Case Studies and Algorithms to Get You Started

Machine Learning

for Hackers

Drew Conway &
John Myles White

O'REILLY®

Machine Learning for Hackers

If you're an experienced programmer interested in crunching data, this book will get you started with machine learning—a toolkit of algorithms that enables computers to train themselves to automate useful tasks. Authors Drew Conway and John Myles White help you understand machine learning and statistics tools through a series of hands-on case studies, instead of a traditional math-heavy presentation.

Each chapter focuses on a specific problem in machine learning, such as classification, prediction, optimization, and recommendation. Using the R programming language, you'll learn how to analyze sample datasets and write simple machine learning algorithms. *Machine Learning for Hackers* is ideal for programmers from any background, including business, government, and academic research.

- Develop a naïve Bayesian classifier to determine if an email is spam, based only on its text
- Use linear regression to predict the number of page views for the top 1,000 websites
- Learn optimization techniques by attempting to break a simple letter cipher
- Compare and contrast US Senators statistically, based on their voting records
- Build a "whom to follow" recommendation system from Twitter data

"This book provides excellent case studies of a dozen different techniques in machine learning. With a focus on process, rather than cookbooks or theory, it should be accessible to anybody with a programming background and a quantitative mind."

—**Max Shron**
OkCupid

Drew Conway, a PhD candidate in Politics at NYU, studies international relations, conflict, and terrorism, using mathematics, statistics, and computer science tools. He spent years as an analyst in the US intelligence and defense communities.

John Myles White is a PhD student in the Princeton Psychology Department, where he studies how humans make decisions both theoretically and experimentally. He's also the lead maintainer for several popular R packages, including ProjectTemplate and log4r.

US $39.99 CAN $41.99
ISBN: 978-1-449-30371-6

Twitter: @oreillymedia
facebook.com/oreilly

O'REILLY®
oreilly.com

Machine Learning for Hackers

Drew Conway and John Myles White

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Machine Learning for Hackers
by Drew Conway and John Myles White

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://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Julie Steele
Production Editor: Melanie Yarbrough
Copyeditor: Genevieve d'Entremont
Proofreader: Teresa Horton

Indexer: Angela Howard
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

February 2012: First Edition.

Revision History for the First Edition:
2012-02-06 First release
See *http://oreilly.com/catalog/errata.csp?isbn=9781449303716* for release details.

ISBN: 978-1-449-30371-6

[LSI]

1328629742

Table of Contents

Preface

Machine Learning for Hackers

To explain the perspective from which this book was written, it will be helpful to define the terms *machine learning* and *hackers*.

What is machine learning? At the highest level of abstraction, we can think of machine learning as a set of tools and methods that attempt to infer patterns and extract insight from a record of the observable world. For example, if we are trying to teach a computer to recognize the zip codes written on the fronts of envelopes, our data may consist of photographs of the envelopes along with a record of the zip code that each envelope was addressed to. That is, within some context we can take a record of the actions of our subjects, learn from this record, and then create a model of these activities that will inform our understanding of this context going forward. In practice, this requires data, and in contemporary applications this often means a lot of data (perhaps several terabytes). Most machine learning techniques take the availability of such data as given, which means new opportunities for their application in light of the quantities of data that are produced as a product of running modern companies.

What is a hacker? Far from the stylized depictions of nefarious teenagers or Gibsonian cyber-punks portrayed in pop culture, we believe a hacker is someone who likes to solve problems and experiment with new technologies. If you've ever sat down with the latest O'Reilly book on a new computer language and knuckled out code until you were well past "Hello, World," then you're a hacker. Or if you've dismantled a new gadget until you understood the entire machinery's architecture, then we probably mean you, too. These pursuits are often undertaken for no other reason than to have gone through the process and gained some knowledge about the *how* and the *why* of an unknown technology.

Along with an innate curiosity for how things work and a desire to build, a computer hacker (as opposed to a car hacker, life hacker, food hacker, etc.) has experience with software design and development. This is someone who has written programs before, likely in many different languages. To a hacker, Unix is not a four-letter word, and command-line navigation and bash operations may come as naturally as working with GUIs. Using regular expressions and tools such as `sed`, `awk`, and `grep` are a hacker's first

line of defense when dealing with text. In the chapters contained in this book, we will assume a relatively high level of this sort of knowledge.

How This Book Is Organized

Machine learning blends concepts and techniques from many different traditional fields, such as mathematics, statistics, and computer science. As such, there are many ways to learn the discipline. Considering its theoretical foundations in mathematics and statistics, newcomers would do well to attain some degree of mastery of the formal specifications of basic machine learning techniques. There are many excellent books that focus on the fundamentals, the classic work being Hastie, Tibshirani, and Friedman's *The Elements of Statistical Learning* ([HTF09]; full references can be found in the Works Cited).[1] But another important part of the hacker mantra is to learn by doing. Many hackers may be more comfortable thinking of problems in terms of the *process* by which a solution is attained, rather than the *theoretical foundation* from which the solution is derived.

