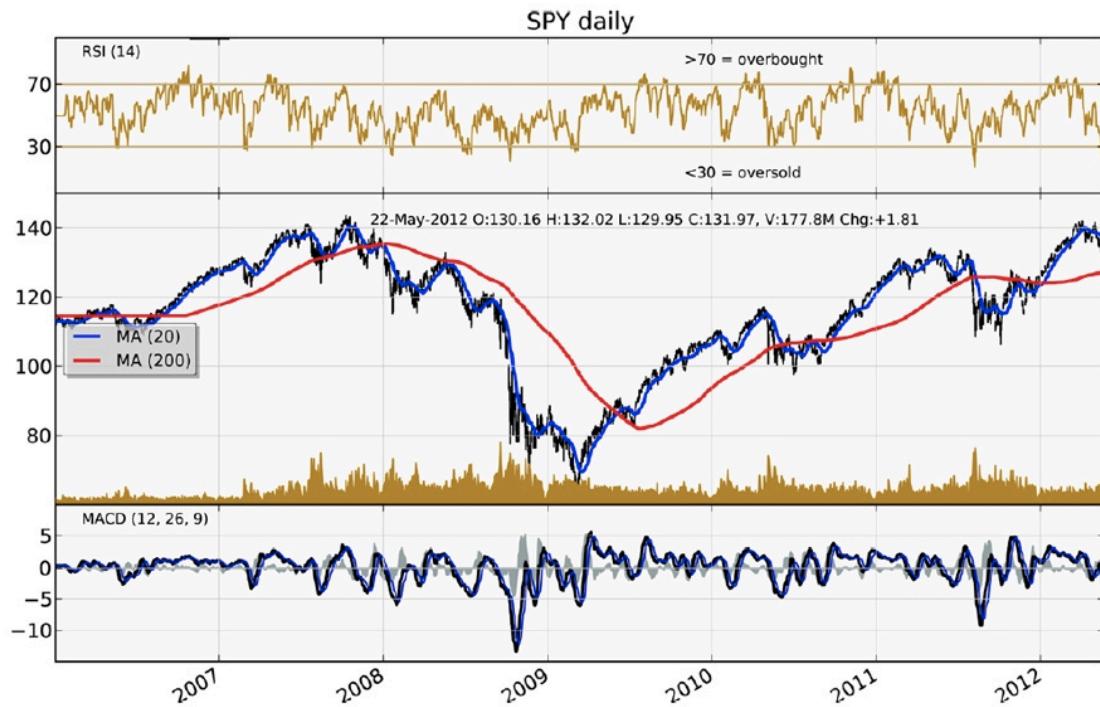
---

Python有很多可视化工具，matplotlib是一个用于创建出版质量图表的桌面绘图包，比较老，也比较通用，无论你用什么做图，都几乎脱不开干系。其他的seaborn、ggplot也都各有特点，或者是d3更是可以建立交互可视化。建议Python可视化先学好matplotlib，再根据需要扩展。

## / Matplotlib是啥东东

---

Mat可以画出很复杂的图形，比如这个金融曲线图（本文多图和例子来自《利用Python进行数据分析》）：



我们常用的pandas中集成了一些matplotlib（好长，简称plot）的函数，但是如果要使用高级功能的画就要学matplotlib了。一般matplotlib的pyplot中包含了大多数API函数，比如每次都用的plot，引入的方式是：

```

1 | import matplotlib.pyplot as plt
2 | import numpy.random
3 | %matplotlib inline
4 | # 在jupyter中要有%这行才能在文档中画出图来

```

## / 画图的组织方式

要想在用plt画图，你要先准备一张画纸，这张纸就是figure，之后再在这张纸上画上你要的图们，这些图就是subplot。在逻辑上一张纸上可以画出啊很多图来，全看你怎么定义，比如：

```

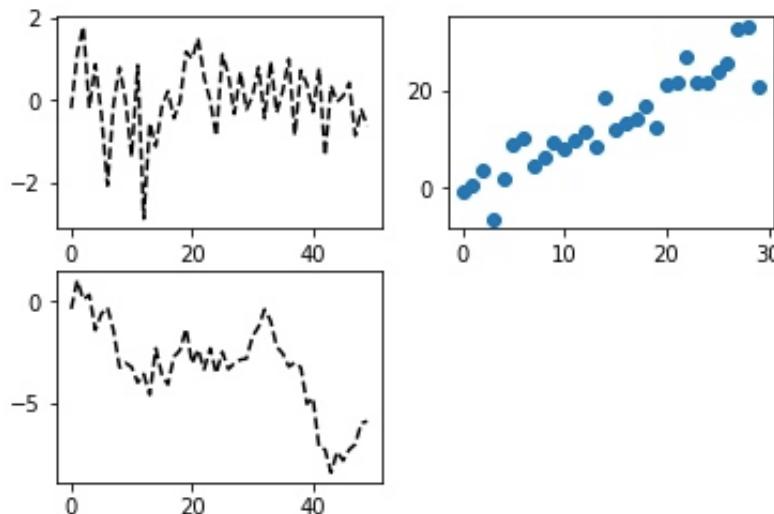
1 | fig = plt.figure()
2 | # 先定义一张画纸
3 |
4 | ax1 = fig.add_subplot(2,2,1)
5 | # 建立一个sub1图像
6 | # 把fig切成2*2的分割（就是一个田字）
7 | # 把sub1画在第1个分割里（就是左上角的那个）
8 | plt.plot(numpy.random(50), 'k--')
9 | # 我们画个50个数的随机值
10 | # plt.plot()实际上会通过plt.gca()获得当前的Axes对象ax，然后再调用ax.plot()方法实

```

```

11 现真正的绘图。
12
13 ax2 = fig.add_subplot(2,2,2)
14 # 定义了画纸上的第2个图
15 # 注意如果画纸上没有画图的话，相应的位置是空的
16
17 ax3 = fig.add_subplot(2,2,3)
18 # 定义了画纸上的第3个图
19 plt.plot(np.random.randn(50).cumsum(), 'k--')
20 # 我们画个50个数的随机值的累积值
21 # k--是告诉matplotlib绘制黑色虚线图
22
23 ax2.scatter(np.arange(30), np.arange(30) + 3 * randn(30))
24 # 注意这个sub2的图形是在上面这行加上的指定加入的
# 如果注释掉了的话，sub2的图是空的

```



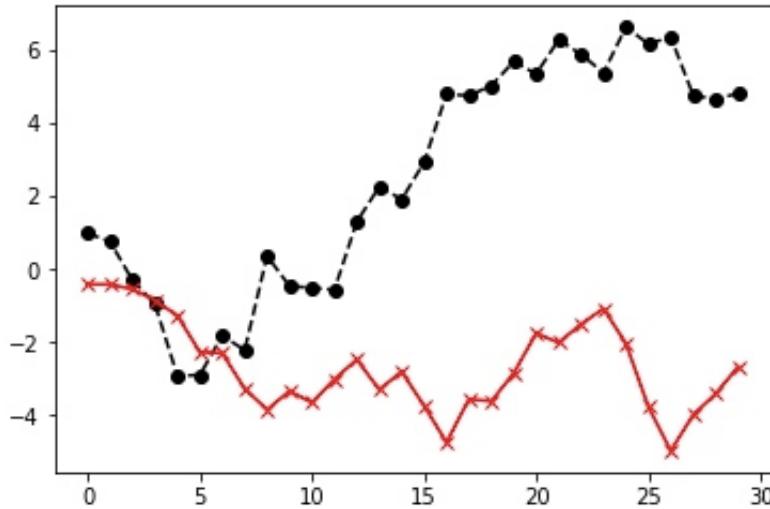
## / 画图的细节

matplotlib是可以制作出出班级的图的，有很多参数可以调整图像，比如线的样式和颜色：

```

1 # 那么接下来就可以指定细节了
2 # 我们以单一sub为例子
3 # 注意如果上来就plt.plot()的话，会默认建好一个fig和对应的sub
4 plt.plot(randn(30).cumsum(), 'ko--')
# 和上面例子一样这是原始30个随机数的累积值
# 线型是黑色 (k) 端点是黑色圆 (o) 线为虚线 (--)
5
6 plt.plot(randn(30).cumsum(), 'rx-')
# 如果再加入内容还是会显示在同一个画图上
7
8

```

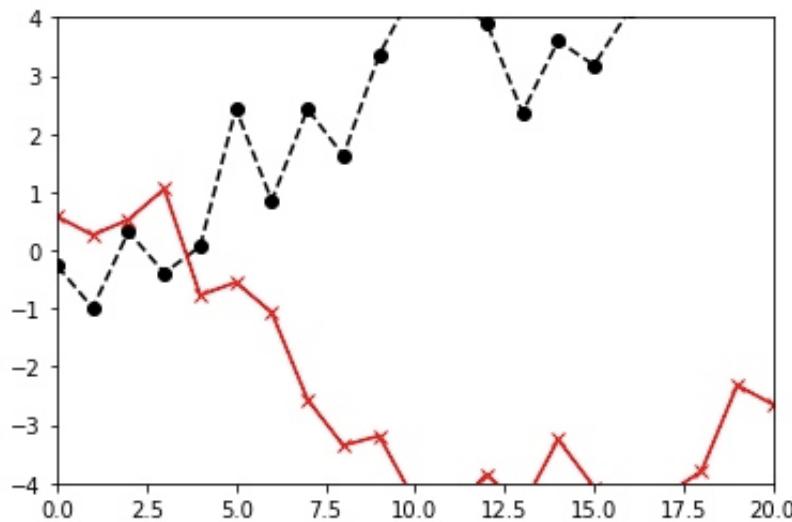


也可以限制x和y的显示范围，再上面的代码中增加：

```

1 | plt.xlim([0, 20])
2 | plt.ylim([-4, 4])
3 | # 通过xlim和ylim限制x轴和y轴的范围
4 | # 还有一个sharex和sharey参数，调节xlim将会影响所有subplot

```



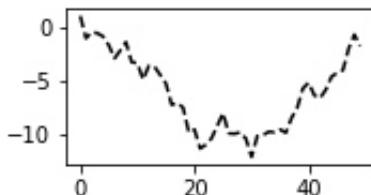
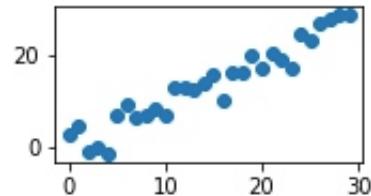
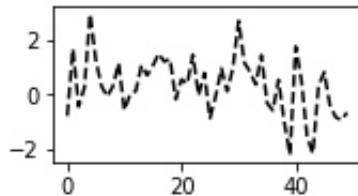
最后，一个fig中的ax的边距也是可以设定的：

```

1 | fig = plt.figure()
2 | sub1 = fig.add_subplot(2,2,1)
3 | plt.plot(randn(50), 'k--')
4 | sub2 = fig.add_subplot(2,2,2)
5 | sub3 = fig.add_subplot(2,2,3)
6 | plt.plot(np.random.randn(50).cumsum(), 'k--')
7 | sub2.scatter(np.arange(30), np.arange(30) + 3 * randn(30))
8 |
9 | plt.subplots_adjust(left=1, bottom=None, right=2, top=None,

```

```
10 |                         wspace=1, hspace=1)
11 | # plt.subplots_adjust可以定义fig中sub的间隔
```



## / 做图标准代码模版

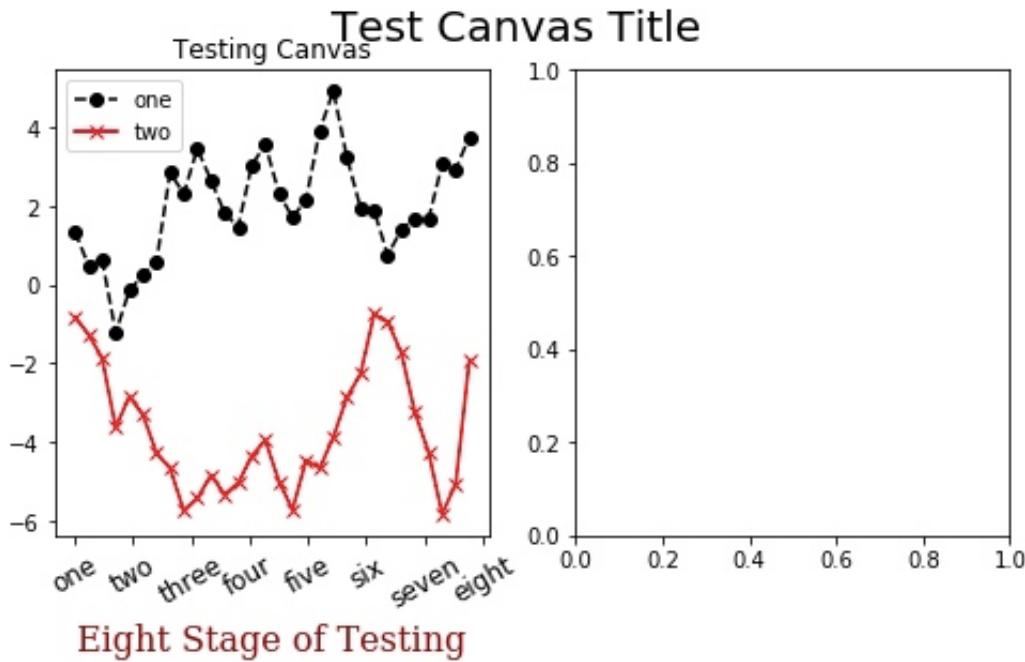
matplotlib做图其实非常强大，但也略显复杂，其实熟悉以后就还好了。如果你只想研究一个python可视化，那就用mat因为绕不开，而且在会了以后，也不是特别的复杂，可以在以后有专项的时候再扩展别的（ps：如果用matplotlib的话也可以import seaborn，即使只用mat，也会把一些图的细节变好看一点）

```
1 | # 比较标准的写法 (主观)
2 | # 1-先定义画纸fig
3 | fig=plt.figure(figsize=(8,4))
4 | # # 加了个figsize比较好看
5 | # 2-定义子图名和位置sub1
6 | ax = fig.add_subplot(1, 2, 1)
7 | ax2 = fig.add_subplot(1, 2, 2)
8 | # 3-画子图名，我习惯带上ax而不是默认plt
9 | ax.plot(randn(30).cumsum(), 'ko--', label = 'one')
10 | ax.plot(randn(30).cumsum(), 'rx-', label = 'two')
11 | # # 要加label = 才能在后面调用legend图例
12 | # 4-设定xlim,注意这里要用plt所以位置要在你需要的sub下面，此处略过
13 | # plt.xlim([0,25])
14 | # 5-设定x轴的显示位置和标签
15 | ax.set_xticks(np.linspace(0,30,8))
16 | # # 先设定xticks是显示几个点，和显示的范围
17 | # # 这里是0到30 (因为数据就是30个点)，平均点8个点
18 | # # np.linspace就是干这个的，感兴趣可以搜索，也可以直接给个列表
19 | ax.set_xticklabels(['one', 'two', 'three', 'four', 'five', 'six', 'seven',
20 |   'eight'],
```

```

21                                     rotation = 30, fontsize = 'large');
22 # # 再接下来就可以设定在这些点上要显示的字和样式了
23 font = {'family' : 'serif',
24         'color' : 'darkred',
25         'weight' : 'normal',
26         'size'   : 16,
27     }
28 ax.set_xlabel('Eight Stage of Testing', fontdict = font)
29 # # 最后是设定轴的名字, 可以用fontdict传入一个字典
30 # 6-设定画图title
31 ax.set_title('Testing Canvas')
32 fig.suptitle('Test Canvas Title', fontsize=20)
33 # # ax用set_title
34 # # fig用suptitle
35 # 因为咱们的图只有一个ax所以两个title就重合了
36 # 7-设定legend
37 ax.legend(loc = 'best')
38 # # best是自己选择最不碍事的位置
39 # 8-将图标保存为文件
40 plt.savefig('testmat.png')
41 # # 注意svg是一种无损矢量图, 拖入到浏览器能打开, 一般png就好了
# # savefig还有dpi和其他参数

```



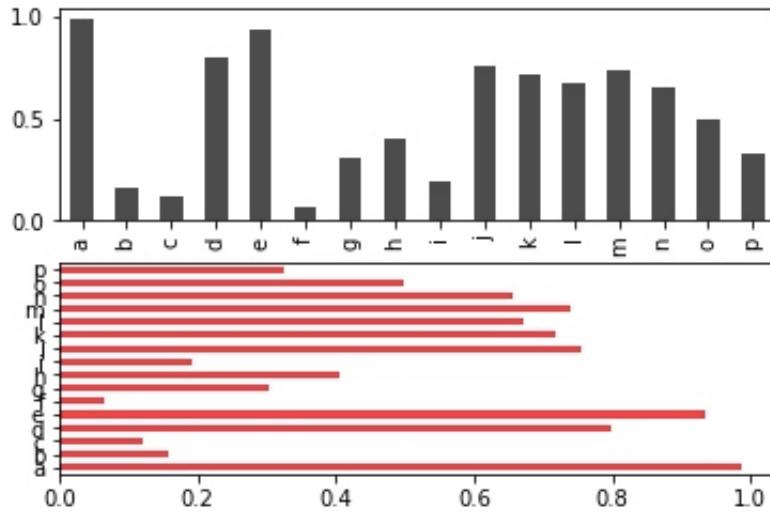
## / 使用Pandas做图

Pandas的做图实际上是集成了matplotlib完成的，用法差不多看看就会了，好处是不用加载mat库，而且可以结合pandas的一些数据功能，坏处么就是一些谢姐需要使用mat的功能定义，我们看段例子（选学，还是那句话，如果学一个就学mat）：

```

1 # 使用pandas的plot语法大同小异，其实pd也是给matplotlib处理的
2 fig, axes = plt.subplots(2,1)
3 # 设定fig和axes
4 data = pd.Series(np.random.rand(16), index = list('abcdefghijklmno'))
5 # 生成16个随机数并指定index
6 data.plot(kind = 'bar', ax = axes[0], color = 'k', alpha = 0.7)
7 data.plot(kind = 'barh', ax = axes[1], color = 'r', alpha = 0.7)
8 # 在ax = axes中指定输出图的位置
9 # 如果结合stacked和df的value_count可以画出挺好看的堆积图

```



## / 后续学习

Python的可视化内容也是不少的，作为刚开始接触请记住以下几点：

- 从matplotlib开始学，先达到能够完成项目的程度！
- 先把matplotlib学熟了，别因为觉得别的可视化简单就换，绕不过mat
- 使用时可以一同倒入seaborn库，即使不用
- matplotlib figure ax title limit 这些词能帮你搜索出需要的信息。再次推荐google（科学 webing）和bing（可以显示国际搜索结果）会帮到你很多
- 遇到了pandas制图的话搜索词变换一下，并且无论是mat 还是 pandas 再加上需要的图形 hist bar 等等过就能得到很多信息
- matplotlib的核心是一套由对象构成的绘图API。由John D. Hunter发起的（John D. Hunter由于癌症于2015年过世），完全免费！感谢John！ - 首先了解mat的结构和语法，剩下的世界尽情去探索吧，少年！

## / 资料

- 首先是官网，里面有很多examples，可以看看代码：<https://matplotlib.org/>

- 翻到一个很详细的mat说明: <https://liam0205.me/2014/09/11/matplotlib-tutorial-zh-cn/>
- 上面的英文原版: <http://www.labri.fr/perso/nrougier/teaching/matplotlib/>
- 数据科学Python入门的老鼠书: [https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/python\\_for\\_data\\_analysis.pdf](https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/python_for_data_analysis.pdf)
  - Uda课程中提供的英文版，我搬运了一下 (aws s3经常被墙的)
  - 绘图的再第8章，其他的也很好
  - 中文版有电子版（多看阅读有），但纸质书貌似第二版



Title: W8 项目实战 [项目: 探索数据集项目3/4]

Tags: 数据分析初级, 实战项目

# W8 项目实战 [项目: 探索数据集项目3/4]

---

- W8 项目实战 [项目: 探索数据集项目3/4]
- /学习地图/
  - / 项目路径
  - / 项目推进
  - / 本周重点
- /目标1/: 数据整理 (TMDB数据示例)
  - / 1.准备
  - / 2.读入数据
  - / 3.检查数据
  - / 4.清理数据
- /目标2/: 探索性数据分析 (TMDB数据示例)
  - / 研究问题1 - 几个数字值的关系如何
  - / 研究问题2 - 是否有Website和Feature的关系
- /目标3/: 得出结论 (TMDB数据示例)
  - / 数据集说明
  - / 研究问题1 - 几个数字值的关系如何
    - // 问题观察和结论
    - // 问题的图形展示
    - // 思考和建议
  - / 研究问题2 - 是否有Website和Feature的关系
- /彩蛋/

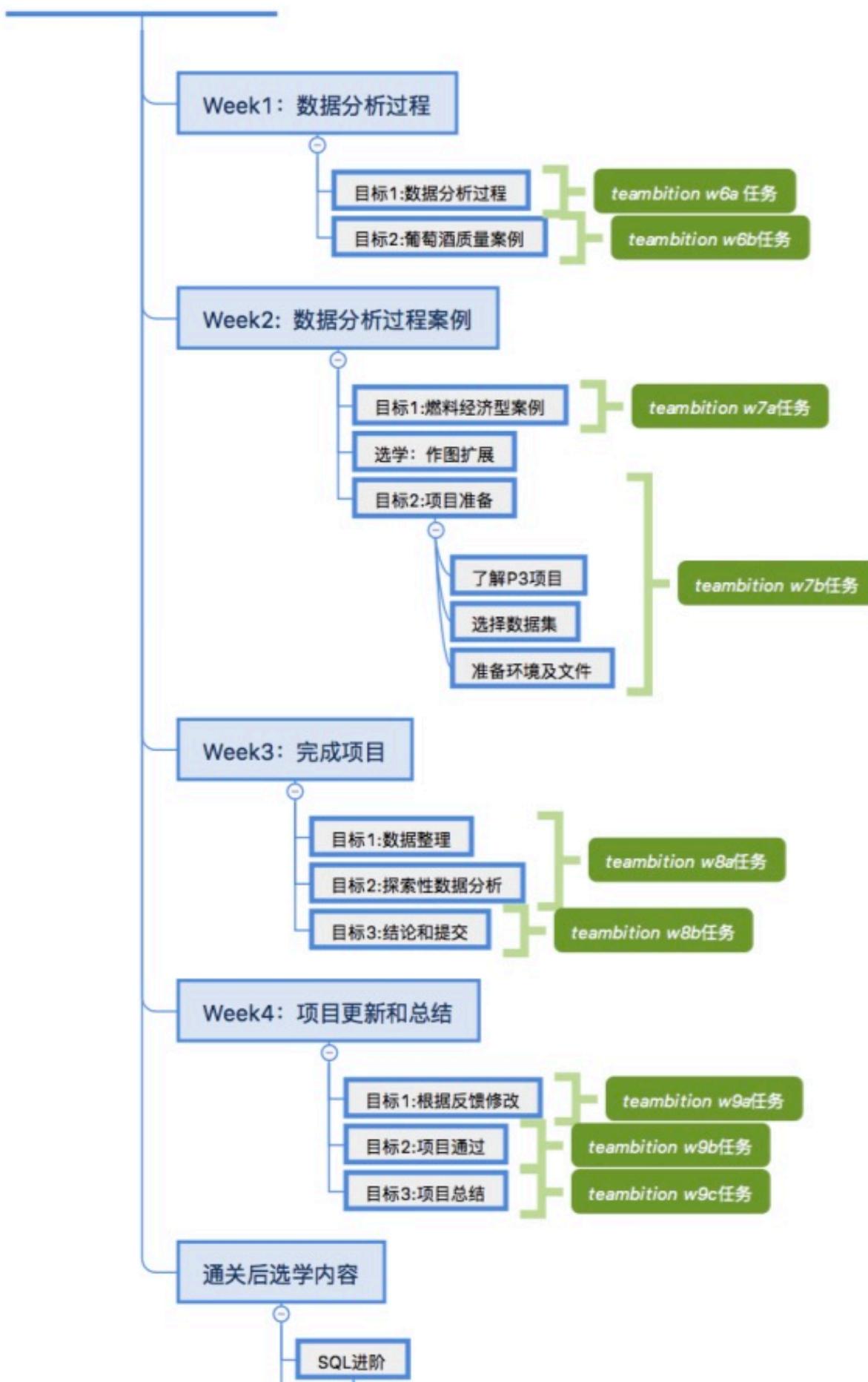
## /学习地图/

---

本周是数据分析课项目3的Part3, 我们要开始做探索数据集项目了, 兴奋不! 其实经过前两周的洗礼, 对于数据集的探索, 我们已经比较熟悉了: 除了课程的内容, 我们还已经完成了两个案例的分析, 也就是说这周我们做探索数据集项目时, 已经是第4遍比较全面的完成这个流程了, 加油吧, 少年们!

# / 项目路径

## 探索数据集





## / 项目推进

### Week1要求（已完成）：

- 完成项目简介部分
- 这次项目中有4个数据可以选择（新手不建议SQL足球那个），请根据项目说明，选一个可心的呦/项目数据说明/

### Week2要求（已完成）：

- 学习红酒质量分析案例
- 学习汽车燃料经济学案例

### Week3要求（肝！少年！）：

- 浏览本导学文件
- 做完项目中的练习
- 根据导学文件，按照项目提示，一步步的完成项目
- 提交最好了！

## / 本周重点

- 项目数据集选择：<http://t.cn/RejvMmS>, (原先的链接比较凶残，我短了它一下)
- 项目环境准备：
  - Uda环境完成：可以在 |3 实战项目 这一节完成，但是要把下载好的数据文件通过 File - Open - Upload 上传到跟目录下。昨晚之后再下载html或者pdf文件在最后的 |4 项目：探索数据集 右下角点击 '提交项目' 提交
  - 本地环境完成：可以在 |3 实战项目 这一节下载ipynb模版文件，之后在本地做完，把最后的文件同样点击 '提交项目' 提交。这个模版文件我搬运了下：[https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/Investigate\\_a\\_Dataset-zh.ipynb](https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/Investigate_a_Dataset-zh.ipynb)

- 开始写项目文件！！！
  - |目标1: 数据整理
  - |目标2: 探索性数据分析
  - |目标3: 得出结论

# /目标1/: 数据整理 (TMDB数据示例)

## / 1.准备

其实在做项目的时候，大家一定要：认真看模版文件！！！不但有整个报告的结构在里面，很多地方还是有提示的。那么在开始我们要干什么事情呢：

- 从头到尾浏览下项目模版文件（链接在本文开始）
- 认真看下评审说明：<https://review.udacity.com/#!/rubrics/306/view>
- 写好import代码框，一般来讲就是这几个家伙了（提示好nice的）：

```
1 # 用这个框对你计划使用的所有数据包进行设置
2 # 导入语句
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import numpy as np
7
8 # 务必包含一个'magic word'，以便将你的视图显示出来
9 %matplotlib inline
10
11 # magicwords的说明：
12 # http://ipython.readthedocs.io/en/stable/interactive/magics.html
```

## / 2.读入数据

接下来就是把你下载好的数据加载，如果下载遇到问题可以找我友情搬运（保证不干坏事），但要注意以下情况：

- 下载后先检查数据的扩展名，看下大小（一般不会特别大）
- 注意用excel/numbers打开数据浏览下内容，有个直观的概念
- 读取csv的时候看下需不需要加参数
- 读取完成后，检查数据是否如预期

```
1 # 加载数据并打印几行。进行这几项操作，来检查数据
2 df = pd.read_csv('tmdb-movies.csv')
3 print(df.columns)
4 # 看下都有哪些列
5 print(df.head(50))
6 # 看下50行，可以粗略的看下NaN和各feature的情况
7 df.dtypes
8 # 看下各列的数据类型
```

## / 3. 检查数据

检查数据么，一般是看4个东西：维度、缺失值、重复值、唯一值

- 看维度（上面框是输入，下面框是输出，后同）：

```
1 | df.shape
```

```
1 | (10866, 21)
```

- 看缺失值

```
1 | df.isnull().sum()
```

|    |                      |      |
|----|----------------------|------|
| 1  | id                   | 0    |
| 2  | imdb_id              | 10   |
| 3  | popularity           | 0    |
| 4  | budget               | 0    |
| 5  | revenue              | 0    |
| 6  | original_title       | 0    |
| 7  | cast                 | 76   |
| 8  | homepage             | 7930 |
| 9  | director             | 44   |
| 10 | tagline              | 2824 |
| 11 | keywords             | 1493 |
| 12 | overview             | 4    |
| 13 | runtime              | 0    |
| 14 | genres               | 23   |
| 15 | production_companies | 1030 |
| 16 | release_date         | 0    |
| 17 | vote_count           | 0    |
| 18 | vote_average         | 0    |

```
19 | release_year          0  
20 | budget_adj            0  
21 | revenue_adj           0
```

也可以加一行，看看缺失值的比例（人这种动物看超过100的数会心跳加速的，所以有人发明了百分比）：

```
1 | round(df.isnull().sum()/df.shape[0],2)
```

```
1 | id                      0.00  
2 | imdb_id                0.00  
3 | popularity              0.00  
4 | budget                  0.00  
5 | revenue                 0.00  
6 | original_title          0.00  
7 | cast                    0.01  
8 | homepage                0.73  
9 | director                0.00  
10 | tagline                 0.26  
11 | keywords               0.14  
12 | overview                0.00  
13 | runtime                 0.00  
14 | genres                  0.00  
15 | production_companies   0.09  
16 | release_date            0.00  
17 | vote_count               0.00  
18 | vote_average             0.00  
19 | release_year             0.00  
20 | budget_adj               0.00  
21 | revenue_adj              0.00
```

## • 看重复值

重复值是要去掉的，本例子中只有一个重复值，其实不干掉也不会太影响（虽然就1个，我还是要干掉，强迫症，，，，，其实即使是0也应该做的，因为不知道下一次的数据会是什么样子的，或者以后会变成一个处理数据的产品也说不定呢），标准打法（检查，去掉，检查）：

```
1 | print(df.shape)  
2 | print(df.duplicated().sum())  
3 | df.drop_duplicates(inplace = True)  
4 | print(df.shape)  
5 | print(df.duplicated().sum())
```

```
1 | (10866, 21)
2 | 1
3 | (10865, 21)
4 | 0
```

- 看唯一值

这里可以通过唯一值观察一些内容，比如budget（预算）的唯一值有557个，而budget\_adj(修正后的预算)有2617个，是不是说明指定预算时有点拍脑袋。或者是看runtime居然有247个唯一值，要知道这是按照分钟统计的（这么多种，真是没想到！）。如果是要输出含有非唯一值的列，可以参考案例2中写个小循环自己输出。

```
1 | df.nunique()
```

```
1 | id                      10865
2 | imdb_id                 10855
3 | popularity               10814
4 | budget                   557
5 | revenue                  4702
6 | original_title           10571
7 | cast                     10719
8 | homepage                 2896
9 | director                 5067
10 | tagline                  7997
11 | keywords                 8804
12 | overview                 10847
13 | runtime                  247
14 | genres                   2039
15 | production_companies    7445
16 | release_date              5909
17 | vote_count                1289
18 | vote_average              72
19 | release_year              56
20 | budget_adj                2614
21 | revenue_adj                4840
```

## / 4.清理数据

其实上一节的去掉重复值应该是这个环节做的（因为反正也不复杂就顺手处理了）。接下来是清理数据，为探索数据分析提供一个干净的数据：

- 丢弃列

拿到数据后，可以对于无用的列进行处理，前提是理解了列都是什么，需要仔细看下数据说明。比如TMDB数据中，我们考察预算和收入使用\_adj调整后的真实列更加准确，那么之前版本的列就可以删除，当然如果要比较之前版本和调整版本之间的差异，就不能删掉。另外imdb\_id和id都是unique的编号，留一个就好了。

```
1 # 1 清理无用列
2 # buget \ revenue 使用adj调整后数值，丢弃
3 # imdb id 与 id作用相同，丢弃
4 # homepage 缺失很多，其中的内容
5 print(df.shape)
6 drop_list = ['budget', 'revenue', 'imdb_id']
7 df.drop(drop_list, axis = 1, inplace = True)
8 print(df.shape)
9 df.columns
```

```
1 (10865, 21)
2 (10865, 18)
3 Index(['id', 'popularity', 'original_title', 'cast', 'homepage', 'direct
4 or',
5         'tagline', 'keywords', 'overview', 'runtime', 'genres',
6         'production_companies', 'release_date', 'vote_count', 'vote_averag
7 e',
8         'release_year', 'budget_adj', 'revenue_adj'],
9          dtype='object')
```

- 处理空值

实话说，在这个环节我并没有处理空值，没有特别注意，后来在探索性数据分析的环节因为图像的比较奇怪才想起来。这里就没有重新写，给大家一个比较实在的分析过程（厚脸皮就是有这点好处）。同时，也说明了数据清理和探索性数据分析（EDA）这两个环节是交织在一起的。有的时候你通过EDA才去想要一些数据的特点，再去调整数据。不过，有时候把回答不同问题的数据存为不同版本会更加清晰。

- 改变列

这个一般是要先有个问题，或者怀疑点的，比如我看到了homepage这一列缺失值超多的，就在洗澡的时候联想到（阿基米德裸奔称皇冠的故事了解一下？）是不是有自己网站的电影都更重要一点，是不是能在预算和收入中有所体现？所以我决定建立一个新列，用于区分这两种情况：

```
1 # 2 调整列
2 # homepage 缺失很多，但突然想看看是不是有homepage的
```

```
3 # 有更加多的投入和收入 (因为重视么)
4 # 显示最有一列是谁 (碰巧了在最后就不用都输出了)
5 if 'homepage' in df.columns:
6     df['has_homepage'] = df['homepage'].notnull()
7     df.drop('homepage', axis = 1, inplace = True)
8 else:
9     print('data already processed')
10 # 写了个小循环如果执行过就提示下不会报错
11 # 万一评审老师点了两次, 报错了不是会很尴尬的
12 # 开玩笑了, 提交项目是html文件, 但是自己用起来方便
13 print(df.columns[-1:])
14 df.head()
```

```
1 data already processed
2 Index(['has_homepage'], dtype='object')
```

## /目标2/：探索性数据分析（TMDB数据示例）

根据项目模版结构，这部分就开始是问题导向的方法了，这里我举了两个问题（没有包括数据说明那个里面的区分电影分类的方法，那个解决需要多一点点的工作，大家也可以尝试），这里的两个例子是我写项目时候的思维过程的记录。请不要把这个直接copy到项目提交中（非常重要，一定要理解后，自己完成呦！）

### / 研究问题1 - 几个数字值的关系如何

观察数据集，有几个列是数值的，我来看看他们之间有什么关系（这一个项目全靠眼睛，统计学的判断方法咱们会在最后一个项目里学他个昏天黑地 兴高采烈！）

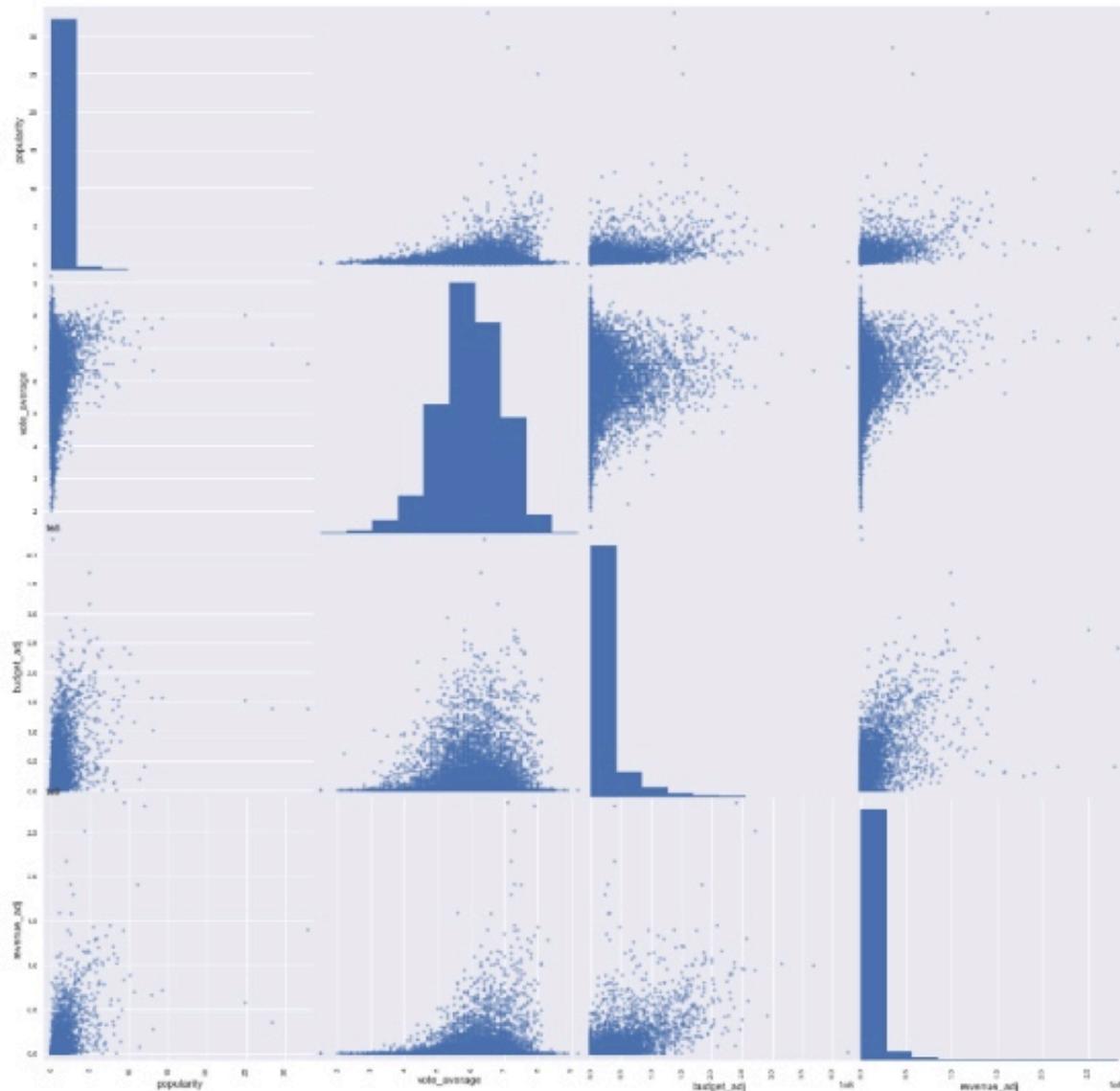
- 整理数据

```
1 # 先把需要保留的列列出来
2 smlist = ['popularity', 'vote_average', 'budget_adj', 'revenue_adj']
3 # 制作一个新的数据用于scater matirx显示 (否则太慢了, 也看不太清)
4 dfsm = pd.DataFrame()
5 for i in smlist:
6     dfsm[i] = df[i]
7 dfsm.head()
```

```
1 | popularity  vote_average    budget_adj  revenue_adj
2 | 0   32.985763    6.5 1.379999e+08      1.392446e+09
3 | 1   28.419936    7.1 1.379999e+08      3.481613e+08
4 | 2   13.112507    6.3 1.012000e+08      2.716190e+08
5 | 3   11.173104    7.5 1.839999e+08      1.902723e+09
6 | 4   9.335014     7.3 1.747999e+08      1.385749e+09
```

