利用Python进行数据分析

作者：Wes McKinney

O'Reilly精品图书系列

利用Python进行数据分析

Python for Data Analysis

（美）麦金尼（McKinney,W.） 著

唐学韬 译

ISBN：978-7-111-43673-7

本书纸版由机械工业出版社于2014年出版，电子版由华章分社（北京华章图文信息有限公司）全球范围内制作与发行。

客服热线：+ 86-10-68995265

客服信箱：service@bbbvip.com

官方网址：www.hzmedia.com.cn

新浪微博 @研发书局

腾讯微博 @yanfabook

目　录

O'Reilly Media,Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了Make杂志，从而成为DIY革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

译者序

说句真心话，我非常感谢有机会翻译这本书，所以这可算是第一篇我自己真正想写的译者序。虽然之前也翻译过好几本书，但都没有这次的感悟这么多、这么深！这本书是我花精力和时间最多，同时也是最不满意的一本，就是因为这些感悟——我始终觉得，如果再多点时间的话，我还可以翻译得更好。

本书的内容非常好，至少有一点非常好——集中火力对付特定的应用领域。市面上介绍编程的书多如牛毛，但几乎没有几本书是针对特定应用场景的。这本书对新手来说绝对是福音，因为每看完一点就可以马上将自己手上的工作直接拿来当例子练手，这种立竿见影的学习效果，绝对会增强新手的学习信心。

本书内容虽好，但由于作者是编辑界牛人，平时的工作肯定不少，写书方面的精力自然就不可能太多。加之美式英语本来就很口语化，导致原书口水话非常多，有些地方的从句跟绕口令似的。我在翻译的过程中尽量排除了一些，两次校稿的过程中又删除或大幅修改了一些废话，虽然这种“口水话”还存在不少，但至少不会对阅读造成太大影响。如果实在觉得语言不通顺，请随时发邮件给我，欢迎大家的善意指导（tonytang1999@126.com）。

此外，在翻译的过程中发现了不少小问题，用词方面的错误几乎都是直接改的（小部分写了译者注，因为编辑要求我尽量标出一些来以便核对），而其他错误则几乎全部采用译者注的形式说明，还有一些原文有歧义或不详尽的地方也通过译者注的形式给出了简单说明。

本书共12章，除非你已经什么都会了，否则我建议全部阅读。如果没有学过Python，建议先看看本书后面的附录。本书所用到的Python编程基础知识很少，所以只看那个附录完全足够了。但是，如果你一点儿编程基础都没有的话，可能需要再看一本有关Python入门的书才行（比如《Python编程实践》[编注1]）。

对了，还有几件事情需要说明一下：

·每章的代码示例最好在一个IPython会话中完成，否则可能会出现一些不必要的麻烦，比如“xxx未定义”。

·如果在Windows里面用IPython，复制代码的时候建议使用cpaste，这个不多解释了。

·有关地图的那段代码可能需要找英文资料看才行，我在译者注中也说明了。这可能需要花不少时间和精力。

·由于原文各种说法不统一（甚至包括术语），虽然我尽量做了统一处理，但由于精力和时间有限，无法完全修改，所以译文中的“xxx接受yyy”、“将yyy传入xxx”说的都是“xxx函数有yyy这么个参数”；“选项”、“位置参数”、“关键字参数”、“形参”、“实参”说的都是“参数”……还有不少，我也记不清了。

·“金融和经济数据”那一章翻译得非常痛苦，因为我根本不了解那个行业，原文的术语又不标准，于是我基本都是用wikipedia和bing查英文资料，看懂之后再到baidu找中文资料，并最终确定译文。因此，可能会有不准确的情况，如果您发现了，请及时通过邮件告诉我，万分感谢。

此外，我必须感谢华章公司的编辑们。非常感谢他们能够给我这样的机会，也非常感谢他们在整个过程中给予我的各种支持和理解。希望以后还能有更加愉快的合作。

本书大部分内容的翻译工作以及全书的统稿工作由我完成，参与本书翻译校对工作的还有黄惠庄、卢彦良、蒲巧惠、陈丽丽、胡元江、张杨、赵杰、吴斌、郭敏、林丹、王跃等。

由于译者水平有限，书中肯定会存在一些错误或不妥之处，因此，在阅读过程中发现有任何问题，请随时联系我们（tonytang1999@126.com）或机械工业出版社，我们将及时更新本书的勘误表。当然，也非常欢迎大家对本书提出宝贵的意见和建议。

