Community Experience Distilled

Mastering Python Data Analysis

Become an expert at using Python for advanced statistical analysis of data using real-world examples

Magnus Vilhelm Persson
Luiz Felipe Martins

[PACKT] PUBLISHING open source* community experience distilled

Mastering Python Data Analysis

Become an expert at using Python for advanced statistical analysis of data using real-world examples

Magnus Vilhelm Persson

Luiz Felipe Martins

BIRMINGHAM - MUMBAI

Mastering Python Data Analysis

Publishing Month: June 2016

Production reference: 1230616

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-78355-329-7

www.packtpub.com

Credits

Authors

Magnus Vilhelm Persson
Luiz Felipe Martins

Reviewers

Hang (Harvey) Yu
Laurie Lugrin
Chris Morgan
Michele Pratusevich

Commissioning Editor

Akram Hussain

Acquisition Editor

Vinay Argekar

Content Development Editor

Arun Nadar

Technical Editors

Bharat Patil
Pranil Pathare

Copy Editor

Tasneem Fatehi

Project Coordinator

Ritika Manoj

Proofreader

Safis Editing

Indexer

Monica Ajmera Mehta

Graphics

Kirk D'Penha
Jason Monteiro

Production Coordinator

Nilesh Mohite

About the Authors

Magnus Vilhelm Persson is a scientist with a passion for Python and open source software usage and development. He obtained his PhD in Physics/Astronomy from Copenhagen University's Centre for Star and Planet Formation (StarPlan) in 2013. Since then, he has continued his research in Astronomy at various academic institutes across Europe. In his research, he uses various types of data and analysis to gain insights into how stars are formed. He has participated in radio shows about Astronomy and also organized workshops and intensive courses about the use of Python for data analysis.

You can check out his web page at `http://vilhelm.nu`.

This book would not have been possible without the great work that all the people at Packt are doing. I would like to highlight Arun, Bharat, Vinay, and Pranil's work. Thank you for your patience during the whole process. Furthermore, I would like to thank Packt for giving me the opportunity to develop and write this book, it was really fun and I learned a lot. There where times when the work was little overwhelming, but at those times, my colleague and friend Alan Heays always had some supporting words to say. Finally, my wife, Mihaela, is the most supportive partner anyone could ever have. For all the late evenings and nights where you pushed me to continue working on this to finish it, thank you. You are the most loving wife and best friend anyone could ever ask for.

Luiz Felipe Martins holds a PhD in applied mathematics from Brown University and has worked as a researcher and educator for more than 20 years. His research is mainly in the field of applied probability. He has been involved in developing code for open source homework system, WeBWorK, where he wrote a library for the visualization of systems of differential equations. He was supported by an NSF grant for this project. Currently, he is an associate professor in the department of mathematics at Cleveland State University, Cleveland, Ohio, where he has developed several courses in applied mathematics and scientific computing. His current duties include coordinating all first-year calculus sessions.

About the Reviewer

Hang (Harvey) Yu is a data scientist in Silicon Valley. He works on search engine development and model optimization. He has ample experience in big data and machine learning. He graduated from the University of Illinois at Urbana-Champaign with a background in data mining and statistics. Besides this book, he has also reviewed multiple other books and papers including *Mastering Python Data Visualization* and *R Data Analysis Cookbook both* by Packt Publishing. When Harvey is not coding, he is playing soccer, reading fiction books, or listening to classical music. You can get in touch with him at `hangyu1@illinois.edu` or on LinkedIn at `http://www.linkedin.com/in/hangyu1`.

www.PacktPub.com

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

The use of Python for data analysis and visualization has only increased in popularity in the last few years. One reason for this is the availability and continued development of a number of excellent tools for conducting advanced data analysis and visualization. Another reason is the possibility of rapid and easy development, deployment, and sharing of code. For these reasons, Python has become one of the most widely used programming and scripting language for data analysis in many industries.

The aim of this book is to develop skills to effectively approach almost any data analysis problem, and extract all of the available information. This is done by introducing a range of varying techniques and methods such as uni- and multi-variate linear regression, cluster finding, Bayesian analysis, machine learning, and time series analysis. Exploratory data analysis is a key aspect to get a sense of what can be done and to maximize the insights that are gained from the data. Additionally, emphasis is put on presentation-ready figures that are clear and easy to interpret.

