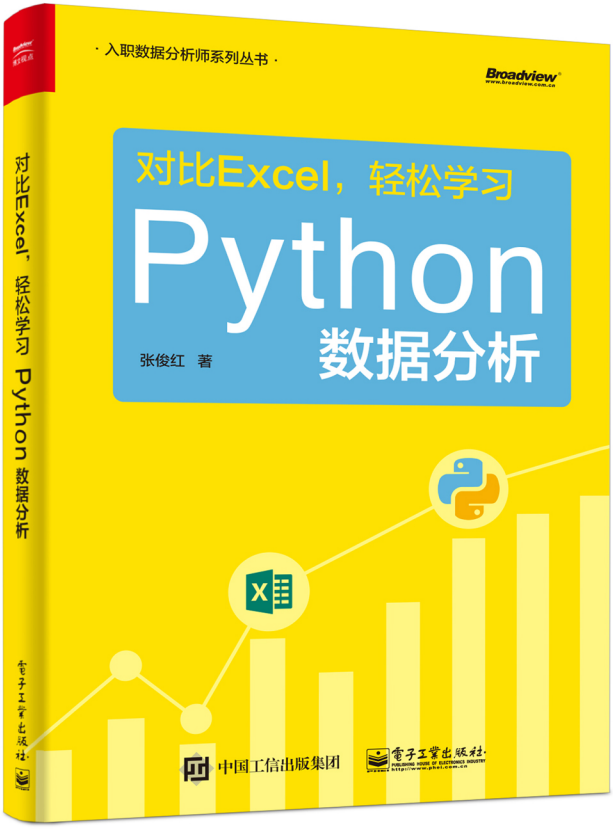

·入职数据分析师系列丛书·
Broadview
对比Excel，轻松学习
Python
数据分析
张俊红 著
中国工信出版集团
電子工業出版社
對比Excel，輕松學習 Python 數據分析
電子工業出版社

内 容 简 介

集Python、Excel、数据分析为一体是本书的一大特色。

本书围绕整个数据分析的常规流程：熟悉工具—明确目的—获取数据—熟悉数据—处理数据—分析数据—得出结论—验证结论—展示结论进行Excel和Python的对比实现，告诉你每一个过程中都会用到什么，过程与过程之间有什么联系。本书既可以作为系统学习数据分析操作流程的说明书，也可以作为一本数据分析师案头必备的实操工具书。

本书通过对比Excel功能操作去学习Python的代码实现，而不是直接学习Python代码，大大降低了学习门槛，消除了读者对代码的恐惧心理。适合刚入行的数据分析师，也适合对Excel比较熟练的数据分析师，以及从事其他岗位想提高工作效率的职场人。

图书在版编目（CIP）数据

对比 Excel，轻松学习 Python 数据分析 / 张俊红著. —北京：电子工业出版社，2019.2
（入职数据分析师系列）

ISBN 978-7-121-35793-0

Ⅰ. ①对… Ⅱ. ①张… Ⅲ. ①软件工具－程序设计 Ⅳ. ①TP311.561

中国版本图书馆 CIP 数据核字(2018)第 279763 号

策划编辑：张慧敏
责任编辑：汪达文
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：720×1000　1/16　印张：17.75　字数：365 千字　彩插：1
版　　次：2019 年 2 月第 1 版
印　　次：2019 年 2 月第 1 次印刷
印　　数：3000 册　　定价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

序　言

有幸收到张俊红的做序邀请，我非常高兴。

从 PC 时代到移动互联网时代一路走来，每个人都感受到了数据爆炸性的增长，以及其中蕴含的巨大价值。

从 PC 时代开始，我们用键盘、扫描仪等设备使信息数据化。在移动互联网时代，智能手机通过摄像头、GPS、陀螺仪等各种传感器将我们的位置、行动轨迹、行为偏好，甚至情绪等信息数据化。截至 2000 年，全人类存储了大约 12EB 的数据，要知道 1PB=1024TB，而 1EB=1024PB。但是到了 2011 年，一年所产生的数据就高达 1.82ZB（注：1ZB=1024EB），数据已经变成了一种人造的“新能源”。

在商业领域，从信息到商品，从商品到服务，越来越多我们熟悉的事物被标准的数据所度量。无论是在线广告的精准营销，还是电子商务的个性化推荐，又或者是互联网金融的人脸识别，互联网的每一次效率提升都依赖于对传统信息、物品，甚至人的数据化。

在使用数据进行效率变革及商业化的道路上，Excel 和 Python 扮演了关键的角色，它们帮助数据分析师高效地从海量数据中发现问题，验证假设，搭建模型，预测未来。

作为一本数据分析的专业书籍，作者从数据采集、清洗、抽取，以及数据可视化等多个角度介绍了日常工作中数据分析的标准路径。通过对比 Excel 与 Python 在数据处理过程中的操作步骤，详细说明了 Excel 与 Python 间的差异，以及用 Python 进行数据分析的方法。

虽与作者素未谋面，但是对于 Python 在处理海量数据和建模上的高效性与便捷性，以及 Python 在机器学习中的重要性，我们的观点是一致的。同时我们也相信对于数据分析从业者来说，掌握一种用于数据处理的编程语言是非常必要的，而从 Excel 到 Python 的学习方法则是一条学好数据分析的“捷径”。

王彦平

（网名“蓝鲸”，电子书《从 Excel 到 Python——数据分析进阶指南》《从 Excel 到 R——数据分析进阶指南》《从 Excel 到 SQL——数据分析进阶指南》的作者）

2019 年 1 月 8 日

前　言

为什么要写这本书

本书既是一本数据分析的书，也是一本 Excel 数据分析的书，同时还是一本 Python 数据分析的书。在互联网上，无论是搜索数据分析，还是搜索 Excel 数据分析，亦或是搜索 Python 数据分析，我们都可以找到很多相关的图书。既然已经有这么多同类题材的书了，为什么我还要写呢？因为在我准备写这本书时，还没有一本把数据分析、Excel 数据分析、Python 数据分析这三者结合在一起的书。

为什么我要把它们结合在一起写呢？那是因为，我认为这三者是一个数据分析师必备的技能，而且这三者本身也是一个有机统一体。数据分析让你知道怎么分析以及分析什么；Excel 和 Python 是你在分析过程中会用到的两个工具。

为什么要学习 Python

既然 Python 在数据分析领域是一个和 Excel 类似的数据分析工具，二者实现的功能都一样，为什么还要学 Python，把 Excel 学好不就行了吗？我认为学习 Python 的主要原因有以下几点。

1．在处理大量数据时，Python 的效率高于 Excel

当数据量很小的时候，Excel 和 Python 的处理速度基本上差不多，但是当数据量较大或者公式嵌套太多时，Excel 就会变得很慢，这个时候怎么办呢？我们可以使用 Python，Python 对于海量数据的处理效果要明显优于 Excel。用 Vlookup 函数做一个实验，两个大小均为 23MB 的表（6 万行数据），在未作任何处理、没有任何公式嵌套之前，Excel 中直接在一个表中用 Vlookup 函数获取另一个表的数据需要 20 秒（我的计算机性能参数是 I7、8GB 内存、256GB 固态硬盘），配置稍微差点的计算机可能打开这个表都很难。但是用 Python 实现上述过程只需要 580 毫秒，即 0.58 秒，是 Excel 效率的 34 倍。

2．Python 可以轻松实现自动化

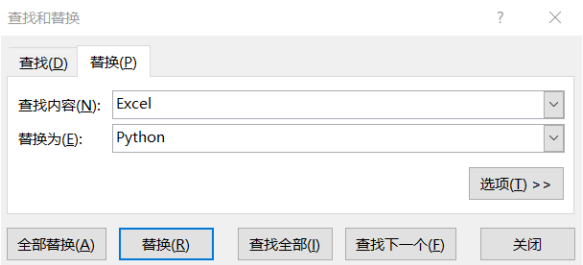
你可能会说 Excel 的 VBA 也可以自动化，但是 VBA 主要还是基于 Excel 内部的自动化，一些其他方面的自动化 VBA 就做不了，比如你要针对本地某一文件夹下面的文件名进行批量修改，VBA 就不能实现，但是 Python 可以。

3．Python 可用来做算法模型

虽然你是做数据分析的，但是一些基础的算法模型还是有必要掌握的，Python 可以让你在懂一些基础的算法原理的情况下就能搭建一些模型，比如你可以使用聚类算法搭建一个模型去对用户进行分类。

为什么要对比 Excel 学习 Python

Python 虽然是一门编程语言，但是在数据分析领域实现的功能和 Excel 的基本功能一样，而 Excel 又是大家比较熟悉、容易上手的软件，所以可以通过 Excel 数据分析去对比学习 Python 数据分析。对于同一个功能，本书告诉你在 Excel 中怎么做，并告诉你对应到 Python 中是什么样的代码。例如数值替换，即把一个值替换成另一个值，对把“Excel”替换成“Python”这一要求，在 Excel 中可以通过鼠标点选实现，如下图所示。



在 Python 中则通过具体的代码实现，如下所示。

```
df.replace("Excel","Python")    # 表示将表 df 中的 Excel 替换成 Python
```

