Table of Contents

Preface v

Chapter 1: First Steps 1

- **Introducing data science and Python** 2
- **Installing Python** 3
  - Python 2 or Python 3? 3
  - Step-by-step installation 4
  - A glance at the essential Python packages 5
    - NumPy 5
    - SciPy 6
    - pandas 6
    - Scikit-learn 6
    - IPython 7
    - Matplotlib 7
    - Statsmodels 8
    - Beautiful Soup 8
    - NetworkX 8
    - NLTK 9
    - Gensim 9
    - PyPy 9
  - The installation of packages 10
  - Package upgrades 11
- **Scientific distributions** 12
  - Anaconda 12
  - Enthought Canopy 13
  - PythonXY 13
  - WinPython 13
- **Introducing IPython** 13
  - The IPython Notebook 15

Datasets and code used in the book 22
Scikit-learn toy datasets 22
The MLdata.org public repository 26
LIBSVM data examples 26
Loading data directly from CSV or text files 27
Scikit-learn sample generators 30
Summary 31

Chapter 2: Data Munging 33

The data science process 34
Data loading and preprocessing with pandas 35
Fast and easy data loading 35
Dealing with problematic data 38
Dealing with big datasets 41
Accessing other data formats 45
Data preprocessing 47
Data selection 49
Working with categorical and textual data 52
A special type of data – text 54
Data processing with NumPy 60
NumPy's n-dimensional array 61
The basics of NumPy ndarray objects 62
Creating NumPy arrays 63
From lists to unidimensional arrays 63
Controlling the memory size 64
Heterogeneous lists 65
From lists to multidimensional arrays 66
Resizing arrays 68
Arrays derived from NumPy functions 69
Getting an array directly from a file 71
Extracting data from pandas 71
NumPy fast operation and computations 72
Matrix operations 75
Slicing and indexing with NumPy arrays 76
Stacking NumPy arrays 79
Summary 81

Chapter 3: The Data Science Pipeline 83

Introducing EDA 83
Feature creation 87
Dimensionality reduction 90
The covariance matrix 90
Principal Component Analysis (PCA) 91

A variation of PCA for big data–randomized PCA 95
Latent Factor Analysis (LFA) 96
Linear Discriminant Analysis (LDA) 97
Latent Semantical Analysis (LSA) 97
Independent Component Analysis (ICA) 98
Kernel PCA 99
Restricted Boltzmann Machine (RBM) 100
The detection and treatment of outliers 102
Univariate outlier detection 103
EllipticEnvelope 105
OneClassSVM 110
Scoring functions 114
Multilabel classification 114
Binary classification 116
Regression 117
Testing and validating 118
Cross-validation 123
Using cross-validation iterators 125
Sampling and bootstrapping 127
Hyper-parameters' optimization 129
Building custom scoring functions 132
Reducing the grid search runtime 135
Feature selection 136
Univariate selection 137
Recursive elimination 139
Stability and L1-based selection 140
Summary 142
Chapter 4: Machine Learning 143
Linear and logistic regression 143
Naive Bayes 147
The k-Nearest Neighbors 150
Advanced nonlinear algorithms 152
SVM for classification 152
SVM for regression 155
Tuning SVM 156
Ensemble strategies 158
Pasting by random samples 158
Bagging with weak ensembles 159
Random Subspaces and Random Patches 160
Sequences of models – AdaBoost 162

Gradient tree boosting (GTB) 162
Dealing with big data 163
Creating some big datasets as examples 164
Scalability with volume 165
Keeping up with velocity 167
Dealing with variety 169
A quick overview of Stochastic Gradient Descent (SGD) 171
A peek into Natural Language Processing (NLP) 172
Word tokenization 173
Stemming 174
Word Tagging 174
Named Entity Recognition (NER) 175
Stopwords 176
A complete data science example – text classification 177
An overview of unsupervised learning 179
Summary 184
Chapter 5: Social Network Analysis 187
Introduction to graph theory 187
Graph algorithms 192
Graph loading, dumping, and sampling 199
Summary 203
Chapter 6: Visualization 205
Introducing the basics of matplotlib 205
Curve plotting 206
Using panels 208
Scatterplots 209
Histograms 210
Bar graphs 212
Image visualization 213
Selected graphical examples with pandas 215
Boxplots and histograms 216
Scatterplots 218
Parallel coordinates 221
Advanced data learning representation 221
Learning curves 222
Validation curves 224
Feature importance 225
GBT partial dependence plot 227
Summary 228
Index 231

Preface

"A journey of a thousand miles begins with a single step."