唐学韬

2013年6月于广州

编注1:本书已由机械工业出版社出版，ISBN:978-7-111-36478-8。

前言

针对科学计算领域的Python开源库生态系统在过去10年中得到了飞速发展。2011年底，我深深地感觉到，由于缺乏集中的学习资源，刚刚接触数据分析和统计应用的Python程序员举步维艰。针对数据分析的关键项目（尤其是NumPy、matplotlib和pandas）已经很成熟了，也就是说，写一本专门介绍它们的图书貌似不会很快过时。因此，我下定决心要开始这样的一个写作项目。我在2007年刚开始用Python进行数据分析工作时就希望能够得到这样一本书。希望你也能觉得本书有用，同时也希望你能将书中介绍的那些工具高效地运用到实际工作中去。

本书的约定

本书使用了以下排版约定：

斜体（Italic）

用于新术语、URL、电子邮件地址、文件名与文件扩展名。

等宽字体（Constant width）

用于表明程序清单，以及在段落中引用的程序中的元素，如变量、函数名、数据库、数据类型、环境变量、语句、关键字等。

等宽粗体（Constant width bold）

用于表明命令，或者需要读者逐字输入的文本内容。

等宽斜体（Constant width italic）

用于表示需要使用用户提供的值或者由上下文决定的值来替代的文本内容。

注意： 代表一个技巧、建议或一般性说明。

警告： 代表一个警告或注意事项。

示例代码的使用

本书提供代码的目的是帮你快速完成工作。一般情况下，你可以在你的程序或文档中使用本书中的代码，而不必取得我们的许可，除非你想复制书中很大一部分代码。例如，你在编写程序时，用到了本书中的几个代码片段，这不必取得我们的许可。但若将O’Reilly图书中的代码制作成光盘并进行出售或传播，则需获得我们的许可。引用示例代码或书中内容来解答问题无需许可。将书中很大一部分的示例代码用于你个人的产品文档，这需要我们的许可。

如果你引用了本书的内容并标明版权归属声明，我们对此表示感谢，但这不是必需的。版权归属声明通常包括：标题、作者、出版社和ISBN号，例如："Python for Data Analysis by William Wesley McKinney (O'eilly).Copyright 2013William Wesley McKinney,978-1-449-31979-3"。

如果你认为你对示例代码的使用已经超出上述范围，或者你对是否需要获得示例代码的授权还不清楚，请随时联系我们：permissions@oreilly.com。

联系我们

有关本书的任何建议和疑问，可以通过下列方式与我们取得联系：

美国：

O'eilly Media,Inc.

1005Gravenstein Highway North

Sebastopol,CA 95472

中国：

北京市西城区西直门南大街2号成铭大厦C座807室（100035）

奥莱利技术咨询（北京）有限公司

我们会在本书的网页中列出勘误表、示例和其他信息。可以通过http://oreil.ly/Python_for_Data_Analysis访问该页面。

要评论或询问本书的技术问题，请发送电子邮件到：

bookquestions@oreilly.com

想了解关于O'eilly图书、课程、会议和新闻的更多信息，请访问以下网站：

http://www.oreilly.com.cn

http://www.oreilly.com

还可以通过以下网站关注我们：

我们在Facebook上的主页：http://facebook.com/oreilly

我们在Twitter上的主页：http://twitter.com/oreillymedia

我们在YouTube上的主页：http://www.youtube.com/oreillymedia

第1章　准备工作

本书主要内容

本书讲的是利用Python进行数据控制、处理、整理、分析等方面的具体细节和基本要点。同时，它也是利用Python进行科学计算的实用指南（专门针对数据密集型应用）。本书重点介绍了用于高效解决各种数据分析问题的Python语言和库。本书没有阐述如何利用Python实现具体的分析方法。

当书中出现“数据”时，究竟指的是什么呢？主要指的是结构化数据（structured data），这个故意含糊其辞的术语代指了所有通用格式的数据，例如：

·多维数组（矩阵）。

·表格型数据，其中各列可能是不同的类型（字符串、数值、日期等）。比如保存在关系型数据库中或以制表符/逗号为分隔符的文本文件中的那些数据。

·通过关键列（对于SQL用户而言，就是主键和外键）相互联系的多个表。

·间隔平均或不平均的时间序列。

这绝不是一个完整的列表。大部分数据集都能被转化为更加适合分析和建模的结构化形式，虽然有时这并不是很明显。如果不行的话，也可以将数据集的特征提取为某种结构化形式。例如，一组新闻文章可以被处理为一张词频表，而这张词频表就可以用于情感分析。