Knowing how to explore data and present results and conclusions from data analysis in a meaningful way is an important skill. While the theory behind statistical analysis is important to know, to be able to quickly and accurately perform hands-on sorting, reduction, analysis, and subsequently present the insights gained, is a make or break for today's quickly evolving business and academic sector.

What this book covers

`Chapter 1`, *Tools of the Trade*, provides an overview of the tools available for data analysis in Python and details the packages and libraries that will be used in the book with some installation tips. A quick example highlights the common data structure used in the Pandas package.

`Chapter 2`, *Exploring Data*, introduces methods for initial exploration of data, including numeric summaries and distributions, and various ways of displaying data, such as histograms, Kernel Density Estimation (KDE) plots, and box plots.

`Chapter 3`, *Learning About Models*, covers the concept of models in data analysis and how using the cumulative distribution function and probability density function can help characterize a variable. Furthermore, it shows how to make point estimates and generate random numbers with a given distribution.

`Chapter 4`, *Regression*, introduces linear, multiple, and logistic regression with in-depth examples of using SciPy and statsmodels packages to test various hypotheses of relationships between variables.

`Chapter 5`, *Clustering*, explains some of the theory behind cluster finding analysis and goes through some more complex examples using the K-means and hierarchical clustering algorithms available in SciPy.

`Chapter 6`, *Bayesian Methods*, explains how to construct and test a model using Bayesian analysis in Python using the PyMC package. It covers setting up stochastic and deterministic variables with prior information, constructing the model, running the Markov Chain Monte Carlo (MCMC) sampler, and interpreting the results. In addition, a short bonus section covers how to plot coordinates on maps using both the basemap and cartopy packages, which are important for presenting and analyzing data with geographical coordinate information.

`Chapter 7`, *Supervised and Unsupervised Learning*, looks at linear regression, clustering, and classification with two machine learning analysis techniques available in the Scikit-learn package.

`Chapter 8`, *Time Series Analysis*, examines various aspects of time series modeling using Pandas and statsmodels. Initially, the important concepts of smoothing, resampling, rolling estimates, and stationarity are covered. Later, autoregressive (AR), moving average (MA), and combined ARIMA models are explained and applied to one of the data sets, including making shorter forecasts using the constructed models.

`Appendix`, *More on Jupyter Notebook and matplotlib Styles*, shows some convenient extensions of Jupyter Notebook and some useful keyboard shortcuts to make the Jupyter workflow more efficient. The matplotlib style files are explained and how to customize plots even further to make beautiful figures ready for inclusion in reports. Lastly, various useful online resources are listed and described.

What you need for this book

All you need to follow through the examples in this book is a computer running any recent version of Python. While the examples use Python 3, they can easily be adapted to work with Python 2, with only minor changes. The packages used in the examples are NumPy, SciPy, matplotlib, Pandas, statsmodels, PyMC, Scikit-learn. Optionally, the packages basemap and cartopy are used to plot coordinate points on maps. The easiest way to obtain and maintain a Python environment that meets all the requirements of this book is to download a prepackaged Python distribution. In this book, we have checked all the code against Continuum Analytics' Anaconda Python distribution and Ubuntu Xenial Xerus (16.04) running Python 3.

To download the example data and code, an Internet connection is needed.

Who this book is for

This book is intended for professionals with a beginner to intermediate level of Python programming knowledge who want to move in the direction of solving more sophisticated problems and gain deeper insights through advanced data analysis. Some experience with the math behind basic statistics is assumed, but quick introductions are given where required. If you want to learn the breadth of statistical analysis techniques in Python and get an overview of the methods and tools available, you will find this book helpful. Each chapter consists of a number of examples using mostly real-world data to highlight various aspects of the topic and teach how to conduct data analysis from start to finish.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning. Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "This code has the effect of selecting matplotlib stylesheet `mystyle.mplstyle`."

A block of code is set as follows:

```
gss_data = pd.read_stata('data/GSS2012merged_R5.dta',
                         convert_categoricals=False)
gss_data.head()
```

Any command-line input or output is written as follows:

```
python –c 'import numpy'
```

New terms and important words are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Here, you can check the box for **add a toolbar button to open the shortcuts dialog/panel**."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book-what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for this book from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

1. Log in or register to our website using your e-mail address and password.
2. Hover the mouse pointer on the **SUPPORT** tab at the top.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box.
5. Select the book for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this book from.
7. Click on **Code Download**.

