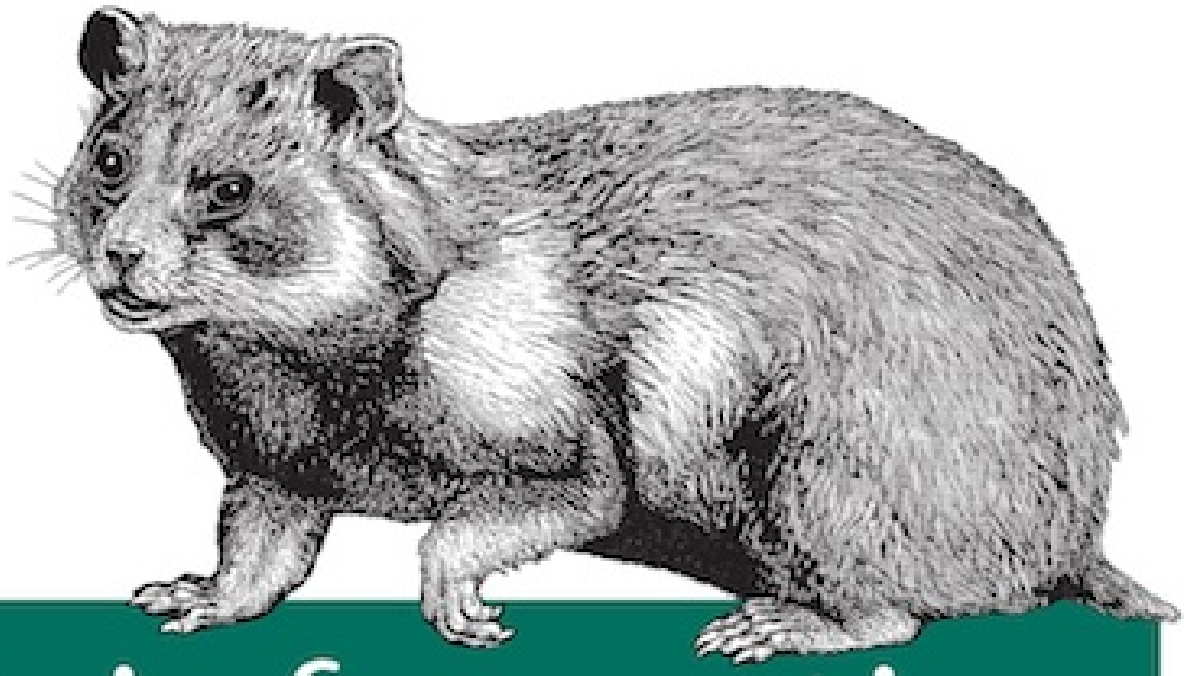
O'REILLY®



Bioinformatics Data Skills

REPRODUCIBLE AND ROBUST RESEARCH WITH OPEN SOURCE TOOLS

Vince Buffalo

Bioinformatics Data Skills

Vince Buffalo

Bioinformatics Data Skills

by Vince Buffalo

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (http://safaribooksonline.com). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

- Editors: Courtney Nash and Amy Jollymore
- Production Editor: Nicole Shelby
- Copyeditor: Jasmine Kwityn
- Proofreader: Kim Cofer
- Indexer: Ellen Troutman
- Interior Designer: David Futato
- Cover Designer: Ellie Volckhausen
- Illustrator: Rebecca Demarest
- June 2015: First Edition

Revision History for the First Edition

- 2015-06-30: First Release

See http://oreilly.com/catalog/errata.csp?isbn=9781449367374 for release details.

978-1-449-36737-4

[LSI]

Dedication

To my (rather large) family for their continued support: Mom, Dad, Anne, Lisa, Lauren, Violet, and Dalilah; the Buffalos, the Kihns, and the Lambs.

And my earliest mentors for inspiring me to be who I am today: Randy Siverson and Duncan Temple Lang.

Preface

This book is the answer to a question I asked myself two years ago: "What book would I want to read *first* when getting started in bioinformatics?" When I began working in this field, I had programming experience in Python and R but little else. I had hunted around for a terrific introductory text on bioinformatics, and while I found some good books, most were not targeted to the daily work I did as a bioinformatician. A few of the texts I found approached bioinformatics from a theoretical and algorithmic perspective, covering topics like Smith-Waterman alignment, phylogeny reconstruction, motif finding, and the like. Although they were fascinating to read (and I do recommend that you explore this material), I had no need to implement bioinformatics algorithms from scratch in my daily bioinformatics work—numerous terrific, highly optimized, well-tested implementations of these algorithms already existed. Other bioinformatics texts took a more practical approach, guiding readers unfamiliar with computing through each step of tasks like running an aligner or downloading sequences from a database. While these were more applicable to my work, much of those books' material was outdated.

As you might guess, I couldn't find that best "first" bioinformatics book. *Bioinformatics Data Skills* is my version of the book I was seeking. This book is targeted toward readers who are unsure how to bridge the giant gap between knowing a scripting language and practicing bioinformatics to answer scientific questions in a robust and reproducible way. To bridge this gap, one must learn data skills—an approach that uses a core set of tools to manipulate and explore any data you'll encounter during a bioinformatics project.

Data skills are the best way to learn bioinformatics because these skills utilize time-tested, open source tools that continue to be the best way to manipulate and explore changing data. This approach has stood the test of time: the advent of high-throughput sequencing rapidly changed the field of bioinformatics, yet skilled bioinformaticians adapted to this new data using these same tools and skills. Next-generation data was, after all, just data (different data, and *more* of it), and master bioinformaticians had the essential skills to solve problems by applying their tools to this new data. *Bioinformatics Data Skills* is written to provide you with training in these core tools and help you develop these same skills.