From this perspective, an alternative approach to teaching machine learning would be to use "cookbook"-style examples. To understand how a recommendation system works, for example, we might provide sample training data and a version of the model, and show how the latter uses the former. There are many useful texts of this kind as well, and Segaran's *Programming Collective Intelligence* is one recent example [Seg07]. Such a discussion would certainly address the *how* of a hacker's method of learning, but perhaps less of the *why*. Along with understanding the mechanics of a method, we may also want to learn why it is used in a certain context or to address a specific problem.

To provide a more complete reference on machine learning for hackers, therefore, we need to compromise between providing a deep review of the theoretical foundations of the discipline and a broad exploration of its applications. To accomplish this, we have decided to teach machine learning through selected case studies.

We believe the best way to learn is by first having a problem in mind, then focusing on learning the tools used to solve that problem. This is effectively the mechanism through which case studies work. The difference being, rather than having some problem for which there may be no known solution, we can focus on well-understood and studied problems in machine learning and present specific examples of cases where some solutions excelled while others failed spectacularly.

For that reason, each chapter of this book is a self-contained case study focusing on a specific problem in machine learning. The organization of the early cases moves from classification to regression (discussed further in Chapter 1). We then examine topics

1. *The Elements of Statistical Learning* can now be downloaded free of charge at *http://www-stat.stanford.edu/~tibs/ElemStatLearn/*.

such as clustering, dimensionality reduction, and optimization. It is important to note that not all problems fit neatly into either the classification or regression categories, and some of the case studies reviewed in this book will include aspects of both (sometimes explicitly, but also in more subtle ways that we will review). Following are brief descriptions of all the case studies reviewed in this book in the order they appear:

Text classification: spam detection
: In this chapter we introduce the idea of binary classification, which is motivated through the use of email text data. Here we tackle the classic problem in machine learning of classifying some input as one of two types, which in this case is either ham (legitimate email) or spam (unwanted email).

Ranking items: priority inbox
: Using the same email text data as in the previous case study, here we move beyond a binary classification to a discrete set of types. Specifically, we need to identify the appropriate features to extract from the email that can best inform its "priority" rank among all emails.

Regression models: predicting page views
: We now introduce the second primary tool in machine learning, linear regression. Here we explore data whose relationship roughly approximates a straight line. In this case study, we are interested in predicting the number of page views for the top 1,000 websites on the Internet as of 2011.

Regularization: text regression
: Sometimes the relationships in our data are not well described by a straight line. To describe the relationship, we may need to fit a different function; however, we also must be cautious not to overfit. Here we introduce the concept of regularization to overcome this problem, and motivate it through a case study, focusing on understanding the relationship among words in the text from O'Reilly book descriptions.

Optimization: code breaking
: Moving beyond regression models, almost every algorithm in machine learning can be viewed as an optimization problem in which we try to minimize some measure of prediction error. Here we introduce classic algorithms for performing this optimization and attempt to break a simple letter cipher with these techniques.

Unsupervised learned: building a stock market index
: Up to this point we have discussed only supervised learning techniques. Here we introduce its methodological counterpart: unsupervised learning. The important difference is that in supervised learning, we wish to use the structure of our data to make predictions, whereas in unsupervised learning, we wish to discover structure in our data for structure's sake. In this case we will use stock market data to create an index that describes how well the overall market is doing.

Spatial similarity: clustering US Senators by the voting records
: Here we introduce the concept of spatial distances among observations. To do so, we define measures of distance and describe methods for clustering observations basing on their spatial distances. We use data from US Senator roll call voting to cluster those legislators based on their votes.

Recommendation system: suggesting R packages to users
: To further the discussion of spatial similarities, we discuss how to build a recommendation system based on the closeness of observations in space. Here we introduce the *k*-nearest neighbors algorithm and use it to suggest R packages to programmers based on their currently installed packages.

Social network analysis: who to follow on Twitter
: Here we attempt to combine many of the concepts previously discussed, as well as introduce a few new ones, to design and build a "who to follow" recommendation system from Twitter data. In this case we build a system for downloading Twitter network data, discover communities within the structure, and recommend new users to follow using basic social network analysis techniques.

Model comparison: finding the best algorithm for your problem
: In the final chapter, we discuss techniques for choosing which machine learning algorithm to use to solve your problem. We introduce our final algorithm, the support vector machine, and compare its performance on the spam data from Chapter 3 with the performance of the other algorithms we introduce earlier in the book.

The primary tool we use to explore these case studies is the R statistical programming language (*http://www.r-project.org/*). R is particularly well suited for machine learning case studies because it is a high-level, functional scripting language designed for data analysis. Much of the underlying algorithmic scaffolding required is already built into the language or has been implemented as one of the thousands of R packages available on the Comprehensive R Archive Network (CRAN).[2] This will allow us to focus on the *how* and the *why* of these problems, rather than review and rewrite the foundational code for each case.