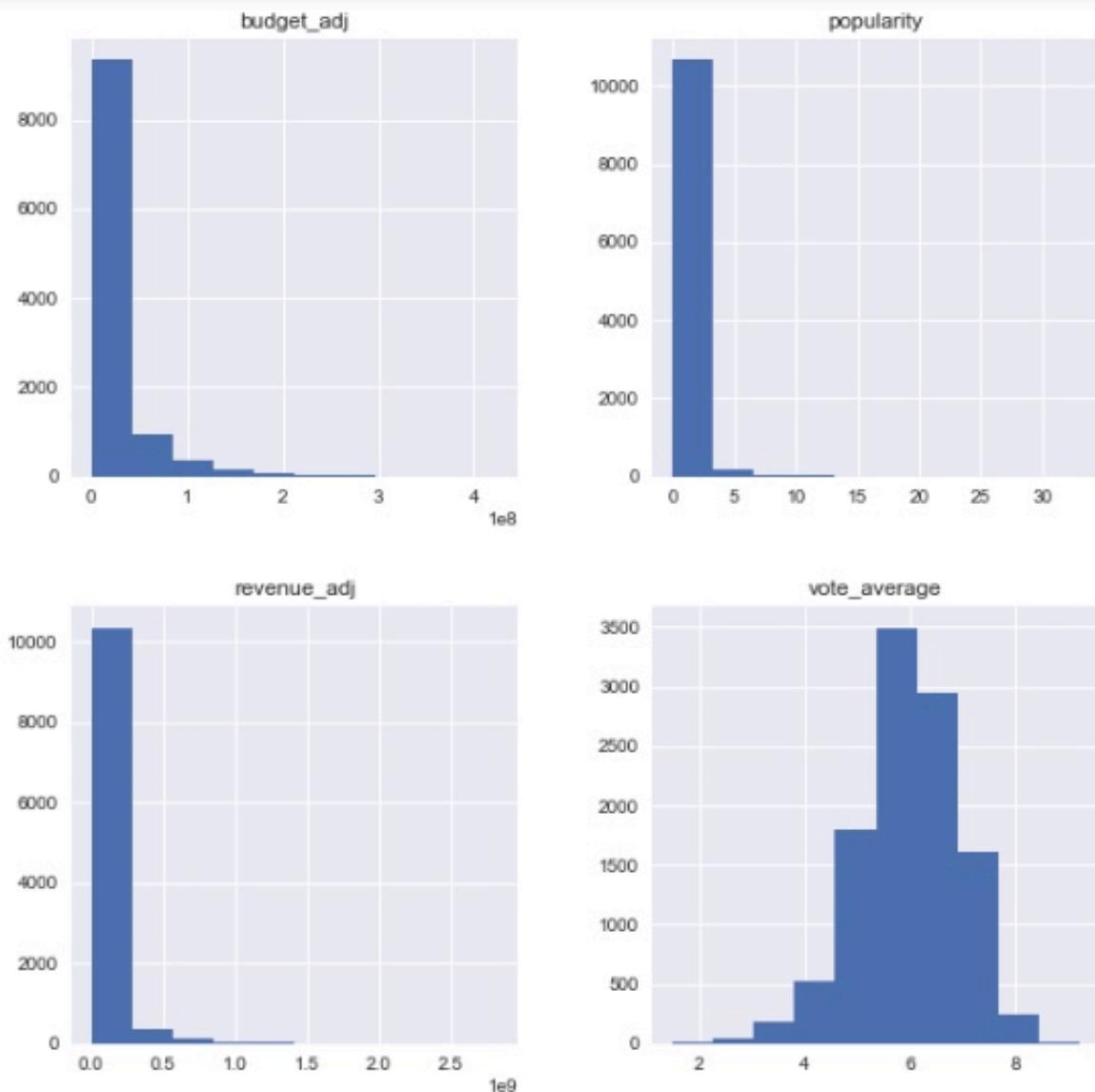
- 画图观察（还记得那个scatter\_matrix，黑客帝国散点矩阵图超Cool！）

```
1 | # OK 我们接下来就可以画个scater matrix图来看看了
2 | pd.plotting.scatter_matrix(dfsm, figsize=(20,20));
```



真是超难看（大家自己做的画全屏能看见字），这4个指标中除了vote\_average其他的都分布的特别靠左，，，，为什么呢？我来hist直接看下分布情况：

```
1 | dfsm.hist(figsize=(10,10));
```



这回就明显了，分布超级偏左，我再看看describe检查下，果然是很奇怪，钱除了vote其他列的mean都是那么的少！：

```
1 | dfsm.describe()
```

|              | popularity   | vote_average | budget_adj   | revenue_adj  |
|--------------|--------------|--------------|--------------|--------------|
| <b>count</b> | 10865.000000 | 10865.000000 | 1.086500e+04 | 1.086500e+04 |
| <b>mean</b>  | 0.646446     | 5.975012     | 1.754989e+07 | 5.136900e+07 |
| <b>std</b>   | 1.000231     | 0.935138     | 3.430753e+07 | 1.446383e+08 |
| <b>min</b>   | 0.000065     | 1.500000     | 0.000000e+00 | 0.000000e+00 |
| <b>25%</b>   | 0.207575     | 5.400000     | 0.000000e+00 | 0.000000e+00 |
| <b>50%</b>   | 0.383831     | 6.000000     | 0.000000e+00 | 0.000000e+00 |
| <b>75%</b>   | 0.713857     | 6.600000     | 2.085325e+07 | 3.370173e+07 |
| <b>max</b>   | 32.985763    | 9.200000     | 4.250000e+08 | 2.827124e+09 |

那么我再看详细点，看看每10%的数据都是多少（偷懒写了个循环，结果太多，就不一一展示

了），结果发现budget和revenue居然在50%的时候都是0！

```
1 # 检查百分位的值
2 # 那么我们查查几个百分位的数值吧，使用linspace每隔5%显示一下！
3 # dfsm['popularity'].quantile(np.linspace(0,1,21))
4 # linspace 是0到1的n个点之间的分割，因为有开始结束所以要增加1个，比如21
5 # 可以看出来过了只有15%的数值超过了1，Max真是太
6 # 这个数值画图意义也不是特别明显
7
8 # 干脆我们写个函数把其他的quantile也看一下
9 # 把平均分为几个作为n值输入
10 def vquantile(data, n):
11     for i in data:
12         quantile = data[i].quantile(np.linspace(0,1,n+1))
13         print('\n', 'Checking:', i , '\n', quantile)
14
15 vquantile(dfsm, 10)
```

```
1 Checking: budget_adj
2 0.0 0.000000e+00
3 0.1 0.000000e+00
4 0.2 0.000000e+00
5 0.3 0.000000e+00
6 0.4 0.000000e+00
7 0.5 0.000000e+00
8 0.6 4.462259e+06
9 0.7 1.404050e+07
10 0.8 2.904090e+07
11 0.9 5.812285e+07
12 1.0 4.250000e+08
```

估计是有空值了吧，而且还不少，我来刷一波好了，因为是分析和公司收入相关的所以我就大胆删除了（这么多0预算，0收入的电影是什么意思...）

```
1 # 根据上面的输出，发现budget有很多空的，特别不靠谱，我要干掉它们
2 print(len(dfsm[dfsm['budget_adj'] == 0]))
3 print(dfsm.shape)
4 # 干脆把budget和revenue的空值都干掉
5 dfsm5k = dfsm[(dfsm['revenue_adj'] != 0)&(dfsm['budget_adj'] != 0)]
6 dfsm5k.shape
```

```
1 5696
2 (10865, 4)
```

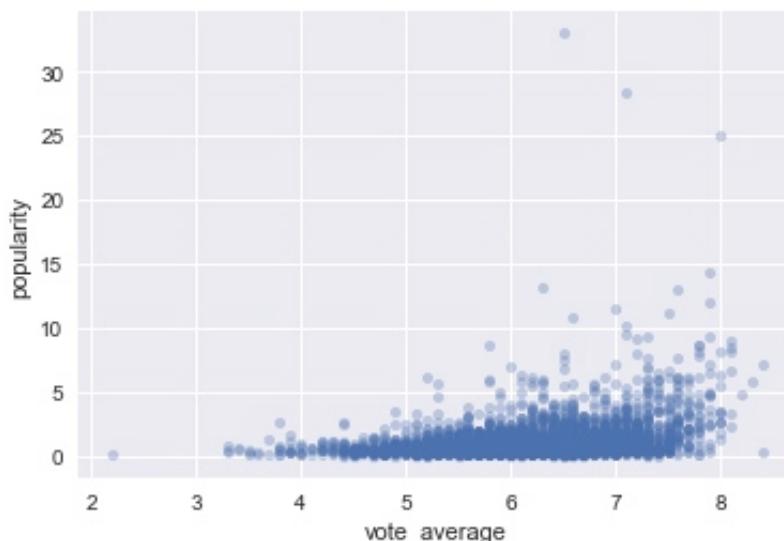
3 | (3854, 4)

```
1 | # 干掉了60%的数据，实际处理时候要慎重，不过数值就好看多了
2 | # ps：同时也要小心对数据好看的追求，不要太极端了多删了数据
3 | print(vquantile(dfsm5k, 10))
```

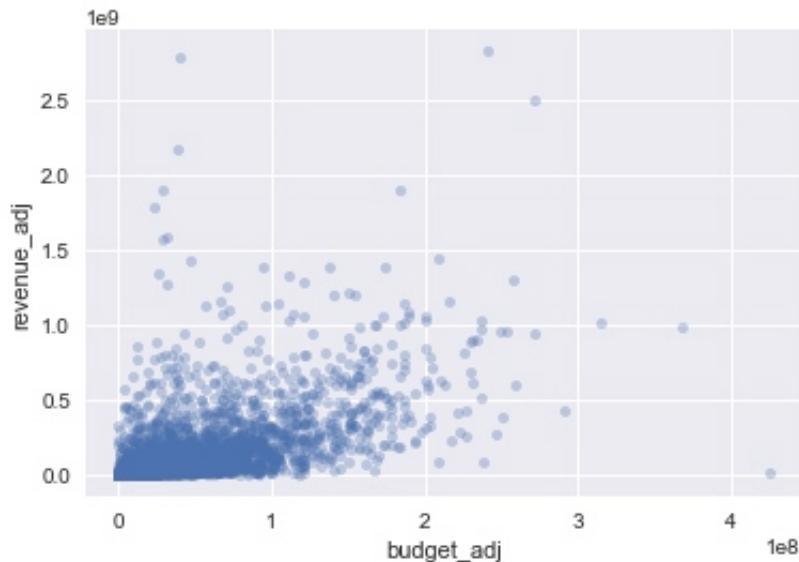
```
1 | Checking: budget_adj
2 | 0.0      9.693980e-01
3 | 0.1      4.612144e+06
4 | 0.2      1.029637e+07
5 | 0.3      1.617956e+07
6 | 0.4      2.229130e+07
7 | 0.5      3.001611e+07
8 | 0.6      3.816024e+07
9 | 0.7      5.162149e+07
10 | 0.8      7.114803e+07
11 | 0.9      1.031006e+08
12 | 1.0      4.250000e+08
```

最后我们用修正过的数据来画图（从scater matrix中挑出来单独画）。在探索阶段，对于细节我们可以不太细究，在最后的得出结论阶段再进行报告级别的调整就好：

```
1 | # 发现1 vote 和 pop有关系的
2 | dfsm5k.plot(x='vote_average',
3 |                 y='popularity', kind='scatter', alpha=0.3);
4 | # 要不要美化一下，少年？
```



```
1 | # 发现2 budget 和 revenue有一些关系，但是没有那么明显
2 | dfsm5k.plot(x='budget_adj',
3 |                 y='revenue_adj', kind='scatter', alpha=0.3);
```



## / 研究问题2 - 是否有Website和Feature的关系

大家还记得之前在清理的时候，我加了1列has\_homepage么，可以区分开是否有homepage的电影的参数之间有没有区别。可以使用groupby进行区分。这里需要注意的是：

- 用了两个代码框进行了区分：收集列和删除空值
- 这两个的前后顺序决定了代码运行的时间，也是在写代码时候要考虑的：
  - 先运行能排除最多的
  - 优先使用pandas和numpy等基于c的包完成
  - 使用time进行测速

```

1 # 继续使用删除了budget和rev为0的（和钱都没关系的不分析了吧）
2 smlist2 = ['popularity', 'vote_average', 'budget_adj', 'revenue_adj', 'has_
3 homepage']
4 # 制作一个新的数据用于scater matirx显示（否则太慢了，也看不太清）
5 dfsm2 = pd.DataFrame()
6 for i in smlist2:
7     dfsm2[i] = df[i]
dfsm2.head()
```

|   | popularity | vote_average | budget_adj   | revenue_adj  | has_homepage |
|---|------------|--------------|--------------|--------------|--------------|
| 0 | 32.985763  | 6.5          | 1.379999e+08 | 1.392446e+09 | True         |
| 1 | 28.419936  | 7.1          | 1.379999e+08 | 3.481613e+08 | True         |
| 2 | 13.112507  | 6.3          | 1.012000e+08 | 2.716190e+08 | True         |
| 3 | 11.173104  | 7.5          | 1.839999e+08 | 1.902723e+09 | True         |
| 4 | 9.335014   | 7.3          | 1.747999e+08 | 1.385749e+09 | True         |

```
1 # 接下来一样干掉空值的项
2 # 根据上面的输出，发现budget有很多空的，特别不靠谱，我要干掉它们
3 print(len(dfsm2[dfsm2['budget_adj'] == 0]))
4 print(dfsm2.shape)
5
6 # 干脆把revenue的空值也干掉
7 dfsm25k = dfsm2[(dfsm2['revenue_adj'] != 0)&(dfsm2['budget_adj'] != 0)]
8 dfsm25k.shape
```

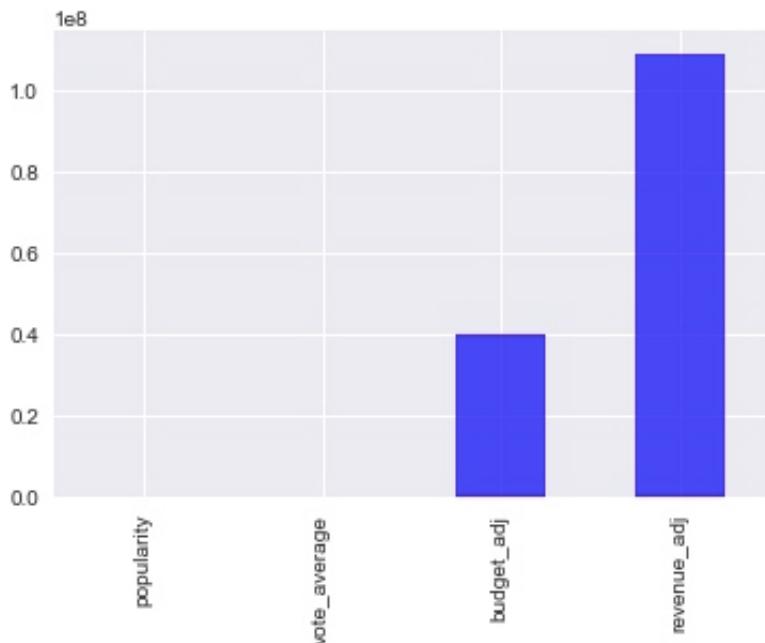
```
1 5696
2 (10865, 5)
3 (3854, 5)
```

数据有了，我们来看看，用groupby求均值，再画图看看：

```
1 mean = dfsm25k.groupby('has_homepage').mean()
2 mean
3 # 可以看出来区别还是很大的
```

| has_homepage | popularity | vote_average | budget_adj   | revenue_adj  |
|--------------|------------|--------------|--------------|--------------|
| False        | 0.905837   | 6.112521     | 3.993240e+07 | 1.089843e+08 |
| True         | 1.678034   | 6.262903     | 5.157438e+07 | 1.848763e+08 |

```
1 # 我们来画图试试，结果相差太多，前两个基本看不到啊
2 dfsm25k.groupby('has_homepage').mean().iloc[0].plot(
3     kind = 'bar', alpha = 0.7, color = 'b')
```



肿么办，我们把数据集都规整到一个数量级吧，就是按列除，结果就好多了：

```

1 # 把数据都搞到一个数量级好了
2 mean_edited = mean.copy()
3 # 需要注意如果不用.copy()的话修改mean_edited 也会修改mean
4 mean_edited['budget_adj'] = mean['budget_adj'] / 10000000
5 mean_edited['revenue_adj'] = mean['revenue_adj'] / 10000000
6 mean_edited

```

|              | popularity | vote_average | budget_adj | revenue_adj |
|--------------|------------|--------------|------------|-------------|
| has_homepage |            |              |            |             |
| False        | 0.905837   | 6.112521     | 3.993240   | 10.898425   |
| True         | 1.678034   | 6.262903     | 5.157438   | 18.487626   |

我们可以看下两个分类之间的倍数关系，直接相除就可以了（向量运算，week2导学中有讲）：

```

1 mean_edited.iloc[1] / mean_edited.iloc[0]
2 # 我们来看看变化有多少，pop和rev的增幅都超过了50%

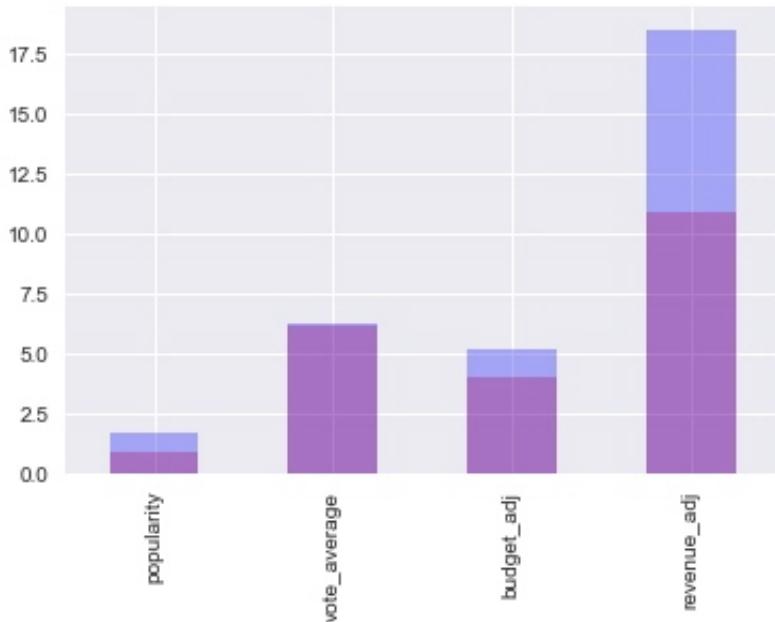
```

把两张图堆叠起来看看，比较奇怪但是要比较的东西已经出来了：

```

1 # 堆叠起来看看，不是特别理想
2 mean_edited.iloc[0].plot(kind = 'bar', color = 'r', alpha = 0.3)
3 mean_edited.iloc[1].plot(kind = 'bar', color = 'b', alpha = 0.3)

```



还是老老实实做图吧，如果下面的代码看着有点难，请看下week7-guide-plus1的matplotlib彩蛋：

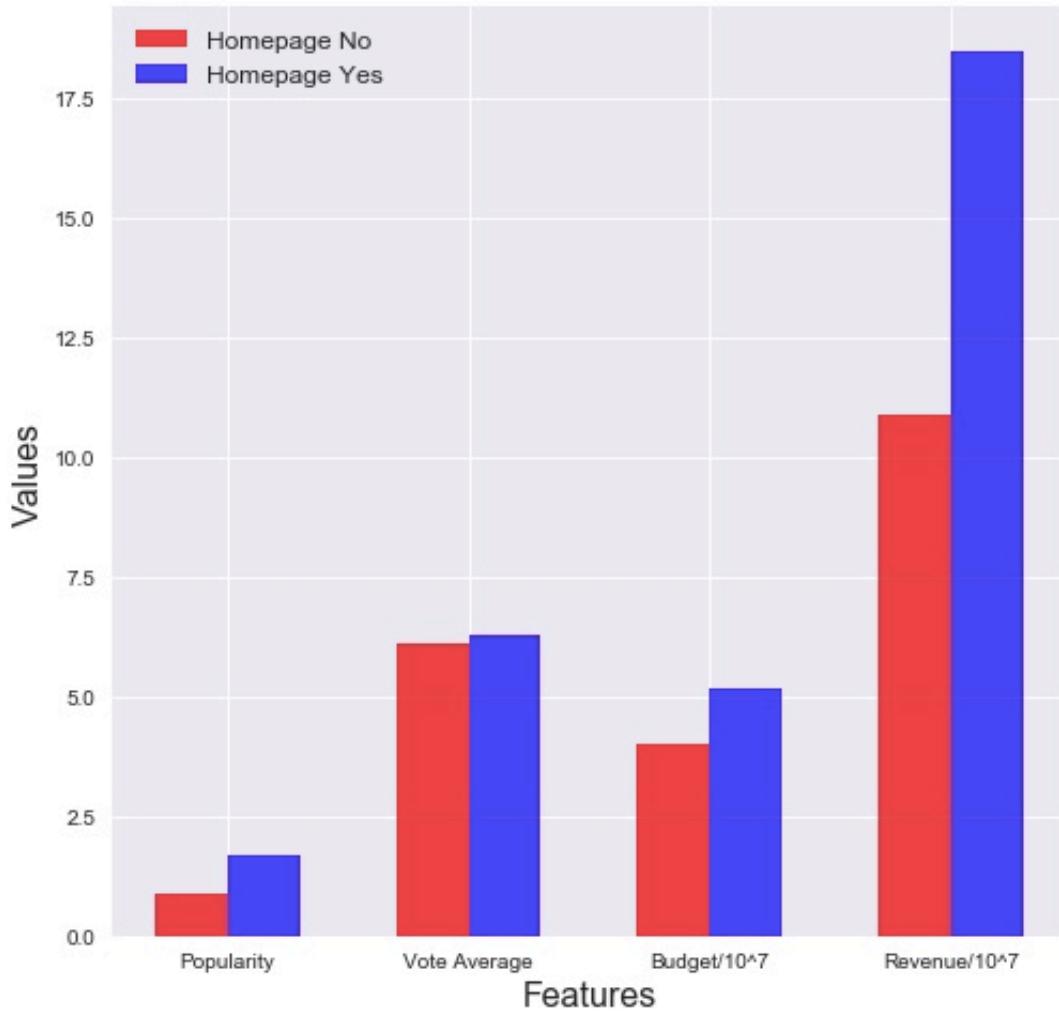
```

1 # 那只好认真的画图了，开始想到了折线图，但是这4个指标没有延续的关系
2 # 折线图比较适合画x轴为时间的数据
3 sns.set(context = 'notebook', style = 'darkgrid')
4 # 使用sns设置底色要放在最前面
5 fig = plt.figure(figsize=(8,8))
6 # 如果使用是import的是matplotlib 而不是 matplotlib.pyplot
7 # 调用的时候会有区别
8 ax = fig.add_subplot(1,1,1)
9 ind = np.arange(len(mean_edited.iloc[0]))
10 # 要想微调位置要设置个变量ind
11 w = 0.3
12 # bar的宽度也是可以设定的
13 ax.bar(ind, mean_edited.iloc[0], color = 'r',
14         alpha = 0.7, width = w, label = "Homepage No")
15 ax.bar(ind+w, mean_edited.iloc[1], color = 'b',
16         alpha = 0.7, width = w, label = "Homepage Yes")
17 #ind是bar的位置, dataared是输入的数据, 多个width是bar的宽度
18 ax.legend(loc = 'best', fontsize = 12)
19 ax.set_xticks(ind+w/2)
20 # 设定x轴标签位置, 在正中间
21 # 根据你要堆叠的数据设定
22 mean_label = ['Popularity', 'Vote Average',
23                 'Budget/10^7', 'Revenue/10^7']
24 # 把标签设定为比较友好的方式
25 # 也可以导出index处理, 比写死了好, 有没有兴趣优化一下?
26 ax.set_xticklabels(mean_label)
27 # 将分类标签作为x标签

```

```
28 | ax.set_ylabel('Values', fontsize = 16)
29 | ax.set_xlabel('Features', fontsize = 16)
30 | fig.suptitle('Compare Features on Homepage or Not',
31 |                 fontsize = 24, color = 'red');
32 |
33 | # 记住在最后的语句后面加个; 就不会显示图之外的信息, 比较友好
```

## Compare Features on Homepage or Not



## /目标3/: 得出结论 (TMDB数据示例)

这个部分大家可以理解为你家老板要copy到ppt里面去忽悠人的部分，有可能他前面都不看，就是要copy这里面的东西走，所以要有逻辑有颜值：

## / 数据集说明

本次分析的数据为TMDB的电影评分数据，内容比较丰富：

- 数据量大：数据有10866行
- 数据维度大：数据有21列，包括涉及评价、分类、预算、宣传语等一些列内容
- 数据质量不错：考虑到数据中有大量0预算和0收入的电影外，重复值只有1个，除了 homepage、tagline和keywords这些可能后期才有或者记录的数据外，只有 production\_companies的缺失百分比超过了1%（其实大部分都一点也不缺）
- 数据收集全：其实上面的0预算的电影也收入了说明收集范围还是挺全的，分析时候可以根据需要进行处理

## / 研究问题1 - 几个数字值的关系如何

### // 问题观察和结论

对于几个数值组成的Feature，我们可以来画散点图进行观察每每两个参数的关系，经过分析以下两个值的一注意：

- 影片得分和流行度正相关：平均投票得分越高，流行度就可能越高
- 影片的投入和收入之间有相关关系，但是没有那么明显

### // 问题的图形展示

```

1 # 最后的图：
2 # # 先是设定图纸和图画位置
3 fig = plt.figure(figsize=(8,8))
4 ax1 = fig.add_subplot(2, 1, 1)
5 ax2 = fig.add_subplot(2, 1, 2)
6
7 # # 把数据加载到图中
8 x1 = dfsm5k['vote_average']
9 y1 = dfsm5k['popularity']
10 x2 = dfsm5k['budget_adj']
11 y2 = dfsm5k['revenue_adj']
12
13 ax1.scatter(x1, y1, color = 'b', alpha = 0.2, label = "All")
14 ax2.scatter(x2, y2, color = 'b', alpha = 0.2, label = "All")
15
16 # # label、title等的细化
17 ax1.set_ylabel('Vote Average', fontsize = 16)
18 ax1.set_xlabel('Popularity', fontsize = 16)
19 ax2.set_ylabel('Budget', fontsize = 16)
20 ax2.set_xlabel('Revenue', fontsize = 16)
21

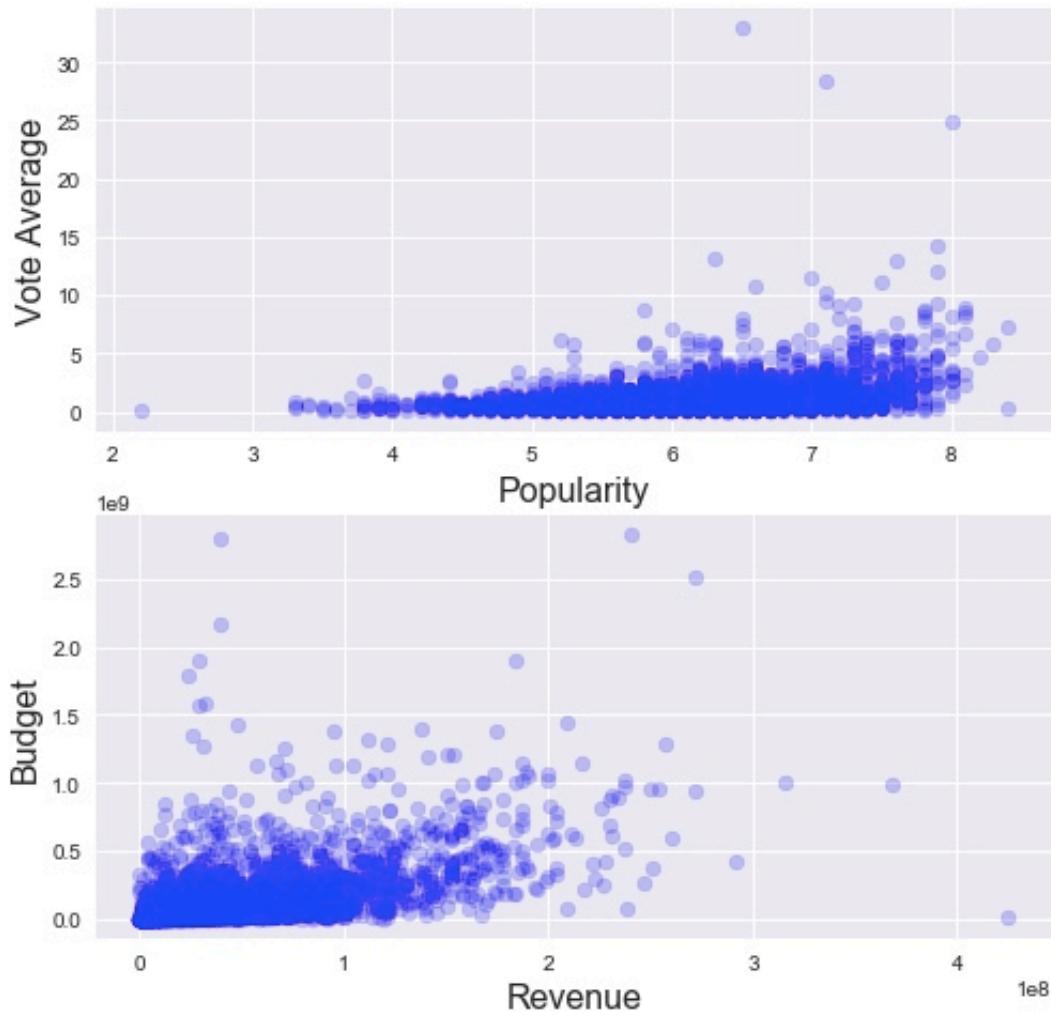
```

```

22 fig.suptitle('Scater on Vote/Pop and Budg/Rev',
23         fontsize = 24, color = 'red');
24
25 # # 其实散点图是可以把分组加进去的（比如homepage有没有）找到一个：
26 # # https://pythonspot.com/en/tag/scatter-plot/
27 # # 大家有需要的可以试试

```

## Scater on Vote/Pop and Budg/Rev



## // 思考和建议

从几个数据来看，TMDB电影的数据与爆款营销理论非常契合，而从那删除掉的60% 0预算和0收入的情况看，长尾理论并不特别契合（这两种并没有冲突）。建议继续更新数据，并且可以增加对电影tagline等比较主观和情感的内容增加分析并加深判断。

## / 研究问题2 - 是否有Website和Feature的关系

上面举了例子，这里就不啰嗦了，图的话在探索部分的最后一张图我也是很满意的了（其实那张图应该放在这个环节做）。

# /彩蛋/

---

本导学内容是以P3中的TMDB数据为例进行的分析，请大家注意以下几点：

- 本文件还原了我在做项目时候的思考和解答过程，这个过程并不是最优的
- 主要是抛砖引玉将一个项目比较完整的过程展现出来，供大家参考下
- **按照项目中的建议，你可以先写一个代码多，尝试多‘比较乱’的版本，之后再提交时候再提交另一个版本。**
- 不过如果你比较懒，如果能够满足‘项目评审标准’又能注意逻辑和描述，一个文件也是OK的
- 除了SQL的那个数据，我们有4个数据可以选的，兴趣是最好的出发点
- 加油加油，争取1周时间能够提交，之后再按照评审老师的反馈修改就更有针对性了。记住：**时间节点比质量更加重要！**（因为还有机会改的，不限次数）

Title: W8 Plus FBI枪支数据项目提示

Tags: 数据分析初级, 实战项目

# Title: W8 Plus FBI枪支数据项目提示

## /目标4/: 数据整理 (FBI数据)

### / 1.准备

其实在做项目的时候，大家一定要：认真看模版文件！！！不但有整个报告的结构在里面，很多地方还是有提示的。那么在开始我们要干什么事情呢：

- 从头到尾浏览下项目模版文件（链接在本文开始）
- 认真看下评审说明：<https://review.udacity.com/#!/rubrics/306/view>
- 写好import代码框，一般来讲就是这几个家伙了（提示好nice的）：

```
1 # 用这个框对你计划使用的所有数据包进行设置
2 # 导入语句
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import numpy as np
7
8 # 务必包含一个'magic word'，以便将你的视图显示出来
9 %matplotlib inline
10
11 # magicwords的说明：
12 # http://ipython.readthedocs.io/en/stable/interactive/magics.html
```

### / 2.读入数据

接下来就是把你下载好的数据加载，如果下载遇到问题可以找我友情搬运（保证不干坏事），但要注意以下情况：

- 下载后先检查数据的扩展名，看下大小（一般不会特别大）

- 注意用excel/numbers打开数据浏览下内容，有个直观的概念
- 读取csv的时候看下需不需要加参数
- 读取完成后，检查数据是否如预期

```

1 # 这里我们要加载xlsx文件，可以使用xlrd第三方库来，有不少选择
2 # 我当然还是用大熊猫了read_excel
3 df = pd.read_excel('gun_data.xlsx')
4 # 注意这里面也可以加参数，和csv的一样，请见以前的导学文件
5 # 看下都有哪些列
6 df.head()
7 # 看下head，分析下数据列的意思（本数据没有列说明）

```

观察了下，数据的结构是这样的（前面是列名）：

- month/state - 按照每个月每个周一的数据，开始第一条是2017-09 Alabama以此类推
- permit/permit\_check - permit应该是批准数（执枪护照），permit check是不是核实，这个数据从比较晚才开始有，而且其中的数量有很多是0，在没有详细说明的情况下不太好使用
- handgun/long\_gun/other - 应该是枪的分类（other是机关枪，榴弹炮，生化危机片场？）但是这3个数字加一起和permit的数量非常不一样，有时候多有时候少，而且差别很大
- multiple - 应该是有2种以上的
- 其他列也是各种的奇怪名称
- 最后一个是total，这个数是和其他的所有数只和相同的
- 所以呢推测这个数据是NICS（FBI的潜质检查数据）的每月的使用记录情况。其中每一列无论是发许可（permit）还是买枪（handgun）或者是退货（return to handgun）都要在NICS中做记录，这一行数据是NICS一个月的记录数
- 我们来用代码检验一下：

```

1 # 先看下第一行的数据
2 print(df.iloc[0])
3
4 # 按照推论，第3个数据开始至倒数第1个数据的和应该等于最后一列的total
5 df.iloc[0][2:-1].sum() == df.iloc[0][-1]

```

|   |                |         |
|---|----------------|---------|
| 1 | month          | 2017-09 |
| 2 | state          | Alabama |
| 3 | permit         | 16717   |
| 4 | permit_recheck | 0       |
| 5 | handgun        | 5734    |
| 6 | long_gun       | 6320    |
| 7 | other          | 221     |
| 8 | multiple       | 317     |
| 9 | admin          | 0       |

```

10 prepawn_handgun           15
11 prepawn_long_gun          21
12 prepawn_other              2
13 redemption_handgun        1378
14 redemption_long_gun       1262
15 redemption_other            1
16 returned_handgun           0
17 returned_long_gun          0
18 returned_other              0
19 rentals_handgun             0
20 rentals_long_gun            0
21 private_sale_handgun        9
22 private_sale_long_gun       16
23 private_sale_other            3
24 return_to_seller_handgun      0
25 return_to_seller_long_gun      0
26 return_to_seller_other          3
27 totals                      32019
28 Name: 0, dtype: object
29
30 True

```

那么按照这个猜测的话，我们不用管那么多列的数据，我比较关注的是handgun、long\_gun、other这3种枪的关系（会不会发现一些地域特征？对于米国地理一窍不通的我表示很为难）

另外一个数据是U.S. Census Data.csv, 是各洲的人口统计信息。我们来看下，结果发现这个数据是按照法每行一次普查为数据行，每个洲的人数为列：

```

1 dfcen = pd.read_csv('U.S. Census Data.csv')
2 dfcen.head()

```

|   | Fact                                              | Fact Note | Alabama   | Alaska  | Arizona   | Arkansas  | California | Colorado  | Connecticut | Delaware | ... | South Dakota | Tennessee | Texas      | Utah      |
|---|---------------------------------------------------|-----------|-----------|---------|-----------|-----------|------------|-----------|-------------|----------|-----|--------------|-----------|------------|-----------|
| 0 | Population estimates,<br>July 1, 2016,<br>(V2016) | NaN       | 4,863,300 | 741,894 | 6,931,071 | 2,988,248 | 39,250,017 | 5,540,545 | 3,576,452   | 952,065  | ... | 865454       | 6651194   | 27,862,596 | 3,051,217 |

那么这个数据我们怎么用呢，有几个想法：

- 如果是只看最近的数据和做分析，那么就使用census最后一行2016年相关的数据就好了
- 之所以说是相关的数据，是2016、2012、2010年的数据除了人数还有很多的百分比数据，比如6岁以下，65岁以上什么的（就是可以作为分类信息了）
- 其他大概70条之后的数据有比较奇怪的title，pass不予考虑（除非有详细的说明）

现在我们有了2个数据：

- 一个是从1999年开始的枪支数据，行为月/州，列为各种枪支数据统计
- 另一个是人口统计数据，行为年/统计特征，列为各州的数量
- 第一个文件州是行，第二个文件州是列，这两个文件的相同点是州
- 有的时候这种情况需要进行行列转换，但这个案例我觉得把第二个文件抽出部分数据更加实用
- 我来过滤下第二个文件包含2016年的行，看看都有什么内容（在检查数据中完成）

## / 3. 检查数据

首先读入Census数据，之后观察到Fact这一列是数值，我们再把Fact检索出来观察名字：

```

1 dfcen = pd.read_csv('U.S. Census Data.csv')
2 dfcen.head()
3 dfcen['Fact'].values
4 # 使用.values直接输出值，不看索引会简洁些

```

之后发现2016年的数据条目都有V2016这个字符作为标识，我们先去掉空值，再做筛选（否则筛选会报错）：

```

1 # 去掉空值，存为新的数据(多次运行shape可以看出对比)
2 print(dfcen.shape)
3 dfcen_edited = dfcen[dfcen['Fact'].notnull()]
4 print(dfcen_edited.shape)

5
6 # 过滤
7 dfcen_2016 = dfcen_edited[dfcen_edited['Fact'].str.contains("2016")]
8 # 也可以加个na = False参数
9 # https://stackoverflow.com/questions/28311655/ignoring-nans-with-str-contains
10
11 # df[df.fact xxxx] 是 operation chain 的方法，举例：
12 # https://stackoverflow.com/questions/11869910/pandas-filter-rows-of-dataframe-with-operator-chaining/28159296#28159296
13
14 # 使用chain方法对str进行筛选：
15 # https://stackoverflow.com/questions/32616261/filtering-pandas-dataframe-rows-by-contains-str
# pandas 还有 .filter的方法也可以实现
dfcen_2016.shape

```

过滤出来17条数据，这部分的代码结果：

```

1 (85, 52)
2 (80, 52)

```

```

3 (17, 52)

4

5 array(['Population estimates, July 1, 2016, (V2016)',
6     'Population estimates base, April 1, 2010, (V2016)',
7     'Population, percent change - April 1, 2010 (estimates base) to J
8 uly 1, 2016, (V2016)',
9     'Persons under 5 years, percent, July 1, 2016, (V2016)',
10    'Persons under 18 years, percent, July 1, 2016, (V2016)',
11    'Persons 65 years and over, percent, July 1, 2016, (V2016)',
12    'Female persons, percent, July 1, 2016, (V2016)',
13    'White alone, percent, July 1, 2016, (V2016)',
14    'Black or African American alone, percent, July 1, 2016, (V2016)
15    ',
16    'American Indian and Alaska Native alone, percent, July 1, 2016,
17 (V2016)',
18    'Asian alone, percent, July 1, 2016, (V2016)',
19    'Native Hawaiian and Other Pacific Islander alone, percent, July
20 1, 2016, (V2016)',
21    'Two or More Races, percent, July 1, 2016, (V2016)',
22    'Hispanic or Latino, percent, July 1, 2016, (V2016)',
23    'White alone, not Hispanic or Latino, percent, July 1, 2016, (V2
016)',

24    'Housing units, July 1, 2016, (V2016)', 'Building permits, 2016
25    '],
26    dtype=object)

```

可以看到这个信息里除了信息之外有两个方面的数据

- 人口年龄组成数据（按照年龄）貌似和持枪关系不大（5岁以下，，，，买把真枪玩玩？），这方面pass
- 人口成分组成，比如女性、白人单身、黑人或非洲裔、美国土著、亚洲人等等，这些分类可以和各州的比率比较看可以发现什么

综上，结合fbi的枪支管制数据我们就可以发现很多问题了，举例两个：

- （最近数据）各州购买枪的比率和各州人员成分比例有关不？（本导学文件介绍）
- 选中的州（可以是排名靠前的几个），的购买频次如何（结合第二个看，和米国一些法规发布有关系，对于米国不是特别感兴趣，最后会放上个资料，大家参考一下）

## / 4.清理数据

- df只保留最近时间（2017-09）的数据（为了防止某个月有个别事件有影响，也可以取1年的做平均值）

- 存为新的df1数据，列只保留handgun、long\_gun、other并将后两者合并为1列（参见New York Times的2个分类）

## // 先处理FBI数据

```

1 # 把我们要的时间选出来
2 dftemp = df[df['month'].str.contains('2017-09')]
3 print(dftemp.shape)
4 # 把需要的3个列选出来, 存为新的数据q1是question1的简写
5 df1 = pd.DataFrame(dftemp.loc[:, 'handgun':'other'])
6 # 看了下, 还要把index给指定成洲的名字
7 df1.index = dftemp['state']
8 # 将其他2个列合并为Others
9 df1['others'] = df1['long_gun'] + df1['other']
10 df1 = df1.drop(['other', 'long_gun'], 1)
11 # 检查下
12 print(df1)
13 # 转换行列, 因为我们看以洲为输入的图
14 df1 = df1.T
15 df1