大部分电子表格软件（比如Microsoft Excel，它可能是世界上使用最广泛的数据分析工具了）的用户不会对此类数据感到陌生。

为什么要使用Python进行数据分析

许许多多的人（包括我自己）都很容易爱上Python这门语言。自从1991年诞生以来，Python现在已经成为最受欢迎的动态编程语言之一，其他还有Perl、Ruby等。由于拥有大量的Web框架（比如Rails（Ruby）和Django（Python）），最近几年非常流行使用Python和Ruby进行网站建设工作。这些语言常被称作脚本（scripting）语言，因为它们可以用于编写简短而粗糙的小程序（也就是脚本）。我个人并不喜欢“脚本语言”这个术语，因为它好像在说这些语言无法用于构建严谨的软件。在众多解释型语言中，Python最大的特点是拥有一个巨大而活跃的科学计算（scientific computing）社区。进入21世纪以来，在行业应用和学术研究中采用Python进行科学计算的势头越来越猛。

在数据分析和交互、探索性计算以及数据可视化等方面，Python将不可避免地接近于其他开源和商业的领域特定编程语言/工具，如R、MATLAB、SAS、Stata等。近年来，由于Python有不断改良的库（主要是pandas），使其成为数据处理任务的一大替代方案。结合其在通用编程方面的强大实力，我们完全可以只使用Python这一种语言去构建以数据为中心的应用程序。

把Python当做粘合剂

作为一个科学计算平台，Python的成功部分源于其能够轻松地集成C、C++以及Fortran代码。大部分现代计算环境都利用了一些Fortran和C库来实现线性代数、优选、积分、快速傅里叶变换以及其他诸如此类的算法。许多企业和国家实验室也利用Python来“粘合”那些已经用了30多年的遗留软件系统。

大多数软件都是由两部分代码组成的：少量需要占用大部分执行时间的代码，以及大量不经常执行的“粘合剂代码”。粘合剂代码的执行时间通常是微不足道的。开发人员的精力几乎都是花在优化计算瓶颈上面的，有时更是直接转用更低级的语言（比如C）。

最近这几年，Cython项目（http://cython.org）已经成为Python领域中创建编译型扩展以及对接C/C++代码的一大途径。

解决“两种语言”问题

很多组织通常都会用一种类似于领域特定的计算语言（如MATLAB和R）对新的想法进行研究、原型构建和测试，然后再将这些想法移植到某个更大的生产系统中去（可能是用Java、C#或C++编写的）。人们逐渐意识到，Python不仅适用于研究和原型构建，同时也适用于构建生产系统。我相信越来越多的企业也会这样看，因为研究人员和工程技术人员使用同一种编程工具将会给企业带来非常显著的组织效益。

为什么不选Python

虽然Python非常适合构建计算密集型科学应用程序以及几乎各种各样的通用系统，但它对于不少应用场景仍然力有不逮。

由于Python是一种解释型编程语言，因此大部分Python代码都要比用编译型语言（比如Java和C++）编写的代码运行慢得多。由于程序员的时间通常都比CPU时间值钱，因此许多人也愿意在这里做一些权衡。但是，在那些要求延迟非常小的应用程序中（例如高频交易系统），为了尽最大可能地优化性能，耗费时间使用诸如C++这样更低级、更低生产率的语言进行编程也是值得的。

对于高并发、多线程的应用程序而言（尤其是拥有许多计算密集型线程的应用程序），Python并不是一种理想的编程语言。这是因为Python有一个叫做全局解释器锁（Global Interpreter Lock，GIL）的东西，这是一种防止解释器同时执行多条Python字节码指令的机制。有关“为什么会存在GIL”的技术性原因超出了本书的范围，但是就目前来看，GIL并不会在短时间内消失。虽然很多大数据处理应用程序为了能在较短的时间内完成数据集的处理工作都需要运行在计算机集群上，但是仍然有一些情况需要用单进程多线程系统来解决。