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.

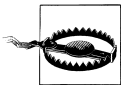
2. For more information on CRAN, see *http://cran.r-project.org/*.

`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.

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book 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: "*Machine Learning for Hackers* by Drew Conway and John Myles White (O'Reilly). Copyright 2012 Drew Conway and John Myles White, 978-1-449-30371-6."

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

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

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://shop.oreilly.com/product/0636920018483.do

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*

Acknowledgements

From the Authors

First off, we'd like to thank our editor, Julie Steele, for helping us through the entire process of publishing our first book. We'd also like to thank Melanie Yarbrough and Genevieve d'Entremont for their remarkably thorough work in cleaning the book up for publication. We'd also like to thank the other people at O'Reilly who've helped to improve the book, but whose work was done in the background.

In addition to the kind folks at O'Reilly, we'd like to thank our technical reviewers: Mike Dewar, Max Shron, Matt Canning, Paul Dix, and Maxim Khesin. Their comments improved the book greatly and as the saying goes, the errors that remain are entirely our own responsibility.

We'd also like to thank the members of the NYC Data Brunch for originally inspiring us to write this book and for giving us a place to refine our ideas about teaching machine learning. In particular, thanks to Hilary Mason for originally introducing us to several people at O'Reilly.

Finally, we'd like to thank the many friends of ours in the data science community who've been so supportive and encouraging while we've worked on this book. Knowing that people wanted to read our book helped us keep up pace during the long haul that writing a full-length book entails.

From Drew Conway

I would like to thank Julie Steele, our editor, for appreciating our motivation for this book and giving us the ability to produce it. I would like to thank all of those who provided feedback, both during and after writing; but especially Mike Dewar, Max Shron and Max Khesin. I would like to thank Kristen, my wife, who has always inspired me and was there throughout the entire process with me. Finally, I would like to thank my co-author, John, for having the idea to write a book like this and then the vision to see it to completion.

From John Myles White

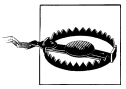
First off, I'd like to thank my co-author, Drew, for writing this book with me. Having someone to collaborate with makes the enormous task of writing an entire book manageable and even fun. In addition, I'd like to thank my parents for having always encouraged me to explore any and every topic that interested me. I'd also like to thank Jennifer Mitchel and Jeffrey Achter for inspiring me to focus on mathematics as an undergraduate. My college years shaped my vision of the world and I'm very grateful for the role you two played in that. As well, I'd like to thank my friend Harek for continually inspiring me to push my limits and to work more. On a less personal note, thanks are due to the band La Dispute for providing the soundtrack to which I've done almost all of the writing of this book. And finally I want to thank the many people who've given me space to work in, whether it's the friends whose couches I've sat on or the owners of the Boutique Hotel Steinerwirt 1493 and the Linger Cafe where I finished the rough and final drafts of this book respectively.

CHAPTER 1
Using R

Machine learning exists at the intersection of traditional mathematics and statistics with software engineering and computer science. In this book, we will describe several tools from traditional statistics that allow you to make sense of the world. Statistics has almost always been concerned with learning something interpretable from data, whereas machine learning has been concerned with turning data into something practical and usable. This contrast makes it easier to understand the term *machine learning*: Machine learning is concerned with teaching *computers* something about the world, so that they can use that knowledge to perform other tasks. In contrast, statistics is more concerned with developing tools for teaching *humans* something about the world, so that they can think more clearly about the world in order to make better decisions.

In machine learning, the *learning* occurs by extracting as much information from the data as possible (or reasonable) through algorithms that parse the basic structure of the data and distinguish the signal from the noise. After they have found the signal, or *pattern*, the algorithms simply decide that everything else that's left over is noise. For that reason, machine learning techniques are also referred to as *pattern recognition algorithms*. We can "train" our machines to learn about how data is generated in a given context, which allows us to use these algorithms to automate many useful tasks. This is where the term *training set* comes from, referring to the set of data used to build a machine learning process. The notion of observing data, learning from it, and then automating some process of recognition is at the heart of machine learning and forms the primary arc of this book. Two particularly important types of patterns constitute the core problems we'll provide you with tools to solve: the problem of classification and the problem of regression, which will be introduced over the course of this book.

In this book, we assume a relatively high degree of knowledge in basic programming techniques and algorithmic paradigms. That said, R remains a relatively niche language, even among experienced programmers. In an effort to establish the same starting point for everyone, this chapter provides some basic information on how to get started using the R language. Later in the chapter we will provide an extended case study for working with data in R.



This chapter does not provide a complete introduction to the R programming language. As you might expect, no such introduction could fit into a single book chapter. Instead, this chapter is meant to prepare the reader for the tasks associated with doing machine learning in R, specifically the process of loading, exploring, cleaning, and analyzing data. There are many excellent resources on R that discuss language fundamentals such as data types, arithmetic concepts, and coding best practices. In so far as those topics are relevant to the case studies presented here, we will touch on all of these issues; however, there will be no explicit discussion of these topics. For those interested in reviewing these topics, many of these resources are listed in Table 1-3.