```

结果就是这个样子滴：

| state   | Alabama | Alaska | Arizona | Arkansas | California | Colorado | Connecticut | Delaware |
|---------|---------|--------|---------|----------|------------|----------|-------------|----------|
| handgun | 5734.0  | 2320.0 | 11063.0 | 4347.0   | 37165.0    | 15751.0  | 4834.0      | 1414.0   |
| others  | 6541.0  | 3149.0 | 8866.0  | 6228.0   | 27565.0    | 14455.0  | 2267.0      | 1604.0   |

2 rows × 55 columns

## // 再处理Census数据

看了下，其实要保留的列也不是特别多，我们来看下dfcen\_2016['Fact']这一列（列名太凶残了，我还是用index筛选数据好了）

```

1 dfcen_2016['Fact']

0 Population estimates, July 1, 2016, (V2016)
1 Population estimates base, April 1, 2010, (V2...
2 Population, percent change - April 1, 2010 (es...
4 Persons under 5 years, percent, July 1, 2016, ...
6 Persons under 18 years, percent, July 1, 2016, ...
8 Persons 65 years and over, percent, July 1, 2...
10 Female persons, percent, July 1, 2016, (V2016)
12 White alone, percent, July 1, 2016, (V2016)
13 Black or African American alone, percent, July...
14 American Indian and Alaska Native alone, perce...
15 Asian alone, percent, July 1, 2016, (V2016)
16 Native Hawaiian and Other Pacific Islander alo...
17 Two or More Races, percent, July 1, 2016, (V2...
18 Hispanic or Latino, percent, July 1, 2016, (V...
19 White alone, not Hispanic or Latino, percent, ...
22 Housing units, July 1, 2016, (V2016)
29 Building permits, 2016

Name: Fact, dtype: object

```

由于FBI的数据是数量，而Census里的人口分类信息是百分比，所以我们保留全部人口，用作FBI数据的百分比转换（第0行）；再就是保留第12, 13, 15, 18行（保持在统一维度并且干掉过小的值），经过观察各洲的分布情况还真的有很大区别。

```

1 # 筛选出行
2 dfcen_2016_filter = dfcen_2016.loc[[0,12,13,15,18],:]
3 # 干掉Fact Note
4 dfcen_2016_filter = dfcen_2016_filter.drop('Fact Note', 1)
5 print(dfcen_2016_filter.shape)

```

## // 处理两个数据列数不同的问题

```

1 # 55 vs 51 看来两个数据的洲不对应，需要处理一下
2
3 # 来个小循环把相同的列找出来
4 def matchlist(a,b):
5     matchlist = []
6     for i in list(a.columns):
7         if i in list(b.columns):
8             matchlist.append(i)
9         else:
10             print(i)
11     return matchlist
12
13 # 有50个ok的，而且cen是被dfq1包含的

```

```

14
15 matchlist(dfq1,dfcen_2016_filter)
16 matchlist(dfcen_2016_filter,dfq1)
17 # 本来想把Fact这列变成index, 名字太长了, 自己指定把
18 cenindex = ['Population', 'White Alone',
19             'Black or African American', 'Asian Alone',
20             'Hispanic or Latino']
21 dfcen_2016_filter.index = cenindex
22
23 # 删除dfcen[Fact]
24 if 'Fact' in dfcen_2016_filter.columns:
25     dfcen_2016_filter = dfcen_2016_filter.drop(['Fact'],1)
26
27 dfcen_2016_filter

```

dfcen\_2016\_filter做了规整, 还是比较好看:

|                           | Alabama   | Alaska  | Arizona   | Arkansas  | California | Colorado  | Connecticut |
|---------------------------|-----------|---------|-----------|-----------|------------|-----------|-------------|
| Population                | 4,863,300 | 741,894 | 6,931,071 | 2,988,248 | 39,250,017 | 5,540,545 | 3,576,452   |
| White Alone               | 69.30%    | 66.10%  | 83.30%    | 79.40%    | 72.70%     | 87.50%    | 80.60%      |
| Black or African American | 26.80%    | 3.80%   | 4.90%     | 15.70%    | 6.50%      | 4.50%     | 11.80%      |
| Asian Alone               | 1.40%     | 6.30%   | 3.40%     | 1.60%     | 14.80%     | 3.30%     | 4.70%       |
| Hispanic or Latino        | 4.20%     | 7.00%   | 30.90%    | 7.30%     | 38.90%     | 21.30%    | 15.70%      |

5 rows x 50 columns

接下来我们删除df1多处来的列, 并且把index改为首字母大写:

```

1 # 删除dfq1的列
2
3 for i in df1.columns:
4     if i not in matchlist(dfcen_2016_filter,dfq1):
5         df1 = df1.drop(i,1)
6
7 # index不太好, 改个名吧 (在week07-guide中有讲)
8 df1.rename(index = lambda x: x.replace('handgun', 'Handgun'),
9             inplace = True)
10 df1.rename(index = lambda x: x.replace('others', 'Others'),
11             inplace = True)
12 df1

```

| state   | Alabama | Alaska | Arizona | Arkansas | California | Colorado | Connecticut | Delaware |
|---------|---------|--------|---------|----------|------------|----------|-------------|----------|
| Handgun | 5734.0  | 2320.0 | 11063.0 | 4347.0   | 37165.0    | 15751.0  | 4834.0      | 1414.0   |
| Others  | 6541.0  | 3149.0 | 8866.0  | 6228.0   | 27565.0    | 14455.0  | 2267.0      | 1604.0   |

## // 修改数据类型

有的数据是个数，有的数据是比例，接下来将购买枪的孩子从个数转成比率才能达成一致，结果开始做报错，结果发现dfcen的Pop这列是str字符型，字符串和数字是无法运算的（这是开始检查数据时候留的一个坑，开始改好就方便了，擦屁股，，），Pop的数据是正阳的 4,863,330 所以不是简单的转换，我们先要把逗号去掉：

```

1 # 使用.str.replace替换
2 # 官方说明 https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.str.replace.html
3 dfcen_2016_filter.loc['Population'] = dfcen_2016_filter.loc['Population']
4 ].str.replace(',', '')
5 # 看一眼数据
6 dfcen_2016_filter.iloc[0].head()
7
8
9 # 这回可以转换了
10 dfcen_2016_filter.iloc[0] = pd.to_numeric(dfcen_2016_filter.iloc[0])
11
12 # 看一下，普遍比较小，扩大100倍存档
13 df1.iloc[0].values / dfcen_2016_filter.loc['Population'].values
14 df1_r = df1.copy()
15 df1_r.shape
16 df1_r.iloc[0] = 100 * df1.iloc[0].values / dfcen_2016_filter.loc['Population'].values
df1_r.iloc[1] = 100 * df1.iloc[1].values / dfcen_2016_filter.loc['Population'].values
df1_r

```

| state   | Alabama  | Alaska   | Arizona  | Arkansas | California | Colorado | Connecticut | Delaware |
|---------|----------|----------|----------|----------|------------|----------|-------------|----------|
| Handgun | 0.117903 | 0.312713 | 0.159615 | 0.145470 | 0.094688   | 0.284286 | 0.135162    | 0.1485   |
| Others  | 0.134497 | 0.424454 | 0.127917 | 0.208416 | 0.070229   | 0.260895 | 0.063387    | 0.1684   |

接下来处理带%的数据：

```

1 # 干掉百分号
2 def hunchange(df, list):
3     for i in list:

```

```

4     df.loc[i] = df.loc[i].str.replace('%', '')
5
6 hunlist = dfcen_2016_filter.index[1:]
7
8 hunchange(dfcen_2016_filter, hunlist)
9
10 dfcen_2016_filter.head()

```

|                           | Alabama | Alaska | Arizona | Arkansas | California | Colorado | Connecticut |
|---------------------------|---------|--------|---------|----------|------------|----------|-------------|
| Population                | 4863300 | 741894 | 6931071 | 2988248  | 39250017   | 5540545  | 3576452     |
| White Alone               | 69.30   | 66.10  | 83.30   | 79.40    | 72.70      | 87.50    | 80.60       |
| Black or African American | 26.80   | 3.80   | 4.90    | 15.70    | 6.50       | 4.50     | 11.80       |
| Asian Alone               | 1.40    | 6.30   | 3.40    | 1.60     | 14.80      | 3.30     | 4.70        |
| Hispanic or Latino        | 4.20    | 7.00   | 30.90   | 7.30     | 38.90      | 21.30    | 15.70       |

最后按照列转换一下数据类型，并切合并两个数据就可以了：

```

1 # 因为dataframe的类型是按照列来定的，所以这里要转换列
2 humstat = dfcen_2016_filter.columns
3 for i in humstat:
4     dfcen_2016_filter[i] = pd.to_numeric(dfcen_2016_filter[i]) / 100
5 # .to_numeric就是咱们pd转换数字的方法
6
7 dfcen_2016_filter.head()
8 # OK 既然都转换完了，那么我们把Pop这行干掉，并合并数据
9 dfq1 = dfcen_2016_filter.drop(['Population'], 0)
10 dfq1
11 # 合并df1_r的数据
12 dfq1.append(df1_r)
13 # Perfect，可以画图了！

```

数据总算准备好了

| state                     | Alabama  | Alaska   | Arizona  | Arkansas | California | Colorado | Connecticut |
|---------------------------|----------|----------|----------|----------|------------|----------|-------------|
| White Alone               | 0.693000 | 0.661000 | 0.833000 | 0.794000 | 0.727000   | 0.875000 | 0.806000    |
| Black or African American | 0.268000 | 0.038000 | 0.049000 | 0.157000 | 0.065000   | 0.045000 | 0.118000    |
| Asian Alone               | 0.014000 | 0.063000 | 0.034000 | 0.016000 | 0.148000   | 0.033000 | 0.047000    |
| Hispanic or Latino        | 0.042000 | 0.070000 | 0.309000 | 0.073000 | 0.389000   | 0.213000 | 0.157000    |
| Handgun                   | 0.117903 | 0.312713 | 0.159615 | 0.145470 | 0.094688   | 0.284286 | 0.135162    |
| Others                    | 0.134497 | 0.424454 | 0.127917 | 0.208416 | 0.070229   | 0.260895 | 0.063387    |

6 rows × 50 columns

## /彩蛋/

## / FBI数据补充

- FBI数据的特点是需要将两个数据文件进行清理和规整
- 和Gapminder使用多个数据文件有点相似
- 其实上面最后做图的数据来讲还是有些问题，大家可以自己做的时候研究下：
  - 因为是按照bar图，所以行数太多，可以通过先总体观察，再把影响不大的数据干掉的方法降低（感觉4个会比较好，1个枪的数据，3个人口分类的数据）
  - 在开始并没有把数据类型做检查和转换，为中间的工作增加了复杂度
  - 州太多了，不好展示，要不要top10之类的来一波？

## / 注意事项

本导学内容是以P3中的FBI枪支数据为例进行的分析，请大家注意以下几点：

- 本文件还原了我在做项目时候的思考和解答过程，这个过程并不是最优的
- 主要是抛砖引玉将一个项目比较完整的过程展现出来，供大家参考下
- 按照项目中的建议，你可以先写一个代码多，尝试多‘比较乱’的版本，之后再提交时候再提交另一个版本。
- 不过如果你比较懒，如果能够满足‘项目评审标准’又能注意逻辑和描述，一个文件也是OK的
- 除了SQL的那个数据，我们有4个数据可以选的，兴趣是最好的出发点

- 加油加油，争取1周时间能够提交，之后再按照评审老师的反馈修改就更有针对性了。记住：**时间节点比质量更加重要！**（因为还有机会改的，不限次数）

Title: W9 项目总结 [项目: 探索数据集项目4/4]

Tags: 数据分析初级, 实战项目

# W9 项目总结 [项目: 探索数据集项目4/4]

- [W9 项目总结 \[项目: 探索数据集项目4/4\]](#)
- [/学习地图/](#)
  - [/ 项目路径](#)
  - [/ 项目推进](#)
- [/知识点回顾/](#)
- [/项目提交说明/](#)
- [/彩蛋/ - CheetSheet](#)

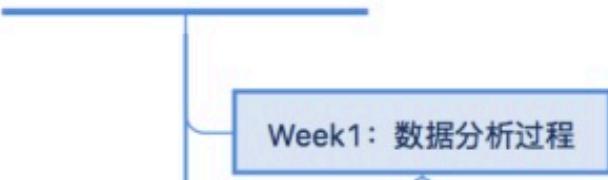
## /学习地图/

本周是数据分析课项目3的Part3，我们要开始做探索数据集项目了，兴奋不！其实经过前两周的洗礼，对于数据集的探索，我们已经比较熟悉了：除了课程的内容，我们还已经完成了两个案例的分析，也就是说这周我们做探索数据集项目时，已经是第4遍比较全面的完成这个流程了，加油吧，少年们！

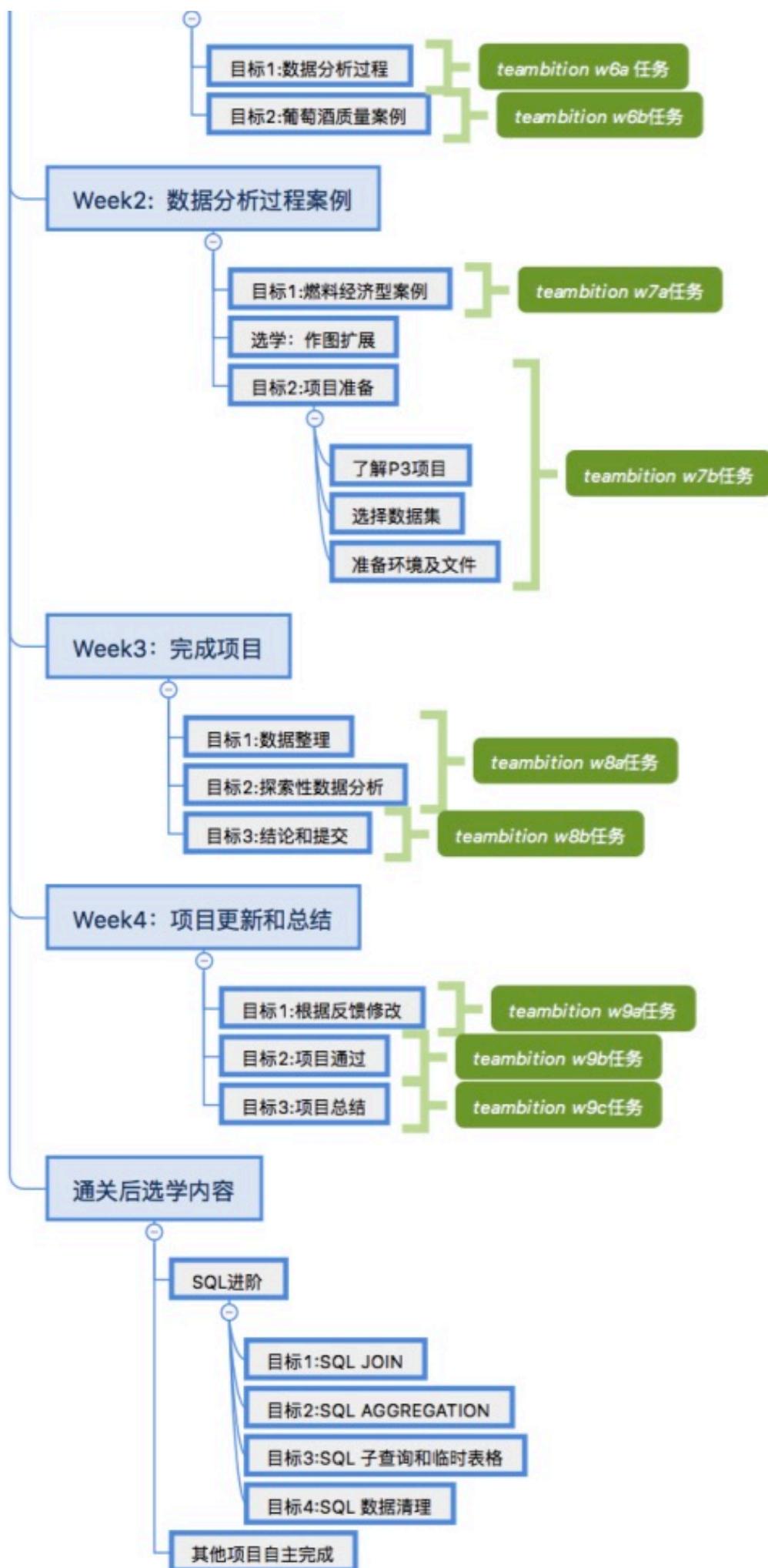
数据分析入门是数据分析微学位的第3个项目，项目时间为4周，在完成学习之后。可以从5个数据集中选取一个进行数据分析的项目提交。其中4个数据使用Python处理，是重点。如果习惯使用SQL进行处理，也有一个足球的数据集和相关内容（建议优先完成Python的，SQL的可以以后回来复盘再学）。这个项目的学习过程就比较完全，在课程中会串讲一遍知识点（第一遍讲解），之后是红酒的案例分析（一遍运用中讲解），最后是燃料经济学的案例分析（又是一遍运用中讲解）。在这3个部分都学习完成后，就可以着手选择项目完成了。需要注意，本部分不包括统计学的内容，将会在下一个部分引入。具体每周内容和知识点汇总见后面小节。

## / 项目路径

### 探索数据集



Week1: 数据分析过程



# / 项目推进

---

## Week1要求（已完成）：

- 完成项目简介部分
- 这次项目中有4个数据可以选择（新手不建议SQL足球那个），请根据项目说明，选一个可心的呦/[项目数据说明/](#)

## Week2要求（已完成）：

- 学习红酒质量分析案例
- 学习汽车燃料经济学案例

## Week3要求（已完成）：

- 浏览本导学文件
- 做完项目中的练习
- 根据导学文件，按照项目提示，一步步的完成项目
- 提交最好了！

## Week4要求（本周任务）

- 项目通关
- 知识点回顾
- 整理笔记

# /知识点回顾/

---

- Week6 数据分析过程
  - <http://t.cn/ReUxOla>
  - 数据分析的应用场景
  - 数据分析过程
  - pandas读入csv文件
  - dataframe的loc和iloc
  - pandas的.value\_count()方法
  - dataframe的空值处理
  - pandas绘图

- Week6 Plus1 酒品质分析案例

- <http://t.cn/RD2cANG>
- 重命名列
- 写入CSV的index参数
- 使用可视化探索数据
- Pandas Groupby
- 由值生成分类数据
- Pandas Query
- 图形展示入门
- ipynb和数据文件打包下载：
- <http://t.cn/RDhnxe4>

- Week7 汽车燃料经济性案例分析

- <http://t.cn/Re71w9t>
- 数据评估
- 清理列标签
- 列值类型转换
- 复合数据拆分
- 图示展示结论
- ipynb和数据文件打包下载：
- <http://t.cn/RDhnxe4>

- Week7 Plus1 介绍Matplotlib

- <http://t.cn/ReffqHI>
- Matplotlib是啥东东
- 画图的组织方式
- 画图的细节
- 做图标准代码模版
- 使用Pandas做图
- 后续学习
- 资料

- Week8 项目TMDB数据导学

- <http://t.cn/ReQ2hxm>
- 数据整理 (TMDB数据)
  - 准备
  - 读入数据
  - 检查数据
  - 清理数据
- 探索性数据分析 (TMDB数据)
  - 提出问题1 - 几个数字值的关系如何
  - 提出问题2 - 是否有Website和Feature的关系

- 得出结论 (TMDB数据)
  - 数据集说明
  - 研究问题1 - 几个数字值的关系如何
    - 问题观察和结论
    - 问题的图形展示
    - 思考和建议
  - 研究问题2 - 是否有Website和Feature的关系
  - 探索数据集项目
- Week8 Plus1 项目FBI枪支数据导学
  - <http://t.cn/RDh36Sg>
  - 数据整理 (FBI数据)
    - 读入数据 (Excel)
    - 检查数据
    - 清理数据 (两个数据文件结合清理)

## /项目提交说明/

---

- 项目数据集选择: <http://t.cn/RejvMmS>
- 项目环境准备:
  - Uda环境完成: 可以在 |3 实战项目 这一节完成, 但是要把下载好的数据文件通过 File - Open - Upload 上传到跟目录下。昨晚之后再下载html或者pdf文件在最后的 |4 项目: 探索数据集 右下角点击 '提交项目' 提交。
  - 本地环境完成: 可以在 |3 实战项目 这一节下载ipynb模版文件, 之后在本地做完, 把最后的文件同样点击 '提交项目' 提交。这个模版文件我搬运了下: [https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/Investigate\\_a\\_Dataset-zh.ipynb](https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project3/Investigate_a_Dataset-zh.ipynb)
- 开始写项目文件
  - 1 数据整理
  - 2 探索性数据分析
  - 3 得出结论

## /彩蛋/ - CheetSheet

---

找到了一套Python的CheatSheet, 里面的Matplotlib、Pandas Basic、Pandas3个比较适合这个项目, 也可以打印出来看: <http://t.cn/RD2J9lj>

Title: W9 Plus 其他数据讲解

Tags: 数据分析初级, 实战项目

# W9 Plus 其他数据讲解

---

- [W9 Plus 其他数据讲解](#)
- [/FBI数据讲解/](#)
  - [/ 1.准备](#)
  - [/ 2.读入数据](#)
  - [/ 3.检查数据](#)
  - [/ 4.清理数据](#)
    - [// 先处理FBI数据](#)
    - [// 再处理Census数据](#)
    - [// 处理两个数据列数不同的问题](#)
    - [// 修改数据类型](#)
- [/FBI数据补充/](#)
- [/彩蛋/](#)

## /FBI数据讲解/

---

### / 1.准备

---

其实在做项目的时候，大家一定要：认真看模版文件！！！不但有整个报告的结构在里面，很多地方还是有提示的。那么在开始我们要干什么事情呢：

- 从头到尾浏览下项目模版文件（链接在本文开始）
- 认真看下评审说明：<https://review.udacity.com/#!/rubrics/306/view>
- 写好import代码框，一般来讲就是这几个家伙了（提示好nice的）：

```
1 # 用这个框对你计划使用的所有数据包进行设置
2 # 导入语句
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import numpy as np
```

```
7  
8 # 务必包含一个'magic word'，以便将你的视图显示出来  
9 %matplotlib inline  
10  
11 # magicwords的说明：  
12 # http://ipython.readthedocs.io/en/stable/interactive/magics.html
```

## / 2. 读入数据

接下来就是把你下载好的数据加载，如果下载遇到问题可以找我友情搬运（保证不干坏事），但要注意以下情况：

- 下载后先检查数据的扩展名，看下大小（一般不会特别大）
- 注意用excel/numbers打开数据浏览下内容，有个直观的概念
- 读取csv的时候看下需不需要加参数
- 读取完成后，检查数据是否如预期

```
1 # 这里我们要加载xlsx文件，可以使用xlrd第三方库来，有不少选择  
2 # 我当然还是用大熊猫了read_excel  
3 df = pd.read_excel('gun_data.xlsx')  
4 # 注意这里面也可以加参数，和csv的一样，请见以前的导学文件  
5 # 看下都有哪些列  
6 df.head()  
7 # 看下head，分析下数据列的意思（本数据没有列说明）
```

观察了下，数据的结构是这样的（前面是列名）：

- month/state - 按照每个月每个周一的数据，开始第一条是2017-09 Alabama以此类推
- permit/permit\_check - permit应该是批准数（执枪护照），permit check是不是核实，这个数据从比较晚才开始有，而且其中的数量有很多是0，在没有详细说明的情况下不太好使用
- handgun/long\_gun/other - 应该是枪的分类（other是机关枪，榴弹炮，生化危机片场？）但是这3个数字加一起和permit的数量非常不一样，有时候多有时候少，而且差别很大
- multiple - 应该是有2种以上的的
- 其他列也是各种的奇怪名称
- 最后一个是total，这个数是和其他的所有数只和相同的
- 所以呢推测这个数据是NICS（FBI的潜质检查数据）的每月的使用记录情况。其中每一列无论是发许可（permit）还是买枪（handgun）或者是退货（return to handgun）都要在NICS中做记录，这一行数据是NICS一个月的记录数
- 我们来用代码检验一下：

```
1 # 先看下第一行的数据
2 print(df.iloc[0])
3
4 # 按照推论, 第3个数据开始至倒数第1个数据的和应该等于最后一列的total
5 df.iloc[0][2:-1].sum() == df.iloc[0][-1]
```

```
1 month          2017-09
2 state          Alabama
3 permit         16717
4 permit_recheck      0
5 handgun        5734
6 long_gun       6320
7 other           221
8 multiple         317
9 admin            0
10 prepawn_handgun    15
11 prepawn_long_gun   21
12 prepawn_other        2
13 redemption_handgun  1378
14 redemption_long_gun 1262
15 redemption_other        1
16 returned_handgun      0
17 returned_long_gun      0
18 returned_other          0
19 rentals_handgun        0
20 rentals_long_gun        0
21 private_sale_handgun     9
22 private_sale_long_gun    16
23 private_sale_other        3
24 return_to_seller_handgun    0
25 return_to_seller_long_gun  0
26 return_to_seller_other        3
27 totals          32019
28 Name: 0, dtype: object
29
30 True
```

那么按照这个猜测的话, 我们不用管那么多列的数据, 我比较关注的是handgun、long\_gun、other这3种枪的关系 (会不会发现一些地域特征? 对于米国地理一窍不通的我表示很为难)

另外一个数据是U.S. Census Data.csv, 是各洲的人口统计信息。我们来看下, 结果发现这个数据是按照法每行一次普查为数据行, 每个洲的人数为列:

```
1 dfcen = pd.read_csv('U.S. Census Data.csv')
```

```
2 | dfcen.head()
```

|   | Fact                                        | Fact Note | Alabama | Alaska    | Arizona | Arkansas  | California | Colorado   | Connecticut | Delaware  | ...     | South Dakota | Tennessee | Texas   | Utah       |           |
|---|---------------------------------------------|-----------|---------|-----------|---------|-----------|------------|------------|-------------|-----------|---------|--------------|-----------|---------|------------|-----------|
| 0 | Population estimates, July 1, 2016, (V2016) |           | NaN     | 4,863,300 | 741,894 | 6,931,071 | 2,988,248  | 39,250,017 | 5,540,545   | 3,576,452 | 952,065 | ...          | 865454    | 6651194 | 27,862,596 | 3,051,217 |

那么这个数据我们怎么用呢，有几个想法：

- 如果是只看最近的数据和做分析，那么就使用census最后一行2016年相关的数据就好了
- 之所以说是相关的数据，是2016、2012、2010年的数据除了人数还有很多的百分比数据，比如6岁以下，65岁以上什么的（就是可以作为分类信息了）
- 其他大概70条之后的数据有比较奇怪的title，pass不予考虑（除非有详细的说明）

现在我们有了2个数据：

- 一个是从1999年开始的枪支数据，行为月/州，列为各种枪支数据统计
- 另一个是人口统计数据，行为年/统计特征，列为各州的数量
- 第一个文件州是行，第二个文件州是列，这两个文件的相同点是州
- 有的时候这种情况需要进行行列转换，但这个案例我觉得把第二个文件抽出部分数据更加实用
- 我来过滤下第二个文件包含2016年的行，看看都有什么内容（在检查数据中完成）

## / 3. 检查数据

首先读入Census数据，之后观察到Fact这一列是数值，我们再把Fact检索出来观察名字：

```
1 | dfcen = pd.read_csv('U.S. Census Data.csv')
2 | dfcen.head()
3 | dfcen['Fact'].values
4 | # 使用.values直接输出值，不看索引会简洁些
```

之后发现2016年的数据条目都有V2016这个字符作为标识，我们先去掉空值，再做筛选（否则筛选会报错）：

```
1 | # 去掉空值，存为新的数据(多次运行shape可以看出对比)
2 | print(dfcen.shape)
3 | dfcen_edited = dfcen[dfcen['Fact'].notnull()]
4 | print(dfcen_edited.shape)
5 |
6 | # 过滤
7 | dfcen_2016 = dfcen_edited[dfcen_edited['Fact'].str.contains("2016")]
```

```
8 # 也可以加个na = False参数
9 # https://stackoverflow.com/questions/28311655/ignoring-nans-with-str-co
10 ntains
11 # df[df.fff xxxx] 是 operation chain 的方法, 举例:
12 # https://stackoverflow.com/questions/11869910/pandas-filter-rows-of-dat
13 aframe-with-operator-chaining/28159296#28159296
14 # 使用chain方法对str进行筛选:
15 # https://stackoverflow.com/questions/32616261/filtering-pandas-datafram
e-rows-by-contains-str
# pandas 还有 .filter的方法也可以实现
dfcen_2016.shape
```

过滤出来17条数据, 这部分的代码结果:

```
1 (85, 52)
2 (80, 52)
3 (17, 52)
4
5 array(['Population estimates, July 1, 2016, (V2016)',
6         'Population estimates base, April 1, 2010, (V2016)',
7         'Population, percent change - April 1, 2010 (estimates base) to J
8 uly 1, 2016, (V2016)',
9         'Persons under 5 years, percent, July 1, 2016, (V2016)',
10        'Persons under 18 years, percent, July 1, 2016, (V2016)',
11        'Persons 65 years and over, percent, July 1, 2016, (V2016)',
12        'Female persons, percent, July 1, 2016, (V2016)',
13        'White alone, percent, July 1, 2016, (V2016)',
14        'Black or African American alone, percent, July 1, 2016, (V2016)
15        ',
16        'American Indian and Alaska Native alone, percent, July 1, 2016,
17 (V2016)',
18        'Asian alone, percent, July 1, 2016, (V2016)',
19        'Native Hawaiian and Other Pacific Islander alone, percent, July
20 1, 2016, (V2016)',
21        'Two or More Races, percent, July 1, 2016, (V2016)',
        'Hispanic or Latino, percent, July 1, 2016, (V2016)',
        'White alone, not Hispanic or Latino, percent, July 1, 2016, (V2
016)',

        'Housing units, July 1, 2016, (V2016)', 'Building permits, 2016
'],
        dtype=object)
```

可以看到这个信息里除了信息之外有两个方面的数据

- 人口年龄组成数据 (按照年龄) 貌似和持枪关系不大 (5岁以下, , , , 买把真枪玩

玩? ) , 这个方面pass

- 人口成分组成, 比如女性、白人单身、黑人或非洲裔、美国土著、亚洲人等等, 这些分类可以和各州的比率比较看可以发现什么

综上, 结合fbi的枪支管制数据我们就可以发现很多问题了, 举例两个:

- (最近数据) 各州购买枪的比率和各州人员成分比例有关不? (本导学文件介绍)
- 选中的州 (可以是排名靠前的几个), 的购买频次如何 (结合第二个看, 和米国一些法规发布有关系, 对于米国不是特别感兴趣, 最后会放上个资料, 大家参考一下)

## / 4.清理数据

- df只保留最近时间 (2017-09) 的数据 (为了防止某个月有个别事件有影响, 也可以取1年的做平均值)
- 存为新的df1数据, 列只保留handgun、long\_gun、other并将后两者合并为1列 (参见New York Times的2个分类)

## // 先处理FBI数据

```
1 # 把我们要的时间选出来
2 dftemp = df[df['month'].str.contains('2017-09')]
3 print(dftemp.shape)
4 # 把需要的3个列选出来, 存为新的数据q1是question1的简写
5 df1 = pd.DataFrame(dftemp.loc[:, 'handgun':'other'])
6 # 看了下, 还要把index给指定成洲的名字
7 df1.index = dftemp['state']
8 # 将其他2个列合并为Others
9 df1['others'] = df1['long_gun'] + df1['other']
10 df1 = df1.drop(['other', 'long_gun'], 1)
11 # 检查下
12 print(df1)
13 # 转换行列, 因为我们看以洲为输入的图
14 df1 = df1.T
15 df1
```

结果就是这个样子滴:

```
state Alabama Alaska Arizona Arkansas California Colorado Connecticut Delaware
```

|         |        |        |         |        |         |         |        |        |
|---------|--------|--------|---------|--------|---------|---------|--------|--------|
| handgun | 5734.0 | 2320.0 | 11063.0 | 4347.0 | 37165.0 | 15751.0 | 4834.0 | 1414.0 |
| others  | 6541.0 | 3149.0 | 8866.0  | 6228.0 | 27565.0 | 14455.0 | 2267.0 | 1604.0 |

2 rows × 55 columns

## // 再处理Census数据

看了下，其实要保留的列也不是特别多，我们来看下dfcen\_2016['Fact']这一列（列名太凶残了，我还是用index筛选数据好了）

```
1 dfcen_2016['Fact']
```

```
0 Population estimates, July 1, 2016, (V2016)
1 Population estimates base, April 1, 2010, (V2...
2 Population, percent change - April 1, 2010 (es...
4 Persons under 5 years, percent, July 1, 2016, ...
6 Persons under 18 years, percent, July 1, 2016, ...
8 Persons 65 years and over, percent, July 1, 2...
10 Female persons, percent, July 1, 2016, (V2016)
12 White alone, percent, July 1, 2016, (V2016)
13 Black or African American alone, percent, July...
14 American Indian and Alaska Native alone, perce...
15 Asian alone, percent, July 1, 2016, (V2016)
16 Native Hawaiian and Other Pacific Islander alo...
17 Two or More Races, percent, July 1, 2016, (V2...
18 Hispanic or Latino, percent, July 1, 2016, (V...
19 White alone, not Hispanic or Latino, percent, ...
22 Housing units, July 1, 2016, (V2016)
29 Building permits, 2016
Name: Fact, dtype: object
```

由于FBI的数据是数量，而Census里的人口分类信息是百分比，所以我们保留全部人口，用作FBI数据的百分比转换（第0行）；再就是保留第12, 13, 15, 18行（保持在统一维度并且干掉过小的值），经过观察各洲的分布情况还真的有很大区别。

```
1 # 筛选出行
2 dfcen_2016_filter = dfcen_2016.loc[[0,12,13,15,18],:]
3 # 干掉Fact Note
4 dfcen_2016_filter = dfcen_2016_filter.drop('Fact Note', 1)
5 print(dfcen_2016_filter.shape)
```

## // 处理两个数据列数不同的问题

```
| # 55 vs 51 看来两个数据的洲不对应，需要处理一下
```

```

1 # 来个小循环把相同的列找出来
2 def matchlist(a,b):
3     matchlist = []
4     for i in list(a.columns):
5         if i in list(b.columns):
6             matchlist.append(i)
7         else:
8             print(i)
9     return matchlist
10
11
12 # 有50个ok的，而且cen是被dfq1包含的
13
14 matchlist(dfq1,dfcen_2016_filter)
15 matchlist(dfcen_2016_filter,dfq1)
16 # 本来想把Fact这列变成index，名字太长了，自己指定把
17 cenindex = ['Population', 'White Alone',
18             'Black or African American', 'Asian Alone',
19             'Hispanic or Latino']
20 dfcen_2016_filter.index = cenindex
21
22 # 删除dfcen[Fact]
23 if 'Fact' in dfcen_2016_filter.columns:
24     dfcen_2016_filter = dfcen_2016_filter.drop(['Fact'],1)
25
26 dfcen_2016_filter
27

```

dfcen\_2016\_filter做了规整，还是比较好看：

|                           | Alabama   | Alaska  | Arizona   | Arkansas  | California | Colorado  | Connecticut |
|---------------------------|-----------|---------|-----------|-----------|------------|-----------|-------------|
| Population                | 4,863,300 | 741,894 | 6,931,071 | 2,988,248 | 39,250,017 | 5,540,545 | 3,576,452   |
| White Alone               | 69.30%    | 66.10%  | 83.30%    | 79.40%    | 72.70%     | 87.50%    | 80.60%      |
| Black or African American | 26.80%    | 3.80%   | 4.90%     | 15.70%    | 6.50%      | 4.50%     | 11.80%      |
| Asian Alone               | 1.40%     | 6.30%   | 3.40%     | 1.60%     | 14.80%     | 3.30%     | 4.70%       |
| Hispanic or Latino        | 4.20%     | 7.00%   | 30.90%    | 7.30%     | 38.90%     | 21.30%    | 15.70%      |

5 rows × 50 columns

接下来我们删除df1多处来的列，并且把index改为首字母大写：

```

1 # 删除dfq1的列
2
3 for i in df1.columns:
4     if i not in matchlist(dfcen_2016_filter,dfq1):
5         df1 = df1.drop(i,1)
6
7 # index不太好看，改个名吧（在week07-guide中有讲）
8 df1.rename(index = lambda x: x.replace('handgun', 'Handgun'),
9             inplace = True)
10 df1.rename(index = lambda x: x.replace('others', 'Others'),
11            inplace = True)
12 df1

```

| state   | Alabama | Alaska | Arizona | Arkansas | California | Colorado | Connecticut | Delaware |
|---------|---------|--------|---------|----------|------------|----------|-------------|----------|
| Handgun | 5734.0  | 2320.0 | 11063.0 | 4347.0   | 37165.0    | 15751.0  | 4834.0      | 1414.0   |
| Others  | 6541.0  | 3149.0 | 8866.0  | 6228.0   | 27565.0    | 14455.0  | 2267.0      | 1604.0   |

## // 修改数据类型

有的数据是个数，有的数据是比例，接下来将购买枪的孩子从个数转成比率才能达成一致，结果开始做报错，结果发现dfcen的Pop这列是str字符型，字符串和数字是无法运算的（这是开始检查数据时候留的一个坑，开始改好就方便了，擦屁股，，），Pop的数据是正阳的 4,863,330 所以不是简单的转换，我们先要把逗号去掉：

```

1 # 使用.str.replace替换
2 # 官方说明 https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.str.replace.html
3 dfcen_2016_filter.loc['Population'] = dfcen_2016_filter.loc['Population']
4 .str.replace(',', '')
5 # 看一眼数据
6 dfcen_2016_filter.iloc[0].head()
7
8 # 这回可以转换了
9 dfcen_2016_filter.iloc[0] = pd.to_numeric(dfcen_2016_filter.iloc[0])
10
11 # 看一下，普遍比较小，扩大100倍存档
12 df1.iloc[0].values / dfcen_2016_filter.loc['Population'].values
13 df1_r = df1.copy()
14 df1_r.shape
15 df1_r.iloc[0] = 100 * df1.iloc[0].values / dfcen_2016_filter.loc['Popula
16 tion'].values
17 df1_r.iloc[1] = 100 * df1.iloc[1].values / dfcen_2016_filter.loc['Popula
18 tion'].values

```

```
df1_r
```

| state   | Alabama  | Alaska   | Arizona  | Arkansas | California | Colorado | Connecticut | Delaware |
|---------|----------|----------|----------|----------|------------|----------|-------------|----------|
| Handgun | 0.117903 | 0.312713 | 0.159615 | 0.145470 | 0.094688   | 0.284286 | 0.135162    | 0.1485   |
| Others  | 0.134497 | 0.424454 | 0.127917 | 0.208416 | 0.070229   | 0.260895 | 0.063387    | 0.1684   |

接下来处理带%的数据：

```
1 # 干掉百分号
2 def hunchange(df,list):
3     for i in list:
4         df.loc[i] = df.loc[i].str.replace('%','')
5
6 hunlist = dfcen_2016_filter.index[1:]
7
8 hunchange(dfcen_2016_filter, hunlist)
9
10 dfcen_2016_filter.head()
```

|                           | Alabama | Alaska | Arizona | Arkansas | California | Colorado | Connecticut |
|---------------------------|---------|--------|---------|----------|------------|----------|-------------|
| Population                | 4863300 | 741894 | 6931071 | 2988248  | 39250017   | 5540545  | 3576452     |
| White Alone               | 69.30   | 66.10  | 83.30   | 79.40    | 72.70      | 87.50    | 80.60       |
| Black or African American | 26.80   | 3.80   | 4.90    | 15.70    | 6.50       | 4.50     | 11.80       |
| Asian Alone               | 1.40    | 6.30   | 3.40    | 1.60     | 14.80      | 3.30     | 4.70        |
| Hispanic or Latino        | 4.20    | 7.00   | 30.90   | 7.30     | 38.90      | 21.30    | 15.70       |

最后按照列转换一下数据类型，并切合并两个数据就可以了：

```
1 # 因为dataframe的类型是按照列来定的，所以这里要转换列
2 humstat = dfcen_2016_filter.columns
3 for i in humstat:
4     dfcen_2016_filter[i] = pd.to_numeric(dfcen_2016_filter[i]) / 100
5 # .to_numeric就是咱们pd转换数字的方法
6
7 dfcen_2016_filter.head()
8 # OK 既然都转换完了，那么我们把Pop这行干掉，并合并数据
9 dfq1 = dfcen_2016_filter.drop(['Population'],0)
10 dfq1
```

```
11 # 合并df1_r的数据  
12 dfq1.append(df1_r)  
13 # Perfect, 可以画图了!
```