这并不是说Python不能执行真正的多线程并行代码，只不过这些代码不能在单个Python进程中执行而已。比如说，Cython项目可以集成OpenMP（一个用于并行计算的C框架）以实现并行处理循环进而大幅度提高数值算法的速度。

重要的Python库

考虑到那些还不太了解Python科学计算生态系统和库的读者，下面我先对各个库做一个简单的介绍。

NumPy

NumPy（Numerical Python的简称）是Python科学计算的基础包。本书大部分内容都基于NumPy以及构建于其上的库。它提供了以下功能（不限于此）：

·快速高效的多维数组对象ndarray。

·用于对数组执行元素级计算以及直接对数组执行数学运算的函数。

·用于读写硬盘上基于数组的数据集的工具。

·线性代数运算、傅里叶变换，以及随机数生成。

·用于将C、C++、Fortran代码集成到Python的工具。

除了为Python提供快速的数组处理能力，NumPy在数据分析方面还有另外一个主要作用，即作为在算法之间传递数据的容器。对于数值型数据，NumPy数组在存储和处理数据时要比内置的Python数据结构高效得多。此外，由低级语言（比如C和Fortran）编写的库可以直接操作NumPy数组中的数据，无需进行任何数据复制工作。

pandas

pandas提供了使我们能够快速便捷地处理结构化数据的大量数据结构和函数。你很快就会发现，它是使Python成为强大而高效的数据分析环境的重要因素之一。本书用得最多的pandas对象是DataFrame，它是一个面向列（column-oriented）的二维表结构，且含有行标和列标：

```
>>> frame
    total_bill  tip   sex      smoker  day   time size
1   16.99       1.01  Female   No       Sun  Dinner  2
2   10.34       1.66  Male     No       Sun  Dinner  3
3   21.01       3.5   Male     No       Sun  Dinner  3
4   23.68       3.31  Male     No       Sun  Dinner  2
5   24.59       3.61  Female   No       Sun  Dinner  4
6   25.2        4.71  Male     No       Sun  Dinner  4
7   8.77        2     Male     No       Sun  Dinner  2
8   26.88       3.12  Male     No       Sun  Dinner  4
9   15.04       1.96  Male     No       Sun  Dinner  2
10  14.78       3.23  Male     No       Sun  Dinner  2
```

pandas兼具NumPy高性能的数组计算功能以及电子表格和关系型数据库（如SQL）灵活的数据处理功能。它提供了复杂精细的索引功能，以便更为便捷地完成重塑、切片和切块、聚合以及选取数据子集等操作。pandas将是我在本书中使用的主要工具。

对于金融行业的用户，pandas提供了大量适用于金融数据的高性能时间序列功能和工具。事实上，我一开始就是想把pandas设计为一款适用于金融数据分析应用的工具。

对于使用R语言进行统计计算的用户，肯定不会对DataFrame这个名字感到陌生，因为它源自于R的data.frame对象。但是这两个对象并不相同。R的data.frame对象所提供的功能只是DataFrame对象所提供的功能的一个子集。虽然本书讲的是Python，但我偶尔还是会用R做对比，因为它毕竟是最流行的开源数据分析环境，而且很多读者都对它很熟悉。

pandas这个名字本身源自于panel data（面板数据，这是计量经济学中关于多维结构化数据集的一个术语）以及Python data analysis（Python数据分析）。

matplotlib

matplotlib是最流行的用于绘制数据图表的Python库。它最初由John D.Hunter（JDH）创建，目前由一个庞大的开发人员团队维护。它非常适合创建出版物上用的图表。它跟IPython（马上就会讲到）结合得很好，因而提供了一种非常好用的交互式数据绘图环境。绘制的图表也是交互式的，你可以利用绘图窗口中的工具栏放大图表中的某个区域或对整个图表进行平移浏览。

IPython

IPython是Python科学计算标准工具集的组成部分，它将其他所有的东西联系到了一起。它为交互式和探索式计算提供了一个强健而高效的环境。它是一个增强的Python shell，目的是提高编写、测试、调试Python代码的速度。它主要用于交互式数据处理和利用matplotlib对数据进行可视化处理。我在用Python编程时，经常会用到IPython，包括运行、调试和测试代码。