If you have never seen the R language and its syntax before, we highly recommend going through this introduction to get some exposure. Unlike other high-level scripting languages, such as Python or Ruby, R has a unique and somewhat prickly syntax and tends to have a steeper learning curve than other languages. If you have used R before but not in the context of machine learning, there is still value in taking the time to go through this review before moving on to the case studies.

R for Machine Learning

> R is a language and environment for statistical computing and graphics....R provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.
>
> —The R Project for Statistical Computing, *http://www.r-project.org/*

> The best thing about R is that it was developed by statisticians. The worst thing about R is that...it was developed by statisticians.
>
> —Bo Cowgill, Google, Inc.

R is an extremely powerful language for manipulating and analyzing data. Its meteoric rise in popularity within the data science and machine learning communities has made it the de facto *lingua franca* for analytics. R's success in the data analysis community stems from two factors described in the preceding epitaphs: R provides most of the technical power that statisticians require built into the default language, and R has been supported by a community of statisticians who are also open source devotees.

There are many technical advantages afforded by a language designed specifically for statistical computing. As the description from the R Project notes, the language provides an open source bridge to S, which contains many highly specialized statistical operations as base functions. For example, to perform a basic linear regression in R, one must simply pass the data to the `lm` function, which then returns an object containing detailed information about the regression (coefficients, standard errors, residual

values, etc.). This data can then be visualized by passing the results to the `plot` function, which is designed to visualize the results of this analysis.

In other languages with large scientific computing communities, such as Python, duplicating the functionality of `lm` requires the use of several third-party libraries to represent the data (NumPy), perform the analysis (SciPy), and visualize the results (matplotlib). As we will see in the following chapters, such sophisticated analyses can be performed with a single line of code in R.

In addition, as in other scientific computing environments, the fundamental data type in R is a vector. Vectors can be aggregated and organized in various ways, but at the core, all data is represented this way. This relatively rigid perspective on data structures can be limiting, but is also logical given the application of the language. The most frequently used data structure in R is the *data frame*, which can be thought of as a matrix with attributes, an internally defined "spreadsheet" structure, or relational database-like structure in the core of the language. Fundamentally, a data frame is simply a column-wise aggregation of vectors that R affords specific functionality to, which makes it ideal for working with any manner of data.

For all of its power, R also has its disadvantages. R does not scale well with large data, and although there have been many efforts to address this problem, it remains a serious issue. For the purposes of the case studies we will review, however, this will not be an issue. The data sets we will use are relatively small, and all of the systems we will build are prototypes or proof-of-concept models. This distinction is important because if your intention is to build enterprise-level machine learning systems at the Google or Facebook scale, then R is not the right solution. In fact, companies like Google and Facebook often use R as their "data sandbox" to play with data and experiment with new machine learning methods. If one of those experiments bears fruit, then the engineers will attempt to replicate the functionality designed in R in a more appropriate language, such as C.

This ethos of experimentation has also engendered a great sense of community around the language. The social advantages of R hinge on this large and growing community of experts using and contributing to the language. As Bo Cowgill alludes to, R was borne out of statisticians' desire to have a computing environment that met their specific needs. Many R users, therefore, are experts in their various fields. This includes an extremely diverse set of disciplines, including mathematics, statistics, biology, chemistry, physics, psychology, economics, and political science, to name a few. This community of experts has built a massive collection of packages on top of the extensive base functions in R. At the time of writing, CRAN, the R repository for packages, contained over 2,800 packages. In the case studies that follow, we will use many of the most popular packages, but this will only scratch the surface of what is possible with R.

Finally, although the latter portion of Cowgill's statement may seem a bit menacing, it further highlights the strength of the R community. As we will see, the R language has a particularly odd syntax that is rife with coding "gotchas" that can drive away even experienced developers. But all grammatical grievances with a language can eventually be overcome, especially for persistent hackers. What is more difficult for nonstatisticians is the liberal assumption of familiarity with statistical and mathematical methods built into R functions. Using the `lm` function as an example, if you had never performed a linear regression, you would not know to look for coefficients, standard errors, or residual values in the results. Nor would you know how to interpret those results.

But because the language is open source, you are always able to look at the code of a function to see exactly what it is doing. Part of what we will attempt to accomplish with this book is to explore many of these functions in the context of machine learning, but that exploration will ultimately address only a tiny subset of what you can do in R. Fortunately, the R community is full of people willing to help you understand not only the language, but also the methods implemented in it. Table 1-1 lists some of the best places to start.