数据总算准备好了

| state                     | Alabama  | Alaska   | Arizona  | Arkansas | California | Colorado | Connecticut |
|---------------------------|----------|----------|----------|----------|------------|----------|-------------|
| White Alone               | 0.693000 | 0.661000 | 0.833000 | 0.794000 | 0.727000   | 0.875000 | 0.806000    |
| Black or African American | 0.268000 | 0.038000 | 0.049000 | 0.157000 | 0.065000   | 0.045000 | 0.118000    |
| Asian Alone               | 0.014000 | 0.063000 | 0.034000 | 0.016000 | 0.148000   | 0.033000 | 0.047000    |
| Hispanic or Latino        | 0.042000 | 0.070000 | 0.309000 | 0.073000 | 0.389000   | 0.213000 | 0.157000    |
| Handgun                   | 0.117903 | 0.312713 | 0.159615 | 0.145470 | 0.094688   | 0.284286 | 0.135162    |
| Others                    | 0.134497 | 0.424454 | 0.127917 | 0.208416 | 0.070229   | 0.260895 | 0.063387    |

6 rows × 50 columns

## /FBI数据补充/

- FBI数据的特点是需要将两个数据文件进行清理和规整
- 和Gapminder使用多个数据文件有点相似
- 其实上面最后做图的数据来讲还是有些问题，大家可以自己做的时候研究下：
  - 因为是按照bar图，所以行数太多，可以通过先总体观察，再把影响不大的数据干掉的方法降低（感觉4个会比较好，1个枪的数据，3个人口分类的数据）
  - 在开始并没有把数据类型做检查和转换，为中间的工作增加了复杂度
  - 州太多了，不好展示，要不要top10之类的一波？

## /彩蛋/

本导学内容是以P3中的FBI枪支数据为例进行的分析，请大家注意以下几点：

- 本文件还原了我在做项目时候的思考和解答过程，这个过程并不是最优的
- 主要是抛砖引玉将一个项目比较完整的过程展现出来，供大家参考下
- 按照项目中的建议，你可以先写一个代码多，尝试多‘比较乱’的版本，之后再提交时候再提交另一个版本。

- 不过如果你比较懒，如果能够满足‘项目评审标准’又能注意逻辑和描述，一个文件也是OK的
- 除了SQL的那个数据，我们有4个数据可以选的，兴趣是最好的出发点
- 加油加油，争取1周时间能够提交，之后再按照评审老师的反馈修改就更有针对性了。记住：**时间节点比质量更加重要！**（因为还有机会改的，不限次数）

Title: W10 统计学入门 [项目: 分析A/B测试结果1/3]

Tags: 数据分析初级, 实战项目

# W10 统计学入门 [项目: 分析A/B测试结果1/3]

---

- [W10 统计学入门 \[项目: 分析A/B测试结果1/3\]](#)
- [/学习地图/](#)
  - / 项目路径
- [/目标1:/ 描述统计学第一部分](#)
  - \*/ 5.数据类型
  - \*/ 12.概括统计简介
  - \*/ 19.表达式是什么
  - \*/ 22.变量的大小写规则
  - \*/ 25.求和
  - \*/ 26.均值表达式
- [/目标2\(选修\):/ 描述统计学第二部分](#)
  - \*\*/ 1-12.什么是离散程度测量
  - \*/ 13.形状
  - \*/ 20.21.形状与异常值
  - \*/ 27.描述统计与推论统计
- [/目标3: / 概率与二项式](#)
  - \*\*\*/ 概率 (课程4)
  - \*\*/ 二项式 (课程5)
- [/目标4: / 条件概率、贝叶斯规则与Python概率练习](#)
  - \*\*/ 条件概率 (课程6)
  - \*/ 贝叶斯规则 (课程7)
  - \*\*/ Python概率练习 (课程8)
- [/目标5: / 抽样分布与中心极限定理 \(课程10\)](#)
- [/彩蛋/ 统计学深入资料](#)

## /学习地图/

---

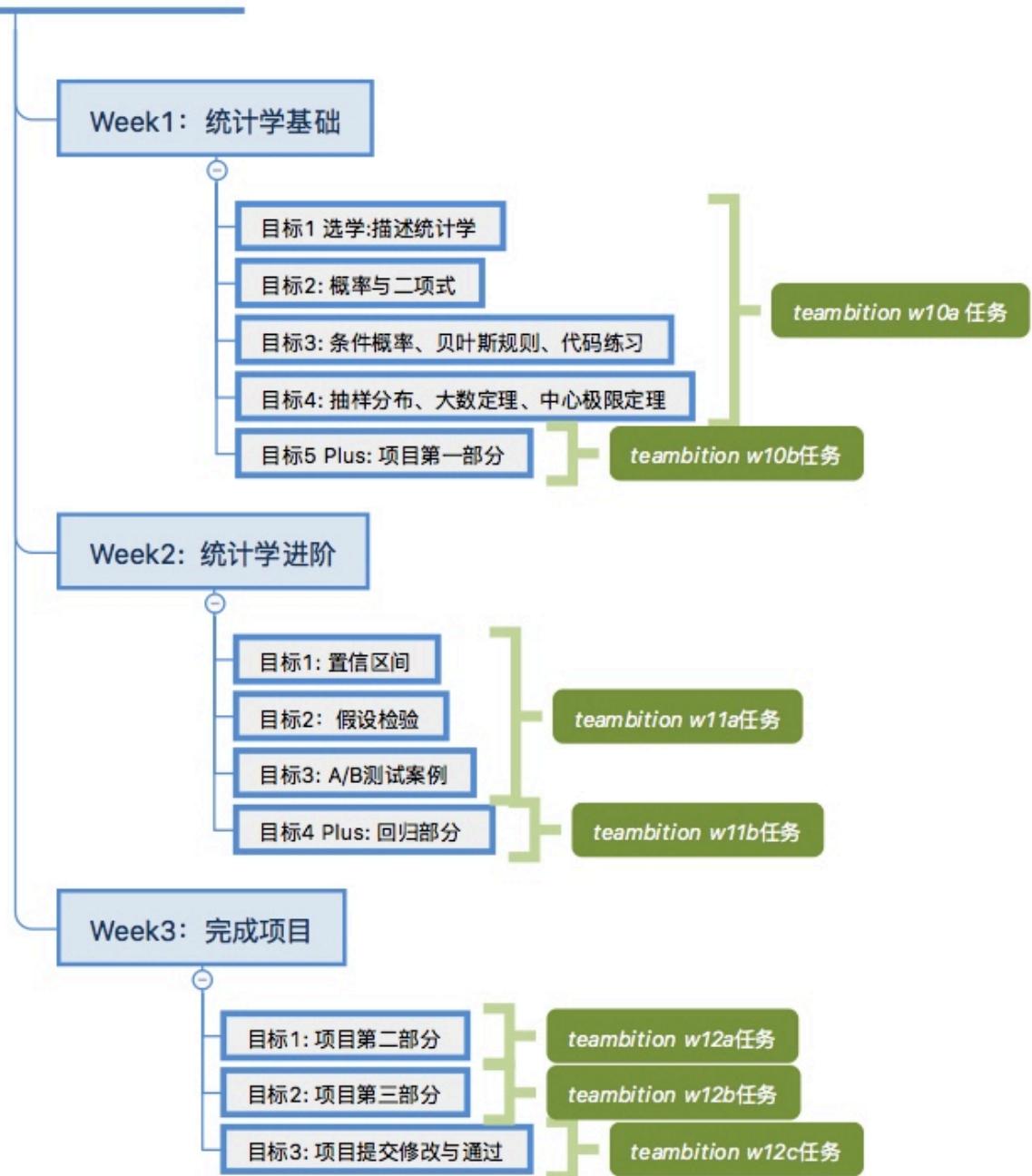
本周是统计学项目的Part1，是项目3的基础内容。导学分为两个部分，目标1、2、3、4是对统计

学的讲解（本文件）。目标5是项目的第一部分完成（不用担心，这一部分大都是项目3中做过的）。统计学是本项目重点，请不要被一堆公式吓一跳，项目中用到的都是基础知识。ps：课程里的内容挺多的

！注意，1星和2星的可以只看本导学，先大致理解就可以。3星的和项目第一部分是本周重点！

## / 项目路径

### 分析 A/B 测试结果



## /目标1:/ 描述统计学第一部分

## \* / 5.数据类型

明白统计学中的两种数量类型：

- Quantitative Data 数值数据：数值数据采用允许我们执行数学运算（例如计算狗的数量）的数值。可以使用数学运算。数量数据又可以分为以下两类（|8 数据类型（连续与离散））
  - Continuous Data 连续数据：比如说狗的年龄。连续数据可以分为更小的单位，并且仍然存在更小的单位。我们可以以年、月、日、小时、秒为单位测量狗的年龄，但是仍然存在可以与年龄关联的更小单位。
  - Discrete Dat 离散数据：比如说狗的数量，仅采用可数值。
- Categorical Data 分类数据：分类数据用于标记一个群体或一组项目（例如狗的品种——牧羊犬、拉布拉多、贵宾犬等）。用来标记一个群体或一组条目。分类数据又可以分成以下两类（|7 数据类型（定序与定类））
  - Ordinal (Ordered) 分类定序：比如说 Rating 排名
  - Nominal (No Order) 分类定类：比如说 Breed 狗的种类
- 课程中举例的是老师对于从眼前走过狗狗的统计，数量（几只）和种类（那种狗狗）：

图1:数据数据与分类数据

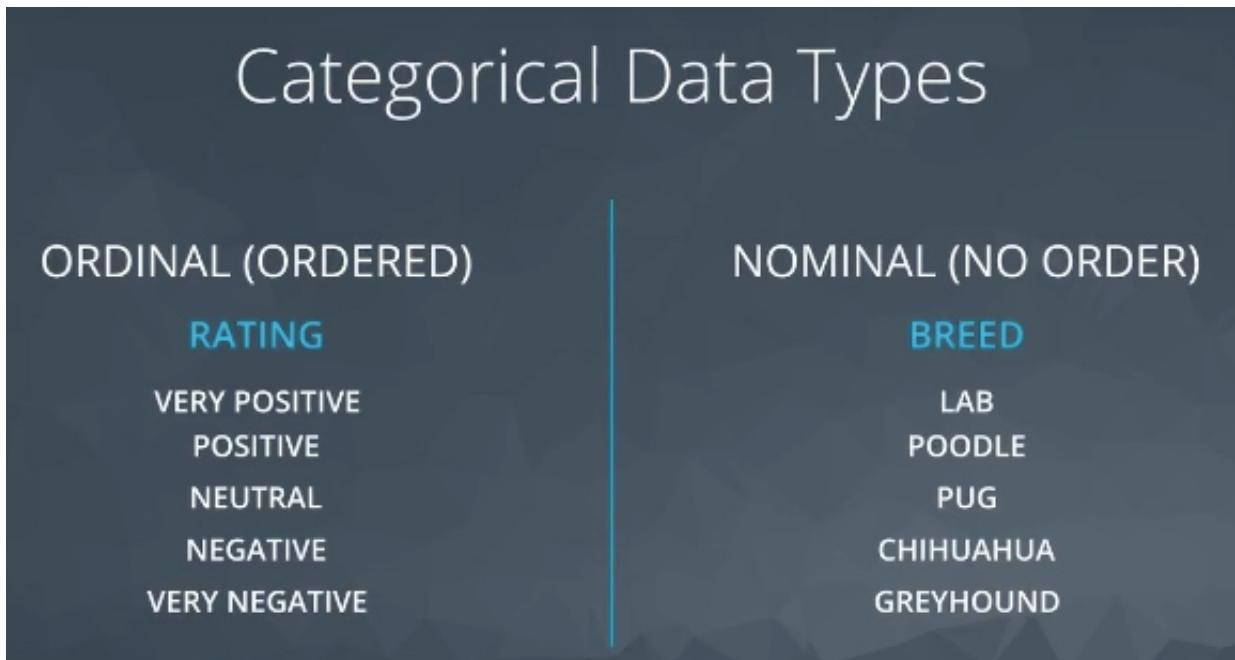
| Quantitative   | Categorical   |
|----------------|---------------|
| NUMBER OF DOGS | BREED OF DOGS |
| 0              | Lab           |
| 1              | Pug           |
| 2              | Poodle        |

图2:数据数据：为什么狗的年龄是连续值

## Continuous Data



图3:分类数据：分类定序和分类定类的区别



|      |                 |                     |
|------|-----------------|---------------------|
| /数值/ | 连续: 身高、年龄、收入    | 离散: 书中的页数、咖啡店出现狗的只数 |
| /分类/ | 定序: 字母成绩等级、调查评级 | 定类: 性别、婚姻状况、早餐食品    |

## \*/ 12. 概括统计简介

了解了什么是数值数据 (Quantitative Data) 后，大家还记得前几个项目中我们经常采用均值作为数据分析的一项内容么？当你听到“集中趋势测量”的时候，就是指的这个均值（你不但已经会了，都用过好多次了，是不是心情一下很好？）。

但是均值又个问题，比如说计算平均收入：当把我和老婆的收入区平均值，就能算出我家每个人的平均收入，这个有点意义。但是如果我在电梯里面遇到马云，那么即使当时在电梯里面的平均

收入有马云的一半，对我还是没有任何意义。在这种场景下，我们需要进行“离散程度”的测量了，这是一种测量每个值和平均值差异的方法：

- Measures of Center 集中趋势测量（平均数是一种方法，了解学员的一般情况）
- Measures of Spread 离散程度测量（方差，了解学员之间的差异）

除此之外，我们还要观察下数据的形状Shape（熟悉不？）和异常值Outliers（就是有问题的值）

所以说呢，当我们分析数值时，是从这4个方面下手的：

- Center 集中趋势测量：
  - 方法1 均值Mean: Sum of all values divided by the count of values (平均数)
  - 方法2 中位数Median: It is the middle value of a data set (中间那个数，如果是偶数的话，就是把中间两个的值取平均数)
  - 方法3 众数Mode: The most frequent number in a data set (出现次数最多的数；如果次数都相等，就出现无众数的情况；如果有两都出现最多，那么众数就是多个)
- Spread 离散程度测量
- Shape 数据的形状
- Outliers 异常值

## \*/ 19. 表达式是什么

---

这里来扩展下“表达式”是什么（其实加减乘除就是了，你一只在用）。表达式Notation: Common Math language used to communicate 是用来交流的常用数学语言的。可以非常简单（比如说 $5+3$ 中的 $+$ ）也可以很复杂，比如这个梯度上升的例子：[https://en.wikipedia.org/wiki/Gradient\\_boosting](https://en.wikipedia.org/wiki/Gradient_boosting)

## \*/ 22. 变量的大小写规则

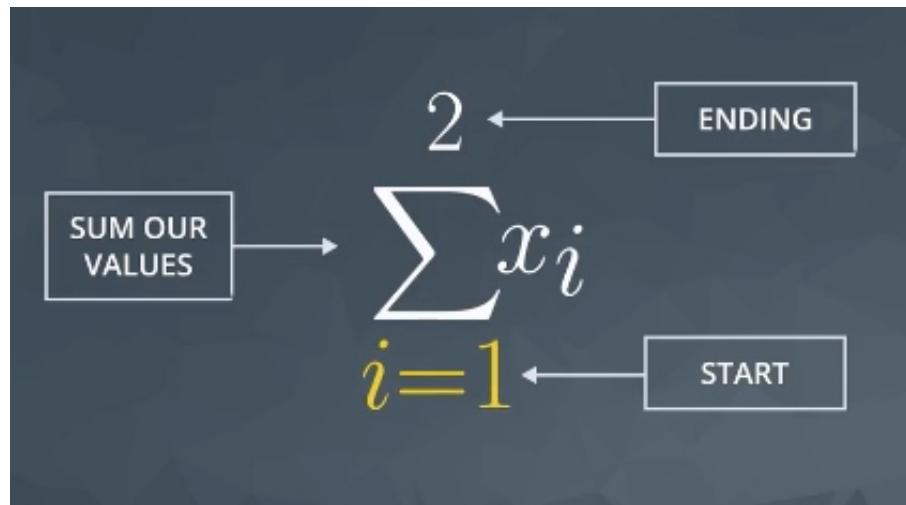
---

大写X代表所有变量，小写x代表单个值 $x_1, x_2, x_n$ 。

## \*/ 25. 求和

---

概率中的聚合有很多种，比如求和公式：



$\Sigma$  符号用于使用求和进行聚合，但是我们可以选择通过其他方式进行聚合。求和是最常见的聚合方式之一。但是，我们可能需要以其他方式进行聚合。如果我们想将所有的值相乘，我们可以使用求积符号  $\Pi$ ，希腊字母  $\pi$  的大写。我们聚合连续值的方式称为积分（微积分中的一种常用技术），它使用以下符号  $\int$ ，就像一个拉长的 s。我们不会在此课的练习中使用积分或乘积，但你将来会见到！

## \*/ 26. 均值表达式

$\bar{x}$ -bar就是在x上吗加个横杠，表示x的均值：

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

## /目标2(选修):/ 描述统计学第二部分

### \*\*/ 1-12. 什么是离散程度测量

Measures of Spread : How far are points from one another

- 5数概括法：Min、Q1、Q2（mean均值）、Q3、最大值（.describe的输出就是这5个数）
- 极差 Range = Max - Min
- 四分卫差 Interquartile Range (IQR) = Q3 - Q1 (中间50%的数据区间)

- 标准差 Standard Deviation: On average, how much each point varies from the mean of the points (每个观测值与均值之差的平均值) 标准差是方差的平方根
- 方差 Variance: Average squared difference of each observation from the mean
- 

**VARIANCE**  $\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$

- 几个概念的总结: <https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/parts/4e7e2f82-e05e-4fbe-b29c-fe3169c6dd77/modules/0596b9e8-4a3a-41c3-a929-6c72c0c93925/lessons/c5b785aa-a0e9-45ac-8c4f-6e192829babe/concepts/5f2d9388-2cdc-46d5-81f8-8293edaafdb4>
- 使用标准差的好处是使用原始数据的单位（方差带平方运算，所以单位不同），意义是只用一个值就能体现离散程度。标准差经常用于评估金融风险，帮助确定药物在医学研究中的意义，以及测量预测类计算的结果误差，例如预测明天的降雨量或通勤时间。

## \*/ 13.形状

---

Shape: How to use histograms to determine shape associated with data (就是从图形形状看出数据的分布情况)

- 直方图的形状
  - Right Skewed (右偏态)
  - Left Skewed (左偏态)
  - Symmetric (对称分布，一种是钟形曲线)

## \*/ 20.21.形状与异常值

---

Outliers: Data points that fall very far from the rest of the values in our dataset (就是明显超出数据集范围的数据，比如年龄统计中有个人322岁，肯定是有问题。注意课程中的例子，扎克伯格的年薪是真实值，但是个异常值，因为他挣得太多了，平均起来没有任何意义)



这一节课程中有个超详细的outlier的识别文章：<http://d-scholarship.pitt.edu/7948/1/Seo.pdf> 简单的说，就是如果发现了Outliers，需要注意以下5点：

1. 注意到它们的存在以及对概括性度量的影响。
2. 如果有拼写错误 —— 删除或改正。
3. 了解它们为什么会存在，以及对我们要回答的关于异常值的问题的影响。
4. 当有异常值时，报告五数概括法的值通常能比均值和标准差等度量更好地体现异常值的存在。
5. 报告时要小心。知道如何提出正确的问题。

## \*/ 27. 描述统计与推论统计

统计可以分为两类：

- 描述统计：是用来描述收集的数据（就是对现有的数据分析后给出结论，各路ppt大神和excel大表哥们，常做的就是这种）。
- 推论统计：在于使用我们收集的数据对更大的总体数据得出结论（是对未来做预测）。

统计学中的4个基本术语：

- 总体 Polulation：我们想要研究的整个群体。
- 样本 Sample：总体的子集
- 统计量 Statistic：描述样本的数值摘要
- 参数 Parameter：描述总体的数值摘要

**POPULATION** = 100,000 students  
**SAMPLE** = 5000 students  
**STATISTIC** = 73%  
**PARAMETER** = Proportion of all 100,000 students  
that drink coffee

大家可以看上面这个图Uda调查有多少学员喝咖啡的例子。这个例子中前Sample和Statistic是描述统计的，这个例子的描述统计：从5000反馈中得出，73%的学生喝咖啡。而推论统计是使用这5000个回复的数据，得出所有Uda学员喝咖啡习惯的结论。就是通过总体、参数、统计量，得出对参数的推论。因为在现实世界，很少能得到总体（就是每一个）的数据，所以要想知道参数（总体的实际情况），就要使用样本和统计量进行推论了，所以叫推论统计。

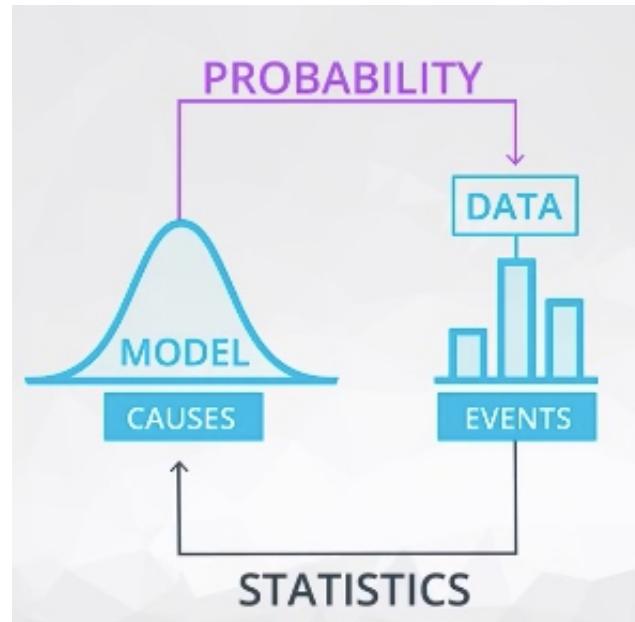
在接下来的课程3:录取案例分析，中对这两节的内容有个实例。在1到11节介绍的是录取案例背景和比率计算，非常友好。在第12节是在workspace中跑一下代码，得出比例的结论（代码的部分之前都接触过很多了，大家看下，run一下就可以得出结论了，看到自己水平的有木有，原来前3个项目没白受虐待啊！）

这课的一个要点是知道统计学的危险，因为当分组不同时（无意的或者为了达到某个结论），分析结果可能完全不同！举了一个辛普森悖论的例子，除了课程中的内容，可以参考这个资料（挺有意思的，有空可以看下，其实根源在基数不同）：<https://baike.baidu.com/item/%E8%BE%9B%E6%99%AE%E6%A3%AE%E6%82%96%E8%AE%BA>

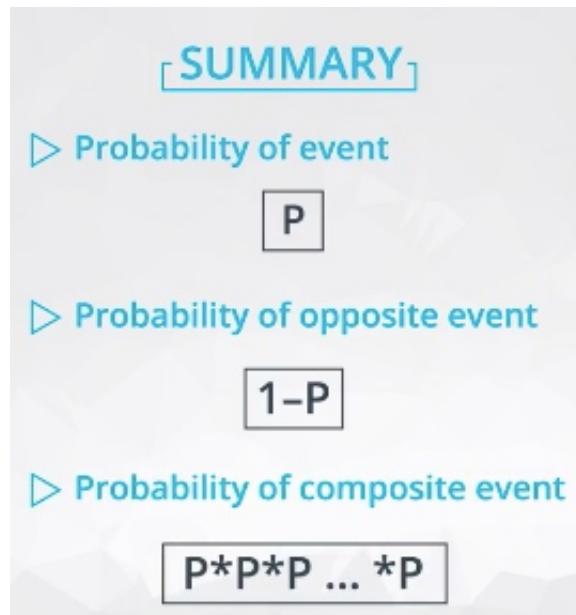
## /目标3： / 概率与二项式

### \*\*\*/ 概率（课程4）

概率入门，要认真看的呦。概率和统计学的关系简单讲就是：统计学是根据数据建立模型，而概率是根据模型去预测数据：

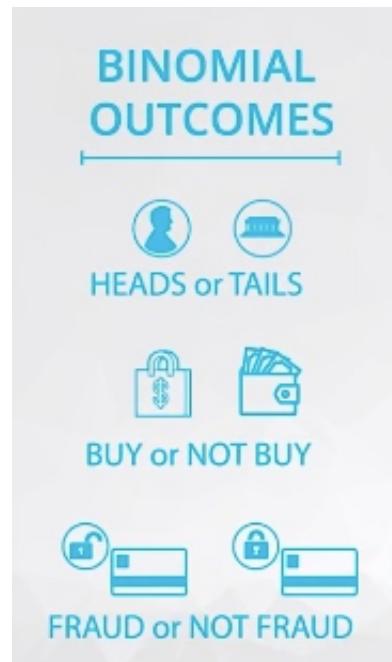


所以呢，为了能够做好统计工作，我们要先了解下概率，本课的小节看着也不少但都很短，普及了概率基础的3个内容（总结的文字内容见 <https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part/4e7e2f82-e05e-4fbe-b29c-fe3169c6dd77/module/0596b9e8-4a3a-41c3-a929-6c72c0c93925/lesson/f8e3fd8e-6759-4dda-aed2-1f094860e2c9/concept/b1455b30-e76a-47e5-a633-b67d04ede148>）



## \*\*/ 二项式（课程5）

明白了概率的基本概念后，我们来看二项式（二项分布）。课程中是从头到尾贯穿了扔硬币的例子，因为硬币有正反两面，所以投掷n次可能会出现很多种组合，这种组合的概率就是二项分布：

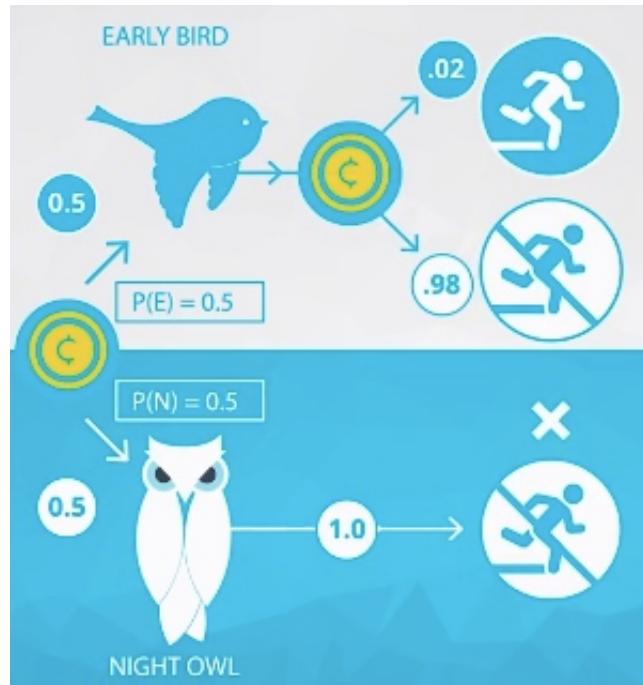


课程中后来引入了公式，如果跟随课程的话可以看懂，有那么一点点的吓人（公式的一个用处不就是吓人一跳么，所以，要学好去吓别人啊）。总结内容在：<https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part/4e7e2f82-e05e-4fbe-b29c-fe3169c6dd77/modules/0596b9e8-4a3a-41c3-a929-6c72c0c93925/lessons/dbd37b8d-f8e0-4708-a919-4b5cebe5ccaf/concepts/d75ae533-4a65-4b15-a8b6-2694b9cf049e>

## /目标4： / 条件概率、贝叶斯规则与 Python概率练习

### \*\*/ 条件概率（课程6）

其实很多时候，生活中的概率是更加复杂的，比如课程中的这个早起和晨跑之间的关系。如果是夜猫子型，则早起跑步的概率是0，如果是早起型呢，就会有2%的概率去跑步。这种情况就是条件概率，我们不再观察独立事件，第一次投硬币的结果会影响第二次的结果，如下面图：



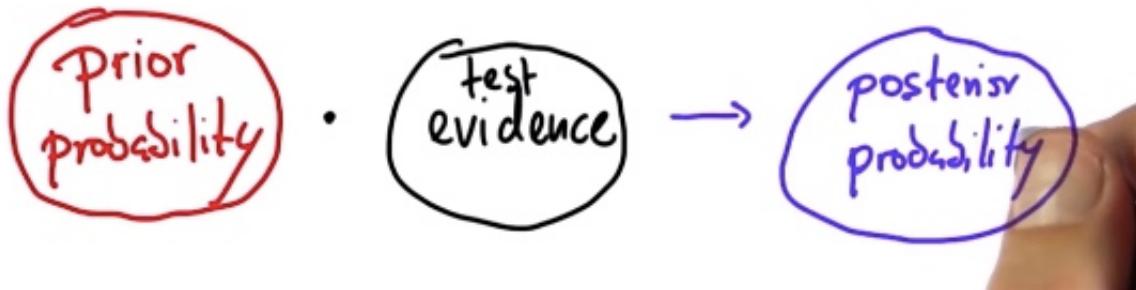
接着就是一个癌症检查的例子，如下面图，做一点解释（不要被吓到，就是多了个|，右边是条件，左边是在这个条件下的概率）：

- $P(\text{Cancer})$  是得癌症的概率， $P(>\text{Cancer})$  是没得癌症的概率，这两个概率只和为1（图最上两行）
- 但是检查结果也可能出错，检查的结果是用Positive表示阳性（就是中招了），Negative表示阴性（安全）
- 所以呢，如果有癌症，检查出来是阳性的几率很大，而检查出来是阴性的几率很小，而且相加为1（图中间两行）
- 同理，如果没有按整，检查出来是阴性的几率很大，而检查出来是阳性的几率很大，而且相加为1（图最后两行）
- 本节课明白这点就行了，后面例子还将到了全概率公式了解就可以，总结链接：[https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part.../lessons/7ddcb2e7-82be-4b2c-acc0-3e251914ac23/concepts/dac5629c-7c17-48be-8fed-1796953d1326](https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part...)

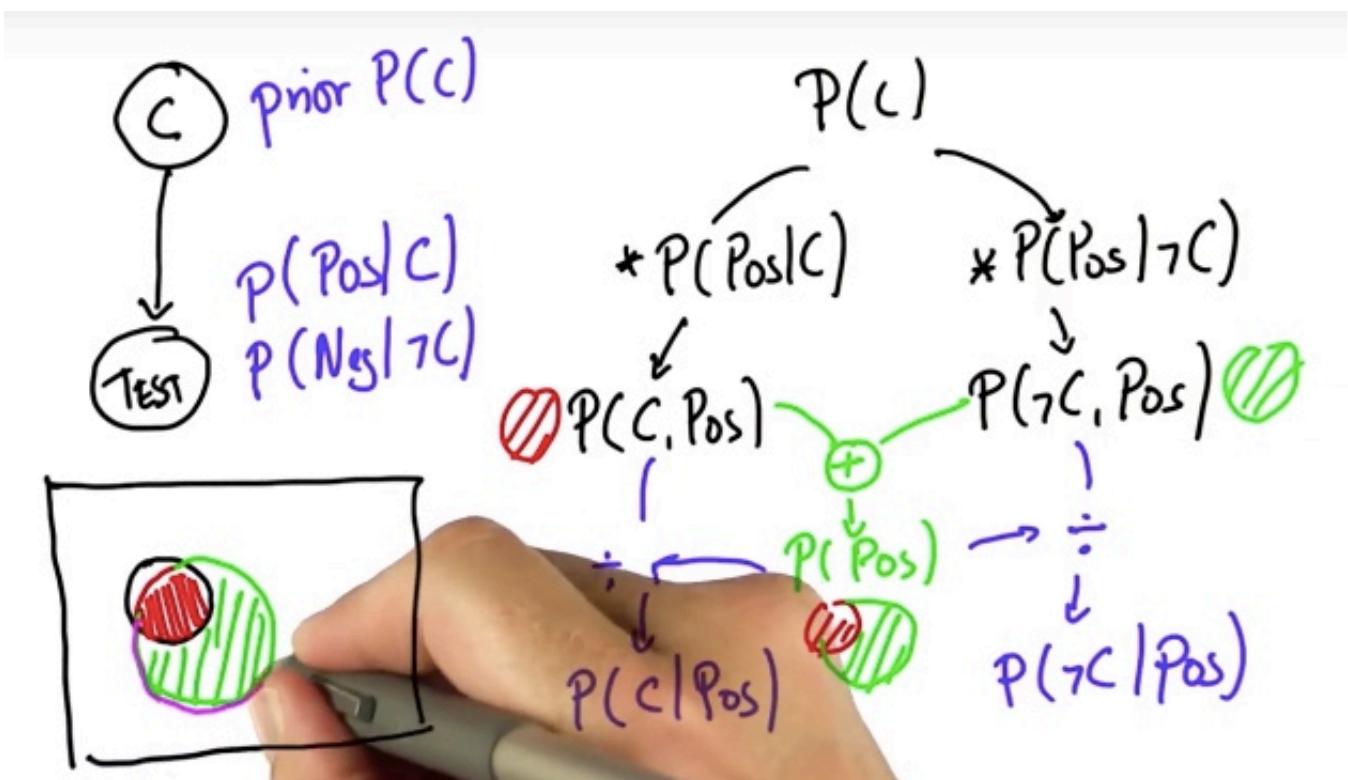
$$\begin{aligned}
 P(\text{CANCER}) &= 0.1 \\
 P(\neg\text{CANCER}) &= 0.9 \\
 \left[ \begin{array}{l} P(\text{POSITIVE} \mid \text{CANCER}) = 0.9 \\ P(\text{NEGATIVE} \mid \text{CANCER}) = 0.1 \\ P(\text{POSITIVE} \mid \neg\text{CANCER}) = 0.2 \\ P(\text{NEGATIVE} \mid \neg\text{CANCER}) = 0.8 \end{array} \right]
 \end{aligned}$$

## \*/ 贝叶斯规则 (课程7)

接下来是到了贝叶斯规则这一课，核心的内容是下面这个图（这部分选学）：



- 先验概率 \* 测试证据 = 后验概率
- $P(C, Pos)$  表示后验概率。其中的 C 是得癌症的几率，Pos 表示化验结果为阳性。
- 课程中举例癌症的例子非常贴切，有时间的可以看下来（如果对概率比较生，看3-5遍，就比较清楚了）
- 这个说明也很不错：[http://blog.sina.com.cn/s/blog\\_87e96ae20101d204.html](http://blog.sina.com.cn/s/blog_87e96ae20101d204.html)
- 如果能看懂20节的小节，就说明掌握了（可以放到通过以后再看）
- 20小节之后可以不看，是表深入的在无人车应用的例子（Uda校长出马的地方和无人车抢相关呦）



## \*\*/ Python概率练习 (课程8)

- 这节其实就是把概率的内容落地到Python代码实现上，把所有内容和实例workspace的代码

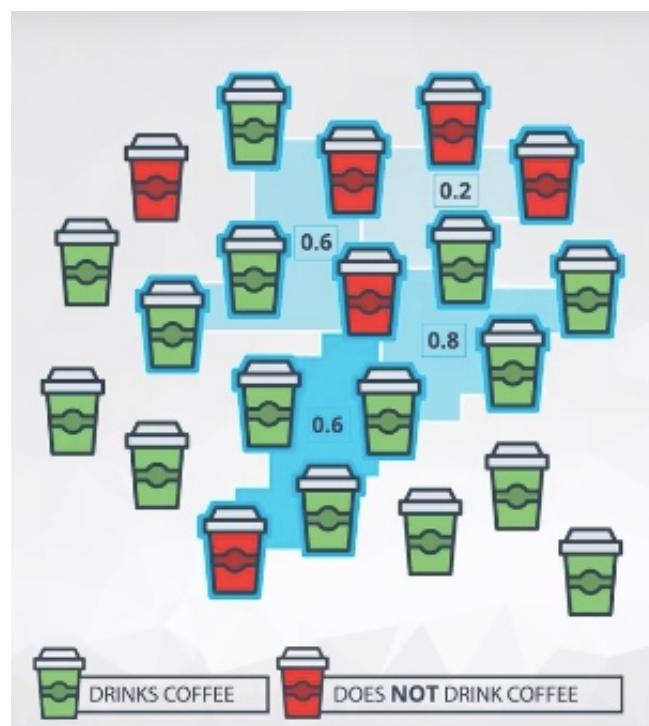
run一遍能看懂就可以了。

- 这部分在项目中有用到，要求看懂能查。
- 第7节的总结非常全面，推荐仔细看完：[https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part.../modules/0596b9e8-4a3a-41c3-a929-6c72c0c93925/lessons/33860189-96c0-40cf-8f12-5772c8b8c790/concepts/d05061c9-efe7-46bf-a135-011ee6773b6d](https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part...)