You can also download the code files by clicking on the **Code Files** button on the book's webpage at the Packt Publishing website. This page can be accessed by entering the book's name in the **Search** box. Please note that you need to be logged in to your Packt account.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Mastering-Python-Data-Analysis`. We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from `https://www.packtpub.com/sites/default/files/downloads/masteringpythondataanalysis_ColorImages.pdf`.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books-maybe a mistake in the text or the code-we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the `Errata` section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
Tools of the Trade

This chapter gives you an overview of the tools available for data analysis in Python, with details concerning the Python packages and libraries that will be used in this book. A few installation tips are given, and the chapter concludes with a brief example. We will concentrate on how to read data files, select data, and produce simple plots, instead of delving into numerical data analysis.

Before you start

We assume that you have familiarity with Python and have already developed and run some scripts or used Python interactively, either in the shell or on another interface, such as the Jupyter Notebook (formerly known as the **IPython notebook**). Hence, we also assume that you have a working installation of Python. In this book, we assume that you have installed Python 3.4 or later.

We also assume that you have developed your own workflow with Python, based on needs and available environment. To follow the examples in this book, you are expected to have access to a working installation of Python 3.4 or later. There are two alternatives to get started, as outlined in the following list:

- Use a Python installation from scratch. This can be downloaded from `https://www.python.org`. This will require a separate installation for each of the required libraries.
- Install a prepackaged distribution containing libraries for scientific and data computing. Two popular distributions are Anaconda Scientific Python (`https://store.continuum.io/cshop/anaconda`) and Enthought distribution (`https://www.enthought.com`).

Even if you have a working Python installation, you might want to try one of the prepackaged distributions. They contain a well-rounded collection of packages and modules suitable for data analysis and scientific computing. If you choose this path, all the libraries in the next list are included by default.

We also assume that you have the libraries in the following list:

- `numpy` and `scipy`: These are available at `http://www.scipy.org`. These are the essential Python libraries for computational work. NumPy defines a fast and flexible array data structure, and SciPy has a large collection of functions for numerical computing. They are required by some of the libraries mentioned in the list.
- `matplotlib`: This is available at `http://matplotlib.org`. It is a library for interactive graphics built on top of NumPy. I recommend versions above 1.5, which is what is included in Anaconda Python by default.
- `pandas`: This is available at `http://pandas.pydata.org`. It is a Python data analysis library. It will be used extensively throughout the book.
- `pymc`: This is a library to make Bayesian models and fitting in Python accessible and straightforward. It is available at `http://pymc-devs.github.io/pymc/`. This package will mainly be used in `Chapter 6`, *Bayesian Methods*, of this book.
- `scikit-learn`: This is available at `http://scikit-learn.org`. It is a library for machine learning in Python. This package is used in `Chapter 7`, *Supervised and Unsupervised Learning*.
- `IPython`: This is available at `http://ipython.org`. It is a library providing enhanced tools for interactive computations in Python from the command line.
- `Jupyter`: This is available at `https://jupyter.org/`. It is the notebook interface working on top of IPython (and other programming languages). Originally part of the IPython project, the notebook interface is a web-based platform for computational and data science that allows easy integration of the tools that are used in this book.

Notice that each of the libraries in the preceding list may have several dependencies, which must also be separately installed. To test the availability of any of the packages, start a Python shell and run the corresponding `import` statement. For example, to test the availability of NumPy, run the following command:

```
import numpy
```

If NumPy is not installed in your system, this will produce an error message. An alternative approach that does not require starting a Python shell is to run the command line:

```
python -c 'import numpy'
```

We also assume that you have either a programmer's editor or Python IDE. There are several options, but at the basic level, any editor capable of working with unformatted text files will do.

Using the notebook interface

Most examples in this book will use the Jupyter Notebook interface. This is a browser-based interface that integrates computations, graphics, and other forms of media. Notebooks can be easily shared and published, for example, `http://nbviewer.ipython.org/` provides a simple publication path.

It is not, however, absolutely necessary to use the Jupyter interface to run the examples in this book. We strongly encourage, however, that you at least experiment with the notebook and its many features. The Jupyter Notebook interface makes it possible to mix formatted, descriptive text with code cells that evaluate at the same time. This feature makes it suitable for educational purposes, but it is also useful for personal use as it makes it easier to add comments and share partial progress before writing a full report. We will sometimes refer to a Jupyter Notebook as just *a notebook*.