The Approach of This Book

Many biologists starting out in bioinformatics tend to equate "learning bioinformatics" with "learning how to run bioinformatics software." This is an unfortunate and misinformed idea of what bioinformaticians actually do. This is analogous to thinking "learning molecular biology" is just "learning pipetting." Other than a few simple examples used to generate data in Chapter 11, this book doesn't cover running bioinformatics software like aligners, assemblers, or variant callers. Running bioinformatics software isn't all that difficult, doesn't take much skill, and it doesn't embody any of the significant challenges of bioinformatics. I don't teach how to run these types of bioinformatics applications in *Bioinformatics Data Skills* for the following reasons:

- It's easy enough to figure out on your own

- The material would go rapidly out of date as new versions of software or entirely new programs a used in bioinformatics
- The original manuals for this software will always be the best, most up-to-date resource on how to run a program

Instead, the approach of this book is to focus on the skills bioinformaticians use to explore and extrac meaning from complex, large bioinformatics datasets. Exploring and extracting information from these datasets is the fun part of bioinformatics research. The goal of *Bioinformatics Data Skills* is to teach you the computational tools and data skills you need to explore these large datasets as you please. These data skills give you freedom; you'll be able to look at any bioinformatics data—in any format, and files of any size—and begin exploring data to extract biological meaning.

Throughout *Bioinformatics Data Skills*, I emphasize working in a robust and reproducible manner. I believe these two qualities—reproducibility and robustness—are too often overlooked in modern computational work. By *robust*, I mean that your work is resilient against silent errors, confounders, software bugs, and messy or noisy data. In contrast, a fragile approach is one that does not decrease the odds of some type of error adversely affecting your results. By *reproducible*, I mean that your work can be repeated by other researchers and they can arrive at the same results. For this to be the case, your work must be well documented, and your methods, code, and data all need to be available so that other researchers have the materials to reproduce everything. Reproducibility also relies on your work being robust—if a workflow run on a different machine yields a different outcome, it is neither robust nor fully reproducible. I introduce these concepts in more depth in Chapter 2, and thes are themes that reappear throughout the book.

Why This Book Focuses on Sequencing Data

Bioinformatics is a broad discipline, and spans subfields like proteomics, metabolomics, structure bioinformatics, comparative genomics, machine learning, and image processing. *Bioinformatics Dat Skills* focuses primarily on handling sequencing data for a few reasons.

First, sequencing data is abundant. Currently, no other "omics" data is as abundant as high-throughpu sequencing data. Sequencing data has broad applications across biology: variant detection and genotyping, transcriptome sequencing for gene expression studies, protein-DNA interaction assays like ChIP-seq, and bisulfite sequencing for methylation studies just to name a few examples. The wa in which sequencing data can be used to answer biological questions will only continue to increase.

Second, sequencing data is terrific for honing your data skills. Even if your goal is to analyze other types of data in the future, sequencing data serves as great example data to learn with. Developing th text-processing skills necessary to work with sequencing data will be applicable to working with mar other data types.

Third, other subfields of bioinformatics are much more domain specific. The wide availability and declining costs of sequencing have allowed scientists from all disciplines to use genomics data to answer questions in their systems. In contrast, bioinformatics subdisciplines like proteomics or high-throughput image processing are much more specialized and less widespread. Still, if you're interested in these fields, *Bioinformatics Data Skills* will teach you useful computational and data skills that will be helpful in your research.

Audience

In my experience teaching bioinformatics to friends, colleagues, and students of an intensive week-long course taught at UC Davis, most people wishing to learn bioinformatics are either biologists, or computer scientists/programmers. Biologists wish to develop the computational skills necessary to analyze their own data, while the programmers and computer scientists wish to apply their computational skills to biological problems. Although these two groups differ considerably in biological knowledge and computational experience, *Bioinformatics Data Skills* covers material that should be helpful to both.

If you're a biologist, *Bioinformatics Data Skills* will teach you the core data skills you need to work with bioinformatics data. It's important to note that *Bioinformatics Data Skills* is *not* a how-to bioinformatics book; such a book on bioinformatics would quickly go out of date or be too narrow in focus to help the majority of biologists. You will need to supplement this book with knowledge of your specific research and system, as well as the modern statistical and bioinformatics methods that your subfield uses. For example, if your project involves aligning sequencing reads to a reference genome, this book won't tell you the newest and best alignment software for your particular system. But regardless of which aligner you use, you will need to have a thorough understanding of alignmen formats and how to manipulate alignment data—a topic covered in Chapter 11. Throughout this book these general computational and data skills are meant to be a solid, widely applicable foundation on which the majority of biologists can build.

If you're a computer scientist or programmer, you are likely already familiar with some of the

computational tools I teach in this book. While the material presented in *Bioinformatics Data Skills* may overlap knowledge you already have, you will still learn about the specific formats, tools, and approaches bioinformaticians use in their work. Also, working through the examples in this book wil give you good practice in applying your computational skills to genomics data.

The Difficulty Level of *Bioinformatics Data Skills*

Bioinformatics Data Skills is designed to be a thorough—and in parts, dense—book. When I started writing this book, I decided the greatest misdeed I could do would be to treat bioinformatics as a subject that's easier than it truly is. Working as a professional bioinformatician, I routinely saw how very subtle issues could crop up and adversely change the outcome of the analysis had they not been caught. I don't want your bioinformatics work to be incorrect because I've made a topic artificially simple. The depth at which I cover topics in *Bioinformatics Data Skills* is meant to prepare you to catch similar issues in your own work so your results are robust.

The result is that sections of this book are quite advanced and will be difficult for some readers. Don feel discouraged! Like most of science, this material *is hard*, and may take a few reads before it fully sinks in. Throughout the book, I try to indicate when certain sections are especially advanced so that you can skip over these and return to them later.

Lastly, I often use technical jargon throughout the book. I don't like using jargon, but it's necessary t communicate technical concepts in computing. Primarily it will help you search for additional resources and help. It's much easier to Google successfully for "left outer join" than "data merge where null records are included in one table."