本书将数据分析过程中涉及的每一个操作都按这种方式对照讲解，让你从熟悉的 Excel 操作中去学习对应的 Python 实现，而不是直接学习 Python 代码，大大降低了学习门槛，消除了大家对代码的恐惧心理。这也是本书的一大特色，也是我为什么要写本书的最主要原因，就是希望帮助你不再惧怕代码，让你可以像学 Excel 数据分析一样，轻松学习 Python 数据分析。

本书的学习建议

要想完全掌握一项技能，你必须系统学习它，知道它的前因后果。本书不是孤立地讲 Excel 或者 Python 中的操作，而是围绕整个数据分析的常规流程：熟悉工具—明确目的—获取数据—熟悉数据—处理数据—分析数据—得出结论—验证结论—展示结论，告诉你每一个过程都会用到什么操作，这些操作用 Excel 和 Python 分别怎么实现。这样一本书既是系统学习数据分析流程操作的说明书，也是数据分析师案头必备的实操工具书。

大家在读第一遍的时候不用记住所有函数，你是记不住的，即使你记住了，如果在工作中不用，那么很快就会忘记。正确的学习方式应该是，先弄清楚一名数据分析师在日常工作中对工具都会有什么需求（当然了，本书的顺序是按照数据分析的常规分析流程来写的），希望工具帮助你达到什么样的目的，罗列好需求以后，再去研究工具的使用方法。比如，要删除重复值，就要明确用 Excel 如何实现，用 Python 又该如何实现，两种工具在实现方式上有什么异同，这样对比次数多了以后，在遇到问题时，你自然而然就能用最快的速度选出最适合的工具了。

数据分析一定是先有想法然后考虑如何用工具实现，而不是刚开始就陷入记忆工具的使用方法中。

本书写了什么

本书分为三篇。

入门篇：主要讲数据分析的一些基础知识，介绍数据分析是什么，为什么要做数据分析，数据分析究竟在分析什么，以及数据分析的常规流程。

实践篇：围绕数据分析的整个流程，分别介绍每一个步骤中的操作，这些操作用 Excel 如何实现，用 Python 又如何实现。本篇内容主要包括：Python 环境配置、Python 基础知识、数据源的获取、数据概览、数据预处理、数值操作、数据运算、时间序列、数据分组、数据透视表、结果文件导出、数据可视化等。

进阶篇：介绍几个实战案例，让你体会一下在实际业务中如何使用 Python。具体来说，进阶篇的内容主要包括，利用 Python 实现报表自动化、自动发送电子邮件，以及在不同业务场景中的案例分析。此外，还补充介绍了 NumPy 数组的一些常用方法。

本书适合谁

本书主要适合以下人群。

- Excel 已经用得熟练，想学习 Python 来丰富自己技能的数据分析师。

- 刚入行对 Excel 和 Python 都不精通的数据分析师。
- 其他常用 Excel 却想通过学习 Python 提高工作效率的人。

Python 虽然是一门编程语言，但是它并不难学，不仅不难学，而且很容易上手，这也是 Python 深受广大数据从业者喜爱的原因之一，因此大家在学习 Python 之前首先在心里告诉自己一句话，那就是 Python 并没有那么难。

致谢

感谢我的父母，是他们给了我受教育的机会，才有了今天的我。

感谢我的公众号的读者朋友们，如果不是他们，那么我可能不会坚持撰写技术文章，更不会有这本书。

感谢慧敏让我意识到写书的意义，从而创作本书，感谢电子工业出版社为这本书忙碌的所有人。

感谢我的女朋友，在写书的这段日子里，我几乎把所有的业余时间全用在了写作上，很少陪她，但她还是一直鼓励我，支持我。

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **提交勘误**：您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/35793

目　录

入门篇

实践篇

进阶篇

第 1 章

数据分析基础

1.1 数据分析是什么

数据分析是指利用合适的工具在统计学理论的支撑下，对数据进行一定程度的预处理，然后结合具体业务分析数据，帮助相关业务部门监控、定位、分析、解决问题，从而帮助企业高效决策，提高经营效率，发现业务机会点，让企业获得持续竞争的优势。

1.2 为什么要做数据分析

在做一件事情之前我们首先得弄清楚为什么要做，或者说做了这件事以后有什么好处，这样我们才能更好地坚持下去。

啤酒和尿布的问题大家应该都听过，如果没有数据分析，相信大家是怎么也不会发现买尿布的人一般也会顺带买啤酒，现在各大电商网站都会卖各种套餐，相关商品搭配销售能大大提高客单价，增加收益，这些套餐的搭配都是基于历史用户购买数据得出来的。如果没有数据分析，可能很难想到要把商品搭配销售，或者不知道该怎么搭配。

谷歌曾经推出一款名为“谷歌流感趋势”的产品，这款产品能够很好地预测流感这种传染疾病的发生时间。这款产品预测的原理就是，某一段时间内某些关键词的检索量会异常高，谷歌通过分析这些检索量高的关键词发现，这些关键词，比如咳嗽、头痛、发烧都是一些感冒/流感症状，当有许多人都搜索这些关键词时，说明这次并非一般性感冒，极有可能是一场带有传染性的流感，这个时候就可以及时采取一些措施来防止流感的扩散。

虽然谷歌流感趋势预测最终以失败告终，但是这个产品的整体思路是值的借鉴参考的。感兴趣的同学，可以上网查一下谷歌流感趋势预测的始末。

数据分析可以把隐藏在大量数据背后的信息提炼出来，总结出数据的内在规律。代替了以前那种拍脑袋、靠经验做决策的做法，因此越来越多的企业重视数据分析。具体来说，数据分析在企业日常经营分析中有三大作用，即现状分析、原因分析、预测分析。

1.2.1 现状分析

现状分析可以告诉你业务在过去发生了什么，具体体现在两个方面。

第一，告诉你现阶段的整体运营情况，通过各个关键指标的表现情况来衡量企业的运营状况，掌握企业目前的发展趋势。

第二，告诉你企业各项业务的构成，通常公司的业务并不是单一的，而是由很多分支业务构成的，通过现状分析可以让你了解企业各项分支业务的发展及变动情况，对企业运营状况有更深入的了解。

现状分析一般通过日常报表来实现，如日报、周报、月报等形式。

例如，电商网站日报中的现状分析会包括订单数、新增用户数、活跃率、留存率等指标同比、环比上涨/下跌了多少。如果将公司的业务划分为华北、东北、华中、华东、华南、西南、西北几个片区，通过现状分析，你可以很清楚地知道哪些区域做得比较好，哪些区域做得比较差。

1.2.2 原因分析

原因分析可以告诉你某一现状为什么会存在。

经过现状分析，我们对企业的运营情况有了基本了解，知道哪些指标呈上升趋势，哪些指标呈下降趋势，或者是哪些业务做得好，哪些做得不好。但是我们还不知道那些做得好的业务为什么会做得好，做得差的业务的原因又是什么？找原因的过程就是原因分析。

原因分析一般通过专题分析来完成，根据企业运营情况选择针对某一现状进行原因分析。

例如，在某一天的电商网站日报中，某件商品销量突然大增，那么就需要针对这件销量突然增加的商品做专题分析，看看是什么原因促成了商品销量大增。

1.2.3 预测分析

预测分析会告诉你未来可能发生什么。

在了解企业运营状况以后，有时还需要对企业未来发展趋势做出预测，为制定企业运营目标及策略提供有效的参考与决策依据，以保证企业的可持续健康发展。

预测分析一般是通过专题分析来完成的，通常在制订企业季度、年度计划时进行。

例如，通过上述的原因分析，我们就可以有针对性地实施一些策略。比如通过原因分析，我们得知在台风来临之际面包的销量会大增，那么我们在下次台风来临之前就应该多准备一些面包，同时为了获得更多的销量做一系列准备。

1.3 数据分析究竟在分析什么

数据分析的重点在分析，而不在工具，那么我们究竟该分析什么呢？

1.3.1 总体概览指标

总体概览指标又称统计绝对数，是反映某一数据指标的整体规模大小，总量多少的指标。

例如，当日销售额为 60 万元，当日订单量为 2 万，购买人数是 1.5 万，这些都是概览指标，用来反映某个时间段内某项业务的某些指标的绝对量。

我们把经常关注的总体概览指标称为关键性指标，这些指标的数值将会直接决定公司的盈利情况。

1.3.2 对比性指标

对比性指标是说明现象之间数量对比关系的指标，常见的就是同比、环比、差这几个指标。

同比是指相邻时间段内某一共同时间点上指标的对比，环比就是相邻时间段内指标的对比；差就是两个时间段内的指标直接做差，差的绝对值就是两个时间段内指标的变化量。

例如，2018 年和 2017 年是相邻时间段，那么 2018 年的第 26 周和 2017 年的第 26 周之间的对比就是同比，而 2018 年的第 26 周和第 25 周的对比就是环比。