To start the notebook interface, run the following command line from the shell or Anaconda command prompt:

```
jupyter notebook
```

The notebook server will be started in the directory where the command is issued. After a while, the notebook interface will appear in your default browser. Make sure that you are using a standards-compliant browser, such as Chrome, Firefox, Opera, or Safari. Once the Jupyter dashboard shows on the browser, click on the **New** button on the upper-right side of the page and select **Python 3**. After a few seconds, a new notebook will open in the browser. A useful place to learn about the notebook interface is `http://jupyter.org`.

Imports

There are some modules that we will need to load at the start of every project. Assuming that you are running a Jupyter Notebook, the required imports are as follows:

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Enter all the preceding commands in a single notebook cell and press *Shift + Enter* to run the whole cell. A new cell will be created when there is none after the one you are running; however, if you want to create one yourself, the menu or keyboard shortcut *Ctrl +M +A/B* is handy (*A* for above, *B* for below the current cell). In `Appendix`, *More on Jupyter Notebook and matplotlib Styles*, we cover some of the keyboard shortcuts available and installable extensions (that is, plugins) for Jupyter Notebook.

The statement `%matplotlib inline` is an example of Jupyter Notebook magic and sets up the interface to display plots inline, that is, embedded in the notebook. This line is not needed (and causes an error) in scripts. Next, optionally, enter the following commands:

```
import os
plt.style.use(os.path.join(os.getcwd(), 'mystyle.mplstyle') )
```

As before, run the cell by pressing *Shift +Enter.* This code has the effect of selecting matplotlib stylesheet `mystyle.mplstyle`. This is a custom style sheet that I created, which resides in the same folder as the notebook. It is a rather simple example of what can be done; you can modify it to your liking. As we gain experience in drawing figures throughout the book, I encourage you to play around with the settings in the file. There are also built-in styles that you can by typing `plt.style.available` in a new cell.

This is it! We are all set to start the fun part!

An example using the Pandas library

The purpose of this example is to check whether everything is working in your installation and give a flavor of what is to come. We concentrate on the Pandas library, which is the main tool used in Python data analysis.

We will use the MovieTweetings 50K movie ratings dataset, which can be downloaded from `https://github.com/sidooms/MovieTweetings`. The data is from the study MovieTweetings: a Movie Rating Dataset Collected From Twitter – by Dooms, De Pessemier and Martens presented during Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys (2013). The dataset is spread in several text files, but we will only use the following two files:

- `ratings.dat`: This is a double colon-separated file containing the ratings for each user and movie
- `movies.dat`: This file contains information about the movies

To see the contents of these files, you can open them with a standard text editor. The data is organized in columns, with one data item per line. The meanings of the columns are described in the `README.md` file, distributed with the dataset. The data has a peculiar aspect: some of the columns use a double colon (`::`) character as a separator, while others use a vertical bar (`|`). This emphasizes a common occurrence with real-world data: we have no control on how the data is collected and formatted. For data stored in text files, such as this one, it is always a good strategy to open the file in a text editor or spreadsheet software to take a look at the data and identify inconsistencies and irregularities.

To read the ratings file, run the following command:

```
cols = ['user id', 'item id', 'rating', 'timestamp']
ratings = pd.read_csv('data/ratings.dat', sep='::',
                      index_col=False, names=cols,
                      encoding="UTF-8")
```

The first line of code creates a Python list with the column names in the dataset. The next command reads the file, using the `read_csv()` function, which is part of Pandas. This is a generic function to read column-oriented data from text files. The arguments used in the call are as follows:

- `data/ratings.dat`: This is the path to file containing the data (this argument is required).
- `sep='::'`: This is the separator, a double colon character in this case.
- `index_col=False`: We don't want any column to be used as an index. This will cause the data to be indexed by successive integers, starting with 1.
- `names=cols`: These are the names to be associated with the columns.

The `read_csv()` function returns a DataFrame object, which is the Pandas data structure that represents tabular data. We can view the first rows of the data with the following command:

```
ratings[:5]
```

This will output a table, as shown in the following image:

	user id	item id	rating	timestamp
0	1	1074638	7	1365029107
1	1	1853728	8	1366576639
2	2	104257	8	1364690142
3	2	1259521	8	1364118447
4	2	1991245	7	1364117717

To start working with the data, let us find out how many times each rating appears in the table. This can be done with the following commands:

```
rating_counts = ratings['rating'].value_counts()
rating_counts
```