Assumptions This Book Makes

Bioinformatics Data Skills is meant to be an intermediate book on bioinformatics. To make sure everyone starts out on the same foot, the book begins with a few simple chapters. In Chapter 2, I cov the basics of setting up a bioinformatics project, and in Chapter 3 I teach some remedial Unix topics meant to ensure that you have a solid grasp of Unix (because Unix is a large component in later chapters). Still, as an intermediate book, I make a few assumptions about you:

You know a scripting language

This is the biggest assumption of the book. Except for a few Python programs and the R material (R is introduced in Chapter 8), this book doesn't directly rely on using lots of scripting. However in learning a scripting language, you've already encountered many important computing concepts such as working with a text editor, running and executing programs on the command line, and basic programming. If you do not know a scripting language, I would recommend learning Pytho while reading this book. Books like *Bioinformatics Programming Using Python* by Mitchell L. Model (O'Reilly, 2009), *Learning Python, 5th Edition*, by Mark Lutz (O'Reilly, 2013), and *Pytho in a Nutshell, 2nd*, by Alex Martelli (O'Reilly, 2006) are great to get started. If you know a scripting language other than Python (e.g., Perl or Ruby), you'll be prepared to follow along with most examples (though you will need to translate some examples to your scripting language of choice).

You know how to use a text editor

It's essential that you know your way around a text editor (e.g., Emacs, Vim, TextMate2, or Sublime Text). Using a word processor (e.g., Microsoft Word) will not work, and I would discourage using text editors such as Notepad or OS X's TextEdit, as they lack syntax highlightin support for common programming languages.

You have basic Unix command-line skills

For example, I assume you know the difference between a terminal and a shell, understand how t enter commands, what command-line options/flags and arguments are, and how to use the up arrow to retrieve your last entered command. You should also have a basic understanding of the Unix file hierarchy (including concepts like your home directory, relative versus absolute directories, and root directories). You should also be able to move about and manipulate the directories and files in Unix with commands like `cd`, `ls`, `pwd`, `mv`, `rm`, `rmdir`, and `mkdir`. Finally, you should have a basic grasp of Unix file ownership and permissions, and changing these with `chown` and `chmod`. If these concepts are unclear, I would recommend you play around in the Unix command line first (carefully!) and consult a good beginner-level book such as *Practical Computing for Biologists* by Steven Haddock and Casey Dunn (Sinauer, 2010) or *UNIX and Perl t the Rescue* by Keith Bradnam and Ian Korf (Cambridge University Press, 2012).

You have a basic understanding of biology

Bioinformatics Data Skills is a BYOB book—*bring your own biology*. The examples don't requir a lot of background in biology beyond what DNA, RNA, proteins, and genes are, and the central dogma of molecular biology. You should also be familiar with some very basic genetics and

genomic concepts (e.g., single nucleotide polymorphisms, genotypes, GC content, etc.). All biological examples in the book are designed to be quite simple; if you're unfamiliar with any topic, you should be able to quickly skim a Wikipedia article and proceed with the example.

You have a basic understanding of regular expressions

Occasionally, I'll make use of regular expressions in this book. In most cases, I try to quickly ste through the basics of how a regular expression works so that you can get the general idea. If you've encountered regular expressions while learning a scripting language, you're ready to go. I not, I recommend you learn the basics—not because regular expressions are used heavily throughout the book, but because mastering regular expressions is an important skill in bioinformatics. *Introducing Regular Expressions* by Michael Fitzgerald (O'Reilly) is a great introduction. Nowadays, writing, testing, and debugging regular expressions is easier than ever thanks to online tools like *http://regex101.com* and *http://www.debuggex.com*. I recommend using these tools in your own work and when stepping through my regular expression examples.

You know how to get help and read documentation

Throughout this book, I try to minimize teaching information that can be found in manual pages, help documentation, or online. This is for three reasons:

- I want to save space and focus on presenting material in a way you can't find elsewhere
- Manual pages and documentation will always be the best resource for this information
- The ability to quickly find answers in documentation is one of the most important skills you can develop when learning computing

This last point is especially important; you don't need to remember all arguments of a command or f function—you just need to know where to find this information. Programmers consult documentation *constantly* in their work, which is why documentation tools like `man` (in Unix) and `help()` (in R) exis

You can manage your computer system (or have a system administrator)

This book does not teach you system administration skills like setting up a bioinformatics server or cluster, managing user accounts, network security, managing disks and disk space, RAID configurations, data backup, and high-performance computing concepts. There simply isn't the space to adequately cover these important topics. However, these are all very, very important—if you don't have a system administrator and need to fill that role for your lab or research group, it' essential for you to master these skills, too. Frankly, system administration skills take years to master and good sysadmins have incredible patience and experience in handling issues that would make most scientists go insane. If you can employ a full-time system administrator shared across labs or groups or utilize a university cluster with a sysadmin, I would do this. Lastly, this shouldn't need to be said, but just in case: constantly back up your data and work. It's easy when learning Unix to execute a command that destroys files—your best protection from losing everything is continual backups.

Supplementary Material on GitHub

The supplementary material needed for this book's examples is available in the GitHub repository. You can download material from this repository as you need it (the repository is organized by chapter), or you can download everything using the Download Zip link. Once you learn Git in Chapter 5, I would recommend cloning the repository so that you can restore any example files shoul you accidentally overwrite them.

Try navigating to this repository now and poking around so you're familiar with the layout. Look in the Preface's directory and you'll find the *README.md* file, which includes additional information about many of the topics I've discussed. In addition to the supplementary files needed for all examples in the book, this repository contains:

- Documentation on how all supplementary files were produced or how they were acquired. In some cases, I've used makefiles or scripts (both of these topics are covered in Chapter 12) to create example data, and all of these resources are available in each chapter's GitHub directory. I've included these materials not only for reproducible purposes, but also to serve as additional learnin material.
- Additional information readers may find interesting for each chapter. This information is in each chapter's *README.md* file. I've also included other resources like lists of recommended books fo further learning.
- Errata, and any necessary updates if material becomes outdated for some reason.

I chose to host the supplementary files for *Bioinformatics Data Skills* on GitHub so that I could keep everything up to date and address any issues readers may have. Feel free to create a new issue on GitHub should you find any problem with the book or its supplementary material.

