# BEFORE YOU BEGIN

## Introduction

The goal of this book is to show you how to use the many tools you already have on your computer to make it do more work for you. You will learn skills for shaving minutes or hours off many small tasks—and because these skills are scalable, you will also be able to use them to accomplish things that would take you many weeks or even years to do by hand. The focus is on general solutions applicable to a range of problems, rather than on cookbook recipes not useful in any other context. In addition to learning new skills, you will begin to recognize when to apply these tools to make your analyses easier. Once you are familiar with the basic tools, you will learn how to combine them to improve the efficiency, flexibility, and reproducibility of your overall workflow. In some cases, you will even be able to ferry your data through a series of analyses without touching a keyboard.

The few computer books that have been written with biologists in mind focus mainly on programming, in particular as it is relevant to bioinformatics. This is not a book on bioinformatics; it takes a broader approach, both in the range of tools and the variety of applications. As you will see, effective computing can benefit many more aspects of science than just molecular sequence analysis.

## Why this book?

There are already many books dedicated to each of the different subjects we cover here, including programming, databases, graphics, and Unix commands. Most of these texts aim to be comprehensive, providing a thorough background on the concepts, theory, and history of the technologies they cover. However, we know many frustrated biologists and eager students to whom these in-depth books are of little use without an initial grounding and a practical overview. In our experience, 90% of the computational problems biologists face can be addressed with less than 10% of the commonly available tools, but a typical scientist has no idea which 10%. This is why we decided to write a problem-centric book to comple-

ment the many technology-centric books that already exist—we wanted to create a resource from the perspective of the life-scientist user base.

In the short term, writing a special program to help with your data analysis may seem like a big investment in time and effort, especially when the choice is between spending six or so hours to write a script, or just working through the dataset in two hours of copying and pasting. In the long run, however, an investment in script-writing quickly pays off, since it subsequently takes just minutes to process hundreds of files. You also have the benefit of rapidly being able to redo your analyses when more samples are added to the dataset.[1] The process of generating tools for analysis becomes easier each time you do it, and as you become more proficient with the languages and tools, eventually you will be able to write a script faster than it would take to process even a single file by hand. You also quickly accumulate a set of programs that take relatively little time to adapt to new purposes.

Learning to write programs is only one aspect of being able to use computing for your research. The more general challenge is to figure out when and how to apply each of the many tools available as you move from data collection to analysis and presentation. Recognizing that a nail needs a hammer and a bolt calls for a wrench is just as important as having these tools in your toolbox and knowing how to use them. This broader perspective is addressed with a flowchart that helps you with the tool-selection process, as well as in a chapter on organizing your dataset before you even start collecting it.

## Why biologists?

We chose to focus on biologists rather than scientists-at-large for a few reasons. We ourselves are biologists who also happen to have backgrounds in computing. This book grew out of years of developing tools for our own research and helping other biologists with their computational problems. Although the methods and background we provide would be just as useful to scientists in other disciplines, as well as to engineers and other professionals, we think it best to stick to the audience we've been serving all along. We also want the examples to represent familiar problems, so that the reader will have a concrete idea of how the material applies to issues they routinely face, even before they have to consider technical details.

Biology is becoming far more computationally intensive, yet undergraduate and graduate biology curricula have not kept up with this new reality. As a result, many biologists find themselves lacking the training and information-processing skills needed to tackle exponentially expanding datasets and more complex analyses. This text aims to fill that gap by providing training that is increasingly important for scientific success. Ultimately, we hope it will encourage biology departments to develop standard computing courses for their students.

---

[1] As our colleague Sönke Johnsen likes to say, "There's a reason they call it re-search."

## Is this about using a particular computer or program?

This book is about using computers practically. We focus on general, flexible, scalable tools, rather than on telling you how to click through the menus of a particular version of a program to accomplish a given task. We concentrate on the technologies used most frequently within the biology community, in order to maximize your ability to understand, recycle, and reuse tools that have already been developed by other life scientists. We also consistently lean towards open source software, largely because of its availability, flexibility, and transparency.

Some of the biggest data analysis challenges faced by biologists are in figuring out how to organize their data and workflows, and these issues have nothing to do with the type of computer you use. However, it would have been too cumbersome for this book to walk through each example with specifics for more than one type of computer or operating system. We have therefore used Apple's OS X as the primary setting for our examples and procedures. OS X is an easily accessible Unix variant. Unix itself has been around for forty years now, and is here to stay. It has long been the standard for demanding scientific analyses, and can now be used on any personal computer.

In addition, Appendix 1 provides extensive detail on how to install and use the tools presented here on computers running either Microsoft Windows or Linux. Microsoft Windows is not based on Unix, so many details are slightly different and some topics—for example, using the command line—require work-arounds. Linux, like OS X, is based on Unix, so in this case, the differences relevant to this book are minimal.[2] Appendix 1 also explains how to install Linux on a Windows computer, which can be a good option for those who have a Windows computer but want to test the Unix waters.