除标准的基于终端的IPython shell外，该项目还提供了：

·一个类似于Mathematica的HTML笔记本（通过Web浏览器连接IPython，稍后将对此进行详细介绍）。

·一个基于Qt框架的GUI控制台，其中含有绘图、多行编辑以及语法高亮显示等功能。

·用于交互式并行和分布式计算的基础架构。

我将在一章中专门讲解IPython，详细地介绍其大部分功能。强烈建议在阅读本书的过程中使用IPython。

SciPy

SciPy是一组专门解决科学计算中各种标准问题域的包的集合，主要包括下面这些包：

·scipy.integrate：数值积分例程和微分方程求解器。

·scipy.linalg：扩展了由numpy.linalg提供的线性代数例程和矩阵分解功能。

·scipy.optimize：函数优化器（最小化器）以及根查找算法。

·scipy.signal：信号处理工具。

·scipy.sparse：稀疏矩阵和稀疏线性系统求解器。

·scipy.special：SPECFUN（这是一个实现了许多常用数学函数（如伽玛函数）的Fortran库）的包装器。

·scipy.stats：标准连续和离散概率分布（如密度函数、采样器、连续分布函数等）、各种统计检验方法，以及更好的描述统计法。

·scipy.weave：利用内联C++代码加速数组计算的工具。

NumPy跟SciPy的有机结合完全可以替代MATLAB的计算功能（包括其插件工具箱）。

安装和设置

由于人们用Python所做的事情不同，所以没有一个普适的Python及其插件包的安装方案。由于许多读者的Python科学计算环境都不能完全满足本书的需要，所以接下来我将详细介绍各个操作系统上的安装方法。我建议使用下列Python安装包之一：

·Enthought Python Distribution[译注1]：来自Enthought（http://continuum.io/downloads）的面向科学计算的Python安装包。包括EPDFree（免费的基本版，带有NumPy、SciPy、matplotlib、Chaco以及IPython）和EPD Full（完整版，含有100多个针对各种领域的科学计算包）。EPD Full对高校免费，非高校用户需要缴纳年费。

·Python(x,y)（http://pythonxy.googlecode.com）：Windows平台上免费的Python科学计算安装包。

我将用EPDFree来说明安装过程，当然如果有需要的话，你也可以选择其他产品。编写本书时，EPD用的是Python 2.7，今后可能会有些变动。安装完毕之后，你将可以用到下面这些包：

·Python科学计算基础库：NumPy、SciPy、matplotlib以及IPython。这些都包含在EPDFree中了。

·IPython Notebook依赖项：tornado和pyzmq。这些也都包含在EPDFree中了。

·pandas（0.8.2版或更高版本）。

在阅读本书的过程中，你可能还需要安装：statsmodels、PyTables、PyQt（PySide也行）、xlrd、lxml、basemap、pymongo以及requests等（它们被用在不同的示例中）。现在暂时还不需要安装这些库，我建议你在需要的时候再安装。例如，在OS X或Linux上安装PyQt或PyTables可能会很困难。目前最重要的事情是先用EPDFree和pandas这种最小配置运行起来再说。

关于各个Python库及其安装文件和帮助信息，请访问Python Package Index（即PyPI，http://pypi.python.org）。你还可以在这里找到不少新的Python库。

注意： 为了简单起见，我将不会讨论pip[译注2]和virtualenv这类比较复杂的环境管理工具。网上可以找到许多介绍这些工具的优秀教程。

警告： 有些用户可能会对诸如IronPython、Jython、PyPy之类的Python实现感兴趣。为了使用本书所介绍的那些工具，（目前）就需要使用基于C的标准Python解释器（也就是CPython）。

Windows

先从http://www.enthought.com下载EPDFree的安装包，它可能是一个名字类似于epd_free-7.3-1-win-x86.msi的MSI安装包[译注3]。运行该安装包并接受默认的安装位置C:\Python27。如果你之前在这里安装过Python，可能需要先将其删除（可以手工删除，也可以使用控制面板中的“添加/或删除程序”功能）。

接下来，你需要验证是否已经成功将Python添加到系统路径，并且没有跟早期安装的Python版本发生冲突。首先，打开命令提示符（打开“开始”菜单，启动“命令提示符”应用程序，即cmd.exe）。输入python尝试启动Python解释器。你应该可以看到与已安装的EPDFree版本相匹配的一段消息：

```
C:\Users\Wes>python
Python 2.7.3 |EPD_free 7.3-1 (32-bit)| (default, Apr 12 2012, 14:30:37) on win32
Type "credits", "demo" or "enthought" for more information.
>>>
```