1.3.3 集中趋势指标

集中趋势指标是用来反映某一现象在一定时间段内所达到的一般水平，通常用平均指标来表示。平均指标分为数值平均和位置平均。例如，某地的平均工资就是一个集中趋势指标。

数值平均是统计数列中所有数值平均的结果，有普通平均数和加权平均数两种。普通平均的所有数值的权重都是 1，而加权平均中不同数值的权重是不一样的，在算平均值时不同数值要乘以不同的权重。

假如你要算一年中每月的月平均销量，这个时候一般就用数值平均，直接把 12

个月的销量相加除以 12 即可。

假如你要算一个人的平均信用得分情况，由于影响信用得分的因素有多个，而且不同因素的权重占比是不一样的，这个时候就需要使用加权平均。

位置平均是基于某个特殊位置上的数或者普遍出现的数，即用出现次数最多的数值来作为这一系列数值的整体一般水平。基于位置的指标最常用的就是中位数，基于出现次数最多的指标就是众数。

众数是一系列数值中出现次数最多的数值，是总体中最普遍的值，因此可以用来代表一般水平。如果数据可以分为多组，则为每组找出一个众数。注意，众数只有在总体内单位充分多时才有意义。

中位数是将一系列值中的每一个值按照从小到大顺序排列，处于中间位置的数值就是中位数。因为处于中间位置，有一半变量值大于该值，一半小于该值，所以可以用这样的中等水平来表示整体的一般水平。

1.3.4 离散程度指标

离散程度指标是用来表示总体分布的离散（波动）情况的指标，如果这个指标较大，则说明数据波动比较大，反之则说明数据相对比较稳定。

全距（又称极差）、方差、标准差等几个指标用于衡量数值的离散情况。

全距：由于平均数让我们确定一批数据的中心，但是无法知道数据的变动情况，因此引入全距。全距的计算方法是用数据集中最大数（上界）减去数据集中最小数（下界）。

全距存在的问题主要有两方面。

- 问题 1，容易受异常值影响。
- 问题 2，全距只表示了数据的宽度，没有描述清楚数据上下界之间的分布形态。

对于问题 1 我们引入四分位数的概念。四分位数将一些数值从小到大排列，然后一分为四，最小的四分位数为下四分位数，最大的四分位数为上四分位数，中间的四分位数为中位数。

对于问题 2 我们引入了方差和标准差两个概念来衡量数据的分散性。

方差是每个数值与均值距离的平方的平均值，方差越小说明各数值与均值之间的差距越小，数值越稳定。

标准差是方差的开方，表示数值与均值距离的平均值。

1.3.5 相关性指标

上面提到的几个维度是对数据整体的情况进行描述，但是我们有的时候想看一下

数据整体内的变量之间存在什么关系，一个变化时会引起另一个怎么变化，我们把用来反映这种关系的指标叫做相关系数，相关系数常用 r 来表示。

$$r(X,Y)=\frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}[X]\mathrm{Var}[Y]}}$$

其中，Cov(X,Y)为 X 与 Y 的协方差，Var[X]为 X 的方差，Var[Y]为 Y 的方差。

关于相关系数需要注意以下几点。

- 相关系数 r 的范围为[-1，1]。
- r 的绝对值越大，表示相关性越强。
- r 的正负代表相关性的方向，正代表正相关，负代表负相关。

1.3.6 相关关系与因果关系

相关关系不等于因果关系，相关关系只能说明两件事情有关联，而因果关系是说明一件事情导致了另一件事情的发生，不要把这两种关系混淆使用。

例如，啤酒和尿布是具有相关关系的，但是不具有因果关系；而流感疾病和关键词检索量上涨是具有因果关系的。

在实际业务中会遇到很多相关关系，但是具有相关关系的两者不一定有因果关系，一定要注意区分。

1.4 数据分析的常规流程

我们再来回顾一下数据分析的概念，数据分析是借助合适的工具去帮助公司发现数据背后隐藏的信息，对这些隐藏的信息进行挖掘，从而促进业务发展。基于此，可以将数据分析分为以下几个步骤。

1.4.1 熟悉工具

数据分析是利用合适的工具和合适的理论挖掘隐藏在数据背后的信息，因此数据分析的第一步就是要熟悉工具。工欲善其事，必先利其器，只有熟练使用工具，才能更好地处理数据、分析数据。

1.4.2 明确目的

做任何事情都要目的明确，数据分析也一样，首先要明确数据分析的目的，即希望通过数据分析得出什么。例如，希望通过数据分析发现流失用户都有哪些特征，希望通过数据分析找到销量上涨的原因。

1.4.3 获取数据

目的明确后我们就要获取数据，在获取数据之前还需要明确以下几点。

- 需要什么指标。
- 需要什么时间段的数据。
- 这些数据都存在哪个数据库或哪个表中。
- 怎么提取，是自己写 Sql 还是可以直接从 ERP 系统中下载。

1.4.4 熟悉数据

拿到数据以后，我们要去熟悉数据，熟悉数据就是看一下有多少数据，这些数据都，是类别型还是数值型的；每个指标大概有哪些值，这些数据能不能满足我们的需求，如果不够，那么还需要哪些数据。

获取数据和熟悉数据是一个双向的过程，当你熟悉完数据以后发现当前数据维度不够，那就需要重新获取；当你获取到新的数据以后，需要再去熟悉，所以获取数据和熟悉数据会贯穿在整个数据分析过程中。

1.4.5 处理数据

获取到的数据是原始数据，这些数据中一般会有一些特殊数据，我们需要对这些数据进行提前处理，常见的特殊数据主要有以下几种。

- 异常数据。
- 重复数据。
- 缺失数据。
- 测试数据。

对于重复数据、测试数据我们一般都是做删除处理的。

对于缺失数据，如果缺失比例高于 30%，那么我们会选择放弃这个指标，即做删除处理。而对于缺失比例低于 30%的指标，我们一般进行填充处理，即使用 0、均值或者众数等进行填充。

对于异常数据，需要结合具体业务进行处理，如果你是一个电商平台的数据分析师，你要找出平台上的刷单商户，那么异常值就是你要重点研究的对象了；假如你要

分析用户的年龄，那么一些大于 100 或者是小于 0 的数据，就要选择删除。

1.4.6 分析数据

分析数据主要围绕上节介绍的数据分析指标展开。在分析过程中经常采用的一个方法就是下钻法，例如当我们发现某一天的销量突然上涨/下滑时，我们会去看是哪个地区的销量上涨/下滑，进而再看哪个品类、哪个产品的销量出现上涨/下滑，层层下钻，最后找到问题产生的真正原因。

1.4.7 得出结论

通过分析数据，我们就可以得出结论。

1.4.8 验证结论

有的时候即使通过数据分析出来的结论也不一定成立，所以我们要把数据分析和实际业务相联系，去验证结论是否正确。

例如，做新媒体数据分析，你通过分析发现情感类文章的点赞量、转发量更高，这只是你的分析结论，但是这个结论正确吗？你可以再写几篇情感类文章验证一下。

1.4.9 展示结论

我们在分析出结论，并且结论得到验证以后就可以把这个结论分享给相关人员，例如领导或者业务人员。这个时候就需要考虑如何展示结论，以什么样的形式展现，这就要用到数据可视化了。

1.5 数据分析工具 Excel 与 Python

数据分析都是围绕常规数据分析流程进行的，在这个流程中，我们需要选择合适的工具对数据进行操作。

例如，导入外部数据。如果用 Excel 实现，那么直接单击菜单栏中的数据选项卡（如下图所示），然后根据外部数据的格式选择不同格式的数据选项即可实现。

如果用 Python 实现，那么需要编写如下代码进行数据导入，即你要根据文件的格式选择不同的代码，来导入不同格式的本地文件。

```
#导入.csv 文件
data = pd.read_csv(filepath + "test.csv",encoding="gbk")

#导入.xlsx 文件
data = pd.read_excel(filepath + "test.xlsx",encoding="gbk")

#导入.txt 文件
data = pd.read_table(filepath + "test.txt",encoding="gbk")

#导入数据库文件
data = pd.read_sql("select * from test", con)
```

通过这个简单的例子，我们可以看到，同一个操作可以使用不同的工具实现，不同工具的实现方式是不一样的，Excel 是通过鼠标点选的方式来操作数据，而 Python 需要通过具体的代码来操作数据。虽然两者的操作方式是不一样的，但都可以达到导入外部数据这一操作的目的。Python 在数据分析领域只不过是和 Excel 类似的一个数据分析工具而已。

本书的编写都是按照这种方式进行的，针对数据分析中的每一个操作，分别用 Excel 和 Pyhon 对比实现。

第 5 章

淘米洗菜——数据预处理

从菜市场买来的菜，总有一些不太好的，所以把菜买回来以后要先做一遍预处理，把那些不太好的部分扔掉。现实中大部分的数据都类似于菜市场的菜品，拿到以后都要先做一次预处理。

常见的不规整数据主要有缺失数据、重复数据、异常数据几种，在开始正式的数据分析之前，我们需要先把这些不太规整的数据处理掉。

5.1 缺失值处理

缺失值就是由某些原因导致部分数据为空，对于为空的这部分数据我们一般有两种处理方式，一种是删除，即把含有缺失值的数据删除；另一种是填充，即把缺失的那部分数据用某个值代替。