Computing Resources and Setup

I've written this entire book on my laptop, a 15-inch MacBook Pro with 16 GB of RAM. Although th is a powerful laptop, it is much smaller than the servers common in bioinformatics computing. All examples are designed and tested to run a machine this size. Nearly every example should run on a machine with 8 GB of memory.

All examples in this book work on Mac OS X and Linux—other operating systems are not supported (mostly because modern bioinformatics relies on Unix-based operating systems). All software required throughout the book is freely available and is easily installable; I provide some basic instructions in each section as software installation is needed. In general, you should use your operating system's package management system (e.g., `apt-get` on Ubuntu/Debian). If you're using a Mac, I highly recommend Homebrew, a terrific package manager for OS X that allows you to easily install software from the command line. You can find detailed instructions on Homebrew's website, Most important, Homebrew maintains a collection of scientific software packages called homebrew-science, including the bioinformatics software we use throughout this book. Follow the directions in homebrew-science's *README.md* to learn how to install these scientific programs.

Organization of This Book

This book is composed of three parts: Part I, containing one chapter on ideology; Part II, which cover the basics of getting started with a bioinformatics project; and Part III, which covers bioinformatics data skills. Although chapters were written to be read sequentially, if you're comfortable with Unix and R, you may find that you can skip around without problems.

In Chapter 1, I introduce why learning bioinformatics by developing data skills is the best approach. also introduce the ideology of this book, and describe reproducible and robust bioinformatics and some recommendations to apply in your own work.

Part II of *Bioinformatics Data Skills* introduces the basic skills needed to start a bioinformatics project. First, we'll look at how to set up and manage a project directory in Chapter 2. This may seem like trivial topic, but complex bioinformatics projects demand we think about project management. I the frenzy of research, there will be files *everywhere*. Starting out with a carefully organized project can prevent a lot of hassle in the future. We'll also learn about documentation with Markdown, a useful format for plain-text project documentation.

In Chapter 3, we explore intermediate Unix in bioinformatics. This is to make sure that you have a solid grasp of essential concepts (e.g., pipes, redirection, standard input and output, etc.). Understanding these prerequisite topics will allow you to focus on analyzing data in later chapters, n struggling to understand Unix basics.

Most bioinformatics tasks require more computing power than we have on our personal workstations meaning we have to work with remote servers and clusters. Chapter 4 covers some tips and tricks to increase your productivity when working with remote machines.

In Chapter 5, we learn Git, which is a version control system that makes managing versions of projec easy. Bioinformatics projects are filled with lots of code and data that should be managed using the same modern tools as collaboratively developed software. Git is a large, powerful piece of software, so this is a long chapter. However, this chapter was written so that you could skip the section on branching and return to it later.

Chapter 6 looks at data in bioinformatics projects: how to download large amounts of data, use data compression, validate data integrity, and reproducibly download data for a project.

In Part III, our attention turns to developing the essential data skills all bioinformaticians need to tackle problems in their daily work. Chapter 7 focuses on Unix data tools, which allow you to quickly write powerful stream-processing Unix pipelines to process bioinformatics data. This approach is a cornerstone of modern bioinformatics, and is an absolutely essential data skill to have.

In Chapter 8, I introduce the R language through learning exploratory data analysis techniques. This chapter prepares you to use R to explore your own data using techniques like visualization and data summaries.

Genomic range data is ubiquitous in bioinformatics, so we look at range data and range operations in Chapter 9. We'll first step through the different ways to represent genomic ranges, and work through range operations using Bioconductor's `IRanges` package to bolster our range-thinking intuition. Then we'll work with genomic data using `GenomicRanges`. Finally, we'll look at the BEDTools Suite of tools for working with range data on the command line.

In Chapter 10, we learn about sequence data, a mainstay of bioinformatics data. We'll look at the

FASTA and FASTQ formats (and their limitations) and work through trimming low-quality bases off of sequences and seeing how this affects the distribution of quality scores. We'll also look at FASTA and FASTQ parsing.

Chapter 11 focuses on the alignment data formats SAM and BAM. Understanding and manipulating files in these formats is an integral bioinformatics skill in working with high-throughput sequencing data. We'll see how to use Samtools to manipulate these files and visualize the data, and step through a detailed example that illustrates some of the intricacies of variant calling. Finally, we'll learn how use Pysam to parse SAM/BAM files so you can write your own scripts that work with these specialized data formats.

Most daily bioinformatics work involves writing data-processing scripts and pipelines. In Chapter 12 we look at how to write such data-processing pipelines in a robust and reproducible way. We'll look specifically at Bash scripting, manipulating files using Unix powertools like `find` and `xargs`, and finally take a quick look at how you can write pipelines using Make and makefiles.

In bioinformatics, our data is often too large to fit in our computer's memory. In Chapter 7, we saw how streaming with Unix pipes can help to solve this problem, but Chapter 13 looks at a different method: out-of-memory approaches. First, we'll look at Tabix, a fast way to access information in indexed tab-delimited files. Then, we'll look at the basics of SQL through analyzing some GWAS da using SQLite.

Finally, in Chapter 14, I discuss where you should head next to further develop your bioinformatics skills.

Code Conventions

Most bioinformatics data has one thing in common: it's large. In code examples, I often need to truncate the output to have it fit into the width of a page. To indicate that output has been truncated, I will always use `[...]` in the output. Also, in code examples I often use variable names that are short to save space. I encourage you to use more descriptive names than those I've used throughout this book in your own personal work.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`

Shows commands or other text that should be typed literally by the user.

Constant width italic

Shows text that should be replaced with user-supplied values or by values determined by context.

> **TIP**
>
> This element signifies a tip or suggestion.

> **NOTE**
>
> This element signifies a general note.

> **WARNING**
>
> This element indicates a warning or caution.

Using Code Examples

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: "*Bioinformatics Data Skills* by Vince Buffalo (O'Reilly). Copyright 2015 Vince Buffalo, 978-1-449-36737-4."

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*.