–Laozi (604 BC - 531 BC)

Data science is a relatively new knowledge domain that requires the successful integration of linear algebra, statistical modelling, visualization, computational linguistics, graph analysis, machine learning, business intelligence, and data storage and retrieval.

The Python programming language, having conquered the scientific community during the last decade, is now an indispensable tool for the data science practitioner and a must-have tool for every aspiring data scientist. Python will offer you a fast, reliable, cross-platform, mature environment for data analysis, machine learning, and algorithmic problem solving. Whatever stopped you before from mastering Python for data science applications will be easily overcome by our easy step-by-step and example-oriented approach that will help you apply the most straightforward and effective Python tools to both demonstrative and real-world datasets.

Leveraging your existing knowledge of Python syntax and constructs (but don't worry, we have some Python tutorials if you need to acquire more knowledge on the language), this book will start by introducing you to the process of setting up your essential data science toolbox. Then, it will guide you through all the data munging and preprocessing phases. A necessary amount of time will be spent in explaining the core activities related to transforming, fixing, exploring, and processing data. Then, we will demonstrate advanced data science operations in order to enhance critical information, set up an experimental pipeline for variable and hypothesis selection, optimize hyper-parameters, and use cross-validation and testing in an effective way.

Finally, we will complete the overview by presenting you with the main machine learning algorithms, graph analysis technicalities, and all the visualization instruments that can make your life easier when it comes to presenting your results.

In this walkthrough, which is structured as a data science project, you will always be accompanied by clear code and simplified examples to help you understand the underlying mechanics and real-world datasets. It will also give you hints dictated by experience to help you immediately operate on your current projects. Are you ready to start? We are sure that you are ready to take the first step towards a long and incredibly rewarding journey.

What this book covers

Chapter 1, *First Steps*, introduces you to all the basic tools (command shell for interactive computing, libraries, and datasets) necessary to immediately start on data science using Python.

Chapter 2, *Data Munging*, explains how to upload the data to be analyzed by applying alternative techniques when the data is too big for the computer to handle. It introduces all the key data manipulation and transformation techniques.

Chapter 3, *The Data Science Pipeline*, offers advanced explorative and manipulative techniques, enabling sophisticated data operations to create and reduce predictive features, spot anomalous cases and apply validation techniques.

Chapter 4, *Machine Learning*, guides you through the most important learning algorithms that are available in the Scikit-learn library, which demonstrates the practical applications and points out the key values to be checked and the parameters to be tuned in order to get the best out of each machine learning technique.

Chapter 5, *Social Network Analysis*, elaborates the practical and effective skills that are required to handle data that represents social relations or interactions.

Chapter 6, *Visualization*, completes the data science overview with basic and intermediate graphical representations. They are indispensable if you want to visually represent complex data structures and machine learning processes and results.

Chapter 7, *Strengthen Your Python Foundations*, covers a few Python examples and tutorials focused on the key features of the language that it is indispensable to know in order to work on data science projects.

This chapter is not part of the book, but it has to be downloaded from Packt Publishing website at `https://www.packtpub.com/sites/default/files/downloads/0429OS_Chapter-07.pdf`.

What you need for this book

Python and all the data science tools mentioned in the book, from IPython to Scikit-learn, are free of charge and can be freely downloaded from the Internet. To run the code that accompanies the book, you need a computer that uses Windows, Linux, or Mac OS operating systems. The book will introduce you step-by-step to the process of installing the Python interpreter and all the tools and data that you need to run the examples.

Who this book is for

This book builds on the core skills that you already have, enabling you to become an efficient data science practitioner. Therefore, it assumes that you know the basics of programming and statistics.

The code examples provided in the book won't require you to have a mastery of Python, but we will assume that you know at least the basics of Python scripting, lists and dictionary data structures, and how class objects work. Before starting, you can quickly acquire such skills by spending a few hours on the online courses that we are going to suggest in the first chapter. You can also use the tutorial provided on the Packt Publishing website.

No advanced data science concepts are necessary though, as we will provide you with the information that is essential to understand all the core concepts that are used by the examples in the book.

Summarizing, this book is for the following:

- Novice and aspiring data scientists with limited Python experience and a working knowledge of data analysis, but no specific expertise of data science algorithms
- Data analysts who are proficient in statistic modeling using R or MATLAB tools and who would like to exploit Python to perform data science operations
- Developers and programmers who intend to expand their knowledge and learn about data manipulation and machine learning

Conventions

In this book, you will find a number of styles of text 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: "When inspecting the linear model, first check the `coef_` attribute."