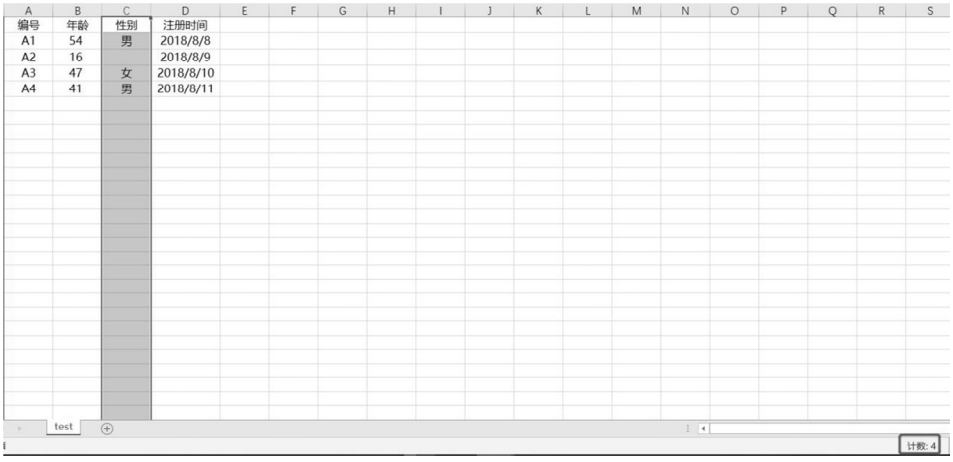
5.1.1 缺失值查看

对缺失值进行处理，首先要把缺失值找出来，也就是查看哪列有缺失值。

Excel 实现

在 Excel 中我们先选中一列没有缺失值的数据，看一下这一列数据共有多少个，然后把其他列的计数与这一列进行对比，小于这一列数据个数的就代表有缺失值，差值就是缺失个数。

下图中非缺失值列的数据计数为 5，性别这一列计数为 4，这就表示性别这一列有 1 个缺失值。



编号	年龄	性别	注册时间
A1	54	男	2018/8/8
A2	16		2018/8/9
A3	47	女	2018/8/10
A4	41	男	2018/8/11

如果想看整个数据表中每列数据的缺失情况，则要挨个选中每一列去判断该列是否有缺失值。

如果数据不是特别多，你想看到具体是哪个单元格缺失，则可以利用定位条件（按快捷键 Ctrl+G 可弹出该对话框）查找。在定位条件对话框中选择空值，单击确定就会把所有的空值选中，如下图所示。

通过定位条件把缺失值选出来的结果，如下图所示。

编号	年龄	性别	注册时间
A1	54	男	2018/8/8
A2	16		2018/8/9
A4	41	男	2018/8/11

Python 实现

在 Python 中直接调用 info()方法就会返回每一列的缺失情况。关于 info()方法我

们在前面就用过，但是没有说明这个方法可以判断数据的缺失情况。

```
>>>df
    编号  年龄    性别     注册时间
0   A1    54     男       2018/8/8
1   A2    16     NaN      2018/8/9
2   A3    47     女       2018/8/10
3   A4    41     男       2018/8/11
>>>df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
编号      4 non-null object
年龄      4 non-null int64
性别      3 non-null object
注册时间    4 non-null object
dtypes: int64(1), object(3)
memory usage: 208.0+ bytes
```

Python 中缺失值一般用 NaN 表示，从用 info()方法的结果来看，性别这一列是 3 non-null object，表示性别这一列有 3 个非 null 值，而其他列有 4 个非 null 值，说明性别这一列有 1 个 null 值。

我们还可以用 isnull()方法来判断哪个值是缺失值，如果是缺失值则返回 True，如果不是缺失值则返回 False。

```
>>>df
    编号  年龄  性别      注册时间
0   A1    54    男       2018/8/8
1   A2    16    NaN      2018/8/9
2   A3    47    女       2018/8/10
3   A4    41    男       2018/8/11
>>>df.isnull()
    编号    年龄     性别    注册时间
0   False   False   False   False
1   False   False   True    False
2   False   False   False   False
3   False   False   False   False
```

5.1.2 缺失值删除

缺失值分为两种，一种是一行中某个字段是缺失值；另一种是一行中的字段全部为缺失值，即为一个空白行。

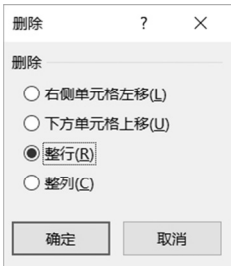
Excel 实现

在 Excel 中，这两种缺失值都可以通过在定位条件（按快捷键 Ctrl+G 可弹出该对

话框）对话框中选择空值找到。

这样含有缺失值的部分就会被选中，包括某个具体的单元格及一整行，然后单击鼠标右键在弹出的删除对话框中选择删除整行选项，并单击确定按钮即可实现删除。

编号	年龄	性别	注册时间
A1	54	男	2018/8/8
A2	16		2018/8/9
A4	41	男	2018/8/11



Python 实现

在 Python 中，我们利用的是 dropna()方法，dropna()方法默认删除含有缺失值的行，也就是只要某一行有缺失值就把这一行删除。

```
>>>df
    编号  年龄  性别  注册时间
0   A1    54    男   2018/8/8
1   A2    16    NaN  2018/8/9
2   A3    47    女   2018/8/10
3   A4    41    男   2018/8/11
>>>df.dropna()
    编号  年龄  性别  注册时间
0   A1    54    男   2018/8/8
2   A3    47    女   2018/8/10
3   A4    41    男   2018/8/11
```

运行 dropna()方法以后，删除含有 NaN 值的行，返回删除后的数据。

如果想删除空白行，只要给 dropna()方法传入一个参数 how = all 即可，这样就会只删除那些全为空值的行，不全为空值的行就不会被删除。

```
>>>df
    编号  年龄  性别  注册时间
0   A1    54    男   2018/8/8
1   A2    16    NaN  2018/8/9
2   NaN   NaN   NaN  NaN
3   A4    41    男   2018/8/11
>>>df.dropna(how = "all")
    编号  年龄  性别  注册时间
```

```
0   A1    54    男   2018/8/8
1   A2    16    NaN  2018/8/9
3   A4    41    男   2018/8/11
```

上表第二行中只有性别这个字段是空值，所以在利用 dropna(how = "all")的时候并没有删除第二行，只是把全为 NaN 值的第三行删掉了。

5.1.3 缺失值填充

上面介绍了缺失值的删除，但是数据是宝贵的，一般情况下只要数据缺失比例不是过高（不大于 30%），尽量别删除，而是选择填充。

Excel 实现

在 Excel 中，缺失值的填充和缺失值删除一样，利用的也是定位条件，先把缺失值找到，然后在第一个缺失值的单元格中输入要填充的值，最常用的就是用 0 填充，输入以后按 Ctrl+Enter 组合键就可以对所有缺失值进行填充。

缺失值填充前后的对比如下图所示。

Before

编号	年龄	性别	注册时间
A1	54	男	2018/8/8
A2	16		2018/8/9
A3		女	2018/8/10
A4	41	男	2018/8/11

After

编号	年龄	性别	注册时间
A1	54	男	2018/8/8
A2	16	0	2018/8/9
A3	0	女	2018/8/10
A4	41	男	2018/8/11

年龄用数字填充合适，但是性别用数字填充就不太合适，那么可不可以分开填充呢？答案是可以的，选中要填充的那一列，按照填充全部数据的方式进行填充即可，只不过要填充几列，需要执行几次操作。

Before

编号	年龄	性别	注册时间
A1	54	男	2018/8/8
A2	16		2018/8/9
A3		女	2018/8/10
A4	41	男	2018/8/11

After

编号	年龄	性别	注册时间
A1	54	男	2018/8/8
A2	16	男	2018/8/9
A3	37	女	2018/8/10
A4	41	男	2018/8/11

上图是填充前后的对比，年龄这一列我们用平均值填充，性别这一列我们用众数填充。

除了用 0 填充、平均值填充、众数（大多数）填充，还有向前填充（即用缺失值的前一个非缺失值填充，比如上例中编号 A3 对应的缺失年龄的前一个非缺失值就是 16）、向后填充（与向前填充对应）等方式。

Python 实现

在 Python 中，我们利用的 fillna()方法对数据表中的所有缺失值进行填充，在 fillna

后面的括号中输入要填充的值即可。

```
>>>df
    编号  年龄  性别  注册时间
0   A1    54    男   2018/8/8
1   A2    16    NaN  2018/8/9
2   NaN   NaN   NaN  NaN
3   A4    41    男   2018/8/11
>>>df.fillna(0)
    编号  年龄  性别  注册时间
0   A1    54    男   2018/8/8
1   A2    16    0    2018/8/9
2   0     0     0    0
3   A4    41    男   2018/8/11
```

在 Python 中我们也可以按不同列填充，只要在 fillna()方法的括号中指明列名即可。

```
>>>df
    编号  年龄  性别  注册时间
0   A1    54    男   2018/8/8
1   A2    16    NaN  2018/8/9
2   A3    NaN   女   2018/8/10
3   A4    41    男   2018/8/11
>>>df.fillna({"性别":"男"})#对性别进行填充
    编号  年龄  性别  注册时间
0   A1    54    男   2018/8/8
1   A2    16    男   2018/8/9
2   A3    NaN   女   2018/8/10
3   A4    41    男   2018/8/11
```