如果你看到的是其他版本的EPD信息或根本什么也看不到，那就需要清理Windows环境变量。在Windows 7上，可以在程序搜索框中输入"environment variables"，然后编辑你的账户下的环境变量。在Windows XP上，需要进入“控制面板”→“系统”→“高级”→“环境变量”。在弹出窗口中找到Path变量。它需要含有下面这两个以分号隔开的目录路径：

```
C:\Python27;C:\Python27\Scripts
```

如果你之前安装了其他版本的Python，那就需要删除系统和用户Path变量中与之相关的一切路径。修改路径之后，需要重启命令提示符才能使修改生效。

能够在命令提示符中成功启动Python之后，就该安装pandas了。最简单的办法就是直接到http://pypi.python.org/pypi/pandas下载合适的二进制安装包。对于EPDFree，应该选择pandas-0.9.0.win32-py2.7.exe。将其安装好之后，接下来启动IPython来验证一下是不是万事俱备了：引入pandas，然后绘制一个简单的matplotlib图形。

```
C:\Users\Wes>ipython --pylab
Python 2.7.3 |EPD_free 7.3-1 (32-bit)|
Type "copyright", "credits" or "license" for more information.

IPython 0.12.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

Welcome to pylab, a matplotlib-based Python environment [backend: WXAgg]. For more information, type 'help(pylab)'.

In [1]: import pandas

In [2]: plot(arange(10))
```

如果成功，就不会出现错误信息，而且会弹出一个绘图窗口。还可以输入下列指令[译注4]来检查IPython HTML notebook是否安装成功：

```
$ ipython notebook --pylab=inline
```

警告： 如果你是在Windows上使用IPython notebook应用程序而且通常使用的是Internet Explorer的话，那你可能需要改用Mozilla Firefox或Google Chrome了[译注5]。

Windows上的EPDFree只有32位版本。如果需要使用64位版本，最简单的办法就是直接使用EPD Full[译注6]。如果你不想购买EPD订阅且愿意自己动手一步步安装，可以试试由加州大学欧文分校的Christoph Gohlke提供的非官方安装包（http://www.lfd.uci.edu/~gohlke/pythonlibs/），它既有32位版也有64位版，且包含本书所需的所有库。

苹果OS X

在OS X上，首先需要安装Xcode，它含有苹果的软件开发工具套件。我们所需的部分是gcc C和C++编译器。Xcode安装包可以在随计算机发布的OS X安装光盘中找到，也可以直接从苹果公司的网站上下载。

装好Xcode之后，到"Applications→Utilities"去启动终端（Terminal.app）。输入gcc并按回车键。你将会看到如下信息：

```
$ gcc
i686-apple-darwin10-gcc-4.2.1: no input files
```

现在就该安装EPDFree了。下载一个名为epd_free-7.3-1-macosx-i386.dmg的磁盘镜像文件。双击该.dmg文件以将其挂载到系统，然后双击其中的.mpkg文件来运行安装程序。

安装文件启动之后，会自动将EPDFree可执行文件的路径添加到你的.bash_profile文件中。该文件位于/Users/your_uname/.bash_profile：

```
# Setting PATH for EPD_free-7.3-1
PATH="/Library/Frameworks/Python.framework/Versions/Current/bin:${PATH}"
export PATH
```

如果在后续步骤中遇到任何问题，首先应该检查一下你的.bash_profile，看看是否需要将上面那个目录添加进去。

现在就该安装pandas了。在终端中执行下面这条命令：

```
$ sudo easy_install pandas
Searching for pandas
Reading http://pypi.python.org/simple/pandas/
Reading http://pandas.pydata.org
Reading http://pandas.sourceforge.net
Best match: pandas 0.9.0
Downloading http://pypi.python.org/packages/source/p/pandas/pandas-0.9.0.zip
Processing pandas-0.9.0.zip
Writing /tmp/easy_install-H5mIX6/pandas-0.9.0/setup.cfg
Running pandas-0.9.0/setup.py -q bdist_egg --dist-dir /tmp/easy_install-H5mIX6/
pandas-0.9.0/egg-dist-tmp-RhLG0z
Adding pandas 0.9.0 to easy-install.pth file

Installed /Library/Frameworks/Python.framework/Versions/7.3/lib/python2.7/
site-packages/pandas-0.9.0-py2.7-macosx-10.5-i386.egg
Processing dependencies for pandas
Finished processing dependencies for pandas
```