A block of code is set as follows:

```
from sklearn import datasets
iris = datasets.load_iris()
```

Since we will be using IPython Notebooks along most of the examples, expect to have always an input (marked as `In:`) and often an output (marked `Out:`) from the cell containing the block of code. On your computer you have just to input the code after the `In:` and check if results correspond to the `Out:` content:

```
In: clf.fit(X, y)
Out: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
degree=3, gamma=0.0, kernel='rbf', max_iter=-1, probability=False,
random_state=None, shrinking=True, tol=0.001, verbose=False)
```

When a command should be given in the terminal command line, you'll find the command with the prefix `$>`, otherwise, if it's for the Python REPL, it will be preceded by `>>>`:

```
$>python
>>> import sys
>>> print sys.version_info
```

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 from your account at `http://www.packtpub.com` for all the Packt Publishing books you have purchased. 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.

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
First Steps

Whether you are an eager learner of data science or a well-grounded data science practitioner, you can take advantage of this essential introduction to Python for data science. You can use it to the fullest if you already have at least some previous experience in basic coding, writing general-purpose computer programs in Python, or some other data analysis-specific language, such as MATLAB or R.

The book will delve directly into Python for data science, providing you with a straight and fast route to solve various data science problems using Python and its powerful data analysis and machine learning packages. The code examples that are provided in this book don't require you to master Python. However, they will assume that you at least know the basics of Python scripting, data structures such as lists and dictionaries, and the working of class objects. If you don't feel confident about this subject or have minimal knowledge of the Python language, we suggest that before you read this book, you should take an online tutorial, such as the Code Academy course at `http://www.codecademy.com/en/tracks/python` or Google's Python class at `https://developers.google.com/edu/python/`. Both the courses are free, and in a matter of a few hours of study, they should provide you with all the building blocks that will ensure that you enjoy this book to the fullest. We have also prepared a tutorial of our own, which you can download from the Packt Publishing website, in order to provide an integration of the two aforementioned free courses.

In any case, don't be intimidated by our starting requirements; mastering Python for data science applications isn't as arduous as you may think. It's just that we have to assume some basic knowledge on the reader's part because our intention is to go straight to the point of using data science without having to explain too much about the general aspects of the language that we will be using.

Are you ready, then? Let's start!

In this short introductory chapter, we will work out the basics to set off in full swing and go through the following topics:

- How to set up a Python **Data Science Toolbox**
- Using IPython
- An overview of the data that we are going to study in this book

Introducing data science and Python

Data science is a relatively new knowledge domain, though its core components have been studied and researched for many years by the computer science community. These components include linear algebra, statistical modelling, visualization, computational linguistics, graph analysis, machine learning, business intelligence, and data storage and retrieval.

Being a new domain, you have to take into consideration that currently the frontier of data science is still somewhat blurred and dynamic. Because of its various constituent set of disciplines, please keep in mind that there are different profiles of data scientists, depending on their competencies and areas of expertise.

In such a situation, what can be the best tool of the trade that you can learn and effectively use in your career as a data scientist? We believe that the best tool is Python, and we intend to provide you with all the essential information that you will need for a fast start.

Also, other tools such as R and MATLAB provide data scientists with specialized tools to solve specific problems in statistical analysis and matrix manipulation in data science. However, only Python completes your data scientist skill set. This multipurpose language is suitable for both development and production alike and is easy to learn and grasp, no matter what your background or experience is.

Created in 1991 as a general-purpose, interpreted, object-oriented language, Python has slowly and steadily conquered the scientific community and grown into a mature ecosystem of specialized packages for data processing and analysis. It allows you to have uncountable and fast experimentations, easy theory developments, and prompt deployments of scientific applications.

At present, the Python characteristics that render it an indispensable data science tool are as follows:

- Python can easily integrate different tools and offer a truly unifying ground for different languages (Java, C, Fortran, and even language primitives), data strategies, and learning algorithms that can be easily fitted together and which can concretely help data scientists forge new powerful solutions.

- It offers a large, mature system of packages for data analysis and machine learning. It guarantees that you will get all that you may need in the course of a data analysis, and sometimes even more.
- It is very versatile. No matter what your programming background or style is (object-oriented or procedural), you will enjoy programming with Python.
- It is cross-platform; your solutions will work perfectly and smoothly on Windows, Linux, and Mac OS systems. You won't have to worry about portability.
- Although interpreted, it is undoubtedly fast compared to other mainstream data analysis languages such as R and MATLAB (though it is not comparable to C, Java, and the newly emerged Julia language). It can be even faster, thanks to some easy tricks that we are going to explain in this book.
- It can work with in-memory big data because of its minimal memory footprint and excellent memory management. The memory garbage collector will often save the day when you load, transform, dice, slice, save, or discard data using the various iterations and reiterations of data wrangling.
- It is very simple to learn and use. After you grasp the basics, there's no other better way to learn more than by immediately starting with the coding.