The first line of code computes the counts and stores them in the `rating_counts` variable. To obtain the count, we first use the `ratings['rating']` expression to select the `rating` column from the table ratings. Then, the `value_counts()` method is called to compute the counts. Notice that we retype the variable name, `rating_counts`, at the end of the cell. This is a common notebook (and Python) idiom to print the value of a variable in the output area that follows each cell. In a script, it has no effect; we could have printed it with the print command, `(print(rating_counts))`, as well. The output is displayed in the following image:

```
8     12012
7     11063
9      7119
6      6373
10     6281
5      3399
4      1696
3       924
1       595
2       533
0         5
Name: rating, dtype: int64
```

Notice that the output is sorted according to the count values in descending order. The object returned by `value_counts` is of the Series type, which is the Pandas data structure used to represent one-dimensional, indexed, data. The Series objects are used extensively in Pandas. For example, the columns of a DataFrame object can be thought as Series objects that share a common index.

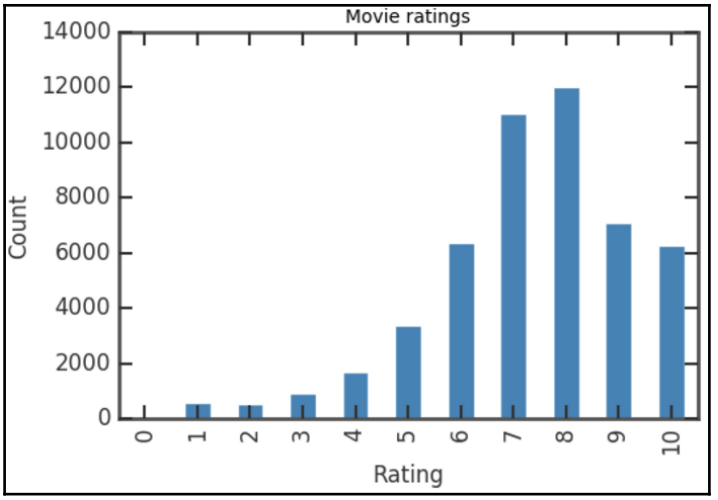
In our case, it makes more sense to sort the rows according to the ratings. This can be achieved with the following commands:

```
sorted_counts = rating_counts.sort_index()
sorted_counts
```

This works by calling the `sort_index()` method of the Series object, `rating_counts`. The result is stored in the `sorted_counts` variable. We can now get a quick visualization of the ratings distribution using the following commands:

```
sorted_counts.plot(kind='bar', color='SteelBlue')
plt.title('Movie ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
```

The first line produces the plot by calling the `plot()` method for the `sorted_counts` object. We specify the `kind='bar'` option to produce a bar chart. Notice that we added the `color='SteelBlue'` option to select the color of the bars in the histogram. `SteelBlue` is one of the HTML5 color names (for example, `http://matplotlib.org/examples/color/named_colors.html`) available in matplotlib. The next three statements set the title, horizontal axis label, and vertical axis label respectively. This will produce the following plot:



The vertical bars show how many voters that have given a certain rating, covering all the movies in the database. The distribution of the ratings is not very surprising: the counts increase up to a rating of `8`, and the count of `9-10` ratings is smaller as most people are reluctant to give the highest rating. If you check the values of the bar for each rating, you can see that it corresponds to what we had previously when printing the `rating_counts` object. To see what happens if you do not sort the ratings first, plot the `rating_counts` object, that is, run `rating_counts.plot(kind='bar', color='SteelBlue')` in a cell.

Let's say that we would like to know if the ratings distribution for a particular movie genre, say `Crime Drama`, is similar to the overall distribution. We need to cross-reference the ratings information with the movie information, contained in the `movies.dat` file. To read this file and store it in a Pandas DataFrame object, use the following command:

```
cols = ['movie id','movie title','genre']
movies = pd.read_csv('data/movies.dat', sep='::',
                     index_col=False, names=cols,
                     encoding="UTF-8")
```

Downloading the example code

Detailed steps to download the code bundle are mentioned in the Preface of this book. Please have a look.

