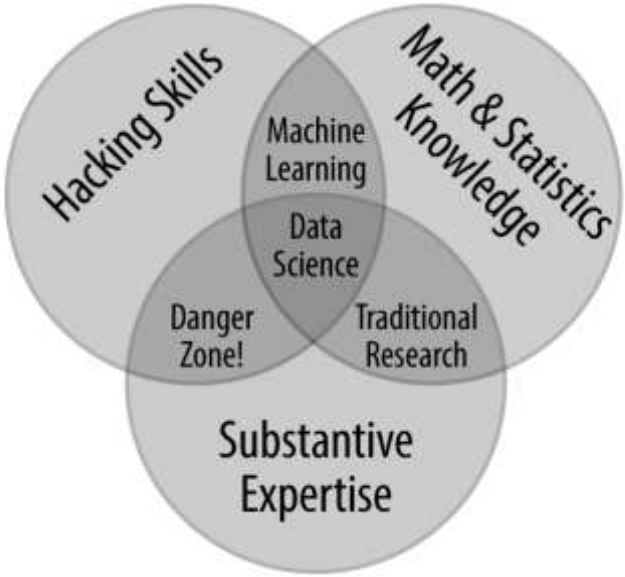
Preface

What Is Data Science?

This is a book about doing data science with Python, which immediately begs the question: what is *data science*? It's a surprisingly hard definition to nail down, especially given how ubiquitous the term has become. Vocal critics have variously dismissed it as a superfluous label (after all, what science doesn't involve data?) or a simple buzzword that only exists to salt resumes and catch the eye of overzealous tech recruiters.

In my mind, these critiques miss something important. Data science, despite its hype-laden veneer, is perhaps the best label we have for the cross-disciplinary set of skills that are becoming increasingly important in many applications across industry and academia. This *cross-disciplinary* piece is key: in my mind, the best existing definition of data science is illustrated by Drew Conway's Data Science Venn Diagram, first published on his blog in September 2010 (see the following figure).



(source: Drew Conway (http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram), used by permission)

While some of the intersection labels are a bit tongue-in-cheek, this diagram captures the essence of what I think people mean when they say "data science": it is fundamentally an interdisciplinary subject. Data science comprises three distinct and overlapping areas: the skills of a *statistician* who knows how to model and summarize datasets (which are growing ever larger); the skills of a *computer scientist* who can design and use algorithms to efficiently store, process, and visualize this data; and the *domain expertise*—what we might think of as "classical" training in a subject—necessary both to formulate the right questions and to put their answers in context.

With this in mind, I would encourage you to think of data science not as a new domain of knowledge to learn, but a new set of skills that you can apply within your current area of expertise. Whether you are reporting election results, forecasting stock returns, optimizing online ad clicks, identifying microorganisms in microscope photos, seeking new classes of astronomical objects, or working with data in any other field, the goal of this book is to give you

Who Is This Book For?

In my teaching both at the University of Washington and at various tech-focused conferences and meetups, one of the most common questions I have heard is this: "How should I learn Python?" The people asking are generally technically minded students, developers, or researchers, often with an already strong background in writing code and using computational and numerical tools. Most of these folks don't want to learn Python per se, but want to learn the language with the aim of using it as a tool for data-intensive and computational science. While a large patchwork of videos, blog posts, and tutorials for this audience is available online, I've long been frustrated by the lack of a single good answer to this question; that is what inspired this book.

The book is not meant to be an introduction to Python or to programming in general; I assume the reader has familiarity with the Python language, including defining functions, assigning variables, calling methods of objects, controlling the flow of a program, and other basic tasks. Instead, it is meant to help Python users learn to use Python's data science stack—libraries such as those mentioned in the following section, and related tools—to effectively store, manipulate, and gain insight from data.

Why Python?