Installing Python

First of all, let's proceed to introduce all the settings you need in order to create a fully working data science environment to test the examples and experiment with the code that we are going to provide you with.

Python is an open source, object-oriented, cross-platform programming language that, compared to its direct competitors (for instance, C++ and Java), is very concise. It allows you to build a working software prototype in a very short time. Did it become the most used language in the data scientist's toolbox just because of this? Well, no. It's also a general-purpose language, and it is very flexible indeed due to a large variety of available packages that solve a wide spectrum of problems and necessities.

Python 2 or Python 3?

There are two main branches of Python: 2 and 3. Although the third version is the newest, the *older* one is still the most used version in the scientific area, since a few libraries (see `http://py3readiness.org` for a compatibility overview) won't run otherwise. In fact, if you try to run some code developed for Python 2 with a Python 3 interpreter, it won't work. Major changes have been made to the newest version, and this has impacted past compatibility. So, please remember that there is no backward compatibility between Python 3 and 2.

In this book, in order to address a larger audience of readers and practitioners, we're going to adopt the Python 2 syntax for all our examples (at the time of writing this book, the latest release is 2.7.8). Since the differences amount to really minor changes, advanced users of Python 3 are encouraged to adapt and optimize the code to suit their favored version.

Step-by-step installation

Novice data scientists who have never used Python (so, we figured out that they don't have it readily installed on their machines) need to first download the installer from the main website of the project, `https://www.python.org/downloads/`, and then install it on their local machine.

> This section provides you with full control over what can be installed on your machine. This is very useful when you have to set up single machines to deal with different tasks in data science. Anyway, please be warned that a step-by-step installation really takes time and effort. Instead, installing a ready-made scientific distribution will lessen the burden of installation procedures and it may be well suited for first starting and learning because it saves you time and sometimes even trouble, though it will put a large number of packages (and we won't use most of them) on your computer all at once. Therefore, if you want to start immediately with an easy installation procedure, just skip this part and proceed to the next section, *Scientific distributions*.

Being a multiplatform programming language, you'll find installers for machines that either run on Windows or Unix-like operating systems. Please remember that some Linux distributions (such as Ubuntu) have Python 2 packeted in the repository, which makes the installation process even easier.

1. To open a python shell, type `python` in the terminal or click on the Python icon.
2. Then, to test the installation, run the following code in the Python interactive shell or REPL:

   ```
   >>> import sys
   >>> print sys.version_info
   ```

3. If a syntax error is raised, it means that you are running Python 3 instead of Python 2. Otherwise, if you don't experience an error and you can read that your Python version has the `attribute major=2`, then congratulations for running the right version of Python. You're now ready to move forward.

To clarify, when a command is given in the terminal command line, we prefix the command with `$>`. Otherwise, if it's for the Python REPL, it's preceded by `>>>`.

A glance at the essential Python packages

We mentioned that the two most relevant Python characteristics are its ability to integrate with other languages and its mature package system that is well embodied by PyPI (the Python Package Index; `https://pypi.python.org/pypi`), a common repository for a majority of Python packages.

The packages that we are now going to introduce are strongly analytical and will offer a complete Data Science Toolbox made up of highly optimized functions for working, optimal memory configuration, ready to achieve scripting operations with optimal performance. A walkthrough on how to install them is given in the following section.

Partially inspired by similar tools present in R and MATLAB environments, we will together explore how a few selected Python commands can allow you to efficiently handle data and then explore, transform, experiment, and learn from the same without having to write too much code or reinvent the wheel.

NumPy

NumPy, which is Travis Oliphant's creation, is the true analytical workhorse of the Python language. It provides the user with multidimensional arrays, along with a large set of functions to operate a multiplicity of mathematical operations on these arrays. Arrays are blocks of data arranged along multiple dimensions, which implement mathematical vectors and matrices. Arrays are useful not just for storing data, but also for fast matrix operations (vectorization), which are indispensable when you wish to solve ad hoc data science problems.

- **Website**: `http://www.numpy.org/`
- **Version at the time of print**: 1.9.1
- **Suggested install command**: `pip install numpy`

As a convention largely adopted by the Python community, when importing NumPy, it is suggested that you alias it as np:

```
import numpy as np
```

We will be doing this throughout the course of this book.

SciPy

An original project by Travis Oliphant, Pearu Peterson, and Eric Jones, SciPy completes NumPy's functionalities, offering a larger variety of scientific algorithms for linear algebra, sparse matrices, signal and image processing, optimization, fast Fourier transformation, and much more.