上面代码中只针对性别这一列进行了填充，其他列未进行任何更改。

也可以同时对多列填充不同的值：

```
#分别对性别和年龄进行填充
>>>df.fillna({"性别":"男","年龄":"30"})
    编号  年龄  性别  注册时间
0   A1    54    男   2018/8/8
1   A2    16    男   2018/8/9
2   A3    30    女   2018/8/10
3   A4    41    男   2018/8/11
```

5.2 重复值处理

重复数据就是同样的记录有多条，对于这样的数据我们一般做删除处理。

假设你是一名数据分析师，你的主要工作是分析公司的销售情况，现有公司 2018

年 8 月的销售明细（已知一条明细对应一笔成交记录），你想看一下 8 月整体成交量是多少，最简单的方式就是看一下有多少条成交明细。但是这里可能会有重复的成交记录存在，所以要先删除重复项。

Excel 实现

在 Excel 中依次单击菜单栏中的数据>数据工具>删除重复值，就可以删除重复数据了，如下图所示。

删除前后的对比如下图所示。

Before

订单编号	客户姓名	唯一识别码	成交时间
A1	张通	101	2018/8/8
A2	李谷	102	2018/8/9
A3	孙凤	103	2018/8/10
A3	孙凤	103	2018/8/10
A4	赵恒	104	2018/8/11
A5	赵恒	104	2018/8/12

After

订单编号	客户姓名	唯一识别码	成交时间
A1	张通	101	2018/8/8
A2	李谷	102	2018/8/9
A3	孙凤	103	2018/8/10
A4	赵恒	104	2018/8/11
A5	赵恒	104	2018/8/12

Excel 的删除重复值默认针对所有值进行重复值判断，比如有订单编号、客户姓名、唯一识别码（类似于身份证号）、成交时间这四个字段，Excel 会判断这四个字段是否都相等，只有都相等时才会删除，且保留第一个（行）值。

你知道了公司 7 月成交明细以后，你想看一下 8 月总共有多少成交客户，且每个客户在 8 月首次成交的日期。

查看客户数量只需要按客户的唯一识别码进行去重就可以了。Excel 默认是全选，我们可以取消全选，选择唯一识别码进行去重，这样只要唯一识别码重复就会被删除，如下图所示。

因为 Excel 默认会保留第一条记录，而我们又想要获取每个客户的较早成交日期，

所以我们需要先对时间进行升序排列，让较早的日期排在前面，这样在删除的时候就会保留较早的成交日期。

删除前后的对比如下图所示。

Before

订单编号	客户姓名	唯一识别码	成交时间
A1	张通	101	2018/8/8
A2	李谷	102	2018/8/9
A3	孙凤	103	2018/8/10
A3	孙凤	103	2018/8/10
A4	赵恒	104	2018/8/11
A5	赵恒	104	2018/8/12

After

订单编号	客户姓名	唯一识别码	成交时间
A1	张通	101	2018/8/8
A2	李谷	102	2018/8/9
A3	孙凤	103	2018/8/10
A4	赵恒	104	2018/8/11

Python 实现

在 Python 中我们利用 drop_duplicates()方法，该方法默认对所有值进行重复值判断，且默认保留第一个（行）值。

```
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
0   A1          张通      101        2018-08-08
1   A2          李谷      102        2018-08-09
2   A3          孙凤      103        2018-08-10
3   A3          孙凤      103        2018-08-10
4   A4          赵恒      104        2018-08-11
5   A5          赵恒      104        2018-08-12
>>>df.drop_duplicates()
    订单编号    客户姓名  唯一识别码   成交时间
0   A1          张通      101        2018-08-08
1   A2          李谷      102        2018-08-09
2   A3          孙凤      103        2018-08-10
4   A4          赵恒      104        2018-08-11
5   A5          赵恒      104        2018-08-12
```

上面的代码是针对所有字段进行的重复值判断，我们同样也可以只针对某一列或某几列进行重复值删除的判断，只需要在 drop_duplicates()方法中指明要判断的列名即可。

```
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
0   A1          张通      101        2018-08-08
1   A2          李谷      102        2018-08-09
2   A3          孙凤      103        2018-08-10
3   A3          孙凤      103        2018-08-10
4   A4          赵恒      104        2018-08-11
5   A5          赵恒      104        2018-08-12
>>>df.drop_duplicates(subset = "唯一识别码")
    订单编号    客户姓名  唯一识别码   成交时间
```

```
0   A1          张通       101          2018-08-08
1   A2          李谷       102          2018-08-09
2   A3          孙凤       103          2018-08-10
4   A4          赵恒       104          2018-08-11
```

也可以利用多列去重，只需要把多个列名以列表的形式传给参数 subset 即可。比如按姓名和唯一识别码去重。

```
>>>df.drop_duplicates(subset = ["姓名","唯一识别码"])
    订单编号    客户姓名  唯一识别码    成交时间
0   A1          张通       101          2018-08-08
1   A2          李谷       102          2018-08-09
2   A3          孙凤       103          2018-08-10
4   A4          赵恒       104          2018-08-11
```

还可以自定义删除重复项时保留哪个，默认保留第一个，也可以设置保留最后一个，或者全部不保留。通过传入参数 keep 进行设置，参数 keep 默认值是 first，即保留第一个值；也可以是 last，保留最后一个值；还可以是 False，即把重复值全部删除。

```
#保留最后一个重复值
>>>df.drop_duplicates(subset = ["姓名","唯一识别码"],
   keep = "last")
    订单编号    客户姓名  唯一识别码    成交时间
0   A1          张通       101          2018-08-08
1   A2          李谷       102          2018-08-09
3   A3          孙凤       103          2018-08-10
5   A5          赵恒       104          2018-08-12
#不保留任何重复值
>>>df.drop_duplicates(subset = ["姓名","唯一识别码"],
   keep = False)
    订单编号    客户姓名  唯一识别码    成交时间
0   A1          张通       101          2018-08-08
1   A2          李谷       102          2018-08-09
```

5.3 异常值的检测与处理

异常值就是相比正常数据而言过高或过低的数据，比如一个人的年龄是 0 岁或者 300 岁都算是一个异常值，因为这和实际情况差距过大。

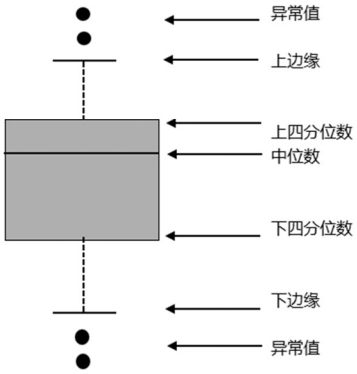
5.3.1 异常值检测

要处理异常值首先要检测，也就是发现异常值，发现异常值的方式主要有以下三种。

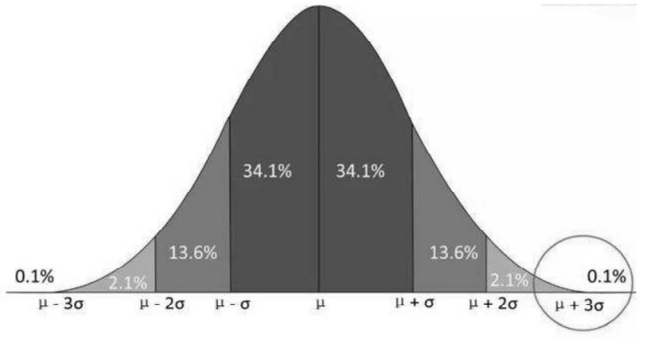
- 根据业务经验划定不同指标的正常范围，超过该范围的值算作异常值。

- 通过绘制箱形图，把大于（小于）箱形图上边缘（下边缘）的点称为异常值。
- 如果数据服从正态分布，则可以利用 3σ 原则；如果一个数值与平均值之间的偏差超过 3 倍标准差，那么我们就认为这个值是异常值。

箱形图如下图所示，关于箱形图的绘制方法我们会在可视化章节介绍。



下图为正太分布图，我们把大于 μ+3σ 的值称为异常值。



5.3.2 异常值处理

对于异常值一般有以下几种处理方式。

- 最常用的处理方式就是删除。
- 把异常值当作缺失值来填充。
- 把异常值当作特殊情况，研究异常值出现的原因。

Excel 实现

在 Excel 中，删除异常值只要通过筛选把异常值对应的行找出来，然后单击鼠标右键选择删除行即可。

对异常值进行填充，其实就是对异常值进行替换，同样通过筛选功能把异常值先找出来，然后把这些异常值替换成要填充的值即可。

Python 实现

在 Python 中，删除异常值用到的方法和 Excel 中的方法原理类似，Python 中是通过过滤的方法对异常值进行删除。比如 df 表中有年龄这个指标，要把年龄大于 200 的值删掉，你可以通过筛选把年龄不大于 200 的筛出来，筛出来的部分就是删除缺失值以后的新表。