Safari® Books Online

> **NOTE**

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving,

learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

- O'Reilly Media, Inc.
- 1005 Gravenstein Highway North
- Sebastopol, CA 95472
- 800-998-9938 (in the United States or Canada)
- 707-829-0515 (international or local)
- 707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/Bio-DS*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

Writing a book is a monumental effort—for two years, I've worked on *Bioinformatics Data Skills* during nights and weekends. This is in addition to a demanding career as a professional bioinformatician (and for the last five months of writing, as a PhD student). Balancing work and life already difficult enough for most scientists; I now know that balancing work, life, and writing a book is nearly impossible. I wouldn't have survived this process without the support of my partner, Helene Hopfer.

I thank Ciera Martinez for continually providing useful feedback and helping me calibrate the tone and target audience of this book. Cody Markelz tirelessly provided feedback and was never afraid to say when I'd missed the mark on a chapter—for this, all readers should be thankful. My friend Al

Marks deserves special recognition not only for proving valuable feedback on many chapters, but als for introducing me to computing and programming back in high school. I also thank Jeff Ross-Ibarra for inspiring my passion for population genetics and presenting me with challenging and interesting projects in his lab. I owe a debt of gratitude to the entire UC Davis Bioinformatics Core for the fantastic time I spent working there; thanks especially to Dawei Lin, Joe Fass, Nikhil Joshi, and Monica Britton for sharing their knowledge and granting me freedom to explore bioinformatics. Mik Lewis also deserves a special thanks for teaching me about computing and being a terrific person to nerd out on techie details with. Peter Morrell, his lab, and the "Does[0]compute?" reading group provided lots of useful feedback that I'm quite grateful for. I thank Jorge Dubcovsky—witnessing his tireless pursuit of science has motivated me to do the same. Lastly, I'm indebted to my wonderful advisor, Graham Coop, for his patience in allowing me to finish this book—with this book out of the way, I'm eager to pursue my future directions under his mentorship.

This book was significantly improved by the valuable input of many reviewers, colleagues, and friends. Thank you Peter Cock, Titus Brown, Keith Bradnam, Mike Covington, Richard Smith-Unna, Stephen Turner, Karthik Ram, Gabe Becker, Noam Ross, Chris Hamm, Stephen Pearce, Anke Schennink, Patrik D'haeseleer, Bill Broadley, Kate Crosby, Arun Durvasula, Aaron Quinlan, and David Ruddock. Shaun Jackman deserves recognition for his tireless effort in making bioinformatics software easy to install through the Homebrew and `apt-get` projects—my readers will greatly benefi from this. I also am grateful for the comments and positive feedback I received from many of the early release readers of this book; the positive reception provided a great motivating push to finish everything. However, as author, I do take full credit for any errors or omissions that have slipped by these devoted reviewers.

Most authors are lucky if they work with one great editor—I got to work with two. Thank you, Courtney Nash and Amy Jollymore, for your continued effort and encouragement throughout this process. Simply put, I wouldn't have been able to do this without you both. I'd also like to thank my production editor Nicole Shelby, copyeditor Jasmine Kwityn, and the rest of the O'Reilly production team for their extremely hard work in editing *Bioinformatics Data Skills*. Finally, thank you, Mike Loukides, for your feedback and for taking an interest in my book when it was just a collection of early, rough ideas—you saw more.

Part I. Ideology: Data Skills for Robust and Reproducible Bioinformatics

Chapter 1. How to Learn Bioinformatics

Right now, in labs across the world, machines are sequencing the genomes of the life on earth. Even with rapidly decreasing costs and huge technological advancements in genome sequencing, we're onl seeing a glimpse of the biological information contained in every cell, tissue, organism, and ecosystem. However, the smidgen of total biological information we're gathering amounts to mountains of data biologists need to work with. At no other point in human history has our ability to understand life's complexities been so dependent on our skills to work with and analyze data.

This book is about learning bioinformatics through developing data skills. In this chapter, we'll see what data skills are, and why learning data skills is the best way to learn bioinformatics. We'll also look at what robust and reproducible research entails.

Why Bioinformatics? Biology's Growing Data

Bioinformaticians are concerned with deriving biological understanding from large amounts of data with specialized skills and tools. Early in biology's history, the datasets were small and manageable. Most biologists could analyze their own data after taking a statistics course, using Microsoft Excel o a personal desktop computer. However, this is all rapidly changing. Large sequencing datasets are widespread, and will only become more common in the future. Analyzing this data takes different tools, new skills, and many computers with large amounts of memory, processing power, and disk space.

In a relatively short period of time, sequencing costs dropped drastically, allowing researchers to utilize sequencing data to help answer important biological questions. Early sequencing was low-throughput and costly. Whole genome sequencing efforts were expensive (the human genome cost around $2.7 billion) and only possible through large collaborative efforts. Since the release of the human genome, sequencing costs have decreased exponentially until about 2008, as shown in Figure 1-1. With the introduction of next-generation sequencing technologies, the cost of sequencing megabase of DNA dropped even more rapidly. At this crucial point, a technology that was only affordable to large collaborative sequencing efforts (or individual researchers with very deep pockets became affordable to researchers across all of biology. You're likely reading this book to learn to work with sequencing data that would have been much too expensive to generate less than 10 years ago.