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Mastering-Python-Data-Analysis`. We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

We are again using the `read_csv()` function to read the data. The column names were obtained from the `README.md` file distributed with the data. Notice that the separator used in this file is also a double colon, `::`. The first few lines of the table can be displayed with the command:

```
movies[:5]
```

Notice how the genres are indicated, clumped together with a vertical bar, `|`, as separator. This is due to the fact that a movie can belong to more than one genre. We can now select only the movies that are crime dramas using the following lines:

```
drama = movies[movies['genre']=='Crime|Drama']
```

Notice that this uses the standard indexing notation with square brackets, `movies[...]`. Instead of specifying a numeric or string index, however, we are using the Boolean `movies['genre']=='Crime|Drama'` expression as an index. To understand how this works, run the following code in a cell:

```
is_drama = movies['genre']=='Crime|Drama'
is_drama[:5]
```

This displays the following output:

```
0     True
1    False
2    False
3    False
4    False
Name: genre, dtype: bool
```

The `movies['genre']=='Crime|Drama'` expression returns a Series object, where each entry is either `True` or `False`, indicating whether the corresponding movie is a crime drama or not, respectively.

Thus, the net effect of the `drama = movies[movies['genre']=='Crime|Drama']` assignment is to select all the rows in the movies table for which the entry in the `genre` column is equal to `Crime|Drama` and store the result in the `drama` variable, which is an object of the DataFrame type.

All that we need is the `movie id` column of this table, which can be selected with the following statement:

```
drama_ids = drama['movie id']
```

This, again, uses standard indexing with a string to select a column from a table.

The next step is to extract those entries that correspond to dramas from the `ratings` table. This requires yet another indexing trick. The code is contained in the following lines:

```
criterion = ratings['item id'].map(lambda x:(drama_ids==x).any())
drama_ratings = ratings[criterion]
```

The key to how this code works is the definition of the variable `criterion`. We want to look up each row of the `ratings` table and check whether the `item id` entry is in the `drama_ids` table. This can be conveniently done by the `map()` method. This method applies a function to all the entries of a Series object. In our example, the function is as follows:

```
lambda x:(drama_ids==x).any()
```

This function simply checks whether an item appears in `drama_ids`, and if it does, it returns `True`. The resulting object `criterion` will be a Series that contains the `True` value only in the rows that correspond to dramas. You can view the first rows with the following code:

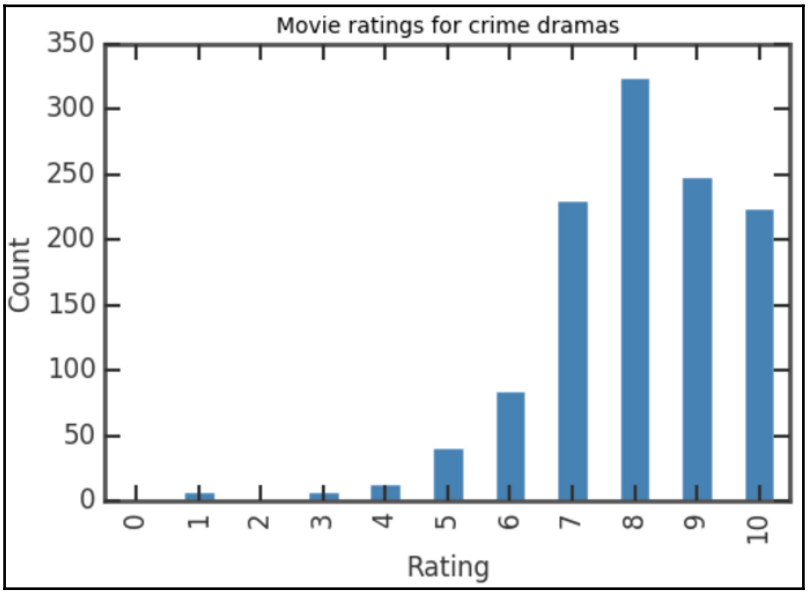
```
criterion[:10]
```

We then use the `criterion` object as an index to select the rows from the `ratings` table.

We are now done with selecting the data that we need. To produce a rate count and bar chart, we use the same commands as before. The details are in the following code, which can be run in a single execution cell:

```
rating_counts = drama_ratings['rating'].value_counts()
sorted_counts = rating_counts.sort_index()
sorted_counts.plot(kind='bar', color='SteelBlue')
plt.title('Movie ratings for dramas')
plt.xlabel('Rating')
plt.ylabel('Count')
```

As before, this code first computes the counts, indexes them according to the ratings, and then produces a bar chart. This produces a graph that seems to be similar to the overall ratings distribution, as shown in the following figure:



Summary

In this chapter, we have seen what tools are available for data analysis in Python, reviewed issues related to installation and workflow, and considered a simple example that requires reading and manipulating data files.

In the next chapter, we will cover techniques to explore data graphically and numerically using some of the main tools provided by the Pandas module.