- **Website**: `http://www.scipy.org/`
- **Version at time of print**: 0.14.0
- **Suggested install command**: `pip install scipy`

pandas

The pandas package deals with everything that NumPy and SciPy cannot do. Thanks to its specific object data structures, DataFrames and Series, pandas allows you to handle complex tables of data of different types (which is something that NumPy's arrays cannot do) and time series. Thanks to Wes McKinney's creation, you will be able to easily and smoothly load data from a variety of sources. You can then slice, dice, handle missing elements, add, rename, aggregate, reshape, and finally visualize this data at your will.

- **Website**: `http://pandas.pydata.org/`
- **Version at the time of print**: 0.15.2
- **Suggested install command**: `pip install pandas`

Conventionally, pandas is imported as `pd`:

```
import pandas as pd
```

Scikit-learn

Started as part of the SciKits (SciPy Toolkits), Scikit-learn is the core of data science operations on Python. It offers all that you may need in terms of data preprocessing, supervised and unsupervised learning, model selection, validation, and error metrics. Expect us to talk at length about this package throughout this book. Scikit-learn started in 2007 as a Google Summer of Code project by David Cournapeau. Since 2013, it has been taken over by the researchers at INRA (French Institute for Research in Computer Science and Automation).

- **Website**: `http://scikit-learn.org/stable/`
- **Version at the time of print**: 0.15.2
- **Suggested install command**: `pip install scikit-learn`

Note that the imported module is named `sklearn`.

IPython

A scientific approach requires the fast experimentation of different hypotheses in a reproducible fashion. IPython was created by Fernando Perez in order to address the need for an interactive Python command shell (which is based on shell, web browser, and the application interface), with graphical integration, customizable commands, rich history (in the JSON format), and computational parallelism for an enhanced performance. IPython is our favored choice throughout this book, and it is used to clearly and effectively illustrate operations with scripts and data and the consequent results.

- **Website**: `http://ipython.org/`
- **Version at the time of print**: 2.3
- **Suggested install** command: `pip install "ipython[notebook]"`

Matplotlib

Originally developed by John Hunter, matplotlib is the library that contains all the building blocks that are required to create quality plots from arrays and to visualize them interactively.

You can find all the MATLAB-like plotting frameworks inside the pylab module.

- **Website**: `http://matplotlib.org/`
- **Version at the time of print**: 1.4.2
- **Suggested install** command: `pip install matplotlib`

You can simply import what you need for your visualization purposes with the following command:

```
import matplotlib.pyplot as plt
```

Downloading the example code

You can download the example code files from your account at `http://www.packtpub.com` for all the Packt Publishing books you have purchased. 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.

Statsmodels

Previously part of SciKits, statsmodels was thought to be a complement to SciPy statistical functions. It features generalized linear models, discrete choice models, time series analysis, and a series of descriptive statistics as well as parametric and nonparametric tests.

- **Website**: `http://statsmodels.sourceforge.net/`
- **Version at the time of print**: 0.6.0
- **Suggested install command**: `pip install statsmodels`

Beautiful Soup

Beautiful Soup, a creation of Leonard Richardson, is a great tool to scrap out data from HTML and XML files retrieved from the Internet. It works incredibly well, even in the case of *tag soups* (hence the name), which are collections of malformed, contradictory, and incorrect tags. After choosing your parser (basically, the HTML parser included in Python's standard library works fine), thanks to Beautiful Soup, you can navigate through the objects in the page and extract text, tables, and any other information that you may find useful.

- **Website**: `http://www.crummy.com/software/BeautifulSoup/`
- **Version at the time of print**: 4.3.2
- **Suggested install command**: `pip install beautifulsoup4`

Note that the imported module is named `bs4`.

NetworkX

Developed by the Los Alamos National Laboratory, NetworkX is a package specialized in the creation, manipulation, analysis, and graphical representation of real-life network data (it can easily operate with graphs made up of a million nodes and edges). Besides specialized data structures for graphs and fine visualization methods (2D and 3D), it provides the user with many standard graph measures and algorithms, such as the shortest path, centrality, components, communities, clustering, and PageRank. We will frequently use this package in *Chapter 5, Social Network Analysis*.

- **Website**: `https://networkx.github.io/`
- **Version at the time of print**: 1.9.1
- **Suggested install command**: `pip install networkx`

Conventionally, NetworkX is imported as `nx`:

```
import networkx as nx
```

NLTK

The **Natural Language Toolkit** (**NLTK**) provides access to corpora and lexical resources and to a complete suit of functions for statistical **Natural Language Processing** (**NLP**), ranging from tokenizers to part-of-speech taggers and from tree models to named-entity recognition. Initially, the package was created by Steven Bird and Edward Loper as an NLP teaching infrastructure for CIS-530 at the University of Pennsylvania. It is a fantastic tool that you can use to prototype and build NLP systems.