对异常值进行填充，就是对异常值进行替换，利用 replace()方法可以对特定的值进行替换。

关于数据筛选和数据替换会在接下来的章节介绍。

5.4 数据类型转换

5.4.1 数据类型

Excel 实现

在 Excel 中常用的数据类型就是在菜单栏中数字选项下面的几种，你也可以选择其他数据格式，如下图所示。

在 Excel 中只要选中某一列就可以在菜单栏看到这一列的数据类型。

当选中成交时间这一列时，菜单栏中就会显示日期，表示成交时间这一列的数据类型是日期格式，如下图所示。

G	H	I	J
订单编号	客户姓名	唯一识别码	成交时间
A1	张通	101	2018/8/8
A2	李谷	102	2018/8/9
A3	孙凤	103	2018/8/10
A4	赵恒	104	2018/8/11

Python 实现

Pandas 不像 Excel 分得那么详细，主要有 6 种数据类型，如下表所示。

类　型	说　明
int	整型数，即整数
float	浮点数，即含有小数点的数
object	Python 对象类型，用 O 表示
string_	字符串类型，经常用 S 表示，S10 表示长度为 10 的字符串
unicode_	固定长度的 unicode 类型，跟字符串定义方式一样
datetime64[ns]	表示时间格式

在 Python 中，不仅可以用 info()方法获取每一列的数据类型，还可以通过 dtype 方法来获取某一列的数据类型。

```
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
0   A1         张通      101        2018-08-08
1   A2         李谷      102        2018-08-09
2   A3         孙凤      103        2018-08-10
3   A3         孙凤      103        2018-08-10
4   A4         赵恒      104        2018-08-11
5   A5         赵恒      104        2018-08-12
>>>df["订单编号"].dtype#查看订单编号这一列的数据类型
dtype('O')
>>>df["唯一识别码"].dtype#查看唯一识别码这一列的数据类型
dtype("int64")
```

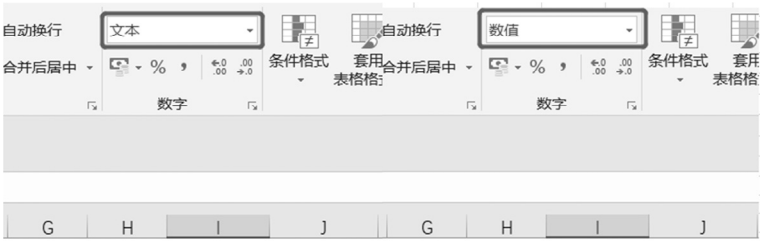
5.4.2　类型转换

我们在前面说过，不同数据类型的数据可以做的事情是不一样的，所以我们需要对数据进行类型转化，把数据转换为我们需要的类型。

Excel 实现

在 Excel 中如果想更改某一列的数据类型，只要选中这一列，然后在数字菜单栏中通过下拉菜单选择你要转换的目标类型。

下图就是将文本类型的数据转换成数值类型的数据，数值类型数据默认为两位小数，也可以设置成其他位数。



Before

订单编号	客户姓名	唯一识别码	成交时间
A1	张通	101	2018/8/8
A2	李谷	102	2018/8/9
A3	孙凤	103	2018/8/10
A4	赵恒	104	2018/8/11

After

订单编号	客户姓名	唯一识别码	成交时间
A1	张通	101.00	2018/8/8
A2	李谷	102.00	2018/8/9
A3	孙凤	103.00	2018/8/10
A4	赵恒	104.00	2018/8/11

Python 实现

在 Python 中，我们利用 astype()方法对数据类型进行转换，astype 后面的括号里指明要转换的目标类型即可。

```
>>>df
    订单编号     客户姓名  唯一识别码    成交时间
0   A1          张通      101         2018-08-08
1   A2          李谷      102         2018-08-09
2   A3          孙凤      103         2018-08-10
3   A3          孙凤      103         2018-08-10
4   A4          赵恒      104         2018-08-11
5   A5          赵恒      104         2018-08-12
>>>df["唯一识别码"].dtype()#查看唯一识别码这一列的数据类型
dtype('int64')
>>>df["唯一识别码"].astype("float64")#将唯一识别码从 int 类型转换为 float
类型
0    101.0
1    102.0
2    103.0
3    103.0
4    104.0
5    104.0
```

5.5 索引设置

索引是查找数据的依据，设置索引的目的是便于我们查找数据。举个例子，你逛超市买了很多食材，回到家以后要把它们放在冰箱里，放的过程其实就是一个建立索引的过程，比如蔬菜放在冷藏室，肉类放在冷冻室，这样在找的时候就能很快找到。

5.5.1 为无索引表添加索引

有的表没有索引，这时要给这类表加一个索引。

Excel 实现

在 Excel 中，一般都是有索引的，如果没索引数据看起来会很乱，当然也会有例外，数据表就是没有索引的。这个时候插入一行一列就是为表添加索引。

添加索引前后的对比如下图所示，序号列为行索引，表头名称为列索引。

Before

A1	张通	101	2018/8/8
A2	李谷	102	2018/8/9
A3	孙凤	103	2018/8/10
A4	赵恒	104	2018/8/11
A5	赵恒	104	2018/8/12

After

序号	订单编号	客户姓名	唯一识别码	成交时间
1	A1	张通	101	2018/8/8
2	A2	李谷	102	2018/8/9
3	A3	孙凤	103	2018/8/10
4	A4	赵恒	104	2018/8/11
5	A5	赵恒	104	2018/8/12

Python 实现

在 Python 中，如果表没有索引，会默认从 0 开始的自然数做索引，比如下面这样：

```
>>>df
    0          1         2          3
0   A1         张通      101        2018-08-08
1   A2         李谷      102        2018-08-09
2   A3         孙凤      103        2018-08-10
3   A4         赵恒      104        2018-08-11
4   A5         赵恒      104        2018-08-12
```

通过给表 df 的 columns 参数传入列索引值，index 参数传入行索引值达到为无索引表添加索引的目的，具体实现如下：

```
#为表添加列索引
>>>df.columns = ["订单编号","客户姓名","唯一识别码","成交时间"]
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
0   A1         张通      101        2018-08-08
1   A2         李谷      102        2018-08-09
2   A3         孙凤      103        2018-08-10
3   A4         赵恒      104        2018-08-11
4   A5         赵恒      104        2018-08-12
```

```
#为表添加行索引
>>>df.index = [1,2,3,4,5,6]
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
1   A1         张通      101        2018-08-08
2   A2         李谷      102        2018-08-09
3   A3         孙凤      103        2018-08-10
4   A4         赵恒      104        2018-08-11
5   A5         赵恒      104        2018-08-12
```

5.5.2 重新设置索引

重新设置索引，一般指行索引的设置。有的表虽然有索引，但不是我们想要的索引，比如现在有一个表是把序号作为行索引，而我们想要把订单编号作为行索引，该怎么实现呢？

Excel 实现

在 Excel 中重新设置行索引比较简单，你想让哪一列做行索引，直接把这一列拖到第一列的位置即可。

Python 实现

在 Python 中可以利用 set_index()方法重新设置索引列，在 set_index()里指明要用作行索引的列的名称即可。

```
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
1   A1         张通      101        2018-08-08
2   A2         李谷      102        2018-08-09
3   A3         孙凤      103        2018-08-10
4   A4         赵恒      104        2018-08-11
5   A5         赵恒      104        2018-08-12
>>>df.set_index("订单编号")
          客户姓名  唯一识别码   成交时间
订单编号
A1        张通      101        2018-08-08
A2        李谷      102        2018-08-09
A3        孙凤      103        2018-08-10
A4        赵恒      104        2018-08-11
A5        赵恒      104        2018-08-12
```

在重新设置索引时，还可以给 set_index()方法传入两个或多个列名，我们把这种一个表中用多列来做索引的方式称为层次化索引，层次化索引一般用在某一列中含有多个重复值的情况下。层次化索引的例子，如下所示，其中 a、b、c、d 分别有多个重复值。

```
a  1    1
   2    2
   3    3
b  1    4
   2    5
c  3    6
   1    7
d  2    8
   3    9
dtype: int32
```