One of the most flexible and powerful skills you will learn in this book is how to write a program. There are two principal aspects of programming: the conceptual building blocks (loops, decisions, arrays, input, output), and the language-specific syntax (the words you type to embody those concepts). Although most beginning programmers are concerned about the syntax, the building blocks are far more important: once you become comfortable with them, you can quickly adjust to any programming language. For this book we have chosen Python as our demonstration language, for several reasons. It is powerful, yet easy to read. It is growing in popularity, and is rapidly being adopted as the standard for bioinformatics, Web development, and introductory programming courses. While your fundamental programming ability will never go out of style, the Python syntax you learn should also serve you well for many years to come.

---

[2] Unix operating systems are used "behind the scenes" in many household devices, including DVRs and the iPhone.

## To readers who will use this book on their own

We expect that many biologists will use this book to improve the efficiency of their research, help scale up existing projects, or develop the skills needed for new types of studies. The book is designed to serve these independent readers, and in fact, our own graduate students and colleagues have used chapters of the manuscript in this way. Much of what we ourselves use in practice was garnered through self-directed experience, and we have tried to collect this knowledge in one place to make it easier for other scientists to do the same.

Some recommendations for working through the book: Try out as many examples as you can and explore beyond their immediate scope as described in the text. Also, start keeping a file on your computer of useful commands and things that you discover, to help you generate your own quick reference card.

When something doesn't work as expected, it can be difficult to troubleshoot in isolation. If you come to a point where you can't sort things out on your own, take advantage of our online community at `http://practicalcomputing.org`. By reading related posts and asking other readers for advice, you can gain a sympathetic ear and get a quicker resolution for your setback. You can provide feedback about what you find most versus least useful in the book, and let us know of errors that may have crept into the text. Finally, you can exchange ideas about applying your new skills to your research.

## To teachers using this book

Although the book is written to stand on its own outside the classroom, it is also designed to be used as a primary or secondary text within the classroom. We use it as a textbook by having students read thematically connected chapters ahead of time. Then, in class, we walk through examples in more detail, and expand on the content in the areas the students find either most confusing or most relevant to their own interests. It helps to keep in mind that students who are knowledgeable and confident when discussing biology may feel intimidated or frustrated by computing concepts, and they may interact differently because they are sensitive to being found "ignorant."

In a classroom setting where students have different types of computers, you will also have to determine how you want to achieve operating system independence. Many of the lessons are platform-independent, but Chapters 4 through 6 assume a Unix command line is available. Windows users might at first feel slighted by these sections, but some general solutions are provided in Appendix 1. Ultimately, you will have to choose whether you conduct the class in a lab with OS X machines available to all; request that students with Windows computers install Linux (which is much simpler now with the widespread use of virtual machines,

as described in Appendix 1); or have students install a software package (Cygwin) that gives them some Unix functionality (also described in Appendix 1).

The book is organized into sections, but only parts I , II, and III need to be covered in sequence; the skills taught there not only build on each other, but provide a foundation for the entire book. Beyond that, you are free to jump around as you see fit. For example, Chapters 4–6 on the Unix shell fit well with more detailed shell material from Chapters 16 and 20. The three chapters on graphics can be covered at any time.

While we give examples and opportunities for practicing new skills as they are developed, we do not provide formal exercises or test questions. There are so many diverse data-analysis issues in biology that it should not be difficult to generate exercises which are especially relevant to your students' subdisciplines, whether molecular, ecological, or biochemical.

## Beyond this book

We hope that this book helps you start solving many problems immediately, as well as showing you how to do things you didn't know how to do before. Beyond that, it is also intended as a stepping-stone to more advanced topics. The long-term goal is to empower you in the use of your computer in such a way that you know how to continue growing, training yourself with both other books and Internet resources. We have found that not knowing where to get started is a major roadblock for many biologists facing new computational challenges, and that once they are pointed in the right direction, they quickly gain traction and expertise. Advice on additional reading is provided at the end of many chapters.

It is our hope that by showing biologists how to apply the powerful tools already installed on their computers, they will be free to spend more time on the pleasures of scientific inquiry and less time on the frustrating aspects of data processing.

## How to use this book

You can, of course, pick and choose the parts of this book that you feel are most relevant. However, we recommend that you follow the chapters of Parts I through III in the order that they are presented. The later chapters within those sections assume understanding of the concepts and skills (and in some cases, the system settings) described in earlier chapters. Part IV takes a broader view of ways to combine tools into processing pipelines and techniques for working with larger datasets. Part V, on graphics, is most relevant when preparing figures for publication or presentation. Part VI includes more specialized topics, which may or may not be relevant to your particular needs. The appendices serve as reference material, largely for the tools described in Parts I through III.

Throughout the book, graphical elements help to highlight certain sections. Icons in the margin point out passages of the book which may be of special interest or importance:


A tip to make your work easier


A potential pitfall that might trip you up


A pitfall that might cause widespread failure


Sections that vary for Windows users. Refer to Appendix 1 or a marginal note


Sections that vary for Linux users, including Linux via Virtual Box within Windows

Text is also formatted to indicate its usage:

System items include files and folders, program names, and menu commands

`Code characters` indicate programming language and terminal commands. The background color indicates whether they are from a `terminal window` or part of a script in a `text document`.

We have customized the `Courier` typeface used to represent code characters to more clearly distinguish between the number `1`, lowercase `l`, and uppercase `I`, as well as between the number `0` and the capital letter `O`. At places in the text, a tab character is represented by the symbol ▵. Further information can be found at the companion site `http://practicalcomputing.org/`.