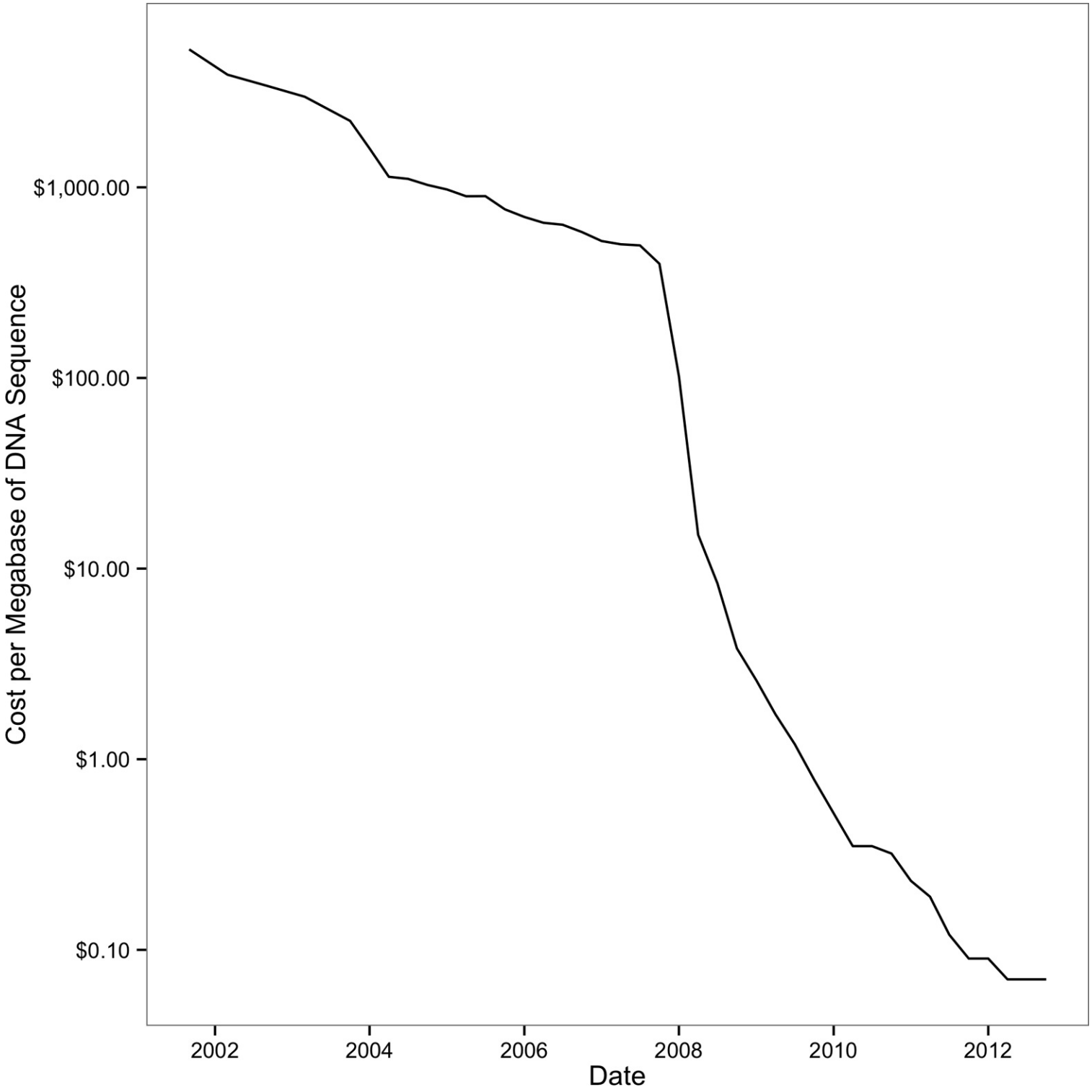


Figure 1-1. Drop of sequencing costs (note the y-axis is on a logarithmic scale); the sharp drop around 2008 was due to the introduction of next-generation sequencing data. (figure reproduced and data downloaded from the NIH)

What was the consequence of this drop in sequencing costs due to these new technologies? As you may have guessed, lots and lots of data. Biological databases have swelled with data after exponentia growth. Whereas once small databases shared between collaborators were sufficient, now petabytes o useful data are sitting on servers all over the world. Key insights into biological questions are stored not just in the unanalyzed experimental data sitting on your hard drive, but also spinning around a dis in a data center thousands of miles away.

The growth of biological databases is as astounding as the drop of sequencing costs. As an example, consider the Sequence Read Archive (previously known as the *Short* Read Archive), a repository of

the raw sequencing data from sequencing experiments. Since 2010, it has experienced remarkable growth; see Figure 1-2.

To put this incredible growth of sequencing data into context, consider Moore's Law. Gordon Moore (a cofounder of Intel) observed that the number of transistors in computer chips doubles roughly eve two years. More transistors per chip translates to faster speeds in computer processors and more random access memory in computers, which leads to more powerful computers. This extraordinary rate of technological improvement—output doubling every two years—is likely the fastest growth in technology humanity has ever seen. Yet, since 2011, the amount of sequencing data stored in the Sho Read Archive has outpaced even this incredible growth, having doubled every year.

To make matters even more complicated, new tools for analyzing biological data are continually bei created, and their underlying algorithms are advancing. A 2012 review listed over 70 short-read mappers (Fonseca et al., 2012; see http://bit.ly/hts-mappers). Likewise, our approach to genome assembly has changed considerably in the past five years, as methods to assemble long sequences (such as overlap-layout-consensus algorithms) were abandoned with the emergence of short high-throughput sequencing reads. Now, advances in sequencing chemistry are leading to longer sequencing read lengths and new algorithms are replacing others that were just a few years old.

Unfortunately, this abundance and rapid development of bioinformatics tools has serious downsides. Often, bioinformatics tools are not adequately benchmarked, or if they are, they are only benchmarke in one organism. This makes it difficult for new biologists to find and choose the best tool to analyze their data. To make matters more difficult, some bioinformatics programs are not actively developed so that they lose relevance or carry bugs that could negatively affect results. All of this makes choosing an appropriate bioinformatics program in your own research difficult. More importantly, it imperative to critically assess the output of bioinformatics programs run on your own data. We'll see examples of how data skills can help us assess program output throughout Part II.