Table 1-1. Community resources for R help

Resource	Location	Description
RSeek	*http://rseek.org/*	When the core development team decided to create an open source version of S and call it R, they had not considered how hard it would be to search for documents related to a single-letter language on the Web. This specialized search tool attempts to alleviate this problem by providing a focused portal to R documentation and information.
Official R mailing lists	*http://www.r-project.org/mail.html*	There are several mailing lists dedicated to the R language, including announcements, packages, development, and—of course—help. Many of the language's core developers frequent these lists, and responses are often quick and terse.
StackOverflow	*http://stackoverflow.com/questions/tagged/r*	Hackers will know StackOverflow.com as one of the premier web resources for coding tips in any language, and the R tag is no exception. Thanks to the efforts of several prominent R community members, there is an active and vibrant collection of experts adding and answering R questions on StackOverflow.
#rstats Twitter hashtag	*http://search.twitter.com/search?q=%23rstats*	There is also a very active community of R users on Twitter, and they have designated the #rstats hash tag as their signifier. The thread is a great place to find links to useful resources, find experts in the language, and post questions—as long as they can fit into 140 characters!
R-Bloggers	*http://www.r-bloggers.com/*	There are hundreds of people blogging about how they use R in their research, work, or just for fun. R-bloggers.com aggregates these blogs and provides a single source for all things related to R in the blogosphere, and it is a great place to learn by example.
Video Rchive	*http://www.vcasmo.com/user/drewconway*	As the R community grows, so too do the number of regional meetups and gatherings related to the language. The Rchive attempts to document the presentations and tutorials given at these meetings by posting videos and slides, and now contains presentations from community members all over the world.

The remainder of this chapter focuses on getting you set up with R and using it. This includes downloading and installing R, as well as installing R packages. We conclude with a miniature case study that will serve as an introduction to some of the R idioms we'll use in later chapters. This includes issues of loading, cleaning, organizing, and analyzing data.

Downloading and Installing R

Like many open source projects, R is distributed by a series of regional mirrors. If you do not have R already installed on your machine, the first step is to download it. Go to *http://cran.r-project.org/mirrors.html* and select the CRAN mirror closest to you. Once you have selected a mirror, you will need to download the appropriate distribution of R for whichever operating system you are running.

R relies on several legacy libraries compiled from C and Fortran. As such, depending on your operating system and your familiarity with installing software from source code, you may choose to install R from either a compiled binary distribution or the source. Next, we present instructions for installing R on Windows, Mac OS X, and Linux distributions, with notes on installing from either source or binaries when available.

Finally, R is available in both 32- and 64-bit versions. Depending on your hardware and operating system combination, you should install the appropriate version.

Windows

For Windows operating systems, there are two subdirectories available to install R: *base* and *contrib*. The latter is a directory of compiled Windows binary versions of all of the contributed R packages in CRAN, whereas the former is the basic installation. Select the *base* installation, and download the latest compiled binary. Installing contributed packages is easy to do from R itself and is not language-specific; therefore, it is not necessary to to install anything from the *contrib* directory. Follow the on-screen instructions for the installation.

Once the installation has successfully completed, you will have an R application in your Start menu, which will open the RGui and R Console, as pictured in Figure 1-1.

For most standard Windows installations, this process should proceed without any issues. If you have a customized installation or encounter errors during the installation, consult the *R for Windows FAQ* at your mirror of choice.