5.5.3 重命名索引

重命名索引是针对现有索引名进行修改的，就是改名字。

Excel 实现

在 Excel 中重命名索引比较简单，就是直接修改表头名字。

Python 实现

在 Python 中重命名索引，我们利用的是 rename()方法，在 rename 后的括号里指明要修改的行索引及列索引名。

```
#重命名列索引
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
1   A1         张通      101        2018-08-08
2   A2         李谷      102        2018-08-09
3   A3         孙凤      103        2018-08-10
4   A4         赵恒      104        2018-08-11
5   A5         赵恒      104        2018-08-12
>>>df.rename(columns = {"订单编号":"新订单编号",
                        "客户姓名":"新客户姓名"})
    新订单编号   新客户姓名 唯一识别码  成交时间
1   A1          张通       101       2018-08-08
2   A2          李谷       102       2018-08-09
3   A3          孙凤       103       2018-08-10
4   A4          赵恒       104       2018-08-11
5   A5          赵恒       104       2018-08-12
#重命名行索引
>>>df
    订单编号    客户姓名    唯一识别码   成交时间
1   A1          张通       101        2018-08-08
2   A2          李谷       102        2018-08-09
3   A3          孙凤       103        2018-08-10
4   A4          赵恒       104        2018-08-11
5   A5          赵恒       104        2018-08-12
```

```
>>>df.rename(index = {1:"一",
                    2:"二",
                    3:"三"})
    订单编号    客户姓名    唯一识别码    成交时间
一    A1         张通        101         2018-08-08
二    A2         李谷        102         2018-08-09
三    A3         孙凤        103         2018-08-10
4     A4         赵恒        104         2018-08-11
5     A5         赵恒        104         2018-08-12
#同时重命名行索引和列索引
>>>df
    订单编号    客户姓名    唯一识别码    成交时间
1    A1         张通        101         2018-08-08
2    A2         李谷        102         2018-08-09
3    A3         孙凤        103         2018-08-10
4    A4         赵恒        104         2018-08-11
5    A5         赵恒        104         2018-08-12
>>>df.rename(columns = {"订单编号":"新订单编号",
                       "客户姓名":"新客户姓名"}
           index = {1:"一",
                    2:"二",
                    3:"三"})
    新订单编号    新客户姓名    唯一识别码    成交时间
一    A1          张通        101         2018-08-08
二    A2          李谷        102         2018-08-09
三    A3          孙凤        103         2018-08-10
4     A4          赵恒        104         2018-08-11
5     A5          赵恒        104         2018-08-12
```

5.5.4 重置索引

重置索引主要用在层次化索引表中，重置索引是将索引列当作一个 columns 进行返回。

在下图左侧的表中，Z1、Z2 是一个层次化索引，经过重置索引以后，Z1、Z2 这两个索引以 columns 的形式返回，变为常规的两列。

Before

		C1	C2
Z1	**Z2**		
A	a	1	2
	b	3	4
B	a	5	6
	b	7	8

After

Z1	Z2	C1	C2
A	a	1	2
A	b	3	4
B	a	5	6
B	b	7	8

在 Excel 中，我们要进行这种转换，直接通过复制、粘贴、删除等功能就可以实

现，比较简单。我们主要讲一下在 Python 中怎么实现。

在 Python 利用的是 reset_index()方法，reset_index()方法常用的参数如下：

```
reset_index(level=None, drop=False, inplace=False)
```

level 参数用来指定要将层次化索引的第几级别转化为 columns，第一个索引为 0 级，第二个索引为 1 级，默认为全部索引，即默认把索引全部转化为 columns。

drop 参数用来指定是否将原索引删掉，即不作为一个新的 columns，默认为 False，即不删除原索引。

inplace 参数用来指定是否修改原数据表。

```
>>>df
       C1  C2
Z1  Z2
A   a   1   2
    b   3   4
B   a   5   6
    b   7   8
>>>df.reset_index()#默认将全部 index 转化为 columns
    Z1  Z2  C1  C2
0   A   a   1   2
1   A   b   3   4
2   B   a   5   6
3   B   b   7   8
>>>df.reset_index(level = 0)#将第 0 级索引转化为 columns
    Z1  C1  C2
Z2
a   A   1   2
b   A   3   4
a   B   5   6
b   B   7   8
>>>df.reset_index(drop = True)#将原索引删除，不加入 columns
    C1  C2
0   1   2
1   3   4
2   5   6
3   7   8
```

reset_index()方法常用于数据分组、数据透视表中。

第 6 章

菜品挑选——数据选择

之前是把所有的菜品都洗好并放在不同的容器里。现在要进行切配了，需要把这些菜品挑选出来，比如做一盘凉拌黄瓜，需要先把黄瓜找出来；要做一盘可乐鸡翅，需要先把鸡翅找出来。

数据分析也是同样的道理，你要分析什么，首先要把对应的数据筛选出来。

常规的数据选择主要有列选择、行选择、行列同时选择三种方式。

6.1 列选择

6.1.1 选择某一列/某几列

Excel 实现

在 Excel 中选择某一列直接用鼠标选中这一列即可；如果要同时选择多列，且待选择的列不是相邻的，这个时候就可以先选中其中一列，然后按住 Ctrl 键不放，再选择其他列。举个例子，是同时选择客户姓名和成交时间这两列，如下所示。

序号	订单编号	客户姓名	唯一识别码	成交时间	销售ID
1	A1	张通	101	2018/8/8	1
2	A2	李谷	102	2018/8/9	2
3	A3	孙凤	103	2018/8/10	1
4	A4	赵恒	104	2018/8/11	2
5	A5	赵恒	104	2018/8/12	3

Python 实现

在 Python 中我们要想获取某列只需要在表 df 后面的方括号中指明要选择的列名即可。如果是一列，则只需要传入一个列名；如果是同时选择多列，则传入多个列名即可，多个列名用一个 list 存起来。

```
>>>df
    订单编号    客户姓名  唯一识别码    成交时间
0   A1          张通      101         2018-08-08
```

```
1   A2          李谷       102         2018-08-09
2   A3          孙凤       103         2018-08-10
3   A4          赵恒       104         2018-08-11
4   A5          赵恒       104         2018-08-12
>>>df["订单编号"]
  订单编号
0 A1
1 A2
2 A3
3 A4
4 A5
>>>df[["订单编号","客户姓名"]]
  订单编号 客户姓名
0 A1       张通
1 A2       李谷
2 A3       孙凤
3 A4       赵恒
4 A5       赵恒
```

在 Python 中我们把这种通过传入列名选择数据的方式称为普通索引。

除了传入具体的列名，我们还可以传入具体列的位置，即第几列，对数据进行选取，通过传入位置来获取数据时需要用到 iloc 方法。

```
#获取第一列和第三列的数据
>>>df
    订单编号     客户姓名   唯一识别码    成交时间
0   A1          张通       101         2018-08-08
1   A2          李谷       102         2018-08-09
2   A3          孙凤       103         2018-08-10
3   A4          赵恒       104         2018-08-11
4   A5          赵恒       104         2018-08-12
>>>df.iloc[:,[0,2]]#获取第 1 列和第 3 列的数值
  订单编号  唯一识别码
0 A1        101
1 A2        102
2 A3        103
3 A4        104
4 A5        104
```

在上面的代码中 iloc 后的方括号中逗号之前表示要获取的行的位置，只输入一个冒号，不输入任何数值表示获取所有的行；逗号之后的方括号表示要获取的列的位置，列的位置同样是也是从 0 开始计数。

我们把这种通过传入具体位置来选择数据的方式称为位置索引。

6.1.2 选择连续的某几列

Excel 实现

在 Excel 中，要选择连续的几列时，直接用鼠标选中这几列即可操作。当然了，

你也可以先选择一列，然后按住 Ctrl 键再去选择其他列，由于要选取的列是连续的，因此没必要这么麻烦。

Python 实现

在 Python 中可以通过前面介绍的普通索引和位置索引获取某一列或多列的数据。当你要获取的是连续的某几列，用普通索引和位置索引也是可以做到的，但是因为你要获取的列是连续的，所以只要传入这些连续列的位置区间即可，同样需要用到 iloc 方法。

```
#获取第一列到第三列的数据
>>>df
    订单编号    客户姓名  唯一识别码    成交时间
0   A1          张通      101         2018-08-08
1   A2          李谷      102         2018-08-09
2   A3          孙凤      103         2018-08-10
3   A4          赵恒      104         2018-08-11
4   A5          赵恒      104         2018-08-12
>>>df.iloc[:,0:3]#获取第 1 列到第 4 列的值
    订单编号    客户姓名  唯一识别码
0   A1          张通      101
1   A2          李谷      102
2   A3          孙凤      103
3   A4          赵恒      104
4   A5          赵恒      104
```

上面代码中，iloc 后的方括号中逗号之前的表示选择的行，当只传入一个冒号时，表示选择所有行；逗号后面表示要选择列的位置区间，0:3 表示选择第 1 列到第 4 列之间的值（包含第 1 列但不包含第 4 列），我们把这种通过传入一个位置区间来获取数据的方式称为切片索引。

6.2 行选择

6.2.1 选择某一行/某几行

Excel 实现

在 Excel 中选择行与选择列的方式是一样的，先选择一行，按住 Ctrl 键再选择其他行。

Python 实现

在 Python 中，获取行的方式主要有两种，一种是普通索引，即传入具体行索引的

名称，需要用到 loc 方法；另一种是位置索引，即传入具体的行数，需要用到 iloc 方法。

为了让大家看得更清楚，我们对行索引进行自定义。

```
#利用 loc 方法
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
一  A1         张通      101        2018-08-08
二  A2         李谷      102        2018-08-09
三  A3         孙凤      103        2018-08-10
四  A4         赵恒      104        2018-08-11
五  A5         赵恒      104        2018-08-12
>>>df.loc["一"]#选择一行
订单编号          A1
客户姓名          张通
唯一识别码        101
成交时间     2018/8/8
Name: 一, dtype: object
>>>df.loc[["一","二"]]#选择多行
    订单编号    客户姓名  唯一识别码   成交时间
一  A1         张通      101        2018-08-08
二  A2         李谷      102        2018-08-09
#利用 iloc 方法
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
一  A1         张通      101        2018-08-08
二  A2         李谷      102        2018-08-09
三  A3         孙凤      103        2018-08-10
四  A4         赵恒      104        2018-08-11
五  A5         赵恒      104        2018-08-12
>>>df.iloc[0]#选择第一行
订单编号          A1
客户姓名          张通
唯一识别码        101
成交时间     2018/8/8
Name: 一, dtype: object
>>>df.iloc[[0,1]]#选择第一和第二行
    订单编号    客户姓名  唯一识别码   成交时间
一  A1         张通      101        2018-08-08
二  A2         李谷      102        2018-08-09
```