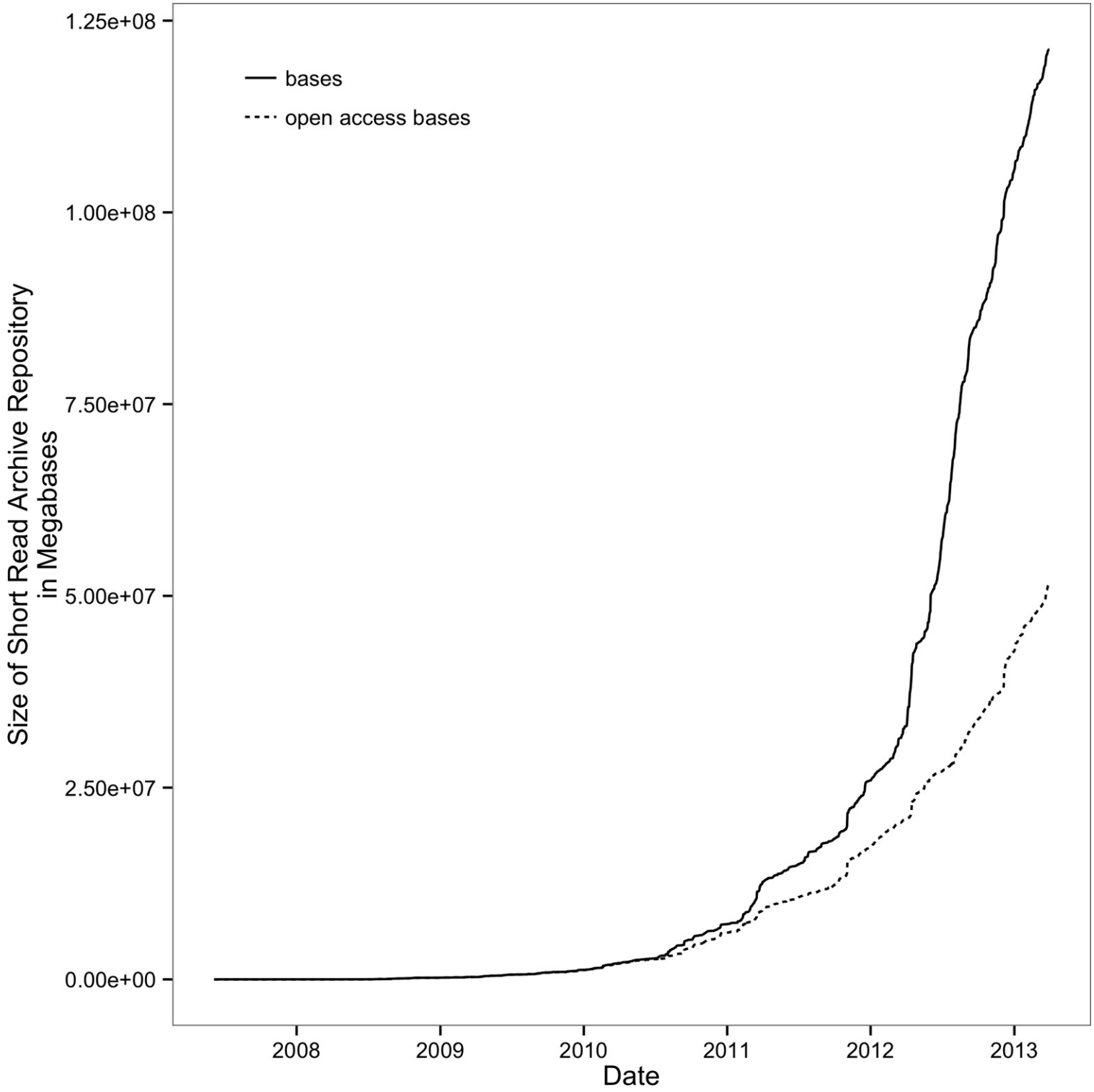


Figure 1-2. Exponential growth of the Short Read Archive; open access bases are SRA submissions available to the public (figure reproduced and data downloaded from the NIH)

Learning Data Skills to Learn Bioinformatics

With the nature of biological data changing so rapidly, how are you supposed to learn bioinformatics? With all of the tools out there and more continually being created, how is a biologist supposed to know whether a program will work appropriately on her organism's data?

The solution is to approach bioinformatics as a bioinformatician does: try stuff, and assess the results. In this way, bioinformatics is just about having the skills to experiment with data using a computer

and understanding your results. The experimental part is easy; this comes naturally to most scientists The limiting factor for most biologists is having the data skills to freely experiment and work with large data on a computer. The goal of this book is to teach you the bioinformatics data skills necessar to allow you to experiment with data on a computer as easily as you would run experiments in the lab

Unfortunately, many of the biologist's common computational tools can't scale to the size and complexity of modern biological data. Complex data formats, interfacing numerous programs, and assessing software and data make large bioinformatics datasets difficult to work with. Learning core bioinformatics data skills will give you the foundation to learn, apply, and assess any bioinformatics program or analysis method. In 10 years, bioinformaticians may only be using a few of the bioinformatics software programs around today. But we most certainly will be using data skills and experimentation to assess data and methods of the future.

So what are data skills? They are the set of computational skills that give you the ability to quickly improvise a way of looking at complex datasets, using a well-known set of tools. A good analogy is what jazz musicians refer to as having "chops." A jazz musician with good chops can walk into a nightclub, hear a familiar standard song being played, recognize the chord changes, and begin playing musical ideas over these chords. Likewise, a bioinformatician with good data skills can receive a hug sequencing dataset and immediately start using a set of tools to see what story the data tells.

Like a jazz musician that's mastered his instrument, a bioinformatician with excellent data chops masters a set of tools. Learning one's tools is a necessary, but not sufficient step in developing data skills (similarly, learning an instrument is a necessary, but not sufficient step to playing musical ideas). Throughout the book, we will develop our data skills, from setting up a bioinformatics project and data in Part II, to learning both small and big tools for data analysis in Part III. However, this book can only set you on the right path; real mastery requires learning through repeatedly applying skills to real problems.

New Challenges for Reproducible and Robust Research

Biology's increasing use of large sequencing datasets is changing more than the tools and skills we need: it's also changing how reproducible and robust our scientific findings are. As we utilize new tools and skills to analyze genomics data, it's necessary to ensure that our approaches are still as reproducible and robust as any other experimental approaches. Unfortunately, the size of our data and the complexity of our analysis workflows make these goal especially difficult in genomics.