为了验证是否一切正常，我们以Pylab模式启动IPython，然后尝试加载pandas并绘制一张图片：

```
$ ipython --pylab
22:29 ~/VirtualBox VMs/WindowsXP $ ipython
Python 2.7.3 |EPD_free 7.3-1 (32-bit)| (default, Apr 12 2012, 11:28:34)
Type "copyright", "credits" or "license" for more information.

IPython 0.12.1 -- An enhanced Interactive Python.
?               -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help              -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.

Welcome to pylab, a matplotlib-based Python environment [backend: WXAgg].
For more information, type 'help(pylab)'.

In [1]: import pandas

In [2]: plot(arange(10))
```

如果成功，将会弹出一个绘图窗口，其中画的是一条直线。

GNU/Linux

注意： 有些（但不是全部）Linux产品自带的Python包版本较新，且可以通过内置的包管理工具（如apt）进行安装。我将详细讲解EPDFree的安装步骤，因为它在不同的Linux发行版之间是差不多的。

对于不同的Linux产品，具体的安装过程会有一些不同，我这里将以基于Debian的GNU/Linux系统（如Ubuntu和Mint）为例来进行讲解。除EPDFree之外，其他的安装过程跟OS X差不多。其安装包是一个只能在终端中执行的shell脚本。根据系统是32位还是64位，需要相应地安装x86版（32位）或x86_64版（64位）。然后你将会得到一个名为epd_free-7.3-1-rh5-x86_64.sh的文件。通过bash执行该脚本即可开始安装：

```
$ bash epd_free-7.3-1-rh5-x86_64.sh
```

在接受了许可协议之后，你需要选择EPDFree文件的存放位置。我建议将这些文件安装在你的home目录中，比如/home/wesm/epd（将wesm替换为你的用户名即可）。

安装完毕之后，你需要将EPDFree的bin目录添加到$PATH变量中去。如果你用的是bash shell（比如Ubuntu默认用的就是这个），则在你的.bashrc中加上下面这句路径添加指令：

```
export PATH=/home/wesm/epd/bin:$PATH
```

很明显，需要将/home/wesm/epd/替换为你所使用的安装目录。做完这些事情之后，你可以启动一个新的终端进程，也可以通过source ~/.bashrc重启你的.bashrc。

接下来还需要用到一个C编译器（比如gcc）。许多Linux产品都含有gcc，但有些则没有。在Debian系统中，可以执行下面这条指令来安装gcc：

```
sudo apt-get install gcc
```

如果在命令行中输入gcc，就可以看到：

```
$ gcc
gcc: no input files
```

现在可以安装pandas了：

```
$ easy_install pandas
```

如果你是以root权限来安装EPDFree的，就需要在命令中加上sudo并输入sudo或root密码。使用跟OS X相同的检测方式即可验证是否一切正常。

Python 2和Python 3

Python社区正在慢慢地从Python 2系列解释器过渡到Python 3系列。在Python 3.0问世以前，所有的Python代码都是向后兼容的。为了让Python语言更加先进，Python社区认为作出一些向后不兼容的修改是必要的。

本书是基于Python 2.7编写的，这是因为大部分Python科学计算社区还没有转向Python 3。好消息是，如果你碰巧正在使用Python 3.2，在学习本书的过程中一般也不会遇到什么麻烦。

集成开发环境（IDE）

当有人问我“你的标准开发环境是怎样的”时，我几乎总是回答“IPython外加一个文本编辑器”。我通常都在IPython中编写和调试程序，而且它可以交互式地处理数据，并通过可视化的方式验证某个数据操作的结果是否正确。诸如pandas和NumPy这样的库也可以轻松便捷地在这个shell中使用。

但是相对于文本编辑器，总有人会更喜欢IDE。因为它们提供了许多不错的“代码智能化”功能，比如自动完成以及快速获取函数和类的文档等。你可以试试下面这些：

·Eclipse +PyDev插件

·Python Tools for Visual Studio（针对Windows用户）

·PyCharm

·Spyder

·Komodo IDE

译注1：已经更名为Enthought Canopy。EPDFree对应的是Enthought Canopy Express。相比来说EPDFree自然更好用，不过为了保证阅读本书时不遇到麻烦，建议按照本书介绍法