## /目标5： / 抽样分布与中心极限定理（课程10）

在开始课程11之前（下周的），先简单的介绍下 课程10:抽样分布与中心极限定理，这一章的内容是延续课程2中的内容。首先要记住推论统计学的定义： Inferential Statistics - Drawing Conclusions about a population based on data collected from a sample of individuals from that population. 根据总体中的样本，得出关于总体的结论。

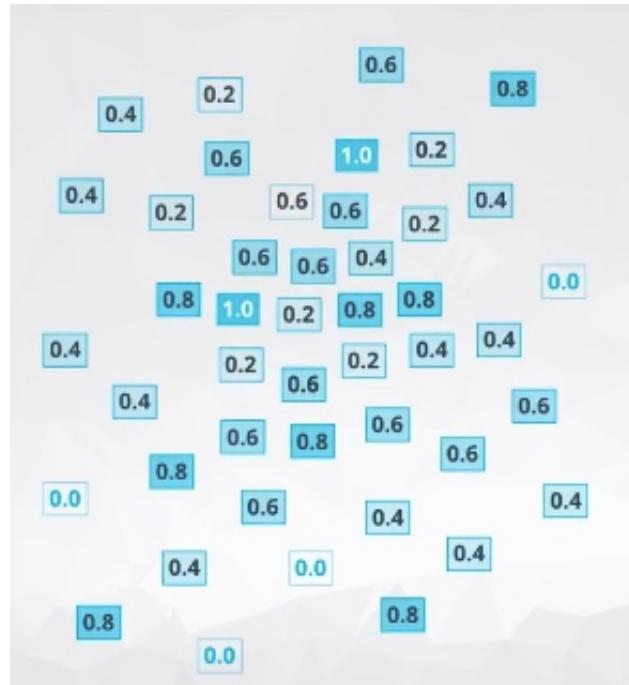
那么既然是通过样本推算出总体，那么如果我们的样本不一样，结果会不会不同呢？当然会不一样，请看课程中下面的图：



这张图特别的赞，我来解释一下：

- 大家看图中有4个深浅不一的蓝色区域，每个区域的边缘都通过了5个咖啡杯。
- 绿色的是不喝咖啡的，红色的是喝咖啡的
- 所以在这5个杯子中，就可以算出不喝咖啡的比率
- 比如有香蕉那个1个绿杯子，4个红杯子，所以不喝咖啡孩子的比率是 $1/5=0.2$

- 而左下角的就是0.6，以此类推
- 那么，如果我们随机取很多次5个样本的话，每个样本就会有一个比率，抽象成下面的图：



- 当我们把这些随机样本的比率做统计的话，会发现他们的分布是正态分布的（中间多，两边少，像个倒扣的钟一样～没见过？准你请假去看《邪不压正》里的彭于晏还不行）。
- 这个规律就是中心极限定理：样本容量足够大，平均数的抽样分布越接近正态分布。适用于一下统计量中：
  1. 样本平均数 ( $\bar{x}$ )
  2. 样本比例 ( $p$ )
  3. 样本平均数的差异 ( $\bar{x}_1 - \bar{x}_2$ )
  4. 样本比例的差异 ( $p_1 - p_2$ )
- 有图有真相：

## SAMPLING DISTRIBUTION



代码练习在第18节，有一点要说一下：

```
1 # 首先设定一个空list
2 means_size_3 = []
3 # 从总体中抽取3个，抽取10000次，把每次的mean存到上面的list里
4 for i in range(10000):
5     mean_size_3 = np.random.choice(pop_data, size = 3).mean()
6     means_size_3.append(mean_size_3)
```

接下来是抽样分布的练习，这里请大家run一遍，熟悉随机数的代码运用（项目里要用的到）：

```
1 # 导入numpy使用随机函数
2 import numpy as np
3 # 设定seed, seed的作用是设定了(n)以后，每次随机都能得到相同的抽样
4 # 这种情况在还原分析过程时非常有用
5 # 设定了之后就一直有效，直到被覆盖
6 np.random.seed(42)
7 students = np.array([1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0])
```

```
1 # 注意在jupyter 中，seed的设置要和choice在一个代码框才生效
2 # 所以再输入一遍
3 np.random.seed(42)
4 # 此处是用np.random.choice把学生进行抽样，()为参数
5 # 第1个参数是数据，就是students
6 # 第2个参数是抽样的个数
7 # 第3个参数replace默认= True，就是可以再次选中被取到的数
8 # True的情况适合用在两个骰子。从1-6取，取到1之后还可以取到1
9 # 学生这个例子要用False，因为一个学生被取走了，就不可以再取了
10 # random.choice的官方文档
```

```

11 | # https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.choice.html
12 |
13 | sample = np.random.choice(students, size = 5, replace = True)
    sample

```

其实课程中在咖啡的例子中是抽样后再放回去的（就是可以再次选中），这种方法叫自展法（bootstrap），就是取出任一个元素，再次选中它的概率不变。这种方式的意义是在样本不变的情况下能够更好的推论出参数：No more data needed to gain a better understanding of the parameter. 自展法为什么有效的说明：<https://stats.stackexchange.com/questions/26088/explaining-to-laypeople-why-bootstrapping-works>

接下来的一节对于项目4的所有统计学符号做了统计，参数是总体的指标，统计量是样本的指标，如下图，如果有疑问请参见（<https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part/4e7e2f82-e05e-4fbe-b29c-fe3169c6dd77/module/0596b9e8-4a3a-41c3-a929-6c72c0c93925/lesson/4757541f-77eb-4df6-9ebf-9485a3383dd3/concept/6802cfab-3e8f-4218-b3e7-bbc535ae27f0#>）：

| 参数              | 统计量                     | 描述                       |
|-----------------|-------------------------|--------------------------|
| $\mu$           | $\bar{x}$               | "数据集的平均值"                |
| $\pi$           | $p$                     | "仅包括数值 0 和 1 的数据集平均数：比例" |
| $\mu_1 - \mu_2$ | $\bar{x}_1 - \bar{x}_2$ | "平均数差异"                  |
| $\pi_1 - \pi_2$ | $p_1 - p_2$             | "比例差异"                   |
| $\beta$         | $b$                     | "回归系数：通常出现在下标"           |
| $\sigma$        | $s$                     | "标准差"                    |
| $\sigma^2$      | $s^2$                   | "方差"                     |
| $\rho$          | $r$                     | "相关系数"                   |

在之后，又出现一个新的定理，大数定理：Law of Large Numbers - The Larger our sample size, the closer our statistic gets to the parameter. 翻译过来有点废话的意思：随着样本容量增加，样本平均数越来越接近总体平均数。接下来是介绍了3中最常见的估算技巧（知道名字就行，想深入点链接）：

- 最大似然估计 [https://en.wikipedia.org/wiki/Maximum\\_likelihood\\_estimation](https://en.wikipedia.org/wiki/Maximum_likelihood_estimation)
- 矩估计方法 <https://onlinecourses.science.psu.edu/stat414/node/193/>

- 贝叶斯估计 [https://en.wikipedia.org/wiki/Bayes\\_estimator](https://en.wikipedia.org/wiki/Bayes_estimator)

后面的代码，先生成了3000个gamma分布的数据，分别取5, 20, 100样本，看看和参数的差距（非常明显的在缩小，符合大数定理）：

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 %matplotlib inline
5 np.random.seed(42)
6
7 # gamma是一种分布（官方有个图比较直观）
8 # 参考: https://zh.wikipedia.org/zh-hans/%E4%BC%BD%E7%8E%9B%E5%88%86%E5%B8%83
9 # 官方: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.gamma.html
10 pop_data = np.random.gamma(1, 100, 3000)
11 plt.hist(pop_data)
12
13 # 参数
14 pop_data.mean()
15 100.35978700795846
16
17 # 取5个样本
18 np.random.choice(pop_data, size = 5).mean()
19 179.26530134867892
20
21 # 取20个样本
22 np.random.choice(pop_data, size = 20).mean()
23 143.10974179021264
24
25 # 取100个样本
26 np.random.choice(pop_data, size = 100).mean()
27 117.0757073569392

```

## /彩蛋/ 统计学深入资料

各位同学如果对统计学感兴趣，想深入一些的话，强烈推荐Uda高级数据分析助教任锐大大的统计学系列：

<https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project4/Statistics-Plus.zip>

也一并附上项目4文件打包：

<https://github.com/mengfanchun2017/DAND-Basic/blob/master/Project4/p4files.zip>

Title: W10 Plus A/B测试数据清理提示

Tags: 数据分析初级, 实战项目

# W10 Plus A/B测试数据清理提示

---

- [W10 Plus A/B测试数据清理提示](#)
- [/目标6/ 项目：分析A/B测试结果](#)
  - [/ 1.准备](#)
  - [/ 2.完成项目的 I-概率 部分的撰写](#)
    - [// 问题1, 基本信息](#)
    - [// 问题2, 处理异常值](#)
    - [// 问题3, 删除重复值](#)
    - [// 问题4, 计算比率](#)
- [/彩蛋/](#)

## /目标6/ 项目：分析A/B测试结果

---

### / 1.准备

---

最后一个项目，我们试一个新的节奏，每周学习完成之后，直接把项目的相关内容写完。所以呢，和以前的项目相同，首先请大家决定是在workspace做项目还是本地做，如果本地做的话，我传送了项目的两个数据文件和jupyter notebook的打包文件，在导学的最后链接。

### / 2.完成项目的 I-概率 部分的撰写

---

根据模版，这个部分完全是项目3中的清理内容为主，大家High起来一鼓作气搞定它！一定要开工哦（助教大大坏笑脸），有几点提醒：

- 这部分对于pandas数据的筛选使用比较凶残、计算数量时可以使用.shape[0], 也可以使用count, 也可以用len
- 独立用户使用.unique()

#### // 问题1, 基本信息

代码如下，前面是小节号：

```
1 # b. 使用下面的单元格来查找数据集中的行数。
2 print(len(df))
3 #使用len可以看出共多少行
4 print(df.count())
5 #使用count可以看到每列多少 (可以观察到空数据)
6 #可以加参数见
7 #https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame
8 .count.html
9 df.shape[0]
10
11 # c. 数据集中独立用户的数量。
df.nunique()
```

到了下一问，用户转化比例就可以用转化了的用户处以总用户，用筛选加count检查不一样的次数，用isnull().sum()检查是否有缺失值：

```
1 # d. 用户转化的比例
2 df['converted'].sum() / df['user_id'].nunique()
3
4 # e. new_page 与 treatment 不一致的次数
5 df[(df['landing_page']=='new_page')&(df['converted']==1)].shape[0]
6
7 # f. 是否有任何行存在缺失值
8 df.isnull().sum()
```

## // 问题2，处理异常值

这里的重点是使用好过滤，并且能求比率：

```
1 # a. 现在，使用测试题的答案创建一个符合测试规格要求的新数据集。将新 dataframe 存储在
2 df2 中。
3 # 先看下有多少这样的值
4 # 检查1看左边是newpage右边不是treatment的
5 dismatch1 = df[(df['landing_page']=='new_page')&(df['group']!='treatment')]
6
7 dismatch1.shape[0]
8
9 >>> 1928
10
11 # 检查2看左边不是newpage但是右边是treatment的
12 dismatch2 = df[(df['landing_page']!='new_page')&(df['group']=='treatment')
```

```

13  ')]
14  dismatch2.shape[0]
15
16 >>> 1965
17
18 # 计算下比率
19 dismatchr = (dismatch1.shape[0] + mismatch2.shape[0]) / df.shape[0]
20 mismatchr
21
22 >>> 0.013220002852505111
23
24 # 发现只有1.3%的数据是这样的，但是回头想一想，应该看dismatch的是和match的比较
25 # 注意复杂过滤的写法，把每个条件放到 () 中
26 match = df[(df['landing_page']=='new_page')&(df['group']=='treatment')].
27 shape[0]
28 mismatchr2 = (dismatch1.shape[0] + mismatch2.shape[0]) / match
29 mismatchr2
30
31 >>> 0.026790814184748574
32
33 # 2.6% (上面计算出不一致的数据)，不算太多，丢弃吧
34 # 建立df2，把符合要求的数据行过滤出来
35 df2 = df[((df['landing_page']=='new_page')&(df['group']=='treatment'))|(
36 (df['landing_page']=='old_page')&(df['group']=='control'))]
37 # 右侧df[()]中括号里面是过滤条件，可以符合
38 # 比如()&()代表与，&替换成|代表或
39 # 可以嵌套，如上面代码是两边先与，结果再求或
40 df2.shape[0]
41
42 >>> 290585
43
44 # 检查下是否数量相同
45 df2.shape[0] + mismatch1.shape[0] + mismatch2.shape[0] == df.shape[0]
46
47 >>> True

# 这样检查也可以
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].shape[0]

>>> 0

```

## // 问题3，删除重复值

同样罗列在下面，一定要看懂自己写出来不要copy paste：

```
1 # b. df2 中有一个重复的 user_id 。它是什么?  
2 # .duplicated()就是输出重复行  
3 df2[df2['user_id'].duplicated()]  
4  
5 # d. 删除 一个 含有重复的 user_id 的行, 但需要确保你的 dataframe 为 df2。  
6 # 在小括弧中使用[]参数定义要判断duplicates的列  
7 # 不加的话就是每列都重复才删除  
8 df2 = df2.drop_duplicates(['user_id'])  
9 df2.shape[0]
```

## // 问题4，计算比率

```
1 # a. 不管它们收到什么页面, 单个用户的转化率是多少?  
2 # 使用转化的除总数  
3 converte_rate = df2['converted'].sum() / df2.shape[0]  
4 # converte因为是0、1区分, 所以sum就是个数  
5 # 使用round保留4位小数  
6 round(convert_rate, 4)  
7  
8 # b. 假定一个用户处于 control 组中, 他的转化率是多少?  
9 # c. 假定一个用户处于 treatment 组中, 他的转化率是多少?  
10 # 还是使用筛选解决, 大家自己试一下  
11  
12 # d. 一个用户收到新页面的概率是多少?  
13 # 也可以直接筛选出来shape转成数字计算  
14 newpage_rate = df2[df2['landing_page'] == 'new_page'].shape[0] / df2.sha  
pe[0]  
15 # 之后用round将结果保留4位小数  
16 round(newpage_rate, 4)
```

在这之后，不要忘了还有e的主观问题。这第一部分就做完了，基本上是上一个项目中加简单的比例计算，大家做好后就可以准备后两个部分的重点内容了，加油！

## /彩蛋/

也一并附上项目4文件打包：[/项目数据文件/](#)

Title: W11 置信区间和假设检验 [项目：分析A/B测试结果2/3]

Tags: 数据分析初级, 实战项目

# W11 置信区间和假设检验 [项目：分析A/B测试结果2/3]

---

- [W11 置信区间和假设检验 \[项目：分析A/B测试结果2/3\]](#)
- [/学习地图/](#)
  - / 项目路径
- [/目标1、2/ 置信区间和假设检验](#)
  - / 置信区间
    - \*\*\*// 什么是置信区间
    - \*\*\*// 置信区间（自展法代码）
    - \*\*// 均数差
    - \*// 统计显著性与实际显著性
    - \*// 置信区间（传统代码）
    - \*// 置信区间宽度和误差范围
  - / 假设检验
    - \*\*// 什么是假设检验
    - \*\*\*// 怎样设置假设检验
    - \*\*\*// 检验的错误类型
    - \*// 如何选择假设检验
    - \*// 置信区间vs假设检验
- [/目标3/ 案例研究：A/B测试](#)
  - \*\*\*/ 什么是A/B测试
  - \*/ Udacity的案例
    - \*\*// Udacity的漏斗模型
    - \*\*// 点击率CTR（代码）
    - \*\*\*// 注册率（代码）
    - \*// 其他单项指标（代码）
    - \*\*// 分析多个指标
    - \*\*\*// 得出结论
- [/彩蛋/](#)
  - \*/ 希腊数字的读法
  - \*/ T检验

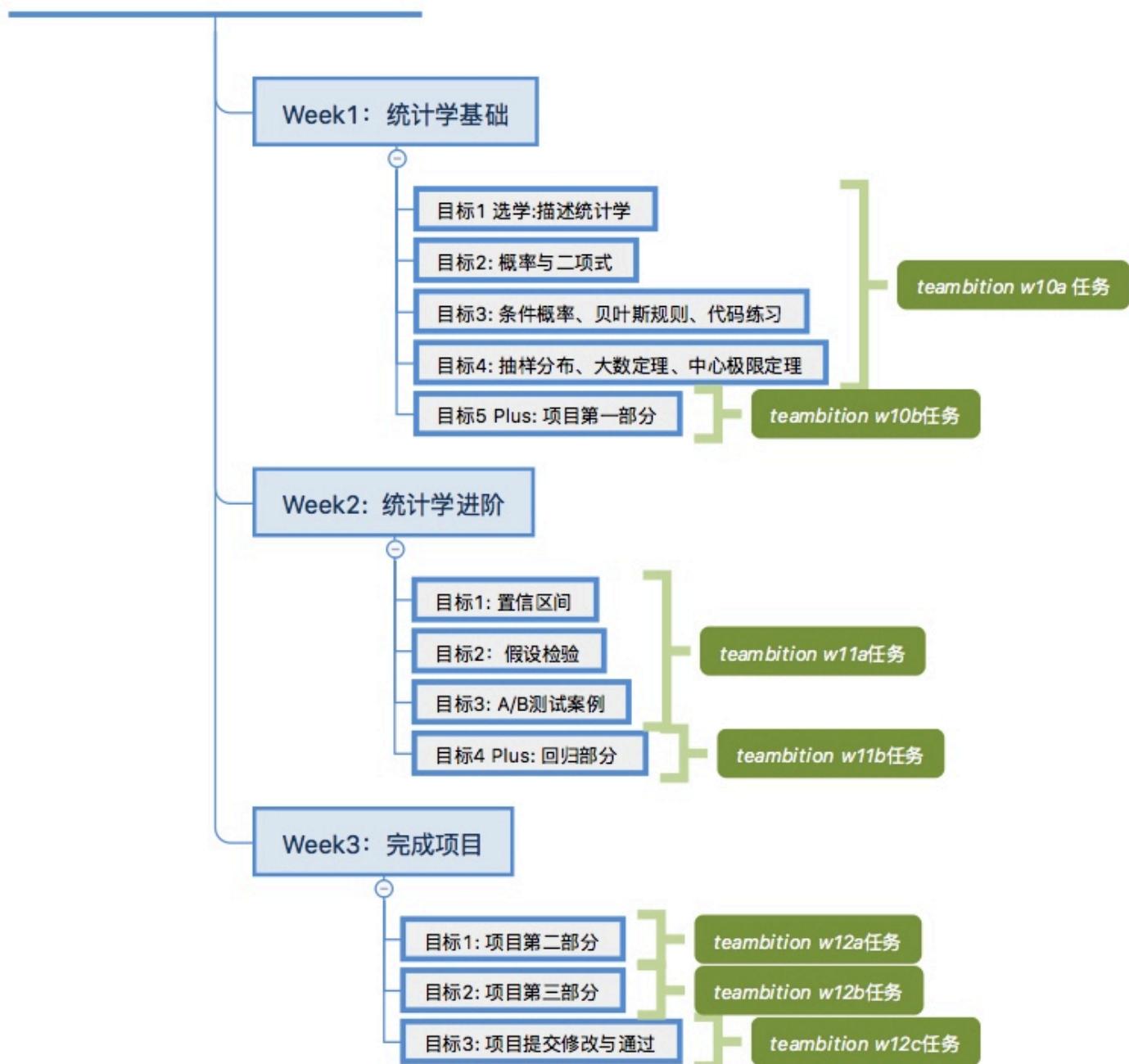
# /学习地图/

本周是统计学项目的Part2，是项目4的推论统计学的扩展部分。目标1、2、3讲解置信区间、假设检验、和一个A/B测试案例。目标4作为Plus发布，讲解了回归的相关知识。

！注意，1星和2星的可以只看本导学，先大致理解就可以。3星的是本周重点！

## / 项目路径

### 分析 A/B 测试结果



# /目标1、2/ 置信区间和假设检验

课程11:置信区间

课程12:假设检验

这课程有2个重点，第一个是明白什么是置信区间(Confidence Intervals)。第二个重点是进行假设检验。

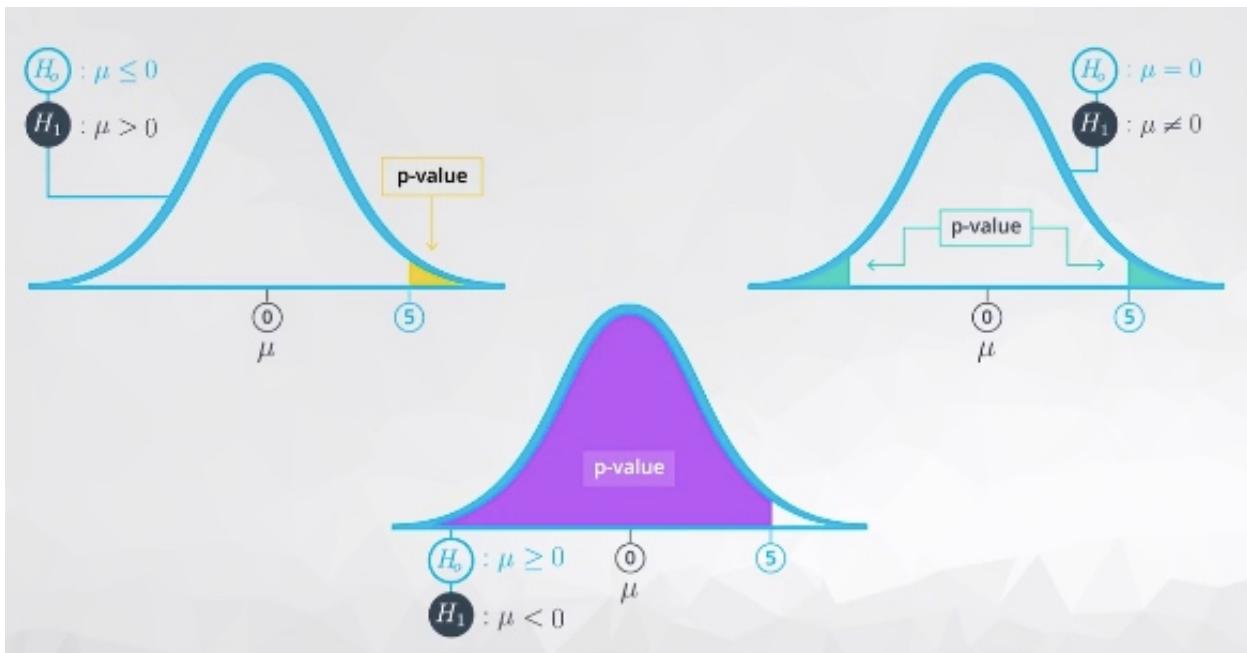
## / 置信区间

课程11:置信区间

### \*\*\*// 什么是置信区间

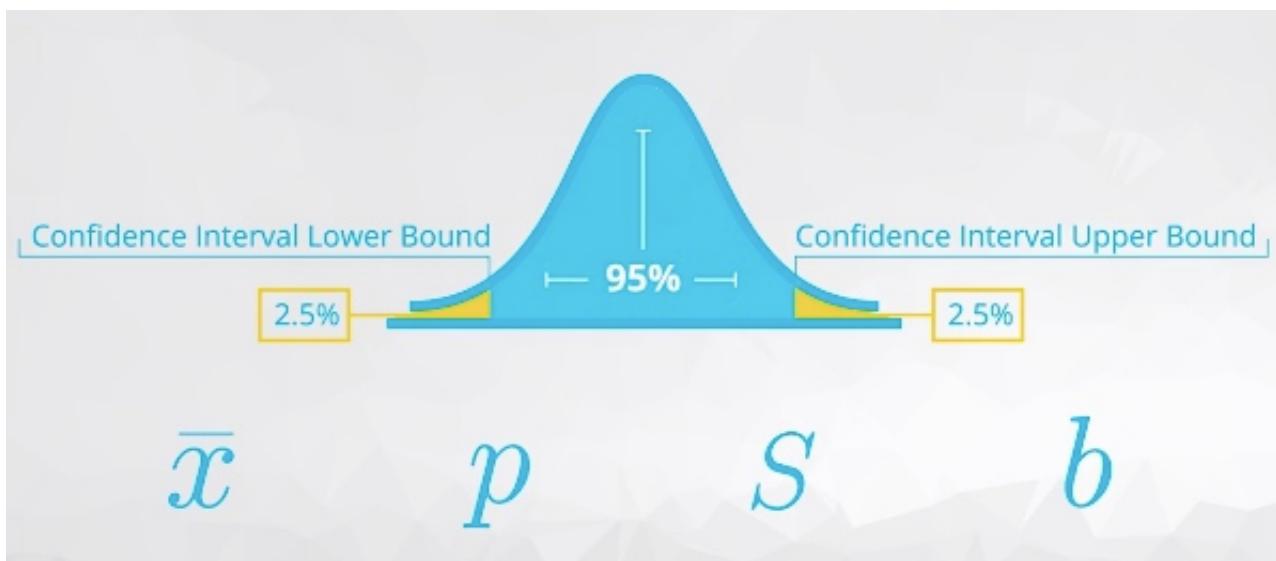
置信区间这是个什么鬼？大家还记得之前项目中有计算出两个变量的关系么，那么在什么情况下我们能确定两个变量是有关系的，什么情况下是没关系的呢（~~当然是用眼睛了~~）。到了这里，我们就可以用置信区间来回答了。简单的说，在区间内就说明有关系，不再就没关系。在操作时有两种方法，课程开始就提示了：Build confidence intervals using sampling distributions and bootstrapping.

- 怎么回答假设检验：既然假设检验要解决的问题是从样本的分布推算出总体参数(How do we use sampling distributions to infer where a parameter is located?)。那么我们怎么回答假设检验的问题呢？通过置信区间来回答。
- 置信区间是什么：是一个根据你假设检验得出下限与上限之间的区域，表示是一个范围（见图），这个范围由你手中的样本与要求的置信水平相关（就是1减去犯1类错误的比率，比如要求置信水平95%就是说这个估计的区间可能犯错的比率是5%，同理99%对应的是1%）
- 置信区间的的意义是什么：如果我们重复取样，每次取样后都用这个方法构造置信区间，有95% 的置信区间会包含真值（就是总体的参数）。
- P值是什么：明白了置信区间之后，我们能不能更明确一下对假设检验判断的描述呢？可以使用P值。P值的定义是：如果零假设为真，观察到统计量(或支持备择假设的更多极端)的概率（显著性水平 $\alpha < 0.05$ ，就等于置信水平为95%，那么 $P<0.05$ 就可以否定零假设了），但是这种情况也要根据假设的设定，如下图区分：



这里理解概念就好了，后面案例中随着代码实现（比较简单的）操作一遍就记住了。p值的好处是直接计算出一个值，可以比较准确的描述。

我们来看课程中Uda学员喝咖啡的例子。统计经常是得不到所有样本的反馈的（Uda的1000000名学生是不是喝咖啡，你真的都要每个人都问下么？），但是这种用样本推算总体的方式也是有限制的（对，概率就是这么严谨）。因为是从部分推送总体，所以我们的确定性是条件的：根据5000个调研数据我得出了一个区间，我们有95%的信心喝咖啡同学的平均身高会落在68.06到68.97英寸之间。



### \*\*\*// 置信区间（自展法代码）

4. Notebook + 练习：构建置信区间

我们的总体是2974个，假设我们做了200个调研，来看看怎么从样本的统计量（200个样本的比例）估算总体参数（2974个总体的比例，假装我们有上帝视角），首先是先抽样出200个，注意seed设置为42保证了每次抽取能复现（见上周导学的描述）：

```
1 # 设定随机种子（以后可复现抽样）
2 np.random.seed(42)
3 # 读入文件
4 coffee_full = pd.read_csv('coffee_dataset.csv')
5 # 使用.sample抽取200个样本
6 # 在实际案例中，这个就应该是你等得到的数据了
7 coffee_red = coffee_full.sample(200)
```

那么我们看看总体和抽样的喝咖啡比例，差不多的：

```
1 coffee_red['drinks_coffee'].mean()
2 >>>
3 0.5949999999999997
4
5 coffee_full['drinks_coffee'].mean()
6 >>>
7 0.58977807666442505
```

光这样我们是算不出置信区间的，这200个样本我们可是要好好利用一下，bootstrap这个方法的牛x之处就体现出来了：虽然我们只有200个样本，但是我们可以用这200个样本模拟出来100000次抽取来。是这样实现的：我们先从200个样本中拿出1个来，记录好；再把它放回去（就是说还可能抽到）；再拿下一个，记录好再放回去。这样的话当我们抽取200次之后，得到的新的记录和这200个就不一样了，而且这个过程可以进行很多遍，每次的结果都不一样。代码是这样的：

```
1 # 设定每次抽出的内容为bootstrap_sample
2 # .sample默认的是replace = True
3 # 就是每次抽取之后，再放回去，就是实现自展（bootstrap）
4 # 这样的话每次运行都会是不一样的集合
5 # .sample不受seed影响
6 bootstrap_sample = coffee_red.sample(200, replace = True)
7 # 检查下，每次都应不一样
8 bootstrap_sample.head()
```

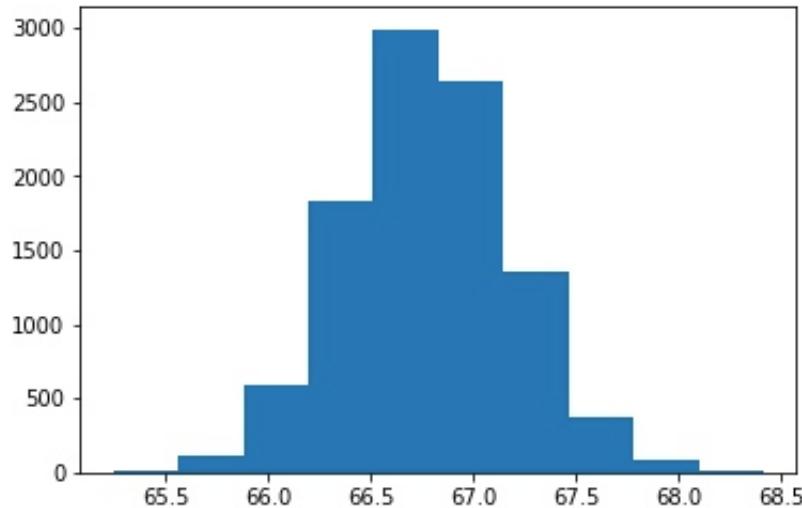
那么接下来，我们要做的就是这种自展抽样来10000遍，看看这10000遍均值的统计规律：

```
1 # 先建立个空列表存储结果
2 boot_mean = []
3 # 使用for in range 循环10000遍
4 for i in range(10000):
5     # 生成sample，自展为True
6     sample = coffee_red.sample(200, replace = True)
7     # 生成mean_s 赋值为不喝咖啡学生的平均身高
```

```

8     mean_s = sample[sample['drinks_coffee'] == False]['height'].mean()
9     # 将这个值附加到列表中
10    boot_mean.append(mean_s)
11
12    # 画个图看一看
13    plt.hist(boot_mean);

```



还记得上面那个95%置信区间的图么，对于这个输出我们将范围截断。左边舍弃2.5%，右边舍弃2.5%。这样剩下中间的95%就是：我们根据200个样本推算出总体（5000个学生）中不喝咖啡同学的平均身高的95%置信区间。这个95%称为置信度。

```

1 # 使用np.percentile计算两边截断点值，中间区间就是95%的截断区间
2 np.percentile(boot_mean, 2.5), np.percentile(boot_mean, 97.5)
3
4 >>> (65.992913281575198, 67.584027382815719)

```

现在我们可以开上帝视角了，把总体的算出来，真的是在这个区间中！帅！：

```

1 # 直接使用coffee_full计算总体的参数
2 coffee_full[coffee_full['drinks_coffee'] == False]['height'].mean()
3
4 >>> 66.443407762147004

```

## \*\*// 均数差

6.Notebook+练习：均数差

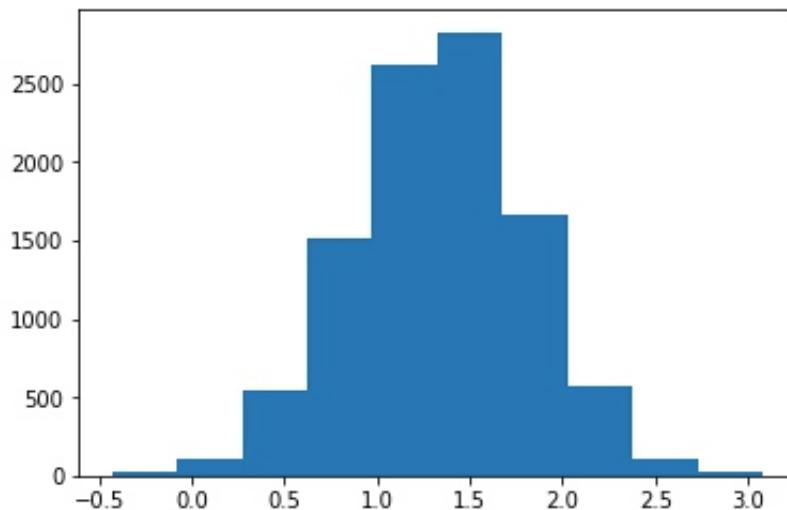
上面的置信区间我们是用不喝咖啡同学的身高距离，介绍了什么事置信区间，那么问题来了，我开始好奇喝咖啡和不喝咖啡的同学身高有没有差别呢？通过比较不喝咖啡平均身高、喝咖啡平均身高，我们就扩展到了两个参数比较了，这里使用问题和结论进行描述：

- Question: What is the difference in the mean height for coffee vs. non-coffee drinkers?
- Conclusion: Since a confidence interval for  $\text{mean\_coff} - \text{mean\_nocoff}$  is (0.59,2.37), we have 95% evidence of the mean height for coffee drinkers is larger than non-coffee drinkers.
- 注意结论中的 (0.59, 2.37) 是置信区间，95%是可信度，是支持结论的数据。代码是在上面的基础上扩展一点，将比较的数值换成了喝咖啡同学身高与不喝咖啡同学身高的差值：

```

1 # 定义新的列表
2 means_f = []
3 for i in range(10000):
4     sample = sample_data.sample(200, replace = True)
5     mean_coffee = sample[sample['drinks_coffee'] == True]['height'].mean()
6     mean_no_coffee = sample[sample['drinks_coffee'] == False]['height'].mean()
7     # 将两类人的身高求差, 存到列表中
8     means_f.append(mean_coffee - mean_no_coffee)
9
10 # 画图看看
11 plt.hist(means_f);

```



！发现问题！居然这个图不是在0周围，肉眼观察差异平均值在1.0到1.5之间，为什么呢？可能是因为喝咖啡的同学营养比较好吧（毕竟得吃饱了饭才喝咖啡）？（开玩笑啦），寻找原因这么主观的事，不是用统计学来玩的。

那么我们看下置信区间（注意数据和课程中有一点不同）：

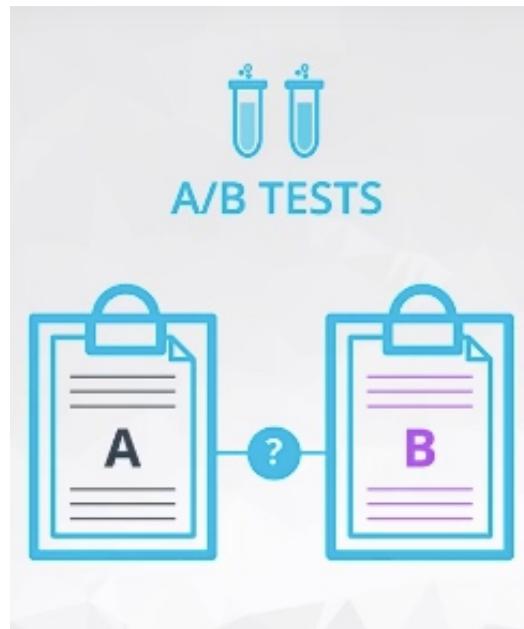
```