The requisite of reproducibility is that we share our data and methods. In the pre-genomics era, this was much easier. Papers could include detailed method summaries and entire datasets—exactly as Kreitman's 1986 paper did with a 4,713bp Adh gene flanking sequence (it was embedded in the middle of the paper). Now papers have long supplementary methods, code, and data. Sharing data is no longer trivial either, as sequencing projects can include terabytes of accompanying data. Reference genomes and annotation datasets used in analyses are constantly updated, which can make reproducibility tricky. Links to supplemental materials, methods, and data on journal websites break, materials on faculty websites disappear when faculty members move or update their sites, and software projects become stale when developers leave and don't update code. Throughout this book, we'll look at what can be done to improve reproducibility of your project alongside doing the actual

analysis, as I believe these are necessarily complementary activities.

Additionally, the complexity of bioinformatics analyses can lead to findings being susceptible to errors and technical confounding. Even fairly routine genomics projects can use dozens of different programs, complicated input parameter combinations, and many sample and annotation datasets; in addition, work may be spread across servers and workstations. All of these computational data-processing steps create results used in higher-level analyses where we draw our biological conclusions. The end result is that research findings may rest on a rickety scaffold of numerous processing steps. To make matters worse, bioinformatics workflows and analyses are usually only ru once to produce results for a publication, and then never run or tested again. These analyses may rely on very specific versions of all software used, which can make it difficult to reproduce on a different system. In learning bioinformatics data skills, it's necessary to concurrently learn reproducibility and robust best practices. Let's take a look at both reproducibility and robustness in turn.

Reproducible Research

Reproducing scientific findings is the only way to confirm they're accurate and not the artifact of a single experiment or analysis. Karl Popper, in *The Logic of Scientific Discovery*, famously said: "non reproducible single occurrences are of no significance to science" (1959). Independent replication of experiments and analysis is the gold standard by which we assess the validity of scientific findings. Unfortunately, most sequencing experiments are too expensive to reproduce from the test tube up, so we increasingly rely on *in silico* reproducibility only. The complexity of bioinformatics projects usually discourages replication, so it's our job as good scientists to facilitate and encourage *in silico* reproducibility by making it easier. As we'll see later, adopting good reproducibility practices can also make your life easier as a researcher.

So what is a reproducible bioinformatics project? At the very least, it's sharing your project's code and data. Most journals and funding agencies require you to share your project's data, and resources like NCBI's Sequence Read Archive exist for this purpose. Now, editors and reviewers will often suggest (or in some cases require) that a project's code also be shared, especially if the code is a significant part of a study's results. However, there's a lot more we can and should do to ensure our projects' reproducibility. By having to reproduce bioinformatics analyses to verify results, I've learned from these sleuthing exercises that the devil is in the details.

For example, colleagues and I once had a difficult time reproducing an RNA-seq differential expression analysis we had done ourselves. We had preliminary results from an analysis on a subset samples done a few weeks earlier, but to our surprised, our current analysis was producing a drastically smaller set of differentially expressed genes. After rechecking how our past results were created, comparing data versions and file creation times, and looking at differences in the analysis code, we were still stumped—nothing could explain the difference between the results. Finally, we checked the version of our R package and realized that it had been updated on our server. We then reinstalled the old version to confirm this was the source of the difference, and indeed it was. The lesson here is that often replication, by either you in the future or someone else, relies on not just dat and code but details like software versions and when data was downloaded and what version it is. Thi *metadata*, or data about data, is a crucial detail in ensuring reproducibility.

Another motivating case study in bioinformatics reproducibility is the so-called "Duke Saga." Dr. Anil Potti and other researchers at Duke University created a method that used expression data from high-throughput microarrays to detect and predict response to different chemotherapy drugs. These methods were the beginning of a new level of personalized medicine, and were being used to determine the chemotherapy treatments for patients in clinical trials. However, two biostatisticians, Keith Baggerly and Kevin Coombes, found serious flaws in the analysis of this study when trying to reproduce it (Baggerly and Coombes, 2009). Many of these required what Baggerly and Coombes called "forensic bioinformatics"—sleuthing to try to reproduce a study's findings when there isn't sufficient documentation to retrace each step. In total, Baggerly and Coombes found multiple serious errors, including:

- An off-by-one error, as an entire list of gene expression values was shifted down in relation to the correct identifier
- Two outlier genes of biological interest were not on the microarrays used
- There was confounding of treatment with the day the microarray was run
- Sample group names were mixed up

Baggerly and Coombes's work is best summarized by their open access article, "Deriving Chemosensitivity from Cell Lines: Forensic Bioinformatics and Reproducible Research in High-Throughput Biology" (see this chapter's GitHub directory for this article and more information abou the Duke Saga). The lesson of Baggerly and Coombes's work is that "common errors are simple, and simple errors are common" and poor documentation can lead to both errors and irreproducibility. Documentation of methods, data versions, and code would have not only facilitated reproducibility, but it likely would have prevented a few of these serious errors in their study. Striving for maximal reproducibility in your project often will make your work more robust, too.

Robust Research and the Golden Rule of Bioinformatics

> *Since the computer is a sharp enough tool to be really useful, you can cut yourself on it.*
>
> *The Technical Skills of Statistics (1964) John Tukey*

In wetlab biology, when experiments fail, it can be very apparent, but this is not always true in computing. Electrophoresis gels that look like Rorschach blots rather than tidy bands clearly indicate something went wrong. In scientific computing, errors can be *silent*; that is, code and programs may produce output (rather than stop with an error), but this output may be incorrect. This is a very important notion to put in the back of your head as you learn bioinformatics.

Additionally, it's common in scientific computing for code to be run only once, as researchers get their desired output and move on to the next step. In contrast, consider a video game: it's run on thousands (if not millions) of different machines, and is, in effect, constantly being tested by many users. If a bug that deletes a user's score occurs, it's exceptionally likely to be quickly noticed and reported by users. Unfortunately, the same is not true for most bioinformatics projects.

Genomics data also creates its own challenges for robust research. First, most bioinformatics analyse produce intermediate output that is too large and high dimensional to inspect or easily visualize. Mos