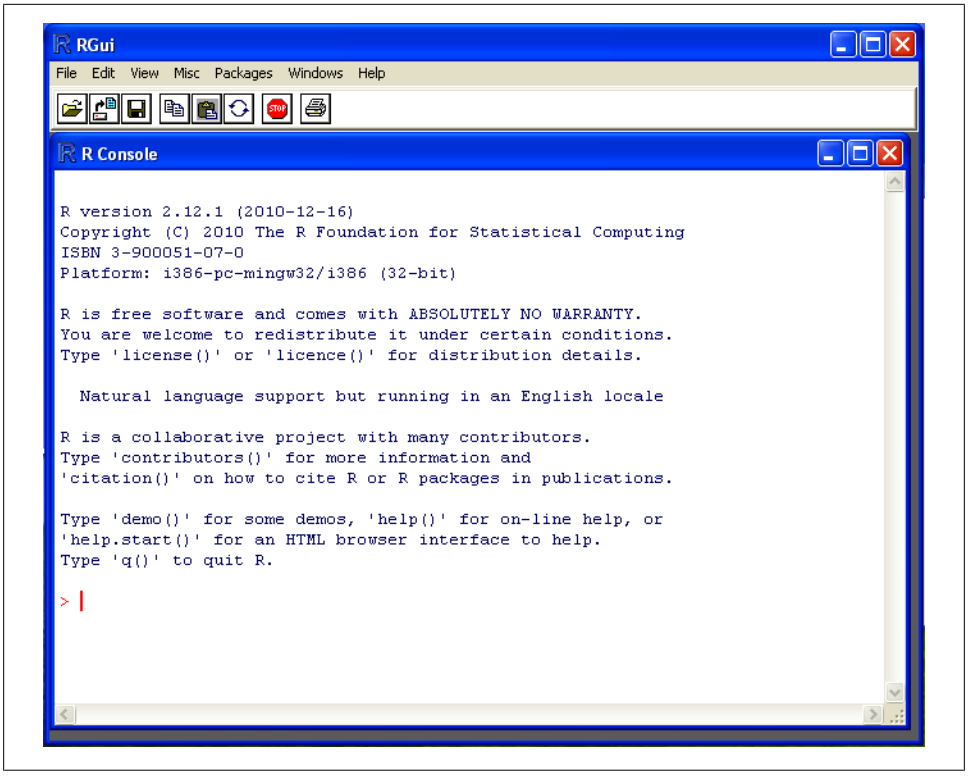


Figure 1-1. The RGui and R Console on a Windows installation

Mac OS X

Fortunately for Mac OS X users, R comes preinstalled with the operating system. You can check this by opening *Terminal.app* and simply typing **R** at the command line. You are now ready to begin! For some users, however, it will be useful to have a GUI application to interact with the R Console. For this you will need to install separate software. With Mac OS X, you have the option of installing from either a compiled binary or the source. To install from a binary—recommended for users with no experience using a Linux command line—simply download the latest version at your mirror of choice at *http://cran.r-project.org/mirrors.html*, and follow the on-screen instructions. Once the installation is complete, you will have both *R.app* (32-bit) and *R64.app* (64-bit) available in your *Applications* folder. Depending on your version of Mac OS X and your machine's hardware, you may choose which version you wish to work with.

As with the Windows installation, if you are installing from a binary, this process should proceed without any problems. When you open your new R application, you will see a console similar to the one pictured in Figure 1-2.

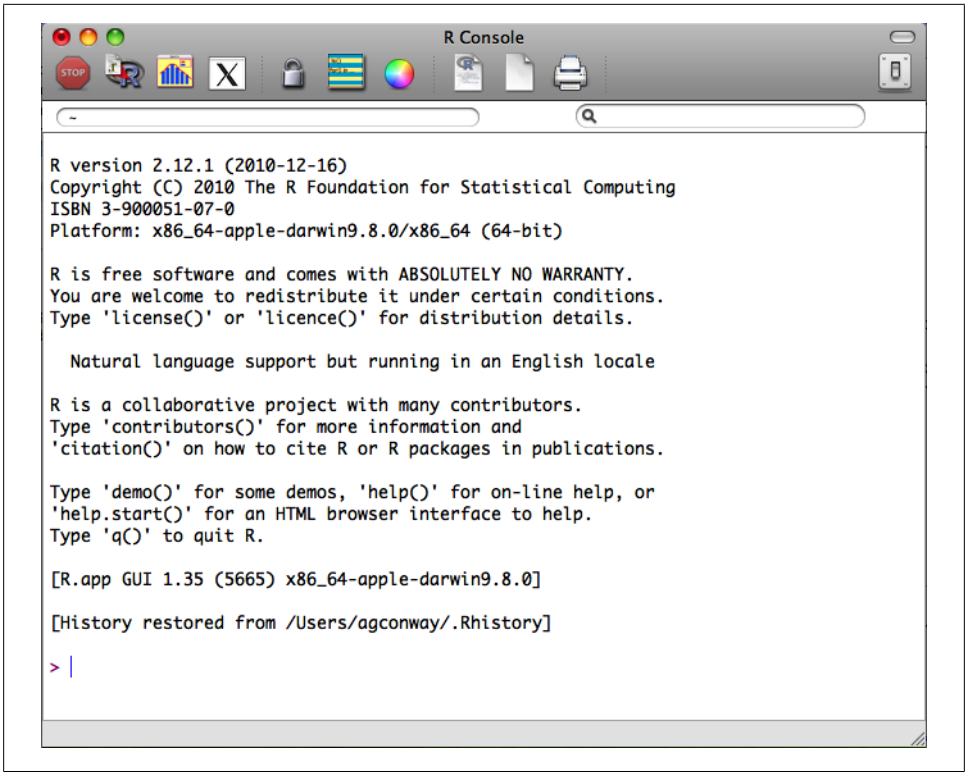


Figure 1-2. The R Console on a 64-bit version of the Mac OS X installation

If you have a custom installation of Mac OS X or wish to customize the installation of R for your particular configuration, we recommend that you install from the source code. Installing R from source on Mac OS X requires both the C and Fortran compilers, which are not included in the standard installation of the operating system. You can install these compilers using the Mac OS X Developers Tools DVD included with your original Mac OS X installation package, or you can install the necessary compilers from the *tools* directory at the mirror of your choice.

Once you have all of the necessary compilers to install from source, the process is the typical configure, make, and install procedure used to install most software at the command line. Using *Terminal.app*, navigate to the folder with the source code and execute the following commands:

```
$ ./configure
$ make
$ make install
```

Depending on your permission settings, you may have to invoke the `sudo` command as a prefix to the configuration step and provide your system password. If you encounter any errors during the installation, using either the compiled binary distribution or the source code, consult the *R for Mac OS X FAQ* at the mirror of your choice.