1 np.percentile(means_f, 2.5),np.percentile(means_f, 97.5)
2
3 >>> (0.39656867909086274, 2.2432588681124224)

```

那么这种描述均值差的置信区间是在什么地方使用呢？对！就是项目4中的A/B Tests，简单的讲

就是把两种解决方案分为A测试组与B测试组，收集来数据看下两组有没有差异，来决定使用那种方案更好。课程中的解释：A/B 测试对全球企业是最重要的内容之一。通过这个技巧，你可以改变网页布局的一些内容，了解它如何影响用户。理想情况下，你需要提供一个页面，可以实现更多点击量、更高收益和/或更高的用户满意度。



## \*// 统计显著性与实际显著性

要注意，我们前面的所有结论是统计显著性的，和实际显著性是有区别的，先来看下定义：

- Statistical Significance: Evidence from hypothesis tests and confidence intervals that H<sub>1</sub> is True. 使用置信区间和假设检验，你能够在做决策时提供 统计显著性。这里的H<sub>1</sub>将在假设检验中解释。
- Practical Significance: Considers real world aspects, not just numbers in making final conclusions. 实际显著性 考虑到所处情况的其他因素，假设检验或置信空间的结果可能不会直接考虑到这种情况。空间、时间 或 金钱 等约束条件对商业决定很重要。但是可能不会在统计测试中直接考虑这些因素。

简单的说就是，虽然统计知道那个更好，但是这种方案可能更贵，在做商业决策时要整体考虑（钱、资源等限制因素并不在统计中体现），看课程中这个例子：



虽然右边的更好但是制作起来更贵更费时，所以呢如果是我的朋友刚刚开始这个业务，我会建议他做左边的（因为也挺有效的）。

需要注意，自展法在样本很小的时候能够很好的估计总体的参数；当样本很多的时候，使用传统的方法一样能够得到有效的估计，但是传统的公式更加复杂。在本课程中使用自展法进行计算。如果有兴趣想要研究传统假设检验的方法，可以参考这个链接：<https://stattrek.com/hypothesis-test/hypothesis-testing.aspx>

## \*// 置信区间（传统代码）

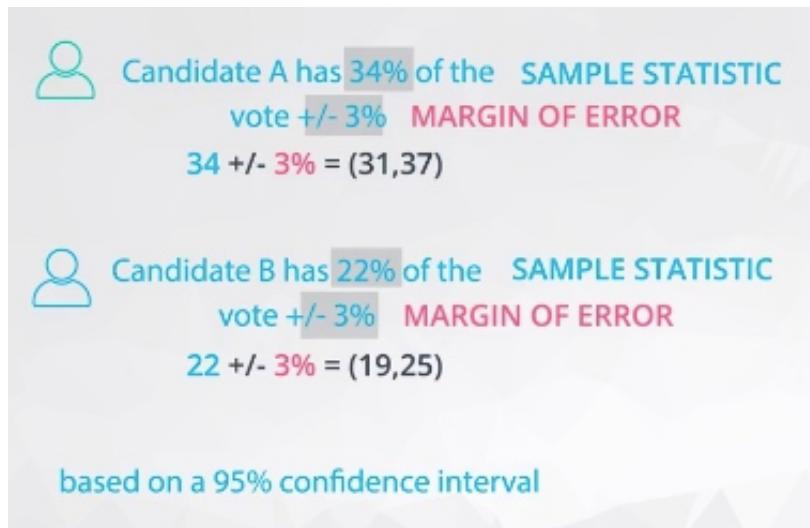
传统的检验方法有很多：

- T-Test 比较总体均值
- Two Sample T-Test 用于比较两个均值
- Paired T-Test 用于将个体与自己比较的T检验
- Z-Test
- Chi-Squared Test
- F-Test

这些都可以被自展法替代（神器！），而且自展法需要的样本更小，这部分的视频有个对比，大家看下就好了，本课程不需研究传统的检验方式。ps：这部分的ipynb文件我已经完成了，需要研究的找我要，就不放在这里瞎胡人了。

## \*// 置信区间宽度和误差范围

先上个图看下误差范围（Margin of Error）指的是什么？（其实就是你买小零食上那个100g +/- 2g的标识：



在上图中，我们得出的区间高值减去区间低值就是置信区间的宽度。比如Candidate A 是针对美国总统小特做的民意调查，我们可以这样描述：根据收集来的数据，我们确认，他的支持率在31%到37%的可能性有95%。置信区间宽度为6%。什么？区间太大，，，老板再给批点钱，增加收集的数据可以减少这个差距。注意MOE也是有计算公式的，这里不做介绍。

需要注意：置信区间采用综合方法，基于数据得出结论，因为这些测试旨在理解总体的参数(即总体数值的集合)。

## / 假设检验

课程12：假设检验

### \*\*// 什么是假设检验

2.假设检验

重点2是假设检验（也叫显著性检验），那么我们怎么科学的，不产生歧义的把两个参数是否有关系说清楚呢，全面、准确、又高big呢？当然有了，就是假设检验！（看名字就肃然起敬了有没有）。其实假设检验和置信区间是相关的（看看上面那个小特的例子）：就是让我们能够通过样本推算出总体的情况：

**Hypothesis Testing and Confidence Intervals** allow us to use only sample data to draw conclusions about an **entire population**.

比如像这个例子，我认为巧克力冰激凌是最受欢迎的口味 ( $H_0$ )，而我的朋友认为香草冰激凌是最受欢迎的口味 ( $H_1$ )，就可以这样描述：



但是并不是所有的假设检验都这么简单，这也就看出假设检验的厉害之处了，可以就复杂的情况做描述，比如一款新药是能让病人更舒服、活得更长、还是减少肿瘤的大小：



假设检验的目的就是帮助企业作出更好、更明确的决策。

### \*\*\*// 怎样设置假设检验

- 3.怎样设置假设检验-第一部分
- 4.怎样设置假设检验-第二部分

- 把问题转换成对立的假设  $H_0$   $H_1$
- $H_0$ 零假设 (null hypothesis)：是在我们收集任何数据之前认为为真的条件（比如认为两组相等，或者效应为零）
- $H_1$ 对立假设 (alternative hypothesis)：与零假设对立的，不重叠的假设（通常与你想要证明为真的事实相关）
- 假设检验使用数学表达
  - 使用p来表达概率，比如我统计了新旧两个页面的转化率，可以用  $p_{new}$   $p_{old}$  分别代表

两个页面的转化比率

- 两个假设的描述的概率要全，即  $H_0 H_1$  包括了判断的所有可能性
- 数学表达式为 =  $\neq$   $\geq$   $\leq$  (使用 latex 比较美观)
- 表达的方式1:

$$H_0 : p_{new} - p_{old} = 0$$
$$H_1 : p_{new} - p_{old} \neq 0$$

- 表达的方式2:

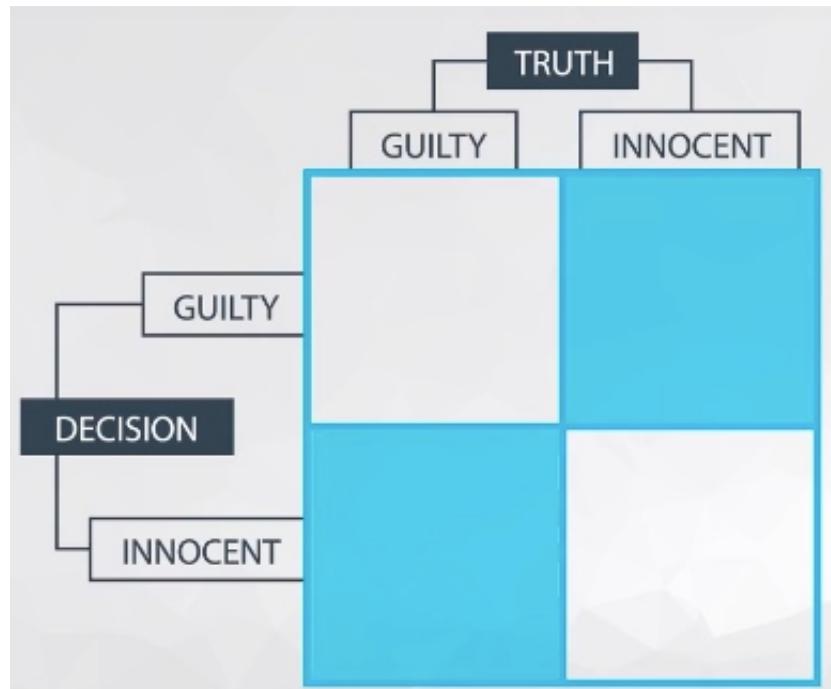
$$H_0 : p_{new} = p_{old}$$
$$H_1 : p_{new} \neq p_{old}$$

## \*\*// 检验的错误类型

6. 错误类型-第一部分

-  
11. 练习：错误类型第三部分

课程中居了陪审团的例子，在下图中蓝色的两个部分是事实与审判不符合的区域，对应的和我们假设检验中犯错的两种类型：一类错误和二类错误。Type I Errors, Type II Errors



- I 类错误（误报）：

- 由  $\alpha$  符号表示。
- 定义是：( $H_0$ ) 为真时，认为备择假设 ( $H_1$ ) 为真。
- I 类错误通常称为 误报。

- 应该设置零假设和备择假设，I类错误是更严重的错误。

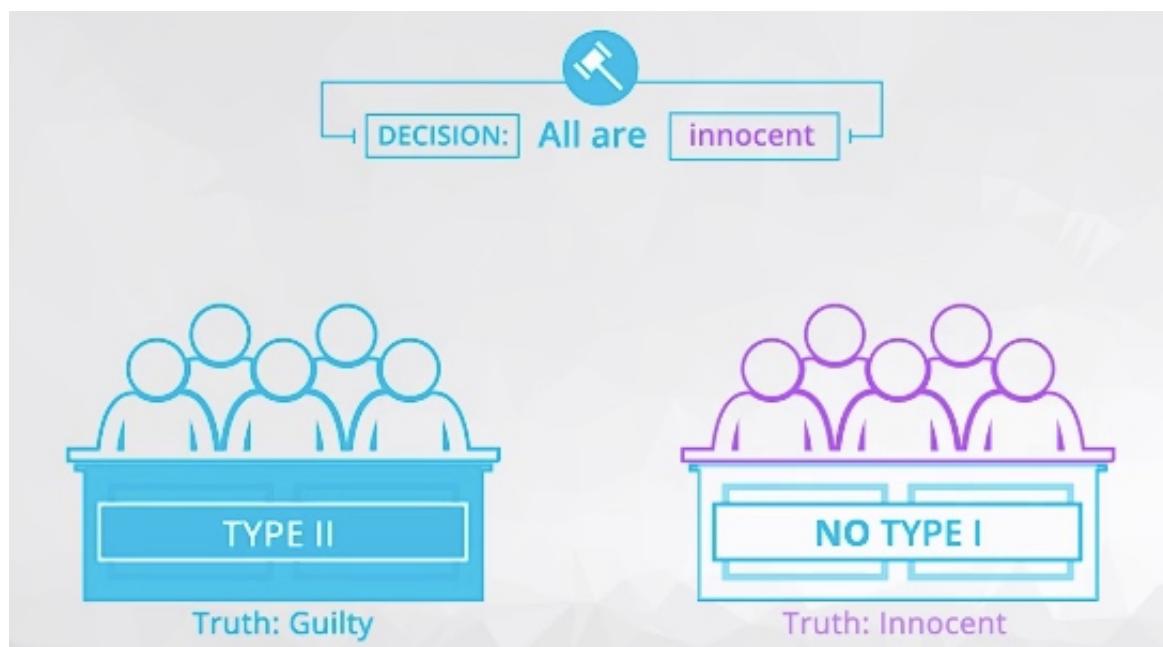
- II类错误（漏报）：

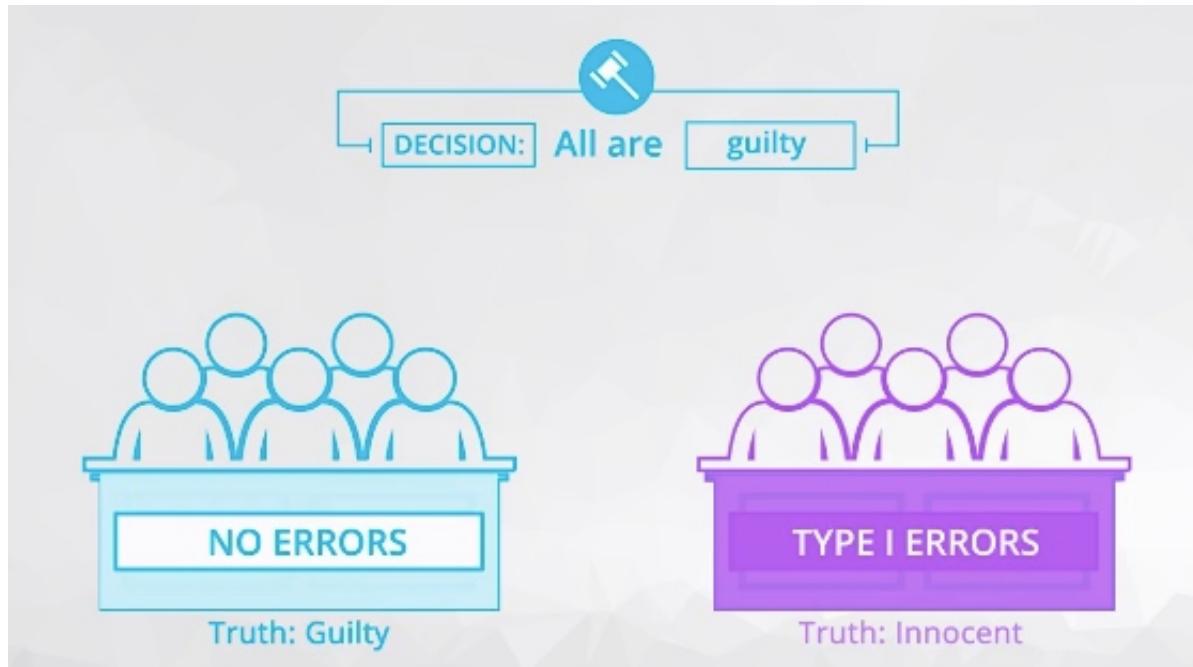
- 由 $\beta$ 符号表示。
- 定义是： $(H_1)$ 为真时，认为零假设 $(H_0)$ 为真。
- II类错误通常称为漏报。

我们继续看上图中法院的例子，既然我们知道I类错误是更严重的错误，那么我们该怎么设置 $(H_1)$ 与 $(H_0)$ 呢？

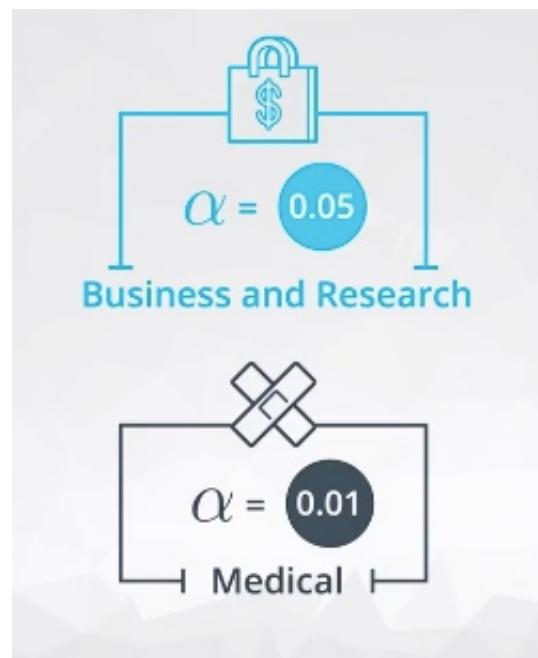
- 首先，我们认为让一个无辜的人被判有罪是更严重的错误：
  - I Type Errors: Claiming an innocent individual as guilty
  - Choosing  $(H_1)$  when  $(H_0)$  is true
  - False Positive (假阳性)
- 那么比较起来，让一个有罪的人被释放就是II类错误：
  - II Type Errors: Setting a guilty individual free
  - Choosing  $(H_0)$  when  $(H_1)$  is true
  - False Negative (假阴性)
- 所以说我们要把 $(H_0)$ 设置为无罪，就是默认所有人都是无罪的。这样假阳性（无罪但被判有罪）就符合I类错误是最严重的错误。而相比之下假阴性（有罪但被判无罪）就是II类错误了。（人之初，性本善，这3字经解释的真到位）

这也就引出了另一个问题，类型I与类型II是否有关系，加入陪审团判定所有人都无罪，那么他们只会犯很多II类错误（因为不会判有罪，就不会有冤枉），而没有I类错误，反之亦然，对比如下2图：

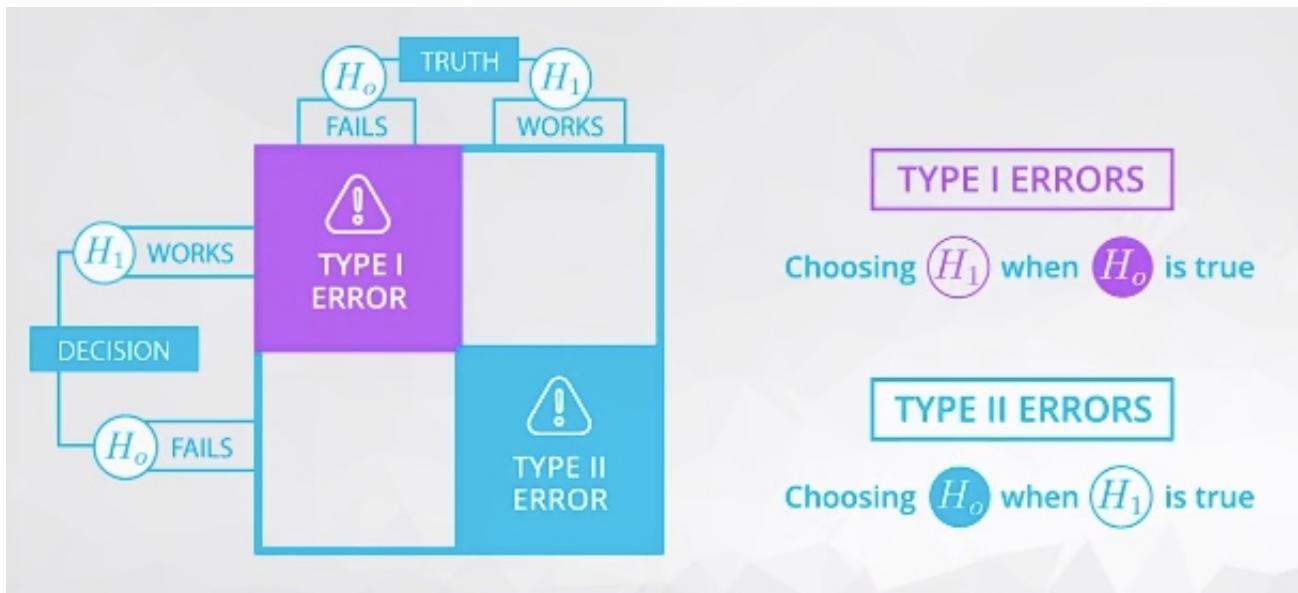




两种方式都不理想（荒谬），所以在处理这两种错误的时候的方式应该为：**为愿意犯的I类错误设置一个阀值，再满足这个阀值的同时尽量减少II类错误。**这个比例取决于具体的情况，比如在商业中和医疗的例子（医疗还是严格点好）：



接下来，我们看个特别严格，但是有实际意义的例子：检查跳伞。如果出现了I类错误，就是人带着坏伞跳下去了，，，，，（我以后还是不尝试这个运动的比较好）：



## \*// 如何选择假设检验

在扩展假设检验的内容之前, 请记住: Hypothesis tests and confidence intervals tell us about parameters, not statistics. 就是说, 我们检验来检验去, 是对参数 (对应总体) 进行判断, 而不是对统计量 (对应样本) 进行判断。因为一个样本抽取出来, 统计量就是定的了, 我们想发现的是总体的规律。这也就是推论统计的重点了。

那么我们接下来还看下咖啡的例子, 课程中的练习可以在空间中通过open看到答案。这里举了两个例子:

- 如果你感兴趣的是喝咖啡的人与不喝咖啡的人身高是否相同 (觉得会不同), 那么检验是这样的:
  - $H_0 : \mu_{coff} - \mu_{no} = 0$
  - $H_1 : \mu_{coff} - \mu_{no} \neq 0$
- 如果你觉得喝咖啡的人要比不喝咖啡的人矮, 那么检验是这样的 (注意有个方向问题) :
  - $H_0 : \mu_{coff} - \mu_{no} \geq 0$
  - $H_1 : \mu_{coff} - \mu_{no} < 0$

本部分的代码还引入了一个np.random.normal的用法:

```

1 | np.random.normal(0, np.std(diffs), 10000)
2 | # 这个方法是对后面的数据进行复合正态分布的抽取
3 | # np.std(diffs)是数据
4 | # 10000是抽取次数
5 | # 0是中间值
6 | # 这个代码的结果就也是正态分布的, 课程中有图
7 | # 看明白用法即可

```

## \*// 置信区间vs假设检验

25.其他要考虑的事情：如何对比置信区间和假设检验？

在本课程中对于置信区间喝假设检验，还有一些更新的观点：

- 假设检验喝置信区间是可以互换的（本来置信区间就可以作为假设建议的方法）
- 假设检验经常引起误解（对比起来假设检验描述比较复杂），而且还有I类错误的问题
- 倾向于使用以下3个方法进行判断
  - 置信区间（所以说是重点）
  - 效应值（以后扩展）
  - 机器学习技巧（以后扩展）

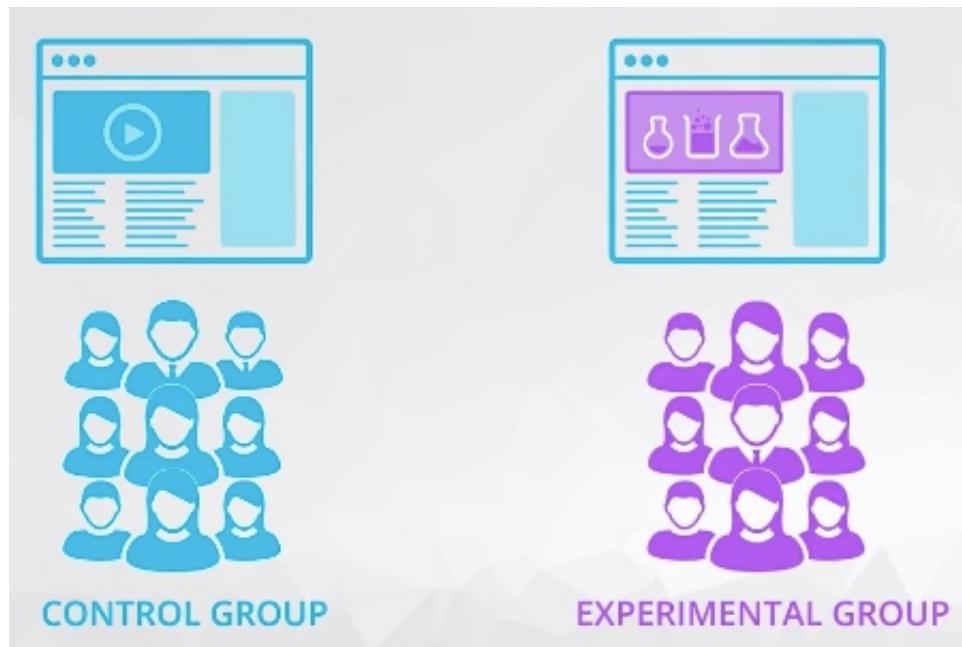


## /目标3/ 案例研究：A/B测试

课程13:案例研究：A/B测试

### \*\*\*/ 什么是A/B测试

既然叫A/B测试，肯定是要对比两个不同的东西了。比如说公司要测试应用的新特性或者某个网页页面的新版本时，会把用户分为两组。不变化的组叫对照组，变化的组叫实验组：

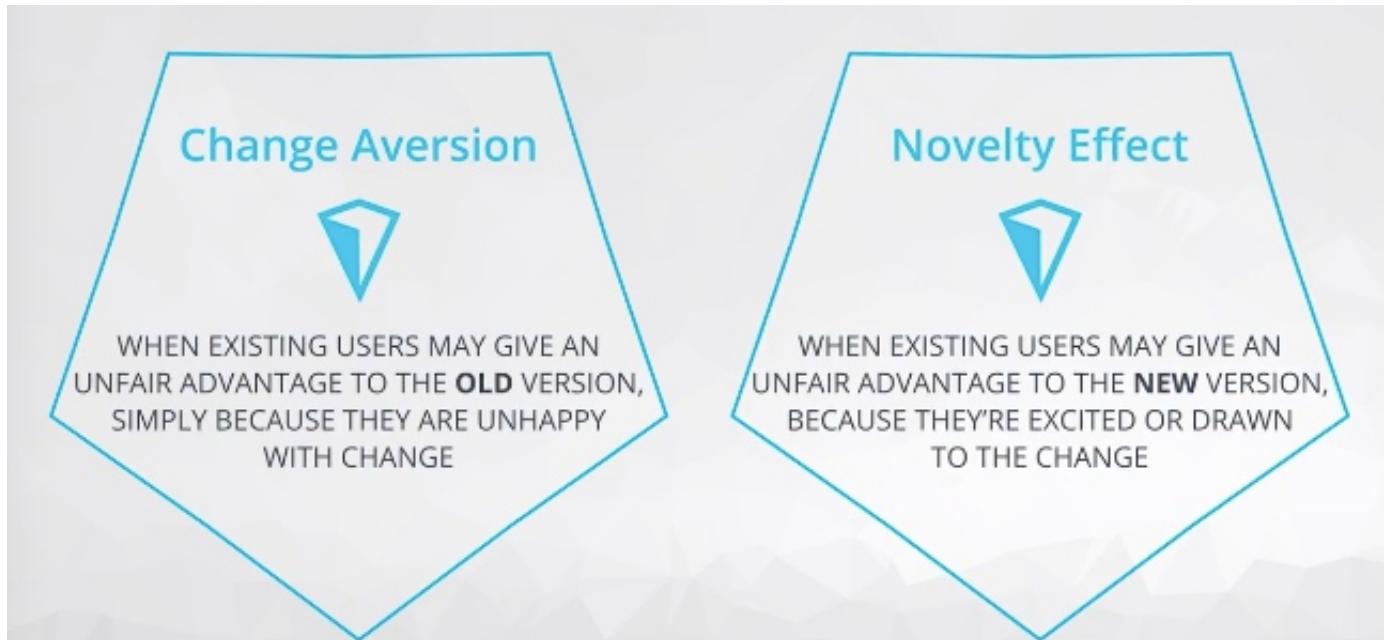


为了量化的进行考量，就要用到上面讲到的假设检验了，设置如下：

- 零假设：新版本不比旧版本好，甚至比旧版本差
- 对立假设：新版本比旧版本好



需要注意的一点是，A/B测试并不适合所有变化（比如上产品A更好还是产品B更好），而且在A/B测试中也可能引入偏见，比如课程中关于两类用户，一类偏向保持原状、另一类偏向变化求新的例子：



## \*/ Udacity的案例

课程中居了Uda家自己的案例，对于最后能达到毕业环节的来说，有一句描述：少数人坚持到了最后。所以呢，为了提高学员参与度，提高每个阶段之间（见后一节漏斗模型）的转化率，Audacity 试着做出一些改动，并对改动进行了 A/B 测试。（开始鸡汤）同学们加油啊！我们的目标是按进度毕业！（成为/赢取白富美，走上人生巅峰！）

## \*\*// Udacity的漏斗模型

漏斗模型用户流就是用户接触到了产品之后，用户在使用过程中的流转情况。以Uda为例：浏览主页 > 探索课程 > 浏览课程概述页面 > 注册课程 > 完成课程



## \*\*\*// 点击率CTR (代码)

### 8.练习：点击率

点击率 (CTR)通常是点击数与浏览数的比例。因为 Udacity 有用cookies，所以可以确认单独用户，确保不重复统计同一个用户的点击率。在考察新版本主页是否可以带来更多的点击时，我们使用CTR (click through rate) 点击率进行衡量： $CTR = \text{单独用户点击数} / \# \text{ 单独用户浏览数}$ 。在指标确定之后，就可以写出假设了：

$$H_0 : CTR_{new} - CTR_{old} \leq 0$$

$$H_1 : CTR_{new} - CTR_{old} > 0$$

这部分的代码主要是筛选和一个datetime的计算，对于过了p3的你来说完全能够胜任。

## \*\*\*// 注册率 (代码)

### 10.指标-注册率

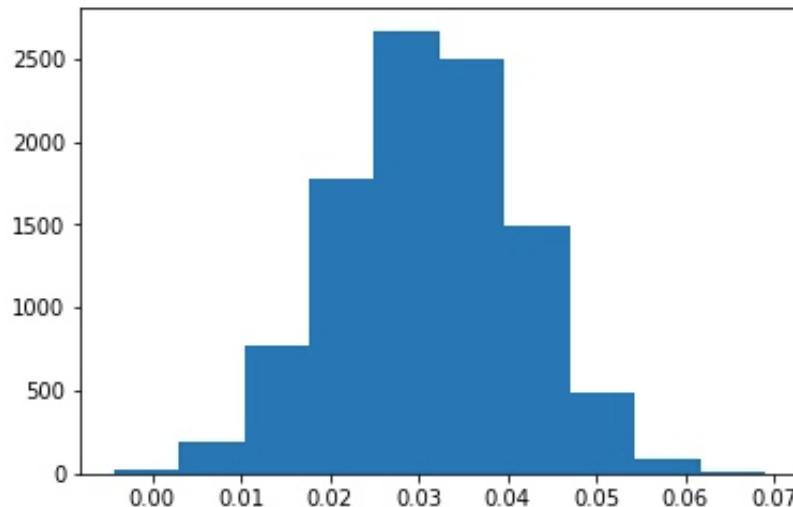
注册率就是用户注册的比率，我们来串一下此处新出现的代码：

- 先是把数据集分开为控制组和实验组。分别计算出两组的ctr，之后求差的出obs\_diff。
- 再就是用自展法抽样了10000遍，计算出这10000次的差异：

```
1 # 设定空列表
2 diffss = []
3 # 循环1w遍
4 for i in range(10000):
5     # 抽样每次抽样都是和df相同的数量（所以会比较慢）
6     # 使用自展法
7     b_samp = df.sample(df.shape[0], replace = True)
8     # 区分control组与experiment组
9     control_df = b_samp.query('group == "control"')
10    experiment_df = b_samp.query('group == "experiment"')
11    # 得出两个组的CTR
12    control_ctr = control_df.query('action == "enroll").id.nunique() /
13    control_df.query('action == "view").id.nunique()
14    experiment_ctr = experiment_df.query('action == "enroll").id.nunique() /
15    experiment_df.query('action == "view").id.nunique()
16    # 计算CTR的差值追加到diffss中
17    diffss.append(experiment_ctr - control_ctr)
18
19 # 之后将列表转为array格式（好用np的特性）
20 diffss = np.array(diffss)
# 画个图
```

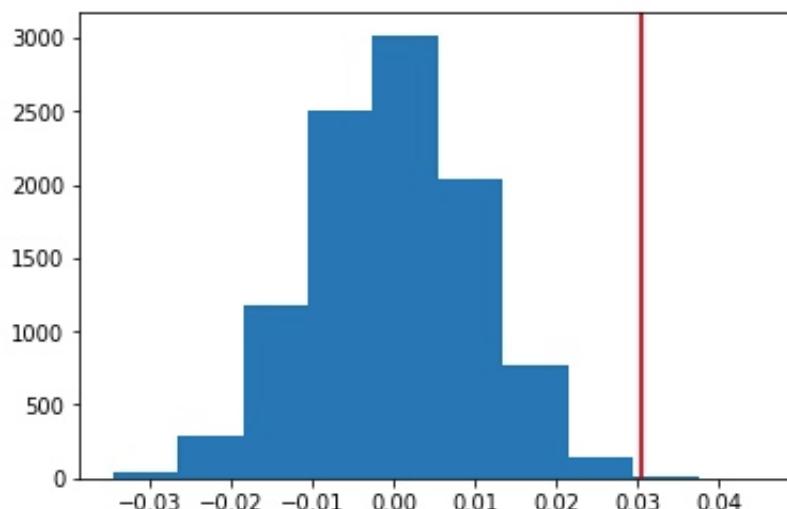
```
plt.hist(diffs)
```

注意，这时候的图是体现实际情况的（有0.03左右的差异，实验组的转化率更高，注意看x轴的标签）：



接下来，我们用实际数据的标准差std模拟下 $H_0$ 假设的时候，两个实验的偏差应该是什么样的：

```
1 # Simulate distribution under the null hypothesis
2 # 使用.random.normal模拟正态分布
3 # 0为正态分布的中点（因为H0假设是没区别，所以中点为0）
4 # 使用实际数据的diffs.std()进行模拟 (std决定了正态分布的宽窄)
5 # 数量依旧和实际保持一致 (diffs.size)
6 null_vals = np.random.normal(0, diffs.std(), diffs.size)
7
8 # 再画个图呗
9 plt.hist(null_vals)
10 plt.axvline(x = obs_diff, color = 'red')
```



如上图，看出区别了没啊？这次的中点在0，因为这是对 $H_0$ 假设的模拟。再往下，就可以把两

个图做比较了， p值出场：