- **Website**: `http://www.nltk.org/`
- **Version at the time of print**: 3.0
- **Suggested install command**: `pip install nltk`

Gensim

Gensim, programmed by Radim Řehůřek, is an open source package that is suitable for the analysis of large textual collections with the help of parallel distributable online algorithms. Among advanced functionalities, it implements **Latent Semantic Analysis** (**LSA**), topic modeling by **Latent Dirichlet Allocation** (**LDA**), and Google's *word2vec*, a powerful algorithm that transforms text into vector features that can be used in supervised and unsupervised machine learning.

- **Website**: `http://radimrehurek.com/gensim/`
- **Version at the time of print**: 0.10.3
- **Suggested install command**: `pip install gensim`

PyPy

PyPy is not a package; it is an alternative implementation of Python 2.7.8 that supports most of the commonly used Python standard packages (unfortunately, NumPy is currently not fully supported). As an advantage, it offers enhanced speed and memory handling. Thus, it is very useful for heavy duty operations on large chunks of data and it should be part of your big data handling strategies.

- **Website**: `http://pypy.org/`
- **Version at time of print**: 2.4.0
- **Download page**: `http://pypy.org/download.html`

The installation of packages

Python won't come bundled with all you need, unless you take a specific premade distribution. Therefore, to install the packages you need, you can either use `pip` or `easy_install`. These are the two tools that run in the command line and make the process of installation, upgrade, and removal of Python packages a breeze. To check which tools have been installed on your local machine, run the following command:

```
$> pip
```

Alternatively, you can also run the following command:

```
$> easy_install
```

If both these commands end with an error, you need to install any one of them. We recommend that you use pip because it is thought of as an improvement over `easy_install`. By the way, packages installed by `pip` can be uninstalled and if, by chance, your package installation fails, `pip` will leave your system clean.

To install pip, follow the instructions given at `https://pip.pypa.io/en/latest/installing.html`.

The most recent versions of Python should already have pip installed by default. So, you may have it already installed on your system. If not, the safest way is to download the `get-pi.py` script from `https://bootstrap.pypa.io/get-pip.py` and then run it using the following:

```
$> python get-pip.py
```

The script will also install the setup tool from `https://pypi.python.org/pypi/setuptools`, which also contains `easy_install`.

You're now ready to install the packages you need in order to run the examples provided in this book. To install the generic package `<pk>`, you just need to run the following command:

```
$> pip install <pk>
```

Alternatively, you can also run the following command:

```
$> easy_install <pk>
```

After this, the package `<pk>` and all its dependencies will be downloaded and installed. If you're not sure whether a library has been installed or not, just try to import a module inside it. If the Python interpreter raises an `ImportError` error, it can be concluded that the package has not been installed.

This is what happens when the NumPy library has been installed:

```
>>> import numpy
```

This is what happens if it's not installed:

```
>>> import numpy
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named numpy
```

In the latter case, you'll need to first install it through `pip` or `easy_install`.

> Take care that you don't confuse packages with modules. With pip, you install a package; in Python, you import a module. Sometimes, the package and the module have the same name, but in many cases, they don't match. For example, the sklearn module is included in the package named Scikit-learn.

Finally, to search and browse the Python packages available for Python, take a look at `https://pypi.python.org`.

Package upgrades

More often than not, you will find yourself in a situation where you have to upgrade a package because the new version is either required by a dependency or has additional features that you would like to use. First, check the version of the library you have installed by glancing at the `__version__` attribute, as shown in the following example, `numpy`:

```
>>> import numpy
>>> numpy.__version__ # 2 underscores before and after
'1.9.0'
```

Now, if you want to update it to a newer release, say the 1.9.1 version, you can run the following command from the command line:

```
$> pip install -U numpy==1.9.1
```

Alternatively, you can also use the following command:

```
$> easy_install --upgrade numpy==1.9.1
```

Finally, if you're interested in upgrading it to the latest available version, simply run the following command:

```
$> pip install -U numpy
```

You can alternatively also run the following command:

```
$> easy_install --upgrade numpy
```

Scientific distributions

As you've read so far, creating a working environment is a time-consuming operation for a data scientist. You first need to install Python and then, one by one, you can install all the libraries that you will need (sometimes, the installation procedures may not go as smoothly as you'd hoped for earlier).