6.2.2 选择连续的某几行

Excel 实现

在 Excel 中选择连续的某几行与选择连续的某几列方法一致，不再赘述。

Python 实现

在 Python 中，选择连续的某几行时，你同样可以把要选择的每一个行索引名字或者行索引的位置输进去。很显然这是没有必要的，只要把连续行的位置用一个区间表示，然后传给 iloc 即可。

```
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
一   A1         张通      101        2018-08-08
二   A2         李谷      102        2018-08-09
三   A3         孙凤      103        2018-08-10
四   A4         赵恒      104        2018-08-11
五   A5         赵恒      104        2018-08-12
>>>df.iloc[0:2]#选择第一到第三行
    订单编号    客户姓名  唯一识别码   成交时间
一   A1         张通      101        2018-08-08
二   A2         李谷      102        2018-08-09
三   A3         孙凤      103        2018-08-10
```

6.2.3 选择满足条件的行

前两节获取某一列时，获取的是这一列的所有行，我们还可以只筛选出这一列中满足条件的值。

比如年龄这一列，需要把非异常值（大于 200 的属于异常值），即小于 200 岁的年龄筛选出来，该怎么实现呢？

Excel 实现

在 Excel 中我们直接使用筛选功能，将满足条件的值筛选出来，筛选方法如下图所示。

筛选年龄小于 200 的数据前后的对比如下图所示。

Before

订单编号	客户姓名	唯一识别码	年龄	成交时间
A1	张通	101	31	2018/8/8
A2	李谷	102	45	2018/8/9
A3	孙凤	103	23	2018/8/10
A4	赵恒	104	240	2018/8/11
A5	赵恒	104	240	2018/8/12

After

订单编号	客户姓名	唯一识别码	年龄	成交时间
A1	张通	101	31	2018/8/8
A2	李谷	102	45	2018/8/9
A3	孙凤	103	23	2018/8/10

Python 实现

在 Python 中，我们直接在表名后面指明哪列要满足什么条件，就可以把满足条件的数据筛选出来。

```
>>>df
    订单编号  客户姓名  唯一识别码  年龄  成交时间
0   A1        张通      101        31    2018-08-08
1   A2        李谷      102        45    2018-08-09
2   A3        孙凤      103        23    2018-08-10
3   A4        赵恒      104        240   2018-08-11
4   A5        赵恒      104        240   2018-08-12
>>>df[df["年龄"]<200]#选择年龄小于 200 的数据
    订单编号  客户姓名  唯一识别码  年龄  成交时间
0   A1        张通      101        31    2018-08-08
1   A2        李谷      102        45    2018-08-09
2   A3        孙凤      103        23    2018-08-10
```

我们把上面这种通过传入一个判断条件来选择数据的方式称为布尔索引。

传入的条件还可以是多个，如下为选择的年龄小于 200 且唯一识别码小于 102 的数据。

```
>>>df[(df["年龄"]<200) & (df["唯一识别码"]<102)]
    订单编号  客户姓名  唯一识别码  年龄  成交时间
0   A1        张通      101        31    2018-08-08
```

6.3 行列同时筛选

上面的数据选择都是针对单一的行或列进行选择，实际业务中我们也会用到行列同时选择，所谓的行列选择就是选择出行和列的相交部分。

例如，我们要选择第二、三行和第二、三列相交部分的数据，下图中的阴影部分就是最终的选择结果。

订单编号	客户姓名	唯一识别码	年龄	成交时间
A1	张通	101	31	2018/8/8
A2	**李谷**	**102**	45	2018/8/9
A3	**孙凤**	**103**	23	2018/8/10
A4	赵恒	104	240	2018/8/11
A5	赵恒	104	240	2018/8/12

行列同时选择在 Excel 中主要是通过鼠标拖曳实现的，与前面的单一行列选取一致，此处不再赘述，接下来主要讲讲在 Python 中如何实现。

6.3.1 普通索引+普通索引选择指定的行和列

普通索引+普通索引就是通过同时传入行和列的索引名称进行数据选择，需要用到 loc 方法。

```
#获取第一行、第三行和第一列、第三列数据
>>>df
    订单编号     客户姓名   唯一识别码    成交时间
一    A1         张通       101          2018-08-08
二    A2         李谷       102          2018-08-09
三    A3         孙凤       103          2018-08-10
四    A4         赵恒       104          2018-08-11
五    A5         赵恒       104          2018-08-12

#用 loc 方法传入行列名称
>>>df.loc[["一","三"],["订单编号","唯一识别码"]]
    订单编号     唯一识别码
一   A1          101
三   A2          102
```

loc 方法中的第一对方括号表示行索引的选择，传入行索引名称；loc 方法中的第二对方括号表示列索引的选择，传入列索引名称。

6.3.2 位置索引+位置索引选择指定的行和列

位置索引+位置索引是通过同时传入行列索引的位置来获取数据，需要用到 iloc 方法。

```
#获取第一行、第三行和第一列、第三列数据
>>>df
    订单编号     客户姓名   唯一识别码    成交时间
一    A1         张通       101          2018-08-08
二    A2         李谷       102          2018-08-09
三    A3         孙凤       103          2018-08-10
四    A4         赵恒       104          2018-08-11
五    A5         赵恒       104          2018-08-12
#用 iloc 方法传入行列位置
>>>df.iloc[[0,2],[0,2]]
    订单编号     唯一识别码
一   A1          101
二   A2          102
```

在 iloc 方法中的第一对方括号表示行索引的选择，传入要选择行索引的位置；第二个方括号表示列索引的选择，传入要选择列索引的位置。行和列索引的位置都是从 0 开始计数。

6.3.3 布尔索引+普通索引筛选指定的行和列

布尔索引+普通索引是先对表进行布尔索引筛选行，然后通过普通索引筛选列。

```
>>>df
    订单编号  客户姓名  唯一识别码  年龄  成交时间
0   A1        张通      101        31    2018-08-08
1   A2        李谷      102        45    2018-08-09
2   A3        孙凤      103        23    2018-08-10
3   A4        赵恒      104        240   2018-08-11
4   A5        赵恒      104        240   2018-08-12

>>>df[df["年龄"]<200][["订单编号","年龄"]]
    订单编号    年龄
一   A1         31
二   A2         45
三   A3         23
```

上面的代码表示筛选年龄小于 200 的订单编号和年龄，先通过布尔索引筛选出年龄小于 200 的所有行，然后通过普通索引筛选订单编号和年龄这两列。

6.3.4 切片索引+切片索引筛选指定的行和列

切片索引+切片索引是通过同时传入行、列索引的位置区间进行数据筛选。

```
#筛选第一到第三行，第二列到第三列
>>>df
    订单编号    客户姓名  唯一识别码   成交时间
一  A1          张通      101        2018-08-08
二  A2          李谷      102        2018-08-09
三  A3          孙凤      103        2018-08-10
四  A4          赵恒      104        2018-08-11
五  A5          赵恒      104        2018-08-12
>>>df.iloc[0:2,1:2]
    客户姓名    唯一识别码
一  张通        101
二  李谷        102
三  孙凤        103
```

6.3.5 切片索引+普通索引选择指定的行和列

前面我们说过，如果是普通索引，就直接传入行或列名，用 loc 方法即可；如果是切片索引，也就是传入行或列的位置区间，要用 iloc 方法。如果是普通索引+切片索引，也就是行（列）用切片索引，列（行）用普通索引，这种交叉索引要用 ix 方法。

```
#筛选第一到第三行，客户姓名和唯一识别码这两列
>>>df
    订单编号    客户姓名  唯一识别码    成交时间
一  A1         张通      101         2018-08-08
二  A2         李谷      102         2018-08-09
三  A3         孙凤      103         2018-08-10
四  A4         赵恒      104         2018-08-11
五  A5         赵恒      104         2018-08-12
>>>df.ix[0:2,["客户姓名","唯一识别码"]]
    客户姓名   唯一识别码
一  张通       101
二  李谷       102
三  孙凤       103
```

感谢大家阅读本书试读部分，请购买本书继续学习本书剩余内容，该书目前已经在淘宝、京东、 当当网全面上线，可直接搜索《对比 Excel，轻松学习 Python 数据分析》进行选购。

京东 APP 扫描进行购买

淘宝 APP 扫描进行购买

当当 APP 扫描进行购买

扫描下方二维码关注作者公众号，可加入读者群与作者线上随时交流，更有免费视频课赠送哦。