```
1 # 直接比较 p值
2 # null_vals > obs_diff 是直接使用的向量计算
3 # 生成的是一个True False的 array
4 # 直接使用mean就会得出True的比率
5 # 因为True相当于1, False相当于0
6 (null_vals > obs_diff).mean()
```

## \*// 其他单项指标（代码）

- 11. 指标-平均浏览时长
- 12. 指标-平均教室逗留时长
- 13. 指标-完成率

这几部分都大同小异了，大家按照逻辑自己完成就好（有的内容在讲解的视频里已经又了代码，可以暂停参考）

## \*\*// 分析多个指标

你评估的指标越多，你观察到显著差异的偶然性就越高。这就多重比较问题（[https://en.wikipedia.org/wiki/Multiple\\_comparisons\\_problem](https://en.wikipedia.org/wiki/Multiple_comparisons_problem)）。我们可以使用Bonferroni（邦弗朗尼）校正来修复，就是把p值再除以比较的指标数，再进行判断（比如说这个案例中有4个参数进行了比较，我们就除以4）。那么0.05的显著值就会变味0.0125的显著值。所以这4个指标中，就变成只有平均浏览时长有统计显著性了。

在课程的最后答案中，其实对于这次的修正也表达了态度：虽然有待商榷，但考虑到结果没几个具有显著性的，而且指标之间具明显关联，所以我们可以认为的确太保守了。（正话反话都说了，所以说数据分析中的经验和直觉也很重要啊！）

## \*\*\*// 得出结论

那么我们怎么半，正常的检测4个指标有3个是显著性，但Bonferroni矫正以后关键指标就不显著了。其实这在正常中是经常发生的，好在我们还有其他不那么保守的方法进行修正：

- 封闭测试程序 [https://en.wikipedia.org/wiki/Closed\\_testing\\_procedure](https://en.wikipedia.org/wiki/Closed_testing_procedure)
- Boole-Bonferroni联合校正 [https://en.wikipedia.org/wiki/Boole%27s\\_inequality](https://en.wikipedia.org/wiki/Boole%27s_inequality)
- Holm-Bonferroni方案 [https://en.wikipedia.org/wiki/Holm%E2%80%93Bonferroni\\_method](https://en.wikipedia.org/wiki/Holm%E2%80%93Bonferroni_method)

这中间的要点是不要为了显著性去凑p值（很容易的），而是为了真正的考察你的数据，否则就

变成了作弊，看看这个就知道你是可以用数据干坏事的了：<https://freakonometrics.hypotheses.org/19817>

对于A/B测试的难点，课程的最后一节总结非常到位，建议完成：<https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part/4e7e2f82-e05e-4fbe-b29c-fe3169c6dd77/modules/0596b9e8-4a3a-41c3-a929-6c72c0c93925/lessons/f3088061-6562-445e-b1a7-a86c9389c2b5/concepts/f6eae35d-536b-48d0-b97e-93e4fe93459d#>

# /彩蛋/

## \*/ 希腊数字的读法

在刚开始学统计学的时候，最尴尬的就是看着一堆鬼画符他认识我，我不认识他。一张嘴就念错就糗大了，特意放上下面这张图，让大家想说就说，说的痛快：

| 序号 | 大写 | 小写 | 英文注音    | 国际音标注音    | 中文注音 | 意义                    |
|----|----|----|---------|-----------|------|-----------------------|
| 1  | A  | α  | alpha   | a:lfa     | 阿尔法  | 角度；系数                 |
| 2  | B  | β  | beta    | bet       | 贝塔   | 磁通系数；角度；系数            |
| 3  | Γ  | γ  | gamma   | ga:m      | 伽马   | 电导系数（小写）              |
| 4  | Δ  | δ  | delta   | delt      | 德尔塔  | 变动；密度；屈光度             |
| 5  | Ε  | ε  | epsilon | ep`silοn  | 伊普西龙 | 对数之基数                 |
| 6  | Z  | ζ  | zeta    | zat       | 截塔   | 系数；方位角；阻抗；相对粘度；原子序数   |
| 7  | H  | η  | eta     | eit       | 艾塔   | 磁滞系数；效率（小写）           |
| 8  | Θ  | θ  | thet    | θit       | 西塔   | 温度；相位角                |
| 9  | I  | ι  | iot     | aiot      | 约塔   | 微小，一点儿                |
| 10 | K  | κ  | kappa   | kap       | 卡帕   | 介质常数                  |
| 11 | Λ  | λ  | lambda  | lambd     | 兰布达  | 波长（小写）；体积             |
| 12 | M  | μ  | mu      | mju       | 缪    | 磁导系数；微（千分之一）；放大因数（小写） |
| 13 | N  | ν  | nu      | nju       | 纽    | 磁阻系数                  |
| 14 | Ξ  | ξ  | xi      | ksi       | 克西   |                       |
| 15 | O  | ο  | omicron | omik`ron  | 奥密克戎 |                       |
| 16 | Π  | π  | pi      | pai       | 派    | 圆周率=圆周÷直径=3.1416      |
| 17 | P  | ρ  | rho     | rou       | 肉    | 电阻系数（小写）              |
| 18 | Σ  | σ  | sigma   | ˋsigma    | 西格马  | 总和（大写），表面密度；跨导（小写）    |
| 19 | T  | τ  | tau     | tau       | 套    | 时间常数                  |
| 20 | Υ  | υ  | upsilon | jup`silon | 宇普西龙 | 位移                    |
| 21 | Φ  | φ  | phi     | fai       | 佛爱   | 磁通；角                  |
| 22 | X  | χ  | chi     | phai      | 西    |                       |
| 23 | Ψ  | ψ  | psi     | psai      | 普西   | 角速；介质电通量（静电力线）；角      |
| 24 | Ω  | ω  | omega   | o`miga    | 欧米伽  | 欧姆（大写）；角速（小写）；角       |

## \*/ T检验

虽然我们在项目中将会使用自展法（替代传统的统计方法了）进行计算，但为了能够对课程中的偶尔蹦出的传统方式更好的理解，做一点点扩展，感兴趣的看一遍就好了。

- 首先概括讲下z-Test和t-Test的区别：

- 首先要知道，他俩都是用作正态分布的数据上的
- 其实这两个是差不多的，都是平均值差异的检验方法
- z-Test对应的是拥有总体数据（每一个数据）
- t-Test对应的是拥有样本数据（而且还比较少，30个以下）
- 所以说z-Test考察的是总体的参数；而t-Test考察的是样本的统计值
- 当然在单样本，统计量很大的时候，可以与z-Test互换
- 因为我们大都是收集到的样本，所以用的z-Test最多
- 课程中的附加说明：<http://www.differencebetween.net/miscellaneous/difference-between-z-test-and-t-test/>

- 接下来我们说下t-Test的分类

- 单样本T检验，就是检验你的样本的均值。比如说课程中的扩展例子：一个橄榄球教练，用他球队的数据（就是样本了）进行计算，检查他球队的均值 ( $\bar{x}$ ) 和总体球队的平均值 ( $\mu$ 之前已经知的，官方数据）。既然是t检验，就会计算出一个t值来 (t-value)，之后根据t-table（就是根据你假设检验的参数：a置信区间、b数据个数、c双尾还是单尾的输入，将不同组合的结果写成的一张速查表）查询到t-critical值，将这两个作出对比。t-value > t-critical 就是显著的。但在这个例子里，t-value < t-critical，所以说数据表明这位教练的球队不比其他的球队更优秀。（<https://www.cliffsnotes.com/study-guides/statistics/univariate-inferential-tests/one-sample-t-test>）
- 双样本t检验，就是说在整个的比较中，从样本中生成了两个可以区分的内容进行检验。细分的话，可以有更多的小点。
  - 首先是这个对比是不是equal的，那么什么是equal的呢？就是说你对比的两个样本中的每一个元素都是对应的。我们来举个例子，比如说我们考察班上50名同学的测试成绩。第一个样本是上复习课之前的测验成绩，从 $x_1$ 到 $x_{50}$ 一共有50个；在上过数学课之后我们再测试一下，得到的成绩是从 $y_1$ 到 $y_{50}$ 一共有也是50个。每个x和y的小标都是学号，所以说这两个样本只是成绩不同，而每个同学都参加了2个测试。这种情况就是equal的，特点可以看出来两个样本是相同的。
  - 再就有我们还可以发现上面的这个例子还是paired（配对的），因为每个学生都参与了2个测试（比equal还多个条件，一一对应）。对于paired t-test的详细说明：<http://www.statstutor.ac.uk/resources/uploaded/paired-t-test.pdf>
  - 我们来举个不是equal的测试例子：调查了很多人，根据吸烟和不吸烟分为两个样本，这里两个区分人数不同，所以就不是equal的了。详细的例子：<https://onlinecourses.science.psu.edu/stat414/node/268/>
  - 当然在计算机很发达的时候，是不是equal的并没有那么重要，使用工具（比如python）也就是一个参数的区别（几种的公式是有些区别的）。two-sample t-test 的扩展：<https://www.itl.nist.gov/div898/handbook/eda/section3/eda353.htm>



Title: W11 Plus 回归分析

Tags: 数据分析初级, 实战项目

# W11 Plus 回归分析

---

- [W11 Plus 回归分析](#)
- [/学习地图/](#)
  - [/ 项目路径](#)
- [/目标4/ 机器学习入门](#)
  - [/ 机器学习的两种分类](#)
    - [\\*\\*// 监督学习: Supervised Learning](#)
    - [\\*\\*// 非监督学习: Unsupervised Learning](#)
  - [\\*\\*\\*/ 线性回归: Simple Liner Regression](#)
    - [\\*\\*\\*// 相关系数](#)
    - [\\*// 回归线由什么决定](#)
    - [\\*\\*// 拟合回归线](#)
    - [\\*\\*// 拟合回归线 \(代码\)](#)
    - [\\*// 决定系数 \(相关系数的平方 \\$ r^2 \\$\)](#)
  - [\\*/ 多元线性回归](#)
    - [\\*\\*// 多元线性回归能解决什么问题](#)
    - [\\*\\*// 虚拟变量](#)
    - [// 多重线性 \(选学\)](#)
    - [\\*\\*// 交叉验证和k折交叉验证](#)
  - [\\*\\*\\*/ 逻辑回归](#)
    - [\\*// 模型诊断](#)
    - [\\*\\*\\*// 精确率和召回率](#)
    - [\\*// 统计学到机器学习](#)

## /学习地图/

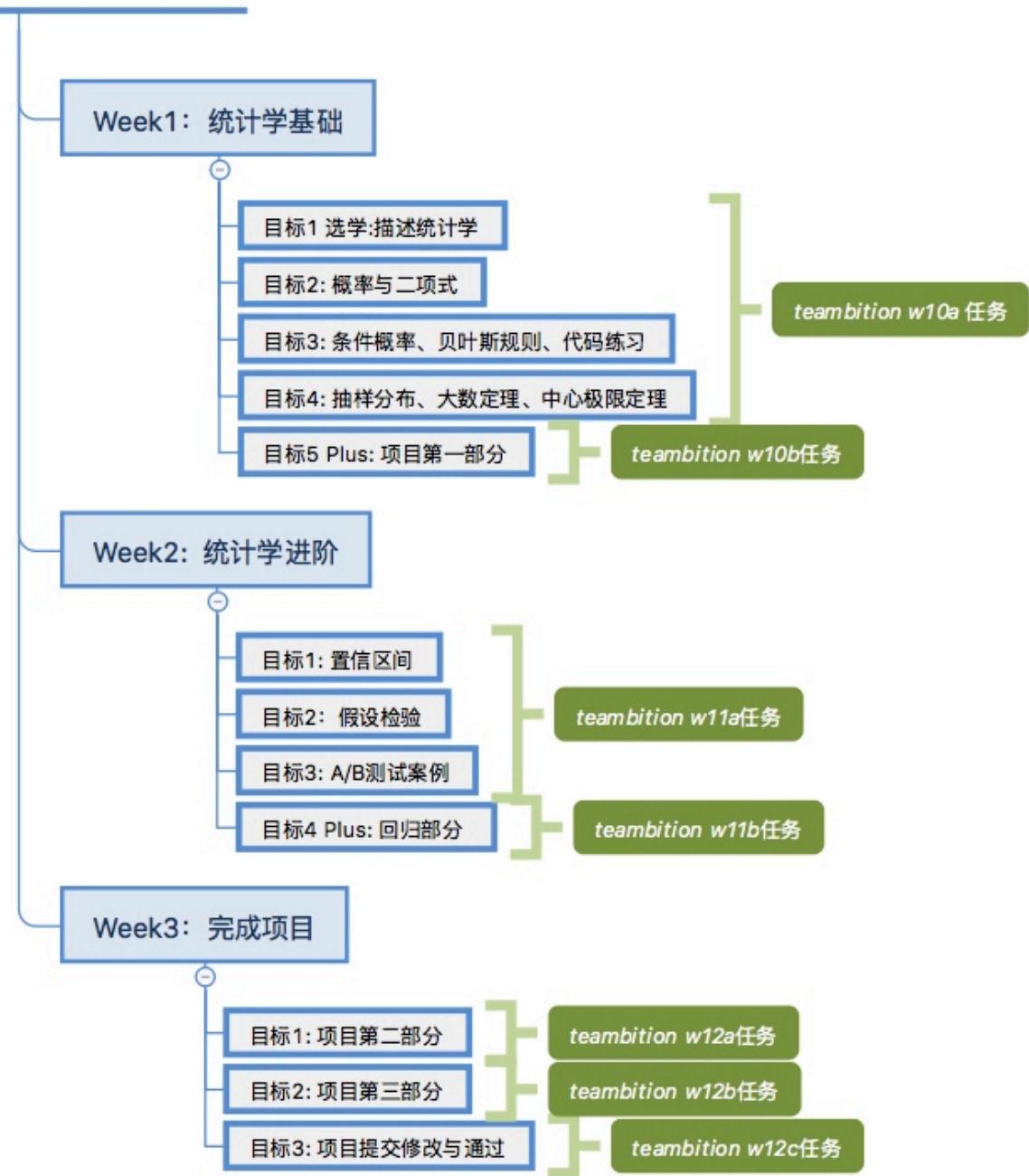
---

本周是统计学项目的Part2，是项目4的推论统计学的扩展部分。目标1、2、3讲解置信区间、假设检验、和一个A/B测试案例。目标4作为Plus发布，讲解了回归的相关知识。

! 注意，1星和2星的可以只看本导学，先大致理解就可以。3星的是本周重点！

# / 项目路径

## 分析 A/B 测试结果



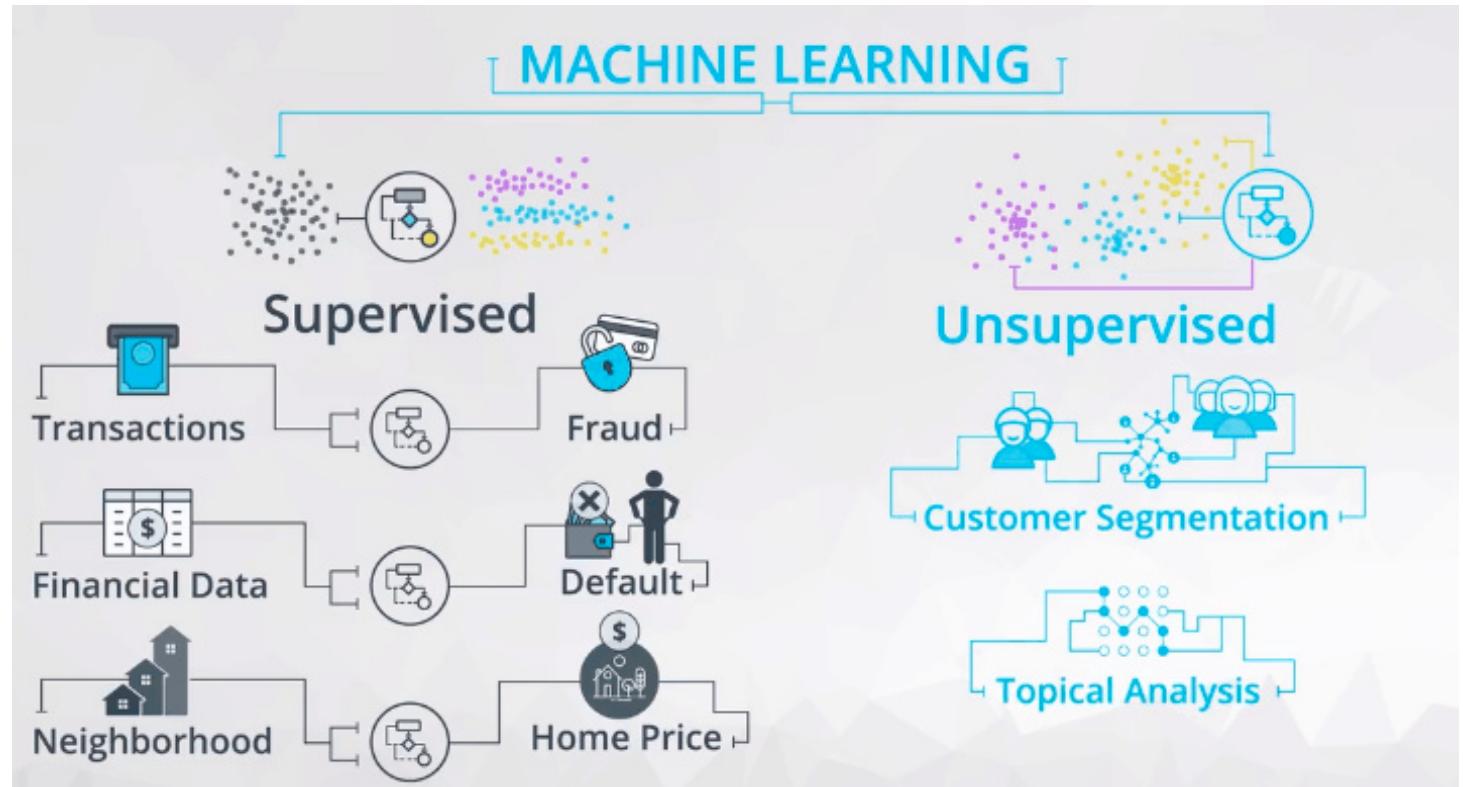
## / 目标4/ 机器学习入门

这部分是机器学习中的内容（高大上不，啧啧！），其实包括了3个入门部分：

- 识别可以使用回归的场景
- 理解回归如何工作
- 使用代码实现回归

# / 机器学习的两种分类

机器学习就是machine learning，可以分为两类，如下图（请配合后面解释看，图来自Uda课程）：



## \*\*// 监督学习：Supervised Learning

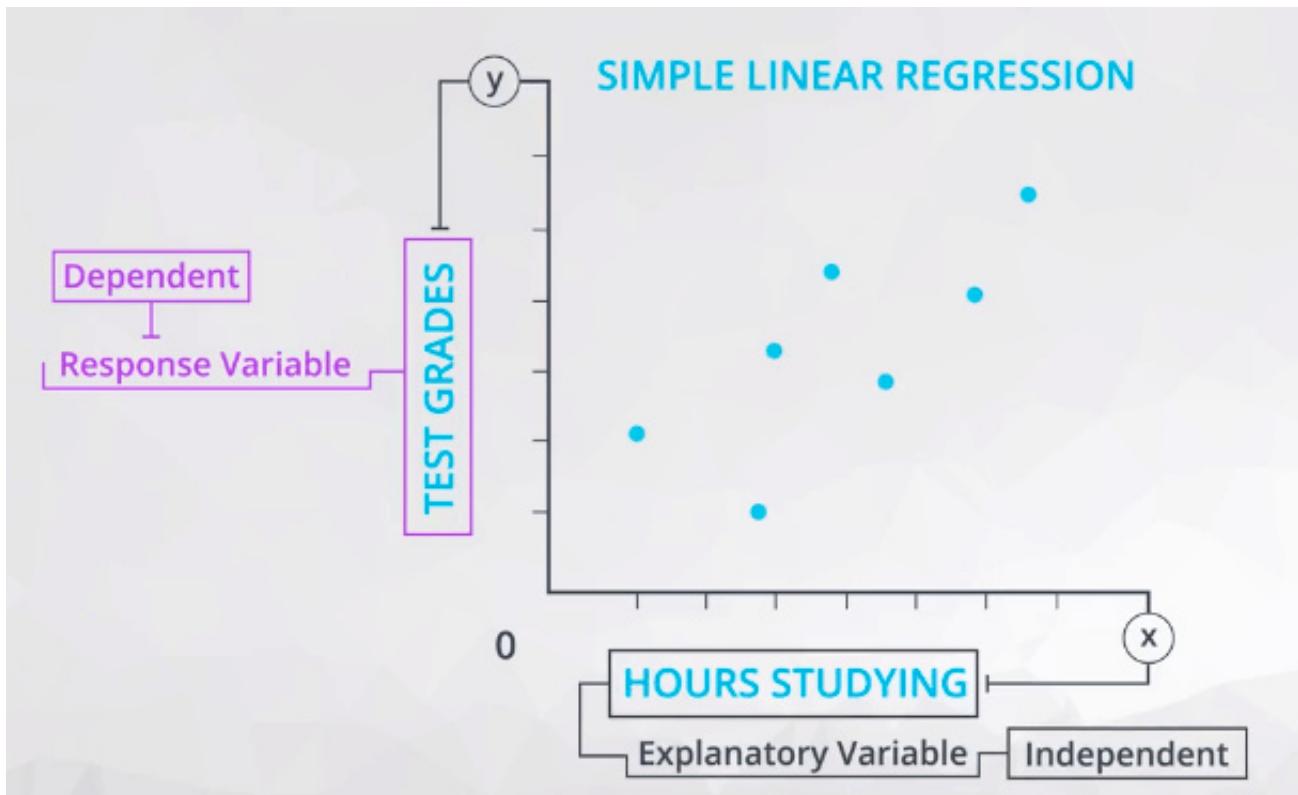
- Input data is used to predict a value of label
- 目的是根据现有模型预测新的数据，比如说购买行为，交易是否有欺诈
- 线性回归和逻辑回归都是监督学习的分支

## \*\*// 非监督学习：Unsupervised Learning

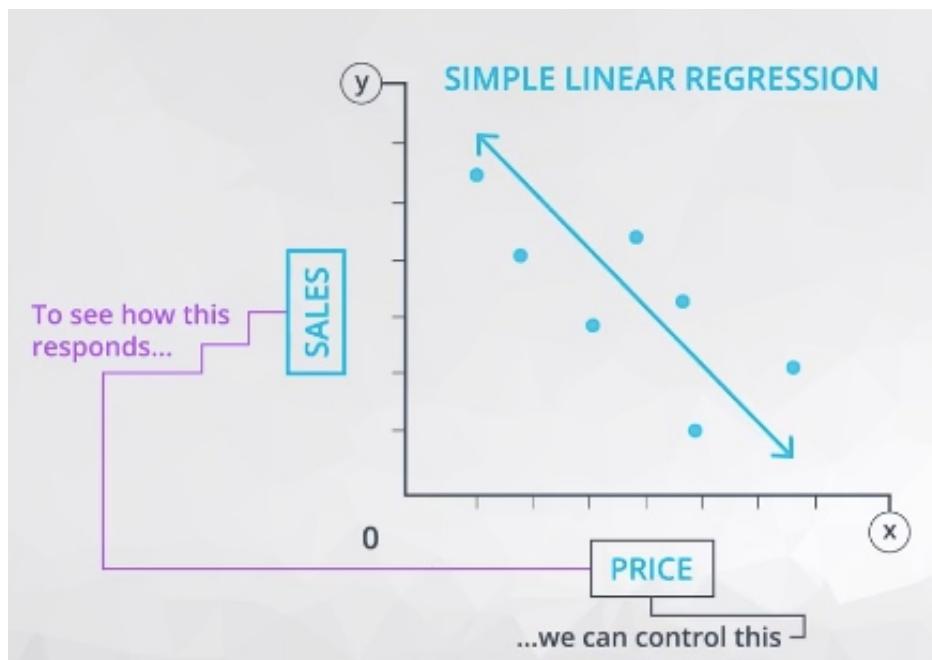
- Clustering data based on common characteristics
- 用于自动发现数据的规律，比如说对给定数据创建聚类
- 本学位不涉及，有兴趣请入坑机器学习纳米工程师学位

## \*\*\*/ 线性回归：Simple Linear Regression

线性回归就是对两个变量做比较。比如看体重和摄入热量之间的变化关系啊，或者看价格和销量的这种关系，听着挺高大上，其实我们日常生活中已经见过很多了。参见下面这个图我们说一下几个名词的意思，大家就好理解了：



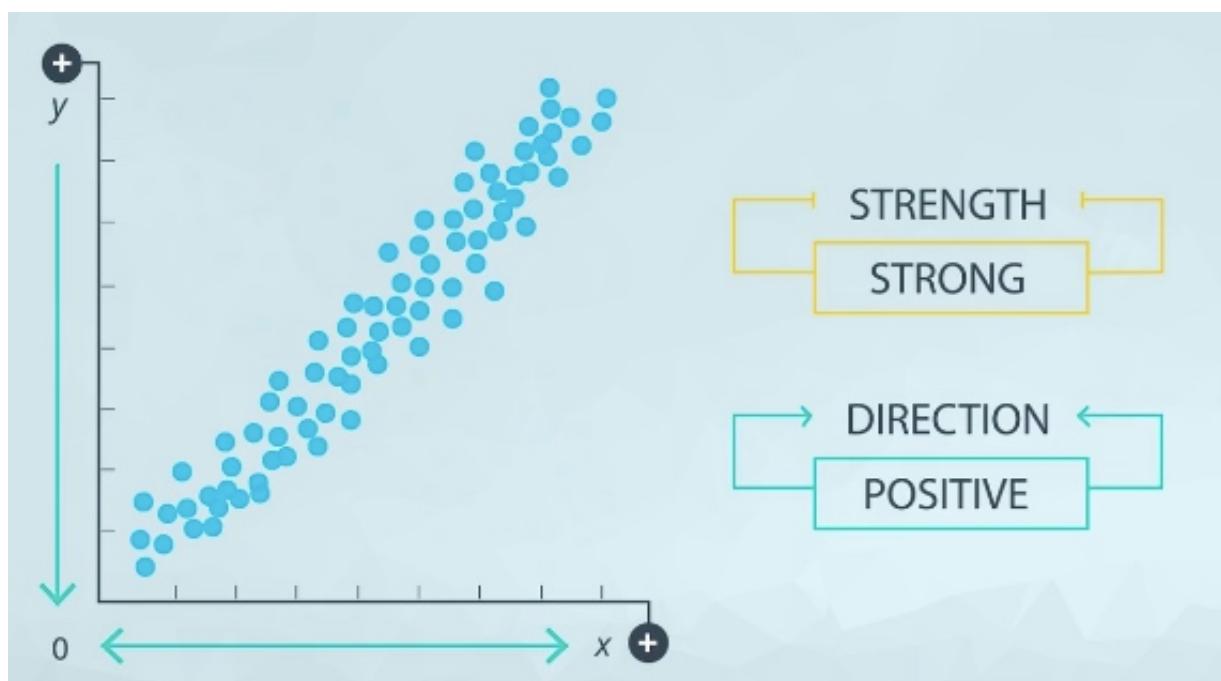
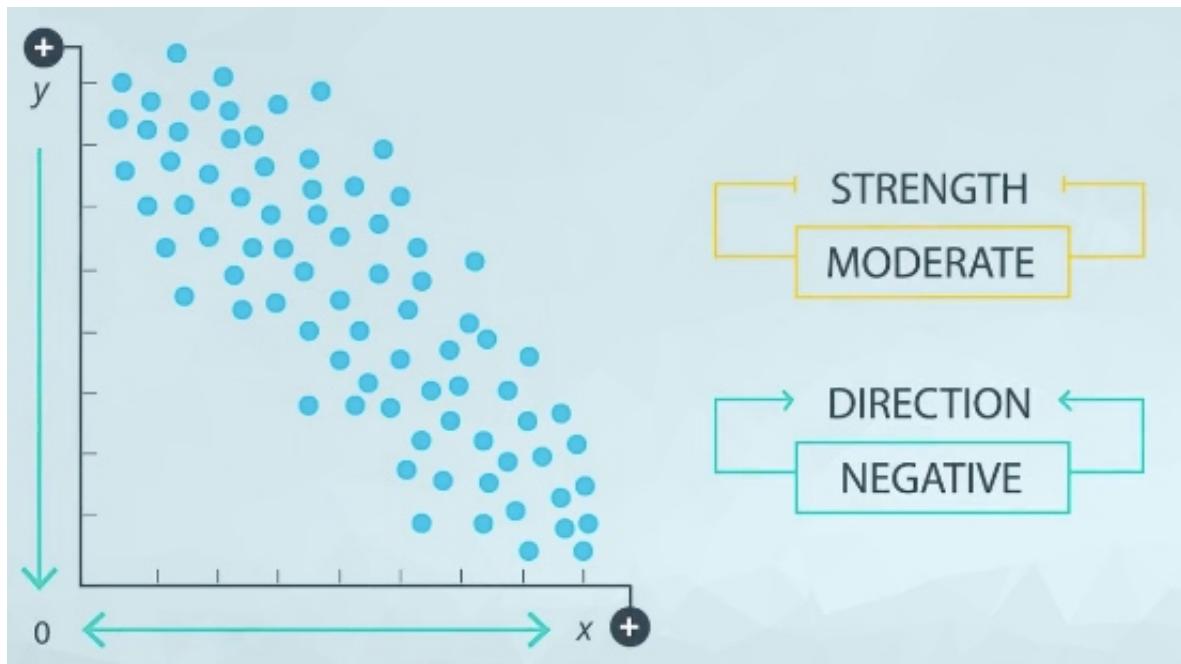
- 反应变量 **Response Variable (y)** 是你想预测的变量（也叫依赖变量Dependent）
- 解释变量 **Explanatory Variable (x)** 是用于预测反应变量的变量（也叫自变量Independent）
- 其实把就事一个散点图，x轴是基准的变量，y值是要考量的变量，画个图就能看出两个变量的关系了（也能看出一些趋势），如下图根据这些点，我们可以做出一条线，这条线就叫线性回归 **Simple Liner Regression**



## \*\*\*// 相关系数

那么既然我们能够画出了散点图，那么我们怎么用术语描写散点图的趋势呢。首先我们使用强度

**Strength**表示连个变量的关系是否足够明确，然后我们使用方向 **Direction**描述两个变量的相互关系，有图有真相：



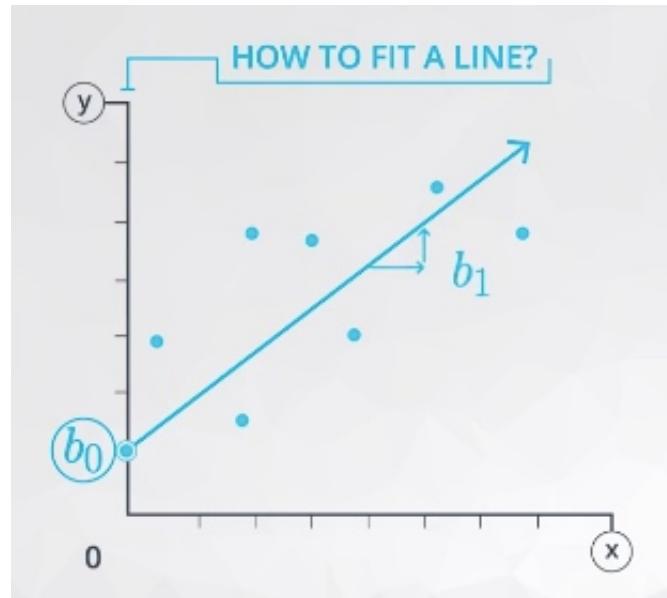
那么我们怎么比较专业的衡量这两个趋势呢（靠眼神？！），这就有了下一个概念：**相关系数 Correlation Coefficient**：反应一段关系的强度和关系（好像走错片场到了心里咨询室）。The strength and direction of a linear relationship. 用小写r表示。

- 具体的解释：[https://en.wikipedia.org/wiki/Pearson\\_correlation\\_coefficient](https://en.wikipedia.org/wiki/Pearson_correlation_coefficient)
- 也有其他的可选：<http://www.statisticssolutions.com/correlation-peacock-kendall-spearman/>
- 斯皮尔曼相关性系数则不只衡量线性关系，可能更适用于关联两个变量的场合：[https://en.wikipedia.org/wiki/Spearman%27s\\_rank\\_correlation\\_coefficient](https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient)

## \*// 回归线由什么决定

既然我们能够画出一条回归线，那么我们又怎么定义这条回归线呢？答案是斜率-定义线的方向，截距-定义线与y轴的交点：

- 截距的定义为：当  $x$  变量为 0 时，反应变量的预测值。
- 斜率的定义为： $x$  变量每增加一个单位引起的反应变量的预测变化



为了我们的课程能够高大上一点，我们还是要说下公式的： $\hat{y} = b_0 + b_1 x_1$

- $\hat{y}$  为回归线反应变量的预测值。
- $b_0$  为截距。
- $b_1$  为斜率。
- $x_1$  为解释变量。
- $y$  为数据集某个数据点的实际反应变量值 (不是回归线的预测值，就是对应一个给定  $x$  的  $y$  值)。

## \*\*// 拟合回归线

**最小二乘法 Least Squares Algorithm:** 计算每个点距离回归线的距离平方，找到所有值总和最小的那条线 Minimize the sum of the squared vertical distances from the line to points. 说白了就是找一条线，让所有点到线的距离的平方只和最小，公式：

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2$$

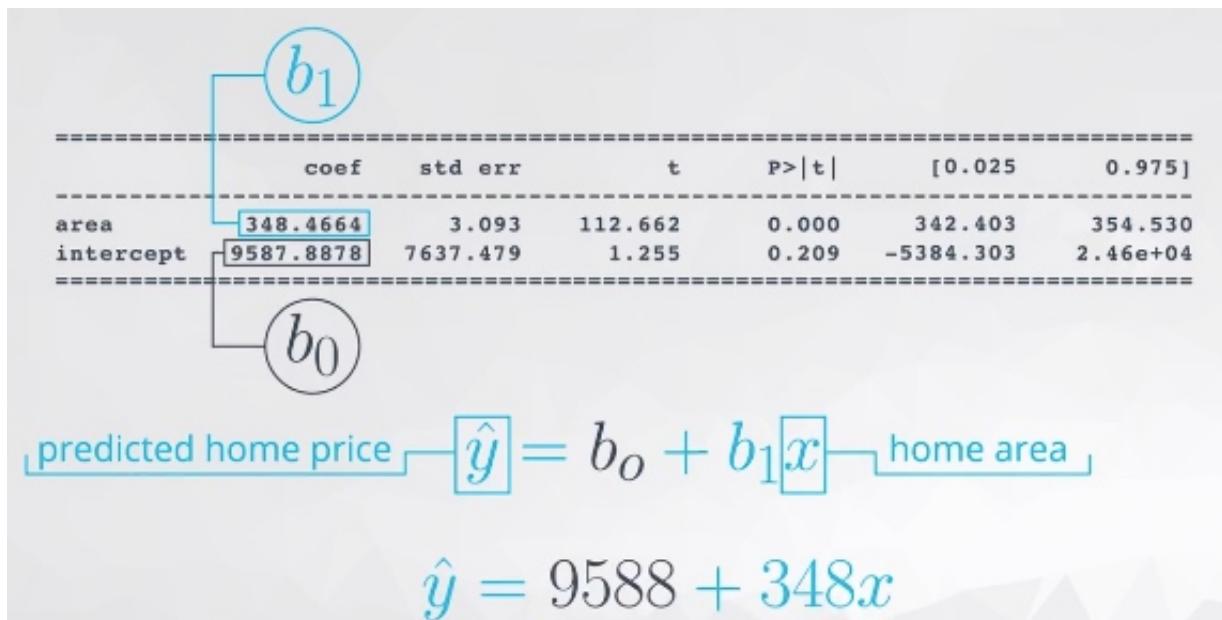
在这里平方是因为要去掉负值，和对距离远的进行惩罚。当然我们还有其他方法，但最小二乘法在很多场景都很适用，就讲着一个先。具体的数学计算在后面小节中有，如果有兴趣可以在excel中一步一步计算下（当然是完成任务以后）：<https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part/4e7e2f82-e05e-4fbe-b29c-fe3169c6dd77/modules/0596b9e8-4a3a-41c3-a929-6c72c0c93925/lessons/d780a3b0-a08e-4282-858e-6a28e8d524aa/concepts/8282fff9-1876-4d21-9093-4af37ec40455#>

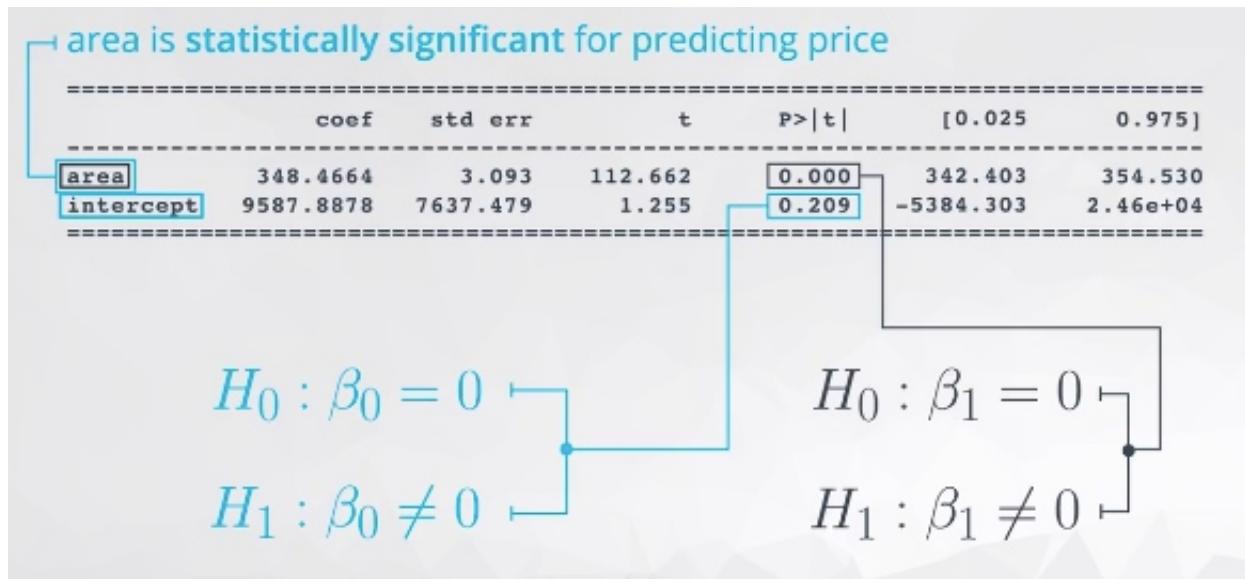
课程中也有扩展内容：<http://www.statisticshowto.com/probability-and-statistics/regression-analysis/find-a-linear-regression-equation/>

## \*\*// 拟合回归线（代码）

接下来我们就在python中搞定这个回归线，注意在输入参数的时候，截距不可少（见这个帖子：<https://stats.stackexchange.com/questions/7948/when-is-it-ok-to-remove-the-intercept-in-a-linear-regression-model>）。其实对于两个参数的预测，对于截距我们是不用太关心的，因为当对每个x变量进行假设检验，测试所涉及的两组为：总体斜率等于0 vs. 参数不等于0的其它情况（对立假设）。因此，如果斜率不等于0（即对立假设为真），那我们就能证明与那个系数有关的x变量与反应变量间有具统计显著性的线性关系，也就意味着x变量能帮我们预测反应变量（或者，最起码意味着模型里有x变量比没有好）。

回归代码结果：

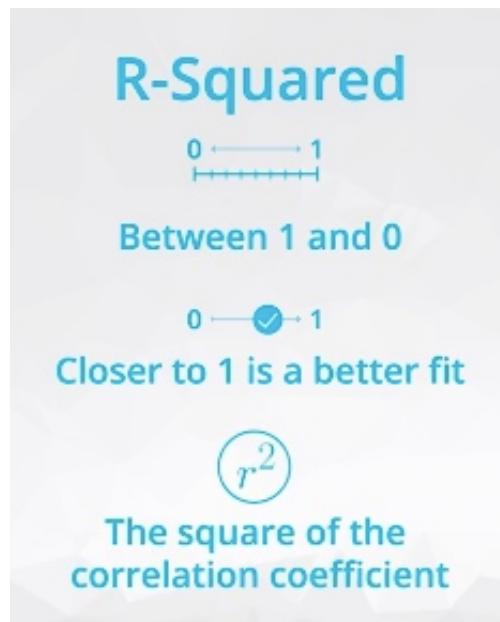




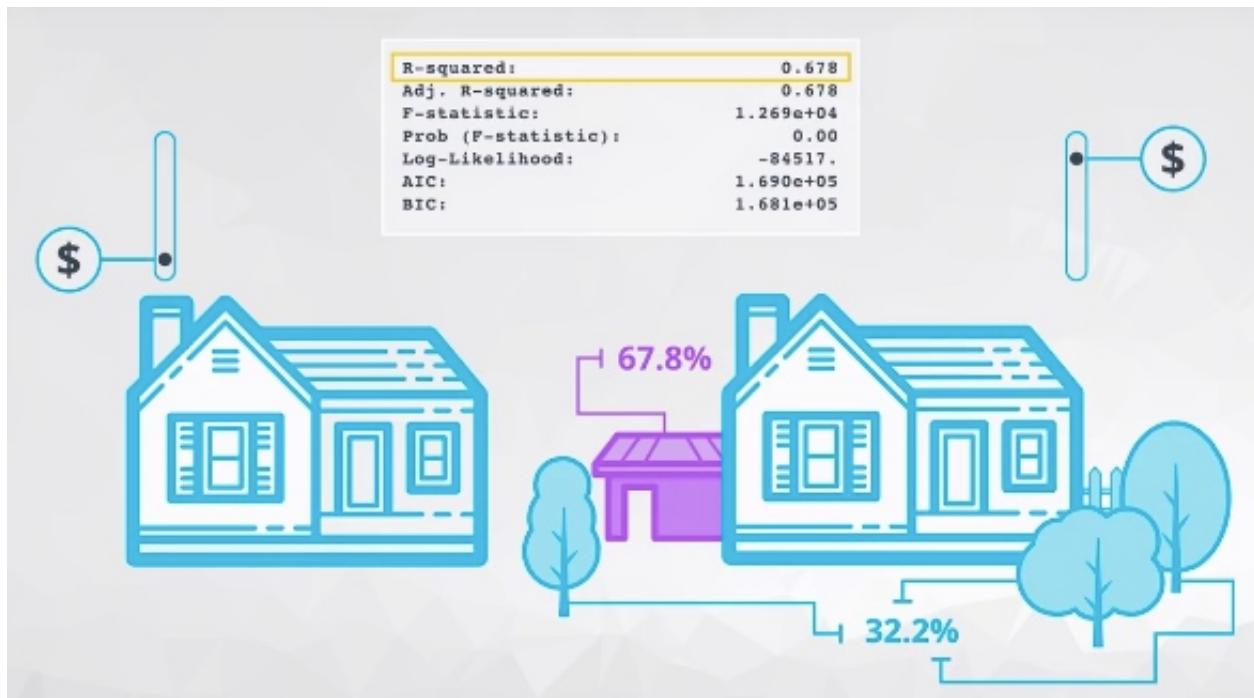
## \*// 决定系数 (相关系数的平方 $r^2$ )

那么根据相关系数是不是我们就可以推断出两个变量互相影响的百分比了呢？其实专门有个系数是反应这个比率的，就是决定系数（别急，就是相关系数的平方），这个转换和实际的比较吻合（要知道相关和决定是不同的，所以取平方，让数值下降了）。当然也有人觉得这个决定系数不是好的标准： (<https://data.library.virginia.edu/is-r-squared-useless/>) 。

**R-squared** 决定系数： The Amount of variability in the response (Y) Explained by your model. X可以解释Y变量的程度，如下图：



课程中房价的结果（后面的空间中有代码，大家可以对应一下）：



课程后面还有个钻石大小和重量之间关系分析的例子，代码解释一下：

```

1 # 首先需要设置截距，1为默认，但是要加这一列（默认操作）
2 df['intercept'] = 1
3 # 接着建立分析模式，这里用到了sm是一个统计模块（见import那里）
4 # sm.OLS就是调用最小二乘法
5 # 第一个参数是因变量（y）
6 # 第二个参数是自变量（x），注意这个后面还有把刚建立的intercept加上
7 mode = sm.OLS(df['price'], df[['carats','intercept']])
8 # 之后用mode.fit做回归分析
9 res = mode.fit()
10 # 查看结果
11 res.summary()
```

后面还有个房价的分析，大同小异，但有一个说一下，在sklearn中是内置了一些数据集的 (<http://blog.csdn.net/kancy110/article/details/73716317>) :

```

1 # 引入sklean中的load_boston
2 from sklearn.datasets import load_boston
3 # 使用load_boston()载入数据
4 boston_data = load_boston()
```

## \*/ 多元线性回归

本节将会学习多元线性回归 **Multiple Linear Regression**，用于处理多个输入，以及这种方式的优缺点及结果解释。这节还有本电子书《统计学习简介》 (<http://www-bcf.usc.edu/~gareth/ISL>

/ISLR%20First%20Printing.pdf)。除了这本书，Uda推荐了一个可汗学院免费线性代数的课程作为扩展（为了同学们，优达君也是拼了）<https://www.khanacademy.org/math/linear-algebra>。

## \*\*// 多元线性回归能解决什么问题

课程中的例子是如果你想买房，那么除了考虑面积，可能还要考虑区域（学区房了解一下？）、卧室数量、卫生间数量以及房子造型等等。在这种情况下，线性回归就不行了，就要使用多元线性回归进行分析了。多元线性回归实际就是把多个要考虑的参数转化成矩阵，使用线性代数的方式得出结论。对比下面要考量的参数和线性代数的对比就能明白（实际上代码是多输入几个参数，其他的一点变化都木有啊）：

The diagram illustrates a house price table with six columns: PRICE, NEIGHBORHOOD, AREA, BEDROOMS, BATHROOMS, and STYLE. Above the table are six circular icons representing: Price (\$), Neighborhood, Area (square outline), Bedrooms (bed), Bathrooms (shower), and Style (house).

| PRICE  | NEIGHBORHOOD | AREA | BEDROOMS | BATHROOMS | STYLE     |
|--------|--------------|------|----------|-----------|-----------|
| \$634K | A            | 1600 | 4        | 2         | ranch     |
| \$120K | B            | 3200 | 7        | 3         | victorian |
| \$920K | C            | 1200 | 2        | 2         | ranch     |
| \$124K | B            | 700  | 1        | 1         | lodge     |

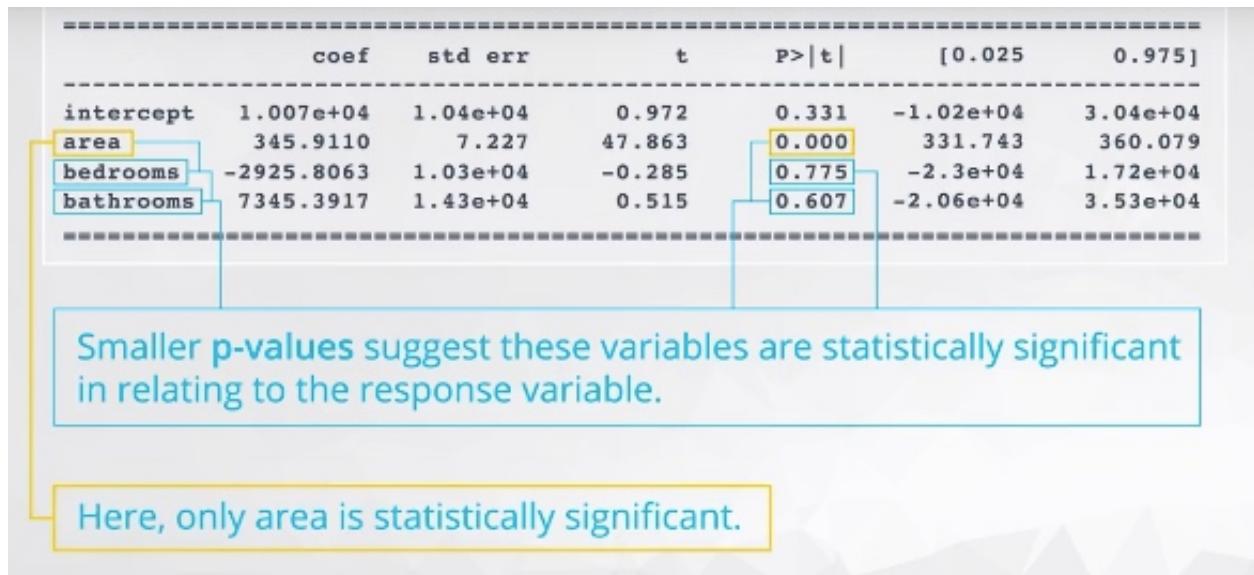
Below the table, a diagram shows a matrix multiplication process:  $A * 7 = ?$  followed by  $victorian * 7$ . A blue bracket groups the first row of the table with the first part of the equation, and another blue bracket groups the last row with the second part.

The diagram shows the same house price table converted into a matrix form. The left side is labeled 'y' and the right side is labeled 'X'.

|        | NEIGHBORHOOD | AREA | BEDROOMS | BATHROOMS | STYLE     |
|--------|--------------|------|----------|-----------|-----------|
| 634000 | A            | 1600 | 4        | 2         | ranch     |
| 920000 | B            | 3200 | 7        | 3         | victorian |
| 320000 | C            | 1200 | 2        | 2         | ranch     |
| 124000 | B            | 700  | 1        | 1         | lodge     |

(这行选看) 在结论的解释时，描述上有一点变化：在模型其它变量不变的情况下，解释变量每

增加一个单位，反应变量会随之增加的预测幅度，这种有条件的解释叫做Slope Interpretation。那么我们再观察下结论，只有面积是统计显著的：



这个结论中有亮点要注意：一是做线性回归有统计显著性的指标在做多元线性回归时可能会消失（比如bathroom，也有可能是bathroom越多，房子的面积可能会越大）。还有就是多元线性回归时无法做定类分析的：



## \*\*// 虚拟变量

那么为了对类别进行分析（比如说上面房子区域的分类），我们需要引入**虚拟变量 Dummy Variables**，将类别对应为一个矩阵（详细的方法在后面有，感兴趣请参考）：



接下来就是代码实现了：

```

1 | # 使用.get_dummies 增加列
2 | df[['A','B','C']] = pd.get_dummies(df['neighborhood'])
3 | df[['victorian','lodge','ranch']] = pd.get_dummies(df['style'])

```

| A | B | C | victorian | lodge | ranch |
|---|---|---|-----------|-------|-------|
| 0 | 1 | 0 | 0         | 1     | 0     |
| 0 | 1 | 0 | 0         | 0     | 1     |
| 0 | 1 | 0 | 0         | 1     | 0     |
| 1 | 0 | 0 | 0         | 1     | 0     |
| 0 | 1 | 0 | 0         | 0     | 1     |

再有就是检查截距（看标准状态下的价格）