If you want to save time and effort and want to ensure that you have a fully working Python environment that is ready to use, you can just download, install, and use the scientific Python distribution. Apart from Python, they also include a variety of preinstalled packages, and sometimes, they even have additional tools and an IDE. A few of them are very well known among data scientists, and in the sections that follow, you will find some of the key features of each of these packages.

We suggest that you first promptly download and install a scientific distribution, such as Anaconda (which is the most complete one), and after practicing the examples in the book, decide to fully uninstall the distribution and set up Python alone, which can be accompanied by just the packages you need for your projects.

Anaconda

Anaconda (`https://store.continuum.io/cshop/anaconda`) is a Python distribution offered by Continuum Analytics that includes nearly 200 packages, which include NumPy, SciPy, pandas, IPython, Matplotlib, Scikit-learn, and NLTK. It's a cross-platform distribution that can be installed on machines with other existing Python distributions and versions, and its base version is free. Additional add-ons that contain advanced features are charged separately. Anaconda introduces `conda`, a binary package manager, as a command-line tool to manage your package installations. As stated on the website, Anaconda's goal is to provide enterprise-ready Python distribution for large-scale processing, predictive analytics and scientific computing.

Enthought Canopy

Enthought Canopy (`https://www.enthought.com/products/canopy/`) is a Python distribution by Enthought, Inc. It includes more than 70 preinstalled packages, which include NumPy, SciPy, Matplotlib, IPython, and pandas. This distribution is targeted at engineers, data scientists, quantitative and data analysts, and enterprises. Its base version is free (which is named Canopy Express), but if you need advanced features, you have to buy a front version. It's a multiplatform distribution and its command-line install tool is `canopy_cli`.

PythonXY

PythonXY (`https://code.google.com/p/pythonxy/`) is a free, open source Python distribution maintained by the community. It includes a number of packages, which include NumPy, SciPy, NetworkX, IPython, and Scikit-learn. It also includes Spyder, an interactive development environment inspired by the MATLAB IDE. The distribution is free. It works only on Microsoft Windows, and its command-line installation tool is pip.

WinPython

WinPython (`http://winpython.sourceforge.net`) is also a free, open-source Python distribution maintained by the community. It is designed for scientists, and includes many packages such as NumPy, SciPy, Matplotlib, and IPython. It also includes Spyder as an IDE. It is free and portable (you can put it in any directory, or even in a USB flash drive). It works only on Microsoft Windows, and its command-line tool is the **WinPython Package Manager** (**WPPM**).

Introducing IPython

IPython is a special tool for interactive tasks, which contains special commands that help the developer better understand the code that they are currently writing. These are the commands:

- `<object>? and <object>??`: This prints a detailed description (with `??` being even more verbose) of the `<object>`
- `%<function>`: This uses the special `<magic function>`

Let's demonstrate the usage of these commands with an example. We first start the interactive console with the `ipython` command that is used to run IPython, as shown here:

```
$> ipython
Python 2.7.6 (default, Sep  9 2014, 15:04:36)
Type "copyright", "credits" or "license" for more information.
IPython 2.3.1 -- 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.
In [1]: obj1 = range(10)
```

Then, in the first line of code, which is marked by IPython as `[1]`, we create a list of 10 numbers (from 0 to 9), assigning the output to an object named `obj1`:

```
In [2]: obj1?
Type:        list
String form: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Length:      10
Docstring:
list() -> new empty list
list(iterable) -> new list initialized from iterable's items
In [3]: %timeit x=100
10000000 loops, best of 3: 23.4 ns per loop
In [4]: %quickref
```

In the next line of code, which is numbered `[2]`, we inspect the `obj1` object using the IPython command `?`. IPython introspects the object, prints its details (`obj` is a list that contains the values `[1, 2, 3..., 9]` and elements), and finally prints some general documentation on lists. It's not the case in this example. However, for complex objects, the usage of `??`instead of `?`gives a more verbose output.

In line `[3]`, we use the magic function `timeit` to a Python assignment (`x=100`). The `timeit` function runs this instruction many times and stores the computational time needed to execute it. Finally, it prints the average time that was taken to run the Python function.

We complete the overview with a list of all the possible IPython special functions by running the helper function `quickref`, as shown in line `[4]`.

As you noticed, each time we use IPython, we have an input cell and optionally, an output cell, if there is something that has to be printed on `stdout`. Each input is numbered, so it can be referenced inside the IPython environment itself. For our purposes, we don't need to provide such references in the code of the book. Therefore, we will just report inputs and outputs without their numbers. However, we'll use the generic `In:` and `Out:` notations to point out the input and output cells. Just copy the commands after `In:` to your own IPython cell and expect an output that will be reported on the following `Out:`.