Linux

As with Mac OS X, R comes preinstalled on many Linux distributions. Simply type `R` at the command line, and the R console will be loaded. You can now begin programming! The CRAN mirror also includes installations specific to several Linux distributions, with instructions for installing R on Debian, RedHat, SUSE, and Ubuntu. If you use one of these installations, we recommend that you consult the instructions for your operating system because there is considerable variance in the best practices among Linux distributions.

IDEs and Text Editors

R is a scripting language, and therefore the majority of the work done in the case studies that follow will be done within an IDE or text editor, rather than directly inputted into the R console. As we show in the next section, some tasks are well suited for the console, such as package installation, but primarily you will want to work within the IDE or text editor of your choice.

For those running the GUI in either Windows or Mac OS X, there is a basic text editor available from that application. By either navigating to *File→New Document* from the menu bar or clicking on the blank document icon in the header of the window (highlighted in Figure 1-3), you will open a blank document in the text editor. As a hacker, you likely already have an IDE or text editor of choice, and we recommend that you use whichever environment you are most comfortable in for the case studies. There are simply too many options to enumerate here, and we have no intention of inserting ourselves in the infamous Emacs versus Vim debate.

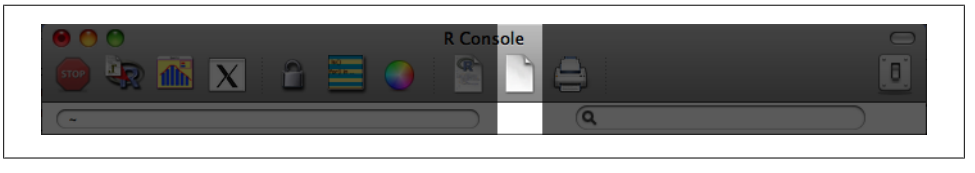


Figure 1-3. Text editor icon in R GUI

Loading and Installing R Packages

There are many well-designed, -maintained, and -supported R packages related to machine learning. With respect to the case studies we will describe, there are packages for dealing with spatial data, text analysis, network structures, and interacting with web-based APIs, among many others. As such, we will be relying heavily on the functionality built into several of these packages.

Loading packages in R is very straightforward. There are two functions to perform this: `library` and `require`. There are some subtle differences between the two, but for the purposes of this book, the primary difference is that `require` will return a Boolean (`TRUE` or `FALSE`) value, indicating whether the package is installed on the machine after attempting to load it. For example, in Chapter 6 we will use the `tm` package to tokenize text. To load these packages, we can use either the `library` or `require` functions. In the following example, we use `library` to load `tm` but use `require` for `XML`. By using the `print` function, we can see that we have `XML` installed because a Boolean value of `TRUE` was returned after the package was loaded:

```
library(tm)
print(require(XML))
#[1] TRUE
```

If we did not have `XML` installed—i.e., if `require` returned `FALSE`—then we would need to install that package before proceeding.

If you are working with a fresh installation of R, then you will have to install a number of packages to complete all of the case studies in this book.

There are two ways to install packages in R: either with the GUI or with the `install.packages` function from the console. Given the intended audience for this book, we will be interacting with R exclusively from the console during the case studies, but it is worth pointing out how to use the GUI to install packages. From the menu bar in the application, navigate to Packages & Data→Package Installer, and a window will appear, as displayed in Figure 1-4. From the Package Repository drop-down, select either "CRAN (binaries)"or "CRAN (sources)", and click the Get List button to load all of the packages available for installation. The most recent version of packages will be available in the "CRAN (sources)" repository, and if you have the necessary compilers installed on your machine, we recommend using this sources repository. You can now select the package you wish to install and click Install Selected to install the packages.