```

1 | df['intercept'] = 1
2 | # 这里3个参数只选择2个，余下的一个截距就是intercept的值
3 | # 注意这里正负并没有关系
4 | lm = sm.OLS(df['price'],df[['intercept','victorian','ranch']])
5 | results = lm.fit()
6 | results.summary()

```

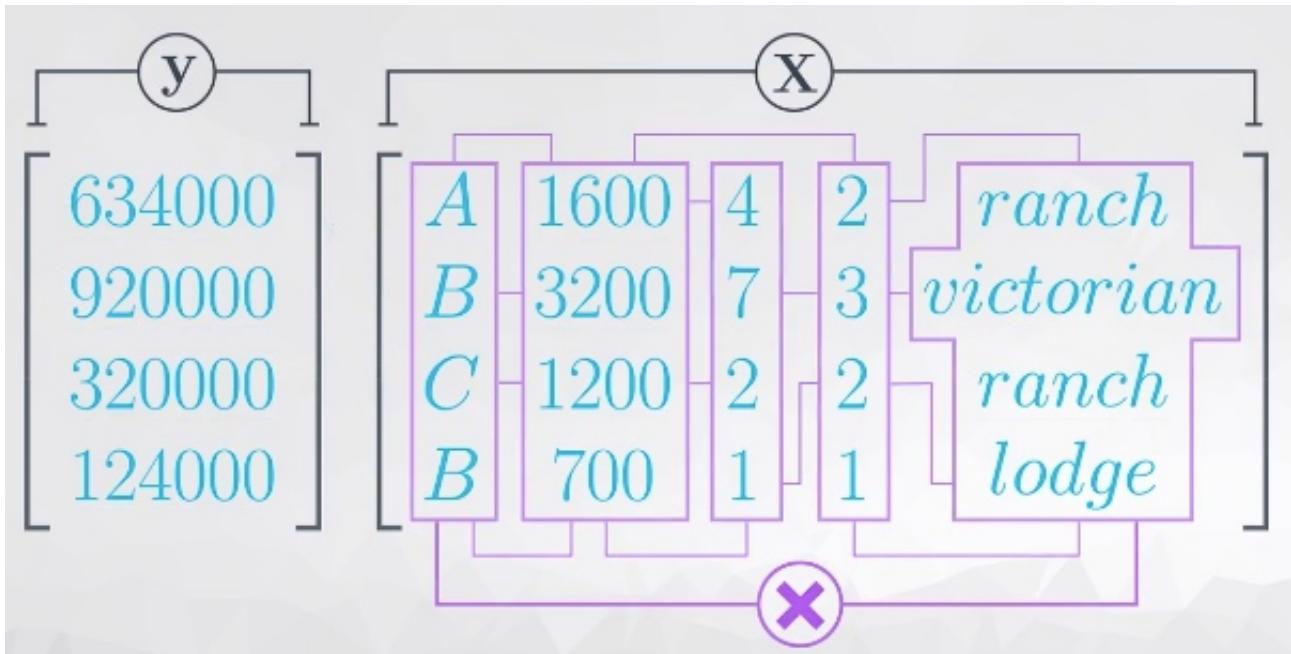
对应的就是这3个值：

| Dep. Variable:    | price            | R-squared:          | 0.339     |          |           |           |
|-------------------|------------------|---------------------|-----------|----------|-----------|-----------|
| Model:            | OLS              | Adj. R-squared:     | 0.339     |          |           |           |
| Method:           | Least Squares    | F-statistic:        | 1548.     |          |           |           |
| Date:             | Thu, 23 Aug 2018 | Prob (F-statistic): | 0.00      |          |           |           |
| Time:             | 07:42:10         | Log-Likelihood:     | -86683.   |          |           |           |
| No. Observations: | 6028             | AIC:                | 1.734e+05 |          |           |           |
| Df Residuals:     | 6025             | BIC:                | 1.734e+05 |          |           |           |
| Df Model:         | 2                |                     |           |          |           |           |
| Covariance Type:  | nonrobust        |                     |           |          |           |           |
|                   | coef             | std err             | t         | P> t     | [0.025    | 0.975]    |
| intercept         | 5.751e+05        | 1e+04               | 57.354    | 0.000    | 5.55e+05  | 5.95e+05  |
| victorian         | -2.701e+05       | 1.57e+04            | -17.153   | 0.000    | -3.01e+05 | -2.39e+05 |
| ranch             | 4.71e+05         | 1.27e+04            | 37.115    | 0.000    | 4.46e+05  | 4.96e+05  |
| Omnibus:          | 1340.120         | Durbin-Watson:      |           | 2.004    |           |           |
| Prob(Omnibus):    | 0.000            | Jarque-Bera (JB):   |           | 3232.810 |           |           |
| Skew:             | 1.230            | Prob(JB):           |           | 0.00     |           |           |
| Kurtosis:         | 5.611            | Cond. No.           |           | 4.12     |           |           |

这个部分就到此为止，其实后面还有标为[选学]的扩展内容，对这个分析的5个假设做了探讨，并且具有超多的额外资料，建议复盘时候再学习：<https://classroom.udacity.com/nanodegrees/nd002-cn-basic-vip/part/4e7e2f82-e05e-4fbe-b29c-fe3169c6dd77/module/0596b9e8-4a3a-41c3-a929-6c72c0c93925/lesson/49462f74-b030-4bb6-bf67-8281c9181404/concept/df69d406-341a-4dd1-827d-31a85e9d8ac1#>

## // 多重线性（选学）

(选学) 那么接下来，我们展开刚才提到的一个问题，我们要预测房价 (y)，我们有很多因素 (x)，我们希望x之间不要有关系（比如房价例子中的卫生间、卧室和面积之间其实是强相关的）。如果发生了这种情况，就是产生了多重共线性 (VIF)，就如同下图示例：



在实际操作的时候我们可以通过散点图的矩阵或者计算VIFs来考察x之间的关系：



当我们发现VIFs的时候，解决方法是去掉VIF>10的参数中比较不重要的那个，比如下面的Bedrooms和Bathrooms中干掉一个：



## \*\*// 交叉验证和k折交叉验证

那么假设我们对房价的回归分析选好了相关的feature，也做出了预测，怎么知道我们的选择是合适的呢。干脆我们把能得到的数据分为2部分，90%用来训练我们的参数选择，另外留下10%用来测试我们得出的解。这种方式就叫做交叉验证：

| TRAIN <i>Fit the model on this data</i> |              |      |          |           |           |
|-----------------------------------------|--------------|------|----------|-----------|-----------|
| PRICE                                   | NEIGHBORHOOD | AREA | BEDROOMS | BATHROOMS | STYLE     |
| \$634K                                  | A            | 1600 | 4        | 2         | ranch     |
| \$120K                                  | B            | 3200 | 7        | 3         | victorian |
| \$920K                                  | C            | 1200 | 2        | 2         | ranch     |
| \$134K                                  | A            | 1600 | 4        | 2         | ranch     |
| \$200K                                  | B            | 3200 | 5        | 3         | victorian |
| \$220K                                  | C            | 1200 | 6        | 2         | ranch     |
| \$133K                                  | B            | 700  | 2        | 1         | lodge     |

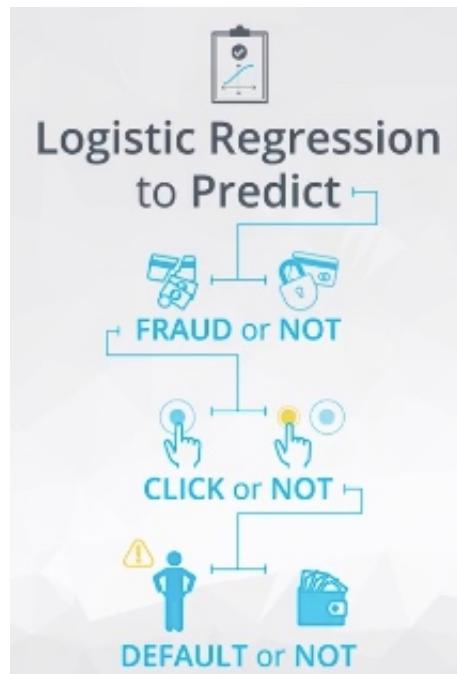
  

| TEST <i>Obtain the metric on this data</i> |  |  |  |  |  |
|--------------------------------------------|--|--|--|--|--|
|--------------------------------------------|--|--|--|--|--|

但有时候我们的数据集来之不易，如果简单的这么分割，会有10%的数据不能用来训练，而沿着也只有10%。于是就有了一种k折叠交叉验证，折叠的意思就是把数据集分为n份，轮流的进行训练和测试，再把结果统一起来（这样就利用到所有的数据进行训练和测试了，有兴趣的选学看课程）。

## \*\*\*/ 逻辑回归

前面讲的线性回归是对数值的预测，那么最后的这一节，我们来讲讲对分类的预测（逻辑回归），比如说是否欺诈、是否点击等：



逻辑回归就是用来预测范围在 0 和 1 之间的概率的。（使用的是线性模型预测堆书几率，如后面的公式图，了解下就好了）：

| TRANSACTIONS |           | FRAUD |   |
|--------------|-----------|-------|---|
|              |           | FRAUD | 1 |
| FRAUD        | NOT FRAUD | 0     |   |
|              | NOT FRAUD | 0     |   |
| NOT FRAUD    |           | 0     |   |
|              |           | FRAUD | 1 |
|              |           | FRAUD | 1 |
|              |           | FRAUD | 1 |
| NOT FRAUD    |           | 0     |   |

$$\log\left(\frac{p}{1-p}\right) = b_0 + b_1x_1 + b_2x_2 + \dots$$

probability of category 1 occurring

Taking the log controls our predictions to be between 0 and 1

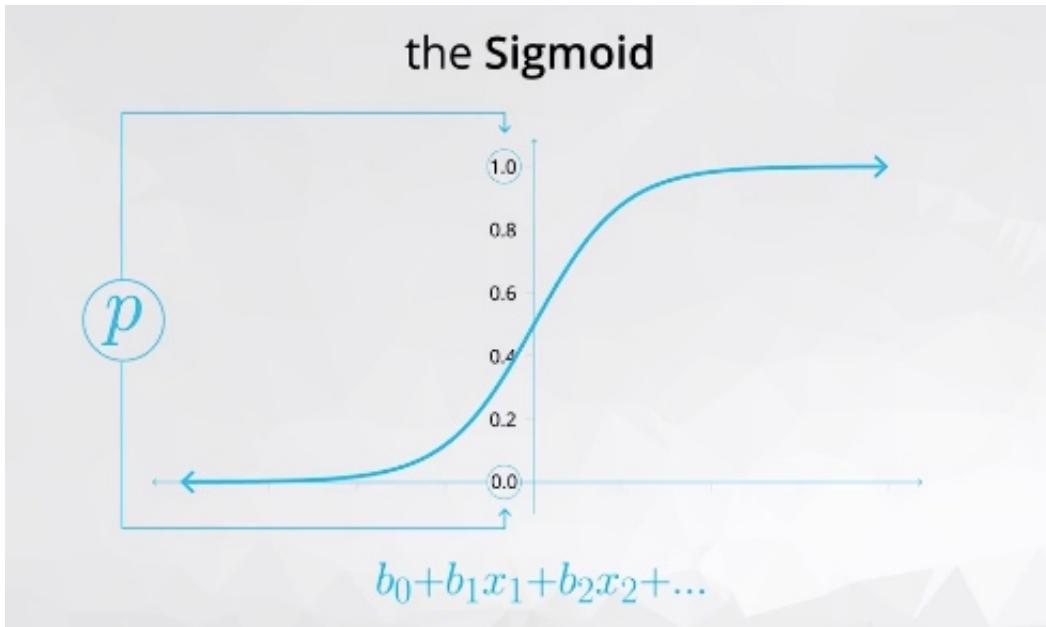
$$\log\left(\frac{p}{1-p}\right) = b_0 + b_1x_1 + b_2x_2 + \dots$$

通过转换，我们就得出了 $p$ 的表达式，就是Sigmoid函数。他的作用就是根据现行回归直线得到值，并将概率控制在0和1之间：

$$\log\left(\frac{p}{1-p}\right) = b_0 + b_1x_1 + b_2x_2 + \dots$$

⊕ ⊖ ⊗ ⊚

$$p = \frac{e^{b_0+b_1x_1+b_2x_2+\dots}}{1 + e^{b_0+b_1x_1+b_2x_2+\dots}}$$



其实在代码中实现起来非常类似，就是换了一个拟合回归的方式：

```
df['intercept'] = 1
logit_mode = sm.Logit(df['fraud'], df[['intercept', 'duration']])
results = logit_mode.fit()
results.summary()
```

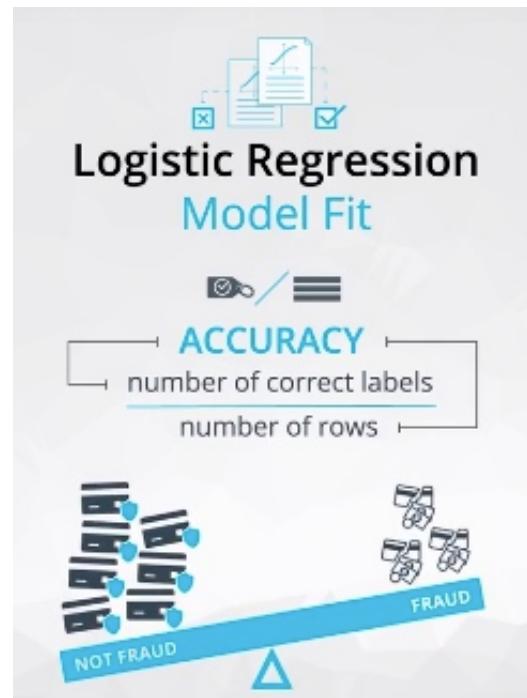
结果也是很熟悉的（截距的解释比较绕，大家可以先不看）：

|           | coef    | std err | t     | P> t  | [0.025 | 0.975] |
|-----------|---------|---------|-------|-------|--------|--------|
| intercept | 9.8709  | 1.944   | 5.078 | 0.000 | 6.061  | 13.691 |
| duration  | -1.4637 | 0.290   | 0.290 | 0.000 | -2.033 | -0.894 |
| weekday   | 2.5465  | 0.904   | 0.904 | 0.000 | 0.774  | 4.319  |

These suggest that both duration and weekday are statistically significant in predicting if a transaction is fraud or not

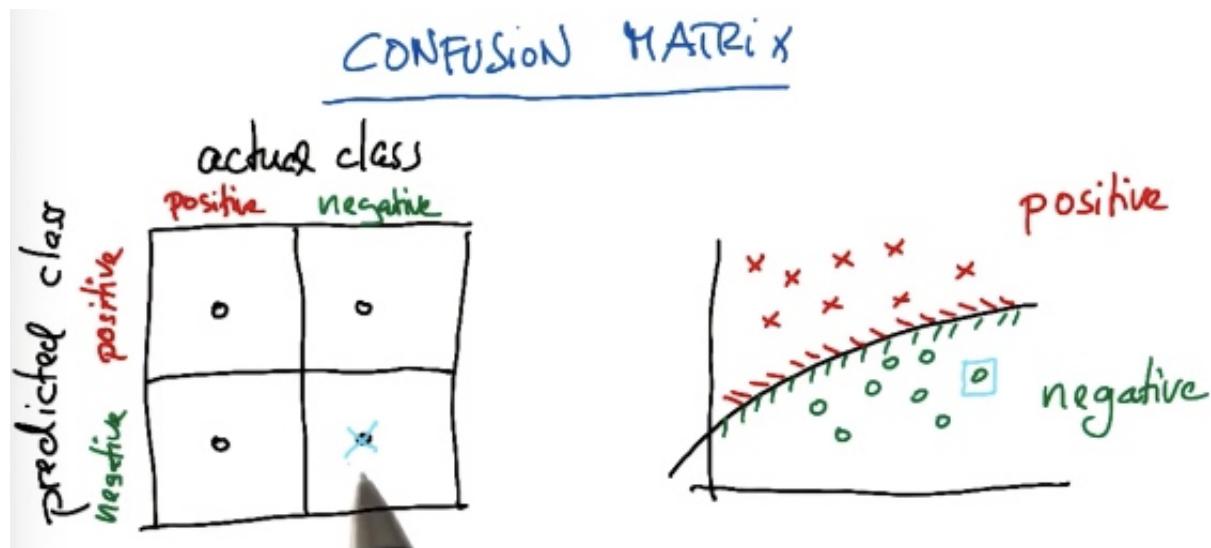
## \*// 模型诊断

完成了上面的工作以后，我们就得出了自己的模型，那么怎么检查自己的模型是不是够厉害呢？我们需要进行**模型诊断 Model Diagnostics**：使用的是**准确率 Accuracy**这个指标，其实就是在标记出来的行（数据）除以总行数，但是这种情况当分类数据偏差太大（欺诈的很少，没欺诈的很多）的时候效果不太好：

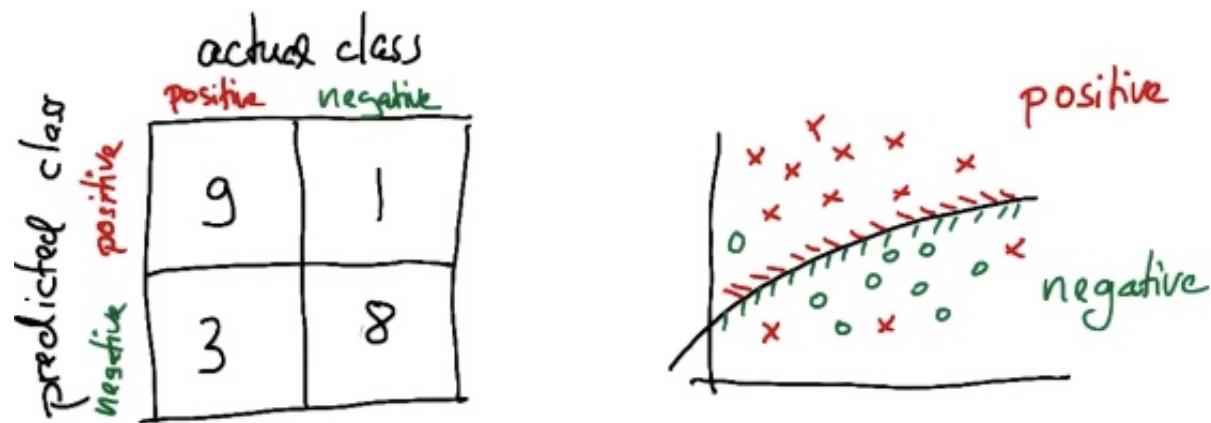


接下来扩展一下几个在衡量时候的概念（选学）：

**混淆矩阵**：对于实际分类和算法分类结果的描述矩阵（就是之前癌症检验，真假阳性、真假阴性那种矩阵）：



这正的用下来之后，就可以发现模型跑出来的误差了：



之后就跳转到主成分分析 PCA的概念了，从这之后就不用看了，课程还特别卡位的提供了机器学习入门课程的链接（免费的不要说是我告诉你的）：<https://cn.udacity.com/course/intro-to-machine-learning--ud120>

## \*\*\*// 精确率和召回率

最后再八卦一个概念，精确率和召回率（我们继续用校长图像识别的这个例子）：

- **查全率（召回率）Recall**是指用你算法判断出来的结果，和真实的比是否都找出来了。（就是看跳出来的n个图片，和真实m个图片的比例关系） - **查准率（精确率）Precision**：是指用你算法判断出来的结果，是否为真。（就是看跳出来的n个图片，是不是都对的，没挑出来的不管）

|                   | PREDICTED |    |    |     |    |    |      |
|-------------------|-----------|----|----|-----|----|----|------|
| Ariel Sharon      | [ 13      | 4  | 1  | 1   | 0  | 0  | 1 ]  |
| Colin Powell      | [ 0       | 55 | 0  | 8   | 0  | 0  | 0 ]  |
| Donald Rumsfeld   | [ 0       | 1  | 25 | 8   | 0  | 0  | 2 ]  |
| George W Bush     | [ 0       | 3  | 0  | 123 | 0  | 0  | 1 ]  |
| Gerhard Schroeder | [ 0       | 1  | 0  | 7   | 14 | 0  | 4 ]  |
| Hugo Chavez       | [ 0       | 3  | 0  | 2   | 1  | 10 | 0 ]  |
| Tony Blair        | [ 0       | 0  | 1  | 7   | 0  | 0  | 26 ] |

$$\text{RECALL} ("Hugo\ Chavez") = \frac{1}{16}$$

$$\text{PRECISION} ("Hugo\ Chavez") = 1$$

这里的代码实现大家看一下就行了，我们将会在项目讲解中解释。课程中扩展了3个内容（选学）：

- sklearn的逻辑回归：[http://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
- sklearn的混乱矩阵：[http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion\\_matrix.html](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)
- 为什么我们要将数据划分为训练集和测试集：<https://info.salford-systems.com/blog/bid/337783/Why-Data-Scientists-Split-Data-into-Train-and-Test>

## \*// 统计学到机器学习

最后的一部分，课程中讲解了统计学到机器学习的意义：

- **统计学 Statistics:** Determine relationships and understand the driving mechanisms. Are relationships due to chance?
- **机器学习（监督） Machine Learning(Supervised):** Work to predict as well as possible. Often without regard to why it works well.
- 变化其实是我们减少了对输入的关注，增加了对输出的关注；更加注重预测结果的正确性，而不是输入的正确性。
- sklearn这个最流行的机器学习库的特点之一是模型诊断的最后内容是利用交叉验证。

Title: W11 项目指导 [项目：分析A/B测试结果2/3]

Tags: 数据分析初级，实战项目

# W11 项目指导 [项目：分析A/B测试结果2/3]

---

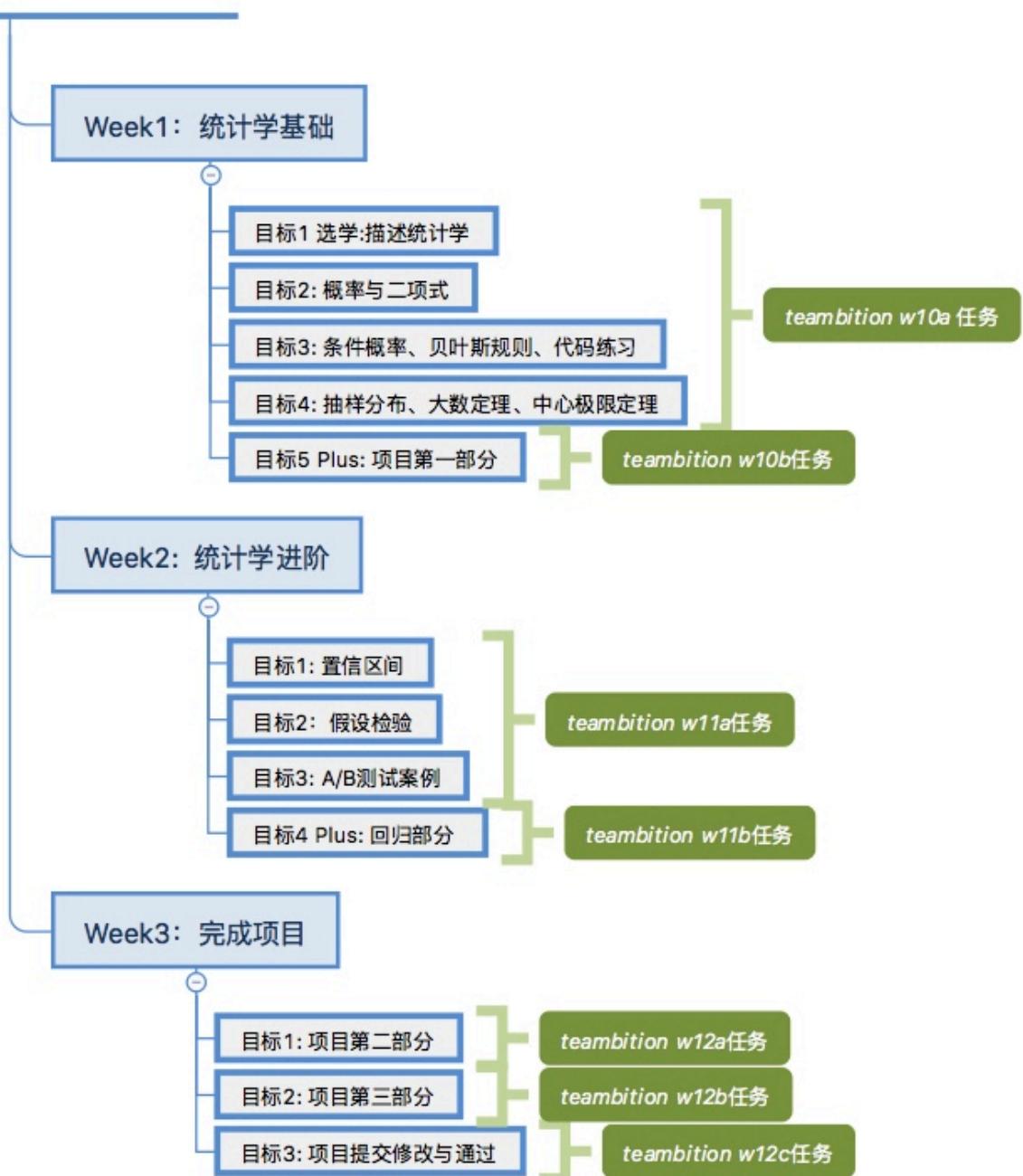
- [W11 项目指导 \[项目：分析A/B测试结果2/3\]](#)
- [/学习地图/](#)
  - / 项目路径
- [/目标5/ 分析A/B测试结果项目 - II A/B测试](#)
  - / 1.写出假设检验
    - // 优雅的输入假设
  - / 2.相关数值计算
    - // a,b,c,d 数量计算
    - // e,f,g 生成new/old\_page\_converted和他们的差
    - // h 把上述模拟进行1万次
    - // i 绘制直方图
    - // j 比较刚才模拟的数据和真实的数据差距
    - // k 对上面结果做描述
    - // l 使用置信区间进行A/B假设的判定1
    - // m 使用置信区间进行A/B假设的判定2
    - // n 使用置信区间进行A/B假设的判定3
- [/目标6/ 分析A/B测试结果项目 - III 回归分析法之一](#)
  - / 1.使用回归来完成A/B测试的计算
    - // a 使用逻辑回归（因为是算分类信息）
    - // b 使用statsmodels来进行计算1
    - // c 使用statsmodels来进行计算2
    - // d 根据计算得出结论
    - // e 给出你的答案
    - // f 增加更多因素到回归模型中
    - // g 我们还有个国家的数据，来分析一下吧
    - // h 那我我们把那么我们来吧国家和新旧页面联合考察下
- [/彩蛋/](#)
  - / 课外书
  - / 小艾怎么办系列

# /学习地图/

本周是统计学项目的Part2，是项目4的推论统计学的扩展部分。目标1、2、3讲解置信区间、假设检验、和一个A/B测试案例。目标4作为Plus发布，讲解了回归的相关知识。最后部分是对项目内容的指导，请先自己完成，卡住了较长时间后可以参考本导学的相应内容。

## / 项目路径

### 分析 A/B 测试结果



## /目标5/ 分析A/B测试结果项目 - II A/B测

# 试

本部分将会指导项目的后两个部分，大家加油（第一部分在week10导学中有讲解）。

## / 1.写出假设检验

此处考察的是能否理解假设检验的内容，根据后面2.问题描述中的提示，零假设为新旧页面的转化率相同。

### // 优雅的输入假设

首先呢是写假设检验，根据本周第一个导学文件，直接在项目空间中markdown cell这样写（如果不想这么麻烦直接用英文大小写描述也没有问题的，z中间包住的内容就是latex的公式格式内容）：

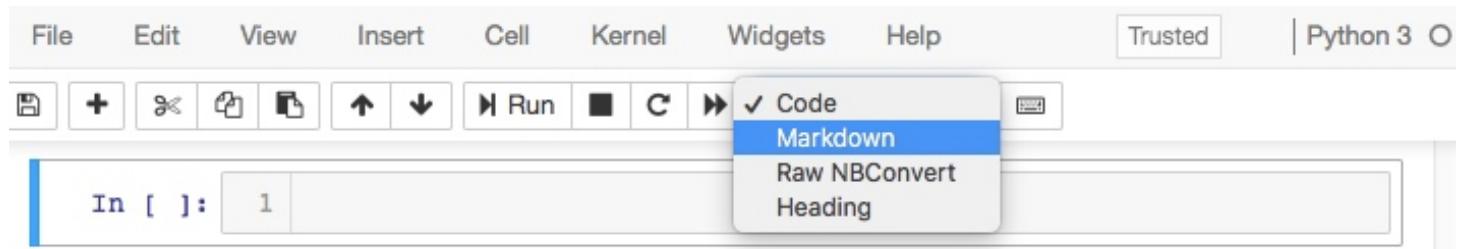
```
1 # 这个是错的呦!
2 $ H_0:p_{new}-p_{old}=0 $
3
4 $ H_1:p_{new}-p_{old}\neq0 $
```

在开始做的时候，我先按照这部分2.中的描述做了解答，但实际上这里回答1的问题，是要按照1中的描述：**如果你想假定旧的页面效果更好，除非新的页面在类型I错误率为5%的情况下才能证明效果更好。**来写假设（提示下，这次是单尾假设了），大于等于和小于等于的代码是：

```
1 $ \leq $
2 $ \geq $
```

样子就是这样的： $\leq \geq$

切换markdown cell的方式是点中，之后点击上面的选项：



选择之后前面的 In[] 会消失，直接编辑就行。Latex如何输入大家直接搜索，比如“latex 不等号”就能找到很多例子。其实是一种数学公式的输入方法，非常强大，更重要的是非常优雅漂亮！更改之后的markdown cell的内容是这样的：

```
1 $ H_0:p_{new}-p_{old}=0 $
2
3 $ H_1:p_{new}-p_{old}\neq0 $
```

shift回车之后就会渲染（这是错的例子，请自行解决）：

$$H_0 : p_{new} - p_{old} = 0$$

$$H_1 : p_{new} - p_{old} \neq 0$$

## / 2. 相关数值计算

### // a,b,c,d 数量计算

这里和上一节基本一样，再按照空回答一遍就好了，不过要注意，此处的假设已经变了：“**假定在零假设中，不管是新页面还是旧页面都具有等于成功率的“真”成功率**”

```
1 # “假定在零假设中，不管是新页面还是旧页面都具有等于成功率的“真”成功率”
2 # 因为这句话，此处模拟0假设的时候要按照两个一样去计算
3 # 都是收集样本的转化率
4 p_new = df2.converted.mean()
5 round(p_new, 4)
```

```
1 # 新旧相同的，自己解决
2 p_old = 你的代码
```

本部分得出的几个变量：

- $p_{new} = p_{old} = 0.1196$  转化率
- $n_{new} = 145310$  新页面的数量
- $n_{old} = 145274$  旧页面的数量 **需要注意：上面的4个值是收集来的所有页面信息的统计数。**

### // e,f,g 生成new/old\_pageConverted和他们的差

这里先拆解下要求：

- 使用 $p_{new}$ 转化率模拟 $n_{new}$ 交易
- 将模拟的结果存为`new_pageConverted`
- 此处的意义应该是根据算出的 $p_{new}$ 再模拟（每次运行都不一样）抽样分布

```

1 # 模拟么要使用random.choice
2 # ()中的参数解释
3 # 参数1 - 2, 表明随机选择的是0, 1
4 # 参数2 - nnew, 就是上面的145310, 代表抽取多少次
5 # 此处可以这么理解: 既然是要模拟nnew的情况, 那么抽取次数要和nnew相同
6 # 参数3 - p = [1-p_new,p_new], 就是对应参数1的两个值, 各自取的概率
7 # rnew就是上面得出的0.1188, 对应的是1的取值
8 new_page_converted = np.random.choice([0, 1], nnew, p = [1-p_new,p_new])
9 # 输出是个array
10 type(new_page_converted)
11 # 结果是numpy.ndarray, 内容是由0与1组成的array
12 # 直接求均值就是转化率了
13 new_page_conr = round(new_page_converted.mean(),6)
14 new_page_conr

```

把另一个算出来之后, 就可以把这两个值相减了 (g的任务), 得出p\_compare, 请同学们自己完成 (结果会有随机性)

## // h 把上述模拟进行1万次

**重要提示:** 此处是进行10000遍模拟, 大家可以在开始做的时候改为range (1000) 改为1000次。等文件都写完了之后再改为10000, 之后提交报告, 可以缩短此处代码的运行时间。

```

1 # 建立p_diffs存放每次模拟抽样后的转化率
2 p_diffs = []
3 # 将上面的代码循环1万遍, 得出这1万遍模拟新旧转化率的差异
4 for i in range(10000):
5     new_page_converted = np.random.choice([0,1], nnew, p = [1-p_new,p_new])
6     old_page_converted = np.random.choice([0,1], nold, p = [1-p_old,p_old])
7     p_diffs.append(new_page_converted.mean()-old_page_converted.mean())

```

## // i 绘制直方图

熟悉吧, 这里就是将这1万次的模拟画出图来, 明显的正态分布 (中心极限定理还记得不?)

```

1 # 直接把p_diffs画图 (列表输入, arrays输入都ok的)
2 plt.hist(p_diffs);

```

## // j 比较刚才模拟的数据和真实的数据差距

```
1 # 下面的3行其实是1行代码，为了不超过79个字符的规则，使用 \ 分割到下一行
2 # 这是的obs_diffs是真实的值（从数据中推算出来的）
3 # obs_diffs就是根据所有数据计算出的新页面转化率和旧页面转化率的差
4 obs_diffs = \
5 df2.query('landing_page == "new_page"')['converted'].mean() - \
6 df2.query('landing_page == "old_page"')['converted'].mean()
7 # 得出obs
8 (p_diffs > obs_diffs).mean()
```

## // k 对上面结果做描述

- 此处应该转换为markdown框作答
- 根据上面的答案要明确回答是否有统计显著性

## // i 使用置信区间进行A/B假设的判定1

```
1 # 首先引入统计学模块
2 import statsmodels.api as sm
3 # 将转化了的都筛选出来
4 # 使用的是.query 里面是筛选的条件
5 con_1 = df2.query('converted == "1"')
6 # 测出新旧页面数量和总数量
7 convert_old = con_1.query('landing_page == "old_page"').shape[0]
8 convert_new = con_1.query('landing_page == "new_page"').shape[0]
9 n_old = df2.query('landing_page == "old_page"').shape[0]
10 n_new = df2.query('landing_page == "new_page"').shape[0]
```

## // m 使用置信区间进行A/B假设的判定2

```
1 # 使用.stat.proportions_ztest计算z_score
2 # 同时会输出p (p的概念就是：如果零假设为真，观察到统计量的概率
3 # 其中的3个参数
4 # 参数1 - 新旧页面对比的转化率 [convert_old, convert_new]
5 # 参数2 - 新旧页面的数量 [n_old, n_new]
6 # 参数3 - 计算参数延用就好
7 z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new],
8 [n_old, n_new], alternative='smaller')
9 # 最后输出 z_score,p_value
z_score, p_value
```

输出的值是：(1.3109241984234394, 0.9050583127590245)

## // n 使用置信区间进行A/B假设的判定3

这里要理解怎样判断z-score是否显著，简单的说就是根据0.95的置信度，双尾这两个情况去查表。根据表中查处具有统计学显著意义的临界值（上面条件是1.96），和计算出的值做比较。

- 根据得出的z-score和p-value的值：
- 检查z表格，在 $\alpha = 0.05$ （双尾）时的值是1.96（大于才是，所以1.31无法否认零假设）
- p值为0.905无法否认0假设（和0.95比较）
- 其实因为n很大（采样），只看z的就可以确定了
- 所以认为新旧页面的转化率无区别，结论与j和k中的结果一致

到此，A/B测试的第二部分已经完成，其实已经得出了结论（后面的一节是用机器学习中线性回归的方法再判断一遍）。简单的说就是根据我们收集到的样本，我们假设的p值为0.905，I类错误率超过0.05（ $1 - 0.0905 > 0.05$ ）

# /目标6/ 分析A/B测试结果项目 - III 回归分析法之一

## / 1. 使用回归来完成A/B测试的计算

### // a 使用逻辑回归（因为是算分类信息）

此处使用的是逻辑回归，如果是对数值做估算就要使用线性回归和多元线性回归。

### // b 使用statsmodels来进行计算1

使用的就是week11-guide1中的虚拟变量的方式，并且调用statsmodels来完成计算，先用虚拟变量的方式建立0/1数值的列：

```
1 # 根据课程中的.get_dummies一列的信息转换为两列1/0标识的信息
2 # 截距为1
3 df2['intercept'] = 1
4 # 此处get_dummies是将landing_page的值拆分出来，将两列值赋值给df2的新列
5 # 生成了2个新列 new_page old_page(用于回答后续问题)
6 df2[['new_page', 'old_page']] = pd.get_dummies(df2['landing_page'])
7 # 将ab_page的两个字段应设成1/0 (1表示是新页面, 0表示的是旧页边)
8 # 之后就可以用这个列和converted做比较了 (后续代码框)
```

```

9 # 此处使用了.map(之前导学文件中有)把后面的()中的逐一对每个元素执行
10 # 执行的是lambda x: 行内函数
11 # 要干的就是把x变成1, 如果x=treatment; 否则变成0
12 df2['ab_page'] = df['group'].map(lambda x: '1' if x=="treatment" else "0"
13 ")
df2.head()

```

|   | user_id | timestamp                  | group     | landing_page | converted | intercept | new_page | old_page | ab_page |
|---|---------|----------------------------|-----------|--------------|-----------|-----------|----------|----------|---------|
| 0 | 851104  | 2017-01-21 22:11:48.556739 | control   | old_page     | 0         | 1         | 0        | 1        | 0       |
| 1 | 804228  | 2017-01-12 08:01:45.159739 | control   | old_page     | 0         | 1         | 0        | 1        | 0       |
| 2 | 661590  | 2017-01-11 16:55:06.154213 | treatment | new_page     | 0         | 1         | 1        | 0        | 1       |
| 3 | 853541  | 2017-01-08 18:28:03.143765 | treatment | new_page     | 0         | 1         | 1        | 0        | 1       |
| 4 | 864975  | 2017-01-21 01:52:26.210827 | control   | old_page     | 1         | 1         | 0        | 1        | 0       |

## // c 使用statsmodels来进行计算2

```

1 # 定义模型
2 # 使用sm.Logit(因为是新旧分类ab_page列)
3 # 因变量是converted
4 # 自变量是ab_page(intercept默认要有的)
5 logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
6 # 对模型进行适配
7 results = logit_mod.fit()

```

## // d 根据计算得出结论

```

1 results.summary()
2 # 使用summary2的话P为4位小数

```

根据结果，我们可以看出这里检查的是converted，使用的模型是Logit，结论是ab\_page的p\_value是0.190：

| Dep. Variable: | converted        | No. Observations: | 290584      |       |        |        |
|----------------|------------------|-------------------|-------------|-------|--------|--------|
| Model:         | Logit            | Df Residuals:     | 290582      |       |        |        |
| Method:        | MLE              | Df Model:         | 1           |       |        |        |
| Date:          | Sat, 25 Aug 2018 | Pseudo R-squ.:    | 8.077e-06   |       |        |        |
| Time:          | 16:21:49         | Log-Likelihood:   | -1.0639e+05 |       |        |        |
| converged:     | True             | LL-Null:          | -1.0639e+05 |       |        |        |
|                |                  | LLR p-value:      | 0.1899      |       |        |        |
|                | coef             | std err           | z           | P> z  | [0.025 | 0.975] |
| intercept      | -1.9888          | 0.008             | -246.669    | 0.000 | -2.005 | -1.973 |
| ab_page        | -0.0150          | 0.011             | -1.311      | 0.190 | -0.037 | 0.007  |

此处还有一点要回答和上面II部分的p\_value为什么是不一样的，提示考虑下：

- II 中的p\_value计算是单尾还是双尾的
- 此处的p\_value计算是单尾还是双尾的（结果中可是P>|z|呦）

## // e 给出你的答案

p值都有了，就可以给出答案了，这一步该你自己走了！此处注意要回答II部分和本部分p值不同的原因，提示是单尾检验还是双尾检验。

## // f 增加更多因素到回归模型中

更多因素会带来更准确训练结果，但是太多了的话又会造成对样本很管用（解决欠拟合问题），但对新的数据就没那么适用了（产生过拟合问题），其实就是找个平衡点就好了。

## // g 我们还有个国家的数据，来分析一下吧

数据集中还有个countries.csv，把他读进来，附加在df2上(这半句代码如下)：

```

1 # 此处代码如果多次运行会报错（因为已经join过了）
2 # cell - run all 就好了
3 # 使用.join把country文件中的数据根据user_id进行整合
4 # .set_index(), on是按照那一列进行整合
5 # 这里两个数据都是user_id所以前后是一样的
6 df2 = df2.join(country.set_index('user_id'), on='user_id')
7 df2.head()

```

之后一样进行虚拟变量的转换：

```

1 | # 同样，使用虚拟变量进行变换
2 | df2[['CA','UK','US']] = pd.get_dummies(df2['country'])

```

再之后就再次进行逻辑回归了：

```

1 | # 那么再来一遍Logit分析
2 | # 后面的参数选择的是CA、UK（根据虚拟变量那节的解释可以代表3个变量的趋势）
3 | # 默认的斜率 = 1
4 | df2['intercept'] = 1
5 | # 描述模型
6 | lm = sm.Logit(df2['converted'],df2[['intercept','CA','UK']])
7 | # 进行适配
8 | result_country = lm.fit()
9 | # 输出结果
10 | result_country.summary()

```

| Logit Regression Results |                  |                   |             |       |        |        |
|--------------------------|------------------|-------------------|-------------|-------|--------|--------|
| Dep. Variable:           | converted        | No. Observations: | 290584      |       |        |        |
| Model:                   | Logit            | Df Residuals:     | 290581      |       |        |        |
| Method:                  | MLE              | Df Model:         | 2           |       |        |        |
| Date:                    | Sat, 25 Aug 2018 | Pseudo R-squ.:    | 1.521e-05   |       |        |        |
| Time:                    | 20:07:06         | Log-Likelihood:   | -1.0639e+05 |       |        |        |
| converged:               | True             | LL-Null:          | -1.0639e+05 |       |        |        |
|                          |                  | LLR p-value:      | 0.1984      |       |        |        |
|                          | coef             | std err           | z           | P> z  | [0.025 | 0.975] |
| intercept                | -1.9967          | 0.007             | -292.314    | 0.000 | -2.010 | -1.983 |
| CA                       | -0.0408          | 0.027             | -1.518      | 0.129 | -0.093 | 0.012  |
| UK                       | 0.0099           | 0.013             | 0.746       | 0.456 | -0.016 | 0.036  |

## // h 那我我们把那么我们来吧国家和新旧页面联合考察下

根基提示，我们还是要创建新的列（请按照提示自己完成），之后对新的列做逻辑回归：

```

1 | df3['intercept'] = 1
2 | lm = sm.Logit(df2['converted'],df3[['intercept','ab_page','UK','US','new
3 | _UK','new_US']])
4 | result_country = lm.fit()
result_country.summary()

```

结果如下，大家看看和前面的结论一致不？项目到此完成，大家代码部分可以参考，但一定要理解并且自己写出来！

| <b>Dep. Variable:</b> | converted        | <b>No. Observations:</b> | 290584      |       |        |        |
|-----------------------|------------------|--------------------------|-------------|-------|--------|--------|
| <b>Model:</b>         | Logit            | <b>Df Residuals:</b>     | 290581      |       |        |        |
| <b>Method:</b>        | MLE              | <b>Df Model:</b>         | 2           |       |        |        |
| <b>Date:</b>          | Sun, 26 Aug 2018 | <b>Pseudo R-squ.:</b>    | 1.082e-05   |       |        |        |
| <b>Time:</b>          | 11:18:28         | <b>Log-Likelihood:</b>   | -1.0639e+05 |       |        |        |
| <b>converged:</b>     | True             | <b>LL-Null:</b>          | -1.0639e+05 |       |        |        |
|                       |                  | <b>LLR p-value:</b>      | 0.3164      |       |        |        |
|                       | coef             | std err                  | z           | P> z  | [0.025 | 0.975] |
| intercept             | -1.9926          | 0.008                    | -252.910    | 0.000 | -2.008 | -1.977 |
| new_UK                | 0.0112           | 0.018                    | 0.626       | 0.532 | -0.024 | 0.046  |
| new_US                | -0.0144          | 0.012                    | -1.155      | 0.248 | -0.039 | 0.010  |

## / 彩蛋/

---

## / 课外书

---

课程中有几本相关的书籍，放到下面的链接了[/资料链接/](#)：（目前包括以下内容：）

- SQL必知必会第4版
- 利用Python进行数据分析
- Guide to Numpy
- An Introduction to Statistical Learning
- PythonCheatSheet (ps: cheatsheet就是常用命令的一张纸展示，可以打印出来放桌上，据说可以镇邪)

## / 小艾怎么办系列

---

小ai的软件思维分享：

- 出错了怎么办：<https://www.bilibili.com/video/av27003276>
- 拿到项目一脸懵怎么办：<https://www.bilibili.com/video/av26838723>
- 我懒得敲那么多下键盘怎么办：<https://www.bilibili.com/video/av26801634>