Python has emerged over the last couple of decades as a first-class tool for scientific computing tasks, including the analysis and visualization of large datasets. This may have come as a surprise to early proponents of the Python language: the language itself was not specifically designed with data analysis or scientific computing in mind. The usefulness of Python for data science stems primarily from the large and active ecosystem of third-party packages: *NumPy* for manipulation of homogeneous array-based data, *Pandas* for manipulation of heterogeneous and labeled data, *SciPy* for common scientific computing tasks, *Matplotlib* for publication-quality visualizations, *IPython* for interactive execution and sharing of code, *Scikit-Learn* for machine learning, and many more tools that will be mentioned in the following pages.

If you are looking for a guide to the Python language itself, I would suggest the sister project to this book, [https://www.oreilly.com/library/view/a-whirlwind-tour/9781492037859](https://www.oreilly.com/library/view/a-whirlwind-tour/9781492037859%5D)(_A Whirlwind Tour of the Python Language_). This short report provides a tour of the essential features of the Python language, aimed at data scientists who already are familiar with one or more other programming languages.

Outline of the Book

Each numbered part of this book focuses on a particular package or tool that contributes a fundamental piece of the Python data science story, and is broken into short self-contained chapters that each discuss a single concept:

- *Part I, Jupyter: Beyond Normal Python*, introduces IPython and Jupyter. These packages provide the computational environment in which many Python-using data scientists work.
- *Part II, Introduction to NumPy*, focuses on the NumPy library, which provides the `ndarray` for efficient storage and manipulation of dense data arrays in Python.
- *Part III, Data Manipulation with Pandas*, introduces the Pandas library, which provides the `DataFrame` for efficient storage and manipulation of labeled/columnar data in Python.
- *Part IV, Visualization with Matplotlib*, concentrates on Matplotlib, a library that provides capabilities for a flexible range of data visualizations in Python.
- *Part V, Machine Learning*, focuses on the Scikit-Learn library, which provides efficient and clean Python implementations of the most important and established machine learning algorithms.

The PyData world is certainly much larger than these six packages, and is growing every day. With this in mind, I make every attempt throughout this book to provide references to other interesting efforts, projects, and packages that are pushing the boundaries of what can be done in Python. Nevertheless, the packages I concentrate on are currently fundamental to much of the work being done in the Python data science space, and I expect they will remain important even as the ecosystem continues growing around them.

Using Code Examples

Supplemental material (code examples, figures, etc.) is available for download at http://github.com/jakevdp/PythonDataScienceHandbook/ (http://github.com/jakevdp/PythonDataScienceHandbook/). This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Python Data Science Handbook*, 2nd edition, by Jake VanderPlas (O'Reilly). Copyright 2023 Jake VanderPlas, 978-1-098-12122-8."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com (mailto:permissions@oreilly.com).

Installation Considerations

Installing Python and the suite of libraries that enable scientific computing is straightforward. This section will outline some of the things to keep in mind when setting up your computer.

Though there are various ways to install Python, the one I would suggest for use in data science is the Anaconda distribution, which works similarly whether you use Windows, Linux, or macOS. The Anaconda distribution comes in two flavors:

- Miniconda (http://conda.pydata.org/miniconda.html) gives you the Python interpreter itself, along with a command-line tool called *conda* which operates as a cross-platform package manager geared toward Python packages, similar in spirit to the apt or yum tools that Linux users might be familiar with.
- Anaconda (https://www.continuum.io/downloads) includes both Python and conda, and additionally bundles a suite of other preinstalled packages geared toward scientific computing. Because of the size of this bundle, expect the installation to consume several gigabytes of disk space.

Any of the packages included with Anaconda can also be installed manually on top of Miniconda; for this reason I suggest starting with Miniconda.

To get started, download and install the Miniconda package—make sure to choose a version with Python 3—and then install the core packages used in this book:

```
[~]$ conda install numpy pandas scikit-learn matplotlib seaborn jupyt
er
```

Throughout the text, we will also make use of other more specialized tools in Python's scientific ecosystem; installation is usually as easy as typing `conda install packagename`. If you ever come across packages that are not available in the default conda channel, be sure to check out *conda-forge* (https://conda-forge.org/), a broad, community-driven repository of conda packages.

For more information on conda, including information about creating and using conda environments (which I would *highly* recommend), refer to conda's online documentation (http://conda.pydata.org/docs/).