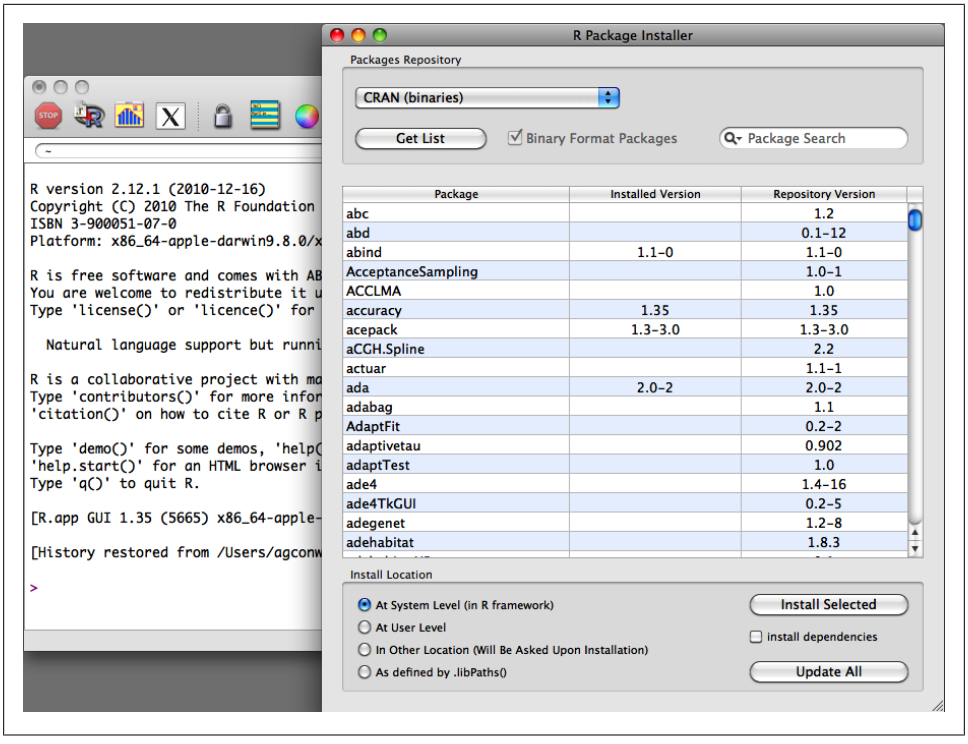


Figure 1-4. Installing R packages using the GUI

The `install.packages` function is the preferred way to install packages because it provides greater flexibility in how and where packages get installed. One of the primary advantages of using `install.packages` is that it allows you to install from local source code as well as from CRAN. Though uncommon, occasionally you may want to install a package that is not yet available on CRAN—for example, if you're updating to an experimental version of a package. In these cases you will need to install from source:

```
install.packages("tm", dependencies=TRUE)
setwd("~/Downloads/")
install.packages("RCurl_1.5-0.tar.gz", repos=NULL, type="source")
```

In the first example, we use the default settings to install the `tm` package from CRAN. The `tm` provides a function used to do text mining, and we will use it in Chapter 3 to perform classifications on email text. One useful parameter in the `install.packages` function is `suggests`, which by default is set to `FALSE`, but if activated will instruct the function to download and install any secondary packages used by the primary installation. As a best practice, we recommend always setting this to `TRUE`, especially if you are working with a clean installation of R.

Alternatively, we can also install directly from compressed source files. In the previous example, we installed the `RCurl` package from the source code available on the author's

website. Using the `setwd` function to make sure the R working directory is set to the directory where the source file has been saved, we can simply execute the command shown earlier to install directly from the source code. Note the two parameters that have been altered in this case. First, we must tell the function not to use one of the CRAN repositories by setting `repos=NULL`, and we also specify the type of installation using `type="source"`.

Table 1-2. R packages used in Machine Learning for Hackers

Name	Location	Author(s)	Description and use
`arm`	*http://cran.r-project.org/web/packages/arm/*	Andrew Gelman, et al.	Package for doing multilevel/hierarchical regression models.
`ggplot2`	*http://cran.r-project.org/web/packages/glmnet/index.html*	Hadley Wickham	An implementation of the grammar of graphics in R. The premier package for creating high-quality graphics.
`glmnet`	*http://had.co.nz/ggplot2/*	Jerome Friedman, Trevor Hastie, and Rob Tibshirani	Lasso and elastic-net regularized generalized linear models.
`igraph`	*http://igraph.sourceforge.net/*	Gabor Csardi	Routines for simple graphs and network analysis. Used for representing social networks.
`lme4`	*http://cran.r-project.org/web/packages/lme4/*	Douglas Bates, Martin Maechler, and Ben Bolker	Provides functions for creating linear and generalized mixed-effects models.
`lubridate`	*https://github.com/hadley/lubridate*	Hadley Wickham	Provides convenience function to making working with dates in R easier.
`RCurl`	*http://www.omegahat.org/RCurl/*	Duncan Temple Lang	Provides an R interface to the `libcurl` library for interacting with the HTTP protocol. Used to import raw data from the Web.
`reshape`	*http://had.co.nz/plyr/*	Hadley Wickham	A set of tools used to manipulate, aggregate, and manage data in R.
`RJSONIO`	*http://www.omegahat.org/RJSONIO/*	Duncan Temple Lang	Provides functions for reading and writing JavaScript Object Notation (JSON). Used to parse data from web-based APIs.
`tm`	*http://www.spatstat.org/spatstat/*	Ingo Feinerer	A collection of functions for performing text mining in R. Used to work with unstructured text data.
`XML`	*http://www.omegahat.org/RSXML/*	Duncan Temple Lang	Provides the facility to parse XML and HTML documents. Used to extract structured data from the Web.

As mentioned, we will use several packages through the course of this book. Table 1-2 lists all of the packages used in the case studies and includes a brief description of their purpose, along with a link to additional information about each. Given the number of prerequisite packages, to expedite the installation process we have created a short script that will check whether each required package is installed and, if