Therefore, the basic notations will be:

- The `In:` command
- The `Out:` output (wherever it is present and useful to be reported in the book)

Otherwise, if we expect you to operate directly on the Python console, we will use the following form:

```
>>> command
```

Wherever necessary, the command-line input and output will be written as follows:

```
$> command
```

Moreover, to run the `bash` command in the IPython console, prefix it with a "`!`" (an exclamation mark):

```
In: !ls
Applications    Google Drive    Public          Desktop
Develop
Pictures        env             temp
...
In: !pwd
/Users/mycomputer
```

The IPython Notebook

The main goal of the IPython Notebook is easy storytelling. Storytelling is essential in data science because you must have the power to do the following:

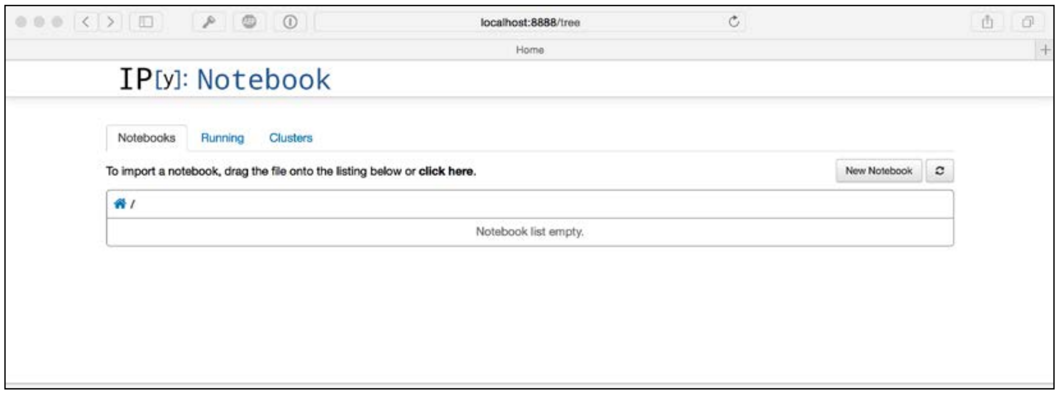
- See intermediate (debugging) results for each step of the algorithm you're developing
- Run only some sections (or cells) of the code
- Store intermediate results and have the ability to version them
- Present your work (this will be a combination of text, code, and images)

Here comes IPython; it actually implements all the preceding actions.

1. To launch the IPython Notebook, run the following command:

   ```
   $> ipython notebook
   ```

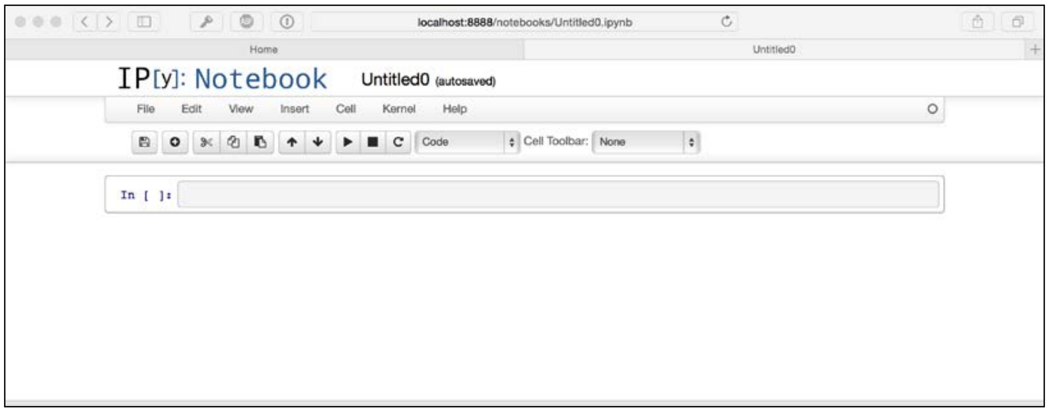
2. A web browser window will pop up on your desktop, backed by an IPython server instance. This is the how the main window looks:



To import a notebook, drag the file onto the listing below or **click here**.

Notebook list empty.

3. Then, click on **New Notebook**. A new window will open, as shown in the following screenshot:



This is the web app that you'll use to compose your story. It's very similar to a Python IDE, with the bottom section (where you can write the code) composed of cells.

A cell can be either a piece of text (eventually formatted with a markup language) or a piece of code. In the second case, you have the ability to run the code, and any eventual output (the standard output) will be placed under the cell. The following is a very simple example of the same: